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

June 2013

Abstract

Preface

Contents

Chapter 1

Introduction

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, it allows the calculation of physical properties also at microscopic and up to subatomic length scales. 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.

This thesis is based on the theories and methods of quantum physics, too, and it it therefore necessary to explain the main and general concepts forming the basis of topics more closely related to this thesis. This chapter deals with those basic concepts and especially introduce 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.

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 century and particularly in the beginning of the 20th century, a number of experiments cast doubts on the 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 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.

In the following sections, we will discuss the basic features of quantum mechanics using the standard formalisms used today. Unless explicit references are stated, we base our explanations, as in this section on [1–4].

2.2 Hilbert space and Dirac notation

All physical states we will consider, lie in a complex vector space, which we refer to as *Hilbert space* $\mathcal{H}$, named after David Hilbert [5]. 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 will be discussed in detail in section 2.4 and describe our quantum mechanical states, we use the *bra-ket* notation developed by the physicist Paul Dirac [6]. 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 column vector,

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

and its dual bra-state as Hermitian transpose

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

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

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

Let us at this stage summarize some definitions which we will frequently use 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,

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

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

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

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*, which means

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

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

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

which can easily be proved by applying definition (2.13). This list is not complete and summarizes just those properties most relevant for this thesis. For more properties, we refer to [1].
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 will shortly summarize following [2], which we also refer to for the corresponding proofs.[3]

- All eigenvalues of Hermitian operators are real.
- Eigenfunctions belonging to distinct eigenvalues are orthonormal.

[2]For derivation, we refer to [2], pp.42-43.

[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.

- 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 can not 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 will look in more detail on how quantum mechanical systems can be represented by functions Ψ, referred to as *wavefunctions*, 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 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, dividing by $\int_{\mathcal{H}} d\mathbf{r}\ |\Psi(\mathbf{r}, t,)|^2$.

2.4.2 Time-dependent Schrödinger's equation

To get a concrete expression for the wave function, let us first consider the easy case that our system consists of only 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 expression 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 V acts as an operator, possibly containing derivatives etc.
Equation (2.21) is the *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 dimension, it reads

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

The Schrödinger equation can be simplified to

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t) \tag{2.23}$$

by defining the Hamiltonian operator,

$$\hat{H} = \hat{T} + \hat{V}. \tag{2.24}$$

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

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

and $\hat{V}$ as before the operator of the potential energy. Hence our Hamiltonian represents the total energy of this particle and, for systems of more than just one particle, can be extended to correspond to the total energy of the system, 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 with separation of variables, and we make 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 of such 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 solution, too.

For each of the solutions, Schrödinger's equation now implies

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

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

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

Note that we drop the t- and $\mathbf{r}$-dependence for better readability. We observe that now the left side is only a function of time t, whereas the right side is only a function of space $\mathbf{r}$. This can only hold true if both expressions equal a constant, which we denote 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. (2.24). 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 (2.23) is now solved by the wave function

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

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 [1].

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

$$\hat{A}\psi_i = a_i\psi_i,$$

meaning that the operator $\hat{A}$ has different eigenvalues with corresponding eigenfunctions. 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 the wave function $\Psi(\mathbf{r}, t)$ is given by the *time-dependent Schrödinger equation*

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

2.6 Special case: Harmonic oscillator

One quantum-mechanical system of highest interest is the harmonic oscillator. Not only is it easily analytically solvable and allows to demonstrate the concepts of the previous sections, but many more complex problems can be reduced to the harmonic oscillator and get thereby exactly solvable.

Serving as basis for the Hamiltonian, it will have 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,

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

For the quantum-mechanical analogue, we use Eqs. (2.24) and (2.25) combined with the oscillator potential (2.37), where we replace x with the corresponding operator $\hat{x}$, and obtain for one dimension

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

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

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

and can be solved in different ways.

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,

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

which simplifies 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 behave as

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

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 $x \to \infty$.
We we make the ansatz $\psi_n(x) = H(x)e^{-x^2/2}$ and get the following differential equation for $H(x)$,

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

For solving this equation, we use Fuchs' ansatz

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

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

$$H''(x) - 2xH'(x) + 2nH(x).$$

This equation is solved by the Hermite polynomials $H_n(x)$, fulfilling 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

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

That way, we have found our solution ψ_n expanded in Hermite polynomials,

$$\psi_n(x) = N_n H_n(x)e^{-x^2/2}. \tag{2.46}$$

with corresponding eigenvalues $\lambda_n = 2n + 1$, suggesting

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right), \tag{2.47}$$

[4]See [3] for mathematical details.

and normalization factors

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

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 shown here represents the standard approach, known to yield the correct solution. However, in addition, there exists a more elegant way, which is also of conceptual importance.

2.6.2 Elegant solution with ladder operators

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

$$a^\dagger = \sqrt{\frac{m\omega}{2\hbar}}\left(\hat{x} - \frac{i\hat{p}}{m\omega}\right) \tag{2.48}$$

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

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

Moreover, we define the number operator

$$\hat{N} = a^\dagger a, \qquad \hat{N}|\psi\rangle = n|\psi\rangle, \tag{2.50}$$

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

$$[a, a^\dagger] = 1, \qquad [\hat{N}, a^\dagger] = a^\dagger, \qquad [\hat{N}, a] = -a. \tag{2.51}$$

The reversion of the latter operators yields

$$\hat{x} = \sqrt{\frac{\hbar}{2m\omega}}(a + a^\dagger), \tag{2.52}$$

$$\hat{p} = i\sqrt{\frac{\hbar m\omega}{2}}(a^\dagger - a). \tag{2.53}$$

Inserting this into our Hamiltonian (2.38), we get

$$\hat{H} = \hbar\omega\left(a^\dagger a + \frac{1}{2}\right) = \hbar\omega\left(\hat{N} + \frac{1}{2}\right). \tag{2.54}$$

[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 already here.

[6]For the more or less straightforward proofs in this section, we refer to [2].

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. (2.47), $E_n = \hbar\omega\left(n+\frac{1}{2}\right)$. 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:

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

where we make use of the relations in Eq. (2.51). Hence the operators $a^\dagger$ and a increase/decrease the eigenvalue n of a eigenstate $|\psi_n\rangle$ by 1, which explains the terms *creation* and *annihilation* operator, respectively.
To stop this iteration, one defines for the lowest value $n=0$

$$a|\psi_0\rangle = 0. \tag{2.55}$$

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

$$|\psi_n\rangle = \frac{(\hat{a}^\dagger)^n}{\sqrt{n!}}|\psi_0\rangle,$$

and we get the lowest-lying state by solving Eq. (2.55) explicitly:

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

With the normalization constant N specified, we obtain

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

For a coordinate representation of our wave functions, we form the inner product

$$\langle x|\psi_n\rangle = \left(\sqrt{\frac{m\omega}{\pi\hbar}}\frac{1}{2^n n!}\right)^{1/2} H_n\left(\sqrt{\frac{m\omega}{\hbar x}}\right)e^{-\frac{m\omega x^2}{2\hbar}}, \tag{2.56}$$

and get the same result as the conventional approach yielded in Eq. (2.46), provided that we rewrite it 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 two or three 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

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + \frac{1}{2}m\omega^2 r^2, \tag{2.57}$$

which for $d = 2$ dimensions explicitly reads

$$\hat{H} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + \frac{1}{2}m\omega^2(x^2 + y^2), \tag{2.58}$$

and for $d = 3$ dimensions

$$\hat{H} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) + \frac{1}{2}m\omega^2(x^2 + y^2 + z^2). \tag{2.59}$$

For the example $d = 2$, 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

$$\hat{H}|n\rangle = \left(\hat{H}_x|n_x\rangle\right) \otimes |n_y\rangle + |n_x\rangle \otimes \left(\hat{H}_y|n_y\rangle\right) \overset{!}{=} E_n\left(|n_x\rangle \otimes |n_y\rangle\right). \tag{2.60}$$

Inserting the well-known solution in one dimension, $E_{n_i} = \hbar\omega\left(n_i + \frac{1}{2}\right)$ for $i \in \{x, y\}$, the total energy is

$$\begin{aligned}E_n(n_x, n_y) &= \hbar\omega\left(n_x + \frac{1}{2}\right) + \hbar\omega\left(n_y + \frac{1}{2}\right) \\ &= \hbar\omega\left(n_x + n_y + 1\right). \end{aligned}\tag{2.61}$$

In other words, one simply adds the eigenvalues of each dimension. For $d = 3$ dimensions, an analogue derivation yields

$$\begin{aligned}E_n(n_x, n_y, n_z) &= \hbar\omega\left(n_x + \frac{1}{2}\right) + \hbar\omega\left(n_y + \frac{1}{2}\right) + \hbar\omega\left(n_z + \frac{1}{2}\right) \\ &= \hbar\omega\left(n_x + n_y + n_z + \frac{3}{2}\right). \end{aligned}\tag{2.62}$$

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, and the many-body Schrödinger equation includes more terms than a pure addition 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 will follow the explanations in [7, 8].

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.(2.23). 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 a 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 $\mathcal{H}_N$ are direct products of the corresponding single-particle basis states:

$$\Phi_N(\mathbf{R}, t) = \phi_1(\mathbf{r}_1, t) \otimes \phi_2(\mathbf{r}_2, t) \otimes \cdots \otimes \phi_N(\mathbf{r}_N, t), \tag{3.2}$$

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

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

With the single-particle functions ϕ_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.(2.23) has to be extended to include the contributions from all particles. A first approach is to start with non-interacting particles, 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 is 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 [9]. For a N-particle function, the entries of this determinant are N single-particle functions $\phi_\alpha, \phi_\beta, \cdots, \phi_\delta$, 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

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

where the factor $1/\sqrt{N!}$ accounts for the indistinguishability of the particles, ensuring 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 in addition to the 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 $\phi_i(\mathbf{r}_j)$ is strictly speaking composed of two parts, namely a spatial and a spin part:

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

where σ_j denotes the spin orientation of particle j.

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

$$\Psi(\mathbf{R}) = \sum_{\alpha,\beta\cdots\delta} C_{\alpha,\beta\cdots\delta}\Phi_{\alpha,\beta,\cdots,\delta}(\mathbf{r}_1,\cdots,\mathbf{r}_N), \tag{3.10}$$

where the expansion coefficients $C_{\alpha,\beta\cdots\delta}$ have the same meaning as in Eq. (3.3) and the Slater determinants $\Phi_{\alpha,\beta,\cdots,\delta}(\mathbf{r}_1,\cdots,\mathbf{r}_N)$ are defined by Eq. (3.8). Such an expansion is possible for all times t, such that we will suppress the t-dependence in the following.

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 how they can be applied to treat 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_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_0 to the Slater determinant, creating a $(N+1)$- from a N-particle state. On the other hand, in Eq. (3.13), the annihilation operator a_{α_1} removes a state ϕ_1, thus transforming a N- to a $(N-1)$-particle state.
We define the vacuum state $|0\rangle$ as state where none of the orbitals are occupied, specified by the relation

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

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 and 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 larger 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 $i, j, k, \ldots$, and to the ones above the Fermi level as *particle states*, labelled with $a, b, c, \ldots$ General single-particle states that can lie above or below the Fermi level, will be labelled with $p, q, r, \ldots$

With this notation, the N-particle reference state is obtained by applying all N creation operators corresponding to hole states on the vacuum state,

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

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

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

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

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

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

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

is referred to as 3p-2h excitation.

[1]For a proof of the relations, we refer to [7].

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, applying 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 increased, 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,

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

Moreover, we define the *contraction* between two operators as

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

[2]We start with normal-ordering *with respect to the vacuum state*, as originally used by Wick [10].

The four possible contractions are

$$\begin{aligned}
\wick{\c a^\dagger_p \c a^\dagger_q} &= a^\dagger_p a^\dagger_q - a^\dagger_p a^\dagger_q = 0,\\
\wick{\c a_p \c a_q} &= a_p a_q - a_p a_q = 0,\\
\wick{\c a^\dagger_p \c a_q} &= a^\dagger_p a_q - a^\dagger_p a_q = 0,\\
\wick{\c a_p \c a^\dagger_q} &= a_p a^\dagger_q - (-a^\dagger_q a_p) = \{a_p, a^\dagger_q\} = \delta_{pq}.
\end{aligned} \tag{3.23}$$

The concept of normal-ordering can be extended from the vacuum state $|0\rangle$ to the reference state $|\Phi_0\rangle$. In this case, the normal-ordered product $\left\{\hat{A}\hat{B}\hat{C}\dots\right\}$ 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, we get again the useful property that the expectation value with respect to the reference state vanishes,

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

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

$$\begin{aligned}
\wick{\c a^\dagger_i \c a_j} &= a^\dagger_i a_j - (-a_j a^\dagger_i) = \delta_{ij},\\
\wick{\c a_a \c a^\dagger_b} &= a_a a^\dagger_b - (-a^\dagger_b a_a) = \delta_{ab}.
\end{aligned} \tag{3.25}$$

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

Statement of the theorem Wick's theorem [10] states that any product of creation and annihilation operators can be expressed as their normal-ordered product plus the sum of all possible normal products with contractions. Symbolically, this means

$$\begin{aligned}
\hat{A}\hat{B}\hat{C}\hat{D}\hat{E}\hat{F}\dots = &\left\{\hat{A}\hat{B}\hat{C}\hat{D}\hat{E}\hat{F}\dots\right\}\\
&+\left\{\wick{\c{\hat{A}}\c{\hat{B}}}\hat{C}\hat{D}\hat{E}\hat{F}\dots\right\}+\left\{\wick{\c{\hat{A}}\hat{B}\c{\hat{C}}}\hat{D}\hat{E}\hat{F}\dots\right\}+\dots\\
&+\left\{\wick{\c1{\hat{A}}\c2{\hat{B}}\c1{\hat{C}}\c2{\hat{D}}}\hat{E}\hat{F}\dots\right\}+\left\{\hat{A}\wick{\c1{\hat{B}}\c2{\hat{C}}\c1{\hat{D}}\c2{\hat{E}}}\hat{F}\right\}+\dots\\
&+\dots
\end{aligned} \tag{3.26}$$

$$+\left\{\wick{\c1{\hat{A}}\c2{\hat{B}}\c3{\hat{C}}\c1{\hat{D}}\c2{\hat{E}}\c3{\hat{F}}}\dots\right\}+\dots \tag{3.27}$$

In words, the first term is the normal-ordered string, followed by all possible normal products with contractions between two operators. Afterwards, we have 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, where all operators are contracted, can give a non-zero contribution. Regarding the previous example, this means

$$\begin{aligned}\langle pq|rs\rangle &= \langle 0|\left(a_p a_q a_r^\dagger a_s^\dagger\right)|0\rangle \\ &= \langle 0|\left(\wick{\c1 a_p \c2 a_q \c2 a_r^\dagger \c1 a_s^\dagger} + \wick{\c1 a_p \c2 a_q \c1 a_r^\dagger \c2 a_s^\dagger}\right)|0\rangle \\ &= \delta_{ps}\delta_{qr} - \delta_{pr}\delta_{qs},\end{aligned}$$

which is the same result as before, obtained with much less effort, and demonstrates 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 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

$$\hat{N} = \sum_p a_p^\dagger a_p. \tag{3.28}$$

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

$$\begin{aligned}\hat{N}|\alpha_1\alpha_2\ldots\alpha_N\rangle &= \sum_p a_p^\dagger a_p\;|\alpha_1\alpha_2\ldots\alpha_N\rangle \\ &= a_1^\dagger|\alpha_2\alpha_3\ldots\alpha_N\rangle + a_2^\dagger|\alpha_1\alpha_3\ldots\alpha_N\rangle + \ldots \\ &= |\alpha_1\alpha_2\ldots\alpha_N\rangle + |\alpha_1\alpha_2\ldots\alpha_N\rangle + \ldots \\ &= N|\alpha_1\alpha_2\ldots\alpha_N\rangle. \end{aligned}\tag{3.29}$$

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 (3.29), 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 to express the more complex operators of the Hamiltonian in a similar manner:

Restricted to maximally two-body interactions, our 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]

$$\hat{H}_0 = \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q. \tag{3.30}$$

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

$$\langle p|\hat{h}^{(0)}|q\rangle = \int d\mathbf{r}\ \phi_p^*(\mathbf{r})\hat{h}^{(0)}\phi_q(\mathbf{r}), \tag{3.31}$$

representing the probability that the operator moves a particle from single-particle state $|q\rangle$ to $|p\rangle$.

For the two-body operator of the Hamiltonian, we have analogously

$$\hat{V} = \frac{1}{2}\sum_{pqrs}\langle pq|\hat{v}|rs\rangle a_p^\dagger a_q^\dagger a_s a_r, \tag{3.32}$$

$$\langle pq|\hat{v}|rs\rangle = \int\int d\mathbf{r}_1 d\mathbf{r}_2\ \phi_p^*(\mathbf{r}_1)\phi_q^*(\mathbf{r}_2)\hat{v}(\mathbf{r}_1,\mathbf{r}_2)\phi_r(\mathbf{r}_1)\phi_s(\mathbf{r}_2). \tag{3.33}$$

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 (3.33), one does not account for the antisymmetry of $|rs\rangle$, as stated in Eq. (3.15). This relation would suggest

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

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 $|q\rangle$, and 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 uses so-called *antisymmetric elements*, defined as

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

With these elements, the two-body interaction operator reads

$$\hat{V} = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r, \tag{3.36}$$

and the full Hamiltonian

$$\begin{aligned}\hat{H} &= \hat{H}_0 + \hat{V} \\ &= \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r.\end{aligned} \tag{3.37}$$

[3]For a profound motivation of this representation, see [7].

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 is applied:

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

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 denoted as transition from state $|k\rangle$ to $|j\rangle$, this means

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

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

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

where 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

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

and again, we apply Wick's theorem to get

$$\begin{aligned}\langle i|\hat{H}_0|j\rangle &= \langle 0|a_i \sum_{pq} \langle p|\hat{h}^{(0)}q\rangle a_p^\dagger a_q a_j^\dagger|0\rangle \\ &= \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle \langle 0|\wick{\c1 a_i \c1 a_p^\dagger \c2 a_q \c2 a_j^\dagger}|0\rangle \\ &= \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle \delta_{ip}\delta_{qj} = \langle i|\hat{h}^{(0)}|j\rangle.\end{aligned}$$

In the case that both determinants have the same occupation 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 will work with a Hamiltonian that is restricted to two-body interactions, the interaction part can in principle contain higher-order interactions, too, and is straightforwardly extended to

$$\hat{H}_I = \frac{1}{2!}\sum_{pqrs} \langle pq|\hat{v}|rs\rangle + \frac{1}{3!}\sum_{pqrstu} \langle pqr|\hat{v}^{(3)}|stu\rangle + \cdots + \frac{1}{N!}\sum_{\substack{pqr\dots \\ stu\dots}} \langle pqr\cdots|\hat{v}^{(N)}|\cdots stu\rangle, \quad (3.38)$$

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

In-medium formulation of $\hat{H}$

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

$$\begin{aligned}\hat{H} = \sum_{pq} \langle p|\hat{h}^{(0)}|q\rangle a_p^\dagger a_q + \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r \\ + \frac{1}{36}\sum_{pqrstu} \langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\, a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s + \dots,\end{aligned} \tag{3.39}$$

we use Wick's theorem 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 is rewritten the following way:

$$a_p^\dagger a_q = \{a_p^\dagger a_q\} + \wick{\c a_p^\dagger \c a_q} = \{a_p^\dagger a_q\} + \delta_{pq<F}. \tag{3.40}$$

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

$$\hat{H}_N^{(0)} = \sum_{pq}\langle p|\hat{h}^{(0)}|q\rangle\{a_p^\dagger a_q\} + \sum_i \langle i|\hat{h}^{(0)}|i\rangle, \tag{3.41}$$

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

$$\begin{aligned} a_p^\dagger a_q^\dagger a_s a_r &= \{a_p^\dagger a_q^\dagger a_s a_r\} + \left\{\wick{\c a_p^\dagger a_q^\dagger \c a_s a_r}\right\} + \left\{\wick{\c a_p^\dagger a_q^\dagger a_s \c a_r}\right\} + \left\{\wick{a_p^\dagger \c a_q^\dagger \c a_s a_r}\right\} \\ &+ \left\{\wick{a_p^\dagger \c a_q^\dagger a_s \c a_r}\right\} + \left\{\wick{\c1 a_p^\dagger \c2 a_q^\dagger \c1 a_s \c2 a_r}\right\} + \left\{\wick{\c1 a_p^\dagger \c2 a_q^\dagger \c2 a_s \c1 a_r}\right\} \\ &= \{a_p^\dagger a_q^\dagger a_s a_r\} - \delta_{ps<F}\{a_q^\dagger a_r\} + \delta_{pr<F}\{a_q^\dagger a_s\} + \delta_{qs<F}\{a_p^\dagger a_r\} \\ &- \delta_{qr<F}\{a_p^\dagger a_s\} - \delta_{ps<F}\delta_{qr<F} + \delta_{pr<F}\delta_{qs<F}. \end{aligned} \tag{3.42}$$

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

$$\hat{V} = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle a_p^\dagger a_q^\dagger a_s a_r \tag{3.43}$$

$$\begin{aligned} &= \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\} - \frac{1}{4}\sum_{qri}\langle iq||ri\rangle\{a_q^\dagger a_r\} + \frac{1}{4}\sum_{qsi}\langle iq||is\rangle \\ &+ \frac{1}{4}\sum_{pri}\langle pi||ri\rangle\{a_p^\dagger a_r\} - \frac{1}{4}\sum_{psi}\langle pi||is\rangle\{a_p^\dagger a_s\} - \frac{1}{4}\sum_{ij}\langle ij||ji\rangle + \frac{1}{4}\sum_{ij}\langle ij||ij\rangle. \end{aligned} \tag{3.44}$$

With the antisymmetry relation (3.15) suggesting that

$$\langle pq||rs\rangle = -\langle qp||rs\rangle = -\langle pq||sr\rangle = \langle qp||sr\rangle, \tag{3.45}$$

expression (3.44) can be simplified to

$$\hat{V} = \frac{1}{4}\sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\} + \sum_{pqi}\langle pi||qi\rangle\{a_p^\dagger a_q\} + \frac{1}{2}\sum_{ij}\langle ij||ij\rangle. \tag{3.46}$$

For higher-order interactions, one proceeds analogously, and for the three-body operator we get

$$\begin{aligned}\hat{V}^{(3)} &= \frac{1}{36}\sum_{pqrstu} \langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\, a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s \\ &= \frac{1}{36}\sum_{pqrstu} \langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\, \{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\} + \frac{1}{4}\sum_{i} \langle pqi|\,\hat{v}^{(3)}\,|rsi\rangle\, \{a_p^\dagger a_q^\dagger a_s a_r\} \\ &\quad + \frac{1}{2}\sum_{ij} \langle pij|\,\hat{v}^{(3)}\,|pij\rangle\, \{a_p^\dagger a_q\} + \frac{1}{6}\sum_{ijk} \langle ijk|\,\hat{v}^{(3)}\,|ijk\rangle\,. \end{aligned} \tag{3.47}$$

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

$$\begin{aligned}\hat{H}_N &= \hat{F}_N + \hat{V}_N + \hat{H}_N^{(3)} \\ &= \sum_{pq} f_{pq}\{a_p^\dagger a_q\} + \frac{1}{4}\sum_{pqrs} \Gamma_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\} + \frac{1}{36}\sum_{pqrstu} W_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\},\end{aligned} \tag{3.48}$$

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

$$f_{pq} = \langle p|\,\hat{h}^{(0)}\,|q\rangle + \sum_i \langle pi|\,\hat{v}\,|qi\rangle + \frac{1}{2}\sum_{ij} \langle pij|\,\hat{v}^{(3)}\,|pij\rangle\,, \tag{3.49}$$

$$\Gamma_{pqrs} = \langle pq|\,\hat{v}\,|rs\rangle + \sum_i \langle pqi|\,\hat{v}^{(3)}\,|rsi\rangle\,, \tag{3.50}$$

$$W_{pqrstu} = \langle pqr|\,\hat{v}^{(3)}\,|stu\rangle\,, \tag{3.51}$$

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

$$\begin{aligned}\hat{H}_N &= \hat{F}_N + \hat{V}_N \\ &= \sum_{pq} f_{pq}\{a_p^\dagger a_q\} + \sum_{pqrs}\langle pq||rs\rangle\{a_p^\dagger a_q^\dagger a_s a_r\},\end{aligned} \tag{3.52}$$

with

$$f_{pq} = \langle p|\hat{h}^{(0)}|q\rangle + \sum_i \langle pi||qi\rangle. \tag{3.53}$$

Note that the normal-ordered Hamiltonian does by definition not contain any scalar terms, representing the ground state energy $E_0 = \langle\Phi_0|\hat{H}|\Phi_0\rangle$,

$$\hat{H}_N = \hat{H} - E_0. \tag{3.54}$$

When collecting those scalar terms of Eqs. (3.41), (3.46) and (3.47) , the ground state energy of the three-body Hamiltonian is given by

$$\begin{aligned}E_0 &= \langle\Phi_0|\,H\,|\Phi_0\rangle \\ &= \sum_i \langle i|\,\hat{h}^{(0)}\,|i\rangle + \frac{1}{2}\sum_{ij} \langle ij|\,\hat{v}\,|ij\rangle + \frac{1}{6}\sum_{ijk} \langle ijk|\,\hat{v}^{(3)}\,|ijk\rangle\,,\end{aligned} \tag{3.55}$$

and the one of the two-body Hamiltonian by

$$E_0 = \sum_i \langle i| \hat{h}^{(0)} |i\rangle + \frac{1}{2} \sum_{ij} \langle ij| \hat{v} |ij\rangle . \tag{3.56}$$

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 instead use a library [11], 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 [12] and bases on Adams methods, which we therefore put focus on in the following. Unless otherwise stated, we follow the explanations in [13, 14].

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,

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

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 in simplified notation is described by

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

It seems reasonable that, for well-behaved $\mathbf{f}$, any point $\mathbf{y}_0$ 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, on 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 $\mathbf{y}(x)$ has to be 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 *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 is, for example, to employ a Taylor series, such that

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

For $p = 1$, this is 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 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 those 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

$$\mathbf{k}_j = \mathbf{f}(x_i + \alpha h_i, \mathbf{y} + h_i \sum_{l=1}^{j-1} \beta_{jl} \mathbf{k}_l), \qquad j = 1, 2, \dots J$$
$$\mathbf{y}_{i+1} = \mathbf{y}_i + h_i \sum_{j=1}^{J} \gamma_k \mathbf{k}_j.$$

The constants $a_j, \beta_{jl}, \gamma_j$ 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

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

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 good results, they do only use the information provided by the last point, and not the ones before. For this reason, other methods have been developed which are called *multi-step methods*.

Adam's 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 the 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 the numerical approximation of k 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 one 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 Eq. (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 $\{(x_n, \mathbf{f}_n) | n = i-k+1, \ldots, i\}$. Employing Newton's interpolation formula, this polynomial can be expressed as follows:

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

with the relations

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

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

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

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

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

and Eq. (4.7) can be approximated by

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

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

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

and the first examples are:

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

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

In general, Eq. (4.11) gives a more accurate approximation to the exact solution than Eq. (4.9) 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 defined implicitly, which in general results in a non-linear equation at each step. To solve this equation, J.C. Adams himself used Newton's method, which is still done when encountering stiff equations [14]. Another possibility are predictor-corrector methods, which we already mentioned in the previous section. For the Adams-Moulton method, one proceeds 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 employed in 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 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, which is needed for all following calculations.

5.1 The model Hamiltonian

As discussed in the previous chapter, we can express the Hamiltonian as sum of a non-interacting and an interacting part. The non-interacting part is given as 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 and a potential part.

5.1.1 Harmonic oscillator basis

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

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

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,

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

Inserting for both factors the expression derived in Eq. (2.56), 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 chapter 2, too,

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

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* [15],

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

where n is the nodal quantum number ($n = 0, 1, 2, \ldots$) , and equals the number of nodes in the radial part, m_l is the orbital angular momentum quantum number ($m_l = 0, \pm1, \pm2, \ldots$), and $L_n^{|m_l|}(x)$ are the 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 as

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

For further properties, we refer to [1,2]. 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 take into account that each orbit ϕ_{n,m_l} can be occupied by two particles, and specify the first one to have spin up, $m_s = +\frac{1}{2}$, and the second one to have spin down, $m_s = -\frac{1}{2}$. 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 energy and quantum numbers n and m_l is determined by Eq. (5.8). All eigenfunctions with same energy span a single-particle *shell.* We define the shell number R as[1]

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

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

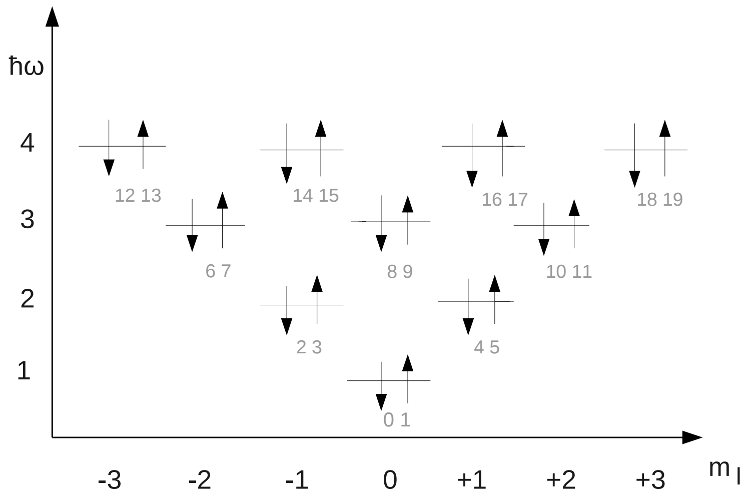


Figure 5.1: Labelling of the single-particle states of the two-dimensional harmonic oscillator. The x-axis shows the angular momentum quantum number, the y-axis the energy in units of $\hbar\omega$. Each energy level defines a shell R, with degenerate eigenvalues ϵ_i for the corresponding single-particle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

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 [16].

5.1.2 Choice of interaction

Having N electrons trapped in a two-dimensional isotropic harmonic oscillator potential, it is necessary to specify the type of interaction. In our case, we choose a two-body Coulomb potential,

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

where $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ denotes the distance between two particles, e the electron charge, ϵ_0 the vacuum permittivity, and ϵ_r the relative permittivity. Note that for the remainder of this thesis, we will use atomic units, setting $\hbar = m = e = 1/4\pi\epsilon_0 = 1$. Thus the energy is given in Hartrees,

$$E_H = \frac{m_e e^4}{(4\pi\epsilon_0\hbar)^2} = 4.35974418 \times 10^{-18} J. \tag{5.10}$$

This system of natural 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. In these units, the total Hamiltonian is given by

$$\begin{aligned}\hat{H} &= \hat{H}_0 + \hat{H}_I \\ &= \sum_{i=1}^{N}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}}.\end{aligned} \tag{5.11}$$

5.2 Model space

The basis functions for 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 particles 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, called *model space*. The model space has a basis $\mathcal{B}$ with a finite number of Slater determinants, and its orthogonal projector is given by

$$P = \sum_{|\Phi_b\rangle \in \mathcal{B}} |\Phi_b\rangle\langle\Phi_b|. \tag{5.12}$$

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:

$$\mathcal{B} = \mathcal{B}_R = \left\{ |\Phi_b\rangle : \max_{i=1\ldots N} R_i \leq R \right\}, \tag{5.13}$$

where R is called energy-cut[2]. 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 $\hat{H}$

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

$$[\hat{H}, \hat{L}_z] = [\hat{H}, \hat{S}_z] = 0. \tag{5.14}$$

Here $\hat{L}_z$ is the angular momentum operator with eigenvalue[3] $M = \sum_{i=1}^N m_i$,

$$\sum_{i=1}^N \hat{L}_z(i)|\Phi\rangle = M|\Phi\rangle. \tag{5.15}$$

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

$$S_z = \frac{1}{2}\sum_{p,\sigma} \sigma a_{p,\sigma}^\dagger a_{p,\sigma}, \tag{5.16}$$

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.

In principle, the matrix blocks could be even smaller, since our Hamiltonian (5.11) also commutes with the total spin $\hat{S}^2$, given by

$$\hat{S}^2 = \hat{S}_z^2 + \frac{1}{2}(\hat{S}_+\hat{S}_- + \hat{S}_-\hat{S}_+), \tag{5.17}$$

[2]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 [15].

[3]Since we are now using dimensionless units and not dealing with masses m any more, we use the common notation $m_l \to m$ for the azimuthal quantum number.

where the spin raising and lowering operators are

$$S_{\pm} = \hat{S}_x \pm i\hat{S}_y = \sum_p a^\dagger_{p\pm} a_{p\mp}. \tag{5.18}$$

However, since the Slater determinants are not eigenfunctions of $\hat{S}^2$, we would have to take a linear combination of them to obtain this property [16]. Therefore we restrict us to identities (5.14).

Part II

METHODS

Chapter 6

SRG

In this chapter, we introduce the theory of 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 of the method: first an application to free space, and afterwards we explain how to model SRG in-medium, when making use of the technique of normal-ordering.

6.1 General aspects

The SRG method was introduced independently by Glazek and Wilson [17, 18] and Wegner [19, 20] 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 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 analogously

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

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 $U_{s=0} = 1$, and $\hat{H}_{s=0} = \hat{H}$. Taking the derivative with respect to s, one obtains

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

Since the transformation U_s is unitary with $U_sU_s^\dagger = 1$, we have that

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

and introduce $\hat{\eta}_s$ as *generator* of the transformation. Inserting Eq.(6.3) into Eq. (6.2) gives

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

the key expression of the SRG method, which describes the flow of the Hamiltonian.

With the generator $\hat{\eta}_s$ introduced, the unitary transformation can formally be rewritten as [21]

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

where T_s denotes s-ordering. This one 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.. This gives

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

with permutations $\pi \in S_n$ such that $\hat{\eta}_{s_{\pi(1)}} \geq \hat{\eta}_{s_{\pi(2)}} \geq \dots \geq \hat{\eta}_{s_{\pi(n)}}$.

6.2 Choice of the generator $\hat{\eta}$

The specific unitary transformation is determined by the choice of $\hat{\eta}_s$, which is subject to the condition

$$\hat{\eta}_s^\dagger = -\hat{\eta}_s,$$

following from Eq.(6.14). 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 [19] 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 extensively been 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 [21]. Due to its many successful applications in condensed matter and nuclear physics, it will be one of the choices in this thesis, too.
However, also other choices are possible and might even have better numerics or efficiency. In our case 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, too [22–25], and later on, we will compare its effect on the flow equations with Wegner's generator.

6.2.2 White's generator

Apart from the canonical generator, there exist several other ones in literature. One of them is White's choice [26], which has been shown to make numerical approaches much more efficient.

The problem with the canonical 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 are needed.
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 barely 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 the 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, this one 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 annihilation operators, and a_α the corresponding coefficient. Hence, in Eq.(6.8), we have $a_1 = \langle p| \hat{h}^{(0)} |q\rangle$ 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 a 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 that should ensure that the $a_\alpha(s)$ corresponding 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 therefore should not 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 $d^\dagger$, which satisfy

$$d_i |\Phi_0\rangle = 0. \tag{6.13}$$

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

$$\{d_i^\dagger, d_j^\dagger\} = \{d_i, d_j\} = 0$$
$$\{d_i^\dagger, d_j\} = \delta_{i,j}.$$

With those operators, the 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

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

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, 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 is dependent on the choice of the generator η.
In order to get a feeling of how the Hamiltonian is driven towards diagonal form with the SRG method, we will start with the easiest generator $\eta_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$, where we suppress the s-dependence for simplicity.

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 generator, the flow of the matrix elements is computed as follows:

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

which finally gives

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

To obtain the same expression one frequently encounters in literature (e.g. [23]), one can rewrite this equation in the space of relative momentum k (with normalization $1 = \frac{2}{\pi}\int_0^\infty |q\rangle q^2 dq \langle q|$ and in atomic units where $\hbar^2/m = 1$), 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 $V_{ij}(s)$ at the first value of s equals the initial potential.

To get an idea of how the off-diagonal elements are suppressed, let us approximate the flow of the Hamiltonian in Eq. (6.15). In a first approximation, the evolution of the matrix elements is given by

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

with the solution

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

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 close 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 of the width of the diagonal band [19, 21]. 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 increase s 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, where 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

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

When running our calculations, we will analyse how the small difference between those 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 [21], which has recently been successfully applied in nuclear physics [27, 28]. This approach is called in-medium SRG (IM-SRG) and allows the evolution of $3, ..., A$-body operators using only two-body machinery. These simplifications arise from the use of normal-ordering with respect to a reference state with finite density.

To explain the concept, let us consider a second-quantized Hamiltonian with two- and thee-body interactions,

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

As demonstrated in chapter 3, normal-ordering with respect to a reference state $|\Psi_0\rangle$ yields

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

with the amplitudes for the one-body operator given by

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

the ones for the two-body operator

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

and the ones for the three-body operator

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

As in chapter 3, we use the notation that indices $\{a, b, c, ...\}$ denote particles states, $\{i, j, k, ...\}$ denote hole states, and $\{p, q, r, ...\}$ can be used for both particle and hole states.

In relation to the full Hamiltonian, $\hat{H}_N$ is obtained as follows (see chapter 3),

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

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

$$\begin{aligned}E_0 &= \langle \Phi_0| \hat{H} |\Phi_0\rangle \\ &= \sum_i \langle i| \hat{h}^{(0)} |i\rangle + \frac{1}{2}\sum_{ij} \langle ij| |ij\rangle + \frac{1}{6}\sum_{ijk} \langle ijk| \hat{v}^{(3)} |ijk\rangle .\end{aligned}$$

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 the generator $\hat{\eta}$ in the language of second quantization and normal-ordering, too.

Using this approach, one faces one of the major challenges of the SRG method, which is the generation of higher and higher order interaction terms during the flow. Each time the derivative is computed, the generator $\hat{\eta}$ and interaction V gain terms of higher order, and this continues in principle 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 fewer orders one includes, the higher the truncation error is. On the other hand, improving the result is quite straightforward by including higher order interaction terms. If one included all generated terms until the result has converged within the desired accuracy, one would obtain the same results as with SRG in free space.

Truncating to normal-ordered three-body operators, the generator $\hat{\eta}$ can be written as

$$\hat{\eta} = \sum_{pq} \eta^{(1)}_{pq} \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} \eta^{(2)}_{pqrs} \{a^\dagger_p a^\dagger_q a_s a_r\} + \frac{1}{36} \sum_{pqrstu} \eta^{(3)}_{pqrstu} \{a^\dagger_p a^\dagger_q a^\dagger_r a_u a_t a_s\},$$

where $\eta^{(1)}_{pq}, \eta^{(2)}_{pqrs}$ and $\eta^{(3)}_{pqrstu}$ are the one-, two- and three-body operator of the generator, respectively. For different choices of $\hat{\eta}$, those amplitudes look differently and we will demonstrate in section 6.4.1, how explicitly to compute them for a Hamiltonian containing maximal two-body interactions.
Including also three-body interactions, the flow equations are given by

$$\begin{aligned}\frac{d\hat{H}_s}{ds} &= \left[\hat{\eta}_s, \hat{H}_s\right] \\ &= \Bigg[\sum_{pq} \eta^{(1)}_{pq} \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} \eta^{(2)}_{pqrs} \{a^\dagger_p a^\dagger_q a_s a_r\} + \frac{1}{36} \sum_{pqrstu} \eta^{(3)}_{pqrstu} \{a^\dagger_p a^\dagger_q a^\dagger_r a_u a_t a_s\}, \\ &\quad \sum_{pq} f_{pq} \{a^\dagger_p a_q\} + \frac{1}{4} \sum_{pqrs} \Gamma_{pqrs} \{a^\dagger_p a^\dagger_q a_s a_r\} + \frac{1}{36} \sum_{pqrstu} W_{pqrstu} \{a^\dagger_p a^\dagger_q a^\dagger_r a_u a_t a_s\}\Bigg],\end{aligned}$$

where we suppressed the s-dependence for simplicity.

Using the commutation relations presented in Appendix A and collecting the constants in E_0, the one-body terms in f, the two-body terms in Γ and the three-body terms in W, we obtain Eq. (6.21-6.24), where we make use of the permutation operators

$$\hat{P}_{pq} f(p,q) = f(q,p), \quad \hat{P}(pq/r) = 1 - \hat{P}_{pq} - \hat{P}_{qr}, \quad \hat{P}(p/qr) = 1 - \hat{P}_{pq} - \hat{P}_{pr}. \tag{6.20}$$

The derivative of E_0 is given by

$$\frac{d}{ds} E_0(s) = \sum_{ia} (1 - \hat{P}_{ia}) \eta^{(1)}_{ia} f_{ai} + \frac{1}{2} \sum_{ijab} \eta^{(2)}_{ijab} \Gamma_{abij} + \frac{1}{18} \sum_{ijkabc} \eta^{(3)}_{ijkabc} W_{abcijk}, \tag{6.21}$$

the one of the one-body operator by

$$
\begin{aligned}
\frac{d}{ds}f_{pq}(s) = & \sum_r (1+\hat{P}_{pq})\eta^{(1)}_{pr}f_{rq} + \sum_{ia}\left(1-\hat{P}_{ia}\right)\left(\eta^{(1)}_{ia}\Gamma_{apiq} - f_{ia}\eta^{(2)}_{apiq}\right) \\
& + \frac{1}{2}\sum_{ija}\left(1+\hat{P}_{pq}\right)\eta^{(2)}_{apij}\Gamma_{ijaq} + \frac{1}{2}\sum_{abi}\left(1+\hat{P}_{pq}\right)\eta^{(2)}_{ipab}\Gamma_{abiq} \\
& + \frac{1}{4}\sum_{ijab}\left(\eta^{(3)}_{ijpqab}\Gamma_{abij} - W_{ijpqab}\eta^{(2)}_{abij}\right) + \frac{1}{12}\sum_{ijabc}\left(\eta^{(3)}_{ijpabc}W_{abcijq} - W_{ijpabc}\eta^{(3)}_{abcijq}\right) \\
& + \frac{1}{12}\sum_{abijk}\left(\eta^{(3)}_{abpijk}W_{ijkabq} - W_{abpijk}\eta^{(3)}_{ijkabq}\right), \qquad (6.22)
\end{aligned}
$$

the derivative of the two-body operator by

$$
\begin{aligned}
\frac{d}{ds}\Gamma_{pqrs}(s) = & \sum_t \left[\left(1-\hat{P}_{pq}\right)\left(\eta^{(1)}_{pt}\Gamma_{tqrs} - f_{pt}\eta^{(2)}_{tqrs}\right) - \left(1-\hat{P}_{rs}\right)\left(\eta^{(1)}_{tr}\Gamma_{pqts} - f_{tr}\eta^{(2)}_{pqts}\right)\right] \\
& + \frac{1}{2}\sum_{ab}\left(\eta^{(2)}_{pqab}\Gamma_{abrs} - \Gamma_{pqab}\eta^{(2)}_{abrs}\right) - \frac{1}{2}\sum_{ij}\left(\eta^{(2)}_{pqij}\Gamma_{ijrs} - \Gamma_{pqij}\eta^{(2)}_{ijrs}\right) \\
& - \sum_{ia}\left(1-\hat{P}_{ia}\right)\left(1-\hat{P}_{pq}\right)\left(1-\hat{P}_{rs}\right)\eta^{(2)}_{aqis}\Gamma_{ipar} + \sum_{ia}\left(1-\hat{P}_{ia}\right)\left(\eta^{(3)}_{apqirs}f_{ai} - W_{apqirs}\eta^{(1)}_{ai}\right) \\
& + \frac{1}{2}\sum_{abi}\left(1-\hat{P}_{pr}\hat{P}_{qs}\hat{P}_{pq} - \hat{P}_{rs} + \hat{P}_{pr}\hat{P}_{qs}\right)\left(\eta^{(3)}_{ipqabs}\Gamma_{abir} - W_{ipqabs}\eta^{(2)}_{abir}\right) \\
& + \frac{1}{2}\sum_{aij}\left(1-\hat{P}_{pr}\hat{P}_{qs}\hat{P}_{pq} - \hat{P}_{rs} + \hat{P}_{pr}\hat{P}_{qs}\right)\left(\eta^{(3)}_{apqijs}\Gamma_{ijar} - W_{apqijs}\eta^{(2)}_{ijar}\right) \\
& + \frac{1}{6}\sum_{iabc}\left(\eta^{(3)}_{ipqars}W_{abcirs} - \eta^{(3)}_{abcirs}W_{ipqabc}\right) + \frac{1}{6}\sum_{iabc}\left(\eta^{(3)}_{ipqars}W_{abcirs} - \eta^{(3)}_{abcirs}W_{ipqabc}\right) \\
& + \frac{1}{4}\sum_{abij}\left(1-\hat{P}_{pq}\right)\left(1-\hat{P}_{rs}\right)\left(\eta^{(3)}_{abpijs}W_{ijqabr} - \eta^{(3)}_{ijpabs}W_{abqijr}\right), \qquad (6.23)
\end{aligned}
$$

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 dependent on the number of shells R.

and finally the one of the three-body operator by

$$\begin{aligned}
\frac{d}{ds}W_{pqrstu}(s) = &\sum_{v}\left[\hat{P}(p/qr)\eta^{(1)}_{pv}W_{vqrstu} - \hat{P}(s/tu)\eta^{(1)}_{vs}W_{pqrvtu}\right] \\
&-\sum_{v}\left[\hat{P}(p/qr)f_{pv}\eta^{(3)}_{vqrstu} - \hat{P}(s/tu)f_{vs}\eta^{(3)}_{pqrvtu}\right] \\
&+\sum_{v}\hat{P}(pq/r)\hat{P}(s/tu)\left(\eta^{(2)}_{pqsv}\Gamma_{vrtu} - \Gamma_{pqsv}\eta^{(2)}_{vrtu}\right) \\
&+\frac{1}{2}\sum_{ab}\hat{P}(p/qr)\left(\eta^{(2)}_{pqab}W_{abrstu} - \Gamma_{pqab}\eta^{(3)}_{abrstu}\right) - \frac{1}{2}\sum_{ij}\hat{P}(p/qr)\left(\eta^{(2)}_{pqij}W_{ijrstu} - \Gamma_{pqij}\eta^{(3)}_{ijrstu}\right) \\
&-\frac{1}{2}\sum_{ab}\hat{P}(s/tu)\left(\eta^{(2)}_{abtu}W_{pqrsab} - \Gamma_{abtu}\eta^{(3)}_{pqrsab}\right) + \frac{1}{2}\sum_{ij}\hat{P}(s/tu)\left(\eta^{(2)}_{ijtu}W_{pqrsij} - \Gamma_{ijtu}\eta^{(3)}_{pqrsij}\right) \\
&-\sum_{ia}\left(1-\hat{P}_{ia}\right)\hat{P}(p/qr)\hat{P}(s/tu)\left(\eta^{(2)}_{apis}W_{iqratu} - \Gamma^{(2)}_{apis}\eta^{(3)}_{iqratu}\right) \\
&+\frac{1}{6}\sum_{ijk}\left(\eta^{(3)}_{pqrijk}W_{ijkstu} - W_{pqrijk}\eta^{(3)}_{ijkstu}\right) + \frac{1}{6}\sum_{abc}\left(\eta^{(3)}_{pqrabc}W_{abcstu} - W_{pqrabc}\eta^{(3)}_{abcstu}\right) \\
&+\frac{1}{2}\sum_{aij}\hat{P}(pq/r)\hat{P}(s/tu)\left(\eta^{(3)}_{ijratu}W_{apqijs} - \eta^{(3)}_{aqriju}W_{pijsta}\right) \\
&+\frac{1}{2}\sum_{abi}\hat{P}(pq/r)\hat{P}(s/tu)\left(\eta^{(3)}_{abritu}W_{ipqabs} - \eta^{(3)}_{iqrabu}W_{pabsti}\right) \qquad (6.24)
\end{aligned}$$

Note that on the right-hand side terms the s-dependence of all the amplitudes has been skipped for better readability. The large number of summations makes the equations computationally quite expensive, as they have to be computed for each integration step. Table 6.1 gives some examples of how the number of equations is quickly increasing with number of shells R.

Especially the computational cost connected to the three-body operators is very high and time-consuming, Therefore, a possible alternative is to truncate the flow equations (6.21)-(6.24) up to normal-ordered two-body operators, an approach which is called IM-SRG(2).

6.4.1 IM-SRG(2) for two-body Hamiltonians

In the case of quantum dots, we look at a Hamiltonian

$$\hat{H} = \sum_{pq} \langle p | \hat{h}^{(0)} | q \rangle a_p^\dagger a_q + \frac{1}{4} \sum_{pqrs} \langle pq | | rs \rangle a_p^\dagger a_q^\dagger a_s a_r. \tag{6.25}$$

Normal-orderding the creation and annihilation operators, Eq.(6.25) is equivalent to

$$\hat{H} = E_0 + \sum_{pq} f_{pq} \{a_p^\dagger a_q\} + \frac{1}{4} \sum_{pqrs} v_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\}, \tag{6.26}$$

with ground state energy

$$E_0 = \sum_{i} \langle i | \hat{h}^{(0)} | i \rangle + \frac{1}{2} \sum_{ij} \langle ij | | ij \rangle , \tag{6.27}$$

and where $v_{pqrs} = \langle pq||rs \rangle$. The amplitudes for the one-body operator are in this case defined by

$$f_{pq} = \langle p | \hat{h}^{(0)} | q \rangle + \sum_{i} \langle pi | | qi \rangle . \tag{6.28}$$

Truncating flow-equations (6.21)-(6.24) to normal-ordered two-body operators, we obtain the following simplified set of ordinary differential equations:

$$\frac{dE_0}{ds} = \sum_{ia} \left(\eta_{ia}^{(1)} f_{ai} - \eta_{ai}^{(1)} f_{ia} \right) + \frac{1}{2} \sum_{ijab} \eta_{ijab}^{(2)} v_{abij} \tag{6.29}$$

$$\begin{aligned} \frac{df_{pq}}{ds} = &\sum_{r} \left(\eta_{pr}^{(1)} f_{rq} + \eta_{qr}^{(1)} f_{rp} \right) + \sum_{ia} \left(1 - \hat{P}_{ia} \right) \left(\eta_{ia}^{(1)} v_{apiq} - f_{ia} \eta_{apiq}^{(2)} \right) \\ &+ \frac{1}{2} \sum_{aij} \left(1 + \hat{P}_{pq} \right) \eta_{apij}^{(2)} v_{ijaq} + \frac{1}{2} \sum_{abi} \left(1 + \hat{P}_{pq} \right) \eta_{ipab}^{(2)} v_{abiq} \end{aligned} \tag{6.30}$$

$$\begin{aligned} \frac{dv_{pqrs}}{ds} = &\sum_{t} \left(1 - \hat{P}_{pq} \right) \left(\eta_{pt}^{(1)} v_{tqrs} - f_{pt} \eta_{tqrs}^{(2)} \right) - \sum_{t} \left(1 - \hat{P}_{rs} \right) \left(\eta_{tr}^{(1)} v_{pqts} - f_{tr} \eta_{pqts}^{(2)} \right) \\ &+ \frac{1}{2} \sum_{ab} \left(\eta_{pqab}^{(2)} v_{abrs} - v_{pqab} \eta_{abrs}^{(2)} \right) - \frac{1}{2} \sum_{ij} \left(\eta_{pqij}^{(2)} v_{ijrs} - v_{pqij} \eta_{ijrs}^{(2)} \right) \\ &- \sum_{ia} \left(1 - \hat{P}_{ia} \right) \left(1 - \hat{P}_{pq} \right) \left(1 - \hat{P}_{rs} \right) \eta_{aqis}^{(2)} v_{ipar}. \end{aligned} \tag{6.31}$$

Compared to the third-order flow equations (6.21)-(6.24), not only the number of terms is considerably reduced, but also number of indices to be summed over reaches maximally four instead of six, a very important fact considering computational efficiency.

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

$$\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}} \right] = \left[\hat{H}^{\mathrm{d}}, \hat{H} \right] . \tag{6.32}$$

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

$$f^d_{pq} = f_{pq}\delta_{pq}, \qquad v^d_{pqrs} = v_{pqrs}\left(\delta_{pr}\delta_{qs} + \delta_{ps}\delta_{qr}\right). \tag{6.33}$$

Using the commutation relations of Appendix A, we get terms of first, second and third order, that can be collected in $\hat{\eta}^{(1)}, \hat{\eta}^{(2)}$ and $\hat{\eta}^{(3)}$, respectively.

In IM-SRG(2), however, we truncate the generator $\hat{\eta}$ to

$$\hat{\eta} = \sum_{pq}\{a^\dagger_p a_q\}\eta^{(1)}_{pq} + \sum_{pqrs}\{a^\dagger_p a^\dagger_q a_s a_r\}\eta^{(2)}_{pqrs}, \tag{6.34}$$

which means that we only consider contributions of $\hat{\eta}^{(1)}$ and $\hat{\eta}^{(2)}$.
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

$$\eta^{(1)}_{pq} = \sum_r \left(f^d_{pr} f_{rq} - f_{pr} f^d_{rq}\right) + f_{pq} v^d_{qppq}\left(n_q - n_p\right) \tag{6.35}$$

$$\begin{aligned}
\eta^{(2)}_{pqrs} =& -\sum_t \{\left(1 - \hat{P}_{pq}\right) f_{pt} v^d_{tqrs} - \left(1 - \hat{P}_{rs}\right) f_{tr} v^d_{pqts}\} \\
&+ \sum_t \{\left(1 - \hat{P}_{pq}\right) f^d_{pt} v_{tqrs} - \left(1 - \hat{P}_{rs}\right) f^d_{tr} v_{pqts}\} \\
&+ \frac{1}{2}\sum_{tu}(1 - n_t - n_u)\left(v^d_{pqtu} v_{turs} - v_{pqtu} v^d_{turs}\right) \\
&+ \sum_{tu}\left(n_t - n_u\right)\left(1 - \hat{P}_{pq}\right)\left(1 - \hat{P}_{rs}\right) v^d_{tpur} v_{uqts}
\end{aligned} \tag{6.36}$$

Considering relations (6.33), the sums can be simplified to

$$\eta^{(1)}_{pq} = \left(f_{pp} f_{pq} - f_{pq} f_{qq}\right) + f_{pq} v_{qppq}\left(n_q - n_p\right) \tag{6.37}$$

$$\begin{aligned}
\eta^{(2)}_{pqrs} =& f_{ps} v^d_{sqsq}\delta_{qr} + f_{qr} v^d_{rprp}\delta_{pr} - f_{pr} v^d_{rqrq}\delta_{qs} - f_{qs} v^d_{spsp}\delta_{pr} \\
&- \left(f_{qr}\delta_{ps} + f_{ps}\delta_{qr} - f_{pr}\delta_{qs} - f_{qs}\delta_{pr}\right) v^d_{pqpq} \\
&+ \left(f^d_{pp} + f^d_{qq} - f^d_{rr} - f^d_{ss}\right) v_{pqrs} \\
&+ \Big(v^d_{pqpq}(1 - n_p - n_q) - v^d_{rsrs}(1 - n_r - n_s) - v^d_{rprp}(n_r - n_p) \\
&- v^d_{rqrq}(n_r - n_q) - v^d_{spsp}(n_s - n_p) - v^d_{sqsq}(n_s - n_q)\Big) v_{pqrs}.
\end{aligned} \tag{6.38}$$

With this simplification, 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 initial generator $\hat{\eta}$ also includes terms of higher order, even if the Hamiltonian itself is only on a two-body level. These terms $\eta^{(3)}_{pqrstu}$ then induce higher order interaction terms in the Hamiltonian, making the Hamiltonian more and more complex. However, in the SRG(2) approach, both, the generator $\hat{\eta}$ and the flow equations are truncated to terms with maximal two creation and two annihilation operators.

IM-SRG(2) with White's generator

In [28], White's generator for the in-medium approach is explicitly derived for nuclear systems, an expression that we also can apply to our system of quantum dots.
In Eq. (6.11), White's generator has been given as

$$\hat{\eta}(s) = \sum_{\alpha} \eta_\alpha(s) h_\alpha, \tag{6.39}$$

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:

$$h_\alpha \in \left\{ \{a^\dagger_p a_h\}, \{a^\dagger_h a_p\}, \{a^\dagger_{p_1} a^\dagger_{p_2} a_{h_2} a_{h_1}\}, \{a^\dagger_{h_1} a^\dagger_{h_2} a_{p_2} a_{p_1}\} \right\}. \tag{6.40}$$

Here, indices h denote hole states, whereas indices p denote particle states. The terms in (6.40) can be divided into two types: The expressions $\{a^\dagger_p a_h\}$ and $\{a^\dagger_{p_1} a^\dagger_{p_2} a_{h_2} a_{h_1}\}$ contain only operators of $d^\dagger$-type (see section 6.2.2) when applied to $|\Phi_0\rangle$. Therefore, they are assigned to the left state. On the other hand, $\{a^\dagger_h a_p\}$ and $\{a^\dagger_{h_1} a^\dagger_{h_2} a_{p_2} a_{p_1}\}$ correspond to d-operators when applied to $|\Phi_0\rangle$ and are therefore assigned to the right state.

Since IM-SRG(2) is restricted to one-particle-one-hole and two-particle-two-hole excitations, 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: $\eta^{(1)}_{ph}$.
Due to anti-hermiticity of the generator $\hat{\eta}$, $\eta^{(1)}_{hp}$ is obtained by the relation $\eta^{(1)}_{hp} = -\eta^{(1)}_{ph}$.
For the non-zero two-body elements, corresponding to operators $h_\alpha = \{a^\dagger_{p_1} a^\dagger_{p_2} a_{h_2} a_{h_1}\}$ and $h_\alpha = \{a^\dagger_{h_1} a^\dagger_{h_2} a_{p_2} a_{p_1}\}$, we need a combination of two particles and two holes. Anti-hermiticity again limits this to one relevant term, since $\eta^{(2)}_{p_1p_2h_1h_2} = -\eta^{(2)}_{h_1h_2p_1p_2}$. To derive the expressions, we will follow [28].

For the one-body elements $\eta^{(1)}_{ph}$, we need the left and right states defined as

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

This yields for the corresponding energies

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

where we suppress the s-dependence on the right-hand side for simplicity. Note that in his original article [26], White even suggests just to take the initial values of E_r^α and E_l^α and mentions the s-dependent values only as a further possibility. In [28], 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

$$\eta_{ph}^{(1)}(s) = \frac{1}{f_{pp}(s) - f_{hh}(s) - v_{phph}(s)} f_{ph}(s), \tag{6.41}$$

where we have renamed the indices appropriately. For the two-body elements $\eta^{(2)}_{p_1p_2h_1h_2}$, we need the left and right states

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

This yields the following left and right energies:

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

where we again skipped the s-dependence for better readability and introduced

$$A_{abmn} = v_{abab} + v_{mnmn} - v_{amam} - v_{anan} - v_{bmbm} - v_{bnbn} \tag{6.42}$$

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

$$\eta_{p_1p_2h_1h_2}^{(2)}(s) = \frac{1}{f_{p_1p_1}(s) + f_{p_2p_2}(s) - f_{h_1h_1}(s) - f_{h_2h_2}(s) + A_{p_1p_2h_1h_2}(s)} v_{p_1p_2h_1h_2}(s) \tag{6.43}$$

Inserting Eqs. (6.41) and (6.43) in Eq. (6.39) , White's generator truncated to second order can in total be written as

$$\hat{\eta} = \sum_{ai} \frac{f_{ai}}{f_a - f_i - v_{aiai}} \{a_a^\dagger a_i\} - \text{hc} + \sum_{abij} \frac{v_{abij}}{f_a + f_b - f_i - f_j + A_{abij}} \{a_a^\dagger a_b^\dagger a_j a_i\} - \text{hc}, \quad (6.44)$$

with the common notation $f_p \equiv f_{pp}$, "hc" denoting the hermitian conjugate and a suppressed s-dependence for better readability. When summing over indices, we use as before $\{a, b\}$ for particle and $\{i, j\}$ for hole states.

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 solving 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 [29]. 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 for example [30]) 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 [29].

As an ansatz, one assumes 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, that are to be minimized. In this thesis, we use the approach to expand the single-particle states $|p\rangle$, which we refer to as

HF 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. (3.37), we restate the ground state energy:

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

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 is in principal infinitely large.
To minimize the energy functional (7.3), we employ the technique of Lagrange multipliers, with 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

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

and 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 $\hat{h}^{HF}$ links only 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 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 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 natural 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.
In our case, one is only interested in projecting the ground state component of Ψ out. Therefore a constant energy shift E_T, called a *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

$$|\Psi(\mathbf{R},t)\rangle = \sum_i e^{-(\hat{H}-E_0)t}|\mathbf{R}_i'\rangle\langle\mathbf{R}_i'|\Psi(\mathbf{R}',0)\rangle. \tag{7.9}$$

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

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

The Green's function represents the probability that the system moves from $\mathbf{R}$ to $\mathbf{R}'$ in an imaginary time interval τ and is given by

$$\begin{aligned} G(\mathbf{R}',\mathbf{R};\tau) &= \langle\mathbf{R}'|e^{-(\hat{H}-E_0)\tau}|\mathbf{R}\rangle & (7.10)\\ &= \langle\mathbf{R}'|e^{-\left(\hat{T}+\hat{V}-E_T\right)\tau}|\mathbf{R}\rangle, & (7.11)\end{aligned}$$

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

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

which is of the form of an extended diffusion equation.

The basic idea is now 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 $D = \frac{\hbar^2}{2m}$. 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 [7] is applied in the limit $\tau \to 0,\ t = n\tau$, yielding

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

with an error to second order in τ. Since it is only valid for small time steps τ, Eq. (7.13) is called a *short time approximation* of the Green's function. The Green's function, ignoring normalization factors, thus reads [31]

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

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

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

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

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

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 $E_T \to \epsilon_0$, the local energy is 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 good as possible. One the one hand, the fixed-node approximation 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 walker 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. At the same time, however, 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 this function as input for the subsequent DMC runs.

[1]For details, see [31].

Our ansatz for the wave function

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

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

where ω is the oscillator frequency. The wave functions of the different levels of excitation, which are solutions to the single-particle Hamiltonians

$$\hat{h}_0\phi = \epsilon\phi,$$

are given by a product of Hermite polynomials (see section), namely

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

Following [32], 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

$$\psi_S(\mathbf{R}_1,...,\mathbf{R}_N) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \phi_1(\mathbf{R}_1) & \phi_1(\mathbf{R}_2) & \cdots & \phi_1(\mathbf{R}_N) \\ \phi_2(\mathbf{R}_1) & \phi_2(\mathbf{R}_2) & \cdots & \phi_2(\mathbf{R}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\mathbf{R}_1) & \phi_N(\mathbf{R}_2) & \cdots & \phi_N(\mathbf{R}_N) \end{vmatrix}, \tag{7.16}$$

where the functions ϕ_i are the single-particle orbitals discussed above 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 has been shown [33] that if one uses instead of the full Slater determinant the product of these two half-sized determinants, one gets exactly the same energy as with the full one, provided that the Hamiltonian is spin independent:

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

with

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

and

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

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 infinitely long 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, only the ground-state Slater determinant is used 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. One can therefore assume that it is very hard to excite a particle from the outer-most filled shell and it is therefore 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 taking care of the singularity one gets by having zero distance between two particles.
Here, the *Pade-Jastrow* function is used

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

where $a = \frac{1}{3}$ for particles of equal spin and $a = 1$, else. In this factor, 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 interactions is small. On the other hand, the smaller β becomes, the more central is the role of correlations in the system.

The variational parameter in the Slater determinant will be α and is included in the single-particle wave functions the following way

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

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, our total trial wave function is

$$\Psi_T(\alpha, \beta) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,J(\beta). \tag{7.17}$$

Extracting the parameters using Variational Monte Carlo (VMC)

Our ansatz for the trial wave function Ψ_T now 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

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

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, meaning that we combine Monte-Carlo integration with the Metropolis algorithm governing the transition of states.

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

$$I = \int_a^b f(x) dx. \tag{7.18}$$

From statistics it is known 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:

$$\langle f(x) \rangle = \int_a^b P(x) f(x) dx. \tag{7.19}$$

The main idea now is to bring integral (7.18) into the form of Eq.(7.19). 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 the integral (7.18) is replaced 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:

$$I = \langle \frac{f(x)}{P(x)} \rangle \simeq \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{P(x_i)}. \tag{7.20}$$

The points x_i are randomly generated from the probability distribution $P(x)$.
Obviously, Monte Carlo integration is a statistical method, and yields 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 As stated above, the main task to determine the desired wave function is to compute the integral

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

and determine those variational parameters that yield a minimum. The idea of Variational Monte Carlo is to calculate this integral using Monte Carlo integration and the Metropolis algorithm.

In principle, one could use any arbitrary PDF for Eq.(7.20) and compute the expectation value. A smarter choice, however, is to take a PDF which behaves similar to the original function and results in a smoother curve to integrate. In particular, one wants 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.

In the case of 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 local energy. In other words, the integral has been replaced by the expectation value of the local energy. This one is specific for a given trial wave function and therefore dependent on the set of variational parameters.

The sample variance is in this case expressed by

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

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

The Metropolis algorithm To calculate the integral

$$E\left[\Psi_T\right] = \int d\mathbf{R} E_L(\mathbf{R}) P(\mathbf{R}), \tag{7.21}$$

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. (7.21) must be chosen with care: In order to get the true expectation value, one has to follow the PDF in a correct way.

We therefore employ a *Markov chain*, i.e. 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. Marcov processes fulfil this requirement because all moves are independent of the previous history, which means that at every 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 mathematical. From transport theory one has the famous *master equation*, 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 any more, therefore we demand $\frac{dw}{dt} = 0$ and are left with

$$\sum_j \left[W(j \to i)\; w_j - W(i \to j)\; w_i\right] = 0. \tag{7.22}$$

Detailed balance now ensures the generation of the correct distribution by demanding that the system follows the trivial solution of Eq. (7.22), 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

$$\frac{W(j \to i)}{W(i \to j)} = \frac{w_i}{w_j}, \tag{7.23}$$

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 a 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. (7.23) gives

$$\frac{g(j \to i)A(j \to i)}{g(i \to j)A(i \to j)} = \frac{w_i}{w_j}. \tag{7.24}$$

In the standard *Metropolis* algorithm, one assumes that the walker's probability of picking the transition form i to j should not differ from picking the opposite direction j to i. This simplifies Eq. (7.24) 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

$$A(j \to i) = \left|\frac{\psi_i}{\psi_j}\right|^2 A(i \to j). \tag{7.25}$$

The probability for accepting a move to a state with higher probability equals 1. If in Eq. (7.25), state i has a higher probability than state j, then $A(i \to j) = 1$, and we get

$$A(j \to i) = \begin{cases} \left|\frac{\psi_i}{\psi_j}\right|^2 & |\psi_i|^2 > |\psi_j|^2 \\ 1 & \text{else.} \end{cases} \tag{7.26}$$

That way, we ensures that the walkers follow the path of the PDF, since it is the probability ratio between new and old state that decides whether a move is accepted or not. 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 above 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. (7.24) and choose a more advanced model for g, which pushes the walkers into the direction of higher probabilities.
As a starting point serves the *Fokker-Planck equation*, which describes an isotropic diffusion process where the particles are effected by an external potential:

$$\frac{\partial \rho}{\partial t} = D\nabla(\nabla - F)\rho. \tag{7.27}$$

The parameter D denotes the diffusion constant, which, considering Schrödinger's equation with dimensionless units, in our case is $D = \frac{1}{2}$. The variables 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

$$\frac{\partial \rho(\mathbf{R}, t)}{\partial t} = \sum_{i=1}^{N} D\nabla_i \left[\nabla_i - \mathbf{F}(\mathbf{r}_i)\right] \rho(\mathbf{r}_i, t). \tag{7.28}$$

The quantum force is determined by solving the Fokker-Planck equation (7.27) for stationary densities, in its simplest form when all terms in the sum are zero:

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

which has for our $\rho(\mathbf{R}_i) = |\psi_i|^2$ the solution

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

This expression tells very well what the quantum force is actually doing: The gradient of the wave function determines in which direction the walker should move to get to 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).

This diffusion equation, giving the desired distribution, can be used as input for the MC algorithm. In statistical mechanics, Fokker-Planck trajectories are generated via the *Langevin equation*, which in the case of Eq. (7.28) is

$$\frac{\partial \mathbf{r}(t)}{\partial t} = D\mathbf{F}\left(\mathbf{r}(t)\right) + \eta, \tag{7.29}$$

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

In order to make this equation numerically practicable, it has to be discretized in time $t \to t_n = n\Delta t$. Integrating over the short time interval Δt, we obtains an expression to generate the new trial positions,

$$\mathbf{r}' = \mathbf{r} + D\Delta t \mathbf{F}\left(\mathbf{r}\right) + \chi, \tag{7.30}$$

where χ is now 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 [31]

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

For the Metropolis-Hastings algorithm, one uses this solution instead of setting $g(i \to j) = g(j \to i)$. For Eq. (7.24), we thus get

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

The probability to perform a move is then given by

$$A(j \to i) = \begin{cases} R = \frac{G(\mathbf{R}', \mathbf{R}; \Delta t)}{G(\mathbf{R}, \mathbf{R}'; \Delta t)} \left|\frac{\psi_i}{\psi_j}\right|^2 & R < 1 \\ 1 & \text{else.} \end{cases} \tag{7.31}$$

The fraction involving the Greens functions can be simplified to

$$\frac{G\left(\mathbf{R}', \mathbf{R}; \Delta t\right)}{G\left(\mathbf{R}, \mathbf{R}'; \Delta t\right)} = e^{-\frac{1}{2}(\mathbf{F}(\mathbf{R}) + \mathbf{F}(\mathbf{R}'))\left(\frac{\Delta t}{4}(\mathbf{F}(\mathbf{R}) - \mathbf{F}(\mathbf{R}')) + (\mathbf{R} - \mathbf{R}'\right)}. \tag{7.32}$$

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. After closing in the region of contemplable parameters α and β, by setting up a coarse grid for various values, a linear algebra method called *Davidon-Fletcher-Powell* algorithm (DFP) [34] is used. This method bases on the *Conjugate gradient method* (CGM).

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

$$\hat{\mathbf{A}}\mathbf{x} = \mathbf{b}, \tag{7.33}$$

namely those where the matrix $\hat{\mathbf{A}}$ is symmetric and positive-definite.

To solve Eq. (7.33) iteratively, the residual $\mathbf{r} = \mathbf{b} - \hat{\mathbf{A}}\mathbf{x}$ must be minimized. It gets zero when the minimum of the quadratic equation

$$P(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\hat{\mathbf{A}}\mathbf{x} - \mathbf{x}^{\mathbf{T}}\mathbf{b}$$

is reached.

The CG method uses the properties of conjugate vectors, where two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be *conjugate* if they obey the relation

$$\mathbf{u}^{\mathbf{T}}\hat{\mathbf{A}}\mathbf{v} = 0. \tag{7.34}$$

The basic philosophy is now to find a sequence $\{\mathbf{p}_k\}$ of conjugate directions in which the search for minima is performed, and compute the expansion coefficients λ_k, such that

$$\mathbf{x} = \sum_i \lambda_i \mathbf{p}_i.$$

Hence at each iteration i, the approximation is improved by

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \lambda_i \mathbf{p}_i.$$

The DFP algorithm now takes a function $f(\mathbf{x})$ of variational parameters 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:

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

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

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

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

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

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 (7.17) 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 developed for this thesis. We discuss how we structured it, how the different items are related to each other to make the code as flexible and efficient as possible, and of course we show how we specifically implemented the methods of the two last chapters.
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. Analogous to its predecessor, 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 for the high-level performance of the code, which 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 mainly concentrate on those aspects that are relevant for understanding the chosen 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 [35].

8.1.1 Classes and objects

To combine data structures and methods for data manipulation handily into one package, C++ provides so-called *classes*. A class is used to specify the form of an object, and the 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 of 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, where 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 just 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 possibility is used in this thesis 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 the *constructor*, is called. In our example of *SPstate*, this is the function declared in line 8. A destructor, see line 9 in the 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 class, 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 base-class
```

Here *base-class* is the name of any previously defined class, and *access-specifier* must be one of {*public, protected, private*}. By default, *private* is used, and the specifier *protected* means that data members can be accessed within the class and all derived subclasses.

As an example, we consider the class *System* of our SRG code, which has a 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.

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 routines 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 *System*, as well as in *System_2DQdot*, and if we had further base classes of *System*, each of the classes could have one such a function with a specific implementation. To make it possible to select the specific function to be called, based on the kind of object for which it is called, we have defined *mapping* to be *virtual* in the base class. Defining a virtual function in a base class, with another version in a derived class, 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 its usage, consider the following example of our code:

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

The class *System* contains a member function *setup*, calling the function *mapping*, which we already explained to be a pure virtual function in the class definition of *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 example 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 object-oriented in 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. 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 complete matrix, with the matrix elements obtained by the action of creation and annihilation operators. In medium, on the other hand, it is more convenient to store the elements f_{pq} and v_{pqrs} separately, 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 the same argumentation holds for other parts of the code, e.g. the classes *Basis* and *SRG*, we decided on two separate codes.

Nevertheless, we tried to find as many common data structures and methods as possible, enabling us to have common classes that can be used by both methods. To keep everything as structured and transparent as possible, we chose the same names for the classes that are specific for the two approaches. 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 to contain the basis which the Hamiltonian is set up in. For IM-SRG, this class is in particular responsible for creating and administering a two-particle basis.
- Class *SPstate*: A class for holding the single-particle states.

Solver classes

- Class *SRG*: A 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 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 from the OpenFCI [37] library. The folder *OpenFCI* should be placed next to the *SRG* folder.
- File *Makefile*: Compiles our code by simply calling *make*.
- File *runSerial.py*: Python script for running the code with one specific choice for the input parameters at a time, usually several different runs after each other. The script

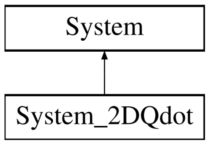


Figure 8.1: The class *System* is a virtual base class, and needs to be extended by subclasses containing system-specific methods and data structures.

calls the code in 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 previous files.

- File *runParallel.py*: Python script with the same functionality as *runSerial.py*. 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*. Parallelized with MPI (Message Passing Interface).

8.3 Implementation of SRG - general aspects

In the following, we will present the common parts of our program, before we move on and look at those parts that are specific for the free-space and in-medium implementation, respectively.

8.3.1 Class *System*

The first common class is the class *System*, whose task it is to hold all the data structures and methods that characterize a specific system, i.e. Hamiltonian, single-particle basis etc. 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 and nuclear systems, and in arbitrary many Euclidean dimensions. In order to have a general program that can easily be adopted to different kinds of physical problems, we have therefore constructed *System* as a virtual base class, giving the possibility to implement system-specific routines 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*.

α	n	m	m_s	R	α	n	m	m_s	R
0	0	0	1	1	10	0	0	1	
1	0	0	-1		11	0	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	1	3	1	
5	0	1	-1		15	1	3	-1	
6	0	-2	1	3	16	1	-1	1	
7	0	-2	-1		17	1	-1	-1	
8	1	2	1		18	0	1	1	
9	1	2	-1		19	0	1	-1	

Table 8.1: Overview of the mapping between single-particle states α and corresponding quantum numbers n, m, m_s in a harmonic oscillator basis, here for the first four shells. Each state α has been assigned a specific set of quantum numbers with allowed values given by $n = 0, 1, ...,$ $m = 0, \pm 1, \pm 2, \ldots$ and $m_s = \pm\frac{1}{2}$, where the latter has been simplified to $m_s \pm 1$ such that only integers have to be stored.

8.3.2 Class *SPstate*

Another general class 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, e.g. a harmonic oscillator basis, which is a separate class in our program, can then contain an array of those single-particle states, with its size depending on the size of the basis.
Since the algorithm specifying the quantum numbers is system-dependent, we have chosen to put the mapping function, which maps 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 demonstrate how the mapping is performed in the case of 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 presented in chapter 5, a shell number 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 form the lowest shell, each single particle state is assigned an index α, as illustrated in figure 5.1.
We now perform a mapping

$$|\alpha\rangle \to |n, m, m_s\rangle ,$$

such that each index α corresponds to a specific set of quantum numbers $n(\alpha), m(\alpha), m_s(\alpha)$. The quantum number n is the nodal quantum number as given in Eq. (5.5), m denotes the angular momentum quantum number and m_s the spin projection. An example for a two-dimensional quantum dot and the first four shells is given in table 8.1. In order to make our code compatible with the OpenFCI [ref] library generating the interaction elements later on, we have slightly changed the indexing within a shell R.

Listing 8.1: Algorithm performing the mapping $|\alpha\rangle \to |n, m, m_s\rangle$. The array *qnumbers* contains the quantum numbers in the order n, m, m_s. Line 12, for example, then means that we access m_s of the 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     }
```

Studying this table and the shell structure in figure 5.1, we observe a pattern which makes it possible to automize 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, m starts with $m = 1 - R$ and increasing in steps of 2, always first with negative, then with positive sign. That way, one ends up with the algorithm demonstrated in listing 8.1, which performs the mapping for two-dimensional quantum dots and additionally computes the single-particle energies ϵ_i.

As stated above, this algorithm is specific for each considered system. Implementing the function *mapping* as a virtual function, makes it therefore to adopt it to each subclass of the general class *System*.
Moreover, we did not hardcode the quantum numbers n, m, m_s as three integers for each single-particle state, but collected them in the array *qnumbers*, whose usage is demonstrated in listing 8.1. The dimension of this array is variable, such that one 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 in-medium SRG method. This section serves to describe how we implemented the free-space case, which algorithms we used and which computationally 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 will give an overview of the structure of the classes *Hamiltonian* and *Basis*. To demonstrate the data flow and how the classes work together, we will give a detailed description of how the system is set up. Afterwards, we will demonstrate how the solver, in our case the SRG method, is applied to the system. In the course of this, we will describe the class *SRG*, which as solver is independent of the previous 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 computed from 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 contained functions can be subdivided into two classes: One group of functions serves to read in the one-particle and two-particle elements from file, which 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. In this notation, 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 element.
After the elements have been read in, the task of the second class of 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 will be given at a later stage, when we explain how the 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 with the Slater determinant basis for the Hamiltonian, and an array containing all considered single-particle states until the made cut-off. This array contains objects of the type *SPstate*, which themselves contain all quantum numbers, single-particle energy etc. The single-particle states are in our case the ones of a harmonic oscillator basis, but our implementation also takes care of a Hartree-Fock basis, which then is based on the previous harmonic oscillator basis. The only difference with a Hartree-Fock basis is that the one-body and two-body elements (8.1) and (8.2) are this time not directly obtained from the OpenFCI library [ref!], but from a preceding Hartree-Fock calculation.
Since the *Basis*-class is responsible to hold the Slater determinant basis, the contained functions involve several routines for setting up this basis, transforming it to binary representation, choosing the right states for a given channel etc. All this methods will be explained in the following, when we explain how a specific system is set up.

8.4.2 Setting up the system

Setting up the system 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 two particles and four sp-states, one obtains for example 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 state where single-particle state i and j are occupied by a particle and the remaining ones are unoccupied.
In general, there are $\binom{n_{sp}}{N}$ possible Slater determinants and one needs an algorithm to systematically create all combinations. In this thesis, we use 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, one makes sure to include all possibilities. For 3 particles and 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 *Basis*-class. Given an initial occupation scheme, it is used to move the "odometer" one step further.

Listing 8.2: The function *odometer* accepts the N-dimensional array *occ* with the current occupation scheme of the N particles and returns the next one, using the odometer algorithm.

```cpp
void Basis::odometer(ivec& occ, int numpart) {
    int l;
    // Loop over all particles, 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;
        }
    }
}
```

In order to set up the whole Hamiltonian matrix, one theoretically has to store all possible Slater determinants in order to compute the matrix elements. However, the size of this matrix increases rapidly with the number of particles and sp-states, which requires large amounts of memory and CPU time in the subsequent diagonalization.
Moreover, in the case of quantum dots, this matrix would be 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 one is only interested in the ground state energy, it would be an enormous overkill to diagonalize the whole Hamiltonian matrix. In the case of closed-shell systems, that are studied in this thesis, one can assume that the ground state fulfils $M = M_s = 0$ and therefore one has 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 is decreased to $n = 1490$ states and the required memory to about 17MB. Thus the memory requirement has been drastically reduced and the computation is even possible 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 if these ones are fulfilled, the Slater determinant is added to the array which saves the basis states.

In order to make the code as general as possible, the function checking M and M_s is not restricted to only those two quantities, but could also check for further ones. In a more general context, one can have 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 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

$$_{\lambda',\pi'}\langle\alpha| \hat{H} |\beta\rangle_{\lambda,\pi} = 0, \qquad \text{if } \lambda' \neq \lambda,$$

which 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, one could add the parity τ as 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, the Slater determinant basis is not stored 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

$$\begin{pmatrix} |0,1,2,3\rangle \\ |0,1,2,6\rangle \\ |0,1,3,8\rangle \\ \cdots \end{pmatrix} \leftrightarrow \begin{pmatrix} |15\rangle \\ |71\rangle \\ |267\rangle \\ \cdots \end{pmatrix}.$$

On the one hand, this approach saves memory since instead of N integers, just one integer is saved per Slater determinant. On the other hand, it makes the usage of time saving bit manipulation operations possible when the Hamiltonian acts on those basis states, something we will explain at a later stage.
However, in this first approach, this method has a limitation, since an integer is restricted to 32

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* with the quantum numbers of each sp-state, starting with the second index. In our case, *qnumbers* contains the quantum numbers n, m, m_s and if we are interested in the channel with $M = M_s = 0$, the vector *spes_channel* therefore has to contain $(0, 0)$. For other systems with more quantum numbers, one could simply extend the array *qnumbers* and, if those quantum numbers define the channel, also the vector *spes_channel*.

```cpp

// Return whether the occupation scheme "occ" has the correct channel
bool Basis::correct_channel(ivec& occ, int numpart,ivec& spes_channel) {
    ...
    int fixed_channel = spes_channel.size(); // number of quantum numbers to be
         checked is specified by the input vector

    for(int i = 1; i<= fixed_channel; i++){
        sum = 0;
        for( int j = 0; j< numpart; j++){
            sum += singPart[occ(j)].qnumbers[i];
        }

        if(sum != spes_channel(i-1))
            return false;
    }
    return true;
}
```

or 64 bits, depending on the operating system. In order to have the possibility to treat systems with more than $R = 7$ or $R = 10$ shells, respectively, the Slater determinants are therefore saved in the form of *bitsets* [35]. 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.

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 effective 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 even much easier since one simply has to add the corresponding contributions on the diagonal.

Algorithm: Setup of the Hamiltonian matrix (interaction part)

```
Loop over all ket-states (right 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 exists 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 Hamilto-
         nian 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, all Slater determinants are saved as a bit pattern. Acting with $a_i^\dagger a_j^\dagger a_l a_k$ on such a bit pattern means to remove the bits k and l, and to add the 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 $a_i^\dagger a_j^\dagger = -a_j^\dagger a_i^\dagger$. The concrete procedure is therefore as follows:
First, we check whether both bits k and l are occupied, using the the corresponding method of the C++ class *bitset*. If not, no further calculations are necessary since $a_p|\Phi\rangle = a_p a_{q_1}^\dagger a_{q2}^\dagger \dots a_{q_N}^\dagger |0\rangle = 0$ for $p \notin \{q_1, q_2, \dots q_n\}$. If both bits are set, first the bit k, then the bit l is 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 $a_p^\dagger|\Phi\rangle = a_p^\dagger a_{q_1}^\dagger a_{q_2}^\dagger \dots a_{q_N}^\dagger|0\rangle = 0$ for $p \in \{q_1, q_2, \dots q_n\}$. 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 that one. The results are the bra-state $\langle\alpha|$ in bit representation as well as the phase, which one has to multiply with the transition amplitude.

Algorithm to determine $|\alpha\rangle = \pm a_i^\dagger a_j^\dagger a_l a_k |\beta\rangle$

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 is occupied in $|\beta\rangle$.
4. If yes, then return zero, otherwise add first bit j, then i, again following the phase.

8.4.3 Applying the SRG solver

When the Hamiltonian matrix is set up, the SRG method can be used to diagonalize it. Since the flow equations (6.15) represent a set of coupled ordinary differential equations (ODEs), one needs an ODE solver that performs the integration. In this thesis, we use a solver based on the algorithm by Shampine and Gordon [12], provided as C++ version on [11].

ODE algorithm by Shampine/Gordon

In the following we will explain the basic concepts of the Shampine/Gordon ODE algorithm. The code itself, however, is much more advanced than that and involves dealing with discontinuities and stiffness criteria, controls for propagated roundoff errors and detecting requests for high accuracies. This makes it a very powerful tool for the solution of ordinary differential equations, but the code gets 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 [12].

The ODE algorithm by Shampine and Gordon involves three major functions: the core integrator *step*, a method for interpolation *intrp* and a driver *ode*.
Suppose we have to solve a general ODE problem of the form

$$\begin{aligned} y_1'(t) &= f_1(t, y_1(t), y_2(t), \dots, y_n(t)) \\ y_2'(t) &= f_2(t, y_1(t), y_2(t), \dots, y_n(t)) \\ &\vdots \\ y_n'(t) &= f_n(t, y_1(t), y_2(t), \dots, y_n(t)), \end{aligned}$$

with initial conditions $y_1(a), y_2(a), \ldots, y_n(a)$ given. 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 that one with a complete specification of the problem. This involves the concrete equations (8.4), 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 etc., 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, in order for the user to know about success of the integration and 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 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 one only has to provide the function with appropriate input parameters, call the *ode* function and collect the output results in the end.

The concrete equations (8.4) must be supplied as subroutine of the form

$$f(t, y, dy), \tag{8.5}$$

where t is a specific integration point, y an array containing the values $y_i(t), i = 1 \ldots n$ at that point and dy an array with the same dimension as y, holding the derivatives.
In our specific case of SRG, 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.

$$\text{Input: } v \leftarrow \langle pq|\hat{H}|rs\rangle$$
$$\text{Output: } v = v(s).$$

The mathematically most fundamental routine of the algorithm is the routine *step*, which is the basic integrator and advances the solution of the differential equations exactly one step at a time, that is from y_n to $y_{n+1} = y_n + h$. It uses a variable order version of the Adams formulas in a PECE combination with local extrapolation [12], which means that the predictor is of order k and the corrector of order $k + 1$.

In order to use Adams methods most effectively, the algorithm aims at using the largest step size h yielding the requested accuracy. Most ODE codes require the user to guess an initial step size, which can be a source of error if the user does not know how to estimate a suitable value. The ODE algorithm by Shampine/Gordon therefore includes a routine to estimate this value automatically, trying to limit the problem to a minimum of function evaluations, but at the same time maintaining stability.
The determination of the step length is motivated by experimentation done by F.T.Krogh [**?**] and C.W.Gear [38] and based on various error predictions and interpolation. Concerning selection of the right order, the algorithm tends to use low orders since they have 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 one has found an order appropriate to the problem and with lowest possible error.
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 [ref].

With a step size determined by error propagation and stability, it is very likely that the integration points of the algorithm do not coincide with the desired integration output point s. Moreover, results are most accurate if the equations are integrated just beyond that point s and then interpolated. Exactly this task is carried out by the routine *intrp*, performing such an interpolation to obtain the solution at specified output points.

Class *SRG*

Everything concerning the SRG method itself, i.e. the flow of the Hamiltonian, is 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 main data structure this class holds, is an object 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 [11] to our needs, such that it fits our system of data storage and the derivative function. The modified functions are now part of 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 derivative equations (8.4). In our case, 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 above, we have a separate function, called *oscillator_ derivative*, that computes the derivatives at a certain point of integration.
Concerning the integration interval, one theoretically has to integrate down to $\lambda = 0$. Prac-

Listing 8.4: The function *oscillator_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);
}
```

tically, one can stop the integration after 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 and detect whether the change of E_0 lies within the given bounds. On the one hand, this step length should be chosen small enough that integration is not performed unnecessarily close to the zero point, because the required CPU time increases drastically with the length of integration, as we will see later on. On the other hand, it should be chosen large enough that differences of E_0 exceed the error tolerance if convergence has not been reached yet and the integration has not to stop too often for interpolation.
We solved this problem 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 there is still no convergence, 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. In the case of free-space SRG, we will look at the simple generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$ and Wegner's canonical choice $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$. The only computational difference between both generators occurs when computing the derivatives. Therefore, we designed the derivative-function as a kind of wrapper that chooses the generator-specific routine from an array containing the routines 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 $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$ and the second one for Wegner's choice $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$, are presented in listings (8.6) and (8.7). Since there is only a minimal 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 by the class *System*: If we want to access the kth element of the array HO, containing the diagonal, non-interacting elements, 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 efficiency reasons, 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 is a great benefit for

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 *oscillator_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];
    ...
}
```

Listing 8.6: Derivative function for generator $\hat{\eta}_1$. The function receives the current interaction elements at *k_lambda* in the array *v* and returns the corresponding derivatives in the array *dv*. The calculation is performed according to Eq. (6.15). Via the object *Sys* of type *System*, the Hamiltonian can effectively be accessed.

```
void SRG::deriv_eta1(double k_lambda, double** v, double** dv) {

    ...

    k3 = k_lambda * k_lambda*k_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 for generator $\hat{\eta}_1$, receiving the current interaction elements at *k_lambda* in the array *v* and returning the corresponding derivatives in the array *dv*. The calculation is performed according to Eq. (6.19).

```
void SRG::deriv_eta2(double k_lambda, double** v, double** dv) {

    ...

    k3 = k_lambda * k_lambda*k_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;
}
```

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 [39].

8.5 Implementation specific for in-medium SRG

Having explained the implementation for the free-space case, we will now proceed in a similar way for the in-medium implementation. This section will present all classes that are specific for the in-medium case, 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 also 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), meaning that one basis state is determined by two single-particle states (sp-states). For this idea, we have been inspired by [40] and [41], since we made the observation that from a computational point of view, many routines of the SRG method resemble ones used in a Coupled Cluster implementation. 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 the matrix elements, enabling us to make us of the obtained 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 stage, we will demonstrate how this array is needed when setting up the tp-basis.
The core of the whole class *Basis* are three arrays holding this 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 quantum number of the two particles, respectively. For each value of the so-called *channel* λ, 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, 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 by [40] and demonstrated for the case of one particle and one hole in listing 8.9. At an earlier stage of our program, the maximal M-value for two particles is computed as

$$M_{max} = 2 \cdot (R - 1)$$

and with

$$\begin{aligned} M &= 0, \pm 1, \pm 2, \dots, \pm M_{max} \\ M_s &= -1, 0, +1 \end{aligned}$$

the total number of channels λ is

$$lbd\_dim = 2 \cdot M_{max} \cdot 3 + 3.$$

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, the second index j to a hole. Using the quantum numbers stored in the array *singPart*, containing 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.

Those three functions establishing the tp-basis constitute one group of functions in our class *Basis*. Another group of functions is responsible to map between a certain particle and/or hole combination and the corresponding index in the two-particle basis. This will be needed later 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. In functions of the class *Hamiltonian*, using the tp-basis, we often have to loop over all basis states, which therefore should be as effective as possible. Studying Fig.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, here for the *ph*-basis, is given in listing 8.11.

Listing 8.9: Setting up the two-particle basis for the case of one particle and one hole. A detailed explanation can be found 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: in our case Ms/2= {-1,0,1}
}
```

Listing 8.11: Getting the first and last occupied channel λ, here an extract for the *ph*-basis. The boundaries are saved in an integer-matrix, which is returned by the function.

```
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;

    ...
}
```

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 [40, 41] and save the elements in a special, effective way.

First of all, we aim to make use of the symmetries of the Hamiltonian

$$\langle pq||rs\rangle = -\langle qp||rs\rangle = -\langle pq||sr\rangle = \langle qp||sr\rangle, \tag{8.6}$$

enabling us just to store one of four possibilities, which 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. Therefore 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:

$$|pq\rangle \leftrightarrow |M, M_s\rangle.$$

Considering 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, showing the purpose of our structure of the class *Basis*. That way, we store the Hamiltonian in block-diagonal matrices, one block for each channel λ. To make us of the tp-basis and, furthermore, since the flow equations (6.29)-(6.31) depend on whether the indices correspond to particle or hole states, we differentiate between the following combinations

$$\begin{aligned} v_{hhhh} &\mathrel{\widehat{=}} \langle ij||kl\rangle, \\ v_{phhh} &\mathrel{\widehat{=}} \langle aj||kl\rangle, \\ v_{pphh} &\mathrel{\widehat{=}} \langle ab||kl\rangle, \\ v_{phph} &\mathrel{\widehat{=}} \langle aj||cl\rangle, \\ v_{ppph} &\mathrel{\widehat{=}} \langle ab||cl\rangle, \\ v_{pppp} &\mathrel{\widehat{=}} \langle ab||cd\rangle, \end{aligned} \tag{8.7}$$

where 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), obtaining

$$\begin{aligned} &v_{hhhh}, \\ &v_{phhh} = -v_{hphh} = -v_{hhhp} = v_{hhph}, \\ &v_{pphh} = v_{hhpp}, \\ &v_{phph} = -v_{hpph} = -v_{phhp} = v_{hphp}, \\ &v_{ppph} = -v_{pphp} = -v_{hppp} = v_{phpp}, \\ &v_{pppp}, \end{aligned} \tag{8.8}$$

such that 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 the 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. (6.28), 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 $f_{hp} = f_{ph}$.

Listing 8.12: Access to f-elements via the function *get_f_elem*. The main idea is to determine which of the three possible matrices f_{hh}, f_{ph} and f_{pp} has to be used.

```cpp
double Hamiltonian::get_f_elem(int p, int q, std::vector<arma::mat>& mat_elems)
     const {

    // f_hh
    if ((p < Bas->hole_states) && (q < Bas->hole_states))
        return mat_elems[1](p, q);

    // f_ph
    if ((p >= Bas->hole_states) && (q < Bas->hole_states))
        return mat_elems[2](p - Bas->hole_states, q);

    // f_hp
    if ((p < Bas->hole_states) && (q >= Bas->hole_states))
        return mat_elems[2](q - Bas->hole_states, p);

    // f_pp
    return mat_elems[3](p - Bas->hole_states, q - Bas->hole_states);

}
```

In total, the complete Hamiltonian is stored in the array *mat_elems*, whose items are matrices, handled by the Armadillo library.

```cpp
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 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 1.2GB of memory. The required memory for our array, however, is only about 76MB. 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* can be found in listing 8.13. To obtain a specific interaction element v_{pqrs}, the function receives the four sp-state indices p, q, r, s. Inside the function, we first check whether the quantum

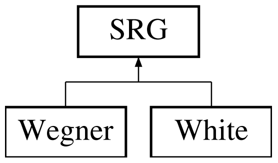


Figure 8.2: The class *SRG* is a virtual base class and needs to be extended by subclasses containing the specific expressions for the flow equations, depending on the generator $\hat{\eta}$.

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. Second (lines 15-23) we bring the indices into right order, since, as explained before, we only store one of the possible combinations for the *hh*- and *pp*-basis. 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 access the matrix elements of our array *mat_elems*. However, 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 the *Basis*-class, yielding the index in the tp-basis and, finally, the element can be obtained. The many *if-else* statements in listing 8.13 might look a bit clumsy, but enabled 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 demonstrated in 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 remembering the change of sign due to the ordering $|p_2p_1\rangle \rightarrow |p_1p_2\rangle$, 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 ground state energy E_0 and the f-elements have to be computed according to Eqs. (6.27) and (6.28).

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 (6.29)-(6.31). Since the concrete terms for these flow equations depend on the generator $\hat{\eta}$, we designed the class *SRG* as a 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, depending on $\hat{\eta}$. This procedure allows us to make use of the benefits that object-orientation in C++ offers: On the one hand, it gets uncomplicated to switch between different generators, on the other hand the program can easily be extended by further generators. In our case, 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

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 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);}
         }
     }
}
```

system-specific information, as well as an array called *eta*, where the values of the generator are stored during the integration. The size and structure of this array depend on the analytic expression of the 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 case, and concerning integration limits, step length etc., we therefore refer to subsection 8.4.3.

Class *Wegner*

The class *Wegner* is the subclass of *SRG* which is designed for Wegner's canonical generator $\hat{\eta} = \left[\hat{H}^{\rm d}, \hat{H}^{\rm od}\right]$. 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 (6.29)-(6.31) can be 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 (6.29)-(6.31) straightforwardly, and each time an element of $\hat{\eta}$ is needed, that one is computed according to Eqs. (6.37) and (6.38). 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 for each integration step, is to compute first all elements of $\hat{\eta}$, store them, and then continue with the derivatives using these elements.

To store the $\hat{\eta}$-elements most efficiently, with an uncomplicated way of looping over and accessing them, we decided on the same structure as for the matrix elements of the Hamiltonian. The reasons are as follows:
First of all, the organization of the elements is similar, with the one- and two-body elements of $\hat{\eta}$ 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, based on the Hamiltonian by $\hat{\eta} = \left[\hat{H}^{\rm d}, \hat{H}^{\rm od}\right]$, 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 $\hat{\eta}_s^\dagger = -\hat{\eta}_s$, 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 Eq. (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 functions concerning 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. (6.37) and (6.38), 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. (6.38), 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. (6.38).

In order to test our implementation for $N = 2$ particles against the exact result, and for easy extension from IM-SRG(2) to IM-SRG(3), we have opened up the possibility of including three-body terms of $\hat{\eta}$. The function

```
double SRG::get_eta3(int p, int q, int r, int s, int t, int u, std::vector<arma
    ::mat>& mat_elems)
```

computes the elements $\eta^{(3)}_{pqrstu}$ following the commutation relations of Appendix A:

$$\eta^{(3)}_{pqrstu} = P(pq/r)P(s/tu)\sum_w \left(v^d_{pqsw}v_{wrtu} - v_{pqsw}v^d_{wrtu}\right). \tag{8.12}$$

In order to avoid summing over terms that are known to be zero in advance, we have made

Listing 8.14: Excerpt of the function *E0_deriv:*, which evaluates Eq. (6.29). Here the second term of Eq. (6.29) is demonstrated. For detailed explanation, 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;
```

use of the definitions of Eq. (6.33) and simplified the expression to

$$\begin{aligned}\eta^{(3)}_{pqrstu} = P(pq/r)P(s/tu)\left(v_{pqsq}v_{qrtu}\delta_{ps} + v_{pqsp}v_{prtu}\delta_{qs}\right.\\ \left. - v_{pqst}v_{trtu}\delta_{ru} - v_{pqsu}v_{urtu}\delta_{rt}\right). \end{aligned} \tag{8.13}$$

Computing the derivatives The flow equations are in principle computed according to Eqs. (6.29)-(6.31), 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, but 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. (6.29),

$$\frac{1}{2}\sum_{ijab}\eta^{(2)}_{ijab}v_{abij}, \tag{8.14}$$

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 (8.14) 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. Since 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 *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 (8.14), 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 (8.14), but for each single term in the flow equations (6.29)-(6.31). This results often in splitting sums up 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 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.

Loop terms for IM-SRG(2/3) As explained for the implementation of the generator $\hat{\eta}$, our code has the possibility of including the loop terms of three-body interactions. With those loop terms we mean effective two-body terms of the form

$$\langle pq||rs\rangle = \sum_i \langle pqi|v^{(3)}|rsi\rangle,$$

and effective one-body terms of the form

$$f_{pq} = \frac{1}{2}\sum_{ij} \langle pij|v^{(3)}|qij\rangle,$$

where i and j sum over all hole states. To optionally include those terms, our implementation of the flow equations includes segments like listing 8.15, which gives the example for the derivative of the ground state energy E_0. The optional portions of the code are surrounded by a *#if* directive, determining during compile-time whether those lines of code are included or not.

Since our model of the Hamiltonian does not include three-body terms W_{pqrstu} itself and we do not consider induced three-body interactions in IM-SRG(2/3), either, the rather complex Eq. (6.24) reduces to the simple contribution

$$\frac{d}{ds}v^{(3)}_{pqrstu} = \sum_v P(pq/r)P(s/tu)\left(\eta^{(2)}_{pqsv}v_{vrtu} - v_{pqsv}\eta^{(2)}_{vrtu}\right) \tag{8.15}$$

To stress the difference between IM-SRG(2/3) and IM-SRG(3) once more, we remind that IM-SRG(3) would require to store and consider all induced three-body interactions, involving the complete Eqs. (6.21)-(6.24), whereas IM-SRG(2/3) simply extends the second order flow equations (6.29)-(6.31) by adding the loop terms to the f- and v-elements.

Improving efficiency Compared to one of the first versions of our code, where we implemented the flow equations absolutely straightforwardly, we have obtained a speed-up 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: Optional inclusion of loop terms corresponding to three-body interactions, here the contribution to dE_0/ds.

```
#if SRG23
    for (int lbd = lbd_limits(0, 0); lbd <= lbd_limits(0, 1); lbd++)
        for (int ij = 0; ij < Sys->H->Bas->hh_basis[lbd].size(); ij++) {
            i = Sys->H->Bas->hh_basis[lbd][ij](0);
            j = Sys->H->Bas->hh_basis[lbd][ij](1);
            for (int k = 0; k < hole_states; k++)
                contr += get_dv3(i, j, k, i, j, k, mat_elems);
        }
    d_ret += (1.0 / 3.0) * contr;
#endif
```

Listing 8.16: Term (8.16) 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 (6.29)-(6.31) as possible in terms of matrix-matrix multiplication. This often effectively reduces the 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

$$\sum_r \left(\eta^{(1)}_{pr} f_{rq} + \eta^{(1)}_{qr} f_{rp}\right) \tag{8.16}$$

of Eq. (6.30) for the matrix of form f_{ph}. In a first, straightforward approach, we expressed the term as shown in listing 8.16. 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 getter functions to a minimum and access elements, as often as possible, directly. We took this step of course not only in Eq. (6.30), 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 (8.16) in listing 8.17.
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 the case that r corresponds to a particle in term (8.16), the element $\eta^{(1)}_{qr}$ needs to be transformed from the form $\eta^{(1)}_{hp}$ to

Listing 8.17: Improvement of implementation in listing 8.16 by accessing all elements directly, without getter functions.

```
1  for(int p_ind = 0; p_ind< part_states ; p_ind++){
2          p = p_ind +hole_states;
3          for(int q = 0; q < hole_states; q++){
4  
5              // r is hole state
6              for(int r = 0; r < hole_states; r++)
7                  dv[2](p_ind,q) += eta[2](p_ind,r]*mat_elems[1](r,q)
8                          + eta[1](q,r)*mat_elems[2](p_ind,r);
9  
10             // r is particle state
11             for(int r = hole_states; r < all_states; r++)
12                 r_ind = r - hole_states; // r as index in matrices
13                 dv[2](p_ind,q) += eta[3](p_ind,r_ind)*mat_elems[2](r_ind,q)
14                         - eta[2](r_ind,q)*mat_elems[3](r_ind,p_ind);
15         ...
16         }
17 }
```

Listing 8.18: Further optimizing expression (8.16) by using matrix-matrix multiplication. The function *strans* is used from 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];
```

$\eta_{hp}^{(1)} = -\eta_{ph}^{(1)}$. This use is demonstrated in line 15 of listing 8.17, where also the sign has been changed correspondingly.
As mentioned before, some expressions, like our example term (8.16), even admit to be optimized further by using matrix-matrix multiplications. Considering basic matrix properties for the elements of f_{ph}, the contribution of term (8.16) can be rewritten as

$$\sum_r \left(\eta_{pr}^{(1)} f_{rq} + \eta_{qr}^{(1)} f_{rp}\right) \widehat{=} \eta_{ph}^{(1)} \cdot f_{hh} + \left(\eta_{hh}^{(1)} \cdot f_{ph}\right)^T \quad \text{(for hole states } r\text{)}$$

$$+ \eta_{pp}^{(1)} \cdot f_{ph} - \left(\eta_{ph}^T \cdot f_{pp},\right)^T \quad \text{(for particle states } r\text{)}$$

where '$\cdot$' denotes matrix-matrix multiplication and T denotes the matrix transpose. The corresponding implementation can be found in listing 8.18. Matrices f_{hh} and f_{pp} are treated analogously.

Apart from this, our code involves many smaller optimizations for special expressions. One further example are cases where using getter functions is unavoidable but two-body elements $\eta_{pqrs}^{(2)}$ 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

Listing 8.19: Effective way for adding the two contributions of Eq. (8.18). The variables *v_comb1* and *v_comb2* are used to access the right matrix elements, *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_ind* conveys the correct sign. The same explanations hold for the variables of the other function, starting with prefix *eps_*.

```
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, eps_comb1,
            eps_comb2, eps_ind, eps_sign);

        if (abs(v_sign) > 0 && abs(eps_sign) > 0)
                dv[dv_ind](ai, kl) += eps_sign * eta[eps_ind](eps_comb1,
                    eps_comb2) * v_sign * mat_elems[v_ind](v_comb1, v_comb2);

        }
```

in the array, and we store $\hat{\eta}$ and $\hat{H}$ in arrays of the same structure, it makes sense to avoid finding this 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& eps_ind, int& eps_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. (6.31),

$$-\sum_{ia} (1-P_{ia})(1-P_{pq})(1-P_{rs})\,\eta^{(2)}_{aqis} v_{ipar}, \tag{8.17}$$

which when multiplied out, contains the contribution

$$-\eta^{(2)}_{aqis} v_{ipar} + v^{(2)}_{aqis} \eta_{ipar}. \tag{8.18}$$

Instead of searching for the index corresponding to *aqis* and *ipar* twice, we do it only for the first term by using those two functions, demonstrated in listing 8.19.

Class *White*

The class *White* is the other subclass of *SRG* and used for White's generator (6.39). 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 (6.29)-(6.31) 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 will show how we adapted the data structures and functions explained for Wegner's generator to White's generator. We will 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 6.4.1, 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 8.20 shows how we implemented Eqs. (6.41) and (6.43) 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 8.20, although the code gets harder to read than the excerpt in listing 8.21, 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 8.21:
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 index in the tp-basis.

Computing the derivatives The derivatives (6.29)-(6.31) are independent of the used generator, which means that we principally could use the same implementation as for Wegner's generator. However, in this case we could 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 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. (6.31), the

Listing 8.20: 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));
        }
    }

#if SRG23
                ... // three-body terms
#endif

    ////////////////////////////////////////////////////////////////////
    //                    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

#if SRG23
                ... // three-body terms
#endif
            }
        }
}
```

Listing 8.21: 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));
          }
     }
```

following expression has to be evaluated:

$$
\begin{aligned}
\frac{dv_{klmn}}{ds} &= \sum_t \left(\eta^{(1)}_{kt} v_{tlmn} - \eta^{(1)}_{lt} v_{tkmn} - f_{kt}\eta^{(2)}_{tlmn} + f_{lt}\eta^{(2)}_{tkmn}\right) \\
&- \sum_t \left(\eta^{(1)}_{tm} v_{kltn} - \eta^{(1)}_{tn} v_{kltm} - f_{tm}\eta^{(2)}_{kltn} + f_{tn}\eta^{(2)}_{kltm}\right) \\
&+ \frac{1}{2}\sum_{ab}\left(\eta^{(2)}_{klab} v_{abmn} - v_{klab}\eta^{(2)}_{abmn}\right) - \frac{1}{2}\sum_{ij}\left(\eta^{(2)}_{klij} v_{ijmn} - v_{klij}\eta^{(2)}_{ijmn}\right) \\
&- \sum_{ia}\Big(\eta^{(2)}_{alin} v_{ikam} - \eta^{(2)}_{akin} v_{ilam} - \eta^{(2)}_{alim} v_{ikan} + \eta^{(2)}_{akim} v_{ilan} \\
&- \eta^{(2)}_{ilan} v_{akim} + \eta^{(2)}_{ikan} v_{alim} + \eta^{(2)}_{ilam} v_{akin} - \eta^{(2)}_{ikam} v_{alin}\Big) . \qquad (8.19)
\end{aligned}
$$

Eliminating all terms containing $\hat{\eta}$-elements of form $\eta^{(1)}_{hh}, \eta^{(1)}_{pp}, \eta^{(2)}_{hhhh}, \eta^{(2)}_{phhh}, \eta^{(2)}_{phph}, \eta^{(2)}_{pppph}$ and $\eta^{(2)}_{pppp}$, Eq. (8.19) gets simplified to

$$
\begin{aligned}
\frac{dv_{klmn}}{ds} &= \sum_a \left(\eta^{(1)}_{ka} v_{almn} - \eta^{(1)}_{la} v_{akmn}\right) - \sum_a \left(\eta^{(1)}_{am} v_{kltn} - \eta^{(1)}_{an} v_{kltm}\right) \\
&+ \frac{1}{2}\sum_{ab}\left(\eta^{(2)}_{klab} v_{abmn} - v_{klab}\eta^{(2)}_{abmn}\right) , \qquad (8.20)
\end{aligned}
$$

which 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 (6.29)-(6.29) completely generally without this differentiation.
Apart from these 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 an own 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 the in-medium SRG case, such that a direct use of the transformed elements is possible. However, the class can without any problems be used for the 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 code of Simen Kvaal [37].
The transformation occurs in two steps: First, we perform a HF calculation based on the convergence of the HF energy. This yields the coefficient matrix C, which is 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 routine, which takes as input an untransformed single-particle basis and 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 he 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, this 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 8.3. 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 only required to be $N \times n_{sp}$, 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 $n_{sp} \times n_{sp}$. At the beginning of the

```
                         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 eigenvalues 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 8.3: Summary of the Hartree-Fock algorithm.

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 [40, 41], we store $\hat{h}^{HF}$ in block-form: Similar to storing the Hamiltonian in our SRG part, we use that 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 [40], 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 [40].
Having computed all blocks of $\hat{h}^{HF}$ using the C-matrix, the 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 diagonalizing.
As summarized in the algorithm in figure 8.3, we let those eigenvalues 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

$$E\left[\Phi_0^{HF}\right] = \sum_i \sum_\alpha C_{i\alpha}^2 \epsilon_\alpha + \frac{1}{2} \sum_{ij} \sum_{\alpha\beta\gamma\delta} C_{i\alpha}^* C_{j\beta}^* C_{i\gamma} C_{j\delta} \langle\alpha\beta||\gamma\delta\rangle, \tag{8.21}$$

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 try to rewrite as many sums as possible into matrix operations, similar to the

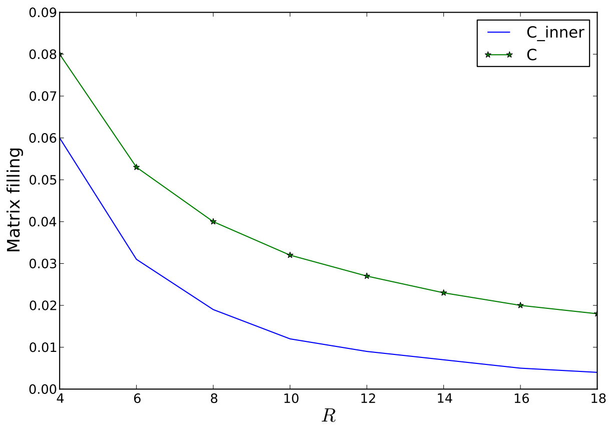


Figure 8.4: Sparsity of the coefficient matrices for systems with $N = 2$ particles. The filling of the matrices is given by the percentage of non-zero elements.

optimizations of the SRG code. Inspired by the code of [41], we define a new matrix

$$C^i_{pq} = \sum_k C_{kp} C_{kq}, \tag{8.22}$$

which can be expressed by a matrix-matrix multiplication the following way:

$$C^i = C^T_{h\times all} \cdot C_{h\times all},$$

where '$\cdot$' denotes matrix multiplication and T the matrix transpose. The matrix $C_{h\times all}$ has dimension $N \times n_{sp}$ and contains the first N columns of the C-matrix, where N is the number of particles. With the matrix C^i, Eq. (7.4) can be simplified to

$$E\left[\Phi_0^{HF}\right] = \sum_{\alpha\beta} C^i_{\alpha\beta}\langle\alpha|\hat{h}_0|\beta\rangle + \frac{1}{2}\sum_{\alpha\beta\gamma\delta} C^i_{\alpha\gamma} C^i_{\beta\delta}\langle\alpha\beta||\gamma\delta\rangle. \tag{8.23}$$

That way, the first term is reduced from $\mathcal{O}(Nn_{sp}^2)$ to $\mathcal{O}(n_{sp}^2)$ and the second one from $\mathcal{O}(N^2 n_{sp}^4)$ to $\mathcal{O}(n_{sp}^4)$, where n_{sp} denotes the number of single-particle states. 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 8.4. When updating the HF energy according to Eq. (8.23), 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 $\alpha, \beta, \gamma, \delta$, this allows for the second term of Eq. (8.23) to sum over only those combinations α, γ and β, δ, such that the corresponding matrix elements $C^i_{\alpha\gamma}$ and $C^i_{\beta\delta}$ are different from zero. The respective implementation can be found in listing 8.22.

Listing 8.22: Calculation of the two-body contribution to the Hartree-Fock energy as given in Eq. (8.23). To minimize the number of terms, the array *cCombs* contains those combinations of indices p and q such that 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;
```

8.6.2 Transformation of basis

The coefficient matrix C from the Hartree-Fock calculation is used to transform the one- and two-body elements according to

$$\langle p|\hat{h}^{(0)}|q\rangle \leftarrow \sum_{\alpha\beta} C^*_{p\alpha} C_{q\beta} \langle\alpha|\hat{h}^{(0)}|\beta\rangle \tag{8.24}$$

$$\langle pq||rs\rangle \leftarrow \sum_{\alpha\beta\gamma\delta} C^*_{p\alpha} C^*_{q\beta} C_{r\gamma} C_{s\delta} \langle\alpha\beta||\gamma\delta\rangle \tag{8.25}$$

Improving efficiency Equations (8.24) and (8.25) are computationally very expensive and should be optimized to keep the overall code efficient. Regarding Eq. (8.24), 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

$$\langle p|\hat{h}^{(0)}|q\rangle \leftarrow \sum_{\alpha} C^2_{p\alpha}\epsilon_\alpha, \tag{8.26}$$

where $\epsilon_\alpha = \langle\alpha|\hat{h}^{(0)}|\alpha\rangle$ is 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. (8.25). This equations demands theoretically looping over eight indices when all elements $\langle pq||rs\rangle$ are to be computed. This corresponds to a number of floating-point operations of order $\mathcal{O}(n^8_{sp})$, which should definitely be avoided.
As for the SRG method, we therefore make 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

Listing 8.23: Instead of summing over all arrangements of indices as in Eq. (8.25), 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. (8.25) is implemented similar to the example of v_{hhhh}, which is given in listing 8.24.

```
// Transformation of two-body elements
    transform_vhhhh();
    transform_vphhh();
    transform_vpphh();
    transform_vphph();
    transform_vppph();
    transform_vpppp();
```

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 in our class *Hamiltonian* for the in-medium case, and we therefore refer to section 8.5 for further details. The function responsible for the basis transformation then calls six routines, one for each of the possible combinations (8.7), see listing 8.23.

The second optimization, which can be observed in listing 8.24, is that analogous to the update of the HF energy in Eq. 8.23, most of the contributions are zero due to the great sparsity of the C-matrix. For not looping unnecessarily over most of the combinations not giving a contribution anyway, we make again use of an array that saves those indices i and j corresponding to non-zero matrix elements C_{ij}. The typical implementation, here given for elements of form v_{hhhh}, i.e. where all four indices correspond to hole states, can be found in listing 8.24. 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*.

8.7 Implementation of Diffusion Monte Carlo

We developed the Diffusion Monte Carlo code independently of the SRG code, as part of the project in [42]. 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.

Listing 8.24: Transformation of two-body elements of the form v_{hhhh}, i.e. 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;
            }
        }
    }
```

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 8.5: The main algorithm of Variational Monte Carlo calculations.

	Metropolis	Metropolis_Hastings
Trial position	$\chi \in [-1, 1]$	$\chi \in gaussian()$
		$\mathbf{F}(\mathbf{R}_i) = 2\frac{1}{\Psi_T}\nabla_i\Psi_T$
	$\mathbf{R}' = \mathbf{R} + \chi\delta$	$\mathbf{R}' = \mathbf{R} + D\Delta t\mathbf{F}\left(\mathbf{R}\right) + \chi$
Acc. ratio		$G\left(\mathbf{R}, \mathbf{R}'; \Delta t\right) = e^{-\frac{\mathbf{R}' - \mathbf{R} - D\Delta t\mathbf{F}(\mathbf{R}))^2}{4D\Delta t}}$
	$\mathrm{R} = \left\lvert\frac{\psi_i}{\psi_j}\right\rvert^2$	$R = \frac{G(\mathbf{R}',\mathbf{R};\Delta t)}{G(\mathbf{R},\mathbf{R}';\Delta t)}\left\lvert\frac{\psi_i}{\psi_j}\right\rvert^2$

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.

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 $\{\alpha, \beta\}$ 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. Therefore 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 a class called 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 8.5. The main 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 *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, the unperturbed and interaction part of the Hamiltonian are computed separately. If the interaction is switched on, one uses both contributions to compute the local energy, otherwise one simply leaves out the one from the interaction part.

Moreover, in the class *Kinetic*, one can choose whether the Laplacian in the term $\left(-\frac{1}{2}\nabla^2\right)$ 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

$$\frac{1}{\Psi_T}\nabla_i^2\Psi_T = \frac{\Psi_T\left(\mathbf{r}_i + h\right) + \Psi_T\left(\mathbf{r}_i - h\right) - 2\Psi_T\left(\mathbf{r}_i\right)}{h^2\Psi_T\left(\mathbf{r}_i\right)},$$

where $\mathbf{r}_i$ is the position vector of the ith particle.

The analytical expression involves some more mathematics. First of all,

$$\begin{aligned}\frac{\nabla_i^2\Psi_T}{\Psi_T} &= \frac{\nabla_i^2\left(|D_\uparrow|\,|D_\downarrow|\,J\right)}{|D_\uparrow|\,|D_\downarrow|\,J} \\ &= \frac{\nabla_i^2\,|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i^2\,|D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i^2 J}{J} + 2\frac{\left(\nabla_i\,|D_\uparrow|\right)\left(\nabla_i\,|D_\downarrow|\right)}{|D_\uparrow|\,|D_\downarrow|} \\ &\quad + 2\frac{\left(\nabla_i\,|D_\uparrow|\right)\left(\nabla_i J\right)}{|D_\uparrow|\,J} + 2\frac{\left(\nabla_i\,|D_\downarrow|\right)\left(\nabla_i J\right)}{|D_\downarrow|\,J}.\end{aligned}$$

Since only one of the 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 taken from [32]. For the Slater determinant, we have

$$\frac{\nabla_i|D|}{|D|} = \sum_{j=1}^{N}\left(\nabla_i\phi_j\left(\mathbf{r}_i\right)\right)D_{ji}^{-1}, \tag{8.27}$$

where the gradient of the single particle orbital $\phi_j\left(\mathbf{r}_i\right)$ is

$$\nabla\phi_{n_x,n_y} = \left\{\left(2n_x\sqrt{\omega\alpha}\frac{H_{n_x-1}}{H_{n_x}} - x\omega\alpha\right)\vec{e}_x + \left(2n_x\sqrt{\omega\alpha}\frac{H_{n_x-1}}{H_{n_x}} - x\omega\alpha\right)\vec{e}_y\right\}\phi_{n_x,n_y}.$$

The gradient of the Jastrow factor is straightforward,

$$\frac{1}{J}\frac{\partial J}{\partial x_i} = \sum_{k=1,k\neq i}^{N}\frac{a_{ki}(x_i - x_k)}{r_{ki}(1+\beta r_{ki})^2}. \tag{8.28}$$

Note that a_{ki} is written as matrix element. Since calculating the matrix a for every run in the loop would request a lot of CPU time, the elements are specified only in the beginning and stored in a matrix once and for all.
The Laplacians are as follows:

$$\frac{\nabla_i^2|D|}{|D|} = \sum_{j=1}^{N}\left(\nabla_i^2\phi_j\left(\mathbf{r}_i\right)\right)D_{ji}^{-1}, \tag{8.29}$$

where the Laplacian of the single-particle orbitals is

$$\begin{aligned}\nabla^2 \phi_{n_x,n_y} =& \omega\alpha\phi_{n_x,n_y}\Bigg(4n_x(n_x-1)\frac{H_{n_x-2}}{H_{n_x}} \\ &+4n_y(n_y-1)\frac{H_{n_y-2}}{H_{n_y}} + \omega\alpha\left(x^2+y^2\right) \\ &-4\sqrt{\omega\alpha}n_x x\frac{H_{n_x-1}}{H_{n_x}} - 4\sqrt{\omega\alpha}n_y y\frac{H_{n_y-1}}{H_{n_y}} - 2\Bigg).\end{aligned}$$

For the Jastrow factor, we have

$$\frac{\nabla_i^2 J}{J} = \left(\frac{\nabla_i J}{J}\right)^2 + \sum_{k=1,k\neq i}^{N} \frac{a_{ki}(1-\beta r_{ik})}{r_{ki}(1+\beta r_{ki})^2}, \tag{8.30}$$

with the already stated expression for the gradient.

Implementation of the wave function

As stated before, our wave function has the structure

$$\Psi_T(\alpha,\beta) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,J(\beta). \tag{8.31}$$

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 containing the single-particle orbitals and a Jastrow factor. If one is interested in how good the pure Slater determinant without correlation factor approximates the true wave function, one simply leaves out the Jastrow object. On the other hand, this procedure opens up the possibility to create several subclasses with different forms of the correlation factor that can be studied.

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 [32]

$$R_D = \frac{|D^{new}|}{|D^{old}|} = \sum_{j=1}^{n} \phi_j\left(\mathbf{r}_{i,new}\right)\left(D_{ji}^{old}\right)^{-1}.$$

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,

$$D = [D_\uparrow D_\downarrow] \qquad D^{-1} = \left[D_\uparrow^{-1} D_\downarrow^{-1}\right].$$

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.

That way, the correct matrix is accessed automatically.
To update the inverse of the Slater matrix after 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,

$$\frac{\Psi_{new}}{\Psi_{old}} = R_D R_J.$$

One additional remark should be made concerning our implementation of Slater determinants. In this thesis, only closed shell systems are considered, which means systems where all single-particle states up to a given energy are occupied. To compute the Slater determinant, the single-particle states depicted in Fig. 5.1 have to be accessed and therefore brought into a specific order. Tables 8.3 and 8.4 show how we implemented this:
Each single-particle state is assigned an index j, which basically is 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, because each level j can additionally be occupied by a particle with spin down.
That way, the number of particles can easily be varied from run to run: One simply divides the particles in two halves and this determines how many levels j are needed.

Quantum Force

As derived above, the expression for the quantum force is

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

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.

To split it up into Slater and Jastrow part, we rewrite

$$\begin{aligned}\mathbf{F}(\mathbf{R}_i) &= 2\frac{\nabla_i \left(|D_\uparrow|\,|D_\downarrow|\,J\right)}{|D_\uparrow|\,|D_\downarrow|\,J} \\ &= 2\left(\frac{\nabla_i\,|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla_i\,|D_\downarrow|}{|D_\downarrow|} + \frac{\nabla_i J}{J}\right).\end{aligned}$$

Since particle i is only contained in one of the Slater determinants (spin up or spin down), the expression simplifies to

$$\mathbf{F}(\mathbf{R}_i) = 2\left(\frac{\nabla_i|D|}{|D|} + \frac{\nabla_i J}{J}\right),$$

where $|D|$ is the determinant with the considered spin.

Optimization: Storing positions and distances

Since a lot of the calculations involve 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 much more often used and thus yields a better efficiency.
The relative differences between all particles are stored 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 efficient feature of this matrix is that when moving one particle at a time, we have to update only parts of the matrix (one row and one column). This means that we can reuse those elements not containing the particle from the previous update of the matrix.

DFP algorithm

To implement the DFP algorithm, we use the function *dfpmin* from [43], which is a function that utilizes the DFP algorithm to find the minimum of a function f.

As discussed above, 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 is given by

$$\frac{\partial \langle E \rangle}{\partial \alpha_i} = \left\langle \frac{\Psi_i}{\Psi} E_L + \frac{H\Psi_i}{\Psi} - 2\bar{E}\frac{\Psi_i}{\Psi} \right\rangle \tag{8.32}$$

$$= 2\left\langle \frac{\Psi_i}{\Psi}(E_L - \bar{E} \right\rangle \qquad \text{(by Hermiticity)} \tag{8.33}$$

$$= 2\left\langle \frac{\Psi_{T,i}}{\Psi_T} E_L \right\rangle - 2\left\langle \frac{\Psi_{T,i}}{\Psi_T} \right\rangle \langle E \rangle, \tag{8.34}$$

with the definition $\Psi_{T,i} = \partial \Psi_T / \partial \alpha_i$.
Not to run the VMC machinery two times, one time to get the energy and another time to get the gradients of Eq. (8.34), we compute and store the latter ones at the same time the energy is computed.

For computing the derivative of the wave function 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,

$$\frac{1}{\Psi_T}\frac{\partial \Psi_T}{\partial \beta} = \frac{\Psi_T(\alpha, \beta + h) - \Psi_T(\alpha, \beta - h)}{2h\Psi_T} + \mathcal{O}(h^2),$$

where h is a small step length, here chosen $h = 0.002$.
However, this function has a very bad efficiency since for each iteration, the complete wave function has to be evaluated four times, including the very time consuming Slater determinant. Therefore 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 [32] and use for both parts, spin up and down,

$$\frac{1}{|D|}\frac{\partial |D|}{\partial \alpha} = \operatorname{tr}\left(D^{-1} \frac{\partial D}{\partial \alpha} \right) = \sum_{i,j=1}^{n} D_{ij}^{-1} \frac{\partial D_{ji}}{\partial \alpha}.$$

This means that we only have to take the derivative of each single-particle wave function of the Slater matrix with respect to its variational parameter and finally take the trace of $D^{-1}(\alpha)\dot{D}(\alpha)$.
The derivatives of the single-particle orbitals are

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

where 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 analogous for H_{n_y}.

To get a better convergence of the algorithm, we performed two changes in the provided functions *dfpmin* and *lnsrhc*:
First of all, we noticed that sometimes it 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, making it require more iterations then are actually necessary. Therefore, we normalize the first computed gradient to a norm equal 1. That way, the trial parameters stay close to the starting guess and we observed less iterations for convergence and better and more stable final results.

Parallelizing the code

To reduce the needed time for the MC runs, the whole VMC algorithm is parallelized 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.

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 8.6. 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 the walkers

```
1. Loop over all cycles:
   (a) WALK: Loop over all walkers:
       i. Loop over all particles
          A. Calculate new trial position $x = y + DF(y)\tau + \chi$.
          B. Compute acceptance ratio $R$ according to Eq. (7.31)
          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 $E_{\text{local}}$
   (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 $\epsilon$ 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 8.6: Our DMC algorithm. We follow this procedure for each of the samples.

really represents the desired distribution. At this stage there is a steady flow: Walkers are created in regions with low local energy and killed in the ones with higher local energy.
However, if the walkers are initially distributed far away from 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 [44]:

$$E_T(t + \Delta t) = E_{\text{est}}(t) - \frac{1}{g\Delta t} \log \frac{W_t}{W_0}.$$

Here $E_{\text{est}}(t)$ is an estimate of the energy at time t, which we have chosen as 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 [44], 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 above all to stabilize the number of walkers and obtain the right steady-state distribution.
Later, in the sampling phase where the walkers are in a more or less stationary state, the trial energy can be updated less frequently and is set 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 [44], 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

$$\begin{aligned} E_{\text{mix}} &= \frac{\int \Psi \hat{H} \Psi_T dR}{\int \Psi \Psi_T dR} = \frac{\int \Psi_T \Psi \frac{1}{\Psi_T} \hat{H} \Psi_T dR}{\int \Psi \Psi_T dR} \\ &= \frac{\int E_L f(\mathbf{R}, t) dR}{\int f(\mathbf{R}, t) dR}. \end{aligned}$$

In our case

$$f(\mathbf{R}, t) = \sum_{i=1}^{N} w_i \delta(\mathbf{R} - \mathbf{R_i}),$$

where w_i is 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 $\Psi \to \Phi_0$, 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 variational energy associated with Ψ_T. In the $\Delta t \to 0$ limit, the cut-off has no effect, such that the results for small time steps Δt, especially if extrapolated, are correct.

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.

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 [45, 46], 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 greater, too. 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 [44].

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).

N	ω	E_{DMC}	E_{DMC} in [30]
2	1.0	3.00000(3)	3.00000(3)
6	0.28	7.6001(2)	7.6001(1)
6	1.0	20.1598(4)	20.1597(2)
12	0.28	25.636(1)	25.6356(1)
12	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 [30].

An additional suggestion of [44] 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, one makes 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,

$$\Psi_T(\alpha, \beta) = |D_\uparrow(\alpha)|\,|D_\downarrow(\alpha)|\,.$$

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 8.6 confirms our expectations, which suggests that the tested parts of our program are working correctly.

For the final DMC program, we compare our results with [30], where exactly the same systems, two-dimensional parabolic quantum dots, have been studied with DMC and the Coupled-Cluster method. Table 8.6 compares those of our results that are listed in [30], too. All our results are in excellent agreement with the reference, suggesting that our DMC code produces correct results with reasonable precision. Hence it can be used as reliable reference for our SRG code.

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 making out computational challenges and methods to improve convergence, we continue with in-medium calculations. Here we start with Wegner's canonical generator, which as standard generator serves as first starting point to drive the Hamiltonian to a diagonal form, and identify the problems associated with this generator in our in-medium calculations. Eventually, we devote a whole section to IM-SRG(2) calculations with White's generator, with which we perform our final computations and compare the SRG with other many-body methods.
Note that all results in this chapter are given in atomic units, as introduced in chapter 5.

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 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 therefore can be used as a benchmark to measure the truncation error later on.

9.1.1 Code validation

When developing a computer program, it is crucial to test the code for errors. Large errors are usually easy to detect, since one often has an estimate for the results. Therefore it does usually not need more than a few runs to reveal that there is a bug in the code, which has to be found and correct. A much greater challenge is to identify minor bugs in the code, often leading to results that are incorrect, but still within the estimated range.
In order to guarantee that our implementation is free of errors, we therefore perform numerous tests where we know the exact result. Reproducing those ones, we can be sure that the tested part of program works as it is expected to do, and that this one 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. Two of the major error sources are an incorrect sign after applying the creation and annihilation operators, as well as a wrong interaction element due to incorrect spin considerations when computing direct and exchange term.
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 [40], which makes it possible to compare every single element.

The considered system contains twelve singe-particle states with the following mappings $|\alpha\rangle \to |n, m, m_s\rangle$:

$$\begin{aligned}
|0\rangle &\to |0,0,-1\rangle, & |6\rangle &\to |0,-2,-1\rangle\\
|1\rangle &\to |0,0,+1\rangle, & |7\rangle &\to |0,-2,+1\rangle\\
|2\rangle &\to |0,-1,-1\rangle, & |8\rangle &\to |0,+2,-1\rangle\\
|3\rangle &\to |0,-1,+1\rangle, & |9\rangle &\to |0,+2,+1\rangle\\
|4\rangle &\to |0,+1,-1\rangle, & |10\rangle &\to |0,0,-1\rangle\\
|5\rangle &\to |0,+1,+1\rangle, & |11\rangle &\to |0,0,+1\rangle.
\end{aligned}$$

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 mainly 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 (6.25) and oscillator frequency $\omega = 1.0$, this matrix should read [40]

$$H = \begin{bmatrix}
3.2533 & 0.3133 & -0.3133 & 0.3133 & -0.3133 & 0.1175 & -0.1175 & 0.2350\\
0.3133 & 4.8617 & -0.2350 & -0.0783 & 0.0783 & -0.0881 & 0.0881 & 0.1371\\
-0.3133 & -0.2350 & 4.8617 & 0.0783 & -0.0783 & 0.0881 & -0.0881 & -0.1371\\
0.3133 & -0.0783 & 0.0783 & 4.8617 & -0.2350 & 0.3035 & -0.1469 & 0.1371\\
-0.3133 & 0.0783 & -0.0783 & -0.2350 & 4.8617 & -0.1469 & 0.3035 & -0.1371\\
0.1175 & -0.0881 & 0.0881 & 0.3035 & -0.1469 & 6.7160 & -0.1285 & 0.1395\\
-0.1175 & 0.0881 & -0.0881 & -0.1469 & 0.3035 & -0.1285 & 6.7160 & -0.1395\\
0.2350 & 0.1371 & -0.1371 & 0.1371 & -0.1371 & 0.1395 & -0.1395 & 6.7491
\end{bmatrix}.$$

This matrix is exactly reproduced by our code, proving the correct setup of the Hamiltonian.

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 [48]. For the above Hamiltonian matrix, this function returns the lowest eigenvalue

$$E_0 = 3.0386,$$

which is in accordance to [40]. For two shells, we expect [41]

$$E_0 = 3.1523,$$

which is also reproduced. As in the rest of the thesis, the energy is given in units of $[E_H]$. The last two results imply that also this diagonalizing function is correctly applied and can be relied on when benchmarking the SRG results.

9.1.2 Numerical results

Having verified that the SRG routine obtains the correct input matrix, we perform runs for systems of different size and oscillator frequency 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 for 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. Mathematically, the complete non-interacting energy $E_{non} = \sum_i \epsilon_i = \omega \sum_i (2n + |m_l| + 1)$ 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, leading to a relative increase of the kinetic energy. Experimentally, the frequency can be tuned, too, which allows to analyse the system in terms of different frequencies. For this reason, we perform all our numerical calculations with different values of ω and 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

$$\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right] \tag{9.2}$$

and Wegner's original choice

$$\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right], \tag{9.3}$$

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	0.441446672	0.441446672	0.441446672
	8	0.4412461536	0.4412461536	0.4412461536
	10	0.441135127	0.441135127	0.441135127
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 $[E_H]$) for $N = 2$ particles is compared between 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 can be found in Appendix B.

where we suppress the s-dependence for simplicity and label them as $\hat{\eta}_1, \hat{\eta}_2$ for easy access later on.

In order to compare their effect on the flow of the Hamiltonian, we perform calculations 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).

As can be seen in both tables, both generators reproduce precisely the ground state energy which can be 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 choice of the generator $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$ 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,

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
					43.29176947

Table 9.2: Comparison of the ground state energy E_0 as in table 9.1. The label "nc" denotes non-converging runs. TODO: maybe some more examples for 12 particles

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 that numerical instabilities in the ODE solver are responsible for this effect. The convergence behaviour will be discussed more thoroughly in subsection 9.1.3.

Having found out that, in general, generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$ yields more often a converging result, we were interested in those cases where both generators lead to a converging E_0, and have analysed whether there exist qualitative differences in the flow, too.

Figure 9.2 shows two typical results: Although the final point of convergence is reached for approximately the same value of λ, it is noticeable that the curve of generator $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$ constantly lies below the one of $\hat{\eta}_2 = \left[\hat{H}^{\text{d}}, \hat{H}^{\text{od}}\right]$. This in turn means that even in the converging case, generator $\hat{\eta}_1$ seems to lead to a better and faster decoupling for the studied quantum dots.

In order to verify the decoupling of the ground state energy 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 us 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 in the whole thesis, we have that $[\lambda] = [E] = E_H$ due to $\lambda = s^{-1/2}$. In the case of $\lambda = 2.0$, just the matrix elements connecting states with 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

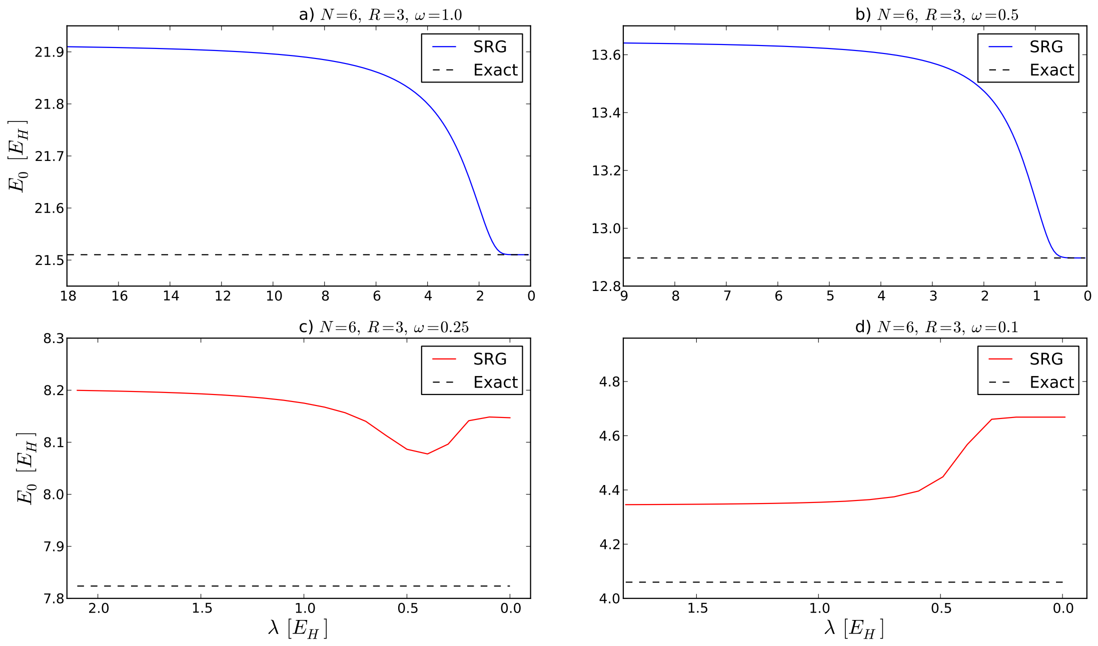


Figure 9.1: Typical plots demonstrating the obtained behaviour of the ground state energy for converging ((a) and (b) and non-converging cases ((c) and (d). In these plots, generator $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$ has been used, but the figure illustrates likewise curves for converging and non-converging runs if generator $\hat{\eta}_1 = \left[\hat{T}_{\mathrm{rel}}, \hat{V}\right]$ is used.

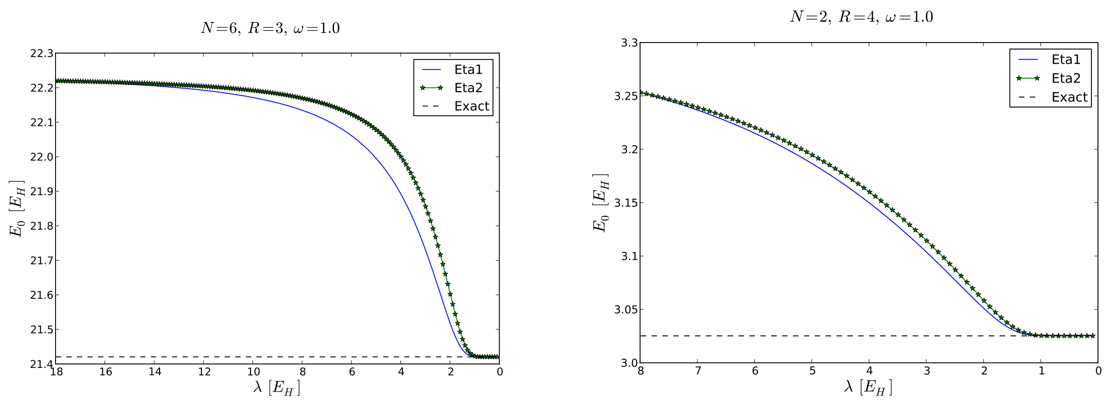


Figure 9.2: Comparison of the convergence behaviour of the the two generators $\hat{\eta}_1 = \left[\hat{T}_{\mathrm{rel}}, \hat{V}\right]$ and $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$. Although both generators lead to a converging result and convergence sets in at approximately the same value of λ, decoupling appears to be faster for $\hat{\eta}_1$.

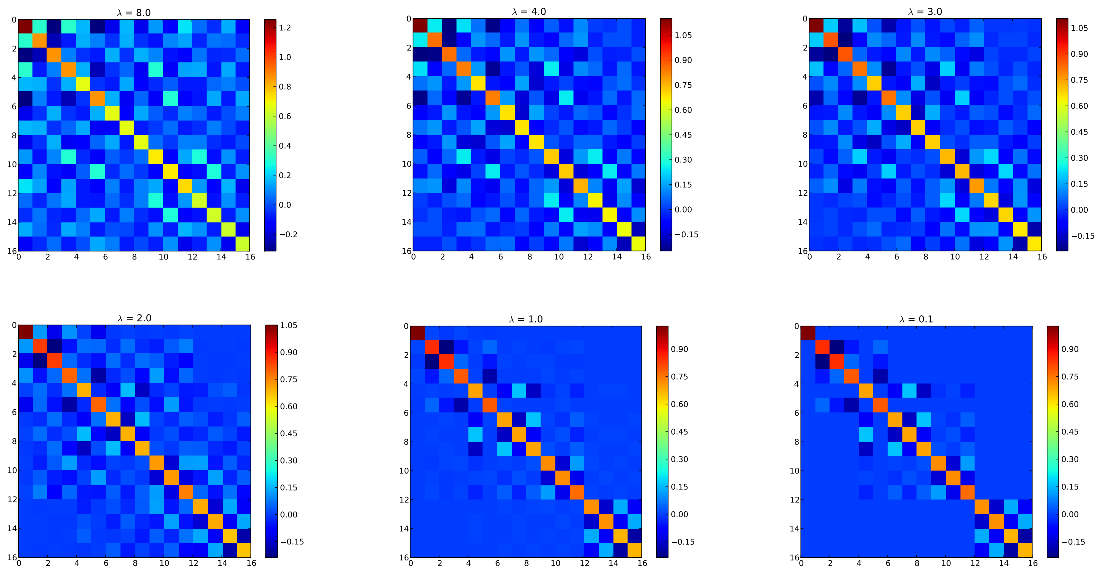


Figure 9.3: Snapshots of the interaction elements of the Hamiltonian matrix at different stages of the flow, specified by the flow parameter λ. The evolution is show for a system of $N = 2$ particles, $R = 4$ shells, oscillator frequency $\omega = 1.0$ and generator $\hat{\eta}_1 = [\hat{T}_{\text{rel}}, \hat{V}]$.

plot, the ground state seems to be completely decoupled from the remaining states. Even if not the whole interaction is completely diagonal, this should mean that the energy at that stage should have converged to its final value.

In order to confirm this statement, we have plotted the evolution of the ground state energy for this example 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 Hamiltonian matrix in figure 9.3 does only slightly change and seems to have converged to its final shape for $\lambda = 0.1$. Analogously, we observe a more or less constant value for E_0. Thus, the latter one reflects very well the behaviour of the whole Hamiltonian.

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 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 $\Sigma_\alpha = \epsilon_1 + \epsilon_2$.

The table illustrates 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 a diagonal form. Further on, 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. Taking account the shell structure of figure 5.1, we see that for the first three states of this 6×6 block, the particle 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

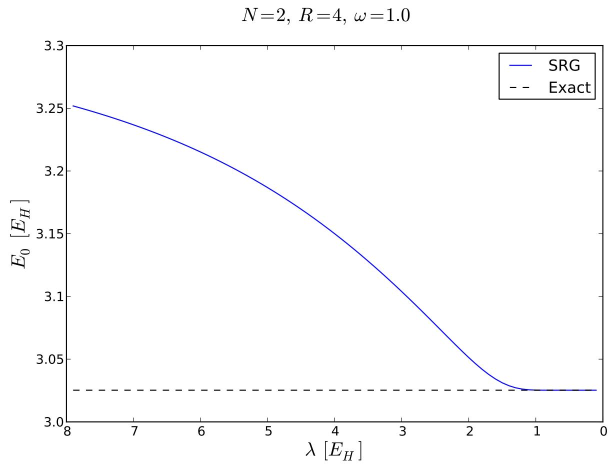


Figure 9.4: Evolution of the ground state energy for a system with $N = 2$ particles, $R = 4$ shells, oscillator frequency $\omega = 1.0$ and generator $\hat{\eta}_1 = [\hat{T}_{\text{rel}}, \hat{V}]$.

$\lvert\alpha\rangle$	Σ_α	$\lvert\alpha\rangle$	Σ_α
$\lvert 0,1\rangle$	2ω	$\lvert 5,16\rangle$	6ω
$\lvert 0,11\rangle$	4ω	$\lvert 6,9\rangle$	6ω
$\lvert 1,10\rangle$	4ω	$\lvert 7,8\rangle$	6ω
$\lvert 2,5\rangle$	4ω	$\lvert 10,11\rangle$	6ω
$\lvert 2,19\rangle$	6ω	$\lvert 12,15\rangle$	8ω
$\lvert 3,4\rangle$	4ω	$\lvert 13,14\rangle$	8ω
$\lvert 3,18\rangle$	6ω	$\lvert 16,19\rangle$	8ω
$\lvert 4,17\rangle$	6ω	$\lvert 17,18\rangle$	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 units of $[E_h]$. The single-particle states are indexed as in figure 5.1
.

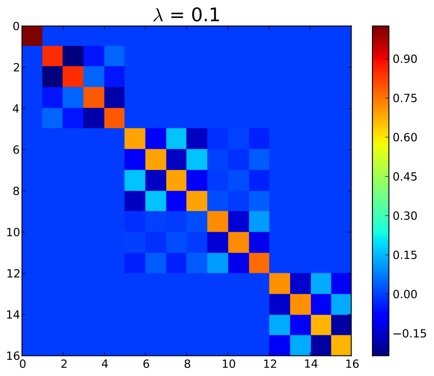


Figure 9.5: Interaction elements of the Hamiltonian for a system of $N = 2$ particles, $R = 4$ shells and $\omega = 1.0$. Snapshot of the evolution at $\lambda = 0.1$. Compared to figure 9.3, the basis states are explicitly ordered by increasing energy, pointing out the block-diagonal form of the final Hamiltonian.

the Hamiltonian matrix depends very much on the 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, one needs to exchange states $|2, 19\rangle$ and $|3, 4\rangle$, which unfortunately have been arranged in the wrong way considering energy Σ_α. With this arrangement, the final Hamiltonian looks as in figure 9.5, clearly illustrating the block-diagonal form due to energy degeneracies. 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 an energy degeneracy of two or more states prevents the interaction connecting those states to converge quickly to zero. This is valid for Wegner's choice of the generator $\hat{\eta}_2 = \left[\hat{H}^{\mathrm{d}}, \hat{H}^{\mathrm{od}}\right]$, as well as $\hat{\eta}_1 = \left[\hat{T}_{\mathrm{rel}}, \hat{V}\right]$.
In the case of quantum dots, however, we do not expect this to have impact on the ground state energy E_0, since that one is uniquely defined without degeneracy of several states. The fact that E_0 converges to the same value as an exact diagonalization results in, 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 with a non-converging ground state energy E_0. In the following, we will try two methods to improve convergence and analyse how the result is affected. First, we will exchange the standard 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*. The effective interaction aims to speed up the convergence rate as function of the numbers of shells R. It has been widely used first in nuclear physics, and later on also for quantum dots [30].

The basis 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 defined as

$$\hat{P} = \sum_{i=1}^{m} |e_i\rangle \langle e_i| \,. \tag{9.5}$$

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:

$$\hat{H} = \begin{bmatrix} \hat{P}\hat{H}\hat{P} & \hat{P}\hat{H}\hat{Q} \\ \hat{Q}\hat{H}\hat{P} & \hat{Q}\hat{H}\hat{Q} \end{bmatrix}$$

The approach in effective interaction theory is to find a unitary transformation

$$\hat{H}' = e^{-s}\hat{H}e^{s},$$

that decouples the model space $\mathcal{P}$ from the complement space $\mathcal{Q}$, namely

$$\hat{P}\hat{H}'\hat{Q} = \hat{Q}\hat{H}'\hat{P} = 0.$$

Since a unitary transformation preserves an operator's eigenvalues, the effective Hamiltonian is given by

$$\hat{H}_{\text{eff}} = \hat{P}\hat{H}'\hat{P} \tag{9.6}$$

The effective Hamiltonian $\hat{H}$ has m eigenvalues identical to the ones of $\hat{H}$ and acts on a smaller space $\mathcal{P}$ than the full Hilbert space $\mathcal{H}$.
In our case, the effective interaction, defined as

$$\hat{V}_{\text{eff}} = \hat{H}_{\text{eff}} - \hat{P}\hat{H}_0\hat{P}, \tag{9.7}$$

is produced by a unitary transformation of the two-body Hamiltonian. For $N = 2$ particles, the effective interaction is expected to yield the exact ground state energy when the Hamiltonian is diagonalized. For more than two particles, one will no longer obtain the exact eigenvalue. However, the idea is that the two-body contributions are still modelled in a more accurate way, such that the whole result gets a better approximation to the exact one. In our case,

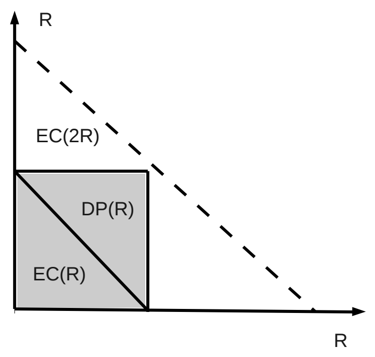


Figure 9.6: Illustration of the different spaces that are involved in forming the effective interaction. The standard approach generates the interaction elements in an energy cut model space $\mathbb{EC}(R)$, corresponding to a cut in the global shell number R (and energy). The SRG method uses a direct 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 $\mathbb{DP}(R)$.

all interaction elements are generated by the OpenFCI library [37] and taken as input for our code.

In order to obtain a well-defined interaction, the effective interaction must be generated in an energy cut model space $\mathbb{EC}(R)$, see figure 9.6. In contrast to the direct product space $\mathbb{DP}(R)$, that one does not cut the single-particle shell numbers R (or energy), but the global shell number. This space $\mathbb{EC}(R)$ contains all the symmetries of the interaction and a generation in $\mathbb{DP}(R)$ would break essential symmetries like conservation of center-of-mass momentum [49]. Although $\mathbb{EC}(R)$ yields exact results for two particles, it has shown to have convergence problems [49]. Therefore the common practice arose [40, 41, 49, 50] 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 the SRG method works in $\mathbb{DP}(R)$, we simply restrict ourselves to the interaction elements in $\mathbb{DP}(R)$ when using an effective interaction without energy cut, which has elements generated in $\mathbb{EC}(2R)$. In the case of a large basis, the error is assumed to be fairly small, but in many cases convergence is considerably improved.

Tables 9.4 and 9.5 compare the results of a standard interaction with the ones obtained using an effective interaction. As before, all the converging runs yield precisely the same result as the exact diagonalization of the Hamiltonian does. 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 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 $[E_H]$) for $N = 2$ electrons and $\omega = 1.0$, which is the exact result that has been analytically derived [51].

$N = 2$

		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	1.06693768
	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	1.71357741
	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	3.00319931

Table 9.4: Comparison of the ground state energy E_0 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 $[E_H]$. All runs have been performed with 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 $[E_H]$) as in table 9.4, this time for $N = 6$ particles. The label ”nc” denotes non-converging runs.

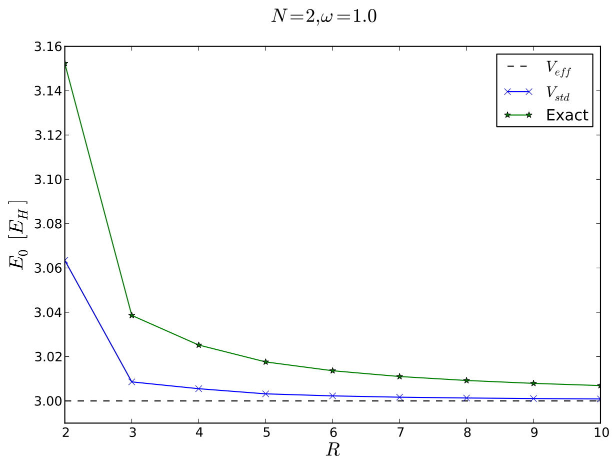


Figure 9.7: Comparison between standard and effective interaction (without energy cut) for $N = 2$ particles. With increasing number of shells R, the ground state energy E_0 converges towards the analytically exact result. The convergence with effective interaction is faster and the result for a given R a better approximation for E_0 than with standard interaction.

Although the effective interaction without energy cut is less exact, table 9.4 shows that it is numerically much more stable. With energy cut, the convergence behaviour turned out to be worth than with standard interaction, therefore we will use the effective interaction only without energy cut from now on. This decision has also be made by previous and fellow master students studying quantum dots with Full Configuration Interaction [50] and Coupled Cluster [41] and will therefore enable us to directly compare results in section 9.2.

Table 9.4 and figure 9.7 illustrate that the effective interaction gives a better convergence for the ground state energy as function of shell numbers R. However, it does not seem to lead to a considerable improvement of integrator's numerical stability. The convergence is getting worse as the oscillator frequency ω is decreasing as the number of particles N is getting larger, which indicates that a higher correlation between the electrons is responsible for the numerical instability.

Hartree-Fock basis

Since changing the interaction has not resulted in an improved numerical stability, we hope to solve this problem by replacing the harmonic oscillator by a Hartree-Fock basis. After the Hartree-Fock calculation, the one-particle-one-hole excitations will be zeroed out. Since those ones are assumed to have greatest impact on the electron interaction, we hope to have fewer problems due to high correlations.

A first important result is that runs with a Hartree-Fock basis give the same result as runs 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, we have shown that the SRG method is also stable with a Hartree-Fock basis and

$N = 2, \omega = 1.0$

	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 $[E_H]$) obtained with a harmonic oscillator (HO) and a Hartree-Fock (HF) basis. The large number of specified decimals shall demonstrate that a Hartree-Fock basis reproduces 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 $[E_H]$.

converges to the correct results.

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 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 [40, 41, 49], we know that convergence sometimes gets better as the number of shells R is increased. Since an increase of R also corresponds to a better result since higher correlations are included, these cases are of higher interest, 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 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 study systems with a

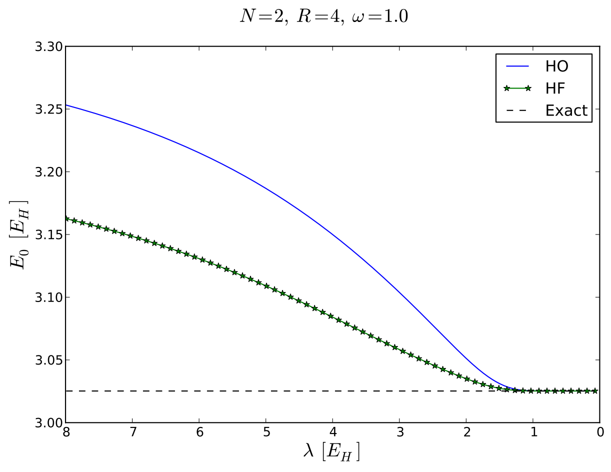


Figure 9.8: Comparison of the convergence behaviour for a harmonic oscillator (HO) and a Hartree-Fock (HF) basis. With a HF basis, the one-particle-one-hole excitations are already zeroed out from the beginning, such that the ground state energy E_0 starts with a lower value. The plot shows a system with standard interaction and $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$.

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 $M = M_s = 0$.

larger number basis states.

9.1.4 Time analysis

The advantage of the SRG method in free space is that no truncation is made and, apart from non-converging cases, the same result is obtained as for exact diagonalization. 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, this might quickly exceed the available memory to store the full $n \times n$ Hamiltonian matrix. On the other hand, the computational cost gets 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 to rather small systems in table 9.2. To get an impression of the required CPU time of such a run, table 9.9 lists how much time is needed for a system with $N = 6$ particles on $R = 4$ shells in order to obtain the ground state energy E_0 up to a certain accuracy. Together with figure 9.9, it gets obvious that especially the last part of the flow, i.e. the last correct digits, are very CPU demanding.

Analysing the program explicitly, we have found out that for larger systems more than 90%

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: The table lists the required CPU time for different values of the flow evolution parameter λ. The corresponding ground state energy E_0 is given in units of $[E_H]$, from $\lambda = s^{-1/2}$ follows that $[\lambda] = [E]$. The studied system consists of $N = 6$ particles on $R = 4$ shells with $\omega = 1.0$ and $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$. 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.

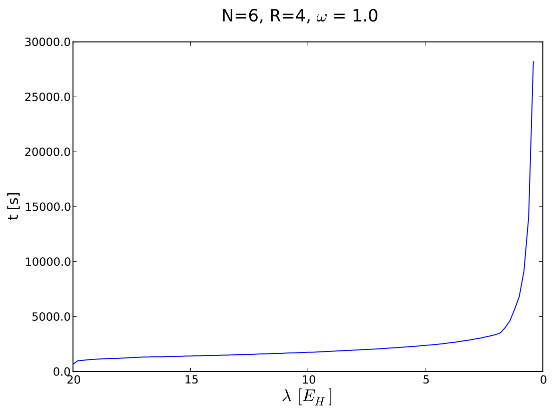


Figure 9.9: Required CPU time as function of the flow parameter λ. The example is for $N = 6$ particles on $R = 4$ shells and $\hat{\eta}_1 = \left[\hat{T}_{\text{rel}}, \hat{V}\right]$.

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 up to this point is that, we in fact do 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 routine of the Armadillo library, the needed CPU time is 3s, which is five orders of magnitude smaller than the corresponding SRG time to obtain an accuracy of ten correct digits.
Although the method itself gives very promising results, this time analysis emphasizes the need for another approach, the in-medium SRG method.

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 will perform the runs in medium. We will especially focus on how the truncation error effects the results. Since the in-medium approach allows us to study systems with a much higher number of basis states, we will study how the ground state energy E_0 behaves as the number of particles N and shells R is increased. Unless otherwise stated, all runs until section 9.3 are performed with Wegner's canonical generator $\hat{\eta}_s = \left[\hat{H}_s^{\mathrm{d}}, \hat{H}_s^{\mathrm{od}}\right]$.

9.2.1 Code validation

As we did for the free-space approach, it is important 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. Since a system with $N = 2$ particles can maximally have two-particle-two-hole excitations, we know that IM-SRG(2) should be exact as long as we also include the loop terms of higher-order interactions. With those loop terms we mean effective two-body terms of the form

$$\langle pq||rs\rangle = \sum_i \langle pqi|v^{(3)}|rsi\rangle,$$

and effective one-body terms of the form

$$f_{pq} = \frac{1}{2}\sum_{ij} \langle pij|v^{(3)}|qij\rangle,$$

where i and j sums over all hole states. We want to remind the reader that in our model of quantum dots, three-body interactions are not present from the beginning on. Instead, they are generated during the flow and higher order-correlations evolve first in the generator $\hat{\eta}$ and are then transferred to the Hamiltonian by $\frac{d\hat{H}_s}{ds} = \left[\hat{\eta}_s, \hat{H}_s\right]$.
If we want to exclude truncation errors for two particles, we therefore need to consider those effective one- and two-body terms for the generator $\hat{\eta}$ as well as for the interaction terms of the Hamiltonian.

ω	R	In-medium SRG	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 9.10: Comparison of the ground state energy E_0 in free space and IM-SRG(2/3) for $N = 2$ particles. With IM-SRG(2/3) we denote second-order IM-SRG(2), extended by the loop terms of three-body interactions. The high number of specified digits shall emphasize that exactly the same result is obtained. All runs are performed with HO basis and standard interaction.

Table 9.10 shows that we obtain exactly the same results as in free space. First, this implies that the flow equations are correctly implemented. Second, we see that as long as the result is converging, the in-medium approach does not introduce greater numerical roundoff errors etc., 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 Numerical results with IM-SRG(2)

Having verified that the flow equations are correctly implemented, we will first start with $N = 2$ particles. This will enable us to make statements about how the truncation error affects the results, since we have the possibility to compare with the IM-SRG(2/3) approach that also includes the loop terms of three-body interactions.

From table 9.11 and figure 9.10 we conclude that the truncation error generally results in a lower ground state energy E_0 than the untruncated result. The relative error lies mainly between 0.1% and 0.2% and seems to diminish slightly as the number of shells R is increased. Since a larger number of basis states takes care of higher order correlations, we are more interested in those results, anyway, and the trend of a smaller relative error is a desirable feature.

Before we move on to more results, it seems necessary to explain why we have chosen not to include the three-body terms in our further calculations. The advantage would be to introduce a smaller truncation error in the flow equations, but the computational cost would be much higher. Instead of Eqs. (6.29)-(6.31), we would have to solve Eqs. (6.21)-(6.24) each time the derivative function is called by the ODE solver. This would involve a considerably larger number of terms, with the maximal number of indices to be summed over increased from four

R	IM-SRG(2)	IM-SRG(2/3)	Rel.Diff.
3	3.033884	3.038605	0.16%
4	3.018303	3.025231	0.23%
5	3.012019	3.017606	0.19%
6	3.008581	3.013626	0.17%
7	3.006509	3.011020	0.15%
8	3.005058	3.009236	0.14%
9	3.004008	3.007929	0.13%
10	3.003198	3.006937	0.12%

Table 9.11: Comparison of the ground state energy E_0 obtained with IM-SRG(2) and IM-SRG(2/3), which is expanded by the loop terms of three-body interactions. We performed the runs for $N = 2$ particles, an oscillator frequency $\omega = 1.0$, standard interaction and HO basis. The ground state energy is given in units of $[E_H]$. The last column lists the relative error $\left|E_{0[\text{IM-SRG}(2/3)]} - E_{0[\text{IM-SRG}(2)]}\right| / E_{0[\text{IM-SRG}(2/3)]}$.

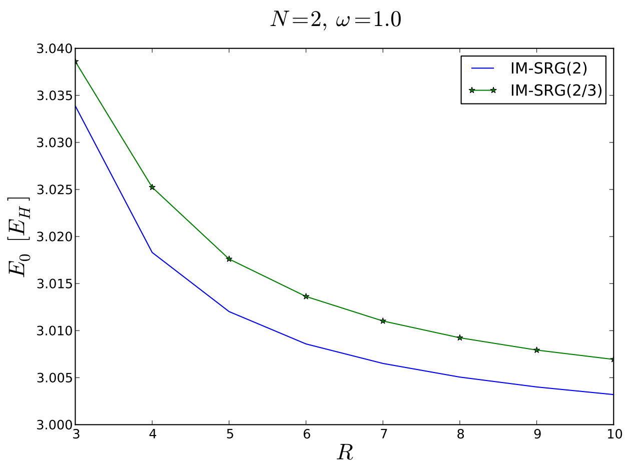


Figure 9.10: Ground state energy E_0 for $N = 2$ particles as function of the shell number R. The result obtained with IM-SRG(2) is compared with the one of IM-SRG(2/3), which also contains the loop terms of three-body interactions. All runs have been performed with oscillator frequency $\omega = 1.0$, HO basis and standard interaction.

R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
2	nc	1.089121	1.767956	3.143526
3	0.4252721	1.022391	1.674052	3.033884
4	0.4254326	1.014839	1.663100	3.018304
5	0.4284273	1.016052	1.661176	3.012019
6	0.4354808	1.017138	1.660178	3.008581
7	0.4368345	1.017620	1.659615	3.006510
8	0.4367619	1.017846	1.659181	3.005059
9	0.4368605	1.017956	1.658851	3.004009
10	0.4371156	1.018009	1.658578	3.003198
11	0.4371817	1.018017	1.658347	3.002555
12	0.4372050	1.018006	1.658150	3.002031
DMC	0.44087(3)	1.02166(3)	1.65975(2)	3.00000(3)

Table 9.12: Ground state energy E_0 (in $[E_H]$) for a system with $N = 2$ particles, HO basis and standard Coulomb interaction. For benchmarking, the last line shows the corresponding result that we got with Diffusion Monte Carlo (DMC). The convergence behaviour is illustrated in figure 9.11.

to six. However, even with IM-SRG(3) we would still have a truncation error for systems with $N = 6, 12, \ldots$ particles. This means that the huge number of additional terms due to three-body interactions would just improve, but not eliminate the truncation error. It therefore remains a question of how much additional CPU time one is willing to spend for a slight improvement of the result. In this thesis, we have decided to restrict us to IM-SRG(2) and rather focus on how to improve convergence within the limitations of this approach.

9.2.3 Convergence analysis

Analogously to the free-space case, we first look at the easiest example of $N = 2$ particles with standard Coulomb interaction and harmonic oscillator basis. That way, we can identify potential problems of the SRG method, before we move on to higher number of particles and methods to improve convergence.

Table 9.12 and figure 9.11 present the obtained results as function of the shell number R. Since we are especially interested in whether an increase of R causes E_0 to converge to the exact result, we also performed Diffusion Monte Carlo (DMC) calculations for benchmarking.

The first observation is that the convergence behaviour is qualitatively different, depending on the oscillator frequency ω. For $\omega = 0.1$ and $\omega = 0.28$, the energy seems to be converging towards the DMC result from below, whereas for $\omega = 1.0$, convergence takes place from above. For $\omega = 0.5$, E_0 is decreasing as function R and it first appears as if the DMC result is reached from above. However, the ground state energy does not stop at the DMC value, but is further decreasing and getting too low. Since a larger number of shells R means that also higher-order correlations are taken into account, we actually hope for convergence towards the analytically correct result for $R \to \infty$. In plot c) with $\omega = 0.5$, this is definitely not reached and a similar trend seems possible for $\omega = 1.0$.

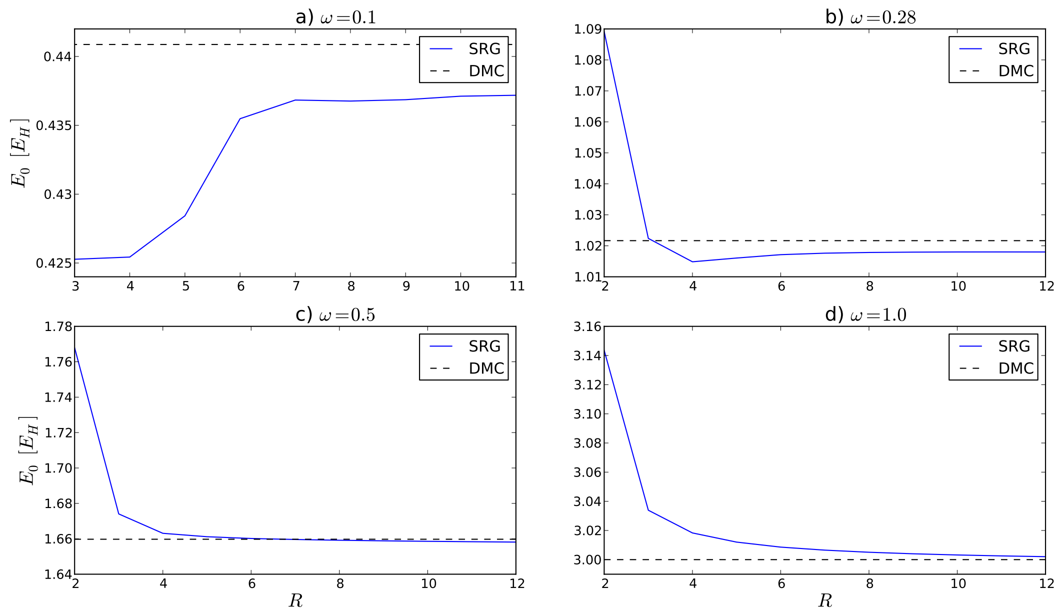


Figure 9.11: Convergence behaviour of the ground state energy E_0 as function of the shell number R. All runs have been performed with $N = 2$ particles, HO basis and standard Coulomb interaction. The SRG results are compared with the ones obtained from Diffusion Monte Carlo (DMC) caculations.

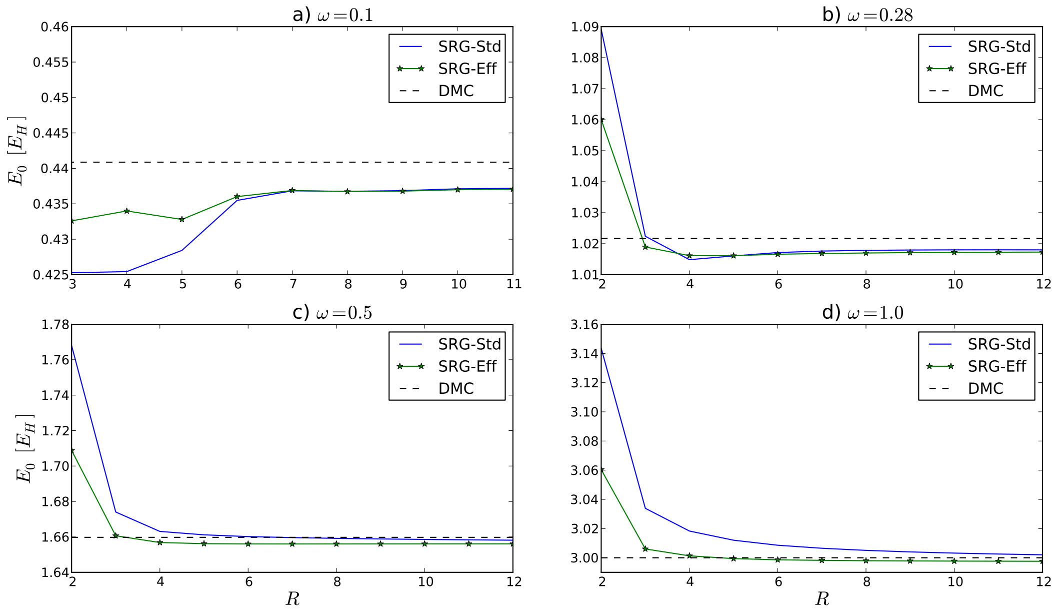


Figure 9.12: Comparison of the ground state energy E_0 between SRG with standard Coulomb interaction (SRG-Std) and effective interaction (SRG-Eff). All runs have been performed with $N = 2$ particles and HO basis. For benchmarking, the plots also include the corresponding DMC result.

Since for $N = 2$ particles, the SRG method involves the smallest truncation error, we do not expect better results for $N = 6$ particles and will therefore try to improve convergence by an effective interaction and a Hartree-Fock basis, before we move on to more particles.

9.2.4 Improving convergence: Effective interaction and Hartree-Fock basis

Effective interaction

As explained in section 9.1.3, effective interactions can be used to speed up convergence as function of the shell number R. For our SRG calculations in medium, we use the same effective interaction as we did in free space, namely a unitary transformation of the two-body Hamiltonian, implemented in the OpenFCI library [37]. As explained before, this should yield the exact ground state energy E_0 for $N = 2$ particles when the Hamiltonian is diagonalized and this result should be independent of the number of shells R. For a larger number of particles N, the result will no longer be exact, but we still expect it to be a better approximation than a standard Coulomb interaction.
Moreover, in analogy to the free-space case, we will use the effective interaction without energy cut. This introduces a small additional error but makes the SRG runs numerically much more stable, especially as the oscillator frequency ω is lowered.

In order to make statements about how the results are affected, we performed the same runs

R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
2	0.4552503	1.060039	1.708803	3.060394
3	0.4325924	1.018938	1.660647	3.006047
4	0.4339930	1.016126	1.656843	3.001313
5	0.4327919	1.016113	1.656178	2.999413
6	0.4360176	1.016597	1.656036	2.998598
7	0.4368824	1.016830	1.656023	2.998209
8	0.4367201	1.016996	1.656036	2.997970
9	0.4367887	1.017110	1.656056	2.997824
10	0.4369907	1.017197	1.656072	2.997719
11	0.4370701	1.017252	1.656082	2.997641
12	0.43711739	1.017292	1.656087	2.997581
DMC	0.44087(3)	1.02166(3)	1.65975(2)	3.00000(3)

Table 9.13: Ground state energy E_0 (in $[E_H]$) for a system with $N = 2$ particles, HO basis and effective interaction. For benchmarking, we also give the results that we got with Diffusion Monte Carlo (DMC).

as in table 9.12, this time using an effective interaction. The numerical results can be found in table 9.13. Figure 9.12 compares directly the runs with effective interaction with the previous ones using a standard Coulomb interaction.
The major observation is that generally, the effective interaction does not yield a better result. Except for $\omega = 0.1$, the obtained ground state energy is lower than with standard interaction, which makes the already too small results (with respect to the DMC calculations) even slightly worse.
We also performed calculations with $N = 6$ particles, where the truncation error made by IM-SRG(2) is larger. As shown in table 9.14, the ground state energy E_0 gets again too low as the number of shells R is increased. Additionally, for low values of the oscillator frequency ω, our results are not converging. This observation is analogous to the free-space case and we suspect numerical instabilities in the ODE solver to be responsible for this effect. Since a lower oscillator frequency corresponds to higher correlations between the particles, it seems that our method gets unstable as the correlations are increased. This behaviour has also been observed for other ab-initio many-body methods, e.g. by previous master students studying circular quantum dots with the Coupled Cluster method [41, 49].
The unsuitable results as the number of shells R is increased, as well as non-convergence for lower values of ω, motivate to exchange the harmonic oscillator by a Hartree-Fock basis and study the convergence behaviour using this basis.

Hartree-Fock basis

Introducing a Hartree-Fock (HF) basis, the elements corresponding to one-particle-one-hole excitations are already zeroed out when the Hamiltonian in this basis is set up. That way, $\hat{H}$ 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 link states with comparably low energy difference.

R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
3	nc	nc	12.36436	20.85382
4	nc	7.69353	11.85851	20.20328
5	nc	7.63149	11.81564	20.18527
6	nc	7.54028	11.77229	20.16104
7	nc	nc	11.76773	20.15487
8	nc	nc	11.76193	20.15073
9	nc	nc	11.76187	20.14857
10	nc	nc	11.76192	20.14719
DMC	-	7.6001(2)	11.7888(2)	20.1598(4)

Table 9.14: Ground state energy E_0 (in $[E_H]$) for a system with $N = 6$ particles, HO basis and effective interaction. The label "nc" denotes non-converging runs.

	Standard				Effective			
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	0.4552503	1.060039	1.708804	3.060394
3	0.4354154	1.027421	1.677713	3.036218	0.4360931	1.020688	1.662112	3.007167
4	0.4267331	1.018093	1.665905	3.020183	0.4330903	1.017199	1.658105	3.002366
5	0.4362793	1.020519	1.664304	3.013951	0.4363536	1.018151	1.657803	3.000586
6	0.4371191	1.020532	1.662897	3.010398	0.4369135	1.018444	1.657671	2.999826
7	0.4382300	1.020842	1.662244	3.008294	0.4376575	1.018825	1.657765	2.999499
8	0.4384303	1.020830	1.661674	3.006790	0.4379233	1.019021	1.657810	2.999289
9	0.4385010	1.020767	1.661239	3.005695	0.4380649	1.019142	1.657844	2.999158
10	0.4385023	1.020669	1.660874	3.004841	0.4381416	1.019214	1.657860	2.999059
11	0.4384803	1.020565	1.660568	3.004162	0.4381797	1.019255	1.657864	2.998982
12	0.4384559	1.020467	1.660310	3.003607	0.4382028	1.019280	1.657862	2.998920
DMC	0.44087(3)	1.02166(3)	1.65975(2)	3.00000(3)	0.44087(3)	1.02166(3)	1.65975(2)	3.00000(3)

Table 9.15: Ground state energy E_0 (in $[E_H]$) for a system with $N = 2$ particles and HF basis. For benchmarking, the last line shows the corresponding result that we got with Diffusion Monte Carlo (DMC). The comparison with HO basis is illustrated in figure 9.13.

Therefore those elements are usually the last ones to be zeroed out and might result in a stiff system of differential equations, requiring many integration steps. Initially giving no contribution with a HF basis, we therefore hope for a faster convergence of our SRG results. Moreover, the Hartree-Fock calculation does not introduce a truncation error like the SRG method does, that is, starting with a Hartree-Fock basis, exact diagonalization yields 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.

Effect on convergence A first aspect to analyze is therefore the effect of the HF basis on the convergence of E_0 as function of the number of shells R.
The numerical results for $N = 2$ particles are listed in table 9.15 and illustrated in figure 9.13. An important result is that generally, the results with HF basis are closer to the DMC values

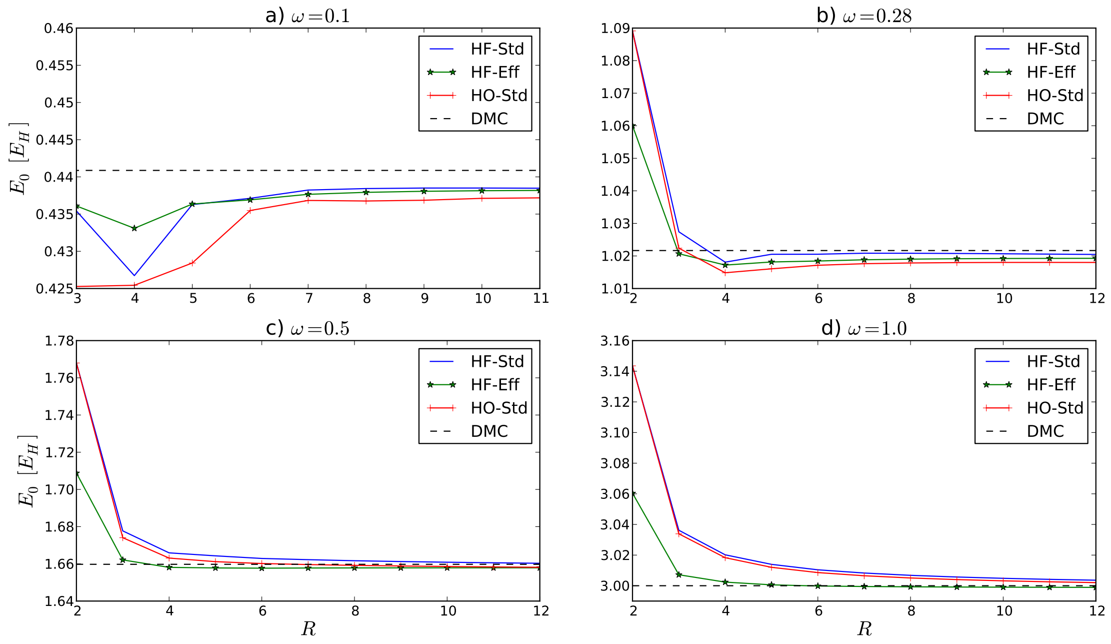


Figure 9.13: Comparison between the results with harmonic oscillator (HO) basis and Hartree-Fock (HF) basis. All runs have been performed with $N = 2$ particles. For the HO basis, the plots show the result with standard interaction (HO-Std); for the HF basis, both standard interaction (HF-Std) and effective interaction (HF-Eff) are shown. For benchmarking, we have also included the corresponding DMC values.

N	6	12	20
R	5	5	6
r	0.90	0.71	0.48

Table 9.16: Ratio between the required CPU time for the SRG calculation with Hartree-Fock (HF) 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 are with $\omega = 1.0$ and effective interaction.

than the ones with HO 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 9.13 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. For large values of R, we observe that the standard interaction sometimes gives a better approximation to E_0. However, this effect is caused by the superposition of two independent errors: The error due to a limited number of single-particle states (and shells R), and the truncation error of the SRG method. Obviously, the addition of these two errors favours sometimes the standard interaction. However, from a physical point of view, we still expect the effective interaction to give more accurate results.

Effect on CPU time As mentioned above, we expect the integration with HF basis to require fewer integration steps than the one with HO basis. A second aspect to compare is therefore the required CPU time with both bases.
In table 9.16, we have listed the ratio between the required CPU time needed with HF basis and HO basis. The chosen configurations of N and R may seem a bit randomly, but the problem was that in most cases the calculation with HO basis did not converge. To compute the ratio, we therefore had to pick out those runs where both calculations gave a converging result.
The main observation is that that the calculations with HF basis require less time than the ones with 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 stiffness of the equation system. With a large number of particles N, there are obviously more of those excitations possible, which explains the larger speedup for a higher number of particles N.

Numerical stability In table 9.14, using a harmonic oscillator basis, we observed numerical instabilities as the oscillator frequency is lowered, resulting in non-converging calculations. Similar problems were encountered during the calculations for table 9.16. This motivates to analyse another aspect, namely how the stability is affected by choosing a HF basis.
In table 9.17 we have repeated the calculations of table 9.14 with HF basis, both with standard Coulomb and effective interaction. We observe that convergence has improved enormously: Apart from a few exceptions for really small R, all runs are converging.

Interested if the HF basis always solves the problem of numerical instabilities, we subsequently

	Standard				Effective			
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	12.89292	21.41679	nc	8.07826	12.36955	20.86051
4	nc	7.84871	12.03399	20.41322	3.67022	7.70061	11.86345	20.20863
5	3.57515	7.68673	11.90040	20.31070	3.60074	7.64366	11.82329	20.19127
6	3.54968	7.61780	11.83243	20.25264	3.55268	7.59622	11.78635	20.17093
7	3.54587	7.609189	11.81643	20.22692	3.54689	7.59392	11.78248	20.16458
8	3.54887	7.60720	11.80874	20.21170	3.5472635	7.59418	11.78119	20.16110
9	3.54941	7.60536	11.80349	20.20133	3.54753	7.59421	11.78039	20.15884
10		7.60405	11.79986	20.19410		7.59429	11.77994	20.15744
DMC	3.5539(1)	7.6001(2)	11.7888(2)	20.1598(4)	3.5539(1)	7.6001(2)	11.7888(2)	20.1598(4)

Table 9.17: Ground state energy E_0 (in $[E_H]$) for a system with $N = 6$ particles and HF basis. The label "nc" denotes non-converging runs.

	Standard				Effective			
R	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 0.1$	$\omega = 0.28$	$\omega = 0.5$	$\omega = 1.0$
4	nc	nc	43.23911	70.29497	nc	nc	nc	68.80404
5	13.79490	27.17979	40.62397	67.00586	13.10228	26.48015	39.92077	66.28846
6	-	26.51326	39.99714	66.47815	-	26.18566	39.64222	66.07147

Table 9.18: First results for the ground state energy E_0 (in $[E_H]$) for a system with $N = 12$ particles and HF basis. The label "nc" denotes non-converging runs. The two calculations with "-" have not been performed to an end due to stiffness of the equation system, see text for further explanations.

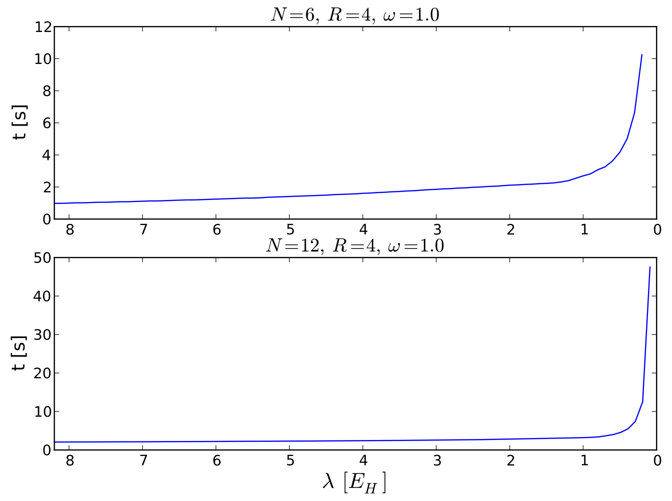


Figure 9.14: Required CPU time as function of the flow parameter λ. The example is with HF basis and effective interaction. The given time includes only the integration, not setup of the system, Hartree-Fock calculation etc.

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 9.19: 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.

performed calculations with higher numbers of particles N.
For $N = 12$ particles, the first results with $R \geq 5$ look quite promising, see table 9.18. However, during our calculations we encountered another problem, which let us restrict to very small values of R: 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 the calculations in table 9.18 and others with $N = 20, 30, \ldots$ 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 **??** in the Appendix. As we already observed in figure 9.9 for the free-space case and now again see for the in-medium approach in figure 9.14, the required CPU time blows up enormously as λ is approaching zero, even if the calculations have a comparably good convergence behaviour. The stiffness of the equations thus results in massive computational costs.
Moreover, for larger numbers of particles N, the calculations are not converging any more. When performing runs with $N = 20, 30, \ldots$ particles in order to analyse if we encounter the same problem of stiffness, almost no calculation with $\omega \in \{0.1, 0.28, 0.5, 1.0\}$ was 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 9.19.

The fact that those 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.
Both observations, blow up of CPU time due to stiffness and non-convergence of the equations, motivated us to implement additionally White's generator and to proceed with this one for larger values of N.

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 led us to 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 ground 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. (6.29)-(6.31) quite a lot, 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.

We proceed the following way: After validating our code, we will compare the convergence behaviour of the SRG method with both, Wegner's and White's, generator . Particularly, we will 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 will 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, comparing with other many-body methods.

9.3.2 Code validation

In accordance with the other implementations, we first performed a number of test calculations in order to be sure that the code is free of bugs. In this case, it makes sense to take the same tests as we did for Wegner's generator: Including additionally the loop term terms of higher order interactions, IM-SRG(2) should be exact for $N = 2$ particles and yield the same ground state energy E_0 as the free-space approach and exact diagonalization.
The results are listed in table 9.20, where we performed calculations with HO as well as HF basis. We observe that exactly the same results are obtained, 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 properties of $\hat{\eta}$, the second reason is that we assume the equations not to be stiff any more, reducing the number if integrations steps. With the number of integration steps as well as the number of floating-point operations per step reduced, we hope for a considerable reduction of CPU time.
Figure 9.15 shows that our assumptions are right: The calculations with White's generator do indeed require much less time . This seems to be more pronounced as the number of shells R is increasing, which 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, making it more straightforward to solve with the ODE solver.

Another interesting aspect is how the numerical results compare with each other. For $N = 2$ particles, figure 9.16 illustrates that Wegner's generator generally gives a better approxima-

		IM-SRG(2/3) White		
ω	R	HO basis	HF basis	Free space SRG
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 9.20: Comparison of the ground state energy E_0 in free space and IM-SRG(2/3) for $N = 2$ particles. With IM-SRG(2/3) we denote second-order IM-SRG(2), extended by the loop terms of three-body interactions. The high number of specified digits shall emphasize that exactly the same result is obtained. All runs have been performed with standard Coulomb interaction.

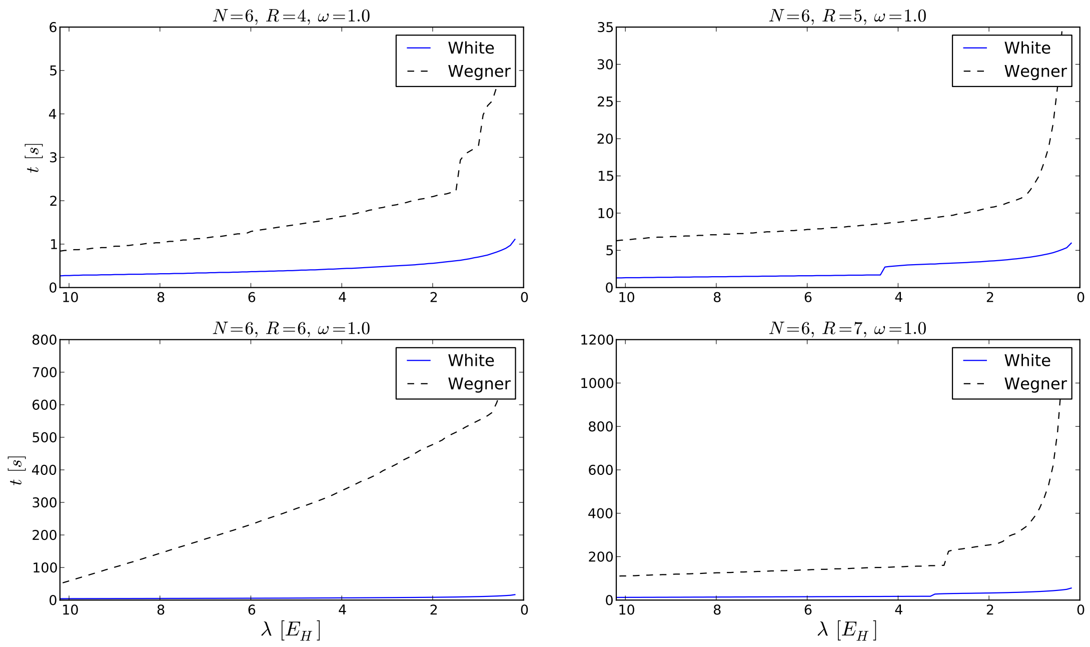


Figure 9.15: Required CPU time on $p = 1$ processor for SRG calculations performed with White's and Wegner's generator. The runs have been performed with HF basis and standard interaction. To give a reasonable comparison, the time does only consider the integration of the flow equations, not setup of the system, Hartree-Fock calculation etc.

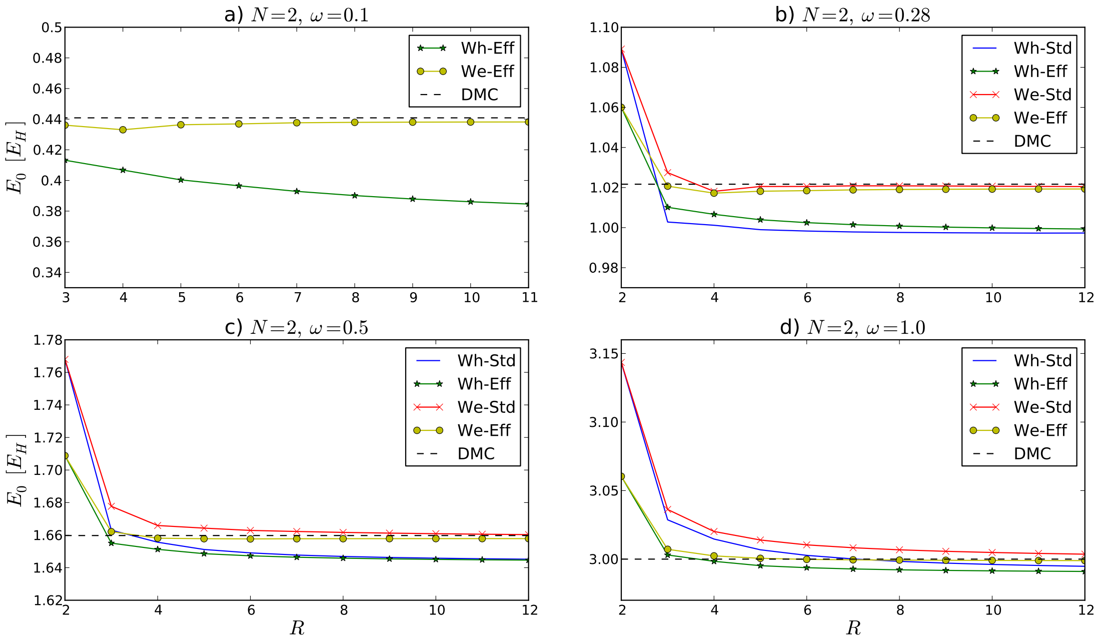


Figure 9.16: Comparison of the convergence behaviour as function of the shell number R with Wegner's generator, standard interaction (We-Std) and effective interaction (We-Eff), and White's generator with standard (Wh-Std) and effective interaction (Wh-Eff). All runs have been performed with $N = 2$ particles and Hartree-Fock basis. For benchmarking, we have also plotted the ground state energy we got with Diffusion Monte Carlo (DMC).

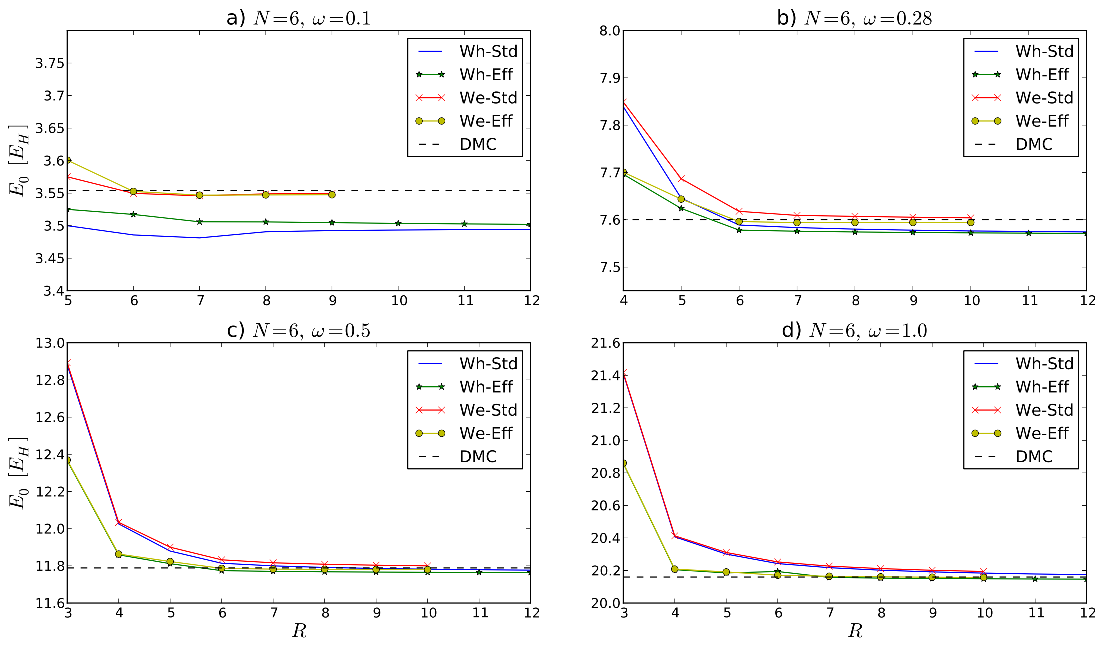


Figure 9.17: Comparison of the convergence behaviour as function of the shell number R with Wegner's generator, standard interaction (We-Std) and effective interaction (We-Eff), and White's generator with standard (Wh-Std) and effective interaction (Wh-Eff). All runs have been performed with $N = 6$ particles and Hartree-Fock basis. For benchmarking, we have also plotted the ground state energy we got with Diffusion Monte Carlo (DMC).

R	5	7	10	12	15	17	20	22	25
n	6640	57602	613572	2092438	9495322	22274038	67647814	130003774	312789734

Table 9.21: Number of coupled ordinary differential equations n that have to be solved for $N = 2$ particles.

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 9.22: Required CPU time (in hours) for runs with $\omega = 1.0$, Hartree-Fock basis and effective interaction. Here, we used the computing cluster "Abel", the Linux cluster at the University of Oslo-

tion to the ground state energy E_0. A similar trend can be observed for $N = 6$ particles and low values for ω in figure 9.17. In these cases, the results with White's generator are constantly slightly too low, suggesting that due to truncation, systematically a positive term is omitted. However, as the number of particles N and oscillator frequency ω is increased, the approximation seems to get better.

Since the flow equations with Wegner's generator get stiff and often do not converge as the number of particles N is increased, we continue our larger computations with this 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.

To get an impression of the dimension of the problem to be solved, table 9.21 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 to analyse the convergence behaviour as function of R, 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 9.22.

9.3.4 Comparison with other many-body methods

In the following, we will examine how our SRG results behave as function of shells R and rate them with respect to other many-body methods. In doing so, we will consider the following methods:

- **Hartree-Fock** (HF): The HF code is written by us and precedes the SRG calculation, such that we can use a HF basis. Regarding theoretical aspects of the method, we refer to section 7.1.
- **Diffusion Monte Carlo** (DMC): We developed an own DMC code, independently of the SRG implementation. As explained in section 7.2, this quantum Monte Carlo method solves Schrödinger equation using a Green's function.
- **Full Configuration Interaction** (FCI): These results are supplied by the former master's student F. B. Olsen, see [50]. 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 benchmark for our results. However, only systems with rather small numbers of included shells R can be compared, since the number of determinants in the basis grows factorially with the number of particles and orbitals. For N electrons in 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 [52]. The largest system run here, with $N = 30$ particles and $R = 20$ shells, corresponding to 420 basis functions, yields approximately $6.5 \cdot 10^{45}$ determinants, which is much too large for an exact diagonalization.
- **Coupled Cluster**: The coupled cluster data we compare our results with, are supplied by the former master's student C. Hirth, see [41]. 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 (single excitations) operator, two-particle two-hole (double excitations) operator 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 [7].
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 errors made by the different methods.

Figures 9.19 to 9.21 show our IM-SRG(2) results as functions of the shell number R and compare them with other many-body methods. First, we consider oscillator frequency $\omega = 1.0$, afterwards we look at lower values for ω.
A first important observation is that with increasing R, the ground state energy E_0 is decreasing and seems to converge. For the comparison with other methods, let us start with the Hartree-Fock calculations:
Evidently, SRG performs much better. The curve of the Hartree-Fock results lies considerably more off the ones of FCI and DMC than our SRG curve does. This meets our expectations, since Hartree-Fock calculations only consider one-particle one-hole excitations, while we also take two-particle two-hole excitations into account.
A next interesting point is how SRG compares to FCI, which can be regarded as exact for a given value of R: Our results lie systematically slightly below the FCI results, suggesting that a postive term is omitted when truncating to IM-SRG(2). Since FCI is limited to rather small basis sizes, we have unfortunately not that many data for comparison available.
However, for larger values of R, we can evaluate our results with respect to DMC and CCSD calculations. In general, the deviation of the SRG from the DMC results are comparable with the ones of CCSD. For larger number of particles, SRG seems even to perform slightly better. However, it is important to remark that also DMC only serves as guideline and does not give the exact result, since it is subject to errors like the fixed-node approximation. For a direct comparison between SRG and CCSD, up to $N = 30$ particles, we also refer to table B.14 in Appendix B.
For lower values of the oscillator frequency ω, that is higher correlations, the deviations of the SRG to the DMC and FCI results get larger, but they still stay comparable to CCSD. In general, the calculations with standard interaction perform slightly better than the ones with effective interaction. One should note that CCSD, although truncated to single and double excitations, implicitly includes higher excitations, too. This arises from the exponential ansatz and explains the overall good performance of the method.
Essentially, our SRG results compare really well with the other many-body methods. In particular when comparing to Hartree-Fock, it gets evident how important the contributions of two-particle two-hole excitations are. Obviously, they are much larger than the corrections coming from higher excitations, which supports our choice to truncate the SRG equations on a two-body level. Expanding our method to IM-SRG(3), IM-SRG(4), etc. is in principle straightforward, and we would then expect to lie even closer to the FCI evaluations. As shown before, for $N = 2$ particles, already including the loop terms of three-body interactions is enough to obtain those exact results.
A full listing of all our results, up to $N = 42$ particles, can be found in Appendix A.

		$\omega = 1.0$			$\omega = 0.28$		
N	R	SRG	FCI	Δ_{FCI}	SRG	FCI	Δ_{FCI}
2	5	3.0068	3.0176	0.0108 (0.36%)	0.9990	1.0266	0.0276 (2.69%)
	8	2.9984	3.0092	0.0108 (0.36%)	0.9976	1.0242	0.0266 (2.60%)
	10	2.9961	3.0069	0.0108 (0.36%)	0.9973	1.0218	0.0245 (2.40%)
	20	2.9923	3.0030	0.0107 (0.36%)	0.9971	1.0225	0.0254 (2.48%)
6	5	20.3009	20.3167	0.016 (0.078%)	7.6456	7.7105	0.0648 (0.84%)
	8	20.2020	20.2164	0.014 (0.071%)	7.5802	7.6155	0.0353 (0.53%)

Table 9.23: Comparison between the ground state energy E_0 (in $[E_H]$) obtained with SRG and FCI [50], both with standard interaction. In the column labelled with Δ_{FCI} we give the absolute and relative difference, $\left|E_{0(SRG)} - E_{0(FCI)}\right|$ and $\left|E_{0(SRG)} - E_{0(FCI)}\right| / E_{0(FCI)}$, respectively.

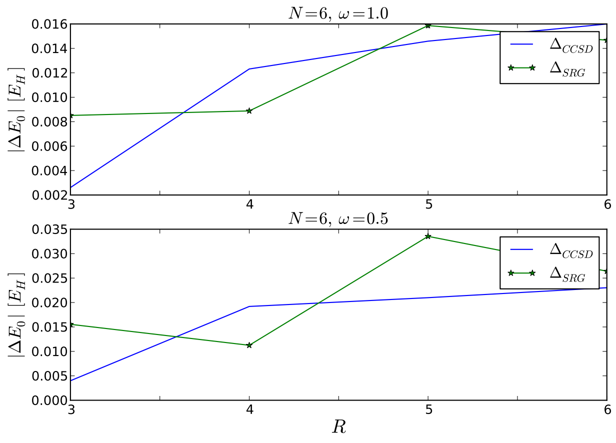
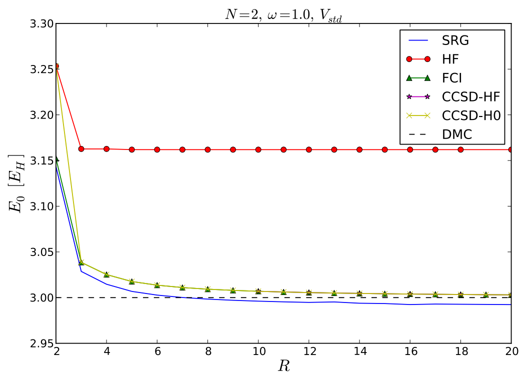
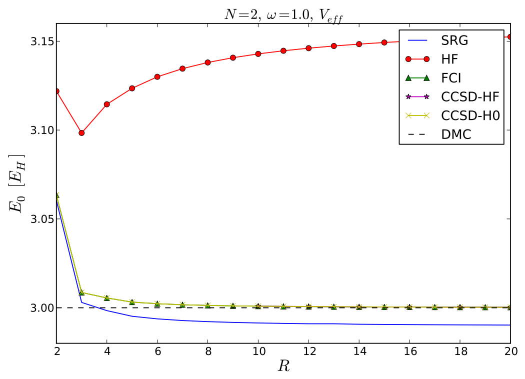


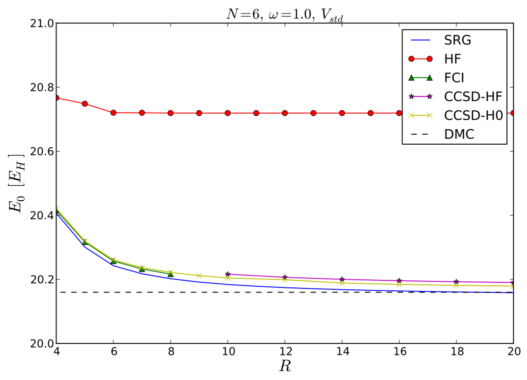
Figure 9.18: Difference of the ground state energy E_0 from the obtained DMC result. For the Coupled Cluster result, this difference is given by $\Delta_{CCSD} = |E_{0(CCSD)} - E_{0(DMC)}|$, with an analogous expression for the SRG result. The runs have been performed with standard interaction and Hartree-Fock basis. The CCSD results are taken from [41].



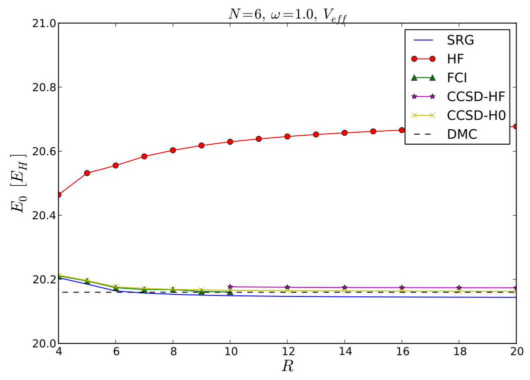
(a) $N = 2, \omega = 1.0$, Std. interaction



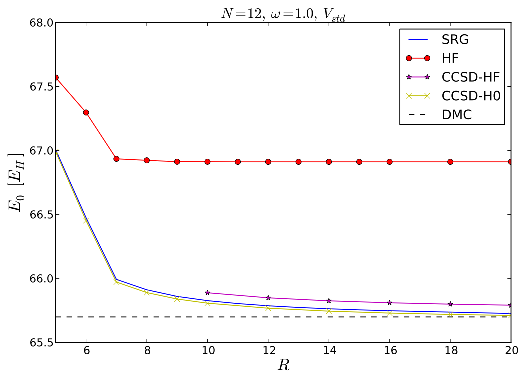
(b) $N = 2, \omega = 1.0$, Eff. interaction



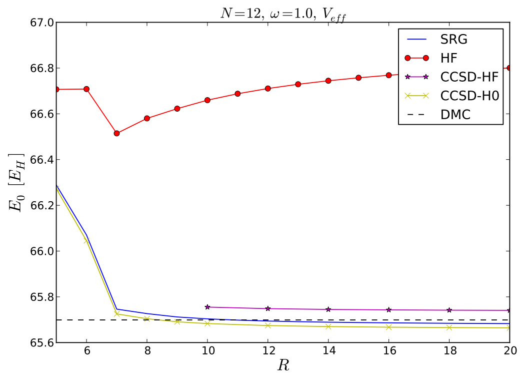
(c) $N = 6, \omega = 1.0$, Std. interaction



(d) $N = 6, \omega = 1.0$, Eff. interaction

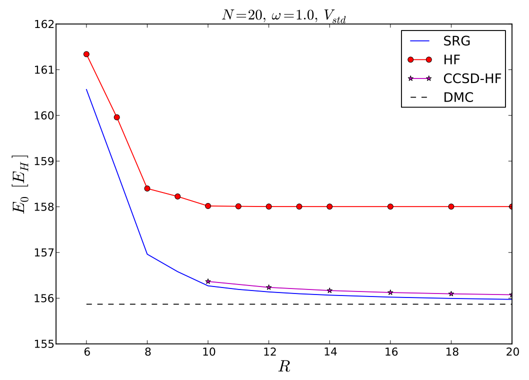


(e) $N = 12, \omega = 1.0$, Std. interaction

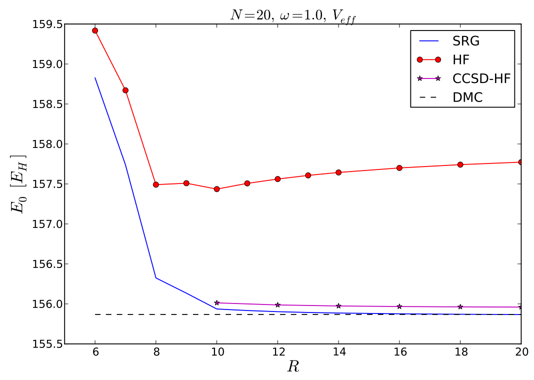


(f) $N = 12, \omega = 1.0$, Eff. interaction

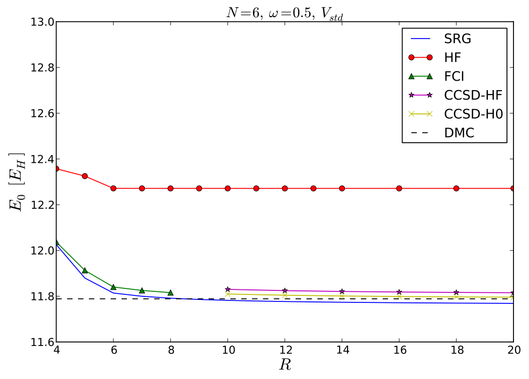
Figure 9.19: Comparison of our SRG results with other many-body methods. We performed the SRG calculations (IM-SRG(2)) with White's generator and a Hartree-Fock basis. The Hartree-Fock (HF) and Diffusion Monte Carlo (DMC) results have been obtained with code developed with us. The Full Configuration Interaction (FCI) results are taken from [50] and the Coupled Cluster (CCSD) results from [41]. The Coupled Cluster calculations include singles and doubles and are plotted both with harmonic oscillator basis (CCSD-HO) and Hartree-Fock basis (CCSD-HF).



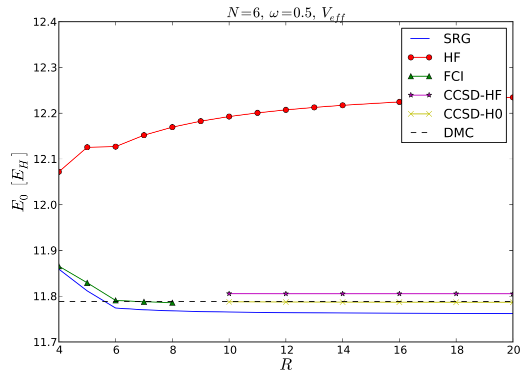
(a) $N = 20, \omega = 1.0$, Std. interaction



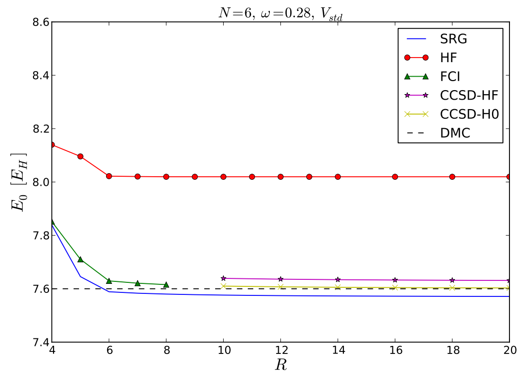
(b) $N = 20, \omega = 1.0$, Eff. interaction



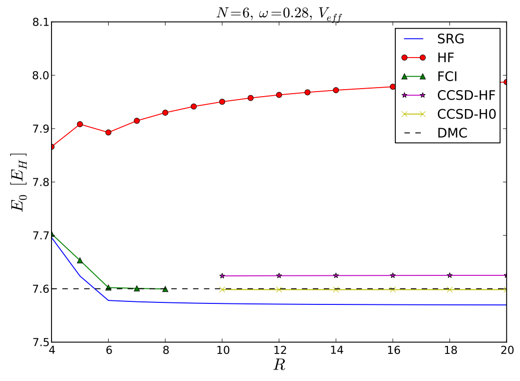
(c) $N = 6, \omega = 0.5$, Std. interaction



(d) $N = 6, \omega = 0.5$, Eff. interaction

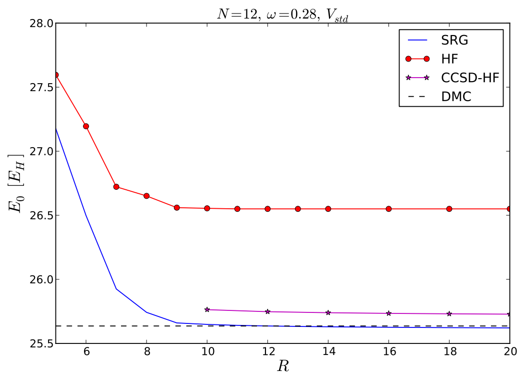


(e) $N = 6, \omega = 0.28$, Std. interaction

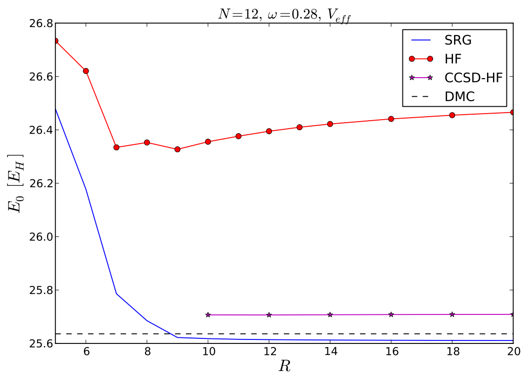


(f) $N = 6, \omega = 0.28$, Eff. interaction

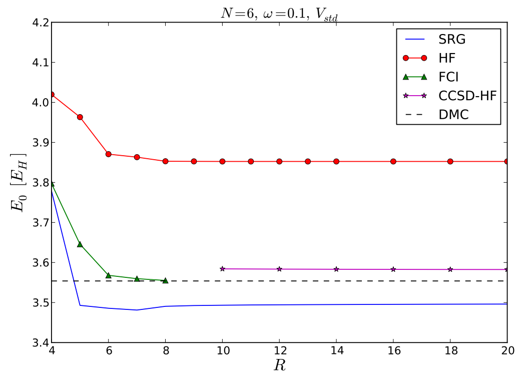
Figure 9.20: Same caption as figure 9.19. First we continue with $\omega = 1.0$ for $N = 20$ particles, then we give examples for lower values of the oscillator frequency ω. For $N = 20$ particles, we had not enough FCI and CCSD-HO data available to include them in the figures.



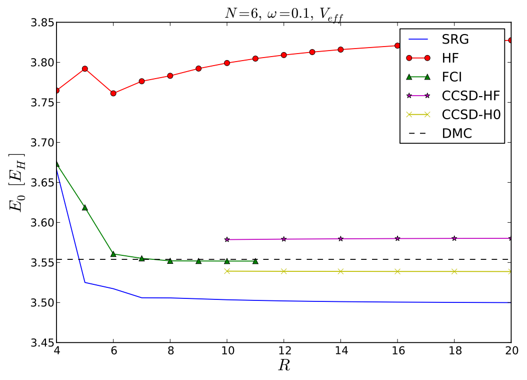
(a) $N = 12, \omega = 0.28$, Std. interaction



(b) $N = 12, \omega = 0.28$, Eff. interaction



(c) $N = 6, \omega = 0.1$, Std. interaction



(d) $N = 6, \omega = 0.1$, Eff. interaction

Figure 9.21: Same caption as figure 9.19, here examples with lower values of the oscillator frequency ω. In the cases where no FCI and CCDS-HO data are included in the plots, we had not enough data available.

Chapter 10

Conclusions

Appendix A

Basic commutation relations

When deriving the flow equations for IM-SRG, both the generator $\hat{\eta}$ and the final flow 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 of the expression for Wegner's canonical generator, $\hat{\eta} = \left[\hat{H}^{\mathrm{d}}, \hat{H}\right]$, which suggests computing

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

where the amplitudes of the diagonal Hamiltonian are defined as

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

For instance, for the first term, $\left[\sum\limits_{pq} f^d_{pq}\{a^\dagger_p a_q\}, \sum\limits_{pq} f_{pq}\{a^\dagger_p a_q\}\right]$, Wick's theorem yields the following contributions for the fully contracted part:

$$\begin{aligned}
\left[\sum_{pq}\{a^\dagger_p a_q\} f^d_{pq}, \sum_{pq}\{a^\dagger_p a_q\} f_{pq}\right] &= \left[\sum_{pq}\{a^\dagger_p a_q\} f^d_{pq}, \sum_{rs}\{a^\dagger_r a_s\} f_{rs}\right] \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left[\{a^\dagger_p a_q\}, \{a^\dagger_r a_s\}\right] \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left(\{a^\dagger_p a_q\}\{a^\dagger_r a_s\} - \{a^\dagger_r a_s\}\{a^\dagger_p a_q\}\right) \\
&= \sum_{pqrs} f^d_{pq} f_{rs}\left(\{\overbrace{a^\dagger_p \overbrace{a_q a^\dagger_r} a_s}\} - \{\overbrace{a^\dagger_r \overbrace{a_s a^\dagger_p} a_q}\}\right) + \text{non-relevant contractions} \\
&= \sum_{ia}\left(f^d_{ia} f_{ai} - f^d_{ai} f_{ia}\right) + \text{non-relevant contractions},
\end{aligned}$$

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 those basic commutation relations that are needed to derive the IM-SRG equations, up to third-order level.
Orienting on the notation for our Hamiltonian, we denote the one-, two and three-body operators as

$$\begin{aligned}
\hat{F} &= \sum_{pq} f_{pq}\{a_p^\dagger a_q^\dagger\}, \\
\hat{V} &= \frac{1}{(2!)^2}\sum_{pqrs} v_{pqrs}\{a_p^\dagger a_q a_s a_r\}, \\
\hat{W} &= \frac{1}{(3!)^2}\sum_{pqrstu} w_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\},
\end{aligned} \tag{A.1}$$

and we assume antisymmetrized elements:

$$v_{pqrs} = -v_{qprs} = -v_{pqsr} = v_{qpsr} \tag{A.2}$$

$$\begin{aligned}
w_{pqrstu} &= -w_{qprstu} = -w_{rqpstu} = -w_{prqstu} \\
&= -w_{pqrtsu} = -w_{pqruts} = -w_{pqrsut} \\
&= w_{qprtsu} = w_{rqptsu} = w_{prqtsu} \\
&= w_{qpruts} = w_{rqputs} = w_{prquts} \\
&= w_{qprsut} = w_{rqpsut} = w_{prqsut}.
\end{aligned} \tag{A.3}$$

With the permutation operators

$$P_{pq} f(p,q) = f(q,p), \quad P(pq/r) = 1 - P_{pq} - P_{qr}, \quad P(p/qr) = 1 - P_{pq} - P_{pr},$$

the fully contracted terms are

$$[\hat{F}^A, \hat{F}^B]^{(0)} = \sum_{ia}\left((1 - P_{ia}) f^A_{ia} f^B_{ai}\right) \tag{A.4}$$

$$[\hat{V}^A, \hat{V}^B]^{(0)} = \frac{1}{4}\sum_{ijab}\left(v^A_{ijab} v^B_{abij} - v^B_{ijab} v^A_{abij}\right) \tag{A.5}$$

$$[\hat{W}^A, \hat{W}^B]^{(0)} = \frac{1}{36}\sum_{ijkabc}\left(w^A_{ijkabc} w^B_{abcijk} - w^A_{abdijk} w^B_{ijkabc}\right), \tag{A.6}$$

the one-body terms are

$$[\hat{F}^A, \hat{F}^B]^{(1)} = \sum_{pq} \{a_p^\dagger a_q\} \sum_r \left(f_{pr}^A f_{rq}^B - f_{pr}^B f_{rq}^A \right) \tag{A.7}$$

$$[\hat{F}, \hat{V}]^{(1)} = \sum_{pq} \{a_p^\dagger a_q\} \sum_{ia} (1 - P_{ia})\, f_{ia} v_{apiq} \tag{A.8}$$

$$\begin{aligned}[\hat{V}^A, \hat{V}^B]^{(1)} = \sum_{pq} \{a_p^\dagger a_q\} \Bigg\{ &\frac{1}{2} \sum_{abi} \left(v_{ipab}^A v_{abiq}^B - v_{ipab}^B v_{abiq}^A \right) \\ &+ \frac{1}{2} \sum_{ija} \left(v_{apij}^A v_{ijaq}^B - v_{apij}^B v_{ijaq}^A \right) \Bigg\}\end{aligned} \tag{A.9}$$

$$[\hat{V}, \hat{W}]^{(1)} = \sum_{pq} \{a_p^\dagger a_q\} \left\{ \frac{1}{4} \sum_{abij} \left(v_{ijab} w_{abpqij} - v_{abij} w_{ijpqab} \right) \right\} \tag{A.10}$$

$$\begin{aligned}[\hat{W}^A, \hat{W}^B]^{(1)} = \sum_{pq} \{a_p^\dagger a_q\} \Bigg\{ &\frac{1}{12} \sum_{ijabc} \left(w_{ijpabc}^A w_{abcijq}^B - w_{ijpabc}^B w_{abcijq}^A \right) \\ &+ \frac{1}{12} \sum_{abijk} \left(w_{abpijk}^A w_{ijkabq}^B - w_{abpijk}^B w_{ijkabq}^A \right) \Bigg\},\end{aligned} \tag{A.11}$$

the two-body terms are

$$[\hat{F}, \hat{V}]^{(2)} = \frac{1}{4} \sum_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\} \sum_t \left((1 - P_{pq}) f_{pt} v_{tqrs} - (1 - P_{rs}) f_{tr} v_{pqts} \right) \tag{A.12}$$

$$[\hat{F}, \hat{W}]^{(2)} = \sum_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\} \sum_{ia} (1 - P_{ia}) f_{ia} w_{apqirs} \tag{A.13}$$

$$\begin{aligned}[\hat{V}^A, \hat{V}^B]^{(2)} = \frac{1}{4} \sum_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\} \Bigg\{ &\frac{1}{2} \sum_{ab} \left(v_{pqab}^A v_{abrs}^B - v_{pqab}^B v_{abrs}^A \right) \\ &- \sum_{ij} \left(v_{pqij}^A v_{ijrs}^B - v_{pqij}^B v_{ijrs}^A \right) \\ &+ \sum_{ia} (1 - \hat{P}_{ia})(1 - \hat{P}_{pq} - \hat{P}_{rs} + \hat{P}_{pq}\hat{P}_{rs}) v_{ipar}^A v_{aqis}^B \Bigg\}\end{aligned} \tag{A.14}$$

$$\begin{aligned}[\hat{V}, \hat{W}]^{(2)} = \frac{1}{4} \sum_{pqrs} \{a_p^\dagger a_q^\dagger a_s a_r\} \Bigg\{ &\frac{1}{2} \sum_{iab} (1 - \hat{P}_{pq}\hat{P}_{pr}\hat{P}_{qs} - \hat{P}_{rs} + \hat{P}_{pr}\hat{P}_{qs}) v_{abri} w_{ipqabs} \\ &+ \frac{1}{2} \sum_{aij} (1 - \hat{P}_{pq}\hat{P}_{pr}\hat{P}_{qs} - \hat{P}_{rs} + \hat{P}_{pr}\hat{P}_{qs}) v_{ijra} w_{apqijs} \Bigg\}\end{aligned} \tag{A.15}$$

$$
\begin{aligned}
[\hat{W}^A, \hat{W}^B]^{(2)} = \frac{1}{4}\sum_{pqrs}\{a_p^\dagger a_q^\dagger a_s a_r\} \Bigg\{ & \frac{1}{6}\sum_{iabc}\left(w^A_{ipqabc}w^B_{abcirs} - w^B_{ipqabc}w^A_{abcirs}\right) \\
& - \frac{1}{6}\sum_{aijk}\left(w^A_{apqijk}w^B_{ijkars} - w^B_{apqijk}w^A_{ijkars}\right) \\
& + \frac{1}{4}\sum_{abij}(1-\hat{P}_{pq})(1-\hat{P}_{rs})\left(w^A_{abpijs}w^B_{ijqabr} - w^A_{ijpabs}w^B_{abqijr}\right)\Bigg\} \qquad \text{(A.16)}
\end{aligned}
$$

and the three-body terms are

$$
\begin{aligned}
[\hat{F}, \hat{W}]^{(3)} &= \frac{1}{36}\sum_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\}\Bigg\{\sum_t (1-\hat{P}_{pq}-\hat{P}_{pr})f_{pt}w_{tqrstu} \\
&\quad -\sum_t (1-\hat{P}_{st}-\hat{P}_{su})f_{ts}w_{tpqrtu}\Bigg\} \qquad \text{(A.17)} \\
[\hat{V}^A, \hat{V}^B]^{(3)} &= \frac{1}{36}\sum_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\}\hat{P}(pq/r)\hat{P}(s/tu)\sum_v (v^A_{pqsv}v^B_{vrtu} - v^B_{pqsv}v^A_{vrtu}) \qquad \text{(A.18)} \\
[\hat{V}, \hat{W}]^{(3)} &= \frac{1}{36}\sum_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\} \qquad \text{(A.19)} \\
&\quad \Bigg\{\sum_{ab}\Big(\hat{P}(pq/r)v_{pqab}w_{abrstu} - \hat{P}(s/tu)v_{abtu}w_{pqrsab}\Big) \\
&\quad -\sum_{ij}\Big(\hat{P}(pq/r)v_{pqij}w_{ijrstu} - \hat{P}(s/tu)v_{ijtu}w_{pqrsij}\Big) \qquad \text{(A.20)} \\
&\quad -\sum_{ia}\Big(1-\hat{P}_{ia})\Big)\, v_{apis}w_{iqratu}\Bigg\} \qquad \text{(A.21)} \\
[\hat{W}^A, \hat{W}^B]^{(3)} &= \frac{1}{36}\sum_{pqrstu}\{a_p^\dagger a_q^\dagger a_r^\dagger a_u a_t a_s\} \\
&\quad \Bigg\{\frac{1}{6}\sum_{ijk}\left(w^A_{pqrijk}w^B_{ijkstu} - w^B_{pqrijk}w^A_{ijkstu}\right) \\
&\quad + \frac{1}{6}\sum_{abc}\left(w^A_{pqrabc}w^B_{abcstu} - w^B_{pqrabc}w^A_{abcstu}\right) \\
&\quad + \frac{1}{2}\sum_{ija}\hat{P}(pq/r)\hat{P}(s/tu)\left(w^A_{ijratu}w^B_{apqijs} - w^A_{aqriju}w^B_{pijsta}\right) \\
&\quad + \frac{1}{2}\sum_{abi}\hat{P}(pq/r)\hat{P}(s/tu)\left(w^A_{abritu}w^B_{ipqabs} - w^A_{iqrabu}w^B_{pabsti}\right)\Bigg\}. \qquad \text{(A.22)}
\end{aligned}
$$

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	0.0762438977	0.0762438977	0.0762438977
		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	0.441183487	0.441183487	0.441183487
		10	0.441135127	0.441135127	0.441135127

Table B.1: The ground state energy E_0 (in $[E_H]$) for $N = 2$ particles is compared between 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.

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	1.78691353	1.78691353	1.78691353
		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	3.01760623	3.01760623	3.01760623
		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 IM-SRG(2) results with White's generator

ω	R	$N = 2$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	2	3.253314	3.143526	3.121868	3.060394	3.00000(3)
	3	3.162691	3.028667	3.09837	3.003012	
	4	3.162691	3.014642	3.114506	2.998399	
	5	3.161921	3.006803	3.123489	2.995219	
	6	3.161921	3.002745	3.129965	2.993752	
	7	3.161909	3.000139	3.134571	2.992814	
	8	3.161909	2.998368	3.138031	2.992198	
	9	3.161909	2.997087	3.140718	2.991764	
	10	3.161909	2.996119	3.142864	2.991443	
	11	3.161909	2.995364	3.144617	2.991198	
	12	3.161909	2.994759	3.146076	2.991004	
	13	3.161909	2.995264	3.147308	2.990848	
	14	3.161909	2.993852	3.148364	2.990721	
	15	3.161908	2.993504	3.149277	2.990614	
	16	3.161908	2.992395	3.150075	2.9905232	
	17	3.161908	2.992947	3.150779	2.9904456	
	18	3.161908	2.992722	3.151404	2.9903784	
	20	3.161908	2.992348	3.152465	2.9902683	
0.5	2	1.886227	1.767956	1.762485	1.708804	1.65975(3)
	3	1.799856	1.663054	1.743415	1.655092	
	4	1.799856	1.655645	1.757266	1.651281	
	5	1.799748	1.651188	1.765537	1.648611	
	6	1.799748	1.649093	1.771214	1.647293	
	7	1.799745	1.647760	1.775285	1.646425	
	8	1.799745	1.646903	1.778345	1.645839	
	9	1.799743	1.646291	1.780727	1.645417	
	10	1.799743	1.645836	1.782634	1.645099	
	11	1.799742	1.645487	1.784195	1.644852	
	12	1.799742	1.645211	1.785497	1.644654	
	13	1.799742	1.644987	1.786598	1.644493	
	14	1.799742	1.644802	1.787541	1.644359	
	16	1.799742	1.644514	1.789074	1.644149	
	18	1.799742	1.644302	1.790266	1.643992	
	20	1.799742	1.644139	1.791219	1.643871	

Table B.3: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot with $N = 2$ 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 got with Diffusion Monte Carlo (DMC).

ω	R	$N = 2$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	2	1.223192	1.089121	1.107677	1.060039	1.02166(3)
	3	1.141775	1.002780	1.092266	1.010128	
	4	1.141775	1.001169	1.104149	1.006616	
	5	1.141741	0.998958	1.111458	1.003919	
	6	1.141741	0.998286	1.116402	1.002470	
	7	1.141717	0.997825	1.119956	1.001461	
	8	1.141717	0.997582	1.122643	1.000762	
	9	1.141713	0.997428	1.124739	1.000244	
	10	1.141713	0.997324	1.126424	0.999850	
	11	1.141712	0.997254	1.127805	0.999539	
	12	1.141712	0.997294	1.128958	0.999287	
	13	1.141712	0.997167	1.129935	0.999080	
	14	1.141712	0.997138	1.130774	0.998907	
	16	1.141712	0.997098	1.132138	0.998632	
	18	1.141712	0.997071	1.133201	0.998425	
	20	1.141712	0.997052	1.134051	0.998263	
0.1	2	0.5963327	nc	0.4987856	0.455250	0.44079(3)
	3	0.5269028	0.361924	0.4891241	0.4132006	
	4	0.5269028	0.372295	0.4977424	0.406776	
	5	0.5256662	nc	0.5027652	0.400372	
	6	0.5256662	nc	0.5063809	0.396534	
	7	0.5256354	nc	0.5089927	0.392878	
	8	0.5256354	nc	0.5109902	0.390129	
	9	0.5256348	nc	0.5125584	0.387898	
	10	0.5256348	nc	0.5138233	0.386095	
	11	0.5256347	nc	0.5148646	0.384605	
	12	0.5256347	nc	0.5157369	0.383332	
	13	0.5256347	nc	0.5164781	0.382191	
	14	0.5256347	nc	0.517119	0.381168	
	16	0.5256347	nc	0.5181570	nc	
	18	0.5256347	nc	0.518971	nc	
	20	0.5256347	nc	0.519625	nc	

Table B.4: Continuation of table B.3, see corresponding caption for explanation.

ω	R	$N = 6$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
1.0	3	21.59320	21.41211	20.9823	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	13.0516	12.88169	12.47376	12.36581	11.7888(2)
	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.5: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot 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 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 got 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	7.570217	
	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.6: Continuation of table B.5, see corresponding caption for explanation.

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		66.80063	65.68304	
0.5	4	43.66327	43.24100	42.13168	nc	
	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	39.14673	40.02056	39.20457	
	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.7: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot 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 got 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	
	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	12.2138	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.8: Continuation of table B.7, see corresponding caption for explanation.

ω	R	$N = 20$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
	5	169.3217	nc	166.0855	nc	
	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	
1.0	11	158.0103	156.192	157.5071	155.9170	155.868(6)
	12	158.0050	156.1371	157.5613	155.9026	
	13	158.0048	156.0968	157.60693	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	157.77252	155.8665	
	5	106.2185	nc	103.0094	nc	
	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	
0.5	11	95.78579	94.09619	95.32660	93.92273	
	12	95.7346	94.020364	95.34072	93.880216	
	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.9: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot 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. For benchmarking, the last column shows the results that we got with Diffusion Monte Carlo (DMC).

ω	R	$N = 20$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	5	73.23369	nc	70.08685	nc	
	6	67.90736	67.31049	65.92851	65.53328	
	7	66.33661	65.35869	64.99235	nc	
	8	64.75479	63.6664	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	63.53944	61.91856	
	16	63.53881	61.95845	63.30323	61.91600	
	18	63.53880	61.94968	63.33695	61.91456	
	20	63.5388	61.94362	63.36208	61.91368	
0.1	5	39.20839	nc	36.27986	nc	
	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.10: Continuation of table B.9, 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	308.562(Jørgen)
	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	187.043(Jørgen)
	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	187.16705	189.5672	187.0408	

Table B.11: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot 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 colums with effective interaction. For benchmarking, the last column shows the results that we got with Diffusion Monte Carlo (DMC).

ω	R	$N = 30$				
		Standard		Effective		
		HF	SRG	HF	SRG	DMC
0.28	6	149.7016	nc	144.1397	nc	123.971(Jørgen)
	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	60.422 (Jørgen)
	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	61.9869	60.43000	

Table B.12: Continuation of table B.11, 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	
	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	
0.5	7	386.8765	nc	377.9255	nc	
	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	331.70316	
0.28	7	270.9290	nc	262.1197	nc	
	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			

Table B.13: Ground state energy E_0 (in $[E_H]$) for a system a circular quantum dot 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 colums with effective interaction

B.3 Comparison with Coupled Cluster results

		$\omega = 1.0$			$\omega = 0.28$		
N	R	SRG	CCSD	Δ_{CCSD}	SRG	CCSD	Δ_{CCSD}
2	10	2.9961	3.0069	0.0108 (0.36%)	0.99732	1.0236	0.0262 (2.57%)
	14	2.9939	3.0046	0.0107 (0.36%)	0.99714	1.0229	0.0258 (2.52%)
	20	2.9923	3.0030	0.0107 (0.35%)	0.99705	1.0225	0.0254 (2.49%)
6	10	20.1841	20.2161	0.0320 (0.16%)	7.5763	7.6390	0.0627 (0.82%)
	14	20.1680	20.2000	0.0320 (0.16%)	7.5730	7.6341	0.0610 (0.80%)
	20	20.1583	20.1901	0.0318 (0.16%)	7.5713	7.6312	0.0599 (0.79%)
12	10	65.8264	65.8880	0.0616 (0.093%)	25.6484	25.7634	0.1195 (0.46%)
	14	65.7628	65.8250	0.0623 (0.094%)	25.6299	25.7396	0.1098 (0.43%)
	20	65.7260	65.7909	0.0649 (0.098%)	25.6212	25.7284	0.1072 (0.21%)
20	10	156.2704	156.3659	0.0955 (0.061%)	62.3502	62.5234	0.1732 (0.28%)
	14	156.0665	156.1671	0.1006 (0.064%)	61.9723	62.1476	0.1753 (0.28%)
	20	155.9737	156.0750	0.1013 (0.065%)	61.94362	62.1134	0.1698 (0.27%)
30	10	310.8837	310.0024	0.8813 (0.28%)	127.8474	126.9647	0.8827 (0.70%)
	14	308.9924	308.7310	0.2614 (0.084%)	124.3451	124.3890	0.0439 (0.035%)
	20	308.7673	308.6894	0.0779 (0.025%)	-	[no data available]	

Table B.14: Comparison between the ground state energy E_0 (in $[E_H]$) obtained with SRG and CCSD (results from [41]), both with standard interaction and Hartree-Fock basis. In the column labelled with Δ_{CCSD} we give the absolute and relative difference, $\left|E_{0(SRG)} - E_{0(CCSD)}\right|$ and $\left|E_{0(SRG)} - E_{0(CCSD)}\right| / E_{0(CCSD)}$, respectively. In general, the relative difference is decreasing as the number of shells R is increased.

Bibliography

[1] R. Liboff, *Introductory quantum mechanics.* Addison-Wesley Pub. Co., 1992. [Online]. Available: http://books.google.no/books?id=VzZRAAAAMAAJ

[2] D. Griffiths, *Introduction to quantum mechanics.* Pearson Prentice Hall, 2005. [Online]. Available: http://books.google.no/books?id=-BsvAQAAIAAJ

[3] M. Gaberdiel, "Quantenmechanik I, Lecture notes," Department of Theoretical Physics, ETH-Hönggerberg, Zuerich, 2006.

[4] G. Rudolph, "Vorlesungen zur Quantenmechanik I," Department of Theoretical Physics, University of Leipzig, 2009.

[5] D. Hilbert, J. Neumann, and L. Nordheim, "Über die Grundlagen der Quantenmechanik," *Mathematische Annalen*, vol. 98, pp. 1–30, 1928. [Online]. Available: http://dx.doi.org/10.1007/BF01451579

[6] P. Dirac, "A new notation for quantum mechanics," *Math. Proc. Cambridge*, vol. 35, pp. 416–418, 1939.

[7] I. Shavitt and R. Bartlett, *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*, ser. Cambridge Molecular Science. Cambridge University Press, 2009.

[8] W. Nolting, *Grundkurs Theoretische Physik 7: Viel-Teilchen-Theorie*, ser. Springer-Lehrbuch. Springer, 2009. [Online]. Available: http://books.google.no/books?id=xlKROLdCxzEC

[9] J. C. Slater, "The Theory of Complex Spectra," *Phys. Rev.*, vol. 34, pp. 1293–1322, Nov 1929. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRev.34.1293

[10] G. C. Wick, "The Evaluation of the Collision Matrix," *Physical Review*, vol. 80, pp. 268–272, Oct 1950.

[11] "Shampine and Gordon ODE Solver," [Online], http://people.sc.fsu.edu/ jburkardt/cpp_src/ode/ode.html.

[12] L. Shampine and M. Gordon, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem.* Freeman, 1975.

[13] J. W. Daniel and R. E. Moore, "Computation and theory in ordinary differential equations," 1970.

[14] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, ser. Springer Series in Computational Mathematics. Springer, 2011. [Online]. Available: http://books.google.no/books?id=tj3vAAAAMAAJ

[15] S. Kvaal, "Harmonic oscillator eigenfunction expansions, quantum dots, and effective interactions," *Phys. Rev. B*, vol. 80, p. 045321, Jul 2009. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevB.80.045321

[16] M. Rontani, C. Cavazzoni, D. Bellucci, and G. Goldoni, "Full configuration interaction approach to the few-electron problem in artificial atoms," *The Journal of Chemical Physics*, vol. 124, no. 12, p. 124102, 2006. [Online]. Available: http://link.aip.org/link/?JCP/124/124102/1

[17] S. D. Głazek and K. G. Wilson, "Renormalization of Hamiltonians," *Phys. Rev. D*, vol. 48, pp. 5863–5872, Dec 1993. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevD.48.5863

[18] S. D. Glazek and K. G. Wilson, "Perturbative renormalization group for hamiltonians," *Phys. Rev. D*, vol. 49, pp. 4214–4218, Apr 1994. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevD.49.4214

[19] F. J. Wegner, "Flow-equations for Hamiltonians," *Ann. Phys.*, vol. 3, pp. 77–91, 1994.

[20] ——, "Flow equations for Hamiltonians," *Phys. Rep.*, vol. 348, pp. 77–89, 2001.

[21] S. Kehrein, *The Flow Equation Approach to Many-Particle Systems*, ser. Springer Tracts in Modern Physics. Springer, 2006, no. 217. [Online]. Available: http://books.google.no/books?id=yfoxZrUsXFwC

[22] E. Anderson, S. Bogner, R. Furnstahl, and R. Perry, "Operator Evolution via the Similarity Renormalization Group I: The Deuteron," *Phys. Rev.C82:054001*, 2010. [Online]. Available: arXiv:1008.1569

[23] S. K. Bogner, R. J. Furnstahl, and R. J. Perry, "Similarity renormalization group for nucleon-nucleon interactions," *Phys. Rev. C*, vol. 75, p. 061001, Jun 2007. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevC.75.061001

[24] S. Bogner, R. Furnstahl, P. Maris, R. Perry, A. Schwenk, and J. Vary, "Convergence in the no-core shell model with low-momentum two-nucleon interactions," *Nuclear Physics A*, vol. 801, no. 1-2, pp. 21 – 42, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0375947407008147

[25] O. Åkerlund, E. 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," *The European Physical Journal A*, vol. 47, pp. 1–7, 2011. [Online]. Available: http://dx.doi.org/10.1140/epja/i2011-11122-4

[26] S. White, "A numerical canonical transformation approach to quantum many body problems," *J. Chem. Phys.*, vol. 117, 2002. [Online]. Available: http://dx.doi.org/10.1063/1.1508370

[27] H. Hergert, S. 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. [Online]. Available: arXiv:1212.1190

[28] K. Tsukiyama, "In-medium similarity renormalization group for nuclear many-body systems," Ph.D. dissertation, Department of Physics, The University of Tokyo, 2011.

[29] J. Thijssen, *Computational Physics.* Cambridge University Press, 2007. [Online]. Available: http://books.google.de/books?id=ThtNausKMn4C

[30] M. Pedersen Lohne, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva, "*Ab initio* computation of the energies of circular quantum dots," *Phys. Rev. B*, vol. 84, p. 115302, Sep 2011. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevB.84.115302

[31] B. Hammond, J. W. A. Lester, and P. Reynolds, *Monte Carlo Methods in Ab Initio Quantum Chemistry*, ser. World Scientific Lecture and Course Notes in Chemistry ; Vol. 1. World Scientific, 1994. [Online]. Available: http://books.google.no/books?id=d1Kbx2JVLw8C

[32] M. Hjorth-Jensen, "Computational Physics, Lecture Notes," Department of Physics, University of Oslo, 2011.

[33] D. A. Nissenbaum, "The stochastic gradient approximation: an application to li nanoclusters," Ph.D. dissertation, Northeastern University, 2008.

[34] J. NOCEDAL and S. Wright, *Numerical Optimization*, ser. Springer Series in Operations Research Series. Springer-Verlag GmbH, 1999. [Online]. Available: http://books.google.no/books?id=epc5fX0lqRIC

[35] "C++ language tutorial - C++ documentation," [Online].
Available: http://www.cplusplus.com/.

[36] "The C++ Programming Language," [Online].
Available: http://www.tutorialspoint.com/cplusplus/.

[37] S. Kvaal, "Open source FCI code for quantum dots and effective interactions," 2008. [Online]. Available: arXiv:0810.2644

[38] C. Gera, "The numerical integration of ordinary differential equations," *Math. Comp.*, vol. 21, pp. 146–156, 1967.

[39] M. Quinn, *Parallel programming in C with MPI and OpenMP*, ser. McGraw-Hill Higher Education. McGraw-Hill Higher Education, 2004. [Online]. Available: http://books.google.no/books?id=YrgZAQAAIAAJ

[40] M. 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.

[41] 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.

[42] Spring 2012, course FYS4411 (Computational physics II: Quantum mechanical systems), University of Oslo.

[43] W. Vetterling, *Numerical Recipes Example Book (C++)*, ser. Numerical recipes series. Cambridge University Press, 2002.

[44] C. Umrigar, M. Nightingale, and K. Runge, "A diffusion Monte Carlo algorithm with very small time-step errors," *J. Chem. Phys.*, vol. 99, 1993.

[45] B. Wells, in *Methods of Computational Chemistry*, S. Wilson ed., Wiley, New York, 1987, vol. 1.

[46] P. Reynolds, D. Ceperley, B. Alder, and W. Lester, *J. Chem. Phys.*, vol. 77, p. 5593, 1982.

[47] I. Kosztin, B. Faber, and K. Schulten, "Introduction to the Diffusion Monte Carlo method," *American Journal of Physics*, vol. 64, no. 5, pp. 633–644, 1996.

[48] "Armadillo, C++ linear algebra library," [Online]. Available: http://arma.sourceforge.net/.

[49] M. 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.

[50] F. 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.

[51] M. Taut, "Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem," *Phys. Rev. A*, vol. 48, pp. 3561–3566, Nov 1993. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.48.3561

[52] N. Shimizu, Y. Utsuno, T. Mizusaki, M. Honma, Y. Tsunoda *et al.*, "Variational procedure for nuclear shell-model calculations and energy-variance extrapolation," *Phys.Rev.*, vol. C85, p. 054301, 2012.