TECHNISCHE
UNIVERSITÄT
DARMSTADT

The Numerical Renormalization Group applied to the Bose Hubbard Model

Sven Binder

Bachelor Thesis

November 2007

Contents

Chapter 1

Introduction

In 1924 S.N. Bose worked out the statistical mechanics of a photon gas. In the sequent year this work was generalized by A. Einstein from massless photons to massive particles with integer spin. Einstein realized that these particles, today known as bosons, prefer sharing the same quantum state. For bosonic quantum gases he predicted the occurance phase transition from the domain of high temperatures where, according to Bose statistics, the bosons are spread over the whole momentum spectrum to a phase at low temperature, where the zero-momentum ground state is macroscopically occupied.

It took 70 years to attain the experimental technique for creating this novel form of matter. Its existence was proved in 1995 when Eric Cornell and Carl Wieman succeeded in the creation of this so-called Bose-Einstein condensate [1]. They used a diluted gas of rubidium atoms trapped in an magnetic trap cooled down to 170 nK.

Since then several properties of Bose-Einstein condensates (BEC) have been investigated. Besides the pure condensates there is a growing interest in BEC subjected to an optical lattice. These lattices can be created from standing optical waves produced by counterpropagating laser beams. In combination with Bose-Einstein condensates these lattices represent a valuable testing environment for strongly correlated quantum mechanical many-body systems. In the BEC phase, the atoms' wave functions are spread out over the whole lattice but a transition into a phase where the wave functions are localized at individual lattice sites can be observed. With the discovery of this Mott insulator phase that has been predicted by Jaksch et al. [2] a regime of ultracold bosonic matter beyond BEC has been reached.

In 1989 Fisher et al. applied the Hubbard model, known from solid-state physics, to bosonic quantum gases. This Bose-Hubbard model is quite simple but it reproduces the properties of BEC held in an optical lattice and in particular exhibits the BEC to Mott-insulator phase transition.

Although the Bose-Hubbard Hamiltonian can be in principle solved exactly

for any lattice system, the computational effort sets an upper limit. The exact solution is obtained by diagonalization of the Hamilton matrix. This matrix grows fast with the system size and so the diagonalization of the full Hamilton matrix soon can not be performed in reasonable time. For this reason this thesis investigates the numerical renormalization group (NRG) approach to the Hubbard model, an approach that avoids these large matrices and that therefore can be applied on problems practically inaccessible to the exact solution method.

Chapter 2

Basics

2.1 Many-Body Systems

Ultracold atoms in an optical lattice represent a quantum mechanical many-body system, in the case of this thesis one of identical bosonic atoms.

2.1.1 Distinguishable Particles

Consider a system of N distinguishable particles. Due to distinguishability it is possible to number these particles, and it makes sense regarding the mathematical treatment.

Let the single-particle Hilbert space of particle i be

$$\mathfrak{H}^{(i)} \tag{2.1}$$

with a complete basis

$$\left\{ |\alpha_j^{(i)}\rangle \right\} , \tag{2.2}$$

where $\alpha_j^{(i)}$ denotes the set of quantum numbers determining the ith particle's jth basis state.

The Hilbert space $\mathfrak{H}_N$ of the entire many-body system is given by the direct product of these single-particle Hilbert spaces

$$\mathfrak{H}_N = \mathfrak{H}^{(1)} \otimes \mathfrak{H}^{(2)} \otimes \cdots \otimes \mathfrak{H}^{(N)} \tag{2.3}$$

and a many-body basis $\{|\phi\rangle\}$ arises from the direct products of the single-particle basis states

$$|\phi_{k_1,k_2,\ldots,k_N}\rangle = |\alpha_{k_1}^{(1)}\rangle \otimes |\alpha_{k_2}^{(2)}\rangle \otimes \ \ldots \ \otimes |\alpha_{k_N}^{(N)}\rangle = |\alpha_{k_1}^{(1)} \ \alpha_{k_2}^{(2)} \ \ldots \ \alpha_{k_N}^{(N)}\rangle \ . \tag{2.4}$$

Any arbitrary eigenstate $|\psi_N\rangle$ of the N-particle system can be expanded in this product basis in the usual way:

$$|\psi_N\rangle = \sum_{k_1} \sum_{k_2} \cdots \sum_{k_N} \xi(k_1, k_2, \ldots, k_N) \ |\phi_{k_1,k_2,\ldots,k_N}\rangle \ . \tag{2.5}$$

2.1.2 Identical Particles

Different from classical mechanics, in quantum mechanics identical particles can not be distinguished from each other in principle. As a consequence of this a physical observable can not apply on single particles as before. Under these circumstances a numbering of the particles seems to be physically senseless though is necessary to be able to treat the problem mathematically. However, an exchange of two particles (and therefore any number of particles) must not affect measurable quantities.

Therefore, in the case of identical particles the observable quantities will be invariant under the exchange of particles. So a transposition operator $\hat{\pi}_{ij}$ can be defined that exchanges the ith and the jth particle. Obviously, exchanging two particles twice represents the identity operator

$$\hat{\pi}_{ij}^2 = \mathbb{1} \tag{2.6}$$

which leads to a restriction of the eigenvalues π of $\hat{\pi}_{ij}$

$$\pi = \pm 1 \ . \tag{2.7}$$

Therefore, besides states without a defined symmetry character, two types of many-body states exist: The symmetric ones, that are invariant under exchange of particles and the antisymmetric ones that exchange their signs when two particles are exchanged. From the spin-statistics theorem from quantum field theory it follows that the symmetric states are associated with bosonic particles whereas the antisymmetric states are associated with fermionic ones.

The Hilbert spaces of symmetric ($\mathfrak{H}^{(+)}$) and antisymmetric ($\mathfrak{H}^{(-)}$) states are subspaces of the Hilbert space $\mathfrak{H}_N$ constructed in (2.1.1). From there it makes sense to start with basis states from the $\mathfrak{H}_N$ and to (anti-) symmetrize them by the use of (anti-) symmetrization operators.

An (anti-) symmetrized basis state $|\phi_{k_1,k_2,\ldots,k_N}\rangle^{(\pm)}$ of $\mathfrak{H}^{(\pm)}$ can be constructed from basis states of $\mathfrak{H}_N$ according to

$$|\phi(k_1,\ldots,k_N)\rangle^{(\pm)} = \frac{1}{N!}\sum_{\mathfrak{P}}(\pm)^p \ \mathfrak{P} \ |\phi_{k_1,k_2,\ldots,k_N}\rangle \ , \tag{2.8}$$

where the sum runs over all possible permutations $\mathfrak{P}$ of the basis states $|\phi(k_1,\ldots,k_N)\rangle$. p denotes the number of transpositions $\hat{\pi}_{ij}$ the permutation $\mathfrak{P}$ consists of.

2.2 Second Quantization

The use of the (anti-) symmetrized states constructed via (2.8) is obviously very laborious. The Second Quantization's formalism facilitates working with these states.

The states constructed via (2.8) contain the information about which particles occupy which states, which is, however, due to indistinguishability meaningless. One is rather interested in how many particles occupy a certain state.

So it is suitable to express a many-body eigenstate in the occupation number representation using occupation numbers n_i

$$|\phi_{k_1,\ldots,k_N}\rangle^{(\pm)} = |n_1, n_2, \ldots, n_A\rangle^{(\pm)} \tag{2.9}$$

where n_i stands for the number of particles that occupy the single-particle state that has been labeled with i. An arbitrary N-particle state $|\psi_N\rangle$ then can be expressed via

$$|\psi_N\rangle = \sum_\alpha C_\alpha \; |\{n_1, n_2, \ldots, n_A\}_\alpha\rangle^{(\pm)} \; . \tag{2.10}$$

Although the representation of eigenstates is drastically simplified by the occupation number representation, the (anti-) symmetrized eigenstates still are completely determined by its occupation numbers.

A creation operator $\hat{a}_i^\dagger$ and an annihilation operator $\hat{a}_i$ that creates and annihilates, resp., a particle in the state $|i\rangle$ can be defined. In the case of bosons these operators must satisfy the commutator relations

$$\begin{aligned} &[\hat{a}_i, \hat{a}_j] = 0 \\ &[\hat{a}_i^\dagger, \hat{a}_j^\dagger] = 0 \\ &[\hat{a}_i, \hat{a}_j^\dagger] = \delta_{ij}. \end{aligned} \tag{2.11}$$

From these relations one can derive the action of these operators on states

$$\begin{aligned} \hat{a}_i|n_1, \ldots, n_i, \ldots, n_A\rangle &= \sqrt{n_i}\; |n_1, \ldots, n_i - 1, \ldots, n_A\rangle \\ \hat{a}_i^\dagger|n_1, \ldots, n_i, \ldots, n_A\rangle &= \sqrt{n_i + 1}\; |n_1, \ldots, n_i + 1, \ldots, n_A\rangle \end{aligned} \tag{2.12}$$

and can so construct an operator $\hat{n}_i \equiv \hat{a}_i^\dagger \hat{a}_i$, whose eigenvalue is the occupation number n_i:

$$\begin{aligned} \hat{a}_i^\dagger \hat{a}_i|n_1, \ldots, n_i, \ldots, n_A\rangle &= \hat{a}_i^\dagger \; \sqrt{n_i}\; |n_1, \ldots, n_i - 1, \ldots, n_A\rangle \\ &= \sqrt{n_i - 1 + 1}\; \sqrt{n_i}\; |n_1, \ldots, n_i - 1 + 1, \ldots, n_A\rangle \\ &= n_i \; |n_1, \ldots, n_i, \ldots, n_A\rangle \; . \end{aligned}$$

Any arbitrary operator can be represented in Second Quantization by using creation and annihilation operators. Thereby, single-particle operators in general take the form

$$\hat{A} = \sum_{i,j} a_{ij} \; \hat{a}_i^\dagger \hat{a}_j \tag{2.13}$$

where a_{ij} is the matrix element

$$a_{ij} = \langle \alpha_i | \hat{A} | \alpha_j \rangle \; . \tag{2.14}$$

Similarly, two-particle operators take the form

$$\hat{A} = \frac{1}{2} \sum_{i,j,k,l} a_{ijkl} \; \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_l \tag{2.15}$$

with matrix elements

$$a_{ijkl} = a_{ij} = \langle \alpha_i \alpha_j | \hat{A} | \alpha_k \alpha_l \rangle \ . \ . \tag{2.16}$$

Both types of operators will appear in the Hubbard Hamiltonian.

2.3 The Hubbard Model

2.3.1 The Hubbard Hamiltonian

For an interacting gas in an optical lattice the Hamiltonian reads [8]

$$\begin{aligned} H &= \int \mathrm{d}^3x \ \hat{\Psi}^\dagger(\mathbf{x}) \Big(-\frac{\hbar}{2m}\nabla^2 + V_L(\mathbf{x}) + V_T(\mathbf{x}) \Big) \hat{\Psi}(\mathbf{x}) \\ &\quad + \frac{g}{2} \int \mathrm{d}^3x \ \hat{\Psi}^\dagger(\mathbf{x}) \hat{\Psi}^\dagger(\mathbf{x}) \hat{\Psi}(\mathbf{x}) \hat{\Psi}(\mathbf{x}) \ , \end{aligned} \tag{2.17}$$

with the lattice potential $V_L(\mathbf{x})$, the trapping potential $V_T(\mathbf{x})$ and the bosonic field operators $\hat{\Psi}^\dagger(\mathbf{x}), \hat{\Psi}(\mathbf{x})$. For the atom-atom interaction a contact potential is assumed what lead to the second term in (2.17). Since the lattice potential is periodic the solutions of the Schrödinger equation can be expanded in Bloch functions. From the Bloch functions' reciprocal space one can move into the more convenient real space via Fourier transformation which leads to the Wannier functions

$$\omega_i(\mathbf{x} - \mathbf{R}_j). \tag{2.18}$$

i is the index for energy bands. In the following it is assumed that the atoms only populate the lowest energy band so the band index will be omitted.

The Wannier functions localized around $\mathbf{R}_i$ fulfill the orthonormalization relation

$$\int \mathrm{d}^3x \ \omega^*(\mathbf{x} - \mathbf{R}_i) \ \omega(\mathbf{x} - \mathbf{R}_j) = \delta_{ij} \ . \tag{2.19}$$

The field operators can be expanded in Wannier functions $\omega(\mathbf{x} - \mathbf{R}_j)$ with the creation and annihilation operators $\hat{a}_i^\dagger$ and $\hat{a}_i$

$$\hat{\Psi}^\dagger(\mathbf{x}) = \sum_i \hat{a}_i^\dagger \ \omega(\mathbf{x} - \mathbf{R}_i) \ . \tag{2.20}$$

(2.20) now is used to rewrite the Hamiltonian (2.17).

Atoms are able to tunnel between lattice sites i, j and the tunneling strength J_{ij} arises from the overlap of the wave functions at lattice sites i and j. One finds

$$J_{ij} = -\int \mathrm{d}^3x \ \omega^*(\mathbf{x} - \mathbf{R}_i) \ \Big(-\frac{\hbar}{2m}\nabla^2 + V_L(\mathbf{x}) + V_T(\mathbf{x}) \Big) \ \omega(\mathbf{x} - \mathbf{R}_j) \ , \tag{2.21}$$

The tunneling probability for hopping to a lattice site that is farther than the next neighbor is small and can be neglected. For an isotropic lattice the probability for tunneling to next neighbors is a constant and thus $J_{ij} \equiv J$.

The case $i = j$ not yet treated in formula (2.21) leads to the terms

$$\begin{aligned}\epsilon_{ii} &= \int \mathrm{d}^3x\ \omega^*(\mathbf{x}-\mathbf{R}_i)\ \Big(-\frac{\hbar}{2m}\nabla^2 + V_L(\mathbf{x}) + V_T(\mathbf{x})\Big)\ \omega(\mathbf{x}-\mathbf{R}_i) \\ &= \int \mathrm{d}^3x\ \omega^*(\mathbf{x}-\mathbf{R}_i)\ V_T(\mathbf{x})\ \omega(\mathbf{x}-\mathbf{R}_i) + \\ &\qquad \int \mathrm{d}^3x\ \omega^*(\mathbf{x}-\mathbf{R}_i)\ \Big(-\frac{\hbar}{2m}\nabla^2 + V_L(\mathbf{x})\Big)\ \omega(\mathbf{x}-\mathbf{R}_i)\ . \end{aligned} \tag{2.22}$$

The second term in (2.22) yields a constant energy offset ΔE at each lattice site

$$\epsilon_{ii} = \int \mathrm{d}^3x\ \omega^*(\mathbf{x}-\mathbf{R}_i)\ V_T(\mathbf{x})\ \omega(\mathbf{x}-\mathbf{R}_i) + \Delta E\ . \tag{2.23}$$

This energy offset can be set to zero, so that $\epsilon_{ii} \equiv \epsilon_i$ only depends on the trapping potential $V_T(\mathbf{R}_i)$ at lattice site i.

The atom-atom interaction is given in Wannier basis representation by

$$U_{ijkl} = \frac{g}{2}\int \mathrm{d}^3x\ \omega^*(\mathbf{x}-\mathbf{R}_i)\ \omega^*(\mathbf{x}-\mathbf{R}_j)\ \omega(\mathbf{x}-\mathbf{R}_k)\ \omega(\mathbf{x}-\mathbf{R}_l)\ . \tag{2.24}$$

Since only on-site interaction is considered the interaction parameter becomes constant $U_{ijkl} \equiv U\ \delta_{ij}\ \delta_{jk}\ \delta_{kl}$.

Using (2.21), (2.23), (2.24) the Hamiltonian takes the form

$$\hat{H} = -J\sum_{\langle i,j\rangle} \hat{a}_i^\dagger \hat{a}_j + \sum_i \epsilon_i \hat{a}_i^\dagger \hat{a}_i + \frac{U}{2}\sum_i \hat{a}_i^\dagger \hat{a}_i^\dagger \hat{a}_i \hat{a}_i\ . \tag{2.25}$$

The first sum runs over adjacent lattice sites $\langle i,j\rangle$ only. With the relations $\hat{n}_i = \hat{a}_i^\dagger \hat{a}_i$ and $\hat{n}_i(\hat{n}_i - 1) = \hat{a}_i^\dagger \hat{a}_i^\dagger \hat{a}_i \hat{a}_i$ the Hamiltonian can be descriptively reformulated into

$$\hat{H} = -J\sum_{\langle i,j\rangle} \hat{a}_i^\dagger \hat{a}_j + \sum_i \epsilon_i \hat{n}_i + \frac{U}{2}\sum_i \hat{n}_i(\hat{n}_i - 1)\ . \tag{2.26}$$

The interaction term contains the expression $\frac{1}{2}\hat{n}_i(\hat{n}_i - 1)$ because from an amount of n_i atoms $\frac{1}{2}n_i(n_i-1)$ pairs can be formed that interact with each other.

With increasing number of particles and lattice sites the Hilbert space dimension grows rapidly, which is the main motivation of the numerical renormalization group ansatz. The Hilbert space dimension of a system consisting of N particles within a lattice of I sites is in the case of bosons given by

$$\dim(\mathfrak{H}) = \frac{(N+I-1)!}{N!(I-1)!}\ . \tag{2.27}$$

Table 2.1 shows exemplarily some Hilbert space dimensions for systems with $I = N$.

$I = N$	$\dim(\mathfrak{H})$
3	10
6	462
12	1.352.078

Table 2.1: *Growth of the Hilbert space dimension for systems with* $I = N$.

2.3.2 Properties of the Ground States

To estimate the capability of the NRG within different domains of the parameter U/J it is useful to consider the exact ground-state of the Hubbard model in two limiting situations.

In the case of the ideal gas where the interaction between the particles vanishes ($U = 0$) and with no trapping potential ($\epsilon_i = 0$) the Hamiltonian reduces to the tunneling term

$$H(U = 0, \epsilon_i = 0) = -J \sum_{\langle i,j \rangle} \hat{a}_i^\dagger \hat{a}_j \ . \tag{2.28}$$

Here, the ground-state energy is minimized if the single-particle wave functions are spread over the entire lattice. The system then is in a superfluid state with maximum phase coherence and the ground state is given by

$$|\psi_{SF}\rangle \propto \left(\sum_i \hat{a}_i^\dagger \right)^N |0 \ \ldots \ 0\rangle \ . \tag{2.29}$$

Otherwise, in the case of dominating interaction $U \gg J$ the system tends towards a state with minimized fluctuation in occupation numbers. If the system has an integer filling, due to the dominating interaction energy tunneling is completely suppressed and the system finds itself in the Mott insulator state. In this state the single-particle wave functions are localized at single lattice sites

$$|\psi_{MI}\rangle \propto \prod_i \hat{a}_i^\dagger \ |0 \ \ldots \ 0\rangle \ , \tag{2.30}$$

so that phase coherence is lost but correlation of occupation numbers n_i gets maximized.

Between the superfluid $U \ll J$ and the Mott insulator regime $U \gg J$, a phase transition takes place at $U/J \approx 4.65$ [10, 11] which is among others characterized by the appearance of a gap in the excitation spectrum.

Chapter 3

The NRG Method

The one-dimensional Hubbard model can be solved exactly by diagonalizing the Hamilton matrix. Determining the Hamilton matrix and the diagonalization do not, in principle, pose a problem. However, the underlying Hilbert space and so the Hamilton matrix dimension grows exponentially with the system size so that in practice only smaller systems can be solved with reasonable effort.

The NRG method circumvents this problem by applying an iterative growing-procedure, where in each iteration step the Hamilton matrix is rotated into a truncated eigenbasis. In that way the dimension of the Hilbert space can be kept constant in spite of the increasing system size. The asset of this method is a linear increase of computational effort instead of the exponential increase when the problem is to be solved within the untruncated Hilbert space. The drawback comes from discarding of basis states that contain information about the exact state. This is associated with a numerical error in every iteration step that may lead step by step farther away from the exact solution. So one only obtains an approximate solution. This error can be minimized by discarding such eigenstates that are assumed not to contribute very much to the exact solution. For the calculation of a ground state, that is of course at the lower end of the energy scale, one will preferably discard the eigenstates at the upper end.

3.1 NRG applied to the Ising Model

To illustrate the main ideas of the NRG method the one-dimensional Ising model is more convenient than the Hubbard model.

3.1.1 The Ising Hamiltonian

The Hamilton operator of the Heisenberg model reads

$$\hat{H} = -\sum_{i,j} \chi_{ij}\ \hat{\mathbf{J}}_i \cdot \hat{\mathbf{J}}_j + \frac{1}{\hbar} g_J \mu_B \sum_i \hat{\mathbf{J}}_i \cdot \mathbf{B}_0\ . \tag{3.1}$$

It describes the interaction of magnetic dipoles localized at lattice sites. The χ_{ij} denote the coupling constants between separate lattice sites with the property

$$\begin{aligned} \chi_{ij} &= \chi_{ji} \\ \chi_{ii} &= 0 \ . \end{aligned} \tag{3.2}$$

The Ising model results as a special case of the Heisenberg model as a special case. The magnetic dipoles considered in the Ising model are spins $\hat{\mathbf{S}}_i$

$$\hat{\mathbf{J}}_i \cdot \hat{\mathbf{J}}_j \to \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j = \xi_x \hat{S}_i^x \hat{S}_j^x + \xi_y \hat{S}_i^y \hat{S}_j^y + \xi_z \hat{S}_i^z \hat{S}_j^z \ , \tag{3.3}$$

where the coefficients ξ_k are given by

$$\xi_x = \xi_y = 0 \ , \quad \xi_z = 1 \tag{3.4}$$

and $\mathbf{B}$ is chosen to point into z-direction

$$\mathbf{B} = B_0 \mathbf{e}_z \ . \tag{3.5}$$

So one has

$$\hat{H} = -\sum_{i,j} \chi_{ij} \ \hat{S}_i^z \hat{S}_j^z - \Gamma \sum_i \hat{S}_i^z \tag{3.6}$$

with

$$\Gamma = -\frac{1}{\hbar} g_J \mu_B B_0 \ . \tag{3.7}$$

As an additional simplification, only the interaction between next neighbors is considered with a constant coupling strength J

$$H = -J \sum_{\langle i,j \rangle} \hat{S}_i^z \hat{S}_j^z - \Gamma \sum_i \hat{S}_i^z \ . \tag{3.8}$$

3.1.2 NRG applied to the Ising Model

Consider a system of N spins on N lattice sites:

Figure 3.1: *Spin system with N lattice sites.*

For open boundary conditions, the Hamiltonian is given by

$$\hat{H} = -J \sum_{\langle i,j \rangle} \hat{S}_i^z \hat{S}_j^z - \Gamma \sum_i \hat{S}_i^z \tag{3.9}$$

and is therefore a sum over characteristic terms like

$$\hat{S}_i^z \hat{S}_j^z \quad \text{und} \quad \hat{S}_i^z \ . \tag{3.10}$$

The first sum in (3.9) runs over adjacent spins so that only terms with index pairs $(i, i+1)$, $(i+1, i)$ remain. A more convenient notation is

$$\hat{H} = -J \sum_{i=1}^{N-1} \left\{ \hat{S}_i^z \hat{S}_{i+1}^z + \hat{S}_{i+1}^z \hat{S}_i^z \right\} - \Gamma \sum_{i=1}^{N} \hat{S}_i^z \ . \tag{3.11}$$

From the total system one picks out a subsystem, in the following called block, with N_B lattice sites and the Hamiltonian

$$\hat{H}_B = -J \sum_{i=1}^{N_B-1} \left\{ \hat{S}_i^z \hat{S}_{i+1}^z + \hat{S}_{i+1}^z \hat{S}_i^z \right\} - \Gamma \sum_{i=1}^{N_B} \hat{S}_i^z \ . \tag{3.12}$$

The block's size will affect the accuracy of the numerical results. Therefore, blocks consisting of a single site are possible, but in general blocks will include at least 2 lattice sites to get suitable results. Setting $N_B = 2$ the Hamiltonian for the isolated block

Figure 3.2: *Separation of the block system.*

is given by

$$\hat{H}_B = -J \left\{ \hat{S}_1^z \hat{S}_2^z + \hat{S}_2^z \hat{S}_1^z \right\} - \Gamma \left\{ \hat{S}_1^z + \hat{S}_2^z \right\} \ . \tag{3.13}$$

The size of the system is increased by appending an additional subsystem adjacent to the block called the site. As with the block system, the site system can in principle take any desired form, but in the case of NRG it will consist of a single lattice site only.

The system formed from block and site

Figure 3.3: *The superblock system.*

will in the following be called superblock system. The superblock Hamiltonian is given by

$$\begin{aligned}
\hat{H}_{\text{Super}} &= -J \left\{ \hat{S}_1^z \hat{S}_2^z + \hat{S}_2^z \hat{S}_1^z + \hat{S}_2^z \hat{S}_3^z + \hat{S}_3^z \hat{S}_2^z \right\} - \Gamma \left\{ \hat{S}_1^z + \hat{S}_2^z + \hat{S}_3^z \right\} \\
&= \underbrace{-J \left\{ \hat{S}_1^z \hat{S}_2^z + \hat{S}_2^z \hat{S}_1^z \right\} - \Gamma \left\{ \hat{S}_1^z + \hat{S}_2^z \right\}}_{\hat{H}_B} \underbrace{-J \left\{ \hat{S}_2^z \hat{S}_3^z + \hat{S}_3^z \hat{S}_2^z \right\}}_{\hat{H}_{BS}} \underbrace{-\Gamma \left\{ \hat{S}_3^z \right\}}_{\hat{H}_S} \\
&= \hat{H}_B + \hat{H}_{BS} + \hat{H}_S \ .
\end{aligned} \tag{3.14}$$

The Hamiltonians $\hat{H}_B, \hat{H}_{BS}$ and $\hat{H}_S$ act on different parts of the Hilbert space. While H_B only acts within the block's Hilbert space $\mathfrak{H}_B$ and H_B only acts within the site's Hilbert space $\mathfrak{H}_S$, the Hamiltonian H_{BS} operates within the Hilbert space $\mathfrak{H}_{BS}$ of the superblock system.

During the NRG procedure the block system is increased iteratively (Fig. 3.1.2). As in (3.14) each iteration step the superblock Hamiltonian can be divided into Hamiltonians acting on $\mathfrak{H}_B, \mathfrak{H}_S$ and $\mathfrak{H}_{BS}$.

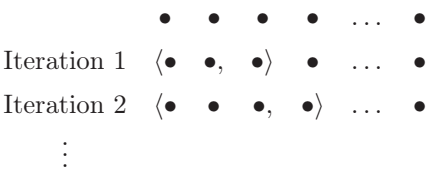

Figure 3.4: *The growth of the block and superblock system over the course of the NRG procedure.*

Let $|+\rangle \equiv |+\rangle_x, |-\rangle = |-\rangle_x$ denote eigenstates to the operator $\hat{S}^x$ that obey the equations

$$\begin{aligned} \hat{S}^x|+\rangle &= |+\rangle \\ \hat{S}^x|-\rangle &= -|-\rangle \, . \end{aligned} \tag{3.15}$$

This awkward choice of the basis produces off-diagonal matrix elements in the matrix representations of the operators, as they will appear in general (and in particular in the case of the Hubbard Hamiltonian). By the help of the relations

$$\begin{aligned} |\pm\rangle &= \frac{1}{\sqrt{2}}\Big\{|+\rangle_z \pm |-\rangle_z\Big\} \\ \hat{S}^z|\pm\rangle_z &= \pm|\pm\rangle_z \end{aligned} \tag{3.16}$$

one finds for the action of $\hat{S}^z$ on the eigenstates of $\hat{S}^x$

$$\begin{aligned} \hat{S}^z|+\rangle &= |-\rangle \\ \hat{S}^z|-\rangle &= |+\rangle \, . \end{aligned} \tag{3.17}$$

In the first iteration step the block consists of 2 lattice sites and so $\mathfrak{H}_B$ is represented in a two-spin basis

$$\Big\{|\epsilon_1 \, \epsilon_2\rangle\Big\}_B = \Big\{|++\rangle, |+-\rangle, |-+\rangle, |--\rangle\Big\} \, . \tag{3.18}$$

In this basis the matrix representation of $\hat{S}^z_{1,2}$ becomes

$$S^z_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad , \qquad S^z_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \, . \tag{3.19}$$

The site basis is given by

$$\Big\{|\epsilon\rangle\Big\}_S = \Big\{|+\rangle, |-\rangle\Big\} \tag{3.20}$$

and the matrix representation of the site operator

$$S_3^z = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \tag{3.21}$$

With this, one obtains the Hamiltonians H_B and H_S represented in their respective bases

$$H_B = \begin{pmatrix} 0 & -\Gamma & -\Gamma & -2J \\ -\Gamma & 0 & -2J & -\Gamma \\ -\Gamma & -2J & 0 & -\Gamma \\ -2J & -\Gamma & -\Gamma & 0 \end{pmatrix} , \qquad H_S = \begin{pmatrix} 0 & -\Gamma \\ -\Gamma & 0 \end{pmatrix} . \tag{3.22}$$

The Hamiltonian $\hat{H}_{BS}$ connects the block and site basis leading to the superblock basis

$$\begin{aligned} \Big\{|\epsilon_1\ \epsilon_2\ \epsilon_3\rangle\Big\}_{\text{Super}} &= \Big\{|\epsilon_1\ \epsilon_2\rangle\Big\}_B \otimes \Big\{|\epsilon_3\rangle\Big\}_S \\ &= \Big\{|++\rangle, |+-\rangle, |-+\rangle, |--\rangle, \\ &\qquad |++\rangle, |+-\rangle, |-+\rangle, |--\rangle\Big\} . \end{aligned} \tag{3.23}$$

The individual Hamiltonians $\hat{H}_B$ and $\hat{H}_S$ and the spin operators $\hat{S}_i^z$ do not connect states of different Hilbert spaces $\mathfrak{H}_B, \mathfrak{H}_S$ or $\mathfrak{H}_{BS}$. Setting

$$\begin{aligned} \Big\{|k\rangle\Big\}_B &:= \Big\{|\epsilon_1\ \epsilon_2\rangle\Big\}_B \\ \Big\{|l\rangle\Big\}_S &:= \Big\{|\epsilon_3\rangle\Big\}_S \end{aligned} \tag{3.24}$$

one finds for the matrix elements

$$(O)_{kk',ll'} = \langle k\ l|\hat{O}|k'\ l'\rangle \tag{3.25}$$

of the Hamiltonians $\hat{H}_B$ and $\hat{H}_S$ and the operators required for constructing $\hat{H}_{BS}$ represented in the superblock basis

$$\begin{aligned} (H_B)_{kk',ll'} &= (H_B)_{kk'}\,\delta_{ll'} \\ (H_S)_{kk',ll'} &= \delta_{kk'}\,(H_S)_{ll'} \\ (S_2^z)_{kk',ll'} &= (S_2^z)_{kk'}\,\delta_{ll'} \\ (S_3^z)_{kk',ll'} &= \delta_{kk'}\,(S_3^z)_{ll'} \ . \end{aligned} \tag{3.26}$$

One obtains

$$H_B = \begin{pmatrix} 0 & -\Gamma & -\Gamma & -2J & 0 & 0 & 0 & 0 \\ -\Gamma & 0 & -2J & -\Gamma & 0 & 0 & 0 & 0 \\ -\Gamma & -2J & 0 & -\Gamma & 0 & 0 & 0 & 0 \\ -2J & -\Gamma & -\Gamma & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\Gamma & -\Gamma & -2J \\ 0 & 0 & 0 & 0 & -\Gamma & 0 & -2J & -\Gamma \\ 0 & 0 & 0 & 0 & -\Gamma & -2J & 0 & -\Gamma \\ 0 & 0 & 0 & 0 & -2J & -\Gamma & -\Gamma & 0 \end{pmatrix} \quad \text{etc.} \tag{3.27}$$

To construct $\hat{H}_{BS}$ one uses operators represented in the superblock basis. Using (3.14) the superblock Hamiltonian becomes

$$H_{\text{Super}} = (-1) \cdot \begin{pmatrix} 0 & \Gamma & \Gamma & 2J & \Gamma & 2J & 0 & 0 \\ \Gamma & 0 & 2J & \Gamma & 2J & \Gamma & 0 & 0 \\ \Gamma & 2J & 0 & \Gamma & 0 & 0 & \Gamma & 2J \\ 2J & \Gamma & \Gamma & 0 & 0 & 0 & 2J & \Gamma \\ \Gamma & 2J & 0 & 0 & 0 & \Gamma & \Gamma & 2J \\ 2J & \Gamma & 0 & 0 & \Gamma & 0 & 2J & \Gamma \\ 0 & 0 & \Gamma & 2J & \Gamma & 2J & 0 & \Gamma \\ 0 & 0 & 2J & \Gamma & 2J & \Gamma & \Gamma & 0 \end{pmatrix} . \tag{3.28}$$

The eigenvalue problem of the superblock Hamiltonian can be solved. One obtains 8 eigenvectors that can be sorted by their energy eigenvalues.

$$|E_1\rangle, \ldots, |E_8\rangle \ .$$

Consider the transformation matrix

$$T_{\text{full}} = \begin{pmatrix} \uparrow & & \uparrow \\ \langle\alpha|E_1\rangle & \ldots & \langle\alpha|E_8\rangle \\ \downarrow & & \downarrow \end{pmatrix} , \tag{3.29}$$

where

$$\begin{pmatrix} \uparrow \\ \langle\alpha|E_j\rangle \\ \downarrow \end{pmatrix} \tag{3.30}$$

stands schematically for the projection of $|E_j\rangle$ on the basis $\{\alpha_i\}$.

Using the transformation matrix T_{full} the superblock Hamiltonian H_{Super} can be rotated into the energy eigenbasis

$$H_{\text{Super, diag}} = T_{\text{voll}}^T \ H_{\text{Super}} \ T_{\text{voll}} \ .$$

The N-site problem can be solved exactly by diagonalizing the superblock Hamiltonian in each iteration step and using it as the block Hamiltonian in the subsequent iteration step. To distinguish the Hamiltonians of different iteration steps an iteration index is introduced. Furthermore, the dimension dim $(H_x^{(i)})$ of a Hamilton matrix of the ith iteration step

$$H_x^{(i)} \ , \qquad x : \text{B, S, Super} \ , \tag{3.31}$$

will in the following be denoted by $D_x^{(i)}$.

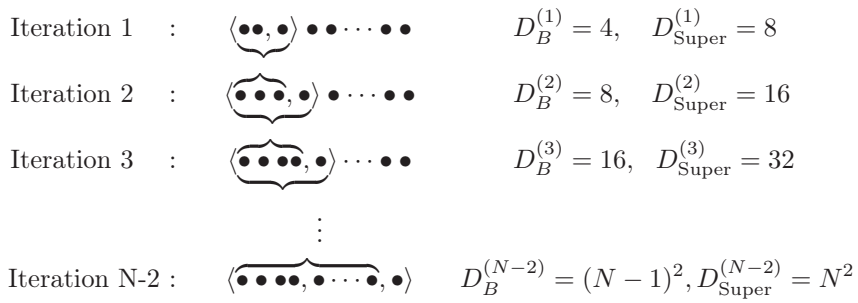

Figure 3.5: *Transformation and growth of the Hilbert space dimensions.*

Rotating the superblock Hamiltonian into the full energy eigenbasis in each iteration step and using this Hamiltonian as the new block Hamiltonian is equivalent to the full solution. In each iteration step the dimension of the block system is multiplied by degrees of freedom of the site system, which means a doubling in this example. Just as in the full solution, in the last iteration step a superblock Hamiltonian of dimension N^2 would have to be solved so that this method would not bear any advantages.

The goal of the NRG method is to keep the block and superblock Hilbert space dimensions constant. Since the size of the block and superblock system is increasing in each iteration step this can only be achieved by a truncation of the underlying bases and therefore their associated Hilbert spaces. In each iteration step the superblock Hamiltonian of dimension D_{Super} has to be transformed into a Hilbert space of dimension D_B.

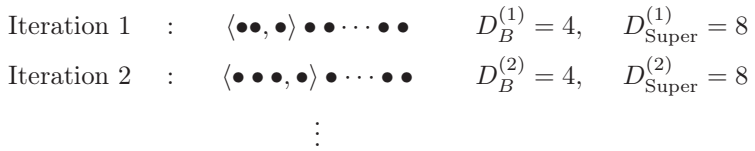

Figure 3.6: *Constant dimensions by Hilbert space truncation.*

This can be achieved by $D_{\text{Super}} \times D_B$ transformation matrices. In general the

main interest is the ground state and the low-lying excited states of a system. These states will mainly consist of basis states that for their part represent low energies. By truncation of the energy eigenbasis one will minimize the error induced by discarding the high-energy basis states. Therefore, the columns of the transformation matrix are built up from the lowest-energy eigenvectors of the superblock Hamiltonian

$$T^{(1)} = \begin{pmatrix} \uparrow & & \uparrow \\ \langle\alpha|E_1\rangle & \dots & \langle\alpha|E_{D_B}\rangle \\ \downarrow & & \downarrow \end{pmatrix} . \tag{3.32}$$

So the block Hamiltonian dimension determines the number of basis states that are maintained in the truncated energy eigenbasis. With increasing dimension of the block Hamiltonian the truncated Hilbert space and the energy values get closer to the exact case.

The superblock Hamiltonian of the second iteration step

$$\text{Iteration 2} \quad : \quad \langle \bullet\bullet\bullet, \bullet\rangle \bullet \cdots \bullet\bullet \tag{3.33}$$

Figure 3.7: *The superblock Hamiltonian in the second iteration step.*

is given by

$$H^{(2)}_{\text{Super}} = H^{(2)}_B + H^{(2)}_{BS} + H^{(2)}_S \ . \tag{3.34}$$

The representation of $H^{(2)}_B = \left(T^{(1)}\right)^T \ H^{(1)}_{\text{Super}} \ T^{(1)}$ in the new block basis ($\equiv$ truncated superblock basis) is already known. Likewise, $H^{(2)}_S \equiv H^{(1)}_S$ can be assumed to be known in the new site basis because neither the site basis nor the site Hamiltonian change in any way during the NRG process.

$H^{(2)}_{BS}$ is built up from the spin operators at the border between the new block and site. On the side of the block this is the spin operator $\hat{S}^z_3$ that has been the operator on the side of the side in the previous iteration step

$$H^{(2)}_{BS} = -J \left\{ S^z_3 S^z_4 + S^z_4 S^z_3 \right\} \ . \tag{3.35}$$

So the old site operator has to be transformed from the old superblock basis into the new block basis, too

$$\left(T^{(1)}\right)^T \ S^3_z \ \ T^{(1)} \ . \tag{3.36}$$

The relations (3.26) still hold for the new block basis. So in the second iteration step one can proceed as in the first one: $H^{(2)}_B$, S^3_z and $H^{(2)}_S$ will be transfered from block and site basis into the new superblock basis and from them $H^{(2)}_{BS}$ and $H^{(2)}_{\text{Super}}$ will be constructed. For $H^{(2)}_{\text{Super}}$ the eigenvalue problem will have to be solved and the transformation matrix $T^{(2)}$ of the second iteration step will have to be built. Again, the superblock Hamiltonian and the site spin operator would have to be transformed and so on.

3.2 NRG Scheme

Summarizing the discussions of Sec. 3.1.2 one can formulate the following NRG algorithm.

1. Consider a subsystem (block system) and the adjacentlattice site (site system)
2. Form the combined block-site system (superblock system) with

$$H^{(1)}_{\text{Super}} = H^{(1)}_B + H^{(1)}_{BS} + H^{(1)}_S$$

3. Calculate $H^{(1)}_B, H^{(1)}_S$ and the operator matrix representations $O^{(1)}_k$ needed to construct $H^{(1)}_{BS}$ in the block and site basis
4. Transfer $H^{(1)}_B, H^{(1)}_S, \hat{O}^{(1)}_k$ into the superblock basis
5. Construct the Hamiltonian $H^{(1)}_{BS}$ from $O^{(1)}_k$ and the superblock Hamiltonian $H^{(1)}_{\text{Super}}$ from $H^{(1)}_B, H^{(1)}_{BS}, H^{(1)}_S$
6. Solve the eigenvalue problem of the superblock Hamiltonian and build the transformation matrix $T^{(1)}$ from the D_B energetically lowest eigenvectors.
7. Transform the superblock Hamiltonian and use it as the new block Hamiltonian in the next iteration step

$$H^{(2)}_B = \left(T^{(1)}\right)^T \; H^{(1)}_{\text{Super}} \; T^{(1)}$$

8. Transform the operator matrices needed to build the new $H^{(2)}_{BS}$ Hamiltonian, $O^{(1)}_j \rightarrow O^{(2)}_j$
9. Continue at step 4 with the next iteration.

Chapter 4

NRG applied to the Hubbard Model

The Hubbard Hamiltonian (2.26) describes a system of $N \equiv \sum_{i=1}^{I} n_i$ atoms in a one-dimensional lattice of length I. Similar to the Ising Hamiltonian it is more convenient to reformulate the sum running over direct neighbors $\langle i, j \rangle$ into a sum running over lattice sites

$$\hat{H} = -J \sum_{i=1}^{I-1} \left\{ \hat{a}_i^\dagger \hat{a}_{i+1} + \hat{a}_{i+1}^\dagger \hat{a}_i \right\} + \sum_{i=1}^{I} \epsilon_i \hat{n}_i + \frac{U}{2} \sum_{i=1}^{I} \hat{n}_i (\hat{n}_i - 1) \ . \tag{4.1}$$

The atom-atom interaction and the trapping potential contribute at isolated lattice sites i

$$\hat{H}_{U_i} = \epsilon_i \hat{n}_i + \frac{U}{2} + \hat{n}_i (\hat{n}_i - 1) \tag{4.2}$$

while the tunneling term

$$\hat{H}_{T_{i,i+1}} = \hat{a}_i^\dagger \hat{a}_{i+1} + \hat{a}_{i+1}^\dagger \hat{a}_i \tag{4.3}$$

connects two neighboring lattice sites. The total Hamiltonian is the sum over the contributions (4.2), (4.3) :

$$\hat{H} = \sum_{i=1}^{I-1} \hat{H}_{T_{i,i+1}} + \sum_{i=1}^{I} \hat{H}_{U_i} \ , \tag{4.4}$$

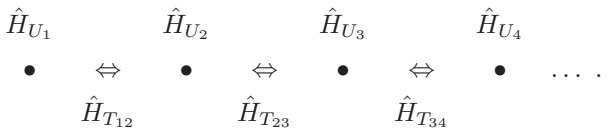

Figure 4.1: *The Hamiltonians associated to lattice sites.*

Again (as in Sec. 3.1.2), without a Hilbert space truncation the full problem could be solved by iterativ increase of the initial system. The aim is the calculation of low-lying energy eigenstates so in each iteration step the system has to be projected into the basis of the lowest-energy eigenvectors.

The system properties are dominated by the interplay of the kinetic term

$$-J \sum_{i=1}^{I-1} \left\{ \hat{a}_i^\dagger \hat{a}_{i+1} + \hat{a}_{i+1}^\dagger \hat{a}_i \right\} \ , \tag{4.5}$$

which strives to spread the single-particle wave functions over the whole lattice, and the interaction term

$$\frac{U}{2} \sum_i \hat{n}_i (\hat{n}_i - 1) \ , \tag{4.6}$$

which tries to localize the wave function at single lattice sites. This leads for $U = 0$ or $J = 0$ to the ground states described in Sec. 2.3.2. Therefore, a simple estimate of the accuracy of the NRG method in the different domains of U/J can be made.

In the superfluid regime the ground state is given by a superposition of all possible basis states. Every eigenstate that will be discarded during the NRG procedure will be missing in this superposition and will distort the result. However, basis states to increasing energies lose their weight to the solution so that NRG sure will not provide its best results in the superfluid regime but should still work in principle.

One will expect more accurate results in the Mott insulator regime that consists in the limit of strong interactions $U/J \to \infty$ only of the state with minimum fluctuation in occupation numbers.

4.1 Particle Numbers

In Sec. 3.1.2 no special consideration of particle numbers was necessary. The Ising model describes a one-dimensional lattice, with one localized spin at each lattice site. Therefore, the number of spins in the block and site system is strictly determined by the number of lattice sites these systems consist of. This limitation is missing in the case of the Hubbard model because the states can show an arbitrary distribution of the atoms over the lattice. So the NRG method applied on the Hubbard Hamiltonian gets slightly more complicated than its Ising counterpart, although the underlying scheme of Sec. 3.2 remains unchanged. Besides the selection of eigenstates with respect to their energy eigenvalues an additional selection condition is given by the particle numbers of the eigenstates which implies an additional error source.

4.2 Block, Site and Superblock Basis

Consider a system with N particles and I lattice sites.

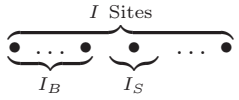

Figure 4.2: *Choice of the block and site system.*

The NRG method begins by choosing the block system with I_B lattice sites. As mentioned in Sec. 3.1.2 the size of this block system will affect the NRG's accuracy and it should consist of at least 2 lattice sites $I_B \geq 2$. In the NRG method usually site systems consisting of a single lattice site adjacent to the block are employed, $I_S = 1$.

Different from the Ising model the number of lattice sites does not fix the number of particles that are in the (sub-) system. It is only known which particle number N the entire system is supposed to have at the end, and possible basis states correspond to arbitrary arrangements of the atoms within the lattice. For example, one possible basis state is given by the state where all particles resides on the last lattice site.

Therefore the basis will have to include a certain range of particle numbers that will in the ideal case cover all possible particle numbers from 0 to N. With increasing particle number the Hilbert space dimension will increase as well, and so it can be reasonable to abandon some particle numbers (which means all basis states to these certain particle numbers) when the block and site basis are constructed. In the case of dominating interaction, the states that exhibits a (almost) constant occupation of each lattice site will be dominant in the ground state. In the superfluid regime the probability distribution of the occupation numbers at a single lattice site is a Poissonian distribution so that states with an (almost) homogeneous occupation will have the highest weight, as well. Therefore the bases for block and site should be constructed around this homogeneous occupation with a range of particle numbers $N_{\mathrm{B,min}}$ ... $N_{\mathrm{B,max}}$ (Block) and $N_{\mathrm{S,min}}$ ... $N_{\mathrm{S,max}}$ (Site), respectively.

$$\text{Blockbasis}: \left\{|k\rangle\right\} := \left\{|n_1, \ldots, n_{I_B}\rangle \ : \ N_{\mathrm{B,min}} \leq \sum_{i=1}^{I_B} n_i \leq N_{\mathrm{B,max}}\right\}$$

$$\text{Sitebasis}: \left\{|l\rangle\right\} := \left\{|n_{I_B+1}, \ldots, n_{I_B+I_S}\rangle \ : \ N_{\mathrm{S,min}} \leq \sum_{i=I_B+1}^{I_B+I_S} n_i \leq N_{\mathrm{S,max}}\right\} .$$

The superblock basis arises from the product of the block and site basis. So the superblock basis includes the particle numbers

$$N_{\mathrm{B,min}} + N_{\mathrm{S,min}} \ \ldots \ N_{\mathrm{B,max}} + N_{\mathrm{S,max}} \ . \tag{4.7}$$

All bases should be constructed in a way that they are sorted with respect to total particle numbers so that their associated Hamiltonians become block diagonal. This is due to the number conserving character of the Hubbard model.

4.3 Block, Site and Superblock Hamiltonian

Similar to the way it was done in Sec. 3.1.2 the superblock Hamiltonian

$$\begin{aligned}\hat{H}^{(1)}_{\text{Super}} &= -J\Big\{\hat{a}_1^\dagger\hat{a}_2+\hat{a}_2^\dagger\hat{a}_1+\hat{a}_2^\dagger\hat{a}_3+\hat{a}_3^\dagger\hat{a}_2\Big\}\\ &\quad+\epsilon_1\hat{n}_1+\epsilon_2\hat{n}_2+\epsilon_3\hat{n}_3\\ &\quad+\frac{U}{2}\Big\{\hat{n}_1(\hat{n}_1-1)+\hat{n}_2(\hat{n}_2-1)+\hat{n}_3(\hat{n}_3-1)\Big\}\end{aligned} \tag{4.8}$$

can be separated and cast in the form

$$\hat{H}^{(1)}_{\text{Super}}=\hat{H}^{(1)}_B+\hat{H}^{(1)}_{BS}+\hat{H}^{(1)}_S \tag{4.9}$$

with

$$\begin{aligned}\hat{H}^{(1)}_B &= -J\Big\{\hat{a}_1^\dagger\hat{a}_2+\hat{a}_2^\dagger\hat{a}_1\Big\}+\epsilon_1\hat{n}_1+\epsilon_2\hat{n}_2+\frac{U}{2}\Big\{\hat{n}_1(\hat{n}_1-1)+\hat{n}_2(\hat{n}_2-1)\Big\}\\ \hat{H}^{(1)}_{BS} &= -J\Big\{\hat{a}_2^\dagger\hat{a}_3+\hat{a}_3^\dagger\hat{a}_2\Big\}\\ \hat{H}^{(1)}_S &= +\frac{U}{2}\Big\{\hat{n}_3(\hat{n}_3-1)\Big\}\,.\end{aligned} \tag{4.10}$$

The upper index denotes the iteration step. Using the creation and annihilation operators represented in the block and site basis from (4.10) one immediately obtains the block and site Hamiltonian, $\hat{H}^{(1)}_B$ and $\hat{H}^{(1)}_S$, represented in their respective basis. Single creation $\hat{a}_i^\dagger$ or annihilation operators $\hat{a}_j$ do not conserve the particle number, however, the operator products $\hat{a}_i^\dagger\hat{a}_j$ of creation and annihilation operators appearing in the Hamiltonians do. Therefore the Hamiltonians only connect number states with same particle numbers which means that the Hamiltonians are block diagonal with respect to particle numbers and take block form because the associated bases have been constructed sorted by particle numbers.

In order to to be able to construct the operator products $\hat{a}_2^\dagger\hat{a}_3$ and $\hat{a}_3^\dagger\hat{a}_2$ appearing in $\hat{H}^{(1)}_{BS}$, every operator has to be transfered into the superblock basis separately. Therefore, a transformation of $\hat{H}_B(1)$ and $\hat{H}^{(1)}_S$ is necessary as well in order to be able to construct the superblock Hamiltonian via (4.9).

In line with the Ising model single operators do not connect states from different Hilbert spaces so that in superblock basis representation the matrix elements

$$(O)_{kk',ll'}=\langle k\ l|\hat{O}|k'\ l'\rangle$$

of $H^{(1)}_B$ and $H^{(1)}_S$ and the operators needed to compose $H^{(1)}_{BS}$ are given by

$$\begin{aligned}\Big(H^{(1)}_B\Big)_{kk',ll'} &= \Big(H^{(1)}_B\Big)_{kk'}\delta_{ll'}\\ \Big(H^{(1)}_S\Big)_{kk',ll'} &= \delta_{kk'}\Big(H^{(1)}_S\Big)_{ll'}\end{aligned}$$

$$\begin{aligned}\Big(\hat{a}_2^\dagger\Big)_{kk',ll'} &= \Big(\hat{a}_2^\dagger\Big)_{kk'}\delta_{ll'}\\ \Big(\hat{a}_2\Big)_{kk',ll'} &= \Big(\hat{a}_2\Big)_{kk'}\delta_{ll'}\end{aligned}$$

$$\begin{aligned}\left(\hat{a}_3^\dagger\right)_{kk',ll'} &= \delta_{kk'}\left(\hat{a}_3^\dagger\right)_{ll'} \\ \left(\hat{a}_3\right)_{kk',ll'} &= \delta_{kk'}\left(\hat{a}_3\right)_{ll'} \ . \end{aligned} \tag{4.11}$$

Hence using (4.9) one obtains the superblock Hamiltonian.

4.4 Transformation

After having determined the superblock Hamiltonian, it has to be transformed into a truncated eigenbasis. This transformation causes a problem with the assignment of basis vectors to particle numbers because after the first transformation the basis vectors do not correspond to a simple occupation number representation anymore. Nonetheless an assignment to particle numbers is imperative since at the end of the NRG procedure one wants to have the result for a particular filling fraction. Here the block structure of the block and superblock Hamiltonian turn out to be useful.

4.4.1 Diagonalization of Block Matrices

Consider a square matrix M of dimension m that has block form with sub-blocks k_j. M is represented in the basis

$$\left\{|\alpha_i\rangle\right\} \ . \tag{4.12}$$

Each sub-block k_j is spanned by a set of basis vectors

$$|\alpha_i^{(k_j)}\rangle \ \epsilon \ \left\{|\alpha_i\rangle\right\} \tag{4.13}$$

that do not contribute to matrix elements outside of k_j. Eigenvectors of M will only be superpositions of basis vectors to one sub-block at a time.

Therefore, the diagonalization matrix T_{full} built up from eigenvectors of M (notation as in (3.29))

$$T_{\text{full}} = \begin{pmatrix} \uparrow & & \uparrow \\ \langle\alpha|E_1\rangle & \dots & \alpha|E_m\rangle \\ \downarrow & & \downarrow \end{pmatrix} \tag{4.14}$$

can be brought into a block form with sub-blocks t_j that is identical to the block structure of M. After diagonalization

$$M' = \left(T_{\text{full}}\right)^T \ M \ T_{\text{full}} \tag{4.15}$$

M' trivially has block form with diagonal sub-blocks k'_j which also can be obtained by

$$k'_j = \left(t_j\right)^T \ . \ k_j \ . \ t_j \ . \tag{4.16}$$

So the problem of diagonalizing a square matrix in block form M can be dissected into their sub-block's diagonalization.

4.4.2 Transformation of the Superblock Hamiltonian

The aim is to rotate the superblock Hamiltonian into an eigenbasis in which it has the dimension and block structure of the block Hamiltonian. Since the superblock basis is the product basis of block and site basis, the superblock Hamiltonian will contain more sub-blocks with different particle numbers than the block Hamiltonian does. Using a transformation matrix built up from the D_B lowest eigenvectors, the dimension can be reduced as desired but the transformed matrix will still have the number of sub-blocks the original superblock Hamiltonian. So the block form of the block Hamiltonian can only be reproduced by restriction to certain sub-spaces that will be transformed following Sec. 4.4.1.

According to Sec. 4.4.1, each sub-block of the superblock Hamiltonian is transformed into a certain sub-block of the block Hamiltonian. In general, a sub-block in the superblock Hamiltonian has greater dimension than its target block in the block Hamiltonian. So it has to be transformed by a rectangular matrix built up from the low-energy eigenstates, as described in Sec. 3.1.2. So, there are two different mechanisms that reduces the dimension of the transformed superblock Hamiltonian: The restriction to certain sub-blocks and the reduction of the dimensions of the transformed blocks.

The transition from the block Hamiltonian $H_B^{(k)}$ to the new block Hamiltonian $H_B^{(k+1)}$ corresponds to an increase of the system size by the site system. Therefore, the transformation will be chosen such that the particle numbers associated to sub-blocks in the block Hamiltonian grow by the average occupation number of the site system.

4.4.3 Transformation Matrix

Since only selected sub-blocks will be transformed it is convenient to introduce a superblock basis that has been reduced to the particle numbers one needs in the following, called the reduced superblock basis, and to represent the superblock Hamiltonian in this reduced basis. In order to directly use the transformation matrix constructed below, this reduced superblock basis should be sorted by particle numbers, so that all operators expressed in this basis assume block form. The transformation matrix that transforms this reduced superblock Hamitonian then has D_B columns and its number of rows matches the dimension of the reduced superblock basis. Furthermore, as discussed in Sec. 4.4.1 the transformation matrix has block form, whereby each block is responsible for the transformation of a certain sub-block of the superblock Hamiltonian. The structure of a sub-block in the transformation matrix is determined by the initial and the target sub-block (Fig. 4.3). The number of columns of such a transformation matrix's sub-block matches the dimension of the target sub-block in the block Hamiltonian and the number of rows matches the dimension of the sub-block in the superblock Hamiltonian that has to be transformed. In

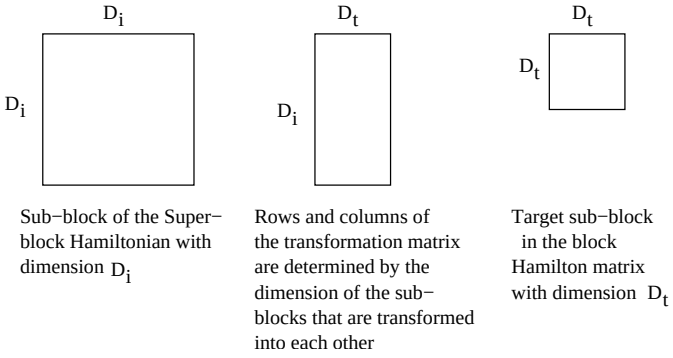

Figure 4.3: *Rows and columns of a sub-block of the transformation matrix.*

the transformation matrix a sub-block to a given particle number consists of the low-energy eigenvectors to this given particle number

$$T = \begin{pmatrix} \uparrow & & & \\ \langle \alpha^{N=n} | E_1^{(N=n)} \rangle & & & \\ \downarrow & & & \\ & \uparrow & \uparrow & \\ & \langle \alpha^{N=n+1} | E_1^{(N=n+1)} \rangle & \langle \alpha^{N=n+1} | E_2^{(N=n+1)} \rangle & \\ & \downarrow & \downarrow & \\ & & & \ddots \end{pmatrix} . \tag{4.17}$$

4.4.4 Transformation of the Operators

After transforming the superblock Hamiltonian block Hamiltonian for the next iteration step is known. In addition to the superblock Hamiltonian the operators that act in the Hilbert space of the new block that are needed to construct $\hat{H}_{BS}^{(2)}$ have to be transformed too. $\hat{H}_{BS}^{(k)}$ is composed of the operators $\hat{a}_B^{(\dagger)}$ and $\hat{a}_S^{(\dagger)}$ on the border between block and site. After the first transformation the block basis contains no (at least no directly accessible) information about occupation numbers. Therefore, a consideration of operators that act on certain lattice sites is no longer possible. However one can interprete $\hat{a}_B^{(\dagger)}$ as boundary operators of the block and transform them using the transformation matrix constructed in Sec. 4.4.3. Since this transformation matrix is designed to transform matrices represented in the reduced superblock basis, the boundary operators have to be expressed in the reduced superblock basis before the transformation.

4.5 Starting the Next Iteration

After the transformation of the superblock Hamiltonian and the block boundary operators into the new block basis the next iteration can be started. The new site basis is identical to the old one and the representations of the site operators do not change. The relations (4.11) still hold for the operators in the next iteration step so they can be expressed in the new superbasis although there is no simple occupation number representation anymore. The iteration is done until the desired number of lattice sites is reached. The diagonal matrix elements of the block Hamiltonian then are the energies and one can take out the ones to the desired particle number.

4.6 Transformation for Filling Factor 1

At the beginning the size of the block and the site subsystem was determined by choosing a certain number of lattice sites and a range of particle numbers their associated bases should contain. The concrete choice of sub-blocks of the superblock Hamiltonian depends on the problem that is to be solved and the parameters that determines the block and site bases. In the following only systems with equal numbers of lattice sites and particles are considered (filling factor 1).

4.6.1 Consideration of All Particle Numbers

A system of $N = 5$ particles and $I = 5$ lattice sites now serves as an example. The block sub-system shall include 2 lattice sites and the site system a single lattice site. The problem takes the easiest form if each block and site basis contain the range of particle numbers from 0 up to the maximum value N. Obviously, states to a particle number $N' > N$ (i.e. a sub-system contains more particles than the total system) will not contribute to the solution.

If block and site basis contain a range of particle number from 0 up to 5 then the superblock basis will exhibit particle numbers from 0 to 10. However, the sub-spaces to particle numbers $N' > N$ can be discarded and the remaining ($\equiv$ reduced) superblock Hamiltonian is transformed on the block Hamiltonian in a way that sub-blocks to certain particle numbers are mapped onto sub-blocks to the same particle numbers (Fig. 4.4). In this way in the last iteration step all sub-spaces to particle numbers inherent in the block and site basis will contribute to the solution. For example the block Hamiltonian sub-block to particle number 5 will combine with the site Hamiltonian sub-block to particle number 0 to form a part of the superblock Hamiltonian sub-block to particle number 5 in which one will search for the total system's solution. Furthermore, the superblock Hamiltonian sub-block to the desired particle number has the largest dimension. This makes sense since with growing Hilbert spaces the information about the prevailing system grows as well.

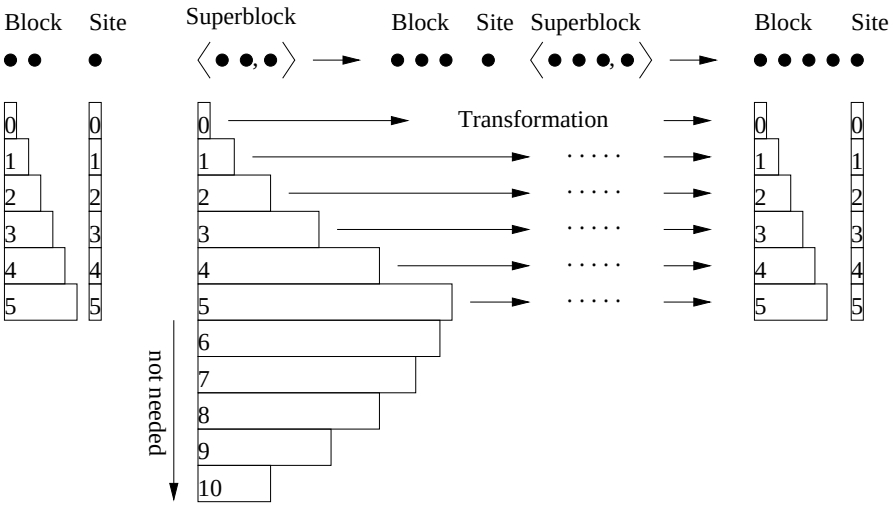

Figure 4.4: *Transformation if block and site bases take all particle numbers into account.*

4.6.2 Consideration of fewer Particle Numbers

If the block and site basis do not take into account all particle numbers from 0 up to N the transformation style has to change. Let the block and site basis of Sec. 4.6.1 contain a range of particle numbers from 0 to 4, then a transformation like in Sec. 4.6.1 is no longer convenient. In the last iteration step sub-blocks will appear in the block and site Hamiltonian that cannot contribute to the solution (in this example the sub-blocks to particle number 0). In the last iteration step the block basis should take the form shown in Fig. 4.5. Therefore,

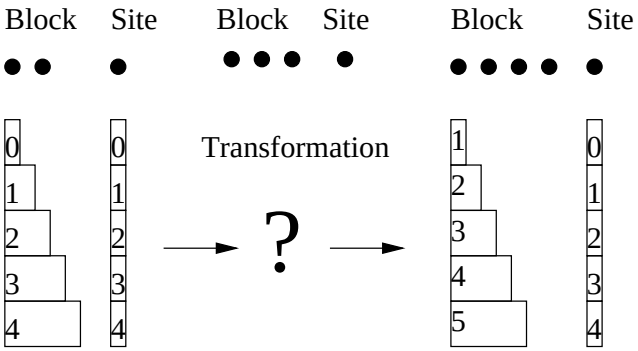

Figure 4.5: *Transformation if block and site bases do not take all particle numbers into account.*

one transformation has to raise the particle numbers of the block basis by transforming sub-blocks to particle numbers $n+1$ of the superblock Hamiltonian to sub-blocks to particle numbers n of the block Hamiltonian. In the case of systems with filling factor 1 it makes sense to increase the particle number as late

as possible. Furthermore, as in Sec. 4.6.1, from dimensional considerations it follows that the desired particle number has to be the maximum particle number in the block basis. In this way one makes sure that in each iteration step the maximum of information is used.

Chapter 5

Results

All NRG calculations were performed with the Mathematica notebook presented in the appendix. In the case of the Hubbard model the NRG method itself allows a number of variations like the modification of the block size or the choice of specific boundary conditions that will have an impact on the numerical results. The influence of the trapping potential on the Hamiltonian (2.26) has not been considered, i.e. $\epsilon_i \equiv 0$.

The NRG method proves itself capable to reproduce qualitatively the dependence of the ground-state energy on the interaction strength U/J. Figure 5.1 shows two sets of results from NRG using different basis dimensions in comparison to the exact solution.

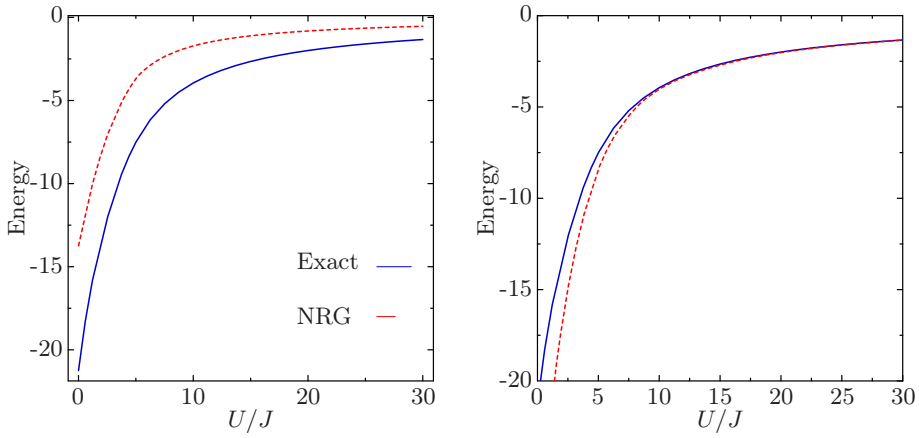

Figure 5.1: *Results of the NRG method applied to an $I = N = 8$ system with different accuracies in comparison with the exact solutions. Left: 2-site block basis with dimension 10; Right: 2-site block basis with dimension 78.*

As expected the energies converge to the exact result in the limit of strong interaction $U \gg J$, where the exact ground states essentially consist only of a

few eigenstates with low fluctuations.

5.1 Influence of Basis Dimensions

The goal of the NRG method is the reduction of the Hilbert space dimension through an iterative truncation to reduce the computational effort. The dimension the Hilbert space is truncated to at every iteration step is determined by the choice of the initial block and the dimension of its associated basis. This dimension depends on the number of sites the block consists of and the number of particles that are taken into account. These will be the main parameters to control the accuracy of the NRG.

5.1.1 2-Site Block Bases

The numerical results presented in this chapter were obtained with an initial block basis consisting of 2 sites. Since we deal with NRG the size of the site system will be fixed to one site but with variable number of particles. Block and site basis will be chosen such that both have the same range of particle numbers. Figure 5.2 introduces the notation for bases that will be used in the following chapters.

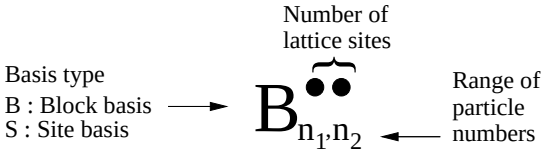

Figure 5.2: *Basis denotation used in the follwing.*

Blockbasis	Sites	$N_{\max}$	Dim.	Sitebasis	Sites	$N_{\max}$	Dim.
$B^{\bullet\bullet}_{0,3}$	2	3	10	$S^{\bullet}_{0,3}$	1	3	4
$B^{\bullet\bullet}_{0,6}$	2	6	15	$S^{\bullet}_{0,6}$	1	6	7
$B^{\bullet\bullet}_{0,8}$	2	8	45	$S^{\bullet}_{0,8}$	1	8	9
$B^{\bullet\bullet}_{0,11}$	2	11	78	$S^{\bullet}_{0,11}$	1	11	12

Table 5.1: *Block and site bases used in this chapter*

Figure 5.3 shows the numerical results of the NRG method using bases from table 5.1. It is evident that the absolute deviations from the exact energies decrease with increasing U/J. As expected, in the domain of dominating interaction the NRG and the exact results are getting closer with increasing block basis dimension. However, for $U/J < 5$ bases with smaller dimensions can lead

to better results. Therefore, one can only expect an improvement of accuracy with increasing basis dimensions in the domain of strong interaction.

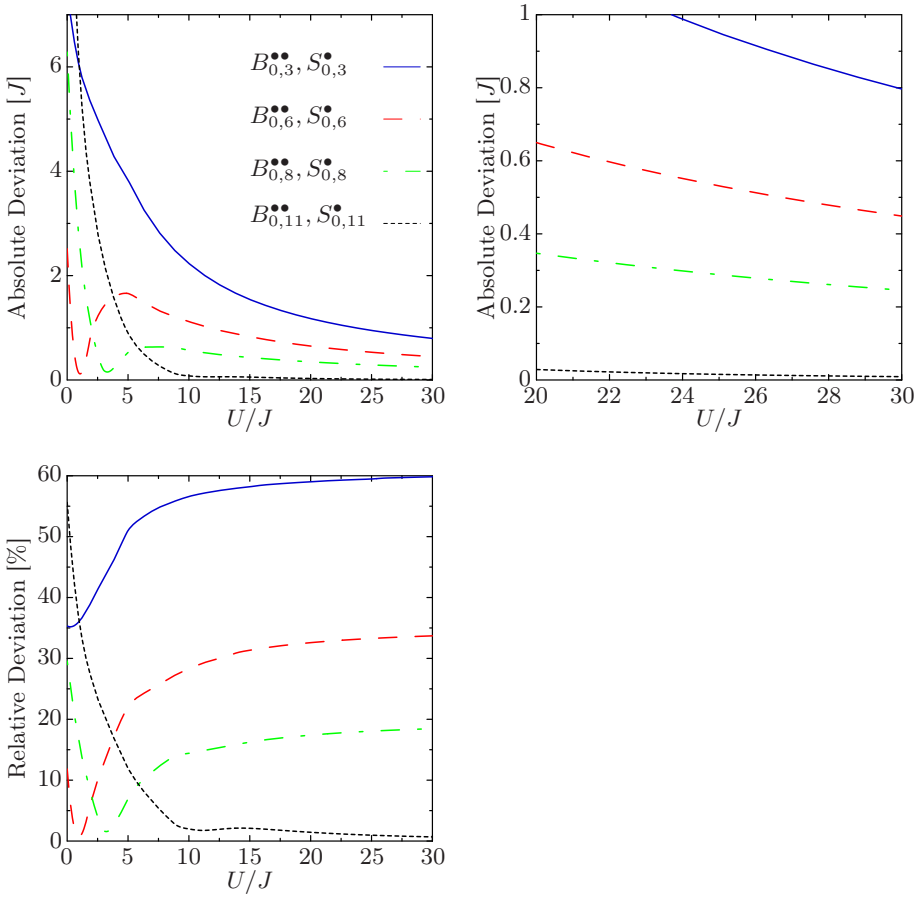

Figure 5.3: *Absolute and percentage deviations of the numerical results to the exact solution of an $I = N = 11$ system.*

A block basis of dimension 78 provides good results for $U/J \geq 10$. The Hilbert space of the exact solution has a dimension of 350.000 so the NRG block basis dimension is tiny what makes this result quite remarkable. For some bases in spite of decreasing absolute deviations the relative deviations increase because the absolute energy gets smaller similar to the behavior shown in Fig. 5.1.

5.1.2 3-Site Block Bases

For the solution of the $I = N = 11$ system from Sec. 5.1.1 it was sufficient to consider particle numbers up to 11 because particle numbers greater than 11 will only produce new subspaces that will not contribute to the solution.

So the dimension of an initial block basis with fixed number of sites has a limit beyond which an improvement of the NRG results is not possible. By

choosing bases for block and site system that take into account particle numbers from 0 up to the maximum number of particles one obtains the best achievable results to a given block basis with fixed number of sites. Therefore, using a 2-site block basis, for the $I = N = 11$ system the limit dimension is determined by $B^{\bullet\bullet}_{0,11}$ to 78 which is quite small.

To increase the block basis dimension anyway, the initial number of sites can be increased. Table 5.2 lists the 3-site block bases and their dimensions used in this chapter.

Blockbasis	Sites	$N_{\max}$	Dim.	Sitebasis	Sites	$N_{\max}$	Dim.
$B^{\bullet\bullet\bullet}_{0,3}$	3	3	20	$S^{\bullet}_{0,3}$	1	3	4
$B^{\bullet\bullet\bullet}_{0,5}$	3	5	56	$S^{\bullet}_{0,5}$	1	5	6
$B^{\bullet\bullet\bullet}_{0,6}$	3	6	84	$S^{\bullet}_{0,6}$	1	6	7
$B^{\bullet\bullet\bullet}_{0,7}$	3	7	120	$S^{\bullet}_{0,7}$	1	7	8

Table 5.2: *Block and site bases used in this chapter*

In comparison to a NRG using 2-site block bases $B^{\bullet\bullet}_{0,N}$ the 3-site block bases $B^{\bullet\bullet\bullet}_{0,N}$ (site basis each time $S^{\bullet}_{0,N}$) will lead to better results. Both bases contain the same number of subspaces to particle numbers, but in the case of 3 initial sites the dimension of each subspace is either equal or greater than in the 2-site case.

Figure 5.5 compares results of the NRG using $B^{\bullet\bullet}_{0,N}$ with the NRG using $B^{\bullet\bullet\bullet}_{0,N}$. Figure 5.4 illustrates the different growth of $B^{\bullet\bullet}_{0,N}$ and $B^{\bullet\bullet\bullet}_{0,N}$ basis dimensions for increasing N.

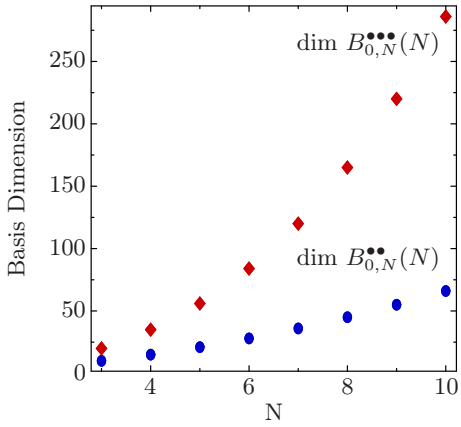

Figure 5.4: *Basis dimensions for 2- and 3-site bases with all particle numbers taken into account.*

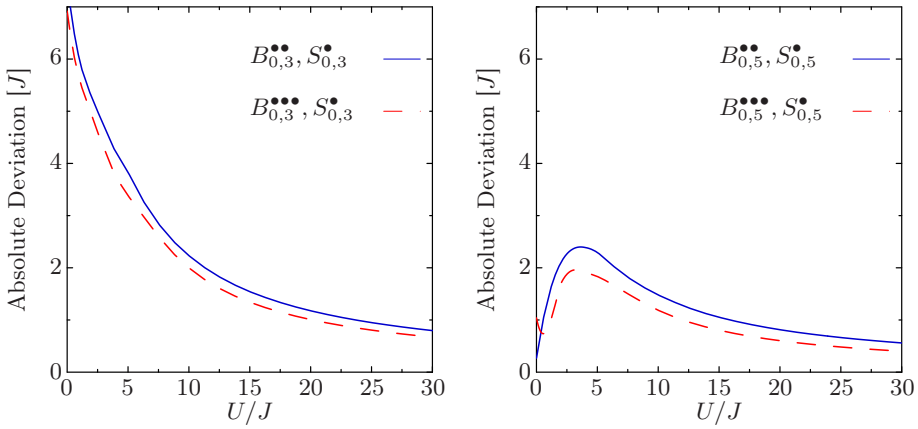

Figure 5.5: *Absolute deviations of 2- and 3-site block bases to the exact solution of an $I = N = 11$ system.*

However, considering only the block basis dimension is not enough to estimate the quality of the results. Figure 5.6 shows that a 3-site block basis with larger dimension than a 2-site basis but with fewer particle-number sub-spaces can yield less accurate results, especially in the domain of strong interaction. Hence, the absence of subspaces has a greater negativ influence on the results than a smaller dimension of these subspaces. Therefore, in practice one should prefer a small number of sites but a large range of particle numbers.

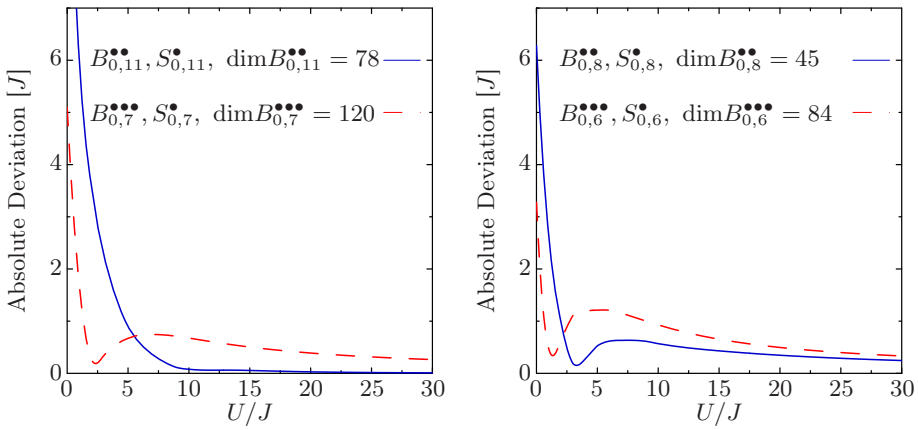

Figure 5.6: *Comparison of the absolute deviations of 2- and 3-site block bases from the exact solution of an $I = N = 11$ system. Although the 3-site basis has a larger dimension it provides worse results than the 2-site basis because its range of particle numbers is smaller.*

5.2 Excited States

In addition to ground states the NRG method is able to produce approximations for excited states too. Figure 5.7 shows NRG results using the bases $B^{\bullet\bullet}_{0,11}, S^{\bullet}_{0,11}$ for the first excited state of an $I = N = 11$ system in comparison with the exact results. The larger relative deviation that emerges around $U/J \approx 8$ is a consequence of the zero-crossing of the energy value. One obtains good results for $20 \leq U/J \leq 30$, where the relative deviation is clearly under 1 %.

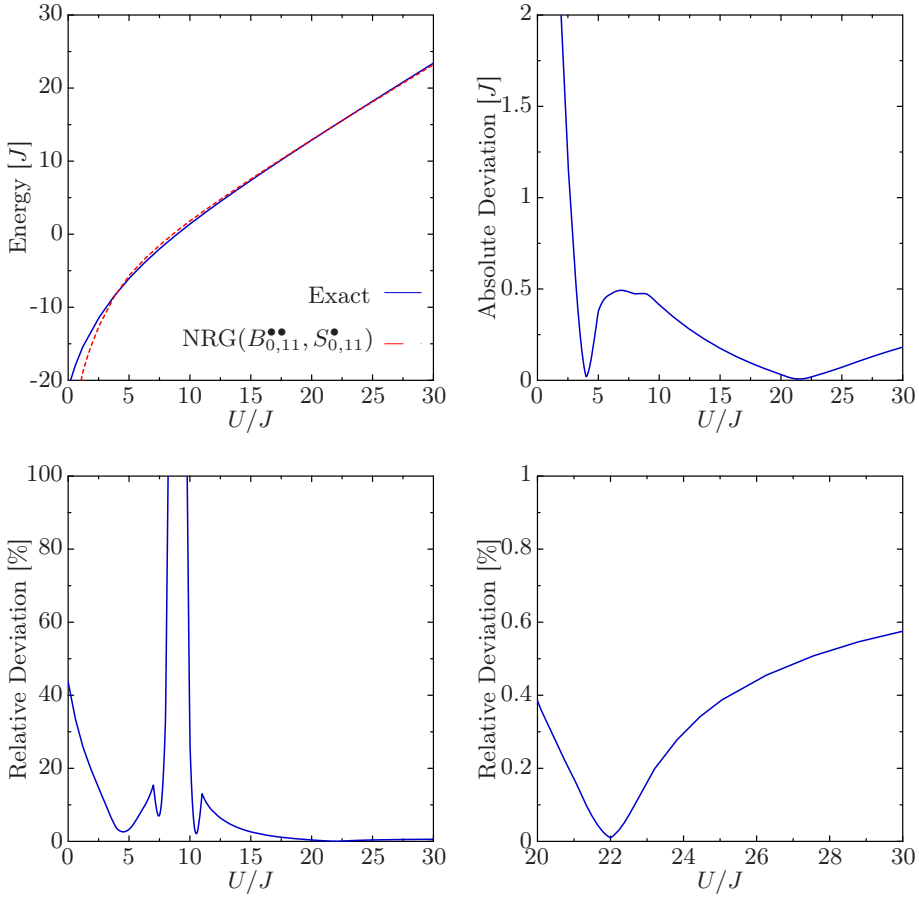

Figure 5.7: *The NRG result for the first excited state of an $I = N = 11$ system in comparison to the exakt solution.*

Since the employed method is designed for the calculation of ground states it is actually not quite correct for the excited states. One expects a loss of accuracy with increasing energy of the excited state. For instance, without an adaption of the choice of the eigenstates used for transformation a particle-number subspace of dimension 1 will always contain only the information of the ground state and no information about any excitation. With increasing excitation level less and less subspaces will contain information about the considered state. Figure

5.8 shows the absolute deviations of NRG results for the first 50 respectively 60 excited states of a $I = N = 9$ resp. $I = N = 10$ system to the exact values.

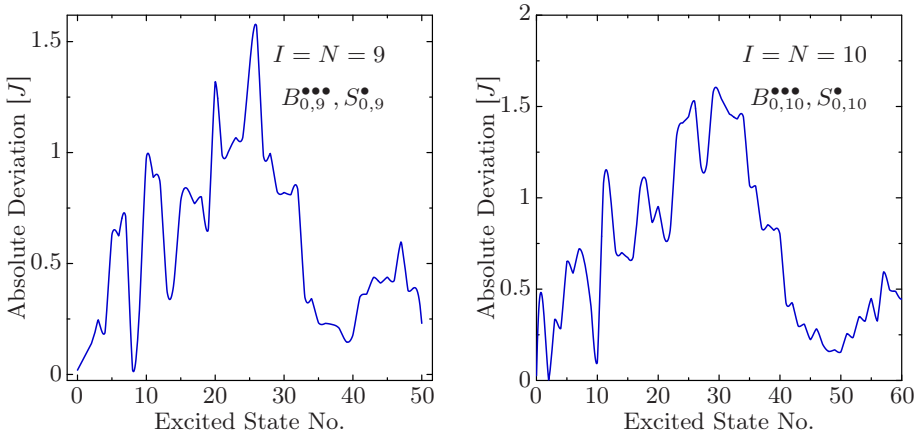

Figure 5.8: *Absolue deviations of the numerical results to the exact solutions for the excited states of an $I = N = 9$ and $I = N = 10$ system at fixed $U/J = 20$.*

The tendency to worsen with excitation level is visible for the first 25 and 30 excited states, then a surprising phase of improvement of the NRG results appears after which the deviations begin to worsen again.

5.3 Growing Lattices and Error Estimate

With growing lattice dimensions more iteration steps are necessary to solve the problem. Since every iteration step is associated to a numerical error, one expects a decay of accuracy for growing lattices. Fig. 5.9 shows the best achievable results of an initial 2-site block basis for growing lattices in comparison to the exakt values.

5.3.1 Error Estimate

Considering the NRG solution of an $I = N = 11$ system using the bases $B^{\bullet\bullet}_{0,11}, S^{\bullet}_{0,11}$ one realizes that the way to this solution leads through the solutions of all $I' = N'$ systems with $I' = N' < 11$ (in this example, in the first iteration step the $I' = N' = 3$ system is solved, in the second step the $I' = N' = 4$ system and so on). Since the initial bases take all particle numbers into account from 0 up to 11, these interim results are the best achievable results for the $I' = N' < 11$ systems.

Therefore the increase of the absolute deviations Δ_i of two subsequent best achievable results (the results from iteration steps $i - 1$ and i) represents the error the NRG method produces in the iteration step i. Looking at Figure 5.9, Δ_i seems to be approximately constant at a fixed U/J so this could be used for

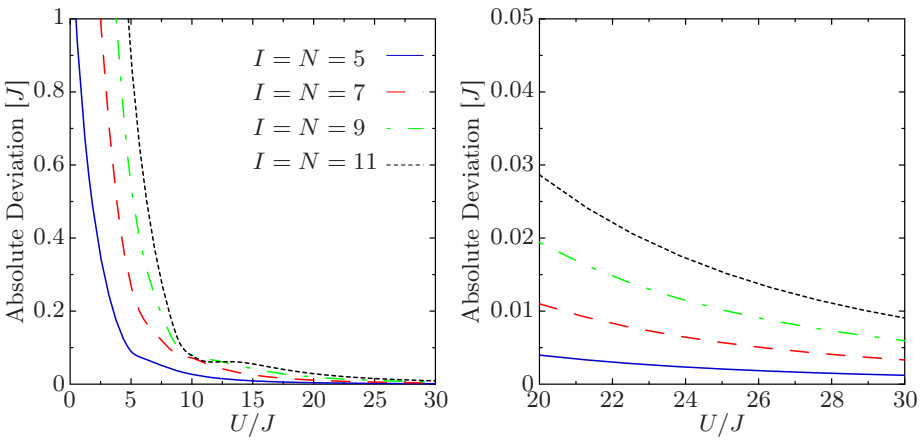

Figure 5.9: *Absolute deviations of best achievable results from the exact solutions. For larger systems these deviations grow since more iteration steps are needed to solve the problem*

a prediction of the error margins. Figure 5.10 (left) displays more precisely how the best achievable results deviate more and more from the exakt solution with each iteration step.

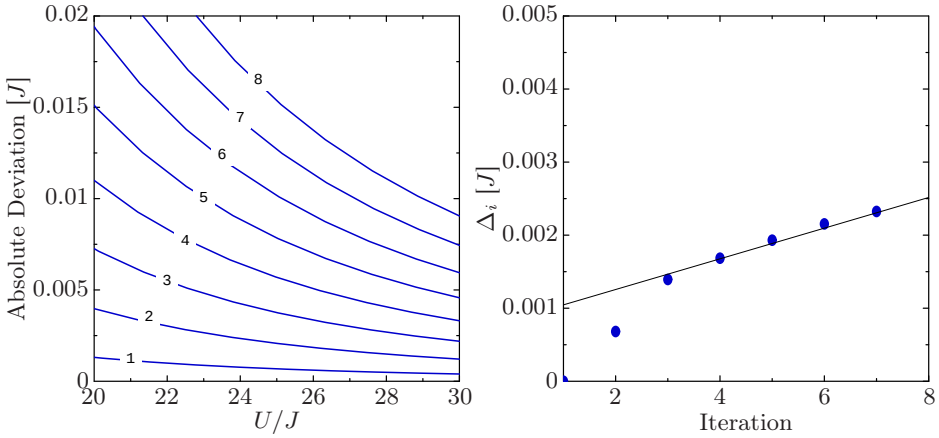

Figure 5.10: *Evolution of the error produced by the NRG procedure. Left: Absolute deviations of the best achievable results of the first 8 iteration steps. Right: Error produced between two iteration steps.*

Figure 5.10 (right) shows the development of Δ_i for a 2-site block basis at fixed $U/J = 25$. Δ_i seems not quite to be constant but shows a rather linear dependence after the third iteration step. On the supposition that this behavior holds for further iteration steps one can predict the error margin of the numerical results for a $I = N$ system for this special case of a 2-site block basis with all particle numbers $B^{\bullet\bullet}_{0,N}$ and fixed $U/J = 25$:

$$\frac{\Delta_{U/J=25}(I)}{J} = 8.36 \cdot 10^{-4} + 2.10 \cdot 10^{-4} \cdot I, \quad I \geq 3. \tag{5.1}$$

A NRG calculation for the $I = N = 20$ system for $U/J = 25$ using $B^{\bullet\bullet}_{0,20}, S^{\bullet}_{0,20}$ yields as the best achievable result -3.076, so, if $\Delta_i(I)$ keeps its linearity, after (5.1) the exact value is estimated to be within the interval -3.076 ± 0.005.

This error estimation applies to NRG results that employ bases that take all particle numbers into account. Under these circumstances the error emerges from the transformations of particle-number subspaces from the superblock Hamiltonian with dimensions D_{Super} into subspaces from the block Hamiltonian with lesser dimensions $D_B < D_{\text{Super}}$.

If the initial bases do not contain all particle numbers, in every iteration step some subspaces have to be discarded because the block Hamiltonian does not provide enough subspaces (see Sec. 4.6). So the choice of specific subspaces that are transformed in every iteration step produces another error that can not be treated as easily as the former consideration.

5.4 Periodic Boundary Conditions

So far, all results were calculated for box boundary conditions. This boundary conditions force the wave functions to vanish at the border of the superblock

For homogeneous systems periodic boundary conditions are often used which requires wave functions that obey

$$\psi(x_{\text{begin}}) = \psi(x_{\text{end}}). \tag{5.2}$$

For periodic boundary conditions, as discussed in Ref. [9], the renormalization group yields results that are much worse than the open boundary counterparts, with relative deviations of up to 2000 %.

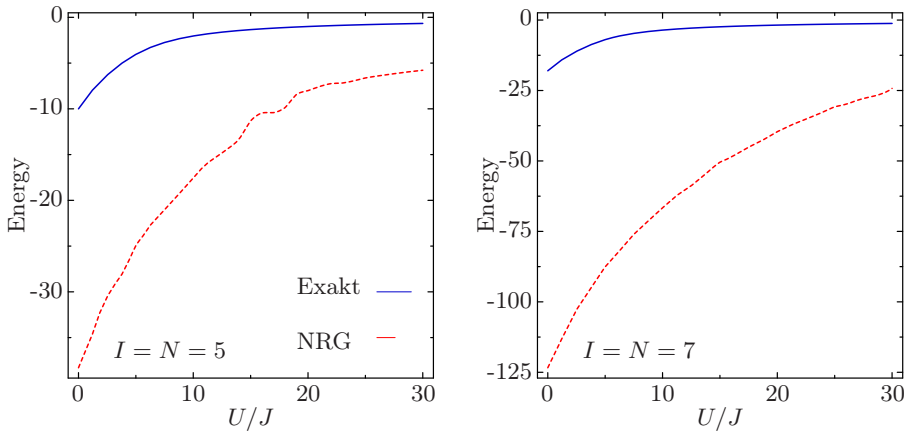

Figure 5.11: *The best possible results of NRG using 2-site block bases and periodic boundary conditions.*

Appendix A

NRG applied to the Hubbard Model with Mathematica

A.1 Modules

This chapter presents the Mathematica notebook used for the NRG calculations. Some parts (A.1.1, A.1.3, A.1.10) were borrowed from Felix Schmitt's implementation of DMRG and many others were at least deeply inspired by him.

The Mathematica environment provides an comparatively easy access to a NRG calculation but it becomes apparent that it suffers from performance problems. Surprisingly, not the actual diagonalization takes most of the CPU time but the matrices' transitions into different bases. Figure A.1 shows the CPU time needed for the different procedures within the NRG calculation and how this CPU times evolves with growing basis dimensions.

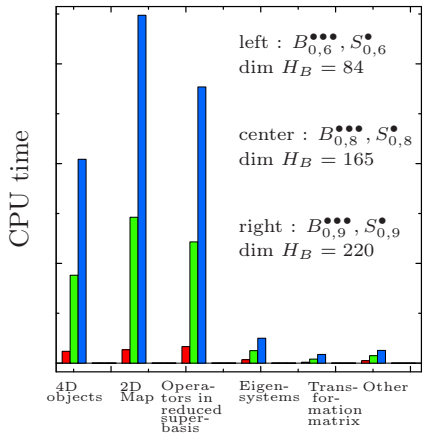

Figure A.1: *Comparison of the CPU time needed for the different operations for increasing block and site basis dimensions.*

A.1.1 CreateBasis[]

`CreateBasis[]` creates the initial bases for the block and the site system. These bases have to contain a certain range of particle numbers. Usually the minimum particle number is 0 since usually one considers small filling rates.

These created bases have to be sorted by particle numbers so that the block Hamiltonian gets block form in respect to particle numbers. The bases created in `CreateBasis[]` are automatically sorted by particle numbers by the way they are created so an explicit sorting will not be necessary.

A complete basis to a given particle number consists of all possible occupation number arrangements on the lattice sites whose sums yield this given particle number. This is exactly what `Compositions[]` does so `Compositions[]` will have to be executed for each particle number of the range the basis contains, and the results have to be stored in `Basis`.

`Basis` has to be flatted one level to get a form in which `Basis` denotes the ith basis vector.

```
CreateBasis[SitesNumber_, ParticlesNumberMin_, ParticlesNumberMax_] :=
 Module[{ Basis, ParticleNumbersLoop },

      Basis = {};

      For[ ParticleNumbersLoop  = ParticlesNumberMin,
           ParticleNumbersLoop <= ParticlesNumberMax,
           ParticleNumbersLoop++,

           Basis = Append[Basis, Compositions[ParticleNumbersLoop, SitesNumber]];
        ]; (* For ParticleNumbersLoop *)

      Basis = Flatten[Basis, 1];

      Return[Basis];
  ]; (* Module *)
```

A.1.2 CreateTransformationsList[]

According to Sec. 4.4 the transformation matrix transforms sub-spaces to particle numbers of the superblock and the block Hamiltonian into each other. The information which sub-spaces are transformed whereto will be stored in `TransformationsList`.

This module only works properly for filling factor 1 so it will have to be modified for other filling rates. Alternatively, `TransformationsList` can in case of need easily be created by hand too.

`TransformationsList` exhibits the structure

$$\begin{array}{llll} \{ & \{ & \{SH_1^{(1)}, BH_1^{(1)}\}, \{SH_2^{(1)}, BH_2^{(1)}\}, \dots \}, \\ & \{ & \{SH_1^{(2)}, BH_1^{(2)}\}, \{SH_2^{(2)}, BH_2^{(2)}\}, \dots \}, \\ & \vdots & \\ & \{ & \{SH_1^{(k)}, BH_1^{(k)}\}, \{SH_2^{(k)}, BH_2^{(k)}\}, \dots \} \quad \} \end{array}$$

where adjoined $SH_i^{(t)}, BH_i^{(t)}$ denote particle numbers to the sub-spaces of block and superblock Hamiltonian that are to be transformed into each other in iteration step t. For example a transformation list for the case that the first iteration sub-spaces to the same particle number and in the second iteration step sub-spaces of the superblock Hamiltonian to particle numbers $n+1$ are transformed into sub-spaces of the block Hamiltonian to particle numbers n (as it would be for the solution of an $I = N = 5$ system using $B_{0,4}^{\bullet\bullet\bullet}, S_{0,4}^{\bullet}$) is given by

$$\begin{array}{lll} \{ & \{ & \{0,0\},\{1,1\},\{2,2\},\{3,3\},\{4,4\}\}, \\ & \{ & \{1,0\},\{2,1\},\{3,2\},\{4,3\},\{5,4\}\} \quad \}. \end{array}$$

The entries each for the block and superblock particle numbers have to be rising to be properly evaluated.

```
CreateTransformationsList[] :=
   Module[{ RaisingTransformationsNumber,ConstantTransformationsNumber,
            ConstantTransformationsLoop, RaisingTransformationsLoop,
            TransformationsList, Transformation },

      TransformationsList = {};

   RaisingTransformationsNumber = FinalParticlesNumber - BlockBasisParticlesNumberMax;
   ConstantTransformationsNumber = IterationsNumber - RaisingTransformationsNumber;

      For[ ConstantTransformationsLoop = 1,
           ConstantTransformationsLoop <= ConstantTransformationsNumber,
           ConstantTransformationsLoop++,

           Transformation = Table[{ BlockBasisParticlesNumberMin + Loop - 1,
                                    BlockBasisParticlesNumberMin + Loop - 1 },
                       { Loop,
                         BlockBasisParticlesNumberMin - BlockBasisParticlesNumberMin + 1,
                         BlockBasisParticlesNumberMax - BlockBasisParticlesNumberMin + 1 }
                                ]; (* Table *)

           TransformationsList = Append[TransformationsList, Transformation];
        ]; (* For ConstantTransformationsLoop *)

     For[ RaisingTransformationsLoop = 1,
          RaisingTransformationsLoop <= RaisingTransformationsNumber,
          RaisingTransformationsLoop++,

          Transformation = Table[
            { BlockBasisParticlesNumberMin + Loop - 1 + RaisingTransformationsLoop,
              BlockBasisParticlesNumberMin + Loop - 1 + RaisingTransformationsLoop - 1 },
                  { Loop,
                    BlockBasisParticlesNumberMin - BlockBasisParticlesNumberMin + 1,
                    BlockBasisParticlesNumberMax - BlockBasisParticlesNumberMin + 1 }
                             ]; (* Table *)

          TransformationsList = Append[TransformationsList, Transformation];
        ]; (* For RaisingTransformationsNumber *)
   Return[TransformationsList];
 ];
```

A.1.3 CreateCreatorAnnihilatorMatrices[]

The Hamiltonian of an I-lattice site system (2.26) can completely be expressed by the use of creation and annihilation operators acting on the lattice sites. For

this reason these operators are calculated for each lattice site and from them the single terms of the Hamiltonian are constructed. Depending on the basis which is given to the module the operators for the block or for the site are calculated.

Submodul: CreateAnnihilatorMatricesOnSite[]

This sub-module calculates the annihilation operator matrices for a given lattice site. The matrix elements are given by

$$(\hat{a})_{ij} = \langle i|\hat{a}|j\rangle \tag{A.1}$$

where for annihilation operators

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle \tag{A.2}$$

holds (boson).

A matrix element $\langle i|\hat{a}|j\rangle$ exists if the new state $\hat{a}|j\rangle$ that arises from applying the operator on $|j\rangle$ is part of the basis too. Then the basis states $|j\rangle$ and $|i\rangle$ are connected by the operator $\hat{a}$, the matrix element is given by (A.1.3) and its position within the matrix is determined by $|j\rangle$ and $|i\rangle$'s positions within the basis.

In general the creation operator is the annihilator's adjoint operator so it suffices to calculate the annihilator matrix elements only and get the creators by transposition.

The module is called with the basis and the lattice site the operator is acting on.

```
CreateAnnihilatorMatricesOnSite[Basis_, Site_] :=
   Module[{ BasisDimension, TempVector, AnnihilatorMatrix,
            BasisVectorLoop, MatrixElement, BasisVectorCompareLoop },
```

First the basis' dimension is determined. The annihilator matrix then is a square matrix of this dimension. This matrix is created as a sparse array so that only the non-zero matrix elements have to be stored.

```
     BasisDimension = Length[Basis];
     AnnihilatorMatrix = SparseArray[{1, 1} -> 0, {BasisDimension, BasisDimension} ];
```

The annihilator is applied on each basis vector what regarding the occupation numbers means a decrease of the lattice site's occupation number. If the annihilator acts on the vacuum the matrix element is 0 and the loop can be aborted. If not, the matrix element is calculated via (A.1.3).

```
     For[ BasisVectorLoop = 1,
          BasisVectorLoop <= BasisDimension,
          BasisVectorLoop++,

          TempVector = Basis[[BasisVectorLoop]];

          TempVector[[RightDoubleBracket]]--;
          If[ TempVector[[Site]] < 0, Continue[] ];

          MatrixElement = Sqrt[ TempVector[[Site]] + 1];
```

The new state `TempVektor` is compared with each basis vector. If `TempVektor` matches an existing basis vector the matrix element is inserted at the right position. Then the loop can be aborted since `TempVektor` can match only one basis vector.

```
          For[ BasisVectorCompareLoop = 1,
               BasisVectorCompareLoop <= BasisDimension,
               BasisVectorCompareLoop++,

               If[ TempVector == Basis[[BasisVectorCompareLoop]],
                   AnnihilatorMatrix += SparseArray[
                           {BasisVectorCompareLoop, BasisVectorLoop} -> MatrixElement,
                           {BasisDimension, BasisDimension}
                                              ]; (* SparseArray *)
                   Break[];
                 ]; (* If *)
             ]; (* For BasisVectorCompareLoop *)
```

After having applied the annihilator to all basis vectors the complete annihilator matrix is returned.

```
  Return[AnnihilatorMatrix];
]; (* Module *)
```

CreateCreatorAnnihilatorMatrices[]

Using the sub-module for all lattice sites the annihilator matrices are calculated and from then the creator matrices are gained which will successively be stored in `AnnihilatorMatrices` and `CreatorMatrices` so one gets access to the annihilator acting on ith lattice site the via `AnnihilatorMatrices[[i]]`.

The corresponding basis has to be handed over to the module.

```
CreateCreatorAnnihilatorMatrices[Basis_] :=
   Module[{ SitesNumber, TempMatrix, AnnihilatorMatrices,
            CreatorMatrices, SitesLoop },
```

The number of lattice sites is determined by a basis vector's length (number of occupation number entries $\equiv$ number of lattice sites).

```
     SitesNumber = Length[ Basis[[1]] ];

     AnnihilatorMatrices = {};
     CreatorMatrices = {};
```

For each lattice sites the annihilator matrix is obtained using the sub-module and is stored with its transpose in the corresponding list.

```
     For[ SitesLoop = 1,
          SitesLoop <= SitesNumber,
          SitesLoop++,

          TempMatrix = CreateAnnihilatorMatricesOnSite[Basis, SitesLoop];

          AnnihilatorMatrices = Append[AnnihilatorMatrices, TempMatrix];

          CreatorMatrices = Append[CreatorMatrices, Transpose[TempMatrix]];
        ]; (* For SitesLoop *)
```

After the loop over all the lattice sites the complete lists can be returned.

```
    Return[ {   CreatorMatrices, AnnihilatorMatrices  } ];
  ]; (* Module *)
```

CreateHamiltonMatrix[]

The Module `CreateHamiltonMatrix[]` creates depending on the basis and list of operators that was handed over the Hamiltonians for block resp. site.

The tunneling term

$$-J \sum_{i=1}^{I-1} \left\{ \hat{a}_i^\dagger \hat{a}_{i+1} + \hat{a}_{i+1}^\dagger \hat{a}_i \right\} \tag{A.3}$$

is calculated and thereafter the interaction term

$$\frac{U}{2} \sum_{i=1}^{I} \hat{a}_i^\dagger \hat{a}_i^\dagger \hat{a}_i \hat{a}_i. \tag{A.4}$$

```
CreateHamiltonMatrix[Basis_, CreatorMatrices_, AnnihilatorMatrices_] :=
   Module[{ SitesLoop, SitesNumber, BasisDimension, HamiltonMatrix },

      BasisDimension = Length[Basis];
      HamiltonMatrix = SparseArray[ {1, 1} -> 0, {BasisDimension , BasisDimension } ];

      SitesNumber = Length[ Basis[[1]] ];

      For[ SitesLoop = 1,
           SitesLoop <= SitesNumber - 1,
           SitesLoop++,

           HamiltonMatrix += - TunnelParameter *
            ( CreatorMatrices[[SitesLoop]] . AnnihilatorMatrices[[SitesLoop + 1]] +
              CreatorMatrices[[SitesLoop + 1]] . AnnihilatorMatrices[[SitesLoop]] );
        ];  (* For SitesLoop *)

      For[ SitesLoop = 1,
           SitesLoop <= SitesNumber,
           SitesLoop++,

           HamiltonMatrix += InteractionParameter / 2 *
            ( CreatorMatrices[[SitesLoop]] . CreatorMatrices[[SitesLoop]] .
              AnnihilatorMatrices[[SitesLoop]] . AnnihilatorMatrices[[SitesLoop]] );
         ]; (* For SitesLoop *)
     Return[HamiltonMatrix];
   ]; (* Module *)
```

OperatorsInFullHilbertSpace

The annihilation and creation operators as well as the block's and site's Hamiltonians were represented in their respective basis. To be able to construct the block-site tunneling term H_{BS} in the superblock Hamiltonian

$$H_{\text{Super}} = H_B + H_{BS} + H_S \tag{A.5}$$

a transfer of these operators into the superblock basis has to take place. More precisely, only the block and site Hamiltonians and the boundary operators of

block and site have to be transfered. The matrix elements in this basis are given by (4.11). There are two separated modules for the block and site operators since the site operators has to transfered only once.

```
SiteOperatorsInFullHilbertSpace[SiteOperator_] :=
   Module[{ SiteOperator4D, ArrayRulesListe,
            OperatorMatrixElementLoop,
            DeltaLoop, RowPositionMatrixElement,
            ColoumnPositionMatrixElement },
```

The operator in the superblock Hilbert space is created as a 4D object since a matrix element $(O)_{kk'll'}$ has 4 indices. Into `ArrayRulesList` the replacement rules for the matrix elements are loaded. Using this list, one gets direct access to the non-zero matrix elements and no loop over the whole object will be needed.

```
      SiteOperator4D = SparseArray[ {1, 1, 1, 1} -> 0,
                              { BlockBasisDimension, BlockBasisDimension,
                                SiteBasisDimension, SiteBasisDimension }
                                  ]; (* SparseArray *)

      ArrayRulesListe = ArrayRules[SiteOperator];
```

Each matrix element is read in from the old operator representation. The loop runs only to `Length[ArrayRulesList] -1` because the last entry of `ArrayRulesList` contains noinformation about a matrix element.

```
      For[ OperatorMatrixElementLoop = 1,
           OperatorMatrixElementLoop <=  Length[ArrayRulesListe] - 1,
           OperatorMatrixElementLoop++,

           RowPositionMatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 1, 1]];
           ColoumnPositionMatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 1, 2]];

           MatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 2]];
```

The matrix element $(O)_{ll'}$ of a site operator will in the 4D objects appear plurally, after (4.11) at all positions $(kk'll')$ where the block basis vectors are equally $k = k'$.

```
           For[ DeltaLoop = 1,
                DeltaLoop <= BlockBasisDimension,
                DeltaLoop++,

                SiteOperator4D[[ DeltaLoop, DeltaLoop,
                                 RowPositionMatrixElement, ColoumnPositionMatrixElement
                               ]] = MatrixElement;
              ]; (* For DeltaLoop *)
         ]; (* For OperatorMatrixElementLoop *)
     Return[SiteOperator4D ];
  ];
```

The same, slightly adapted, considerations hold for the block operators.

```
BlockOperatorsInFullHilbertSpace[BlockOperator_] :=
   Module[{ BlockOperator4D, ArrayRulesListe,
            OperatorMatrixElementLoop, DeltaLoop,
            RowPositionMatrixElement, ColoumnPositionMatrixElement },
```

```
    BlockOperator4D = SparseArray[ {1, 1, 1, 1} -> 0,
                            { BlockBasisDimension, BlockBasisDimension,
                              SiteBasisDimension, SiteBasisDimension }
                                 ]; (* SparseArray *)

    ArrayRulesListe = ArrayRules[BlockOperator];

    For[ OperatorMatrixElementLoop = 1,
         OperatorMatrixElementLoop <= Length[ArrayRulesListe] - 1,
         OperatorMatrixElementLoop++,

         RowPositionMatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 1, 1]];

         ColoumnPositionMatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 1, 2]];

         MatrixElement = ArrayRulesListe[[OperatorMatrixElementLoop, 2]];

         For[ DeltaLoop = 1,
              DeltaLoop <= SiteBasisDimension,
              DeltaLoop++,

              BlockOperator4D[[ RowPositionMatrixElement, ColoumnPositionMatrixElement,
                                DeltaLoop, DeltaLoop ]] = MatrixElement;
             ]; (* For DeltaLoop *)
       ]; (* For OperatorMatrixElementLoop *)
    Return[BlockOperator4D];
  ]; (* Module *)
```

A.1.4 ParticlesNumberInState[]

In various situations it will be necessary to know the particle number of a given state. This particle number is simply determined by the sum over all occupation numbers.

```
ParticlesNumberInState[State_] :=
   Module[{ TempState, SitesNumber, SitesLoop, ParticlesNumber },

      TempState = Flatten[State];
      SitesNumber = Length[TempState];

      ParticlesNumber = Sum[TempState[[SitesLoop]], {SitesLoop, 1, SitesNumber} ];
   Return[ParticlesNumber];
  ]; (* Module *)
```

A.1.5 CreateSuperBasis[]

The superblock basis is the product basis from the block and the site basis. This "full superblock basis" and therefore the superblock Hamiltonian contains the range of particle numbers

$$N_{\text{Block, min}} + N_{\text{Site, min}} \ \ldots \ N_{\text{Block, max}} + N_{\text{Site, max}} \ . \tag{A.6}$$

The superblock Hamiltonian doesn't have to be diagonalized for all these particle numbers, but for those occuring in the transformation list. That is why besides the full superblock basis the reduced superblock basis is created which only consists of basis vectors with needed particle numbers. The whole information

needed then stands in the superblock Hamiltonian expressed in the reduced superblock basis.

However, one can not express the operators in the reduced superblock basis from the beginning because some matrix elements will be lost. This happens because single creation or annihilation operators do not conserve particle numbers but the occuring operator products of annihilators and creators do.

```
CreateSuperBasis[] :=
   Module[{ ParticlesNumberMin, ParticlesNumberMax, SiteBasisLoop, BlockBasisLoop,
            ReducedSuperBasis, FullSuperBasis, ParticlesNumber },
```

The reduced superbasis' range of particle numbers has to be determined. Since this range of particle numbers contains all the particle numbers whose sub-blocks shall be transformed one finds the needed information in the transformation list.

```
     ParticlesNumberMin = Min[Transpose[Transformations[[IterationsLoop]]] [[1]] ];
     ParticlesNumberMax = Max[Transpose[Transformations[[IterationsLoop]]] [[1]] ];

     ReducedSuperBasis = {};
     FullSuperBasis = {};
```

The full superblock basis yields from all possible combinations of a block and a site basis vector.

```
     For[ SiteBasisLoop = 1,
          SiteBasisLoop <= SiteBasisDimension,
          SiteBasisLoop++,

          For[ BlockBasisLoop = 1,
               BlockBasisLoop <= BlockBasisDimension,
               BlockBasisLoop++,

               FullSuperBasis = Append[FullSuperBasis,
                     { BlockBasis[[BlockBasisLoop]], SiteBasis[[SiteBasisLoop]] }
                                ]; (* Append *)
```

To find the reduced superblock's basis vectors the particle number of each created vector is determined. If this particle number lies within the range the reduced superblock's basis particle numbers it will be appended to `ReducedSuperBasis`.

```
               ParticlesNumber = ParticlesNumberInState[{ BlockBasis[[BlockBasisLoop]],
                                                          SiteBasis[[SiteBasisLoop]] }];

               If[ Or[ParticlesNumber < ParticlesNumberMin,
                      ParticlesNumber > ParticlesNumberMax],
                      Continue[];
                 ]; (* If *)

               ReducedSuperBasis = Append[ReducedSuperBasis,
                           { BlockBasis[[BlockBasisLoop]], SiteBasis[[SiteBasisLoop]] }
                                      ]; (* Append *)
          ]; (* For BlockBasisLoop *)
       ]; (* For SiteBasisLoop *)
    Return[{FullSuperBasis, ReducedSuperBasis}];
  ]; (* Module *)
```

A.1.6 OperatorMap2D[]

Since the annihilators, creators and Hamiltonians exist as 4D objects they have to be mapped to a 2D form to be able to build their matrix products from the rules of matrix multiplication. After mapping they are represented in the full superblock basis which then conforms to the basis constructed via `CreateSuperBasis[]`.

```
OperatorMap2D[Operator_] :=
   Module[{ Operator2D, MatrixElementsNumber, MatrixElementLoop,
      BlockRowPosition, BlockColumnPosition, SiteRowPosition,
      SiteColumnPosition },

      Operator2D = SparseArray[ {1, 1} -> 0,
                                { SiteBasisDimension*BlockBasisDimension,
                                  SiteBasisDimension*BlockBasisDimension }
                              ]; (* SparseArray *)

      ArrayRulesListe = ArrayRules[Operator];

      MatrixElementsNumber = Length[ArrayRules[Operator]] - 1;

      For[ MatrixElementLoop = 1,
           MatrixElementLoop <= MatrixElementsNumber,
           MatrixElementLoop++,

           MatrixElement = ArrayRulesListe[[MatrixElementLoop, 2]];

           BlockRowPosition = ArrayRulesListe[[MatrixElementLoop, 1, 1]];
           BlockColumnPosition = ArrayRulesListe[[MatrixElementLoop, 1, 2]];

           SiteRowPosition = ArrayRulesListe[[MatrixElementLoop, 1, 3]];
           SiteColumnPosition = ArrayRulesListe[[MatrixElementLoop, 1, 4]];

           Operator2D[[(SiteRowPosition - 1)*BlockBasisDimension + BlockRowPosition,
                       (SiteColumnPosition - 1)*BlockBasisDimension + BlockColumnPosition
                     ]] = MatrixElement;
        ]; (* For MatrixElementLoop *)
    Return[Operator2D];
  ]; (* Module *)
```

A.1.7 CreateSuperHamiltonMatrix[]

The superblock Hamiltonian is given by

$$H_{\text{Super}} = H_B + H_{BS} + H_S \tag{A.7}$$

with the block-site tunneling matrix H_{BS}

$$H_{BS} = -J\left(\hat{a}_B^\dagger \hat{a}_S + \hat{a}_S^\dagger \hat{a}_B\right) \tag{A.8}$$

where $\hat{a}_{B,S}^{(\dagger)}$ denote the boundary creation and annihilation operators on the border between block and site. Since these operators are expressed in the full superblock basis the superblock Hamiltonian constructed this way still contains all particle numbers (A.6).

```
CreateSuperHamiltonMatrix[] :=
   Module[{ BlockSiteTunnelMatrix2D, SuperHamiltonMatrix },

      BlockSiteTunnelMatrix2D = - TunnelParameter *
           ( BlockBoundaryCreator2D . SiteBoundaryAnnihilator2D +
             SiteBoundaryCreator2D . BlockBoundaryAnnihilator2D );

 SuperHamiltonMatrix = BlockHamiltonMatrix2D + BlockSiteTunnelMatrix2D + SiteHamiltonMatrix2D;

   Return[SuperHamiltonMatrix];
  ]; (* Module *)
```

A.1.8 OperatorInReducedSuperBasis[]

Expressing the superblock Hamiltonian in the reduced superblock Basis has several advantages. The superblock Hamiltonian is block diagonal with respect to particle numbers and by the transistion into the reduced superblock basis the sub-blocks to unneeded particle numbers are dropped. For the construction of the transformation matrix one will have to diagonalize each sub-block separately and the transition into the reduced superblock basis makes sure that all existing sub-blocks have to be diagonalized. Furthermore, all operator matrices that shall be transformed have to be expressed in the reduced superblock basis.

Same as in `CreateSuperBasis[]` the range of particle numbers is gained from the transformation list. The module checks which basis states from the full superblock basis do not lie in the desired range of particle numbers and deletes the corresponding rows and columns from the matrices. This works because the 2D map conserves the sorting of the superblock's basis states. One has to start from behind because deleting rows and columns affect the following rows' and lines' positions.

```
OperatorInReducedSuperBasis[Operator2D_] :=
   Module[{ ParticlesNumberMin, ParticlesNumberMax,
            ReducedOperator, FullSuperBasisLoop },

      ParticlesNumberMin = Min[Transpose[Transformations[[IterationsLoop]]] [[1]] ];
      ParticlesNumberMax = Max[Transpose[Transformations[[IterationsLoop]]] [[1]] ];

      ReducedOperator2D = Operator2D;

      For[ FullSuperBasisLoop = Length[FullSuperBasis],
           FullSuperBasisLoop >= 1 ,
           FullSuperBasisLoop--,

           ParticlesNumber = ParticlesNumberInState[FullSuperBasis[[FullSuperBasisLoop]] ];

           If[ Or[ ParticlesNumber < ParticlesNumberMin,
                   ParticlesNumber > ParticlesNumberMax ],
               ReducedOperator2D = Delete[ReducedOperator2D, FullSuperBasisLoop];
               ReducedOperator2D = Transpose[
                                    Delete[Transpose[ReducedOperator2D], FullSuperBasisLoop]
                                           ]; (* Transpose *)
             ]; (* If *)
         ]; (* For FullSuperBasisLoop*)
    Return[ReducedOperator2D];
  ]; (* Module *)
```

A.1.9 CreatePositionsList[]

`CreatePositionsList[]` returns two different things. `PositionsList` is needed to build the sorting matrix. This matrix will later be used to transfer the reduced superblock Hamiltonian and other operator matrices into a basis in which they have block form. `ParticleSpaceDimensions` provides the information on how big a sub-space to a given particle number is in this basis. This information will be needed at very least when the transformation matrix is built.

```
CreatePositionsList[Basis_] :=
   Module[{ BasisDimension, ParticlesNumberListe, ParticlesNumberMax,
            ParticlesNumberMin, BasisLoop, ColoumnLoop,  ParticlesNumber,
            PositionsList, ParticlesNumberLoop, ParticleSpaceDimensions },
```

First the range of particle number of the basis is determined. Since this module works with any basis this range can not be gained from the transformations list (that only provides the particle numbers of the reduced superblock basis). Using `Map[]` the particle number to each basis state is determined and stored in `ParticlesNumberList`. Minimum and maximum numbers are obtained directly from this list.

```
     BasisDimension = Length[Basis];

     ParticlesNumberList = Map[ParticlesNumberInState, Basis];
     ParticlesNumberMin = Min[ParticlesNumberList];
     ParticlesNumberMax = Max[ParticlesNumberList];
```

At the end `PositionsList` is supposed to have the structure

$$\begin{array}{lll} \{ & \{ & p_1(N_{\min}),\ p_2(N_{\min}),\ \ldots\}, \\ & \{ & p_1(N_{\min}+1),\ p_2(N_{\min}+1),\ \ldots\}, \\ & \vdots & \\ & \{ & p_1(N_{\max}),\ p_2(N_{\max}),\ \ldots\} \quad \} \end{array}$$

where $p_j(N)$ denotes the position of the jth basis vector to particle number N. So `PositionsList` is created as list with `ParticlesNumberMax+1` empty sub-lists. `ParticlesNumberMax+1` since 0 is a possible value for `ParticlesNumberMin`. However, if `ParticlesNumberMin` > 0 one has created too many sub-lists so that at the end these surplus lists will be deleted. This way it is easier to store the positions.

```
     PositionsList = Table[{}, {ColoumnLoop, 1, ParticlesNumberMax + 1}];
```

The particle number N_i of each basis vector i is determined and stored in the $N_i + 1$th sub-list of `PositionsList`.

```
     For[ BasisLoop = 1,
          BasisLoop <= BasisDimension,
          BasisLoop++,

          ParticlesNumber = ParticlesNumberInState[Basis[[BasisLoop]]];
          PositionsList[[ParticlesNumber + 1]] =
                 Append[ PositionsList[[ParticlesNumber + 1]], BasisLoop];
        ]; (* For BasisLoop *)
```

Now the surplus sub-lists are deleted. The kth sub-list corresponds to particle number $k+1$ so `PositionsList` gets the desired form.

```
PositionsList = Drop[PositionsList, ParticlesNumberMin];
```

The determination of the sub-spaces' dimensions confines on counting the basis vectors to a given particle number. This number can directly be obtained by the number of entries of the corresponding sub-lists in `PositionsList`.

`ParticleSpaceDimensions` is supposed to have the form

$$\begin{array}{lll} \{ & \{ & N_{\min},\ \dim\,(N_{\min})\}, \\ & \{ & N_{\min}+1,\ \dim\,(N_{\min}+1)\}, \\ & \vdots & \\ & \{ & N_{\max},\ \dim\,(N_{\max})\} \quad \}\ . \end{array}$$

```
ParticleSpaceDimensions = {};

For[ ParticlesNumberLoop = 1,
     ParticlesNumberLoop <= Length[PositionsList],
     ParticlesNumberLoop++,

     ParticlesNumber = ParticlesNumberMin + ParticlesNumberLoop - 1;

     ParticleSpaceDimensions = Append[ ParticleSpaceDimensions,
            { ParticlesNumber, Length[PositionsList[[ParticlesNumberLoop]]] }
                                      ]; (* Append *)
   ]; (* For ParticlesNumberLoop *)
  Return[{PositionsList, ParticleSpaceDimensions}];
]; (* Module *)
```

A.1.10 CreateSortMatrix[]

The sorting matrix' task is to transfer operators expressed in the reduced superblock basis into the same basis that is sorted by particle numbers so that the operators get block form with respect to particle numbers.

```
CreateSortMatrix[] :=
  Module[{ PositionsList, SortMatrix, SuperBasisLoop },

    PositionsList = CreatePositionsList[SuperBasis][[1]];
```

The sorting matrix shall exchange rows and columns of the operator matrices. This can be achieved by an identity matrix whose entries are shifted in the right way. For example, an identity matrix $\mathbb{1}'$ where the ith and jth row has been exchanged will, when applied to another matrix M, exchange M's ith and jth columns. Therefore, $\mathbb{1}'\ M\ \mathbb{1}'^T$ will exchange M's ith and jth rows and columns as desired in this case.

```
    SortMatrix = SparseArray[{1, 1} -> 0, {SuperBasisDimension, SuperBasisDimension}];

    For[ SuperBasisLoop = 1,
         SuperBasisLoop <= SuperBasisDimension,
         SuperBasisLoop++,
```

```
        SortMatrix += SparseArray[
                        {SuperBasisLoop, Flatten[PositionsList][[ SuperBasisLoop]]} -> 1,
                        {SuperBasisDimension, SuperBasisDimension}
                                ]; (* SparseArray *)
      ]; (* For SuperBasisLoop *)
    Return[SortMatrix];
 ]; (* Module *)
```

A.1.11 CreateSortetSuperHamiltonEigenSystems[]

This module solves the eigenvalue problem of the reduced superblock Hamiltonian that, according to Sec. 4.4.1 can be solved for each sub-space to particle numbers seperately. The Hamiltonian has been sorted by the sorting matrix so it has block form which will be used to direct access to the sub-spaces to particle numbers. The dimensions of these sub-spaces have been stored in `SuperHamiltonParticleSpaceDimensions`.

```
CreateSortetSuperHamiltonEigenSystems[] :=
   Module[{ SubSpacesNumber, SubSpaceLoop, ParticleSpacePosition,
            SubSpaceDimension, SubSpace, SubSpaceEigenSystem,
            SuperHamiltonEigenSystems },

     SuperHamiltonEigenSystems = {};
     SubSpacesNumber = Length[SuperHamiltonParticleSpaceDimensions];
```

`ParticleSpacePosition, ParticleSpacePosition` denotes the positions of the sub-block in the (reduced and sorted) superblock Hamilonian. `SubMatrix[]` takes these sub-blocks at these positions using the dimensions from `SuperHamiltonParticleSpaceDimensions`. Since `SubMatrix[]` does not work with `SparseArray`s one has to transfer them to the standard matrix form first.

```
     ParticleSpacePosition = 1;

     For[ SubSpaceLoop = 1,
          SubSpaceLoop <= SubSpacesNumber,
          SubSpaceLoop++,

          SubSpaceDimension = SuperHamiltonParticleSpaceDimensions[[SubSpaceLoop, 2]];
          SubSpace = SubMatrix[ Normal[SortetSuperHamiltonMatrix],
                                {ParticleSpacePosition, ParticleSpacePosition},
                                {SubSpaceDimension, SubSpaceDimension}
                              ]; (* SubMatrix *)
```

For each sub-block the eigenvalue problem is solved. Since later eigenvectors are selected by their energies the eigensystems are sorted by the energy eigenvalues. All eigensystems are stored in `SuperHamiltonEigenSystems`.

```
          SubSpaceEigenSystem = Eigensystem[N[SubSpace]] // Chop;
          SubSpaceEigenSystem = Transpose[Sort[Transpose[SubSpaceEigenSystem]]];
          SubSpaceEigenSystem[[2]] = Map[Normalize,SubSpaceEigenSystem[[2]]];

          SuperHamiltonEigenSystems = Append[SuperHamiltonEigenSystems, SubSpaceEigenSystem];
```

The sequent's sub-block position is given the by the actual's sub-block position plus its dimension.

```
        ParticleSpacePosition += SubSpaceDimension;
    ]; (* For SubSpaceLoop *)
  Return[SuperHamiltonEigenSystems];
]; (* Module *)
```

A.1.12 CreateTransformationMatrix[]

The transformation matrix transforms the reduced superblock Hamiltonian and the block's boundary operators. As described in 4.4.3 the transformation matrix is a rectangular matrix that transform the superblock Hamiltonians's sub-blocks into the block Hamiltonian's sub-blocks. Since the sub-blocks from the superblock Hamiltonian has in general greater dimensions than the ones from the block Hamiltonian the transformation matrix has to reduce their dimensions so that it will have block form with rectangular blocks. This is the point where the actual truncation of the Hilbert space takes place.

```
CreateTransformationMatrix[] :=
  Module[{ TransformationsMatrix, FrontZeros,  BackZeros,
           SubSpaceLoop, SubSpaceEigenVectors, SubSpaceZeilenLoop },
```

The transformation matrix is rectangular matrix whose dimension is determined by the dimensions of the block and site Hamiltonian/basis.

```
    TransformationsMatrix = SparseArray[
                {1, 1} -> 0, {BlockBasisDimension, SuperBasisDimension}
                                    ]; (* SparseArray *)
```

Due to the block structure there will be a certain number of zero matrix elements before and behind the actual matrix elements of a block. Regarding the module's performance these zero entries at the front and the back are inserted in one stretch so that a loop only has to run over the eigenvectors that have to be inserted. `TransformationsMatrixRow` stores the number of the actual row of the transformation matrix.

```
    FrontZeros = {};
    BackZeros = SparseArray[{1} -> 0, {SuperBasisDimension}];

    TransformationsMatrixRow = 1;
```

All blocks to different particle numbers are created separately. Such a block consists of the number of eigenvectors that matches the dimension of the target sub-block.

```
    For[ SubSpaceLoop = 1,
         SubSpaceLoop <= Length[BlockHamiltonParticleSpaceDimensions],
         SubSpaceLoop++,

         SubSpaceEigenVectors =
         SortetSuperHamiltonEigenSystems[[SubSpaceLoop, 2]];
```

This adjusts the number of zeroes before the eigenvector entries.

```
        If[ SubSpaceLoop != Length[BlockHamiltonParticleSpaceDimensions],
            BackZeros = Take[ BackZeros,
              Length[BackZeros] - SuperHamiltonParticleSpaceDimensions[[SubSpaceLoop, 2]]
                          ], (* Take *)
            BackZeros = {}
          ]; (* If *)
```

These are the actual entries. A line of the transformation matrix is composed by the front zeroes, an eigenvector, and the zeroes to the end of the line.

```
        For[ SubSpaceRowLoop = 1,
             SubSpaceZeilenLoop <= BlockHamiltonParticleSpaceDimensions[[SubSpaceLoop, 2]],
             SubSpaceRowLoop++,

             TransformationsMatrix[[TransformationsMatrixRow]] =
                 { FrontZeros, SubSpaceEigenVectors[[SubSpaceRowLoop]], BackZeros }
                 // Flatten;

             TransformationsMatrixRow++;
           ]; (* For SubSpaceRowLoop *)
```

This adjusts the number of zeroes behind the eigenvector entries.

```
        FrontZeros = SparseArray[ {1} -> 0,
         {Length[FrontZeros] + SuperHamiltonParticleSpaceDimensions[[SubSpaceLoop, 2]]}
                              ]; (* SparseArray *)
      ]; (* For SubSpaceLoop *)
   Return[TransformationsMatrix];
 ]; (* Module *)
```

A.1.13 CreateNewBlockBasis[]

After the first iteration the concrete form of the block basis doesn't matter any more. It is only necessary to know the particle numbers to the basis vectors, for example to determine the particle numbers of the new superblock basis states. Therefore it is enough to store the information about the particle numbers in their corresponding basis vectors.

From the transformation list one obtains the information which sub-spaces of the superblock Hamiltonian have been transformed into which sub-spaces of the block Hamiltonian. Therewith one can determine the particle numbers of the block basis states.

```
CreateNewBlockBasis[] :=
  Module[{ NewBlockBasis, BlockBasisLoop, OldParticlesNumber, NewParticlesNumber },

     NewBlockBasis = BlockBasis;

     For[ BlockBasisLoop = 1,
          BlockBasisLoop <= BlockBasisDimension,
          BlockBasisLoop++,

          OldParticlesNumber = ParticlesNumberInState[BlockBasis[[BlockBasisLoop]]];
          NewParticlesNumber =
               Transformations[[
                              IterationsLoop,
                              Position[
                                      Transpose[
```

```
                                                    Transformations[[IterationsLoop]]
                                                   ][[2]],
                                            OldParticlesNumber
                                           ] // Flatten,
                                      1
                                ]];

          NewBlockBasis[[BlockBasisLoop]] = NewParticlesNumber;
        ];
    Return[NewBlockBasis]
  ]; (* Module *)
```

A.1.14 CreateEnergyList[]

`CreateEnergyList[]` creates lists with the results of the individual iteration steps. Using the `MatrixForm` output it has the form

$$\begin{array}{lll} \text{number of sites} & \text{number of particles} & \text{energy eigenvalue no 1} \\ \text{number of sites} & \text{number of particles} & \text{energy eigenvalue no 2} \\ \vdots & & . \end{array}$$

```
CreateEnergyList[] :=
   Module[{ SitesNumber, NewEnergies, BlockBasisLoop,
            Energy, ParticlesNumber },
      If[ IterationsLoop == 1, EnergyList = {} ];

      SitesNumber = BlockBasisSitesNumber  + IterationsLoop * SiteBasisSitesNumber;

      NewEnergies = {};

      For[ BlockBasisLoop = 1,
           BlockBasisLoop <= BlockBasisDimension,
           BlockBasisLoop++,

           Energy = BlockHamiltonMatrix[[BlockBasisLoop, BlockBasisLoop]];
           ParticlesNumber = BlockBasis[[BlockBasisLoop]];

           NewEnergies = Append[NewEnergies, Flatten[{SitesNumber, ParticlesNumber, Energy} ]];
        ]; (* For BlockBasisLoop *)

      EnergyList = Append[EnergyList, NewEnergies];
    Return[EnergyList];
  ];
```

A.2 NRG

This section shows how the NRG calculations can be done using the modules presented in the preceding chapters.

Some modules use Mathematica procedures from packages that are not standardly loaded at the start-up so these packages have to be loaded manually.

```
<< DiscreteMath‘Combinatorica‘

<< LinearAlgebra‘MatrixManipulation‘

<< LinearAlgebra‘Orthogonalization‘
```

The initial parameters of the NRG method are listed below. `FinalParticlesNumber` and `FinalSitesNumber` determines N and I of the system. The initial block and site bases are determined by their number of sites they consist and their range of particle numbers which is defined by the minimum and maximum initially occuring particle number.

```
TunnelParameter = 1;
InteractionParameter = 20;

FinalParticlesNumber = 11;
FinalSitesNumber = 11;

BlockBasisSitesNumber = 2;
BlockBasisParticlesNumberMin = 0;
BlockBasisParticlesNumberMax = 11;

SiteBasisSitesNumber = 1;
SiteBasisParticlesNumberMin = 0;
SiteBasisParticleNumberMax = 11;
```

`IterationsNumber` is the number of iterations needed to solve the problem. It only depends on the number of lattice sites in the initial block system and the number of sites in the site system that are appended in each iteration step. One has to take care that the desired number of lattice sites can be exactly reached by the chosen sizes of block and sites. However, as long as 1-site site systems are used this problem does not arise.

```
IterationsNumber = (FinalSitesNumber-BlockBasisSitesNumber) / SiteBasisSitesNumber;
```

Block and site basis are created with respect to their properties that has been set above and their dimensions are determined.

```
BlockBasis =  CreateBasis[ BlockBasisSitesNumber,
                           BlockBasisParticlesNumberMin,
                           BlockBasisParticlesNumberMax ];

SiteBasis = CreateBasis[ SiteBasisSitesNumber,
                         SiteBasisParticlesNumberMin,
                         SiteBasisParticleNumberMax ];

BlockBasisDimension = Length[BlockBasis];
SiteBasisDimension = Length[SiteBasis];
```

From the parameters the transformations list is built which tells what subspaces are transformed and where they are transformed to.

```
Transformations = CreateTransformationsList[];
```

Get the matrix representations of the creation and annihilation matrices.

```
{BlockCreatorMatrices, BlockAnnihilatorMatrices} =
                       CreateCreatorAnnihilatorMatrices[BlockBasis];

{SiteCreatorMatrices, SiteAnnihilatorMatrices} =
                       CreateCreatorAnnihilatorMatrices[SiteBasis];
```

From these operator matrices the block and site Hamiltonians are gained.

```
BlockHamiltonMatrix = CreateHamiltonMatrix[ BlockBasis,
                                            BlockCreatorMatrices,
                                            BlockAnnihilatorMatrices ];

SiteHamiltonMatrix = CreateHamiltonMatrix[ SiteBasis,
                                           SiteCreatorMatrices,
                                           SiteAnnihilatorMatrices ];
```

Get the boundary operators for block and site from the operator lists. The block boundary operator acts on the rightmost lattice site of the block system and is therefore the last entry in the list of the block creation matrices. In contrast the site boundary acts on the (if there are more than 1 lattice sites) leftmost lattice site and is given by the first entry of the site creation matrices list. Since the annihilator matrices can be obtained by transposing the creator matrices it is sufficient to get the creator matrices only.

```
BlockBoundaryCreator = BlockCreatorMatrices[[Length[BlockCreatorMatrices]]];
SiteBoundaryCreator = SiteCreatorMatrices[[1]];
```

Now construct the 4D objects of matrix elements $(O)_{kk'll'}$ from the site operators and Hamiltonian. This procedure remains the same for any arbitrary block basis with fixed dimension. So the result only depends on the site basis (which determines the site operator matrix elements). Because the site basis does not change (and the block basis dimension is kept constant) during the NRG procedure these 4D objects do not change either. So it is enough to create them once (which means outside the iteration loop that starts below).

```
SiteBoundaryCreator = SiteOperatorsInFullHilbertSpace[SiteBoundaryCreator];
SiteHamiltonMatrix = SiteOperatorsInFullHilbertSpace[SiteHamiltonMatrix];
```

Since the site 4D objects remain the same over the NRG procedure their 2D representations do not change on their part. So they have to be created only once, too. These matrices represent in each iteration step the operators in the full superblock bases that have not necessarily to be known at this moment.

```
SiteBoundaryCreator2D = OperatorMap2D[SiteBoundaryCreator];
SiteBoundaryAnnihilator2D = Transpose[SiteBoundaryCreator2D];

SiteHamiltonMatrix2D = OperatorMap2D[SiteHamiltonMatrix];
```

Erase no longer needed variables from the memory.

```
Clear[BlockCreatorMatrices];     Clear[BlockAnnihilatorMatrices];
Clear[SiteCreatorMatrices];      Clear[SiteAnnihilatorMatrices];
```

Here the actual iteration starts.

```
For[ IterationsLoop = 1,
     IterationsLoop <= IterationsNumber,
     IterationsLoop++,
```

Create the block operator 4D objects and from them the 2D matrix representations. In each iteration step the block operator's matrix elements change, so this procedure has to be executed each time.

```
BlockBoundaryCreator = BlockOperatorsInFullHilbertSpace[BlockBoundaryCreator];
BlockHamiltonMatrix = BlockOperatorsInFullHilbertSpace[BlockHamiltonMatrix];

BlockBoundaryCreator2D = OperatorMap2D[BlockBoundaryCreator];
BlockHamiltonMatrix2D = OperatorMap2D[BlockHamiltonMatrix];
BlockBoundaryAnnihilator2D = Transpose[BlockBoundaryCreator2D];
```

Create the full superblock basis, `FullSuperBasis`, and the reduced one, `SuperBasis`. Needed in the follwoing is the dimension of the reduced superblock basis so this is stored in an extra variable.

```
FullSuperBasis = CreateSuperBasis[][[1]];

SuperBasis = CreateSuperBasis[][[2]];
SuperBasisDimension = Length[SuperBasis];
```

Create the superblock Hamiltonian and transfer it into the reduced superblock basis. Thereafter the block boundary creator has to be transferd into this basis as well, so that it can later be transformed with the same transformation that transformes the reduced superblock Hamiltonian.

```
SuperHamiltonMatrix = CreateSuperHamiltonMatrix[];

SuperHamiltonMatrix = OperatorInReducedSuperBasis[SuperHamiltonMatrix];
BlockBoundaryCreator2D = OperatorInReducedSuperBasis[BlockBoundaryCreator2D];
```

Create the sorting matrix and sort the reduced superblock Hamiltonian and the block boundary creator (which is a transfer into a reduced superblock basis that is sorted by particle numbers).

```
SortMatrix = CreateSortMatrix[];

SortetSuperHamiltonMatrix = SortMatrix.SuperHamiltonMatrix.Transpose[SortMatrix];
SortetBlockBoundaryCreator2D = SortMatrix.BlockBoundaryCreator2D.Transpose[SortMatrix];
```

For the construction of the transformation matrix the sub-space dimension of the block and superblock Hamiltonians have to be known.

```
BlockHamiltonParticleSpaceDimensions = CreatePositionsList[BlockBasis][[2]]];
SuperHamiltonParticleSpaceDimensions = CreatePositionsList[SuperBasis][[2]];
```

Solve the reduced superblock Hamiltonians eigenvalue problem and create the transformation matrix from its eigensystems.

```
SortetSuperHamiltonEigenSystems = CreateSortetSuperHamiltonEigenSystems[];

TransformationsMatrix = CreateTransformationMatrix[];
```

Transform the the reduced superblock Hamiltonian and the block boundary creator into the new block basis. This is the main step in the NRG procedure because here is where the superblock Hamiltonian is rotated into its truncated eigenbasis.

```
    BlockHamiltonMatrix = TransformationsMatrix.
                          SortetSuperHamiltonMatrix.
                            Transpose[TransformationsMatrix] // Chop;

    BlockBoundaryCreator = TransformationsMatrix.
                           SortetBlockBoundaryCreator2D.
                            Transpose[TransformationsMatrix] // Chop;
```

Create the new block basis that from now on only contains the information about the particle number of the basis vectors.

```
    BlockBasis = CreateNewBlockBasis[];
```

Store this iteration step's energy results.

```
    EnergiesList = CreateEnergiesList[] ;
```

Make some free memory and finish this iteration step.

```
    Clear[BlockBoundaryCreator2D];     Clear[BlockBoundaryAnnihilator2D];
    Clear[BlockHamiltonMatrix2D];      Clear[SiteBoundaryCreator2D];
    Clear[SiteBoundaryAnnihilator2D];  Clear[SiteHamiltonMatrix2D];
    Clear[SuperHamiltonMatrix];        Clear[SortMatrix];
    Clear[SortetSuperHamiltonMatrix];  Clear[SortetBlockBoundaryCreator2D];
    Clear[SortetBlockBoundaryAnnihilator2D];
    Clear[BlockHamiltonParticleSpaceDimensions];
    Clear[SuperHamiltonParticleSpaceDimensions];
    Clear[SortetSuperHamiltonEigenSystems];
    Clear[TransformationsMatrix];

]; (* For IterationsLoop *)
```

After all iterations the result is displayed.

```
EnergiesList[[IterationsNumber]] // MatrixForm
```

Bibliography

[1] **M.H. Anderdon, J.R. Ensher, M.R. Matthews, C.E. Wieman, E.A. Cornell**
*Science **269**, 198-201 (1995)*

[2] **D. Jaksch, C. Bruder, J.I. Cirac, C.W. Gardiner, P. Zoller**
Cold bosonic atoms in optical lattices
*Phys. Review Lett. **81**, 3108-3111 (1998)*

[3] **Wolfgang Nolting**
Grundkurs Theoretische Physik 5.2
Springer Verlag (2004)

[4] **Wolfgang Nolting**
Grundkurs Theoretische Physik 7
Springer Verlag (2005)

[5] **Torsten Fließbach**
Quantenmechanik
Elsevier (2005)

[6] **Walter Greiner**
Theoretische Physik 11
Harri Deutsch (1995)

[7] **Felix Schmitt**
Correlations & Transport Properties of Ultracold Atomic Gases in Optical Lattices
Diploma Thesis, Technische Universität Darmstadt (2005)

[8] **D. Jaksch, P. Zoller**
The cold atom Hubbard toolbox
Annals of Physics 315 (2005)

[9] **U. Schollwöck**
The density-matrix renormalization group
Reviews of Modern Physics, Vol 77 (2005)

[10] **G.G. batrouni, V. Rousseau, R.T. Scalettar**
*Phys. Rev. B **46**, 9051 (1992)*

[11] **J.K. Freericks, H. Monien**
*Phys. Rev. B **53**, 2691 (1996)*

Acknowledgement

Once again the old wisdom was proved well-founded that the topic one works on is less important than the people one works with.

I have to thank Prof. Robert Roth for providing a place in his work group and for his remarkable cooperativeness.

And of course I am deeply indebted to Felix Schmitt who was bothered mostly by all my physical and non-physical questions for answering all of them with extraordinary patience - and for answering some of them over and over again.

Without my parent's support these studies would not have been possible so they should know that I am truly grateful for this.

Many thanks go to Jason Copeland for the kind text corrections.