2 The Density Matrix Renormalization Group

Reinhard M. Noack[1] and Steven R. White[2]

[1] Institut de Physique Théorique, Université de Fribourg
CH-1700 Fribourg, Switzerland
[2] Department of Physics and Astronomy, University of California
Irvine, CA 92697–4575, USA

The Density Matrix Renormalization Group [1] (DMRG) is a numerical technique for finding accurate approximations to the ground state and the low-lying excited states of strongly interacting quantum lattice systems such as the Heisenberg, t-J, and Hubbard models. DMRG traces it roots to Wilson's numerical renormalization group (RG) treatment of impurity problems [2] [see also Chap. 1(I)] and is related to real space renormalization groups. DMRG is remarkable in the accuracy that can be achieved for one-dimensional systems. For example, the ground state energy of the spin-one Heisenberg chain on lattices of hundreds of sites can be calculated to an accuracy of order 10^{-10} with a modest amount of computational effort. In addition, up to a dozen or so excited states with specified conserved quantum numbers as well as virtually any equal-time observable can be calculated. The principal limitation of the method at the present time is dimensionality, or, relatedly, range of the interaction, so that the majority of systems treated up to now have been one-dimensional or quasi-one-dimensional, i.e. strips of finite width. The computational effort necessary to achieve a given accuracy grows rapidly with the width of the system.

In this chapter we will attempt to introduce DMRG in a pedagogical manner. We will first briefly discuss some simpler methods to which DMRG is closely related, including exact diagonalization methods and Wilson's original Hamiltonian-based numerical method. Recently, there have been extensions of the method to the calculation of higher energy properties such as dynamical correlation functions [3], to two-dimensional classical systems [4], to one-dimensional quantum systems at finite temperature [5–7], and to quantum chemical calculation of the states of molecules [8]. Many of these extensions are discussed at length in other chapters of this book. In this chapter, we will concentrate on the original formulation for the ground state properties of quantum lattice systems.

The outline of this chapter is as follows: we first briefly introduce exact diagonalization in Sect. 1, and then discuss Wilson's original numerical renormalization group procedure as applied to quantum lattice systems in Sect. 2. We illustrate the reason for the failure of the Wilson RG for most quantum lattice models using the noninteracting tight-binding particle in Sect. 3. Two

algorithms that overcome these shortcomings for the noninteracting system are also introduced in this section. In Sect. 4, we discuss how to generalize one of these, the superblock procedure, to interacting systems via a projection using the reduced density matrix. The two classes of DMRG algorithms, the infinite system algorithm and the finite system algorithm, are discussed in Sect. 5, along with additional details of the algorithms for interacting systems.

This chapter contains two appendices. Appendix A describes the form of the density matrix for single particle systems, such as the tight-binding chain. Appendix B describes the DMRG algorithm for the tight-binding chain and gives a C++ program to carry out the calculation. We hope that using this program to carry out a simple calculation will provide an excellent introduction to the ideas in DMRG.

1 Exact Diagonalization

Let us assume that we want to find the low-energy properties of strongly interacting quantum lattice models such as the Heisenberg model,

$$H = J \sum_{\langle ij \rangle} \boldsymbol{S}_i \cdot \boldsymbol{S}_j \ , \tag{1}$$

or the Hubbard model,

$$H = -t \sum_{\langle ij \rangle, \sigma} \left(c^\dagger_{i\sigma} c_{j\sigma} + c^\dagger_{j\sigma} c_{i\sigma} \right) + U \sum_i n_{i\uparrow} n_{i\downarrow} \ , \tag{2}$$

where $c^\dagger_{i\sigma}$ creates an electron of spin $\sigma = \uparrow$ or $\downarrow$ on site i, $n_{i\sigma} = c^\dagger_{i\sigma} c_{i\sigma}$, t is the hopping matrix element between neighboring sites, $\langle ij \rangle$ denotes a sum over pairs of nearest neighbor sites, and U is the energy cost due to the Coulomb repulsion of two electrons on the same site. The Heisenberg model can be obtained from the strong coupling (large U) limit of the half-filled Hubbard model with the quantum mechanical spin operator $\boldsymbol{S}_i = c^\dagger_{i\sigma} \boldsymbol{\tau}_{\sigma\sigma'} c_{i\sigma'}$, where $\boldsymbol{\tau}_{\sigma\sigma'}$ is a vector of Pauli matrices, and $J = 4t^2/U$. Here we will consider primarily the one-dimensional version of these models in order to illustrate the DMRG. The sum over nearest neighbors $\langle ij \rangle$ then reduces to a sum over lattice sites i with $j = i + 1$. Although these models can be written down in compact form, they involve many degrees of freedom. The spin-1/2 Heisenberg model on an L-site lattice has 2^L degrees of freedom, and the Hubbard model 4^L.

Unlike classical models whose ground states are usually trivial, but whose thermodynamics are interesting (e.g. the Ising model), the properties of the ground states of these quantum lattice models are difficult to calculate, and are, in general, poorly understood. For example, the question of whether or not the ground state of the doped two-dimensional Hubbard model on a

square lattice is superconducting with $d_{x^2-y^2}$ pairing symmetry and therefore relevant to the high-T_c superconductors has not yet been definitively answered.

Both the Hubbard and the Heisenberg models do have exact solutions in one dimension via the Bethe Ansatz. However, the solution is somewhat unwieldy and only a limited number of properties of the system can be easily extracted. Comparison with the exact solution will nevertheless provide an opportunity to test the performance of the DMRG. In more than one dimension, there are no well-controlled analytical methods to treat these models. Therefore, numerical techniques have become an important means for studying them. The most important numerical methods are quantum Monte Carlo, exact diagonalization, and the numerical renormalization group.

Perhaps the simplest approach to numerically treating a quantum lattice system is to diagonalize the Hamiltonian for a finite-size lattice. A convenient basis for the Hilbert space for such a system has states of the form

$$|1\rangle \otimes |2\rangle \otimes \ldots \otimes |n\rangle \,, \tag{3}$$

where $|i\rangle$ labels a state of a single site of the system, and $\otimes$ denotes a direct product. For example, for a spin-1/2 system, $|i\rangle = \uparrow$ or $\downarrow$. The size of the Hilbert space grows exponentially in the number of sites n. If A is a single-site operator belonging to site i, its matrix elements have the simple form

$$\langle n| \ldots \langle 1|A|1'\rangle \ldots |n'\rangle = \delta_{11'} \ldots \delta_{i-1,i-1'} \delta_{i+1,i+1'} \ldots \delta_{n,n'} A_{i,i'} \,, \tag{4}$$

where $A_{i,i'}$ is the matrix elements of the operator for a single site. Similar expressions hold for products of single-site operators.

The δ-functions in these expressions make the matrix for the Hamiltonian operator in the many-site Hilbert space extremely sparse. Consequently, the usual dense matrix diagonalization methods, such as the Jacobi method or Householder transformation with diagonalization of the resulting tridiagonal matrix, are not the most efficient algorithms for these systems. We are primarily interested in the ground state and possibly a few low-lying excited states. Two widely used and very efficient methods for finding a few eigenstates of a large, sparse matrix are the *Lanczos method* and the *Davidson method*. These methods build up a small set of basis vectors, and minimize the energy within this basis. The reduced set of basis vectors is systematically expanded until convergence is reached. Typically, the original sparse Hamiltonian H only comes into play in the multiplication of a vector v by H, in which case only the nonzero elements of H are relevant. The Davidson method [9], also utilizes the diagonal elements of H in an attempt to generate an improved reduced basis. Consequently, if the Hamiltonian is at all dominated by its diagonal elements, the Davidson method will probably converge more quickly than the Lanczos method.

In efficient exact diagonalization programs, the Hamiltonian matrix is usually not explicitly generated. Instead, a procedure to multiply a vector

by H is used which generates the Hamiltonian matrix elements as they are needed.

Although these diagonalization algorithms work remarkably well for finding the lowest eigenvalues and eigenvectors of the large, sparse matrices found in quantum lattice problems, the maximum system size that can be treated is still severely limited by the exponential growth of the Hilbert space. The principal limitations are then that the computer memory required to store a Hilbert space vector becomes too large to handle, and operations on these vectors become too expensive. In practice, the spin-1/2 Heisenberg model can be diagonalized on up to about 36 sites and the Hubbard model on up to about 20 sites. It would therefore be advantageous to formulate a variational diagonalization scheme that also truncates the Hilbert space used to represent H in a controlled way. This can be done using the *numerical renormalization group.*

2 Wilson's Numerical Renormalization Group

In this section, we will sketch out the basic ideas of Wilson's numerical renormalization group [2], using a notation that will be useful later for discussions of the DMRG. See Chap. 1(I) for a more complete discussion of Wilson's numerical RG as applied to impurity problems. We will postpone discussion of the mechanics of keeping and transforming the operators to a later section, since they are the same as for the DMRG.

The basic idea of the renormalization group is to integrate out unimportant degrees of freedom progressively using a succession of renormalization group transformations. In order to investigate the behavior of the single-impurity Kondo problem, Wilson implemented the RG in a purely *numerical* way [2]. The procedure starts with a numerical representation of the Hamiltonian in a particular basis, then adds degrees of freedom by carrying out a renormalization group transformation, typically by increasing the size of the finite system, and finally numerically transforms the representation of the Hamiltonian to a reduced basis.

To be more concrete, let us consider a one-dimensional quantum lattice model such as the Heisenberg or Hubbard model. In fact, Wilson was able to map the spherically symmetric Kondo problem onto such a one-dimensional quantum lattice model. In the Kondo case, the impurity is represented by the first site and the spherically symmetric momentum shells (logarithmically discretized) of the conduction electrons are represented by a semi-infinite lattice with only near-neighbor interactions.

Wilson's numerical RG procedure then proceeds as follows:

1. Isolate a portion of the system containing L sites. Here L is chosen to be small enough so that the Hamiltonian H_L can be diagonalized exactly.
2. Diagonalize H_L numerically, obtaining the m lowest eigenvalues and eigenvectors.

3. Transform H_L and other operators in the "block" of length L to a new basis consisting of the m lowest eigenvectors of H_L, i.e. form $\bar{H}_L = O_L^\dagger H_L O_L$, $\bar{A}_L = O_L^\dagger A_L O_L$, etc., where the columns of O_L contain the m lowest eigenvectors of H_L, and A_L is an arbitrary operator in the block. Note that $\bar{H}_L$ is a diagonal matrix with m elements.
4. Add a site to $\bar{H}_L$ to form H_{L+1}. In order to do this, the interaction between the block of length L and the additional site added must be reconstructed. We will discuss in more detail how this is done in Sect. 5.
5. Repeat starting with step 2, substituting H_{L+1} for H_L.

This scheme is depicted pictorially in Fig. 1. For a Hamiltonian represented in a real-space basis, the RG step is a real-space blocking scheme. Typically, the number of states m kept at each step is held constant, so the time and memory required for each diagonalization stays the same, and the computer time needed is linear in L.

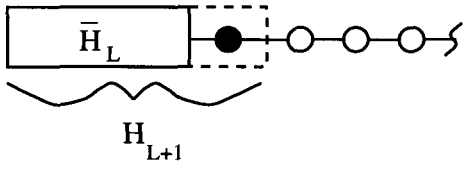

Fig. 1. A pictorial depiction of the Wilson numerical RG procedure.

The basic idea of this scheme is that only the low-energy eigenstates obtained for a system of size L will be important in making up the low-energy states of a system of size $L + 1$. Note that in isolating the block of length L, one has to decide how to treat the boundaries of the block. The simplest thing to do is to neglect connections to surrounding sites, which corresponds to applying open boundary conditions to the system being diagonalized.

This procedure worked well in Wilson's original work on the single impurity Kondo problem and, with minor variations, is still used today for a variety of single and two Kondo and Anderson impurity problems [see Chap. 1(I)]. In his original paper [2], Wilson very carefully justified the truncation by perturbatively calculating the error, and compared the numerical results with analytical analysis of the behavior near the fixed points. Due to the transformation into an energy basis and the logarithmic discretization of the conduction electron "sites", each site corresponds to a successively lower energy scale, and the coupling between successive sites decreases exponentially.

However, when this procedure is applied to other systems for which the lattice model does not include an intrinsic separation of energy scales, such as the one-dimensional Heisenberg or Hubbard models, the accuracy becomes quite poor after just a few iterations [10].

3 Numerical RG for the Particle on a Chain

In order to understand why the Wilson numerical renormalization group procedure breaks down for interacting quantum lattice systems, it is useful to consider first its application to a simple noninteracting quantum lattice problem, a single particle on a tight-binding chain. The Hamiltonian we will consider is

$$H = -\sum_{i=1}^{L-1} (|i\rangle\langle i+1| + |i+1\rangle\langle i|) + 2\sum_{i=1}^{L} |i\rangle\langle i| , \tag{5}$$

where the state $|i\rangle$ corresponds to a localized tight-binding orbital on site i. The matrix elements $H_{ij} = \langle i|H|j\rangle$ are then

$$H_{ij} = \begin{pmatrix} 2 & -1 & 0 & 0 \dots \\ -1 & 2 & -1 & 0 \dots \\ 0 & -1 & 2 & -1 \dots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} . \tag{6}$$

The value 2 on the diagonal is chosen so that this operator is just the discretization of the second derivative operator, $-\partial^2/\partial x^2$.

The Wilson procedure described in the previous section can be carried out on this system with just a few minor differences. First, since this is a one-particle problem, the dimension of the Hilbert space for a lattice of length L is L, rather an exponential of L as in an interacting system. Since the Hilbert space grows less rapidly with system size for the noninteracting systems, we will add two equal-sized blocks in the real-space blocking step, rather than adding a site at a time. Secondly, the mechanics of putting the blocks together is a little simpler.

In step 1 of the Wilson procedure, we isolate a block of length L. We can understand how this is done for the noninteracting system by considering a semi-infinite system broken up into blocks of length L. The Hamiltonian can then be written

$$H = \begin{pmatrix} H_L & T_L & 0 & 0 & \dots \\ T_L^\dagger & H_L & T_L & 0 & \dots \\ 0 & T_L^\dagger & H_L & T_L & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix} . \tag{7}$$

For $L = 1$, the H_1 is a 1×1 matrix with value 2 and T_1 is a 1×1 matrix with value -1. For larger L, H_L has the form of (6), and T_L connects only sites on the block boundaries, i.e. has all zero elements except for a -1 in the lower left corner. Isolating a block then consists of neglecting the T_L, and therefore applies fixed boundary conditions to H_L. Here we use the term "fixed" boundary conditions rather than the equivalent "open" boundary conditions to emphasize that the single-particle wavefunction vanishes at the

boundary. (On the lattice, the wavefunction is actually zero only at the sites *beyond* the boundary, i.e. site 0 and $L+1$, see Fig. 2.)

In step 2, we diagonalize H_L and form

$$O_L = \begin{pmatrix} | & & | \\ \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_m \\ | & & | \end{pmatrix} , \tag{8}$$

where $\boldsymbol{v}_1 \ldots \boldsymbol{v}_m$ are the eigenvectors corresponding to the $m \leq L$ lowest eigenvalues of H_L. In step 3, we form the diagonal $m \times m$ matrix $\bar{H}_L = O_L^\dagger H_L O_L$ and transform the connection between H_L and the rest of the system, T_L to the the new basis by forming $\bar{T}_L = O_L^\dagger T_L O_L$.

We now increase the size of the system, step 4, by putting two blocks of size L together to form a system of size $2L$ with

$$H_{2L} = \begin{pmatrix} \bar{H}_L & \bar{T}_L \\ \bar{T}_L^\dagger & \bar{H}_L \end{pmatrix} \tag{9}$$

and

$$T_{2L} = \begin{pmatrix} 0 & 0 \\ \bar{T}_L & 0 \end{pmatrix} . \tag{10}$$

The procedure can then be repeated starting with step 2 by substituting H_{2L} and T_{2L} for H_L and T_L. The size of the system therefore doubles at each step, but the size of the matrix to be diagonalized is at most $2m \times 2m$. Notice that since we transform to a truncated the basis at each step, the matrix elements of $\bar{H}_L$ and $\bar{T}_L$ can no longer be easily related to the original real-space basis. However, if m were equal to L at each step, the procedure would be exact; it would just be a complicated reshuffling of the original Hamiltonian.

As illustrated in the column labeled "Wilson" in Table 1, this procedure performs quite badly as soon as $m < L$. There are large errors in the energies of the lowest few states after only the first few truncating steps. This failure was pointed out by Wilson at an informal seminar at Cornell University in 1986 as an example of a numerical RG procedure which does not work. He also pointed out that in this simple system it is easy to understand why the procedure fails.

Table 1. Lowest energies after 10 blocking transformations for the noninteracting single particle on a 1-D chain with fixed boundary conditions.

	Exact	Wilson	Fixed-Free
E_0	2.3508×10^{-6}	1.9207×10^{-2}	2.3508×10^{-6}
E_1	9.4032×10^{-6}	1.9209×10^{-2}	9.4032×10^{-6}
E_2	2.1157×10^{-5}	1.9214×10^{-2}	2.1157×10^{-5}
E_3	3.7613×10^{-5}	1.9217×10^{-2}	3.7613×10^{-5}

In the continuum limit, the tight-binding model with fixed boundary conditions describes a particle in a box of length L with an infinitely high po-

tential at the walls. The eigenfunctions are therefore particle-in-a-box eigenfunctions, $\psi_n(x) \sim \sin n\pi x/L$ with n a positive integer, and vanish at the boundaries of the box. In the RG procedure, the lowest few eigenstates of a system of length L are combined to form the low-lying eigenstates of a system of length $2L$. The lowest eigenstates of a system of length L and of length $2L$ are plotted in Fig. 2. Clearly, a combination of the ground states of two systems of length L is a bad approximation to the ground state for a system of length $2L$. Since the wave vectors of the discrete system take on small but finite values on the first and last sites of the system, the "kink" at the boundary between the blocks can be removed, but only by using almost all of the eigenstates of the smaller block.

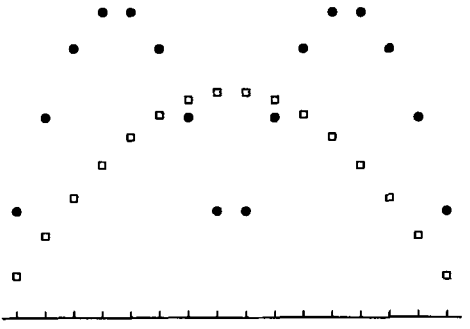

Fig. 2. The lowest eigenstates of two 8-site blocks (solid circles) and a 16-site block (open squares) for the one-dimensional tight-binding model with fixed boundary conditions.

The lesson that is learned is that the treatment of the boundaries of the blocks is crucial in formulating an accurate RG procedure. One could try to fix this problem by applying some other boundary conditions such as periodic boundary conditions to the block rather than fixed boundary conditions. While it is possible to formulate such an RG procedure, one finds that this procedure is also quite inaccurate after the first few truncations. The reason for this is that a periodic wave vector must have the same value at both boundaries of a block, so that the wave vector of the larger block is broken up into "steps", leading to an inaccurate representation of the low-lying states.

White and Noack [11] formulated two types of RG procedures which solve these problems and work quite well for the single-particle problem. Both are based on choosing a new basis for $\bar{H}_L$ which is *not* the eigenbasis of H_L. In the first procedure, called the *combination of boundary conditions* (CBC) method, the new basis is formed from the low-lying eigenstates of several different block Hamiltonians. The Hamiltonians are formed by applying a number of *different* boundary conditions to the edge of the block. For exam-

ple, fixed and free (for free boundary conditions the derivative of the wave function vanishes at the boundary) boundary conditions can be applied. One can form $H_L^{bb'}$ with $b, b' =$ fixed or free. For example,

$$H_{L=2}^{\text{free,fixed}} = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} . \tag{11}$$

We diagonalize $H_L^{bb'}$ for all 4 combinations of boundary conditions, and then form O_L from the $m/4$ eigenvectors associated with the lowest eigenvalues from each combination of boundary conditions. Since the columns of O_L are not orthogonal, we must explicitly orthogonalize them using the Gram-Schmidt procedure. We then form $\bar{H}_L^{bb'} = O_L^\dagger H_L^{bb'} O_L$ and $\bar{T}_L = O_L^\dagger H_L^{bb'} O_L$. Note that $\bar{H}_L^{bb'}$ is not diagonal. The Hamiltonian of the system of size $2L$ is then

$$H_{2L}^{bb'} = \begin{pmatrix} \bar{H}_L^{b,\text{fixed}} & \bar{T}_L \\ \bar{T}_L^\dagger & \bar{H}_L^{\text{fixed,b}'} \end{pmatrix} . \tag{12}$$

That fixed boundary conditions must be used where the blocks are joined together can be understood by considering a procedure in which all the states are kept at each step and requiring it to be exact. The matrix T_{2L} is formed as in (10).

The fixed-free CBC procedure works amazingly well. As can be seen by comparing the "Exact" and "Fixed-Free" columns of Table 1, the ground-state energy is obtained almost exactly (the error is in the tenth digit) even after 10 iterations ($L = 2048$) when $m = 8$ states are kept. The CBC procedure can also be formulated with other combinations of boundary conditions such as periodic and antiperiodic. While the periodic-antiperiodic CBC procedure is not as accurate as the fixed-free procedure, it performs much better than the Wilson procedure with periodic or antiperiodic boundary conditions.

The second type of procedure developed in Ref. [11], called the *superblock* method, chooses a new basis for $\bar{H}_{2L}$ and $\bar{T}_{2L}$ based on the idea that they will eventually be used to make up part of a larger system. In order to do this, a "superblock" (with periodic boundary conditions) made up of $p > 2$ blocks is formed and diagonalized. For example,

$$H_{2L}^{p=4} = \begin{pmatrix} \bar{H}_L & \bar{T}_L & 0 & \bar{T}_L^\dagger \\ \bar{T}_L^\dagger & \bar{H}_L & \bar{T}_L & 0 \\ 0 & \bar{T}_L^\dagger & \bar{H}_L & \bar{T}_L \\ \bar{T}_L & 0 & \bar{T}_L^\dagger & \bar{H}_L \end{pmatrix} . \tag{13}$$

The transformation O_{2L} is then made up by projecting the m lowest-lying eigenstates of H_{2L}^p onto the coordinates of the first two blocks, and then orthonormaling its columns. In other words, if u_j^α (with $j = 1, \ldots, 4m$) is an eigenvector of H_{2L}^4, then a nonorthonormalized column vector of O_{2L} is composed of the first $2m$ elements, $j = 1, \ldots, 2m$, of u_j^α, assuming $\bar{H}_L$ is an

$m \times m$ matrix. This new basis is used to transform $\bar{H}_{2L} = O_{2L}^{\dagger} H_{2L} O_{2L}$ and $\bar{T}_{2L} = O_{2L}^{\dagger} T_{2L} O_{2L}$, as defined in (9) and (10).

The idea is that the fluctuations in the additional blocks surrounding the portion of the system to be transformed effectively apply general boundary conditions, or equivalently, provide the conditions at the boundaries that the transformed blocks would see as part of a larger system. As p becomes large, this procedure becomes exact because it reduces to an exact diagonalization of the complete final system. Another interesting feature of this procedure is that the diagonalization step is decoupled from the real-space blocking step: a different size system is diagonalized than is blocked together. This procedure yields accurate results for the tight-binding particle eigenstates, although not quite as accurate as the fixed-free CBC procedure.

Of course, we are interested in developing an RG procedure for interacting quantum lattice systems, not the single-particle problem, so the crucial question is whether these procedures can be generalized to work on interacting systems. The CBC method is difficult to generalize because it is difficult to find a general enough set of boundary conditions for interacting systems. To see this, consider the many-body wavefunction for a system of noninteracting fermions, i.e. for the Hubbard model at $U = 0$. An arbitrary many-body state is composed of the Slater determinant of the single-particle wavefunctions of the individual particles, some of which may have nodes and some of which may have antinodes at the boundaries of a block. It is easy to find boundary conditions for which every particle on the block has a node or every particle has an antinode at the boundaries, but is is difficult to find boundary conditions which produce different behavior for different particles. However, a general representation for a block that is part of a larger system must provide a complete range of boundary behavior for each particle individually.

The superblock method seems more promising for application to interacting systems, since the general behavior at the boundaries is provided automatically by embedding the block of interest in a larger superblock. However, the projection of the wavefunction of the superblock onto the system block, which is a simple, single-valued coordinate projection in the noninteracting system, becomes multivalued for the interacting system: one state of the superblock can, in general, project onto many states of the system block. The Density Matrix Renormalization Group is based on choosing an optimal way to do this projection and on combining it with a version of the superblock procedure.

4 The Density Matrix Projection

In this section, we will discuss how to generalize the projection of the superblock described in the previous section for the noninteracting system to interacting systems. The procedure involves forming the reduced density matrix for the system block as part of the superblock. We will show that the

basis obtained using this density matrix projection is the *optimal* basis in a particular sense.

First, let us briefly review the properties of density matrices. An excellent treatment is given in Feynman's book on statistical mechanics [12]. The term "density matrix" is used to refer to a number of different, but related mathematical objects, both in quantum mechanics and quantum statistical mechanics. Here we consider a quantum mechanical system in a definite pure state, and consider the properties of a part of that system. Since we will later use this procedure as part of a superblock algorithm, we will label the entire system the *superblock*, the part that we are interested in constructing a basis for the *system block*, and the remainder of the system the *environment block*, as depicted in Fig. 3. Let $|i\rangle$ label the states of the system block, and $|j\rangle$ label the states of the environment block, i.e. the rest of the superblock. If ψ is a state of the superblock,

$$|\psi\rangle = \sum_{ij} \psi_{ij} |i\rangle |j\rangle \ . \tag{14}$$

The reduced density matrix for the system block is defined as

$$\rho_{ii'} = \sum_{j} \psi^*_{ij} \psi_{i'j} \ . \tag{15}$$

By normalization, $\mathrm{Tr}\rho = 1$. The density matrix contains all the information needed from the wavefunction ψ to calculate any property restricted to the system block. If operator A acts only on the system block, then

$$\langle A \rangle = \sum_{ii'} A_{ii'} \rho_{i'i} = \mathrm{Tr}\ \rho A \ . \tag{16}$$

Now let us diagonalize the density matrix. Let ρ have eigenstates $|u^\alpha\rangle$ and eigenvalues $w_\alpha \geq 0$. Since $\mathrm{Tr}\rho = 1$, $\sum_\alpha w_\alpha = 1$. Then for any system block operator A,

$$\langle A \rangle = \sum_{\alpha} w_\alpha \langle u^\alpha | A | u^\alpha \rangle \ . \tag{17}$$

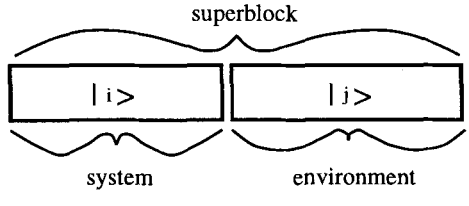

Fig. 3. A superblock divided into a system block and an environment block.

Equation (17) will apply immediately to our numerical renormalization group procedure. Suppose we wish to throw away some states from the system block. If for a particular α, $w_\alpha \approx 0$, we make no error in $\langle A \rangle$, for any A, if we discard $|u^\alpha\rangle$. We have found a way to find which states to keep (those with significant w_α) and which to discard.

This argument can be made much more precise. In particular, we can show that keeping the most probable eigenstates of the density matrix gives the most accurate representation of the state of the superblock, i.e., the system block plus the environment block. Let us assume we have diagonalized the superblock and obtained one particular state $|\psi\rangle$, typically the ground state. We wish to define a procedure for producing a set of states of the system block $|u^\alpha\rangle$, $\alpha = 1, \ldots, m$, with $|u^\alpha\rangle = \sum_i u_i^\alpha |i\rangle$, which are optimal for representing ψ in some sense. Because we allow only m states, we cannot represent $|\psi\rangle$ exactly if $\ell > m$, where ℓ is the number of system block states $|i\rangle$. We wish to construct an accurate expansion for $|\psi\rangle$ of the form

$$|\psi\rangle \approx |\bar{\psi}\rangle = \sum_{\alpha,j} a_{\alpha,j} |u^\alpha\rangle |j\rangle \; . \tag{18}$$

In other words, we wish to minimize

$$\mathcal{S} = \left| |\psi\rangle - |\bar{\psi}\rangle \right|^2 \tag{19}$$

by varying over all $a_{\alpha,j}$ and u^α, subject to $\langle u^\alpha | u^{\alpha'} \rangle = \delta_{\alpha\alpha'}$. Without loss of generality, we can write

$$|\bar{\psi}\rangle = \sum_\alpha a_\alpha |u^\alpha\rangle |v^\alpha\rangle \tag{20}$$

where $v_j^\alpha = \langle j | v^\alpha \rangle = N_\alpha a_{\alpha,j}$, with N_α chosen to set $\sum_j |v_j^\alpha|^2 = 1$. Switching to matrix notation, we have

$$\mathcal{S} = \sum_{ij} (\psi_{ij} - \sum_{\alpha=1}^{m} a_\alpha u_i^\alpha v_j^\alpha)^2 \; , \tag{21}$$

and we minimize $\mathcal{S}$ over all u^α, v^α, and a_α, given the specified value of m. The solution to this minimization problem is known from linear algebra. We now think of ψ_{ij} as a rectangular matrix. The solution is produced by the singular value decomposition [13] of ψ,

$$\psi = U D V^T \; , \tag{22}$$

where U and D are $\ell \times \ell$ matrices, V is an $\ell \times J$ matrix (where $j = 1, \ldots, J$, and we assume $J \geq \ell$), U is orthogonal, V is column-orthogonal, and the diagonal matrix D contains the singular values of ψ. Linear algebra tells us that the u^α, v^α, and a_α which minimize $\mathcal{S}$ are given as follows: the m largest-magnitude diagonal elements of D are the a_α and the corresponding columns of U and V are the u^α and v^α. (We emphasize that the singular

value decomposition is not being used here as a numerical method, only as a convenient factorization which allows us to use a theoretical result from linear algebra.)

These optimal states u^α are also eigenvectors of the reduced density matrix of the block as part of the system. This reduced density matrix for the block depends on the state of the system, which in this case is a pure state $|\psi\rangle$. (The system could also be in a mixed state [see below] or at finite temperature.) The density matrix for the block in this case, where ψ_{ij} is assumed real, is given by

$$\rho_{ii'} = \sum_j \psi_{ij}\psi_{i'j} \; . \tag{23}$$

We see that

$$\rho = UD^2U^T \; , \tag{24}$$

i.e. U diagonalizes ρ. The eigenvalues of ρ are $w_\alpha = a_\alpha^2$ and the optimal states u^α are the eigenstates of ρ with the largest eigenvalues. Each w_α represents the probability of the block being in the state u^α, with $\sum_\alpha w_\alpha = 1$. The deviation of $P_m \equiv \sum_{\alpha=1}^m w_\alpha$ from unity, i.e. the "discarded weight" of the density matrix eigenvalues, measures the accuracy of the truncation to m states.

To summarize, in the previous two paragraphs we have shown that when the superblock is assumed to be in a pure state, the optimal states to keep are the m most significant eigenstates of the reduced density matrix of the system block, obtained from the wavefunction of the superblock via (23).

We can also consider the superblock to be in a mixed state. This is the natural assumption for a system at finite temperature, and it is also useful to assume a mixed state when one wishes to obtain several of the lowest lying states: if we put the superblock with equal probability into each of several states, then the system block states obtained from the density matrix will equally well represent each of these superblock states. We represent the mixed case by saying that the superblock has probability W_k to be in state $|\psi^k\rangle$. If the superblock is at a finite temperature, then the W_k are normalized Boltzmann weights. In this case the appropriate definition for the error in the representation is

$$\mathcal{S} = \sum_k W_k \sum_{ij} (\psi_{ij}^k - \sum_{\alpha=1}^m a_\alpha^k u_i^\alpha v_j^{k,\alpha})^2 \; . \tag{25}$$

Note that we are interested in determining a single set of optimal u^α, whereas we allow the rest of the system additional freedom to choose a different v^α for each state k. Minimizing over the u^α, $v^{k,\alpha}$, and a_α^k, we find

$$\rho u^\alpha = w_\alpha u^\alpha \tag{26}$$

with

$$\rho_{ii'} = \sum_k W_k \sum_j \psi^k_{ij} \psi^k_{i'j} \tag{27}$$

and

$$w_\alpha = \sum_k W_k (a^k_\alpha)^2 \,. \tag{28}$$

This equation for ρ is the definition of the reduced density matrix when the superblock is in a mixed state, and the u^α are the eigenstates of ρ.

Thus the conclusion when the superblock is in a mixed state is identical to the result for a pure state: the optimal states to keep are the eigenvectors of the reduced density matrix with the largest eigenvalues.

The effectiveness of the truncation of the Hilbert space of the system block via the density matrix depends crucially on the distribution of the density matrix eigenvalues w_α. For certain exactly solvable systems, it is possible to determine the density matrix eigenvalues exactly. For these integrable systems, the distribution of the density matrix eigenvalues can be shown to decay exponentially, see Chap. 3.1(II).

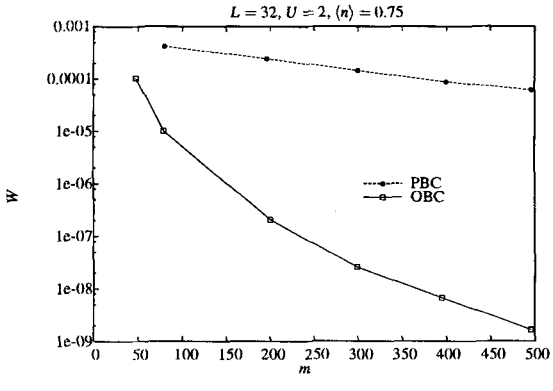

Fig. 4. Sum of the discarded weight of the density matrix eigenvalues for the finite system DMRG algorithm, as a function of the number of states kept m.

As a realistic example of the distribution of the density matrix eigenvalues, we give numerical results for the one-dimensional Hubbard model obtained from the finite size DMRG algorithm, which will be discussed in the next section. In Fig. 4, we plot the sum of the discarded density matrix eigenvalues, $\sum_{\alpha=m+1}^{m_{\max}} w_\alpha$, where $m_{\max}$ is the size of the density matrix. This discarded weight is plotted in Fig. 4 as a function of m on a logarithmic scale for the 32-site Hubbard chain with open and periodic boundary conditions at $n = 0.75$. As can be seen, the discarded weight falls off rapidly with the number of states kept. For open boundary conditions, the decay is clearly slower than exponential for small m, in contrast to the original results found

for the spin-one Heisenberg chain [1]. This could be because the system is in the Luttinger liquid regime and therefore has no gap in either spin or charge excitations. It is possible that the decay becomes exponential at large enough m, when the error in the energy becomes less than the finite energy level spacing always found on a finite system. This discarded weight is strongly correlated with the error in the ground state energy and is often used as a measure of the error. It has been used to to make an $m \to \infty$ extrapolation of the energy [14]. Also depicted in Fig. 4 is the discarded weight for the same system with periodic boundary conditions. The convergence of the discarded weight (and thus the ground state energy) with m is much slower than for open boundary conditions. It is generally found that the accuracy of the energy for a given m is many orders of magnitude worse for periodic than for open boundary conditions. It is therefore usually better to treat systems with open boundary conditions on much larger lattices rather than systems with periodic boundary conditions.

5 DMRG Algorithms

In this section, we will describe how to combine the superblock procedure with the density matrix projection in order to define efficient DMRG algorithms. There are three main ingredients needed to form a DMRG algorithm: first, we have to decide how to add degrees of freedom to the system, i.e. how to build up the system block; second, we have to determine the configuration of the superblock; and finally, we must choose which superblock eigenstate or eigenstates to use to construct the density matrix.

For interacting systems, it is clear that one wants to add the minimum number of degrees of freedom at once to the system block in order keep as large a fraction of the system block states as possible, and to keep the size of the Hilbert space of the superblock as small as possible. Therefore, one usually wants to build up the system block one site at a time in a procedure similar to that described for the Wilson numerical RG in Sect. 2.

The algorithms then fall into two classes, depending on how the environment block is chosen to form the superblock: the infinite system algorithm and the finite system algorithm. We will discuss these algorithms in detail below.

We will call the superblock state or states used to form the reduced density matrix for the system block *target states*. If only ground state properties are desired, it is most accurate to target just the ground state of the superblock. (The Hamiltonian is usually block diagonal in particular quantum numbers such as S_z; by ground state we will mean ground state for a particular quantum number.) If excited states or matrix elements between different states are required, more than one target state can be used. However, for fixed number of states kept m, the accuracy with which the properties of each individual state can be determined goes down as more states are tar-

geted. For simplicity, we will assume that only the ground state is targeted in the following.

The infinite system algorithm

The infinite system algorithm is the most straightforward extension of the Wilson procedure described in Sect. 2 that incorporates the superblock concept. We build up the system block one site at a time, just as in the Wilson procedure, but must choose some sort of environment block. The simplest way of forming the environment block is to use a reflection of the system block. The superblock configuration is shown in Fig. 5. Here $\bar{H}_\ell$ is the Hamiltonian for the system block in the reduced basis, as before, and the solid dots represent single sites. The right block, $\bar{H}_\ell^R$, is formed by relabeling the sites in the system block so that they are reflected onto the right part of the lattice.

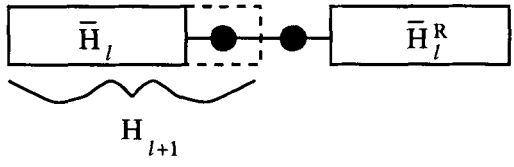

Fig. 5. The superblock configuration for the infinite-system algorithm.

The infinite system algorithm then proceeds as follows:

1. Form a superblock containing L sites which is small enough to be exactly diagonalized.
2. Diagonalize the superblock Hamiltonian H_L^{super} numerically, obtaining *only* the ground state eigenvalue and eigenvector $|\psi\rangle$ using the Lanczos or Davidson algorithm.
3. Form the reduced density matrix $\rho_{ii'}$ for the new system block from $|\psi\rangle$ using (15). Note that $\ell' = \ell = L/2 - 1$.
4. Diagonalize $\rho_{ii'}$ with a dense matrix diagonalization routine to obtain the m eigenvectors with the largest eigenvalues.
5. Construct $H_{\ell+1}$ and other operators in the new system block and transform them to the reduced density matrix eigenbasis using $\bar{H}_{\ell+1} = O_L^\dagger H_{\ell+1} O_L$, $\bar{A}_{\ell+1} = O_L^\dagger A_{\ell+1} O_L$, etc., where the columns of O_L contain the m highest eigenvectors of $\rho_{ii'}$, and $A_{\ell+1}$ is an operator in the system block.
6. Form a superblock of size $L+2$ using $\bar{H}_{\ell+1}$, two single sites and $\bar{H}_{\ell+1}^R$.
7. Repeat starting with step 2, substituting H_{L+2}^{super} for H_L^{super}.

It is clear that this algorithm is very much in the spirit of the original Wilson procedure in that the system being diagonalized grows at each step. There

are, however, a few important differences. First, as in the noninteracting superblock procedure, the diagonalization step and the real-space blocking step take place on different size systems. Therefore, the energy and various expectation values are calculated during the superblock diagonalization, while the density matrix diagonalization rather than a Hamiltonian diagonalization is used to determine the new basis for the system block. Second, the size of the superblock grows by two sites rather than one site at every step. Third, we have assumed that the system is reflection symmetric. It is possible to formulate algorithms that do not assume reflection symmetry, but this is done most easily in the context of the finite system algorithm described below.

The finite system algorithm

In the finite size algorithm, the environment block is chosen in a different way: it is chosen so that the size of the superblock is kept fixed at each step. Suppose that we have run the infinite system algorithm until the superblock reaches size L, but have stored all the $\bar{H}^R_{\ell'}$ for $\ell = 1, \ldots, L/2-2$ as well as the all the additional operators needed to connect the blocks at each step. We can then continue to build up the system block, but keep $L = \ell + \ell' + 2$ fixed by using the appropriate previously stored $\bar{H}^R_{\ell'}$. The finite size algorithm then proceeds as follows:

0. Carry out the infinite system algorithm until the superblock reaches size L, storing $\bar{H}_\ell$ and the operators needed to connect the blocks at each step.
1. Carry out steps 3-5 of the infinite system algorithm to obtain $\bar{H}_{\ell+1}$. Store it. (Now $\ell \neq \ell'$.)
2. Form a superblock of size L using $\bar{H}_{\ell+1}$, two single sites and $\bar{H}^R_{\ell'-1}$. The superblock configuration is given by Fig. 6, where $\ell' = L - \ell - 2$.
3. Repeat steps 1-2 until $\ell = L - 3$ (i.e. $\ell' = 1$). This is the *left to right* phase of the algorithm.
4. Carry out steps 3-5 of the infinite system algorithm, reversing the roles of $\bar{H}_\ell$ and $\bar{H}^R_{\ell'}$, i.e. switch directions to build up the right block and obtain $\bar{H}^R_{\ell'+1}$. Store it.
5. Form a superblock of size L using $\bar{H}_{\ell-1}$, two single sites and $\bar{H}^R_{\ell'+1}$.
6. Repeat steps 4-5 until $\ell = 1$. This is the *right to left* phase of the algorithm.
7. Repeat starting with step 1.

A useful analogy is to think of this procedure as being like running a zipper repeatedly from left to right and then right to left through a superblock that is always the same size. Each time the zipper changes direction, a new set of stored blocks is used as the environment block. In this way, the representations of the stored blocks are iteratively improved and the zipping can be repeated until convergence is reached.

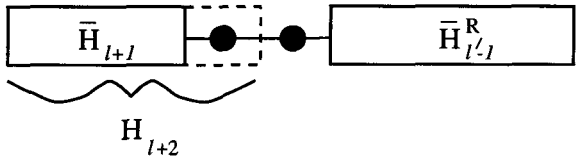

Fig. 6. A typical superblock configuration for the left-to-right phase of the finite-system algorithm.

Note that since the left block and the right blocks are stored independently, we do not have to assume that the lattice is reflection symmetric (at least after step 0). Since the same size superblock is always diagonalized, the algorithm is less dependent than the infinite system algorithm on translational invariance, i.e. on the optimum representation of different size superblocks being similar.

If reflection symmetry is present, it can be used at the point at which $\ell = \ell'$ to shorten the length of the zips. One way of formulating the algorithm in this case is to build up the left blocks from $\ell = 1$ to $\ell = L/2 - 1$, and build up the right blocks from $\ell' = L/2$ to $\ell' = L - 3$, i.e. to only zip from the left side of the superblock to the middle and then back to the left side. The fact that we have used reflection symmetry in the infinite system phase, step 0, is usually not important. However, it is also possible to formulate infinite system algorithms that do not use reflection symmetry. This issue will be discussed in more detail below in the context of algorithms for two-dimensional and fermion systems.

For a given system size L, the finite system algorithm almost always gives substantially more accurate results than the infinite system algorithm, and is therefore usually preferred unless there is a specific reason to go to the thermodynamic limit.

It is instructive to examine how the infinite and finite system algorithms work in detail on a particular system. Here we will utilize the one-dimensional Hubbard model, (2), as an example. This model is fairly complicated, containing both spin and fermionic degrees of freedom, but has a Bethe Ansatz exact solution [15], which has been extended to the case of open boundary conditions [16]. Here we will investigate how the energy obtained from the finite-system DMRG algorithm converges to the exact Bethe Ansatz energy on a particular system. This is done for a fractionally filled 32-site system with open boundary conditions in Fig. 7. The ordinate "Iteration" refers to the iteration of reflection symmetric finite system algorithm, for which each iteration consists of a right-to-left "zip" from center (the reflection symmetric configuration) to the left edge, followed by a left-to-right "zip" back to the center.

As can be seen from the curves which were made for different runs with different numbers of states m kept in the system block, there is virtually no

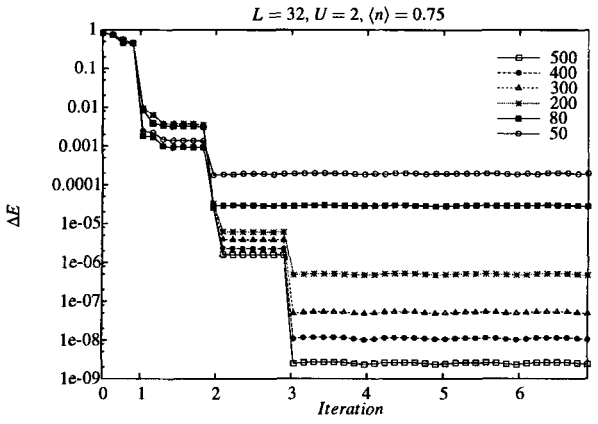

Fig. 7. The difference between the ground-state energy obtained using the finite system DMRG and the exact Bethe Ansatz energy for a 32-site system at $U = 2$ and filling $\langle n \rangle = 0.75$, plotted on a logarithmic scale as a function of iteration of the finite system algorithm. Each point represents a Davidson diagonalization of the superblock (every fourth diagonalization is plotted), and the different symbols represent different runs made with a fixed number $m = 50, \ldots, 500$ of states kept.

dependence of the accuracy of the energy on m in the zeroth iteration, the infinite system phase. This behavior is typical for fractionally filled fermion systems and for finite-width systems. It is due to the poor job the infinite system algorithm does at representing systems with an intrinsic length, such as the wavelength of local density oscillations or the width of a two-dimensional system. After two to three iterations, the error in the energy saturates for a given m, with the saturation occurring after a larger number of iterations for larger m. Therefore, it is only helpful to increase m once sufficient convergence in the number of iterations has been achieved. The optimum strategy is then to increase m fairly rapidly at each iteration. If this is done, the majority of time is spent doing the last iteration, and the finite system algorithm only takes a factor of two to three longer than the infinite system algorithm for the same number of states, but can be orders of magnitude more accurate. Note that the rate of convergence with the number of iterations can vary strongly from model to model, so the optimum strategy will also vary.

Once the optimum convergence is achieved, the crucial question is how the accuracy of the energy depends on the maximum number of states kept in a run, i.e. on how the height of the plateaus in Fig. 7 depends on m. This dependence is illustrated in Fig. 8 for different U values. The form of the decay in the error in the ground state energy with m is very similar to the form of decay of the discarded weight of the density matrix eigenvalues, Fig. 4. The error in the energy is usually proportional to the discarded weight, once the algorithm has converged sufficiently. Another interesting point is that the rate

of convergence is almost completely independent of the interaction strength U. In fact, we have found that even adding a nearest-neighbor Coulomb interaction does not substantially change the convergence with m [17], as long as the system is the Luttinger liquid regime, i.e. has no spin or charge gap. The nature and density of the low-lying excitations in the superblock, which does not change much in a Luttinger liquid, seems to be more important than the nature and range of the interactions.

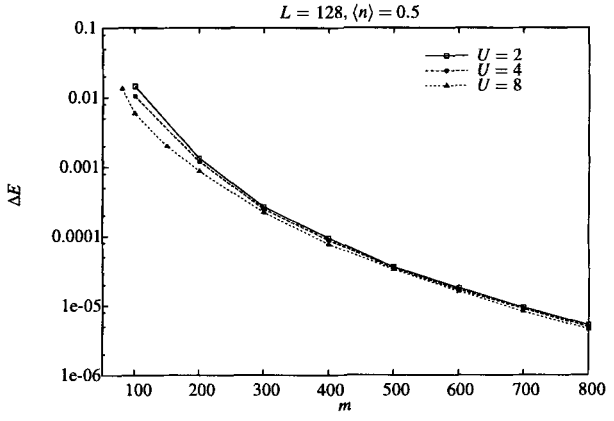

Fig. 8. Convergence of the finite system algorithm as a function of the number of states kept m for different U.

Details for interacting systems

Up to now, we have not considered in detail how to store and transform the operators necessary to carry out the renormalization group transformation for an interacting system. In this section, we will discuss how to do this efficiently.

In order to construct the Hamiltonian of the system, a block must have various operators stored as matrices connecting these states. For example, for the Heisenberg model with exchange terms

$$\boldsymbol{S}_i \cdot \boldsymbol{S}_{i+1} = S_i^z S_{i+1}^z + \frac{1}{2}(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+) \tag{29}$$

one needs to store $m \times m$ matrix representations of S_i^z, S_i^+, and S_i^- for i equal to the left or right end sites of the block. (In practice, one need not store S_i^-, since it can be obtained by taking the Hermitian conjugate of S_i^+). For a Hubbard model one would have to store matrices for $c_{i\sigma}$, with $\sigma = \uparrow$ and $\downarrow$, in order to reconstruct the hopping term $\sum_\sigma (c_{i+1\sigma}^\dagger c_{i\sigma} + c_{i\sigma}^\dagger c_{i+1\sigma})$.

Consider joining two blocks B_1 and B_2 together in a Heisenberg system. In the Wilson procedure, B_2 will typically consist of a single site, and for the DMRG algorithms, it is sufficient to consider how to compose two blocks. If B_1 has m_1 states, and B_2 has m_2 states, the combined block B_1B_2 has m_1m_2 states. We label the combined states by two indices, ij. The matrix representing the Hamiltonian of B_1B_2 is then given by

$$\begin{aligned}[H_{B_1B_2}]_{ij;i'j'} = [H_{B_1}]_{ii'}\,\delta_{jj'} + [H_{B_2}]_{jj'}\,\delta_{ii'} + [S^z_\ell]_{ii'}\,[S^z_{\ell+1}]_{jj'} \\ + \tfrac{1}{2}\,[S^+_\ell]_{ii'}\,[S^-_{\ell+1}]_{jj'} + \tfrac{1}{2}\,[S^-_\ell]_{ii'}\,[S^+_{\ell+1}]_{jj'}\end{aligned} \tag{30}$$

where H_{B_1} is the Hamiltonian matrix of block B_1, and ℓ is its rightmost site. In order for the connections in the Hamiltonian to be restored when the two blocks are combined, each block must contain each of the matrices appearing in (30).

The most time-consuming part of a DMRG calculation is the diagonalization of the system Hamiltonian, which occurs once for every step. Only the ground state or a few low-lying states are needed, and so the Lanczos or Davidson iterative algorithms should be used. These algorithms both require repeated multiplications of superblock vectors ψ by the superblock Hamiltonian H^{super}. However, the actual Hamiltonian matrix should not be constructed and stored. Instead, the following procedure to directly multiply $H^{\text{super}}\psi$ uses much less memory and is also much faster. Consider again, for simplicity, a system formed from two blocks. The superblock Hamiltonian can be written in the general form

$$[H^{\text{super}}]_{ij;i'j'} = \sum_\alpha A^\alpha_{ii'} B^\alpha_{jj'}\ . \tag{31}$$

Then the product $H^{\text{super}}\psi$ can be written as

$$\sum_{i'j'} [H^{\text{super}}]_{ij;i'j'}\,\psi^\alpha_{i'j'} = \sum_\alpha \sum_{i'} A^\alpha_{ii'} \sum_{j'} B^\alpha_{jj'}\psi_{i'j'}\ . \tag{32}$$

For each α, the last sum is performed first, as a matrix-matrix multiplication of B^α and ψ^T, to form a temporary matrix $C^\alpha_{ji'}$. Then a matrix-matrix multiplication of A^α and $[C^\alpha]^T$ forms a partial result, which is added into the result vector, giving a sum on α.

Whenever a site is added onto a block, or more generally two blocks are added, the operator matrices must be updated. The eigenstates of the density matrix can be written in the form u^α_{ij}, which we write as $O_{ij;\alpha}$, $\alpha = 1, \ldots, m$. Here i and j represent state indices of the two blocks that are being added together. Then for each operator A that is needed, $A_{ij;i'j'}$ is replaced by $A_{\alpha\alpha'}$, where

$$A_{\alpha\alpha'} = \sum_{iji'j'} O_{ij;\alpha} A_{ij;i'j'} O_{i'j';\alpha'}\ . \tag{33}$$

The terms appearing in (30) show the various ways operators $A_{ij;i'j'}$ can be formed from single-block operators $A_{ii'}$.

Any efficient DMRG program must make use of quantum numbers to speed up the calculation and reduce storage. For example, in order to construct the system Hamiltonian it may be necessary to store for a block the matrix form of the operator $\hat{S}^+_\ell$, where ℓ is the right-most site of the block. If there are m states in the block, this is an $m \times m$ matrix. However, if states are labeled and grouped by block quantum number S^z, then this matrix is mostly zeroes, with the nonzero parts in rectangular blocks. These blocks connect states with specific quantum numbers, e.g. the states corresponding to the left index of the matrix may have $S_z = 0$, and for the right index $S_z = -1$. It is essential to store only the nonzero elements of this matrix. Although this can done using sparse matrices, the best way to do it is as a set of dense matrices, one for each nonzero rectangular block. The multiplication of $H^{\rm super}\psi$ described above takes place as described in the previous paragraph, except that now there is an additional sum or loop over quantum numbers, and the dense matrices which are multiplied are much smaller. Keeping track of all the matrices, each of a different size, can be very well organized in C++ by defining classes to represent operator matrices, submatrices, etc. The fact that these matrices are all different sizes and have different dimensions at each DMRG step makes it somewhat more difficult to use storage efficiently in Fortran 77, which does not have dynamic allocation of memory.

It is useful at this point to mention typical maximum numbers of states kept, m, for various systems on current computers. For the one-dimensional Hubbard model, $m = 800$ for a system of up to a few hundred sites can be treated on typical workstation, using a few hundred megabytes of main memory [17,18]. For the Heisenberg model, up to $m = 1100$ states have been kept for the spin-one chain with an impurity [19] and $m = 1700$ for the spin-two chain [20]. These Heisenberg-chain calculations have been carried out using a highly optimized Fortran code.

Measurements

Measurements are made using the superblock wavefunction $|\psi\rangle$ to evaluate expectation values of the form $\langle\psi|A|\psi\rangle$. Rather complicated operators can be evaluated fairly easily, but dynamical information is more difficult to obtain. In order to measure A, one must have kept operator matrices for the components of A. For example, to measure the on-site spin-density S^z_j for all sites ℓ, one must keep track of matrices $[S^z_\ell]_{ii'}$, for all each site ℓ in each of the blocks. These operators must be updated using (33) at every step of each iteration. We will once again divide superblock into two parts with states labeled by $|i\rangle$ and $|j\rangle$. One then obtains the expectation value using

$$\langle\psi|S^z_\ell|\psi\rangle = \sum_{i,i',j} \psi^*_{ij}\,[S^z_\ell]_{ii'}\,\psi_{i'j}\ , \tag{34}$$

etc. This procedure gives *exact* evaluations of $\langle\psi|A|\psi\rangle$ for the *approximate* eigenstate $|\psi\rangle$.

For a correlation function such as $\langle\psi|S^z_\ell S^z_m|\psi\rangle$, the evaluation depends on whether ℓ and m are on the same block or not. If they are on different blocks, then one need only have kept track of $[S^z_\ell]_{ii'}$ and $[S^z_m]_{jj'}$, and one has

$$\langle\psi|S^z_\ell S^z_m|\psi\rangle = \sum_{i,i',j,j'} \psi^*_{ij}\,[S^z_\ell]_{ii'}[S^z_m]_{jj'}\,\psi_{i'j'}\ . \tag{35}$$

If ℓ and m are on the same block, one *should not* use

$$\langle\psi|S^z_\ell S^z_m|\psi\rangle \approx \sum_{i,i',i'',j} \psi^*_{ij}\,[S^z_\ell]_{ii'}[S^z_m]_{i'i''}\,\psi_{i''j}\ . \tag{36}$$

This expression does not evaluate the correlation function exactly within the approximate state $|\psi\rangle$. The sum over i' should run over a complete set of states, but does not, whereas the sums over the other variables need run only over those states needed to represent $|\psi\rangle$, since they appear as a subscript in either the $|\psi\rangle$ on the left or on the right.

To evaluate this type of correlation function, one needs to have kept track of $[S^z_j S^z_k]_{ii'}$ throughout the calculation. One then evaluates

$$\langle\psi|S^z_\ell S^z_m|\psi\rangle = \sum_{i,i',j} \psi^*_{ij}\,[S^z_\ell S^z_m]_{ii'}\,\psi_{i'j} \tag{37}$$

to obtain the correlation function.

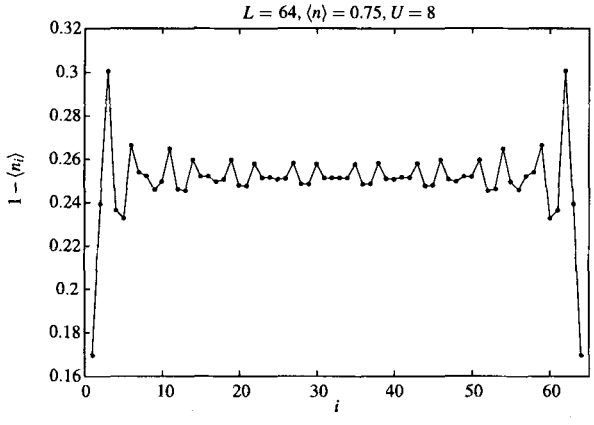

Fig. 9. Local hole density $1 - \langle n_i\rangle$ plotted as a function of i.

As an example of measurements of the first type, (34), we display the local hole density $1 - \langle n_i\rangle$, where $n_i \equiv n_{i\uparrow} + n_{i\downarrow}$ for the one-dimensional

Hubbard model in Fig. 9. The oscillations are Friedel oscillations due to the open boundary conditions.

In Fig. 10, we show the nearest-neighbor spin-spin correlations for the half-filled system as an example of a correlation function between different sites, calculated using (37). This correlation function also has oscillations due to the open boundaries. Since the correlation functions cannot be calculated directly from the Bethe Ansatz, is is not possible to directly calculate the error, as for the ground state energy. The issue of how to form a good approximation to correlation functions in the thermodynamic limit using these finite-lattice correlation functions, as well as a comparison with the asymptotic results obtain from a combination of bosonization and the Bethe Ansatz [21] is discussed in more detail in Chap. 1.1(II).

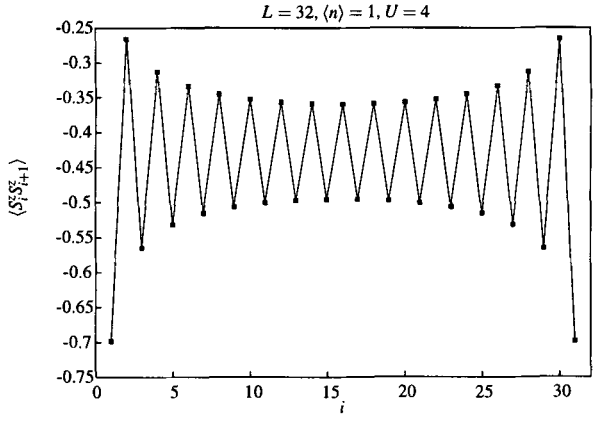

Fig. 10. Nearest neighbor spin-spin correlation function $\langle S_i^z S_{i+1}^z \rangle$ plotted as a function of i for the half-filled system.

Wave function transformations

An important improvement in the efficiency of DMRG comes from keeping track of the wavefunction from step to step [22]. The step referred to here is the process of adding a site to a block and requires the diagonalization of a system configuration of two blocks and two sites [1]. In each DMRG step, an iterative sparse matrix algorithm, such as the Davidson method, is used to find the ground state of the system. Thus far, we have not specified a starting point for the Davidson procedure. To ensure that the DMRG procedure is always stable and convergent, the system ground state usually has to be determined to rather high accuracy. (One diagonalization which converges to a low-lying eigenstate other than the ground state can ruin the accuracy of

the entire DMRG sweep.) Consequently, a substantial number of Davidson steps are necessary to converge to sufficient accuracy, typically 40-100. The total calculation time is proportional to the average number of Davidson steps.

If a very good initial guess is available for the Davidson procedure, the number of Davidson steps can be reduced substantially. An ideal initial guess, for the case of the finite system DMRG algorithm, is the final wavefunction from the previous DMRG step. This wavefunction, however, is in a different basis, corresponding to a different system configuration, but it can be transformed into the basis corresponding to the current configuration, as described below. Use of this transformation to obtain the initial state in a Davidson diagonalization can reduce the number of Davidson steps by one half, typically, assuming that one iterates Davidson until it converges to high accuracy. Use of this initial guess has an even more important advantage: it is not necessary to converge to high accuracy, since there is no danger of converging to an incorrect low-lying eigenstate. The initial guess not only has low energy, it approximately describes the correct eigenstate, as obtained in the previous step. Hence a lower accuracy convergence is possible without damaging the stability of DMRG. In fact, the algorithm can be made completely stable even if the number of Davidson steps is restricted to two or three! Thus one saves a factor of 20-50 in the time required by the Davidson procedure. The overall speedup is somewhat reduced from this factor because the calculation time to perform other parts of the DMRG procedure, such as diagonalizing the density matrix, becomes significant.

A DMRG step adds a site onto a block, constructing an appropriate basis for the new block. The new basis is defined by the eigenvectors of the density matrix, which is determined from the current system wavefunction. The eigenvectors defining the new basis (which we rewrite as matrices L and R below) can then be used to transform the current system wavefunction into the appropriate basis for the *next* step. Here we consider a finite-system DMRG step, in which we move from left to right, adding a site onto the left block, in the standard configuration with two sites between the left and right blocks. Let $|\alpha_\ell\rangle$ be the states of left block ℓ, where ℓ is the rightmost site of the block. The two sites in the middle are $\ell+1$ and $\ell+2$. Let $|s_\ell\rangle$ be the states of site ℓ, $|s_{\ell+1}\rangle$ for site $\ell+1$, etc. Then the basis states for the new left block are given by

$$|\alpha_{\ell+1}\rangle = \sum_{s_{\ell+1},\alpha_\ell} L^{\ell+1}[s_{\ell+1}]_{\alpha_{\ell+1},\alpha_\ell} |\alpha_\ell\rangle \otimes |s_{\ell+1}\rangle \ . \tag{38}$$

This notation is similar to that of Östlund and Rommer [23]. For more details, see Chap. 3(I). The transformation matrix $L^{\ell+1}[s_{\ell+1}]_{\alpha_{\ell+1},\alpha_\ell}$ is a slightly rewritten form of the truncated matrix of density matrix eigenvectors $u^{\alpha_{\ell+1}}$: specifically, $L^{\ell+1}[s_{\ell+1}]_{\alpha_{\ell+1},\alpha_\ell} = u^{\alpha_{\ell+1}}_{s_{\ell+1}\alpha_\ell}$. L includes only the eigenvectors which are retained, i.e. whose corresponding eigenvalues are above a cutoff. Let the states of the right block be $|\beta_{\ell+3}\rangle$, where we note that $\ell+3$ is the

leftmost site of the block. These states were formed at an earlier right-to-left DMRG step, using a different set of density matrix eigenvectors, which we write in terms of a transformation matrix $R^{\ell+3}$:

$$|\beta_{\ell+3}\rangle = \sum_{s_{\ell+3},\beta_{\ell+4}} R^{\ell+3}[s_{\ell+3}]_{\beta_{\ell+3},\beta_{\ell+4}} |s_{\ell+3}\rangle \otimes |\beta_{\ell+4}\rangle \ . \tag{39}$$

Note that reflection symmetry is not assumed here: the L and R matrices are independent.

The wavefunction is written in a basis for the two block plus two site superblock. This superblock basis has basis states of the form

$$|\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}\rangle = |\alpha_\ell\rangle \otimes |s_{\ell+1}\rangle \otimes |s_{\ell+2}\rangle \otimes |\beta_{\ell+3}\rangle \ . \tag{40}$$

A system wavefunction $|\psi\rangle$ is written in this basis as

$$|\psi\rangle = \sum_{\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}} \psi(\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}) |\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}\rangle \ . \tag{41}$$

One needs to transform this wavefunction into the basis appropriate for the next DMRG step, in which the superblock is shifted by one site, with basis states of the form $|\alpha_{\ell+1} s_{\ell+2} s_{\ell+3} \beta_{\ell+4}\rangle$. However, the transformation between the two bases cannot be exact, since there is a truncation in going from $|\alpha_\ell s_{\ell+1}\rangle$ to $|\alpha_{\ell+1}\rangle$. However, the states $|\alpha_{\ell+1}\rangle$ are formed using the density matrix to be ideally adapted for representing $|\psi\rangle$. This means that the wavefunction can be transformed in an approximate but controlled fashion, with the error in the transformation depending on the truncation error in the DMRG step. Since the error in the density matrix is given by the truncation error, and since the density matrix is, roughly speaking, the square of the wavefunction, the error in the wavefunction transformation should be roughly the square root of the truncation error.

The simplest way to derive the transformation is to assume, based on the above argument, that for the transformation of the wavefunction only, one can approximate

$$\sum_{\alpha_{\ell+1}} |\alpha_{\ell+1}\rangle\langle\alpha_{\ell+1}| \approx 1 \ . \tag{42}$$

With this approximation one readily obtains

$$\psi(\alpha_{\ell+1} s_{\ell+2} s_{\ell+3} \beta_{\ell+4}) \approx \sum_{\alpha_\ell s_{\ell+1} \beta_{\ell+3}} L^{\ell+1}[s_{\ell+1}]_{\alpha_{\ell+1},\alpha_\ell} \psi(\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}) R^{\ell+3}[s_{\ell+3}]_{\beta_{\ell+3},\beta_{\ell+4}} \ . \tag{43}$$

This is the desired transformation.

The most efficient way to implement this transformation numerically is to first form the intermediate wavefunction

$$\psi(\alpha_{\ell+1} s_{\ell+2} \beta_{\ell+3}) = \sum_{\alpha_\ell s_{\ell+1}} L^{\ell+1}[s_{\ell+1}]_{\alpha_{\ell+1},\alpha_\ell} \psi(\alpha_\ell s_{\ell+1} s_{\ell+2} \beta_{\ell+3}) \ , \tag{44}$$

and then form the final result

$$\psi(\alpha_{\ell+1}s_{\ell+2}s_{\ell+3}\beta_{\ell+4}) = \sum_{\beta_{\ell+3}} \psi(\alpha_{\ell+1}s_{\ell+2}\beta_{\ell+3})R^{\ell+3}[s_{\ell+3}]_{\beta_{\ell+3},\beta_{\ell+4}} \,. \quad (45)$$

In this form, the transformation requires very little computer time compared to other parts of the calculation.

This transformation is used for one half of the DMRG steps, when a site is being added to the left block. An analogous transformation is used for adding a site to the right block.

Implementing this transformation requires saving all the matrices L and R, which was not necessary in the original formulation of DMRG. The storage for these matrices is typically 20-30% of the storage required for the blocks themselves, so the extra storage is not a major concern. In an efficient DMRG implementation for a typical single processor workstation, both the blocks and the transformation matrices should be stored on disk.

Extension to two dimensions

The issue of how to optimally extend the DMRG to two or more dimensional quantum systems is difficult and is still a subject of active development. It is instructive to consider first how to extend the one-dimensional algorithm to quasi-one-dimensional systems with a finite width, e.g. to the Heisenberg or Hubbard model on a "ladder". One straightforward way to do this would be to replace the single sites added between the blocks with a row of sites. However, for wide systems the extra degrees of freedom in the two center "sites" would make the size of the system's Hilbert space prohibitively large. It is usually better to add single sites by mapping the 2D system onto a 1D system, simply by tracing a path through the lattice. A typical superblock configuration for a ladder system is shown in Fig. 11. The site added to the system block is enclosed by a dashed line and the dotted line shows the order in which sites are added in a sweep. An up-down-up-down path is shown; an up-up-up path would be just as efficient. One can see that it is not possible to reflect the left block onto a right block of the proper geometry at every step, so the finite system algorithm must used. A similar difficulty appears in fermion systems at fractional filling because it is not possible to set the number of particles so that the average density stays constant at all system sizes. The denominator of the fraction of the filling is analogous to the row size for two-dimensional lattices. With the mapping onto a 1D system, the two-dimensional procedure differs from the one-dimensional finite system procedure only in that there are additional connections between the system and environment blocks along the boundary.

Several infinite system approaches can be used in the warmup sweep [24]. The simplest is just to use several sites for the environment block, without truncation. Usually the accuracy of the warmup sweep is not critical; a few

sweeps of the finite-system procedure can make up for a poor quality warmup sweep.

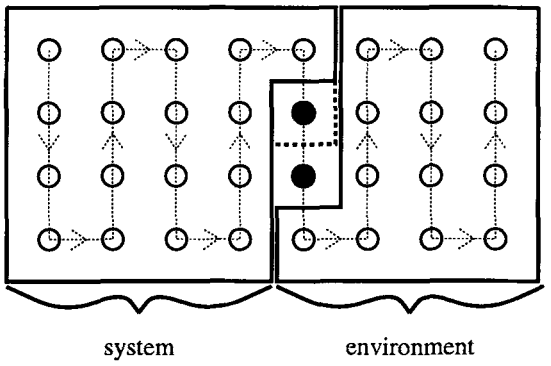

Fig. 11. The superblock configuration for the two-dimensional algorithm. The order in which sites are added to the system block on a series of iterations is given by the dotted line, and the site added to the approximate system block Hamiltonian is outlined by the dashed line.

The number of states needed to maintain a certain truncation error in the density matrix projection procedure depends strongly on the number of operators connecting the two parts of the system. Best accuracy is obtained when the number of connections between the system and environment blocks is minimized. Therefore, we usually study systems with open rather than periodic or antiperiodic boundary conditions. Also, we find that the number of states m needed to maintain a given accuracy depends strongly on the width and weakly on the length of the system.

Just how rapidly the truncation error increases with the width of the system is not clear in general. Liang and Pang [25] studied the error in the energy as a function of width for a gas of noninteracting spinless fermions and found that the number of states needed to maintain a given accuracy grew exponentially with the width of the system. In an interacting system such as the Hubbard model, the detailed structure of the energy spectrum seems to be important. For example, in the two chain Hubbard model at half-filling, where there is a spin and pairing gap, the truncation error for a given m is much smaller than away from half-filling, where the spin gap is reduced and the gap to pairing excitations is no longer present. For Heisenberg ladders, the presence or absence of a gap in the spin spectrum depends on whether the number of chains is even or odd, so the truncation error for a given m depends on the number of chains in a complicated way. In any case, for systems of more than one dimension, it is important to be able to keep as many states m per block as possible. Efficient algorithms are therefore crucial in this case.

6 Remarks

In this chapter, we have described the Density Matrix Renormalization Group as originally developed to determine the properties of low-lying states of interacting quantum systems. We have followed the historical development of the method in order to motivate the ingredients that go into the algorithms. We have also tried to strike a balance between a pedagogical explanation of the basic ideas and the inclusion of sufficient detail to allow the reader to produce efficient implementations. In order to provide a good starting point for understanding the basic ideas and limitations of various numerical RG schemes, the simplest problem, the tight-binding chain, was presented in considerable detail. As the best way to learn the technique is to implement a scheme on the computer and to play with it, we have included, in Appendix B, a program which treats this problem as an example.

The DMRG is still under quite active development. Since the basic ideas seem to be quite general and the algorithms well-suited for experimenting, there is hope that significant improvements will be made to both performance and applicability in the future. As we have described, the most serious limitation of the original ground-state algorithm is the strong dependence of its convergence on the degree of two-dimensionality and on the complexity and range of the interaction. Up to now, progress has been made through improved efficiency of the implementations and better performance of the computers. A variation of the algorithm that scales more favorably for higher dimensional systems would, however, be quite useful. The momentum-space formulation discussed in Chap. 6(I) could be a step in this direction.

There has also been much recent activity on extending the DMRG and in applying it to new problems. In particular, there have been interesting developments in extending the DMRG to calculate dynamic and finite-temperature properties of interacting quantum systems, in applying it to two-dimensional classical systems, and in studying its behavior analytically by examining its behavior near the infinite-system fixed point and by making contact with variational states. These developments are the subject of the remainder of Part I of this book, and a wide variety of applications are discussed in Part II.

A Density Matrix for Single Particle Systems

Here we show, in the case of a single particle, that the density matrix is equivalent to a simple projection of the wavefunction, used in the superblock method for the tight-binding chain described in Sect. 4.

Consider a wavefunction $\psi(k)$, where k runs over the sites of the system, $k = 1, \ldots, L$. We will call sites $1, \ldots, \ell$ the left block, labeled by i, and sites

$\ell+1,\ldots,L$ the right block, labeled by j. In order to write a single-particle wavefunction in a product form

$$|\psi\rangle = \sum_{ij} \psi_{ij}|i\rangle|j\rangle \;, \tag{A.1}$$

it is necessary to construct an enlarged basis which includes zero and two-particle states. Specifically, we use the basis

$$\begin{aligned} &|0\rangle_{\mathrm{L}} \\ &|1\rangle_{\mathrm{L}} = c_1^\dagger|0\rangle_{\mathrm{L}} \\ &\quad\vdots \\ &|\ell\rangle_{\mathrm{L}} = c_\ell^\dagger|\ell\rangle_{\mathrm{L}} \end{aligned} \tag{A.2}$$

for the left block, and similarly for the right block. Here $c_i^\dagger$ creates a particle at site i. Then the wavefunction $\psi(k)$ can be written in matrix form as

$$\psi = \begin{pmatrix} 0 & \psi(\ell+1) & \ldots & \psi(L) \\ \psi(1) & & & \\ \vdots & & 0 & \\ \psi(\ell) & & & \end{pmatrix} . \tag{A.3}$$

In this matrix, the upper left zero represents the amplitude in the state $|0\rangle_{\mathrm{L}} \otimes |0\rangle_{\mathrm{R}}$. This state is included in the basis but since there is one particle, its coefficient is always zero. The rest of the first column represents the states $|i\rangle_{\mathrm{L}} \otimes |0\rangle_{\mathrm{R}}$, and similarly the rest of the first row represents $|0\rangle_{\mathrm{L}} \otimes |j\rangle_{\mathrm{R}}$. The large lower right block of zeros represents two particle states.

Then the density matrix $\rho_{ii'}$ in matrix form is

$$\rho = \psi\psi^\dagger = \begin{pmatrix} w_{\mathrm{R}} & 0 & 0 & \ldots & 0 \\ 0 & \psi_1\psi_1 & \psi_1\psi_2 & \ldots & \psi_1\psi_\ell \\ 0 & \psi_2\psi_1 & \psi_2\psi_2 & \ldots & \psi_1\psi_\ell \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \psi_\ell\psi_1 & \psi_\ell\psi_2 & \ldots & \psi_\ell\psi_\ell \end{pmatrix} , \tag{A.4}$$

where $w_{\mathrm{R}} = 1 - w_{\mathrm{L}} = 1 - \sum_{i=1}^{\ell} |\psi_i|^2$ is the probability that the particle is in the right block. This density matrix has two nonzero eigenvalues, with corresponding eigenvectors $(1\ 0\ \ldots\ 0)^T$ and $w_{\mathrm{L}}^{-1/2}(\psi_1 \ldots \psi_\ell)^T$. The first eigenvector does not need to be explicitly treated; the second is equivalent to the projection of the wavefunction considered in Sect. 4.

B A Program in C++ for the Particle on a Chain

General

In order to illustrate the DMRG method more explicitly, we describe here the DMRG algorithm and a complete DMRG C++ program to obtain the ground state energy and wave function for the particle on-a-chain problem numerically. We hope that after writing or playing with a program based on this appendix, the main part of this chapter should be much easier to understand. The algorithm which we will describe here is a simplified version of the finite system algorithm described in Sect. 5.

Our goal is to solve the particle on a chain problem with Hamiltonian (5) numerically. To make the program more like an efficient DMRG program for an interacting system, in which the dimensions of the matrices are much larger, we assume that our computer has only enough memory to diagonalize a 4×4 matrix. We will assume that it has enough "disk" storage to hold about two vectors of length L. By this we mean that the innermost loops involve manipulations of only a small amount of data, and access to the main data of size $\sim L$ occurs in outer loops, a little at a time. We will assume we do not know the analytic solution to the problem, and the algorithm we present will work equally well if the nonzero elements of H are perturbed slightly, either randomly or not. The solution we present is the DMRG method, in simplified form only because it is applied to a single particle problem.

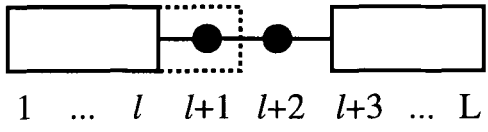

Fig. B.1. Breakup of a system, of length L, into two blocks and two sites.

Our procedure will divide the system up into four pieces, called blocks, as shown in Fig. B.1. (This is the "superblock" of Sect. 5.) Block 1 (the "left block") will consist of sites $1 \ldots \ell$; block 2 will consist of a single site, $\ell + 1$; block 3 will also consist of a single site, $\ell + 2$; and block 4 (the "right block") will consist of sites $\ell + 3 \ldots L$. During the course of the algorithm, the dividing point ℓ will be moved back and forth through the system, so that every site except the first and last is represented by one of the two middle sites during a "sweep" through the system.

Each block will be represented by a single basis state. For the middle blocks, this is not a limitation; for the left and right blocks, it is a severe one. A wavefunction ψ_j, $j = 1, \ldots L$, is written as

$$\psi_j = \begin{cases} a_1 L_j & j \leq \ell \\ a_2 & j = \ell + 1 \\ a_3 & j = \ell + 2 \\ a_4 R_j & j \geq \ell + 3 \end{cases} . \tag{B.1}$$

The basis states L and R are defined only within the corresponding blocks, and they are normalized, $\langle L|L\rangle = \langle R|R\rangle = 1$. Note that any state can be represented in this way if L and R are chosen appropriately for that state. Using (6) and (B.1), we find that the Hamiltonian matrix element between two states ψ and ψ' can be written as

$$\langle\psi|H|\psi'\rangle = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}^T \begin{pmatrix} H_{11} & T_{12} & 0 & 0 \\ T_{12} & 2 & -1 & 0 \\ 0 & -1 & 2 & T_{34} \\ 0 & 0 & T_{34} & H_{44} \end{pmatrix} \begin{pmatrix} a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \end{pmatrix} \tag{B.2}$$

where

$$H_{11} = \langle L|H|L\rangle , \tag{B.3}$$

$$H_{44} = \langle R|H|R\rangle , \tag{B.4}$$

$$T_{12} = \langle L|H|l+1\rangle = -L_\ell , \tag{B.5}$$

and

$$T_{34} = \langle l+2|H|R\rangle = -R_{\ell+3} . \tag{B.6}$$

The 4×4 matrix H in (B.2) represents the Hamiltonian in our restricted basis. Its minimum eigenvalue is an upper bound for the exact ground state energy. When L and R represent the left and right parts of the exact ground state, then the minimum eigenvalue is the exact ground state energy.

In order to turn this into an algorithm for finding the ground state of the system, we will specify an iterative procedure for improving L and R. Let $L(\ell)$ and $R(\ell+3)$ represent the bases for step ℓ. Then we want to improve the set of all basis states, $\{L(\ell)\}$ and $\{R(\ell)\}$. Our basic step in this procedure will be to get an improved basis state $L(\ell+1)$, given $L(\ell)$ and $R(\ell+3)$. Given this improved basis state, we proceed through the lattice from left the right, improving the set $\{L(\ell)\}$. When we reach the right end, we will reverse the procedure, generating an improved $R(\ell+2)$, given $L(\ell)$ and $R(\ell+3)$, as we proceed through the lattice in the reverse direction, eventually improving all elements of the set $\{R(\ell)\}$.

The basic left-to-right step is this: First, we diagonalize our 4×4 Hamiltonian matrix, (B.2), obtaining a ground state wavefunction in this basis, a vector of length 4, which we write as (a_1, a_2, a_3, a_4). We normalize a_1 and a_2

as $a_1' = a_1/N$ and $a_2' = a_2/N$, where $N = (a_1^2 + a_2^2)^{1/2}$. Then the new basis state, properly normalized, is

$$L(\ell+1)' = \begin{pmatrix} a_1' L(\ell)_1 \\ \vdots \\ a_1' L(\ell)_\ell \\ a_2' \end{pmatrix} . \tag{B.7}$$

The new Hamiltonian matrix element, needed to construct H for the next step, is

$$\langle L(\ell+1)'|H|L(\ell+1)'\rangle = {a_1'}^2 \langle L(\ell)|H|L(\ell)\rangle + 2{a_2'}^2 - 2a_1' a_2' L(\ell)_\ell . \tag{B.8}$$

If $L(\ell)$ and $R(\ell+3)$ are the exact left and right pieces of the true ground state of the system, then $L(\ell+1)'$ obtained in this way is the exact left piece of the true ground state, when the system is divided at $\ell+1$ rather than ℓ. Thus if our procedure ever finds the correct ground state at some step ℓ, then after one additional sweep through the lattice, all the $\{L(\ell)\}$ and $\{R(\ell)\}$ will represent exact parts of the true ground state and the procedure will have converged. Furthermore, assume $|\psi(\ell)\rangle$ is the state at step ℓ. Then, with $L(\ell+1)'$ as specified in (B.7), in the new basis at step $\ell+1$, $|\psi(\ell)\rangle$ can be represented exactly. Therefore the energy at step $\ell+1$ cannot be higher than at step ℓ. However, the basis in step $\ell+1$ generally has additional degrees of freedom not in the basis at step ℓ, and consequently, the energy at step $\ell+1$, obtained from diagonalizing the 4×4 Hamiltonian matrix, will generally be lower than at step ℓ. Consequently, we expect a monotonic convergence to the true ground state with this procedure.

In order to initialize this iterative procedure, we need to generate a set of approximate initial blocks $\{L(\ell)\}$ and $\{R(\ell)\}$. A good initial approximation comes from applying our procedure to a set of systems of sizes 4, 6, ..., L. Starting with the system of size 4, the left and right blocks are just single sites, with trivial L and R. We use this system to get an approximate $L(2)$. Proceeding to the system of size 6, this $L(2)$ can be used, and for the block R we use the reflection of $L(2)$, switching the left and right sides of the basis state. At each additional step, a reflection of the left block is used for the right block. In this way, we gradually grow the system up to the full size L, generating an initial set of $\{L(\ell)\}$ and $\{R(\ell)\}$.

Program

We now present a complete C++ program to solve the particle on a chain problem using the steps outlined above. The object-oriented nature of C++ means that the main routine, for a particle on a chain, can be reused with rather little change for a non-trivial interacting problem. The program utilizes the "`MatrixRef`" matrix library, written by the authors. Both the MatrixRef matrix library and a complete version of the program described here are

currently available on the world-wide web at http://hedrock.ps.uci.edu. In what follows, we will list and discuss the important parts of the program.

Blocks: We first define a class for a `Block`.

```
class Block
    {
public:
    Real H11, L_inner;
    Block()        // Default: construct a one-site block
        : H11(2.0), L_inner(1.0) { }
    Block Reflect() const { return *this; }
    };
```

Here type (`Real` is defined in the matrix library as a `double`, equivalent to a Fortran `real*8`). The two variables `H11` and `L_inner` store the values of either $\langle L|H|L\rangle$ or $\langle R|H|R\rangle$, and either L_ℓ or $R_{\ell+3}$, depending on whether the block is a left or right block. The constructor function `Block()` initializes the block in the way appropriate for a single site block, with $H_{11} = 2$ and $L_\ell = 1$. The `Reflect()` function, which reflects a block, does nothing in this simple case because of the dual usage of `L_inner`, but is included to illustrate the more general case.

System: Next we define a class `System`, which contains the four `Blocks` making up the lattice, for a particular step ℓ.

```
class WaveFunction;

class System
    {
public:
    const Block &b1, &b2, &b3, &b4;
    System(const Block& bb1,const Block&bb2,
                const Block&bb3, const Block&bb4)
        : b1(bb1), b2(bb2), b3(bb3), b4(bb4) { }

    Real GetGroundState(WaveFunction& p);
    };
```

A `System` has four data members, references to four blocks, b1, b2, b3, and b4. The function `GetGroundState`, returns a `Real` number, the ground state energy, and finds a ground state wavefunction. The definition of `WaveFunction` is

```
class WaveFunction
    {
public:
```

```
Vector v;
WaveFunction() : v(4) {}
};
```

Wave function: A `WaveFunction` here is simply a `Vector` of length 4. Vectors are defined in the matrix library. The function `GetGroundState` is

```
Real System::GetGroundState(WaveFunction& p)
    {
    Matrix H(4,4), evecs(4,4);
    Vector evals(4);
    H = 0.0;
    H(1,1) = b1.H11;   H(2,2) = b2.H11;
    H(3,3) = b3.H11;   H(4,4) = b4.H11;
    H(1,2) = H(2,1) = -b1.L_inner;
    H(2,3) = H(3,2) = -1.0;
    H(3,4) = H(4,3) = -b4.L_inner;
    EigenValues(H,evals,evecs);
    p.v = evecs.Column(1);
    if(p.v.sumels() < 0.0) p.v *= -1.0;
    Real energy = evals(1);
    return energy;
    }
```

This function defines a 4×4 matrix H, a matrix of eigenvectors evecs, and a vector of eigenvalues evals. H is first initialized to contain all 0's, and then the non-zero elements are defined according to (B.2), (B.5), and (B.6). Then, the Matrix library routine `EigenValues` is called, which diagonalizes H. The `WaveFunction` p is set to the ground state vector, and the ground state energy is returned.

Density matrix: The last class we will define is a `DensityMatrix`. Thus far, in our description of the single particle DMRG algorithm, density matrices have not appeared. However, as shown in Appendix A, the elements a_1 and a_2 of the wavefunction ψ are equivalent to the reduced density matrix for the first two blocks as part of the system as a whole.

```
enum LR {Left, Right};

class DensityMatrix
   {
public:
   Real a,b;
   DensityMatrix(const WaveFunction& psi,LR lr)
      {
      if(lr == Left)
          { a = psi.v(1); b = psi.v(2); }
```

```
    else
        { a = psi.v(4); b = psi.v(3); }
    }
Vector NewBasis()
    {
    Vector res(2);
    Real norm = sqrt(a*a+b*b);
    res(1) = a / norm;
    res(2) = b / norm;
    return res;
    }
};
```

A `DensityMatrix` has two data members, `a` and `b`, representing a_1 and a_2 or a_4 and a_3, depending on the whether one is forming a density matrix for the left or right half of the system. The first line defines a new type, `LR`, which takes on only two symbolic values, `Left` and `Right`, indicating which half of the system we want. One makes a `DensityMatrix` out of a `WaveFunction` `psi` simply by assigning the appropriate values to `a` and `b`. The function `NewBasis` forms a new basis for $L(\ell+1)$, which simply involves normalizing the two-element vector (a,b).

In order to utilize the new basis to form a new block, there are functions `NewLeft` and `NewRight`, which combine two blocks (either the left two blocks [`NewLeft`], or the right two blocks [`NewRight`]) using a new basis, and using (B.8). Here we show `NewLeft`; `NewRight` is quite similar.

```
Block NewLeft(const Block& b1, const Block& b2, const Vector& bas)
    {
    Block res;
    res.H11 = bas(1) * bas(1) * b1.H11 + 2 * bas(2) * bas(2)
        - 2 * bas(1) * bas(2) * b1.L_inner;
    res.L_inner = bas(2);
    return res;
    }
```

Main program: The main program is relatively straightforward, using these classes and routines. First, we read in the length of the system and the number of sweeps to be performed, and initialize an array of blocks which will hold both $\{L(\ell)\}$ and $\{R(\ell)\}$.

```
int main()
    {
    Block siteblock;
    cerr << "Input length, number of iterations: ";
    int i, length, nsweeps;
    cin >> length >> nsweeps;
    cout << "length, nsweeps = " << length SP nsweeps << endl;
```

```
cout << "Exact energy = " << exacten(length) << endl;
exlen = length;
Array1<Block> allblocks(length);
WaveFunction psi;
Real energy;
```

Next, we perform the warmup sweep, which uses the infinite-size algorithm to build up the system from four sites initially to the full system size.

```
// Warmup sweep, using the infinite-size algorithm
   allblocks[1] = siteblock;
   for(i = 1; i < length/2; i++)
      {
      Block rightblock = allblocks(i).Reflect();
      System S(allblocks(i),siteblock,siteblock,rightblock);
      energy = S.GetGroundState(psi);
      cout << i+1 SP psi.v(2) SP energy SP 0 << endl;
      DensityMatrix rho(psi,Left);
      Vector basis = rho.NewBasis();
      allblocks[i+1] = NewLeft(allblocks(i),siteblock,basis);
      }
```

The macro SP in the cout statement makes a space, and is defined in the matrix library. The final 0 to be printed out on each line indicates this is the warmup sweep. Once the system has reached its full size, nsweeps sweeps are performed to converge to the ground state. Each sweep consists of a right-to-left part, and a left-to-right part. In this program, because the system is symmetric under reflections, the sweeps start or end at the middle, symmetric configuration, rather than the far right site. The right-hand blocks in the middle configuration are obtained as reflections of the left hand blocks.

```
// Finite System sweeps
   for(int swp = 1; swp <= nsweeps; swp++)
      {
// We assume reflection symmetry:
      allblocks[length/2 + 2] = allblocks(length/2 - 1).Reflect();
      cout << endl;

// Right to left
      for(i = length/2+2; i > 3; i--)
         {
         System S(allblocks(i-3),siteblock,siteblock,allblocks(i));
         energy = S.GetGroundState(psi);
         cout << i-1 SP psi.v(3) SP energy SP swp - 0.5 << endl;
         DensityMatrix rho(psi,Right);
         Vector basis = rho.NewBasis();
         allblocks[i-1] = NewRight(siteblock,allblocks(i),basis);
         }
```

```
// Left to right
        cout << endl << 1 SP psi.v(1) SP energy SP swp << endl;
        for(i = 1; i < length/2-1; i++)
           {
           System S(allblocks(i),siteblock,siteblock,allblocks(i+3));
           energy = S.GetGroundState(psi);
           cout << i+1 SP psi.v(2) SP energy SP swp << endl;
           DensityMatrix rho(psi,Left);
           Vector basis = rho.NewBasis();
           allblocks[i+1] = NewLeft(allblocks(i),siteblock,basis);
           }
        }
     return 0;
     }
```

At each step i, the energy and value of the wavefunction at site $i+1$ is printed out. (We use i, rather than l, as the step variable because ℓ looks too much like 1 in program listings.)

Results

Results from this program are shown in Fig. B.2. The ground state energy is plotted as a function of the index of the site ℓ just added, for the warmup and first three sweeps. Note that during the warmup, the system length is

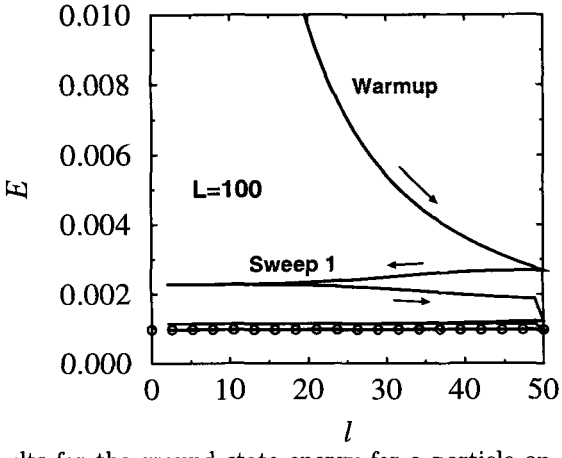

Fig. B.2. Results for the ground state energy for a particle on a chain of length $L = 100$.

less than the eventual system size, which accounts for most of difference in energy from the exact result. (Here $\ell = i + 1$ for left to right sweeps and $\ell = i - 1$ for right to left sweeps.) During the warmup sweep, the system size

is still growing, and at each step the length is 2ℓ. Starting with the beginning of the first sweep, the length is fixed at L. Each sweep starts in the middle, progresses to the left side, and returns. There is a discontinuous jump in the energy between sweeps because at those points a reflection of the left block is used to replace the first right block of the new sweep. During the warmup, this is done every step, but it occurs only once per sweep during the other sweeps. The open circles represent the exact energy, $E = 0.000967435$. (The general formula is $E = 2[1 - \cos(\frac{\pi}{L+1})]$.) Good convergence is obtained after three sweeps.

Results for the wavefunction are shown in Fig. B.3. The wavefunction at the site just added is plotted versus the site index for the first three sweeps. Each half-sweep is labeled by an integer, starting at 1. The open circles represent the exact wavefunction, which is given (in unnormalized form) by

$$\psi_j = \sin(qj) \tag{B.9}$$

with $q = \pi/(L+1)$, and $j = 1, \ldots L$. Convergence of the wavefunction is reasonable after three sweeps; another sweep or two would be needed for precise agreement with the exact results.

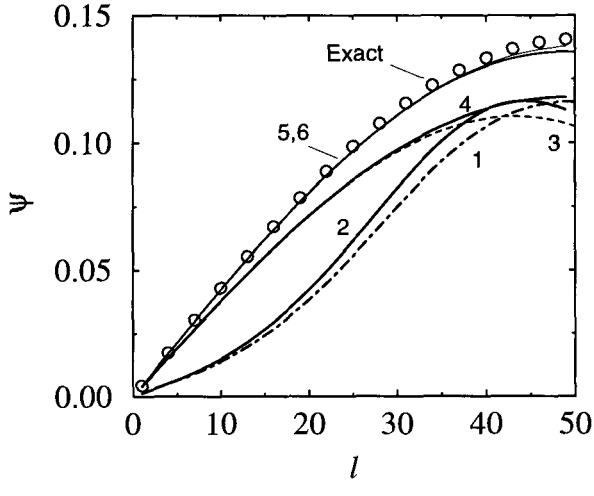

Fig. B.3. Results for the wavefunction for a particle on a chain of length $L = 100$.

The above program can be easily modified to find more than just one state, to deal with modifications to the Hamiltonian, including randomness, etc. A useful exercise would be to modify this program to include randomness or another local potential, to include longer range hopping, etc. Another useful exercise would be to translate this program into another programming language.

References

1. S.R. White, Phys. Rev. Lett. **69**, 2863 (1992), Phys. Rev. B **48**, 10345 (1993)
2. K.G. Wilson, Rev. Mod. Phys. **47**, 773 (1975)
3. K. Hallberg, Phys. Rev. B **52**, 9827 (1995); H. Pang, H. Akhlaghpour and M. Jarrell, Phys. Rev. B **53**, 5086 (1996)
4. T. Nishino, J. Phys. Soc. Jpn. **64**, 3598 (1995)
5. S. Moukouri and L.G. Caron, Phys. Rev. Lett. **77**, 4640 (1996)
6. R.J. Bursill, T. Xiang and G.A. Gehring, J. Phys. Cond. Matt. **8**, 583 (1996)
7. X.Q. Wang and T. Xiang, Phys. Rev. B **56**, 5061 (1997)
8. S.R. White and R.L. Martin, unpublished, cond-mat/9808118
9. E.R. Davidson, J. Comp. Phys. **17**, 87 (1975); E.R. Davidson, Computers in Physics **7**, No. 5, 519 (1993)
10. J.W. Bray and S.T. Chui, Phys. Rev. B **19**, 4876 (1979); T. Xiang and G.A. Gehring, Phys. Rev. B **48**, 303 (1993)
11. S.R. White and R.M. Noack, Phys. Rev. Lett. **68** 3487, (1992)
12. R.P. Feynman, Statistical Mechanics: A Set of Lectures, Benjamin (1972)
13. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge (1986)
14. E. Jeckelmann and S.R. White, Phys. Rev. B **57**, 6376 (1998)
15. E.H. Lieb and F.Y. Wu, Phys. Rev. Lett. **20**, 1445 (1968)
16. H. Schulz, J. Phys. C: Solid State Phys. **18**, 581 (1985)
17. S. Kneer, Diploma Thesis, Universität Würzburg (1997)
18. G. Bedürftig, B. Brendel, H. Frahm and R.M. Noack, Phys. Rev. B **58**, 10225 (1998)
19. X.Q. Wang and S. Mallwitz, Phys. Rev. B **48**, R492 (1996)
20. S.J. Qin, X.Q. Wang, Lu Yu, Phys. Rev. B **56**, R14251 (1997)
21. H.J. Schulz, Int. J. Mod. Phys. B **5**, 57 (1991)
22. S.R. White, Phys. Rev. Lett. **77**, 3633 (1996)
23. S. Östlund and S. Rommer, Phys. Rev. Lett. **75**, 3537 (1995)
24. R.M. Noack, S.R. White and D.J. Scalapino, in Computer Simulations in Condensed Matter Physics VII, Eds. D.P. Landau, K.K. Mon and H.B. Schüttler Spinger, p 85 (1994)
25. S. Liang, Phys. Rev. B **49**, 9214 (1994)