

Implementation of the SU(2) Hamiltonian Symmetry for the DMRG Algorithm

G. Alvarez

Computer Science & Mathematics Division and Center for Nanophase Materials Sciences, Oak Ridge National Laboratory,
Oak Ridge, TN 37831 USA

Abstract

In the Density Matrix Renormalization Group (DMRG) algorithm[1], Hamiltonian symmetries play an important rôle. Using symmetries, the matrix representation of the Hamiltonian can be blocked. Diagonalizing each matrix block is more efficient than diagonalizing the original matrix. This paper explains how the the DMRG++ code[2] has been extended to handle the non-local SU(2) symmetry in a model independent way. Improvements in CPU times compared to runs with only local symmetries are discussed for the one-orbital Hubbard model, and for a two-orbital Hubbard model for iron-based superconductors. The computational bottleneck of the algorithm and the use of shared memory parallelization are also addressed.

Key words: density-matrix renormalization group, dmrg, strongly correlated electrons, generic programming

PACS: 71.10.Fd 71.27.+a 78.67.Hc

PROGRAM SUMMARY

Manuscript title: Implementation of the SU(2) Hamiltonian Symmetry for the DMRG Algorithm

Author: Gonzalo Alvarez

Program title: DMRG++

Licensing provisions: See file LICENSE.

Source code: <http://www.ornl.gov/~gz1/dmrgPlusPlus/>

Programming language: C++

Computer(s) for which the program has been designed: PC

Operating system(s) for which the program has been designed: multiplatform, tested on Linux

RAM required to execute with typical data: 1GB (256MB is enough to run included test)

Has the code been vectorized or parallelized?: Yes

Number of processors used: 1 to 8 with MPI, 2 to 4 cores with *pthread*s

Keywords: density-matrix renormalization group, dmrg, strongly correlated electrons, generic programming

PACS: 71.10.Fd 71.27.+a 78.67.Hc

CPC Library Classification: 23 Statistical Physics and Thermodynamics

External routines/libraries used: BLAS and LAPACK

CPC Program Library subprograms used: None.

Nature of problem: Strongly correlated electrons systems, display a broad range of important phenomena, and their study is a major area of research in condensed matter physics. In this context, model Hamiltonians are used to simulate the relevant interactions of a given compound, and the relevant degrees of freedom. These

studies rely on the use of tight-binding lattice models that consider electron localization, where states on one site can be labeled by spin and orbital degrees of freedom. The calculation of properties from these Hamiltonians is a computational intensive problem, since the Hilbert space over which these Hamiltonians act grows exponentially with the number of sites on the lattice.

Solution method: The DMRG is a numerical variational technique to study quantum many body Hamiltonians. For one-dimensional and quasi one-dimensional systems, the DMRG is able to truncate, with bounded errors and in a general and efficient way, the underlying Hilbert space to a constant size, making the problem tractable.

Running time: Varies

1. Introduction

In the DMRG algorithm[1] and other diagonalization-based methods, Hamiltonian symmetries play an important rôle. An operator \hat{S} is a Hamiltonian symmetry if it commutes with the Hamiltonian, *i. e.*, if $[\hat{H}, \hat{S}] = 0$. If $S|\psi_1\rangle = s_1|\psi_1\rangle$, and $S|\psi_2\rangle = s_2|\psi_2\rangle$, then $\langle\psi_1|H|\psi_2\rangle = 0$ provided that $s_1 \neq s_2$. In words, \hat{H} cannot “connect” states with different symmetries. The matrix representation of \hat{H} is then block diagonal, and diagonalizing each matrix block is more efficient than diagonalizing the original matrix.

Reference [2] introduced DMRG++, a generic implementation of the DMRG algorithm. There it was shown how to take advantage of local symmetries in a generic way, *i. e.*, symmetries \hat{S} , such that $\hat{S} = \sum_i \hat{S}_i$, where \hat{S}_i

acts only on site i . In this paper the DMRG++ code is extended to handle the non-local SU(2) symmetry.

Many Hamiltonians for strongly correlated electronic systems possess this symmetry, since they conserve the full spin. For example, the Heisenberg model with any spin, the Hubbard model[3, 4] for any filling with one or multiple orbitals, and the t-J model[5, 6]. This is true as long as there are no external magnetic fields. The implementation of the SU(2) symmetry is involved, particularly if done in a generic way, but once implemented it provides substantial performance improvements to each of these models, *e. g.*, the Hubbard model with one orbital runs four times faster for $m \geq 400$, as we will show. All this is achieved without introducing any approximations.

Due to the wide applicability to various models, and the performance improvement that this symmetry brings, it is studied in detail in this paper, and is implemented in the accompanying DMRG++ code, which can be found at <http://www.ornl.gov/~gzi/dmrgPlusPlus/>. Section 2 describes the implementation details of the SU(2) symmetry for the Hilbert space basis in a model independent way. The work on Hilbert space operators is described in section 3, including performance improvements by using reduced operators with the help of the Wigner-Eckart theorem. The performance bottleneck of the code is also analyzed, and shared memory parallelization is introduced for the performance critical parts of the code.

In section 4, the method is applied first to the Hubbard model and then to a model for iron-based superconductors[7]. These are new materials whose superconducting pairing mechanism, like in the cuprates, appears to be of electronic origin.

Finally, a summary is presented. The appendices contain a few derivations used in the text, as well as some documentation to be able to run the code.

The problem discussed here was treated originally by McCulloch *et al.*, in References [8, 9]. Comparison to their results is provided.

2. Hilbert Space Basis

2.1. Basis on a Single Site

Consider the usual[10] SU(2) operators S^+ , $S^- = (S^+)^\dagger$, S^z , and $S^2 = \frac{1}{2}(S^+S^- + S^-S^+) + (S^z)^2$. In all physical cases these are spin operators—the SU(2) symmetry is actually a full spin symmetry in the absence of magnetic fields—but this does not concern us at this point. We consider that the basis is diagonal in S^2 and S^z , *i. e.*, $S^2|a\rangle = j(j+1)|a\rangle$, and $S^z|a\rangle = m|a\rangle$. In the code we work with $\tilde{j} = 2j$ instead of j , and with $\tilde{m} = m+j$ instead of m , because \tilde{j} and \tilde{m} are always non-negative integers. Since it is standard notation, we will use (j, m) in the paper; the bijective mapping between (j, m) and (\tilde{j}, \tilde{m}) allows us to use (\tilde{j}, \tilde{m}) in the code.

We consider that q is the quantum number associated with some local operator Q (in the case studies it will

be the “total number of electrons”, N_e , operator). We will consider Hamiltonians that conserve S^2 , S^z , and, Q . Q can actually be formed by more than one local operator, using the effective symmetry procedure described in Ref. [2]. However, Q should not include S^z , that is treated explicitly instead. In general, j , m and q , (or equivalently \tilde{j} , \tilde{m} , q) *does not* completely determine the states of the basis.

In addition to j , m , and q , we now introduce a fourth quantum number, *flavor* or f , that will be useful in our implementation of the SU(2) symmetry for DMRG, but f will not necessarily be conserved. The following definition applies only to states that are eigenstates of S^2 , *i. e.*, that have a well defined j quantum number. We define the following relation $|a\rangle \stackrel{f}{\approx} |b\rangle$, if $\exists p \geq 0$ such that either $(S^+)^p|a\rangle = \eta_{p,j,m}|b\rangle$ or $(S^+)^p|b\rangle = \eta_{p,j,m}|a\rangle$ holds, where $\eta_{p,j,m} = \prod_{x=0}^{x=p-1} g_{j,m-x}$ if $p > 0$ and $\eta_{0,j,m} = 1$; and $g_{j,m} = \sqrt{j(j+1) - m(m+1)}$. That two states have the same flavor (*i. e.*, that $|a\rangle \stackrel{f}{\approx} |b\rangle$) immediately implies that they have the same j value (*i. e.*, that $j_a = j_b$, where the notation j_a refers to the quantum number j of the state $|a\rangle$, and likewise for b). The relation $\stackrel{f}{\approx}$ is an equivalence relation that defines an equivalence class $[|a\rangle]_{\stackrel{f}{\approx}}$ for each element $|a\rangle \in \mathcal{V}$. We assign a different non-negative integer to each equivalence class, and call this number, the *flavor* of that state.

States with the same f and j belong to the same irreducible representation of SU(2). That states have the same j does not, *by itself*, imply that they belong to the same matrix representing SU(2). For example, in the Hilbert space of one site with spin 1/2 electrons and a single orbital, the empty state and the doubly occupied state have both $j = 0$, but they do not belong to the same (one-dimensional) matrix. In other words, these states are not connected by S^+ .

Two states have the same triplet j , m and f , if and only if they are equal (proof in Appendix A). In other words, j , m and f *completely and uniquely* determine the states of the basis.

2.2. Basis on Multiple Sites: Outer Products

Consider two vectors spaces with bases \mathcal{V}_1 and \mathcal{V}_2 , respectively. Assume that the states in these bases are eigenstates of both S^2 and S^z . Consider the vector space created by the outer product, $\mathcal{V}_3 \equiv \mathcal{V}_1 \otimes \mathcal{V}_2$. Let $S^+ : \mathcal{V}_3 \rightarrow \mathcal{V}_3$, be such that $S^+ = S_1^+ + S_2^+$ (where the subindices 1 and 2 indicate that S_1^+ acts only on \mathcal{V}_1 and S_2^+ acts only on \mathcal{V}_2), We define $S^z : \mathcal{V}_3 \rightarrow \mathcal{V}_3$ and $Q : \mathcal{V}_3 \rightarrow \mathcal{V}_3$ in the same way, $S^- = (S^+)^\dagger$, and $S^2 = \frac{1}{2}(S^+S^- + S^-S^+) + (S^z)^2$. How can we construct a basis of this outer product whose states are also eigenstates of S^2 and S^z ? (One immediately notes that the states $|a\rangle \otimes |b\rangle$, with $|a\rangle \in \mathcal{V}_1$, and $|b\rangle \in \mathcal{V}_2$, are not necessarily eigenstates of S^2 .) The most

general solution is

$$|c\rangle = \sum_{a,b} G_{c,a+bN_1} |a\rangle \otimes |b\rangle, \quad (1)$$

where N_1 is the number of states in \mathcal{V}_1 . In the case of S^2 we have an ansatz for G in terms of Clebsch-Gordon coefficients (see, *e. g.*, Ref. [10]).

Before proceeding to create the basis, we need to explain how to assign quantum numbers to the outer product of states. It is true that $Q|a\rangle \otimes |b\rangle = (q_a + q_b)|a\rangle \otimes |b\rangle$, and that $S^z|a\rangle \otimes |b\rangle = (m_a + m_b)|a\rangle \otimes |b\rangle$. But $|a\rangle \otimes |b\rangle$ is not necessarily an eigenvector of S^2 , as mentioned before. Therefore, it does not have a well defined flavor either. We now extend the definition of flavor for these states in the following way: $|a\rangle \otimes |b\rangle \stackrel{f}{\approx} |a'\rangle \otimes |b'\rangle$ if and only if all these equalities hold: $j_a = j_{a'}$, $f_a = f_{a'}$, and $q_a = q_{a'}$; $j_b = j_{b'}$, $f_b = f_{b'}$, and $q_b = q_{b'}$. Again, $\stackrel{f}{\approx}$ is an equivalence relation, and we define equivalence classes, and assign flavors as different non-negative integers to each equivalence class. In the few cases where $|a\rangle \otimes |b\rangle$ is an eigenvector of S^2 , this new definition of f is equivalent to the previous one.

Then, $|a\rangle \otimes |b\rangle$ is assigned the flavor

$$f_{a \otimes b} \equiv f_a + f_b F_1 + (q_a + q_b Q_1) F_1 F_2 + (\tilde{j}_a + \tilde{j}_b \tilde{J}_1) F_1 F_2 Q_1 Q_2, \quad (2)$$

where $f_a < F_1 \forall |a\rangle \in \mathcal{V}_1$, $q_a < Q_1$, $\tilde{j}_a < \tilde{J}_1$, and likewise for \mathcal{V}_2 . The rationale is similar to the one-site case; states with the same flavor belong to the same matrix representation of $SU(2)$. In Eq. (1), the pairs $|a\rangle \otimes |b\rangle$ that contribute to a given state $|c\rangle$ all have the same flavor $f_{a \otimes b}$, which in turn becomes the flavor of state $|c\rangle$.

Each pair of states¹ ($|a\rangle$, $|b\rangle$), with $|a\rangle \in \mathcal{V}_1$, and $|b\rangle \in \mathcal{V}_2$, will contribute to one or more states $|c\rangle$ of \mathcal{V}_3 . We first classify the pair $a + bN_1$ in the following way. We calculate all the allowed j, m that j_a, m_a and j_b, m_b give rise to. Then, we assign the pair $a + bN_1$ to each one of these $\mathcal{S}_{j,m,q \equiv q_a + q_b}$ subspaces. After classifying all pairs we end up with a set of allowed j, m values, and there is one and only one subspace $\mathcal{S}_{j,m,q}$ for each one of those j, m values. The pairwise intersection of these subspaces is not necessarily empty, because one pair of states $a + bN_1$ may contribute to more than one state c in Eq. (1).

Each subspace $\mathcal{S}_{j,m,q}$ is represented by an object of class `JmSubspace`. We now need to determine how many basis states c for \mathcal{V}_3 are to be created, and what the corresponding factors G are. For these tasks, we run the loop given in listing 1.

Listing 1: Loop that each subspace $\mathcal{S}_{j,m,q}$ of the outer product runs to determine the factors G of Eq. (1).

```
size_t flavorSaved=flavorIndices_[0];
flavors_.push_back(flavorIndices_[0]);
size_t counter=0;
```

```
for (size_t k=0;k<indices_.size();k++) {
  if (flavorIndices_[k]!=flavorSaved) {
    flavors_.push_back(flavorIndices_[k]);
    counter++;
    flavorSaved = flavorIndices_[k];
  }
  // G(offset+counter, indices_[perm[k]]) =
  // = values_[perm[k]]
  if (heavy_) factors.set(indices_[perm[k]],
    offset + counter, values_[perm[k]]);
}
```

In this loop `indices_` contains the states $a + bN_1$ for this particular $\mathcal{S}_{j,m,q}$, and `flavorIndices_` the flavor of each $a + bN_1$ state. For each pair $a + bN_1$ we have computed a vector of `values_` that contains the Clebsch-Gordan coefficients $\langle j_a m_a j_b m_b | j m \rangle$. The states of the outer product, \mathcal{V}_3 , are labeled here by `offset+counter`, where `offset` is the number of states created by previous subspaces and `counter` is the number of states created by this subspace. Note that `flavorIndices_` is ordered to simplify the algorithm, which gives rise to a permutation `perm`. This loop does two main things: (i) it sets the flavor of each c , which is simply the flavor of the $a + bN_1$ values that form part of Eq. (1), and (ii) it sets `G(offset + counter, indices_[perm[k]]) = values_[perm[k]]`, as explained before. Finally, flavors can be reassigned new numbers in the basis \mathcal{V}_3 .

Now we have created a completely (*i. e.*, in j, m , and q) ordered basis for $\mathcal{V}_3 = \mathcal{V}_1 \otimes \mathcal{V}_2$ composed of eigenstates of S^2 (and S^z and Q). It is useful to be able to “disable” the $SU(2)$ symmetry, which is done by just taking G in Eq. (1) to be the identity, *i. e.*, $G_{c,a+bN_1} = \delta_{c,a+bN_1}$. We also need a permutation P^{12} to account for effective symmetry ordering[2]. Then Eq. (1) becomes

$$|c\rangle = \sum_{a,b} G_{P^{12}(c),a+bN_1} |a\rangle \otimes |b\rangle. \quad (3)$$

When the $SU(2)$ symmetry is “enabled”, G is non-trivial and P^{12} is the identity, and vice-versa. In the DMRG procedure, three types of outer products will appear, and there will be three factors G and three permutations P at each DMRG step.

The subspaces $\mathcal{S}_{j,m,q}$ for a given outer product space can be heavy or light, and this is denoted by the *boolean heavy_* in listing 1. When adding a new site to the system or to the environment, the subspaces are always heavy. When forming the superblock (by combining system and environment) the subspaces are heavy if $j = j_{target}$ and $q = q_{target}$, and light otherwise, where j_{target} and q_{target} are the j and q values of the ground state to be considered by the DMRG algorithm. Heavy subspaces compute the factors G , light subspaces compute only the offsets. This is done for performance reasons; the factors G are only computed when needed.

2.3. Change of Basis

For the DMRG basis transformation the first order of business is to calculate the density matrix for system and

¹In the code, these pairs are denoted by a single number, $a + bN_1$.

environment. If we label the states with $|j, m, f\rangle$ we get

$$\rho_{j_1 m_1 f_1; j'_1 m'_1 f'_1}^S = \sum_{j_2, m_2, f_2} \psi_{j_1 m_1 f_1; j_2 m_2 f_2}^* \psi_{j'_1 m'_1 f'_1; j_2 m_2 f_2}. \quad (4)$$

One roadblock here is that ρ^S does not necessarily conserve S^2 or S^z . To solve this problem, McCulloch *et al.* successfully proposed[8] to use the SU(2) invariant reduced density matrix,

$$\rho_{Inv. f_1; f'_1}^{S[j_1, m_1]} = \sum_{j_2, m_2, f_2} \psi_{j_1 m_1 f_1; j_2 m_2 f_2}^* \psi_{j_1 m_1 f'_1; j_2 m_2 f_2}, \quad (5)$$

instead of Eq. (4), and modify the DMRG truncation procedure accordingly.

The DMRG truncation procedure with $\rho_{Inv.}^S$ is as follows. We diagonalize $\rho_{Inv.}^S$, and consider its eigenvectors W^S ordered in increasing eigenvalue order. Let m be a fixed number that corresponds to the number of states in $\mathcal{V}(S)$ that are to be kept. If $m \geq \#\mathcal{V}(S)$, then W remains unchanged. But if $m < \#\mathcal{V}(S)$, then W is truncated by discarding all states above m , and thus W becomes a rectangular matrix of size $m \times \#\mathcal{V}(S)$. The basis of $\mathcal{V}(S)$ is transformed by applying the (possibly truncated) linear transformation W^S . Operators are transformed in the usual way $(H^{S \text{ new basis}})_{\alpha, \alpha'} = (W^S)_{\alpha, \gamma}^{-1} (H^S)_{\gamma, \gamma'} W_{\gamma', \alpha'}^S$. This procedure is repeated for the environment block.

The transformed state $W|\alpha\rangle$ has the same flavor as $|\alpha\rangle$ (see Appendix B). If $|j, m, f\rangle$ is to be discarded, then we need to be sure to discard all states $|j, m', f\rangle$ for all m' , else the remaining basis will not preserve the SU(2) symmetry. Alternatively, if $|j, m, f\rangle$ is to be discarded but $|j, m' \neq m, f\rangle$ is not, then $|j, m, f\rangle$ is kept.

3. Operators and Optimizations

3.1. Product of Operators

As mentioned before, three types of outer products need be considered: (i) for the outer product of system and a newly added site(s), (ii) for the outer product of environment and a newly added site(s) and (iii) for the outer product of system and environment. For the first two the corresponding bases are stored in objects of class `DmrgBasisWithOperators`. For the outer product of system and environment, the outer product is done on-the-fly only because of memory storage reasons. If A^S and B^S are both in the system their product is:

$$(A^S B^S)_{c, c'} = \sum_{a, b, a'} G_{PS(c), a+bN_s}^S \left(\tilde{s}_a \sum_l A_{a, l}^S B_{l, a'}^S \right) \times G_{PS(c'), a'+bN_s}^S, \quad (6)$$

where $\tilde{s}_a = (\tilde{f})^{n_a}$, n_a is the number of electrons in state a , and $\tilde{f} = -1$ if A and B anticommute or $\tilde{f} = 1$ if they commute. If A^S is in the system and B^E is in the environment

their product is:

$$(A^S B^E)_{c, c'} = \sum_{a, b, a', b'} G_{P^{SE}(c), a+bN_s}^{SE} (\tilde{s}_a A_{a, a'}^S B_{b, b'}^E) \times G_{P^{SE}(c'), a'+b'N_s}^{SE} \quad (7)$$

3.2. Wigner-Eckart Theorem and Reduced Operators

The sums in Eq. (6) and Eq. (7) can be performed in a faster way[8] thanks to the Wigner-Eckart theorem. If operator A transforms as the representation of SU(2) labeled by J, M , then $\langle f' j' m' | A_M^J | f j m \rangle = C_{m' M m}^{j' J j} \langle f' j' | A^J | f j \rangle$, where

$$\langle f' j' | A^J | f j \rangle \equiv \frac{1}{2j'+1} \times \sum_{m', M, m} C_{m' M m}^{j' J j} \langle f' j' m' | A_M^J | f j m \rangle. \quad (8)$$

Since all operators that appear in constructing the Hamiltonian (for example, c^\dagger in the case of the Hubbard model, and S^z, S^+ in the case of the Heisenberg model) transform as some representation of SU(2), then these formulas can always be applied.

To “reduce”, for example, Eq. (6), we will first write

$$G_{c, a+bN_1} = C_{m_c, m_a, m_b}^{j_c, j_a, j_b} \delta_{f_a \otimes b, f_c}, \quad (9)$$

where $f_a \otimes b$ is given in Eq. (2), and then we will gather the sums over m, M, m' together.

We use throughout the notation $|a\rangle \equiv |f_a j_a m_a\rangle$. Let us assume that we have calculated the reduced operators $\langle f_l j_l | A^S | f_a j_a \rangle$ using definition Eq. (8), and similarly for $\langle f_{a'} j_{a'} | B^S | f_l j_l \rangle$. We assume that A^S and B^S are both in the system, that they commute (and then $\tilde{f} = 1$) or anticommute (and then $\tilde{f} = -1$), that $\tilde{s}_a = (\tilde{f})^{n_a}$ as before, that A^S transforms as the irreducible representation of SU(2) labeled by J_A and M_A , and that B^S transforms as the irreducible representation of SU(2) labeled by J_B and M_B .

We replace $(A^S)_{a, l} = C_{m_l M_A m_a}^{j_l J_A j_a} \langle f_l j_l | A^S | f_a j_a \rangle$, and the equivalent for $(B^S)_{l, a'}$ into Eq. (6). We obtain an expression for $(A^S B^S)_{c, c'}$ in terms of $\langle j_l f_l | A^S | j_a f_a \rangle$ and $\langle f_{a'} j_{a'} | B^S | f_l j_l \rangle$. Then we calculate $\langle f_{c'} j_{c'} | (A^S B^S) | f_c j_c \rangle$ again using Eq. (8) in terms of $(A^S B^S)_{c, c'}$, and replace $(A^S B^S)_{c, c'}$ by the expression we obtained before. The end result is

$$\langle f_{c'} j_{c'} | (A^S B^S) | f_c j_c \rangle = \sum_{a_R, b_R, a'_R, l_R, J_A, J_B} \mathcal{L}^S \times \delta_{f_a \otimes b, f_c} \delta_{f_{a'} \otimes b, f_{c'}} \langle f_l j_l | A^S | f_a j_a \rangle \langle f_{a'} j_{a'} | B^S | f_l j_l \rangle \tilde{s}_a, \quad (10)$$

where

$$\mathcal{L}^S = \sum_{m_a, m'_a, m_b, m_l, m_c, m'_c} C_{m_c, m_a, m_b}^{j_c, j_a, j_b} C_{m'_c, m'_a, m'_b}^{j_{c'}, j_{a'}, j_{b'}} \times C_{m_l, M_A, m_a}^{j_l, J_A, j_a} C_{m_{a'}, M_B, m_l}^{j_{a'}, J_B, j_l}, \quad (11)$$

and a_R represents a sum over f_a and j_a but not over m_a , and likewise for the other indices with subscript R . Note that the factor \mathcal{L}^S depends on a_R, b_R, a'_R, l_R, J_A , and J_B .

We assume now that B^E is an operator in the environment. Then a similar treatment of Eq. (7) yields:

$$\begin{aligned} \langle f_{c'} j_{c'} | (A^S B^E) | f_c j_c \rangle &= \sum_{a_R, b_R, a'_R, b'_R, J_A, J_B} \mathcal{L}^{SE} \times \\ &\times \delta_{f_a \otimes b, f_c} \delta_{f_{a'} \otimes b', f_{c'}} \langle f_{a'} j_{a'} | A^S | f_a j_a \rangle \langle f_{b'} j_{b'} | B^E | f_b j_b \rangle \tilde{s}_a, \end{aligned} \quad (12)$$

where

$$\begin{aligned} \mathcal{L}^{SE} &= \sum_{m_a, m'_a, m_b, m'_b, m_c, m'_c} C_{m_c, m_a, m_b}^{j_c, j_a, j_b} C_{m_{c'}, m_{a'}, m_{b'}}^{j_{c'}, j_{a'}, j_{b'}} \times \\ &C_{m_{a'}, M_A, m_a}^{j_{a'}, J_A, j_a} C_{m_{b'}, M_B, m_b}^{j_{b'}, J_B, j_b}. \end{aligned} \quad (13)$$

In the code, the class `ReducedOperators` keeps track of the reduced operators, and the class `Su2Reduced` calculates Eq. (10) and Eq. (12). This results in a substantial speed-up.

3.3. Shared Memory Parallelization

The most time consuming part of the DMRG method applied to strongly correlated electronic models is the computation of Hamiltonian connections between system and environment. These connections take the form $c_i^\dagger c_j$ for the Hubbard model, and $S_i^+ S_j^-, S_i^z S_j^z$ for the Heisenberg model, and are generically represented by Eq. (7) or its reduced form as explained before. There are a few of these connections in the case of the one-orbital Hubbard model on a one dimensional chain. There are a few dozen in the case of the two-orbital Hubbard model for iron-based superconductors on a ladder. Then, these connections can be parallelized using, for example, *pthread*², and the acceleration brought about by this procedure depends on the model, as the results of the next section show.

4. Case Studies

4.1. One-orbital Hubbard Hamiltonian

The one-orbital Hubbard model is given by:

$$H_U = \sum_{i,j} t_{i,j} c_{i\sigma}^\dagger c_{j\sigma} + U \sum_i n_{i\uparrow} n_{i\downarrow} + \sum_{i\sigma} V_{i\sigma} n_{i\sigma}. \quad (14)$$

This model has the SU(2) symmetry if we define $S^+ = \sum_i c_{i\uparrow}^\dagger c_{i\downarrow}$, $S^z = \sum_i (c_{i\uparrow}^\dagger c_{i\uparrow} - c_{i\downarrow}^\dagger c_{i\downarrow})$, and S^2 as usual from these operators and their transpose conjugates.

We start by reproducing results published in Ref. [8] with $U = 1$, $V_{i\sigma} = -0.5$, $t_{ij} = 1$ between nearest neighbors, and zero elsewhere, on a 60-site chain at half filling.

²*Pthreads* or POSIX threads is a standardized C language threads programming interface, specified by the IEEE POSIX standard.

Symmetry	m	Energy	CPU
Local	226	-76.751582	5332
Local	468	-76.751733	86681
Local	716	-91.751739	320911
SU(2)	226	-76.751582	1103
SU(2)	468	-76.751733	7564
SU(2)	716	-76.751739	36640
SU(2) j=5	226	-74.527742	1574
SU(2) j=5	468	-74.565375	7188
SU(2) j=5	716	-74.570932	20364

Table 1: Results for the Hubbard model with $U = 1$, $V_{i\sigma} = -0.5$, and $t = 1$ on a 60-site chain. Column 2 contains the m total states kept in each case (this is called D in Ref. [8]). Energies are in column 3. A factor of $UN/2 = 1 \times 30/2 = 15$ has been added to all energies to compare with Ref. [8]. CPU times in seconds are in the last column. All rows but the last three refer to the ground-state with $j = 0$. The last three rows are for the lowest eigenstate with $j = 5$.

M	SU(2) 1 proc	SU(2) 2 procs	Local 1 proc
100	42	41	67
200	160	136	319
300	335	290	808
400	544	485	1602
800	3020	2526	>2 hours

Table 2: Times in seconds to run the one-orbital Hubbard model on 32 sites at half filling, with $U = t = 1$. Runs done with 2 processors used shared memory parallelization with *pthread*s.

These results are shown in Table 1 for $j = 0$ and for $j = 5$. The infinite algorithm used m as given in the table, followed by one full sweep with the same m .

Having validated these results Table 2 gives additional CPU times for the Hubbard model on 16 sites. In all cases “Local” denotes the symmetries $n_e = n_\uparrow + n_\downarrow$ and $s_z = n_\uparrow - n_\downarrow$, whereas “SU(2)” denotes the symmetries n_e, s_z and s^2 .

4.2. Spin 1/2 Heisenberg Model

This model is given by the Hamiltonian, $\sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$, and has full spin symmetry. In this case, and using a 32-site chain with $J_{ij} = 1$ only between nearest neighbors, the SU(2) symmetry yields a speed-up factor roughly between 5 to 10, depending on m . However, the shared memory parallelization performs poorly, because this model has few connections between system and environment blocks.

4.3. Hamiltonian of Iron-Based Superconductors

In early 2008, high-temperature superconductivity was discovered[11] in the iron pnictides. Except for the cuprates, the iron-based superconductors now have the highest superconducting critical temperature T_c of any material[12]. Iron-based superconductors contain conducting layers of iron and arsenic. As in the cuprate superconductors, in the pnictides there is also evidence that the superconductivity is not mediated by the electron-phonon

interaction[13], but appears to be of electronic origin instead.

A tight-binding two-orbital Hubbard model for the iron pnictides has been proposed[7, 14]. This model's kinetic energy is given by

$$K = \sum_{i,\alpha,\gamma,\gamma',\sigma} t_{\gamma,\gamma'}^{\alpha} c_{i,\gamma,\sigma}^{\dagger} c_{i+\alpha,\gamma',\sigma}, \quad (15)$$

where

$$t^x = \begin{pmatrix} -t_1 & 0 \\ 0 & -t_2 \end{pmatrix}, t^y = \begin{pmatrix} -t_2 & 0 \\ 0 & -t_1 \end{pmatrix}, \quad (16)$$

$$t^{x+y} = \begin{pmatrix} -t_3 & -t_4 \\ -t_4 & -t_3 \end{pmatrix}, t^{x-y} = \begin{pmatrix} -t_3 & +t_4 \\ +t_4 & -t_3 \end{pmatrix}.$$

The interaction is:

$$H_{int} = U_0 \sum_{i\alpha} n_{i,\alpha,\uparrow} n_{i,\alpha,\downarrow} +$$

$$+ U_1 \sum_i n_{i,x} n_{i,y} + U_2 \sum_i \vec{S}_{i,x} \cdot \vec{S}_{i,y} +$$

$$+ U_3 \sum_{i,\alpha} \bar{n}_{i,\alpha,\uparrow} \bar{n}_{i,\alpha,\downarrow}, \quad (17)$$

where $\bar{n}_{i,\alpha,\sigma} = c_{i,\alpha,\sigma}^{\dagger} c_{i,\bar{\alpha},\sigma}$ and $\bar{x} = y$, $\bar{\uparrow} = \downarrow$ and $\bar{a} = a$. With this definition, $U_0 = U$, $U_1 = U' - J/2$, $U_2 = -2J$ and $U_3 = -J$. Moreover, usually $U' = U - 2J$.

This model has SU(2) symmetry if we define $S^+ = \sum_{i,\gamma} c_{i\uparrow\gamma}^{\dagger} c_{i\downarrow\gamma}$, $S^z = \sum_{i,\gamma} (c_{i\uparrow\gamma}^{\dagger} c_{i\uparrow\gamma} - c_{i\downarrow\gamma}^{\dagger} c_{i\downarrow\gamma})$, and S^2 as usual from these operators and their transpose conjugates. The sum over γ is a sum over the two orbitals, a and b or 0 and 1. In this model, the efficiency achieved by the use of the SU(2) symmetry is modest. This can be seen, for example, in Fig. 1, by comparing open circles with squares. In no case was the gain found to be larger than a factor of 1.5, and in most cases it was only about 20% to 30% depending on m and on the number of lattices sites.

However, the possibility of working with a given total spin ground state facilitates the study of the nature of ground states. For example, using the SU(2) symmetry it is easier to determine if the ground state is a singlet or a triplet. Without the help of the full spin symmetry one would have to run with various S_z target states and infer from them which one has the lowest energy.

Using the SU(2) symmetry, the CPU times for this model, which is implemented in class `FeBasedSc`, are given in Fig. 1. The model is expressed on a 2-leg ladder with parameters[15] $t_1 = 0.058$, $t_2 = 0.2196$, $t_3 = 0.20828$, and $t_4 = 0.079$. The figure shows the run with a single core and with two cores, parallelized via *pthread*s. For $m \geq 300$, CPU times are cut by almost a factor of 2, the theoretical maximum, because this model, being formulated on a ladder, has many connections, making the shared parallelization efficient.

We end this section on a technical note. In this model the real-space basis on a single site has two states that are not eigenstates of S^2 . These states are $|6\rangle \equiv c_{\uparrow a}^{\dagger} c_{\downarrow b}^{\dagger} |0\rangle$

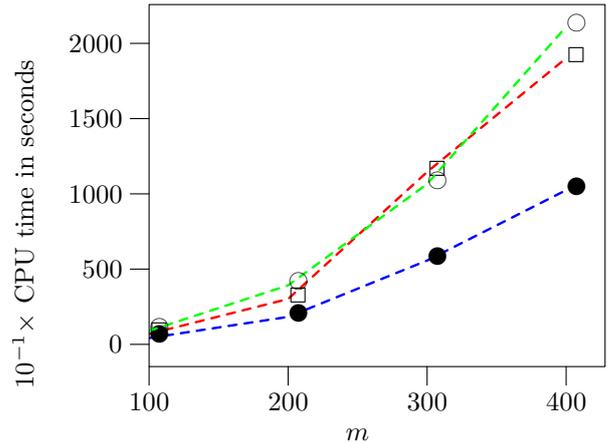


Figure 1: CPU times in seconds divided by 10, for the model given by Eqs. (15-17), running on a single core with full spin symmetry (squares), and with two cores and full spin symmetry (filled circles). The two-core run was done with shared memory parallelization via *pthread*s. For comparison, the open circles are runs with one core and without the SU(2) symmetry. All runs were carried out on a 2×4 ladder, with fixed $m = 100$ for the infinite algorithm, and with a full finite sweep with the indicated m .

and $|9\rangle \equiv c_{\uparrow b}^{\dagger} c_{\downarrow a}^{\dagger} |0\rangle$. In DMRG++, real-space basis states are coded using a binary number representation, the bit x indicates if there's an electron with internal degree of freedom, $x = \gamma + \sigma N_o$, where N_o is the number of orbitals, γ is the orbital number (0 for a and 1 for b), and σ is the spin (0 for \uparrow and 1 for \downarrow). For example, $c_{\uparrow a}^{\dagger} c_{\downarrow b}^{\dagger} |0\rangle$ has binary number 110 or 6.

States $|6\rangle$ and $|9\rangle$ are reinterpreted as $1/\sqrt{2}(|6\rangle + |9\rangle)$ and $1/\sqrt{2}(|6\rangle - |9\rangle)$, respectively. This reinterpretation occurs when calculating operators, such as $c_{\sigma\gamma}^{\dagger}$, in this real-space basis, and allows a binary number representation of states to still be used in this case, even when the original states were not eigenstates of S^2 .

5. Summary

By making use of the full spin symmetry to those models that possess it, the DMRG procedure runs faster. For the one-orbital Hubbard model on a one-dimensional lattice, the speed-up factors were approximately 4 on a 32-site lattice, and approximately 5 to 10 on a 60-site lattice. All these factors depend on m , as detailed in the tables. The speed-up factor for the two-orbital Hubbard model for iron-based superconductors (`FeBasedSc`) on a 2-leg ladder was modest, and never exceeded 1.5.

The efficiency gained by using the SU(2) symmetry is due to the smaller size of the Hamiltonian matrix blocks that need to be diagonalized. This effect is countered by the overhead imposed by performing basis transformations using the factors described in Eq. (1). However, by employing the Wigner-Eckart theorem and using reduced factors and operators, it is possible to bring down the cost of these transformations significantly. The overall effect

is the decrease in CPU times mentioned in the previous paragraph.

Additionally, shared memory parallelization was used to parallelize the calculation of Hamiltonian connections between system and environment. The success of this method depends on the model, and is most effective when there are many connections. For the FeBasedSc model running with 2 cores the speed-up almost reached the theoretical maximum of a factor of 2.

Strongly correlated electronic models for iron-based superconductors (implemented in the FeBasedSc DMRG++ class) is a topic of intense study in condensed matter. Of particular interest is the origin and mechanism of the pairing in these superconductors. The DMRG algorithm provides an accurate way of extracting information from the models in this context (for a recent paper, see, *e. g.*, Ref. [16]).

DMRG++ is a free and open source implementation of the DMRG algorithm. It emphasizes generic programming using C++ templates, friendly user-interface, and as few software dependencies as possible. DMRG++ tries to make writing new models and geometries easy and fast by using a generic DMRG engine.

6. Acknowledgments

The present code uses part of the psimag toolkit, <http://psimag.org/>. I would like to thank Luis G. G. V. Dias da Silva, I. P. McCulloch, M. S. Summers, and J. C. Xavier for helpful discussions. This work was supported by the Center for Nanophase Materials Sciences, sponsored by the Scientific User Facilities Division, Basic Energy Sciences, U.S. Department of Energy, under contract with UT-Battelle. This research used resources of the National Center for Computational Sciences, as well as the OIC at Oak Ridge National Laboratory.

A. Two states have the same triplet j , m and f , if and only if they are equal.

Let $|a\rangle$ and $|b\rangle$ be two states with the same j , m , and f . Without loss of generality we can assume that there $\exists p \geq 0$ such that $(S^+)^p|a\rangle = \eta_{p,j,m}|b\rangle$. Then, because $|a\rangle$ and $|b\rangle$ have the same S^2 and S^z eigenvalue, p has to be zero, implying that $|a\rangle = \eta_{0,j,m}|b\rangle = |b\rangle$. The reciprocal holds because a given state has unique values for j , m , and f . The uniqueness of the first two is trivial. Flavor is also unique in a given basis, since a state cannot belong to two different equivalence classes.

B. The reduced DMRG Transformation Conserves Flavor

Here we prove that $W|j, m, f\rangle$ has well defined flavor. Without loss of generality assume that $(S^+)^p|j, m, f\rangle = \eta_{p,j,m}|j, m + p, f\rangle$. Since ρ conserves j, m , then W

does too, and $W|j, m, f\rangle = \sum_{f'} W_{f,f'}^{j,m}|j, m, f'\rangle$, where $W^{j,m}$ is the matrix block of W corresponding to the good quantum numbers j, m . Then $(S^+)^p W|j, m, f\rangle = \eta_{p,j,m} \sum_{f'} W_{f,f'}^{j,m+p}|j, m + p, f'\rangle$. Since the reduced density matrix does not depend on m , then nor does W . In other words, $W^{j,m+p} = W^{j,m}$, and so $(S^+)^p W|j, m, f\rangle = \eta_{p,j,m} W|j, m + p, f\rangle$, implying that $W|j, m, f\rangle$ has well defined flavor. We also proved that S^+ and W commute, and since applying S^+ does not change flavor and W does not change j or m , then flavors can be assigned in the same way to $W|j, m, f\rangle$ as were assigned to $|j, m, f\rangle$. That flavors can be assigned without applying S^+ saves us from keeping track of it through the DMRG procedure.

C. Building and Running DMRG++

The required software to build DMRG++ is: (i) GNU C++, and (ii) the LAPACK library. This library is available for most platforms. The `configure.pl` script will ask for the `LDLFLAGS` variable to pass to the compiler/linker. If the *Linux* platform was chosen the default/suggested `LDLFLAGS` will include `-llapack`. If the *OSX* platform was chosen the default/suggested `LDLFLAGS` will include `-framework Accelerate`. For other platforms the appropriate linker flags must be given. More information on LAPACK is here: <http://netlib.org/lapack/>.

Optionally, `make` or `gmake` is needed to use the Makefile, and `perl` is only needed to run the `configure.pl` script.

To Build and run DMRG++:

```
cd src
perl configure.pl
(please answer questions regarding model, etc)
make
./dmrg input.inp
```

The perl script `configure.pl` will create the files `main.cpp`, `Makefile` and `input.inp`. This file can be used as input to run the DMRG++ program. To run the MPI code the command `mpirun ./dmrg input.inp` can be used, although the actual command will vary according to the local MPI Installation.

There is also a test suite that can be run for all standard tests:

```
cd TestSuite; ./testsuite.pl --all
```

or a specific test can be selected and run by omitting the `--all` argument in the command above. Further details can be found in the file `README` in the code.

References

- [1] S. White, Phys. Rev. Lett. 69 (1992) 2863.
- [2] G. Alvarez, The density matrix renormalization group for strongly correlated electron systems: A generic implementation, Computer Physics Communications 180 (2009) 1572.
- [3] J. Hubbard, Proc. R. Soc. London Ser. A 276 (1963) 238.
- [4] J. Hubbard, Proc. R. Soc. London Ser. A 281 (1964) 401.

- [5] J. Spalek, A. Oleś, *Physica B* 86-88 (1977) 375.
- [6] J. Spalek, *Acta Physica Polonica A* 111 (2007) 409–24.
- [7] M. Daghofer, A. Moreo, J. A. Riera, E. Arrigoni, D. Scalapino, E. Dagotto, *Phys. Rev. Lett.* 101 (2008) 237004.
- [8] I. P. McCulloch, M. Gulácsi, The non-abelian density matrix renormalization group algorithm, *Europhys. Lett.* 57 (2002) 852.
- [9] I. P. McCulloch, M. Gulácsi, *Australian Journal of Physics* 53 (2000) 597–612.
- [10] J. F. Cornwell, *Group Theory in Physics*, Academic Press, London, 1984.
- [11] Y. Kamihara, T. Watanabe, M. Hirano, H. Hosono, *J. Am Chem. Soc.* 130 (2008) 3296.
- [12] C. Wang, L. Li, S. Chi, Z. Zhu, Z. Ren, Y. Li, Y. Wang, X. Lin, Y. Luo, S. Jiang, X. Xu, G. Cao, Z. Xu, *Europhys. Lett.* 83 (2008) 67006.
- [13] T. Yildirim, *Phys. Rev. Lett.* 102 (2009) 037003.
- [14] A. Moreo, M. Daghofer, E. Dagotto, *Phys. Rev. B* 79 (2008) 104510.
- [15] J. C. Xavier, G. Alvarez, A. Moreo, E. Dagotto, in preparation (2009).
- [16] E. Berg, S. A. Kivelson, D. J. Scalapino, arXiv:0912.0277 (2009).