Phys 879 High Performance Computational Physics Final Project: Density Matrix Renormalization Group Applied to the Heisenberg Spin Chain

Jesse Simmons[1]

[1] *Department of Physics, Engineering Physics and Astronomy, Queen's University, Kingston, ON K7L 3N6, Canada*

Density-matrix renormalization group is an extremely powerful tool for studying strongly correlated lattice-like systems of arbitrary size. Though the method is somewhat constrained to low-dimension, some methods for expanding the range of applicability are gaining traction [1]. Here a finite and infinite DMRG routine is implemented for the Heisenberg spin 1/2 chain with great success. The routine is general and easily applicable to other physical systems. Attempts are made to introduce parallelization through the MPI infrastructure.

I. INTRODUCTION

Exponential scaling with degrees of freedom is often the primary impediment to performing calculations on systems of increasing size. The density-matrix renormalization group (DMRG) is a method for examining properties of large systems while keeping the calculation size manageable. Often other methods such as perturbation theory and Monte Carlo can break down in the presence of strong interactions or include strict approximations, hence the need for a method like DMRG [1]. Since its inception DMRG has proved itself as the most powerful method in studying 1D quantum lattice-like systems [2, 3].

Density-matrix renormalization group was first described in 1992 by Steven R. White as a modification of a more typical renormalization group procedure by which some starting Hamiltonian is modified by iteratively improving or increasing the system size while integrating out or truncating degrees of freedom that are deemed less important [2]. The goal is that the new "renormalized" Hamiltonian that is closer to the desired system will describe all the essential physics while not growing exponentially as is typically the case. Renormalization group was introduced as a method to study the thermodynamic or 'statistical continuum' limit for system sizes on the order of 10^{23} generally described by very large lattices [4]. The two fundamental steps in a renormalization group procedure of this variety are first the expansion of the system size (and as a result the Hilbert space), and the subsequent truncation of the Hilbert space to its original size. Originally the decimation procedure was to find a desired number of the lowest energy eigenstates of the enlarged system and form a new smaller Hamiltonian by projecting in to this basis of eigenstates. It was shown that, though intuitive, energy eigenstates do not optimally represent an interacting system due to the energy states of the original system being a poor basis for the enlarged system before truncation. Instead, White [5] showed that eigenstates of the density matrix for the enlarged system provide the optimal method of reducing the Hilbert space.

II. THEORY

A. DMRG Algorithm

As stated above, the essential features of the DMRG procedure are to, starting with a relatively small system, expand the system size while truncating the Hilbert space and retain only the most important states needed to describe low energy system. The starting point of the conventional DMRG theory begins with sites on a lattice and blocks that contains a number of these sites. We consider a Hamiltonian H_A for a block A containing l sites each with D degrees of freedom consisting of only terms for each site within the block. Though the total size of the full Hilbert space is therefore D^l, it is common to associate an m-dimensional basis to the block that in general can be smaller than the full Hilbert space [1]. The block is then grown by adding a site to form an expanded system H_A^e. If H_A is described by the basis $\{|a_i\rangle_A\}$ and the site by $|\sigma_j\rangle_A$ $i = 1, ..D$ then the new basis is formed

$$|\phi_k\rangle_A = |a_i\rangle_A \otimes |\sigma_j\rangle_A \,, \tag{1}$$

which has a size $m \times D$. Next the expanded block A system is embedded in a larger environment or bath containing a similar block A$'$ with l' and m', and also with an additional site having D degrees of freedom to form a *superblock*. The A$'$ block mimics the A block in that the now two additional sites are between the blocks forming a block-site-site-block structure [1]. Fig. 1 illustrates the expansion and superblock procedure. Originally (non-DMRG) the expansion was done by simply doubling the size of a block A to form AA, finding the lowest m eigenstates of A, and repeating with AA $\rightarrow$ A. White and Noack showed this failed on the basis that that it generates inappropriate boundary conditions [6]. Part of the reasoning was that if one considers a simple particle in a box block A, then the ground state of A will have nodes where the ground state of AA should be a maximum. White and Noack showed that instead of this exponentially increasing system size, the linear recipe of adding a single site and embedding the system in a larger environment more appropriately deals with the boundaries, especially when there are interactions between neighbouring sites [6].

A general state of the superblock is now written

$$|\psi\rangle = \sum_{i,j}^{m\times D, m'\times D} \psi_{ij} \, |\phi_i\rangle_A \otimes |\phi'_j\rangle_{A'} \,. \tag{2}$$

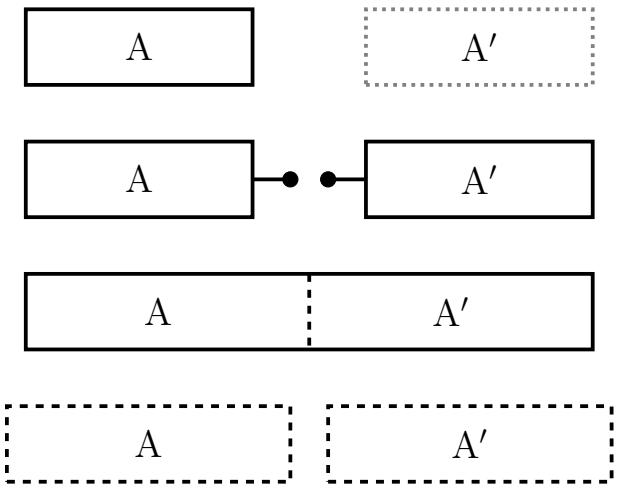

Figure 1. Typical DMRG expansion procedure. Note that in the finite system case, block A′ may not be the same size as block A

The dimension of the full superblock Hamiltonian is now D^2m^2. Typically m is somewhere in the range of $O(100) - O(1000)$, so the Hamiltonian may approach a size where it is too large to diagonalize directly since this is an expensive procedure that requires access to the full matrix explicitly [1]. The ground (and any other target) state of the superblock is calculated via Lanczos or some other iterative method. The question is now which states in the system A given the contribute the most to the larger superblock AA' system and environment combination that will eventually approach the full system size [1]. We do this by considering the reduce density operator ρ_A of the original system and site combination

$$\begin{aligned} \rho_A &= \sum_j \langle \phi'_{A'} | \, \rho \, | \phi'_{A'} \rangle = Tr_{A'} \rho, \\ (\rho_A)_{i,i'} &= \sum_j^{m' \times D} \psi_{ij} \psi_{i'j}, \end{aligned} \tag{3}$$

keeping as a reduced basis the eigenstates $|\omega_\alpha\rangle$ corresponding to the largest m eigenvalues of ρ_A. The new truncated Hamiltonian of the system A is formed by projecting the original Hamiltonian expanded by a single site H_A^e into the basis $|\omega_\alpha\rangle$. The process is then repeated, growing the system and environment size by 1 but maintaining a renormalized Hilbert space. The algorithms for the finite and infinite (approaching thermodynamic) vary only in the formation of the superblock. In the infinite method the environment block A′ is a mirror of the expanded block A, both have the same size. In the finite algorithm, once the superblock reaches the desired full system size L so that both A and A′ have L/2 sites, A continues to grow by a single site whereas A′ shrinks by a site until it contains only one. The process is then reversed, A′ grows a the expense of A, and reversed again, sweeping over L until the desired convergence is reached.

B. The Heisenberg Spin Chain

Most descriptions of DMRG use the Heisenberg spin chain with $S = 1/2$ or 1 as a theoretical example due to its simple nature [1, 2, 5, 7]. This system serves as an excellent starting point, as the ground state energy per site for the antiferromagnetic Hamiltonian (negative of the original) is known in the $S = 1/2$ case for the limit of an infinite number of sites [8]. This Heisenberg chain is one of the few strongly correlated infinite systems with an exact solution and is therefore used as a common benchmark [1]. The antiferromagnetic Hamiltonian with coupling equal to 1 in the presence of no external field is[9]

$$\hat{H} = \sum_i^{L-1} \hat{S}_i \cdot \hat{S}_{i+1}. \tag{4}$$

Where $\hat{S} = \hat{S}^x + \hat{S}^y + \hat{S}^z$ are the typical Pauli matrices scaled by the spin quantity and $\hbar = 1$. To eliminate complex values set $\hat{S} + \pm = \hat{S}^x \pm i\hat{S}^y$ so that Eq. (4) becomes

$$\hat{H} = \frac{1}{2}(\hat{S}_i^+ \hat{S}_j^- + \hat{S}_i^- \hat{S}_j^+) + \hat{S}_i^z \hat{S}_j^z. \tag{5}$$

The solution of Eq. (5) is known only in the spin 1/2 case for the ground state energy per site as L $\to \infty$ is using the Bethe ansatz [8] yielding

$$E = -\ln 2 + \frac{1}{4}. \tag{6}$$

For the finite lattice or S=1 situations the ground state solution must be found by diagonalization of the full Hamiltonian to find an exact solution or approximated by another method numerically.

C. Lanczos Iteration

The most computationally expensive portion of the algorithm is often finding the ground state of the superblock, so this is usually performed with a Lanczos or other iterative method that relies on matrix-vector products rather than a full diagonalization of the superblock Hamiltonian [1]. The basic Lanczos algorithm for finding the eigenvalues and vectors of a matrix A consists of repeatedly applying A to a vector $\vec{v}$ at each step while orthogonalizing the product against the previous two vectors to form a basis. Each step iteratively grows a square, tridiagonal matrix T such that $T = V^T AV$. If sufficiently large, among the eigenvalues of T are those of A, and since T can be much smaller than A and has a simple structure, T is diagonalized instead of A. The basic algorithm used in this report is defined as [10]

$$
\begin{aligned}
&\text{Start with a random, normalized vector } \vec{v}_i, \\
&\text{for i in range(m) :} \\
&\vec{w}_i = A\vec{v}_i, \\
&\alpha_i = \vec{w}_i{}^T \cdot \vec{w}_i \\
&\vec{w}_i = \vec{w}_i - \alpha\vec{w}_i - \beta_i \vec{w_{i-1}} \\
&\beta_{i+1} = ||\vec{w}_i|| \\
&\vec{v}_{i+1} = \vec{w}_i/\beta_{i+1} \\
&T_{ii} = \alpha_i, \quad T_{i,i-1} = \beta_i, \quad T_{i_1,i} = \beta_i
\end{aligned}
\tag{7}
$$

The eigenvectors of A may be found by simply applying the Lanczos vector matrix V to the corresponding eigenvectors of T.

III. COMPUTATION DETAILS

Implementing Eq. (5) computationally was quite challenging. For the Heisenberg spin chain the operators are written as matrices in the $|\uparrow\rangle, |\downarrow\rangle$ basis. Therefore when adding a site, the operators in the new Hilbert space must be formed from a Kronecker product of the block and site combination. Suppressing operator notion, to add a single site D with operators S^z, S^+, and S^-, to a block B, Eq. (5) becomes [7]

$$H_{sys} = H_B \otimes I_D + \frac{1}{2}(S_i^+ \otimes S_j^- + S_i^- \otimes S_j^+) + S_i^z \otimes S_j^z. \tag{8}$$

Additionally, for the system to be paired with an environment block to form the superblock, the spin operators for the enlarged system in the basis of the new enlarged block are needed,

$$
\begin{aligned}
S^z_{sys} &= I_B \otimes S^z, \\
S^+_{sys} &= I_B \otimes S^+, \\
S^-_{sys} &= I_B \otimes S^-.
\end{aligned}
\tag{9}
$$

Next to form the superblock the individual system and environment Hamiltonians must be transformed to the new superblock basis, as well as the interaction between the two at the joining site using the operators in Eq. (9),

$$
\begin{aligned}
H_{superblock} =& H_{sys} \otimes I_{env} + I_{sys} \otimes H_{env} + \\
&\frac{1}{2}(S^+_{sys} \otimes S^-_{env} + S^-_{sys} \otimes S^+_{env}) + S^z_{sys} \otimes S^z_{env}.
\end{aligned}
\tag{10}
$$

It took a sizeable amount of time to write the above steps into code and was by far the most challenging portion of the DMRG algorithm. Performing a step explicitly in the $|\uparrow\rangle, |\downarrow\rangle$ basis offered a great deal of clarity and was ultimately necessary to fully comprehend the algorithm and perform the order of operations correctly. Lastly, the operators must be transformed into the (once again) new basis of the eigenstates of the reduced density operator. It was found that Python dictionaries were vital for being able to reference each of these arrays efficiently (and generally) for any lattice type, being able to loop over dictionary elements clearly by name rather than a multitude of individual arrays or complicated indexing.

A. Performing a Single DMRG Step

Beginning with a system and environment block that are defined in the preamble to grow the lattice the `dmrg_step` function is called. Each block is individually enlarged by a single site by calling the `enlarge` function, modifying the Hamiltonian and the operators at the sites connecting to the superblock. The function `interaction` takes care of the Eq. (5) influence for both a block and site connection to enlarge a block or a system and environment connection to combine a superblock. The superblock is then formed and its ground state energy and wavefunction determined either by the most basic python eigenvalue routine `numpy.linalg.eigh`, the more efficient `scipy.sparse.linalg.eigsh`, which is a wrapper ARPACK [11] Lanczos routine using FORTRAN, or by a Lanczos function written as an addition to the main DMRG code using the `numpy.linalg.eigh` routine to find eigenvalues of the small tridiagonal T matrix. The reduced density matrix is then formed and the eigenvectors corresponding to the largest eigenvalues are extracted to form a transformation matrix. The number of eigenvectors retained determines the truncation of the Hilbert space and is defined prior to the calculation. Finally all operators in the enlarged system block are transformed and returned.

B. Infinite and Finite Algorithms

In the infinite system algorithm the process is straightforward, the Hilbert space grows until the block basis reaches a predefined size at which point he system size grows while maintaining the calculation size. The returned system block is set equal to the environment block before a step is called again.

The finite system however is significantly more complicated. During the growth phase (that mimics the infinite algorithm) to a system size of L, the enlarged blocks are recorded for later use. Once both the enlarged system and environment blocks are both $L/2$, to continue the $L/2$ system block is enlarged to $L/2+1$ and recorded while for the environment block a $L/2-2$ block is read from memory and enlarged to $L/2-1$. This is repeated until the environment block, before enlargement, reaches a size of 1, which completes a sweep, then the system becomes the environment and vise-versa, and the process repeats with a $L-2$ block read from memory.

IV. RESULTS

A. Infinite Convergence

Here the ability of the infinite DMRG algorithm is tested in it's ability to converge to the exact solution of $E = -\ln 2 + 1/4 = -0.443147$ for the S=1/2 chain. The

convergence is tested for truncated basis sizes of m=10, 20, and 30, which for the Heisenberg spin chain reach maximum superblock sizes of 400, 1600, and 3600 respectively (m^2d^2). The infinite system algorithm is run until lattice lengths of 10, 50, 100, 500, 1000, 2000, and 5000 are reached, which appears to sufficient to reach a moderate level of convergence. Results are shown in Fig. 2. It is

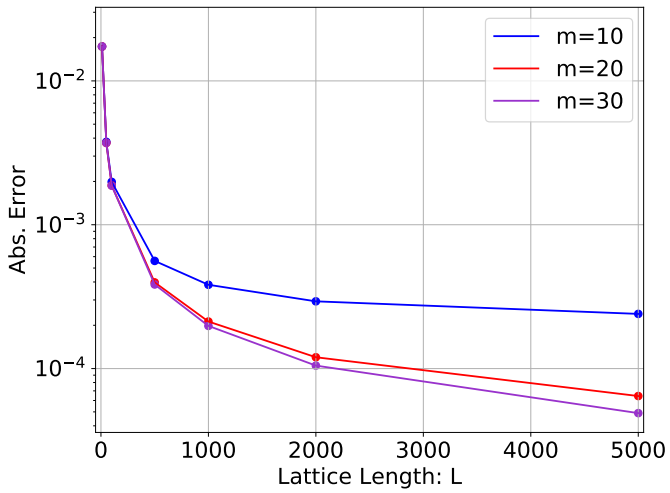

Figure 2. Absolute error in infinite system DMRG compared to exact solution given in Eq. (6) for a Heisenberg S=1/2 chain.

evident that if the truncated basis size is sufficiently large a decent level of convergence may be obtained for a system size that far out of the realm of possibility for a full diagonalization calculation.

For the largest $L = 5000$ lattices, the run times using the `eigsh` method for $m = 10, 20$, and 30 are 49.5s, 780.6s, and 4625.4s respectively. The corresponding times using the Lanczos function made for this work are 65.9s, 725.8s, and 3708.9s, clearly becoming more efficient for larger calculation sizes than the native Python functions even when they are a FORTRAN wrapper.

B. Finite Convergence

In this section the goal is to demonstrate the ability of the finite DMRG algorithm to calculate a ground state energy for which there is no analytic solution. For a S=1/2 lattice size of L=15 the finite DMRG routine is swept until the energy changes by less than 1×10^{-8} at which point the calculation will be considered converged. The code is run for truncation bases of m=10, 20, 30, 40, and 50. The reference energy is calculated by generating the full L=15 Hamiltonian in the complete spin up/down bases and finding the ground state energy with the same eigenvalue routine as DMRG. The time for the exact calculation takes approximately 784.5s. As each additional site of dimension d quadruples the number of elements in the matrix and eigenvalue routines are affected by the matrix size, this time is expected to grow exponentially with additional sites. Although L¿15 is still possible it is already approaching the limit of the typical RAM. A $2^{15} \times 2^{15}$ double precision matrix is approximately 8.5 GB of memory. The absolute errors of the DMRG calculation as well as the run times are shown in Table I.

m	Abs. Error	Time (s)
10	4.88×10^{-5}	0.14
20	8.09×10^{-7}	1.53
30	1.19×10^{-7}	6.51
40	9.45×10^{-9}	14.37
50	5.47×10^{-10}	29.00

Table I. Absolute error in finite system DMRG for a L=15, S=1/2 finite Heisenberg chain compared to an exact calculation in the complete basis. Truncated DMRG basis sizes from 10-50 are used. The run-time for the exact calculation is 784.5s.

C. Optimization and Parallelization

As finding the ground state of the superblock is often one of the most computationally expensive portions of the calculation, parallelization of the Lanczos algorithm was one of the first optimizations attempted. For the Lanczos function, the matrix-vector products in the first step of Eq. (7) are performed by calculating segments of the product on each process which are then gathered and the results to broadcast. Unfortunately since the norm of $\vec{w}_i$ is required for each iteration the expensive communication is unavoidable. Although it completely functional, in a lot of cases communication overhead makes the parallel version of the Lanczos function not worth calling when compared to the regular version even though the matrix products themselves are significantly faster.

After further investigation it was found that another expensive portion of the program for larger truncation sizes was the extensive amount of Kronecker products. A similar communication issue was found for computing the Kronecker product in parallel in the function `my_kron`. To parallelize the product $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, blocks of $\mathbf{C}$ were calculated on different processes by multiplying an element of $\mathbf{A}$ by the matrix $\mathbf{B}$. After finishing a block each process moves on to the next empty segment of $\mathbf{C}$ until completion when the $\mathbf{C}$ matrices residing on each process are added together using the MPI `Reduce` function. It was hoped that this would perform a significantly improved Kronecker product since each process is continuously busy until the last few blocks of $\mathbf{C}$ without placing any restrictions on the number of processes (no need for multiples of 2). Although each process finished the total task in a fraction of the time it would typically take, reduction to the root process and subsequent broadcasting to all others once again caused too much communication overheard to be faster than NumPy's `kron` function. The `kron` function itself appears to be quite efficient and upon investigation uses a combination of the `reshape, outer,` and `concatenate` methods which might be faster in terms of memory than sweeping through the matrix as done in `my_kron`.

V. CONCLUSION

In this report a general and efficient 1D DMRG code for the finite and infinite algorithms was created and tested using the traditional Heisenberg spin chain to find the ground state energy. For S=1/2 the infinite case an analytic solution exists [8] and for the finite case the reference energy is determined by computing the full Hamiltonian in the complete spin up/down basis. Both algorithms proved to be very efficient, and the power of DMRG is demonstrated particularly in the finite lattice case where an accuracy of better than 1×10^{-6} can be obtained in less than 1/100th of the time of an exact calculation. The most computationally expensive portions of the code are finding the ground state energy of the superblock and calculating large Kronecker products for the superblock. Both situations were made to run in parallel and, although the arithmetic speed itself was greatly enhanced, MPI communication overhead greatly hindered utility. For large calculation sizes however, the non-parallel Lanczos function written in this work was comparatively fast to one the best available SciPy functions.

The only portions of the code specific to the Heisenberg spin chain were small interaction and enlargement functions having to do with the Hamiltonian as well as some operator array setup, so it should be somewhat straightforward to apply the routine to other systems of interest. Work was also started but not completed on incorporating a tight-binding model system which can also have exact solutions [12] and lays the groundwork for more physical systems of interest like the Hubbard model [13]. Ultimately the DMRG code works extremely well and, though attempts at parallelization were less successful than hoped, it is conceivable that they can become more efficient with some adjustments.

References

[1] Ulrich Schollwöck, "The density-matrix renormalization group in the age of matrix product states," Annals of Physics **326**, 96 – 192 (2011), january 2011 Special Issue.

[2] U. Schollwöck, "The density-matrix renormalization group," Rev. Mod. Phys. **77**, 259–315 (2005).

[3] Karen A. Hallberg, "New trends in density matrix renormalization," Advances in Physics **55**, 477–526 (2006).

[4] Kenneth G. Wilson, "The renormalization group: Critical phenomena and the kondo problem," Rev. Mod. Phys. **47**, 773–840 (1975).

[5] Steven R. White, "Density matrix formulation for quantum renormalization groups," Phys. Rev. Lett. **69**, 2863–2866 (1992).

[6] S. R. White and R. M. Noack, "Real-space quantum renormalization groups," Phys. Rev. Lett. **68**, 3487–3490 (1992).

[7] A. Malvezzi, "An introduction to numerical methods in low-dimensional quantum systems," Brazilian Journal of Physics **33** (2003), 10.1590/S0103-97332003000100004.

[8] H. Bethe, "Zur theorie der metalle," Z. Physik **71**, 205–226 (1931).

[9] Michael Karbach, Kun Hu, and Gerhard Muller, "Introduction to the bethe ansatz ii," Computers in Physics **12** (1998), 10.1063/1.168740.

[10] Yousef Saad, *Numerical Methods for Large Eigenvalue Problems* (Society for Industrial and Applied Mathematics, 2011).

[11] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide* (Society for Industrial and Applied Mathematics, 1998).

[12] R. J. Baxter, "Exactly solved models in statistical mechanics," in *Integrable Systems in Statistical Mechanics*, pp. 5–63.

[13] J. Hubbard and Brian Hilton Flowers, "Electron correlations in narrow energy bands," Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **276**, 238–257 (1963).