A Scalable Matrix-Free Iterative Eigensolver for Studying Many-Body Localization

Roel Van Beeumen
rvanbeeumen@lbl.gov
Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA, United States

Gregory D. Kahanamoku–Meyer
gregory.meyer@berkeley.edu
Department of Physics
University of California at Berkeley
Berkeley, CA, United States

Norman Y. Yao
norman.yao@berkeley.edu
Department of Physics
University of California at Berkeley
Berkeley, CA, United States

Chao Yang
cyang@lbl.gov
Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA, United States

ABSTRACT

We present a scalable and matrix-free eigensolver for studying two-level quantum spin chain models with nearest-neighbor $XX+YY$ interactions plus Z terms. In particular, we focus on the Heisenberg interaction plus random on-site fields, a model that is commonly used to study the many-body localization (MBL) transition. This type of problem is computationally challenging because the vector space dimension grows exponentially with the physical system size, and the solve must be iterated many times to average over different configurations of the random disorder. For each eigenvalue problem, eigenvalues from different regions of the spectrum and their corresponding eigenvectors need to be computed. Traditionally, the interior eigenstates for a single eigenvalue problem are computed via the shift-and-invert Lanczos algorithm. Due to the extremely high memory footprint of the LU factorizations, this technique is not well suited for large number of spins L, e.g., one needs thousands of compute nodes on modern high performance computing infrastructures to go beyond $L = 24$. The new matrix-free approach, proposed in this paper, does not suffer from this memory bottleneck and even allows for simulating spin chains up to $L = 24$ spins on a single compute node. We discuss the OpenMP and hybrid MPI–OpenMP implementations of matrix-free block matrix-vector operations that are the key components of the new approach. The efficiency and effectiveness of the proposed algorithm is demonstrated by computing eigenstates in a massively parallel fashion, and analyzing their entanglement entropy to gain insight into the MBL transition.

KEYWORDS

quantum many-body problem, many-body localization, eigenvalue problem, matrix-free eigensolver, LOBPCG method, preconditioner, scalability

HPCAsia2020, January 15–17, 2020, Fukuoka, Japan

ACM ISBN 978-1-4503-7236-7/20/01…$15.00
https://doi.org/10.1145/3368474.3368497

ACM Reference Format:
Roel Van Beeumen, Gregory D. Kahanamoku–Meyer, Norman Y. Yao, and Chao Yang. 2020. A Scalable Matrix-Free Iterative Eigensolver for Studying Many-Body Localization. In *International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia2020), January 15–17, 2020, Fukuoka, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3368474.3368497

1 INTRODUCTION

A fundamental assumption in the traditional theory of statistical mechanics is that an isolated system will in general reach an equilibrium state, or *thermalize*. As early as the mid-20$^{\text{th}}$ century, Anderson demonstrated that a single particle moving in a highly disordered landscape can violate this assumption [1]. While surprising, that result does not readily extend to many-particle systems that exhibit strong interactions between the constitutent particles. The question of whether a similar effect could manifest in a strongly-interacting many-body system remained open for decades. This elusive phenomenon has been termed "many-body localization" (MBL).

Recently, advances in both high performance computing and experimental control of individual quantum particles have begun to yield insight into MBL. Both experimental [3, 4, 13, 15, 21, 22] and numerical [2, 5, 11, 14, 18] results have shown evidence of localization in small strongly-interacting multiparticle systems of 10-20 spins. Unfortunately, extrapolating results from these small system sizes to the infinitely-large thermodynamic limit has proven difficult. This lack of clarity has inspired a vigorous debate in the community about precisely what can be learned from small-size results. For example, it has been proposed that certain features do not actually exist at infinite system size [7], and even that MBL itself is only a finite-size effect [23]!

The primary goal of most studies is to identify and characterize a *localization transition*. In the thermodynamic limit, as the strength of the system's disorder increases, theory predicts a sharp, sudden change from a thermal to a localized state. Unfortunately, in the small systems available for study, that sharp transition turns into a smooth *crossover*, leading to the confusion about what constitutes the transition itself. Numerical evidence suggests that the transition

sharpens rapidly as system size increases, so accessing as large systems as possible is imperative for investigating MBL.

In pursuit of that goal, Luitz et al. used large-scale numerical linear algebra to show a localization transition for system sizes up to $L = 22$ [14], and in a following paper extracted useful data up to $L = 24$ [19]. In order to compute interior eigenstates for the MBL problem, the shift-and-invert Lanczos algorithm was used in combination with sparse direct solvers for solving the linear systems. One of the major disadvantages of this technique is that constructing the LU factorizations becomes extremely memory demanding, due to the so called fill in, for large number of spins L. Table 1 shows that the memory footprint of the LU factorization computed via STRUMPACK [9] grows rapidly as function of L. See also [19]. Hence, thousands of nodes on modern high performance computing infrastructures are needed to go beyond $L = 24$.

Table 1: Total memory footprint as a function of the spin chain length L for LU factorizations, computed via STRUMPACK, and the new matrix-free LOBPCG algorithm, with block size 64. The problem size is given by n.

L	n	STRUMPACK	LOBPCG(64)
16	12,870	66 MB	8 MB
18	48,620	691 MB	31 MB
20	184,756	8 GB	118 MB
22	705,432	92 GB	451 MB
24	2,704,156	1 TB	2 GB
26	10,400,600	15 TB	7 GB

In this paper, we introduce an new approach based on the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm to overcome the memory bottleneck that the shift-and-invert Lanczos algorithm faces. As shown in Table 1, we are able to reduce the memory footprint by several orders of magnitude, e.g., from 15 TB to only 7 GB for $L = 26$. This new approach will enable us to simulate spin chains on a single node, even up to $L = 24$. For larger spin chains we only require a few nodes.

The paper is organized as follows. We first review the Heisenberg spin model and MBL metrics in Section 2. Next, the LOBPCG eigensolver with efficient matrix-free block matrix-vector operations is discussed in Section 3. Then in Section 4, we illustrate the new matrix-free LOBPCG eigensolver for Heisenberg spin chains of sizes up to $L = 26$. Finally, the main conclusions are formulated in Section 5.

2 PROBLEM FORMULATION

In this section we briefly review the properties of the spin chain model that most frequently is studied by numerical simulations of MBL.

2.1 Heisenberg Spin Model

We consider the nearest-neighbor interacting Heisenberg spin model with random on-site fields:

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j + \sum_i h_i S_i^z, \tag{1}$$

where the angle brackets denote nearest-neighbor i and j, h_i is sampled from a uniform distribution $[-w, w]$ with $w \in \mathbb{R}_0^+$, and

$$\vec{S}_i \cdot \vec{S}_j = S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z,$$

where $S_i^\alpha = \frac{1}{2}\sigma_i^\alpha$, with σ_i^α the Pauli matrices operating on lattice site i and $\alpha \in \{x, y, z\}$. The parameter w is called the *disorder strength*, and is responsible for inducing the MBL transition. The values h_i are sampled randomly each time the Hamiltonian is instantiated, and the relevant physics lies in the statistical behavior of the set of all such Hamiltonians. The individual Hamiltonians H with independently sampled h_i are called *disorder realizations*.

Note that in (1) each term of each sum has an implied tensor product with the identity on all the sites not explicitly written. Consequently, the Hamiltonian for L spins is a symmetric matrix of dimension $N = 2^L$ and exhibits the following tensor product structure

$$H = \sum_{i=1}^{L-1} I \otimes \cdots \otimes I \otimes H_{i,i+1} \otimes I \otimes \cdots \otimes I + \sum_{i=1}^{L} I \otimes \cdots \otimes I \otimes h_i S_i^z \otimes I \otimes \cdots \otimes I,$$

where $H_{i,i+1} = S_i^x S_{i+1}^x + S_i^y S_{i+1}^y + S_i^z S_{i+1}^z$ is a 4-by-4 real matrix and I is the 2-by-2 identity matrix. Remark that by definition, all matrices $H_{i,i+1}$ are the same and independent of the site i. For our experiments, we use open boundary conditions, meaning that the nearest-neighbor terms do not wrap around at the end of the spin chain. Open boundary conditions can be considered to yield a larger *effective* system size because of the reduced connectivity.

The state of each spin is described by a vector in $\mathbb{C}^2$, and the configuration of the entire L-spin system can be described by a vector on the tensor product space $(\mathbb{C}^2)^{\otimes L}$. In this specific case, however, the Hamiltonian's matrix elements happen to all be real, so we do not include an imaginary part in any of our computations. Furthermore, our Hamiltonian commutes with the total magnetization in the z direction, $S^z = \sum_{i=1}^{L} S_i^z$. Thus it can be block-diagonalized in sectors characterized by $S^z \in [-L/2, -L/2+1, \ldots, L/2-1, L/2]$. The vector space corresponding to each sector has dimension $n = \binom{L}{S^z + L/2}$ such that the largest sector's dimension is $n = \frac{L!}{(L/2)!(L/2)!}$, and this corresponds to the actual dimension of the matrices on which we operate, see Table 1. While these subspaces are smaller than the full space, their size still grows exponentially with the number of spins L. Thus, the problem becomes difficult rapidly as L increases. Furthermore, the density of eigenvalues in the middle of the spectrum increases exponentially with L. Thus the tolerance used to solve for these internal eigenvalues must be made tighter rapidly as L increases.

2.2 Metrics for Localization

With the problem's matrix clearly defined, we now need a way of quantifying localization from the eigenvalues and eigenvectors. There are multiple quantities that can be used for this purpose. We focus on two here: one based on the eigenvalues, and one on the eigenvectors. The eigenvalue-based method (adjacent gap ratio) has been used in multiple previous works [5, 11, 16, 23], but suffers from large statistical noise and thus requires many samples to be

usable. To reduce the number of samples required, we focus on the eigenvector-based method in our experiments.

2.2.1 Adjacent Gap Ratio. Random matrix theory informs us that the statistical distribution of eigenvalues will differ between localizing and thermalizing Hamiltonians [16]. In particular, we expect eigenvalues of a thermal Hamiltonian to *repel* each other, i.e., hybridization of eigenvectors prevents them from generally coming too close to one another. The eigenvalues of a localized Hamiltonian should not display this behavior: we expect them to be Poisson distributed. Therefore, we can measure localization by comparing the relative size of gaps between the eigenvalues. Thermal Hamiltonians will generally have more consistently sized gaps due to level repulsion.

The *adjacent gap ratio* is defined as follows

$$r_i = \frac{\min(\Delta_i, \Delta_{i+1})}{\max(\Delta_i, \Delta_{i+1})}, \qquad \Delta_i = \lambda_i - \lambda_{i-1},$$

where the eigenvalues λ_i are sorted in increasing order. Quantitatively, random matrix theory can inform the precise values we expect in the two cases, averaged over many pairs of neighboring eigenvalues. In the thermal case, we expect $\langle r \rangle \sim 0.53$, while for localizing Hamiltonians we expect $\langle r \rangle \sim 0.39$ [16].

2.2.2 Eigenstate Entanglement Entropy. The eigenvectors of the Hamiltonian can also help inform us about localization. In a thermal system, we expect quantum entanglement to be widespread, while in a localized system, the entanglement is not expected to be extensive. This idea can be quantified by choosing a *cut* which divides the spin chain into two pieces, and measuring the entanglement across it. In practice, this entanglement is measured by removing one of the two pieces (by computing a partial trace), and then measuring the increase in entropy due to its removal.

Mathematically, for an eigenvector x, the entanglement entropy between two subsystems A and B can be computed as follows. Define $\rho \equiv xx^\top$ as the *density matrix* corresponding to the state x, represented as a column vector. Now let $\rho_A \equiv \mathrm{Tr}_B[\rho]$ be the density matrix of subsystem A, where Tr_B is the partial trace over sites in subsystem B. The entanglement entropy is then

$$S_{AB} = -\mathrm{Tr}\left[\rho_A \ln \rho_A\right].$$

Numerically, this quantity is generally computed in the following way: (i) compute ρ_A directly from x, (ii) compute the eigenvalues λ_i of ρ_A, and (iii) compute $S_{AB} = -\sum_i \lambda_i \log \lambda_i$. Note that in the first step, we do not hold ρ itself at any point since it is a dense matrix of dimension $n \times n$, and thus is not feasible to store in memory. Fortunately, it is not hard to compute the partial trace directly from x itself.

In this paper, we focus on the case in which we cut exactly in the middle of the spin chain, such that subsystems A and B are the left and right halves of the system. In this case, for eigenvectors corresponding to eigenvalues near 0, we expect the thermal case to have entanglement entropy [17]

$$S_{L/2} = \frac{L \ln 2}{2} - 0.5 \tag{2}$$

In the localized case the entanglement entropy will not scale with L, but instead will attain some constant value. For finite system sizes, we simply expect the entanglement entropy to decrease from the above value as the system becomes more localized.

Not only the value of the entropy changes during the localization transition: the statistics change as well. When compared across disorder realizations, the thermal entanglement entropy should consistently be the value in Equation (2), and thus have small variance. During the transition, however, we expect the entanglement entropy to depend strongly on the specific disorder realization and thus the statistic will have a large variance. Empirically, examining the *variance* of the entanglement entropy is one of the best ways to identify the localization transition.

2.3 Multiple Levels of Concurrency

The MBL study allows for at least 4 levels of concurrency. The first level corresponds to the need of averaging over (many) different and independently sampled *disorder realizations* in order to obtain relevant statistical behavior. Since the *disorder strength* is responsible for inducing the MBL transition, we also have to vary the disorder strength, giving rise to the second level of concurrency. The third level corresponds to the *eigenvalue chunks*, i.e., for each (large) eigenvalue problem, originating from one disorder realization and a particular disorder strength, we have to compute eigenvalues from different regions of the spectrum and their corresponding eigenvectors.

All previous levels of concurrency are completely independent and can be implemented in a massively parallel fashion by making use of iterative eigensolvers. The next level of parallelism takes place within these eigensolvers. Although most iterative eigensolvers follow a rather sequential procedure, each of the different steps within one iteration can be implemented in parallel.

3 MATRIX-FREE LOBPCG EIGENSOLVER

The Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) algorithm [8, 12] is a widely used eigensolver for computing the smallest or largest eigenvalues and corresponding eigenvectors of large-scale symmetric matrices. Key features of the LOBPCG algorithm are: (i) It is matrix-free, i.e., the solver does not require storing the coefficient matrix explicitly. It access the matrix by only evaluating matrix-vector products; (ii) It is a block method, which allows for efficient matrix-matrix operations on modern computing architectures; (iii) It can take advantage of preconditioning, in contrast to, for example, the Lanczos algorithm.

In Section 3.1 we review the LOBPCG algorithm. Next, we discuss in Section 3.2 how the LOBPCG algorithm can be modified in order to compute interior eigenvalues and its corresponding eigenvectors. Finally, we explain in Section 3.3 how the (block) matrix-vector products can be efficiently implemented in parallel, both in OpenMP and MPI.

3.1 LOBPCG Eigensolver

Let $H \in \mathbb{R}^{n \times n}$ be a symmetric matrix and denote its eigenvalues and corresponding eigenvectors by $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and $x_1, x_2, \ldots, x_n$, respectively. Then the diagonal matrix of the first $k \leq n$ eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$ and the rectangular tall-skiny matrix of corresponding eigenvectors $X = [x_1, x_2, \ldots, x_k]$

satisfy the following eigenvalue problem

$$HX = X\Lambda$$

and X is the solution to the trace minimization problem

$$\min_{X^\top X = I} \text{trace}\left(X^\top H X\right). \tag{3}$$

A similar trace maximization property exists for the eigenvectors corresponding to the k largest eigenvalues of H.

The basic idea of the LOBPCG method introduced by Knyazev [12] is to solve this trace optimization problem only locally in every iteration, in order to converge to the smallest (or largest) eigenvalues and corresponding eigenvectors. This yields the following updating formula

$$X_{i+1} = \arg_{X \in \mathcal{Z}} \min_{X^\top X = I} \text{trace}\left(X^\top H X\right),$$

where

$$\mathcal{Z} = \text{span}\left\{W_i, X_i, X_{i-1}\right\},$$

with X_i and X_{i-1} the current and previous iterates, respectively, and W_i the preconditioned residual

$$W_i = K^{-1}\left(HX_i - X_i\Theta_i\right),$$

with K any preconditioner and $\Theta_i = X_i^\top H X_i \in \mathbb{R}^{k\times k}$. Note that W_i corresponds to the preconditioned gradient of the Lagrangian

$$\mathcal{L}(X,\Lambda) = \frac{1}{2}\text{trace}\left(X^\top H X\right) - \frac{1}{2}\text{trace}\left(X^\top X\Lambda - \Lambda\right)$$

associated to (3) and evaluated at (X_i, Θ_i).

The approximations to the smallest k eigenvalues and eigenvectors, so called Ritz pairs $(\tilde{\lambda}_j, \tilde{x}_j)$, can numerically be obtained from the Rayleigh–Ritz method, i.e.,

$$\Theta_i V = V\tilde{\Lambda},$$

where $\tilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_k)$ is a diagonal matrix containing the Ritz values on its diagonal and $V = [v_1, v_2, \ldots, v_k]$. The corresponding Ritz vectors are given by $\tilde{x}_j = X_i v_j$, for $j = 1, 2, \ldots, k$.

A basic version of the LOBPCG method, given in Algorithm 1, is relatively easy to implement, however, it can suffer from numerical instability if not implemented carefully. Therefore we used the robust variant, introduced in [8], in all our experiments.

Algorithm 1: Basic LOBPCG algorithm

Input: number of eigenpairs k and block size $b \geq k$
$X_0 \in \mathbb{R}^{n\times b}$: matrix of starting vectors with $X_0^\top X_0 = I$
Output: $X \in \mathbb{R}^{n\times k}$: matrix of approximate eigenvectors
$\Lambda \in \mathbb{R}^{k\times k}$: diagonal matrix of approx. eigenvalues

1 Residual: $R = HX_0 - X_0(X_0^\top H X_0)$.
2 Initialize: $X = X_0$, $P = []$, and $n_c = 0$.
while $n_c < k$ **do**
3 Apply preconditioner: $W = K^{-1}R$.
4 Subspace: $Z = [W, X, P]$.
5 Rayleigh–Ritz:

$$(Z^\top H Z)\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}\Lambda$$

6 Update: $X \leftarrow WV_1 + XV_2 + PV_3$ and $P \leftarrow WV_1 + PV_3$.
7 Residual: $R = HX - X\Lambda$.
8 Update number of converged eigenpairs n_c.
end
9 Return first k columns of X and leading $k \times k$ block of Λ.

3.2 Computing Interior Eigenvalues

The LOBPCG algorithm has several advantages, such as blocking and preconditioning, compared to the Lanczos algorithm. However, the standard LOBPCG algorithm, as well as the standard Lanczos algorithm, only allow for computing the lower or upper part of the spectrum.

Within the Lanczos algorithm this issue is solved by a *shift-and-invert* transformation

$$(H - \sigma I)^{-1},$$

where $\sigma \in \mathbb{R}$ is the shift. This spectral transformation maps the eigenvalues closest to the shift σ to the outer part of the transformed spectrum which then can be efficiently computed by the shift-and-invert Lanczos algorithm. The big downside of this transformation is that it requires a memory demanding LU factorization for inverting the shifted matrix. This makes it impractical for large numbers of spins. As reported in [19], the computational cost of the overall algorithm is dominated by the construction of the LU factorization.

In order to avoid storing the matrix and computing memory demanding LU factorizations, we will make use of a different spectral transformation, the so called *spectral fold* [24]

$$(H - \sigma I)^2,$$

where $\sigma \in \mathbb{R}$ is again the shift. This transformation maps all eigenvalues to the positive real axis and the ones closest to the shift σ to the lower edge close to 0. Hence, we can use the LOBPCG eigensolver in combination with matrix-free block matrix-vector operations in order to compute interior eigenvalues and their corresponding eigenvectors. Because the transformed eigenvalue problem

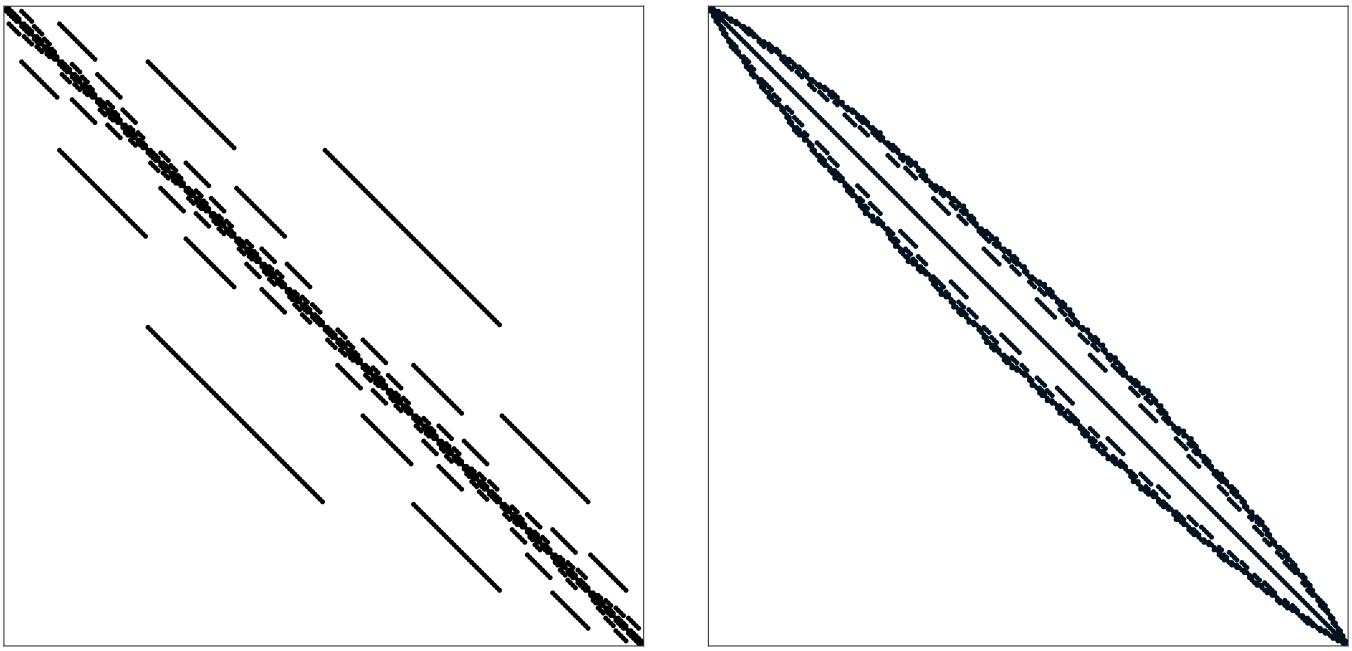
$$(H - \sigma I)^2 x = \lambda x$$

is symmetric positive definite, we will use a diagonal (Jacobi) preconditioned conjugate gradient (PCG) method as preconditioner for the LOBPCG eigensolver, so that we again can make use of matrix-free block matrix-vector operations.

3.3 Matrix-Free Matrix-Vector Product

In contrast to the shift-and-invert Lanczos algorithm, where the dominant computational cost is the construction of the LU factorization, the dominant computational cost of the LOBPCG algorithm is the (block) matrix-vector product. Note also that by applying a spectral fold transformation, the matrix-vector product of the transformed matrix can be implemented simply by repeatedly applying a standard matrix-vector product of our Hamiltonian.

In order to implement this matrix-vector (MATVEC) operation matrix-free and efficiently, we must consider our choice of basis states, as well as our ordering of these states in the vector. We use two different orderings, one for the pure OpenMP and the other



(a) OpenMP state ordering

(b) MPI state ordering

Figure 1: The sparsity structure of the $L = 10$ Hamiltonian using (a) the OpenMP state ordering and (b) the MPI state ordering.

for the hybrid MPI–OpenMP implementation. They are chosen to optimize for SIMD vectorization and communication bandwidth respectively. To conserve memory, we compute all off-diagonal matrix elements on the fly, avoiding explicitly storing them.

3.3.1 Basis States. A convenient basis for the Hamiltonian is the Z-polarized product state basis: the states in which every spin is in a Z-eigenstate. These states can be represented compactly as a bitstring, with zeros representing the spin-up state and ones representing the spin-down state (or vice versa). This representation, which is also used in [10], allows fast computation of the values of matrix elements via bitwise operations. For example, the value of the term $\sum_{\langle i,j \rangle} S_i^z S_j^z$ can be computed in just a couple of operations:

```
inline double ZZ(int state, int L) {
  // number of terms in the sum that are -0.25
  int n_negative = __builtin_popcount(state ^ (state>>1));
  return 0.25*(L - 2*n_negative - 1);
}
```

3.3.2 Data Layout of Block Vectors. LOBPCG is a block solver, meaning that it operates on a block of several vectors at the same time, usually 32 or 64 in our case. A crucial performance consideration is how this data should be stored. When viewing this block as a tall, skinny dense matrix, there are two obvious possibilities: row- or column-major. Other possibilities such as Z Morton ordering exist, but there is no clear reason that they would give performance gains in this context.

Both intuitively and empirically, we find that row-major ordering yields a faster matrix-vector multiplication than column-major does. From a data-locality perspective this makes sense [20]. When we are performing the multiplication for a particular matrix element, in the row-major case the sequence of relevant values in the input vector block are contiguous in memory, meaning that they can all be fetched in the same cache line. Furthermore, the multiplication can be vectorized with SIMD instructions. In the column-major case, those same vector values are spread out in memory by a distance equal to the vector length, and must be accessed separately. Furthermore, there is danger of concurrency issues with false sharing in the column-major case. In particular, if threads are each given a unique set of matrix rows to compute, they will never write to the exact same places in the output vector. However, in the column-major case two threads are more likely to write to *nearby* locations in memory. If these nearby locations are on the same cache line, serious performance degradation can result.

3.3.3 OpenMP MATVEC. This single-node, pure OpenMP implementation targets the Intel Knight's Landing architecture specifically, with the following aspects of the hardware in mind: (i) large number of threads and hyper-threading, (ii) very fast MCDRAM used in cache mode, and (iii) 512-bit SIMD vector units. In order to optimally use these features, an ordering of states was chosen in which the off-diagonal matrix elements form a series of diagonal bands, see Figure 1(a). The state ordering that produces these diagonal bands is simple: the states are simply sorted lexicographically, or equivalently, sorted by their values when the bitstrings are interpreted as integers. Iteration along these bands is very fast because the same operation is being applied repeatedly to neighboring data values. This makes data access patterns easily predictable

for the prefetcher, and also allows easy vectorization with SIMD instructions.

Generally, when computing the off-diagonal elements, a given row index is converted to its corresponding state bitstring, that bitstring is manipulated to yield a new state bitstring, and that new state is converted back into a (column) index. That conversion from state back to index can cause performance issues. In general it can be performed in $O(\log n)$ time using binary search, but that can become a large overhead since it needs to be performed for every matrix element that is computed. Instead, we compute the difference between the row and column indices directly, using the state. Recall that the only off-diagonal elements are flip-flop terms, which exchange two neighboring spins but do not affect any other part of the state. Consider a row corresponding to a state

$$x_{L-1} \cdots x_{i+2}, 0, 1, x_{i-1} \cdots x_0,$$

where x_j represents the configuration of spin j. It will have a matrix element connecting it with the column corresponding to the state

$$x_{L-1} \cdots x_{i+2}, 1, 0, x_{i-1} \cdots x_0.$$

It can be shown that in a lexicographical ordering, the difference in indices between these two states is $\Delta = \binom{i}{\sum_{i=0}^{i-1} x_i}$, where $\binom{n}{k}$ is the binomial coefficient function.

In order to optimize for the prefetcher and for vectorization, we would like to operate on many sequential values in memory, which this reordering allows us to do. However, we would also like to compute all of the matrix elements for a single row of the matrix at the same time, in order to optimize for temporal locality of write operations to the same location in the output vector, such that the data can be stored in the cache in between writes. We can optimally balance these needs by iterating across small *blocks* of rows, of a size that corresponds to the L1 cache size. This allows us to iterate efficiently along each of the sequential elements in a particular block while not losing relevant data from the cache. Each OpenMP thread owns a unique set of rows of the matrix, corresponding to several of these blocks, and thus there is no concern about race conditions or need for atomic operations.

3.3.4 MPI–OpenMP MATVEC. For the hybrid MPI–OpenMP implementation, we use one MPI rank per node, and OpenMP for on-node parallelism. In this case, communication bandwidth is limiting for anything more than a few MPI ranks. Thus, we choose a state ordering that minimizes the amount of data that needs to be communicated. To do so, we employ a breadth-first-search based ordering strategy, that is reminiscent of Cuthill–McKee reordering [6]. We begin with some initial state, and perform a breadth-first search (BFS) through the graph corresponding to the Hamiltonian, recording basis states as we encounter them. Due to the structure of the Hamiltonian, this ordering greatly reduces the bandwidth of the matrix, as can be seen in Figure 1(b). This narrow bandwidth allows communication with only neighboring ranks in a linear topology, for up to ~50 nodes.

This reordering does not permit the fast direct calculation of differences between indices that was used in the pure OpenMP version, though that same method could be used along with a lookup table. While this is ostensibly concerning, the index lookup time is ultimately irrelevant because it only affects the speed of the computational portion of the matrix-vector multiply. The communication is the dominant cost and the communication and computation are overlapped, so there is no overhead from a slightly slower computational portion.

4 NUMERICAL EXPERIMENTS

All numerical experiments were performed on the NERSC super computer called Cori which has 2 different types of compute nodes:

- Intel Xeon "Haswell" compute nodes @2.3 GHz, 2x16 cores and each 2 hyper-threads, 128 GB DDR4 RAM.
- Intel Xeon Phi "Knights Landing" (KNL) compute nodes @1.4 GHz, 68 cores and each with 4 hyper-threads, 96 GB DDR4 RAM, 16 GB MCDRAM.

4.1 OpenMP MATVEC

The full OpenMP, single node implementation was used to solve instances of the problem up to $L = 24$ spins. In this range of system sizes the required memory can easily fit on a single node, and even fully in the MCDRAM of a single KNL node, see Table 1.

Figure 2 shows the parallel speedup of the block MATVEC with block size 32 on both a Haswell and KNL node. The speedup is calculated as the ratio of running the OpenMP MATVEC code using the full number of available threads including hyper-threading, 64 threads and 272 threads, respectively, and using only a single thread. As can be seen in this figure, the implementation makes full use of the many-core architecture of the KNL nodes. At smaller system sizes, there is not quite enough work for each core to do to allow full utilization, but as the system size grows, all of the physical cores become well-utilized.

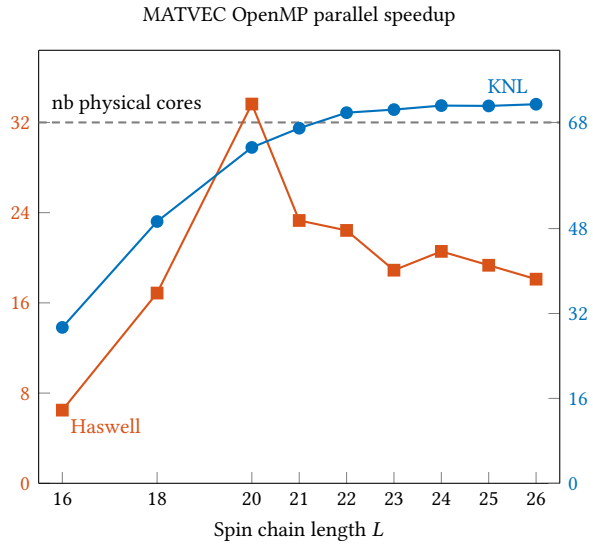


Figure 2: OpenMP parallel speedup for Haswell and KNL of the block MATVEC with block size 32. The vertical axes correspond to the speedup obtained when running using the full number of available threads, when compared to running with only a single thread on the same hardware.

From a priori estimation and empirical evidence, it is clear that memory bandwidth is the limiting factor for the pure OpenMP matrix-vector multiplication. We hypothesize that competition for memory bandwidth is what prevents the Haswell nodes from efficiently using all the cores. On the Haswell nodes the 32 cores are split into two 16-core sockets, with each socket having four DIMMs. This means that four cores are using each DIMM concurrently, and it seems that the DRAM simply can't supply data fast enough to keep the CPUs saturated.

On KNL, however, the (more numerous) cores have both a lower individual clock rate and access to extremely fast 16 GB of MCDRAM with 460 GB/s total bandwidth. Our results show that this hardware, along with the matrix-vector multiplication designed for efficient vectorization, is able to keep all 68 cores supplied with data. In Figure 2 we also note that the speedup is actually slightly higher than the number of physical cores. We hypothesize that this is due to L1 and L2 cache effects, i.e., with less work per core in the parallel case, it is more likely that requested memory location will already be in the cache. Hence, for the remainder of the paper we will use the KNL compute nodes for all numerical experiments.

4.2 MPI–OpenMP MATVEC

For the hybrid MPI–OpenMP, we have implemented 3 different communication mechanisms: blocking using `MPI_Send/Recv`, non-blocking using `MPI_Isend/Irecv`, and one-sided remote memory access (rma) using `MPI_Put`. For the former ones we have also implemented 2 variants: one with and one without overlapping communication and local computation. The overlapping is achieved by explicitly allocating one OpenMP thread to the MPI calls, while the other threads perform the matrix-vector multiplication on the local matrix elements.

The results from a strong scaling experiment for all different variants of the MPI–OpenMP MATVEC with $L = 26$ and block size 64 are shown in Figure 3. The top figures clearly indicate that in this hybrid MPI-OpenMP case, the efficiency of the matrix-vector product is ultimately limited by communication bandwidth since overlapping communication and local computation yields a reduction of the wall time by 18% up to 36% in this case.

In Figure 3 we also note that the different communication mechanism result in very similar timing results. This is probably due to the predictable nature of the communication, i.e., neighboring nodes only and at predictable times. Overall the non-blocking `MPI_Isend/Irecv` implementation was found to be the most performant.

4.3 Eigenstate Entanglement Entropy

Using the pure OpenMP implementation up to system sizes of $L = 24$, we present preliminary data on the entanglement entropy for the localization transition. The performance data for these runs is given in Table 2.

For each disorder realization, 16 eigenpairs with eigenvalues nearest zero were computed, and the half-chain entanglement entropy was computed for each eigenvector. As is described in Section 2.2.2, the entropy in the thermal case in this regime is expected to scale linearly with L, with a constant correction of -0.5. To thus normalize the entropy across different system sizes we defined a

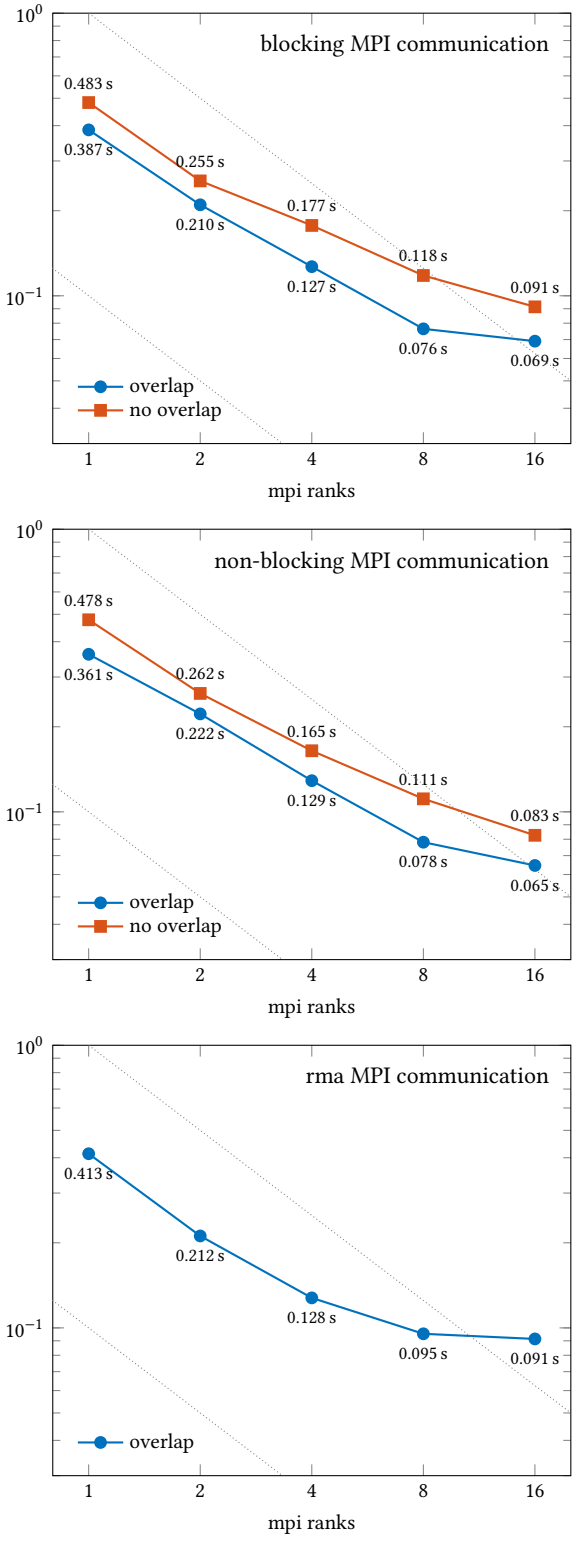


Figure 3: Strong scaling of the $L = 26$ MPI–OpenMP MATVEC and block size 64 with blocking, non-blocking, and rma MPI communication.

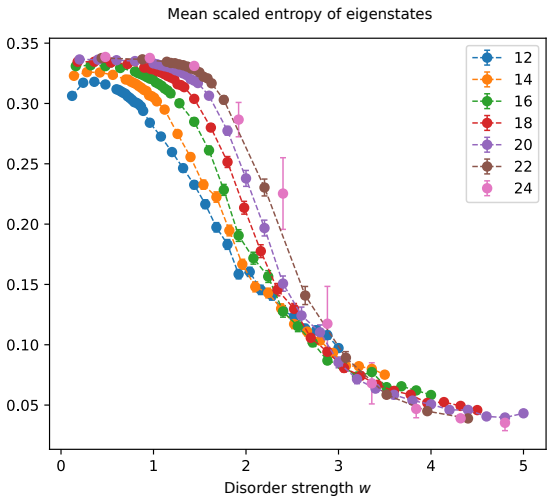


Figure 4: The scaled half-chain entanglement entropy as a function of disorder strength.

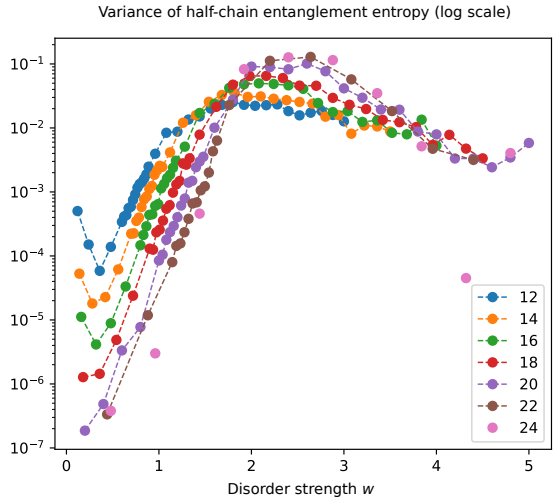


Figure 5: The variance of the half-chain entanglement entropy as a function of disorder strength.

scaled entropy simply as

$$\tilde{S} = \frac{S_{L/2} + 1/2}{L}.$$

In Figure 4, we plot the results of our computations of this scaled entropy. In general, we expect the scaled entropy to attain a value of ln 2 in the thermal state for all system sizes, and then to decrease as the system becomes localized. This corresponds with what we see in our results, up to some small negative corrections at low system sizes, which were also seen in [19], though that paper used a slightly different metric.

When plotting the variance of the entanglement entropy in Figure 5, we see a clear description of the transition. At low disorder, every system size shows an exponential increase towards the transition, manifested as a line in the log-scaled variance plot. This exponential growth starts surprisingly early, and the slope of the growth in log scale, and thus the factor in the exponent, is consistent across system sizes. This can be interpreted as a consistent exponential approach to the transition, with the larger system sizes simply starting at smaller values, with the starting value changing by a constant factor with each increase in system size.

After this exponential growth, we see the curves peak, at a point which can be interpreted as the center of the transition. Our three largest system sizes qualitatively converge on a consistent point for this peak. With data for larger system sizes, we hope to be able to resolve this point precisely. A clear next step is to run full disorder averaging at a full set of disorder points for $L = 24$ and beyond. Timing results for $L = 26$ are given in Table 3. At these system sizes, the exponential growth region should converge with the observed peak, yielding insight into precisely where the transition occurs.

Another next step is to do this same analysis but further away from the middle of the spectrum. Since our solver is faster in that region where the eigenvalues are less dense, as reported in Table 3, we should be able to observe the transition behavior at larger system sizes. Such a speedup is not possible with the shift-and-invert

Table 2: Runtime to solve for the 16 eigenpairs with eigenvalues nearest to 0. Runs used a single KNL node with the full OpenMP solver implementation.

L	LOBPCG tolerance	Mean wall time
12	10^{-5}	0.6 s
14	10^{-5}	1.4 s
16	10^{-5}	4.5 s
18	10^{-5}	30.7 s
20	10^{-5}	4.7 min
22	10^{-5}	1.0 h
22	10^{-6}	1.2 h
24	10^{-6}	16.6 h

Table 3: Timing results for computing 32 eigenpairs of $L = 26$ Hamiltonians as a function of the normalized shift ε. Runs used 32 KNL nodes and LOBPCG with block size 64, tolerance 10^{-6}, and preconditioned with PCG.

ε	LOBPCG iter	PCG iter	time [h]
0.1	18	200	0.15
0.2	11	5,000	2.36
0.3	33	10,000	13.94
0.4	32	20,000	26.05
0.5	30	20,000	24.84
0.6	32	20,000	26.05
0.7	36	10,000	15.36
0.8	10	5,000	2.09
0.9	14	200	0.12

Lanczos algorithm, whose memory usage is ignorant to the value of the shift applied.

5 CONCLUSIONS

We have introduced a new approach to study many-body localization, which requires computing many eigenvalues and corresponding eigenvectors of large Hamiltonians in different regions of the spectrum. Our approach uses the LOBPCG algorithm, in combination with an efficient and matrix-free implementation of the block matrix-vector multiplication on many-core architectures to compute the desired eigenvalues and eigenvectors. Such an approach allows us to overcome the memory bottleneck in the previously used shift-and-invert Lanczos algorithm. As a result, the total memory footprint is reduced by several orders of magnitude, which allows us to compute eigenpairs of spin chains with up to $L = 24$ on a single compute node. We have also developed an hybrid MPI–OpenMP version of the solver that can be run on several nodes. Because, the MBL study requires solving eigenvalue problems for many instances of Hamiltonians with random disorder terms, and computing eigenvalues from different regions of the spectrum, the overall computation can scale to hundreds of thousands of computational cores.

ACKNOWLEDGMENTS

The authors are grateful to Chris Laumann and Francisco Machado for helpful discussions about MBL and thermalization, to Meiyue Shao for pointing out important implementation details of LOBPCG, and Pieter Ghysels for providing the memory footprints of the LU factorizations generated by STRUMPACK.

This work is partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program and Center for Novel Pathways to Quantum Coherence in materials, an Energy Frontier Research Center funded by the US Department of Energy, Director, Office of Science, Office of Basic Energy Sciences under Contract No. DE-AC0205CH11231.

This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] Philip Warren Anderson. 1958. Absence of diffusion in certain random lattices. *Phys. Rev.* 109, 5 (1958), 1492–1505. https://doi.org/10.1103/PhysRev.109.1492

[2] Bela Bauer and Chetan Nayak. 2013. Area laws in a many-body localized state and its implications for topological order. *J. Stat. Mech. Theory Exp.* 2013, 09 (2013), P09005. https://doi.org/10.1088/1742-5468/2013/09/p09005

[3] Pranjal Bordia, Henrik Lüschen, Sebastian Scherg, Sarang Gopalakrishnan, Michael Knap, Ulrich Schneider, and Immanuel Bloch. 2017. Probing slow relaxation and many-body localization in two-dimensional quasiperiodic systems. *Phys. Rev. X* 7, 4 (2017), 41047. https://doi.org/10.1103/PhysRevX.7.041047

[4] Jae-yoon Choi, Sebastian Hild, Johannes Zeiher, Peter Schauß, Antonio Rubio-Abadal, Tarik Yefsah, Vedika Khemani, David A Huse, Immanuel Bloch, and Christian Gross. 2016. Exploring the many-body localization transition in two dimensions. *Science* 352, 6293 (2016), 1547–1552. https://doi.org/10.1126/science.aaf8834

[5] Emilio Cuevas, Mikhail Feigel'man, Lev Ioffe, and Marc Mezard. 2012. Level statistics of disordered spin-1/2 systems and materials with localized Cooper pairs. *Nat. Commun.* 3 (2012), 1128. https://doi.org/10.1038/ncomms2115

[6] Elizabeth Cuthill and James McKee. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference (ACM '69)*. ACM, New York, NY, USA, 157–172. https://doi.org/10.1145/800195.805928

[7] Wojciech De Roeck, Francois Huveneers, Markus Müller, and Mauro Schiulaz. 2016. Absence of many-body mobility edges. *Phys. Rev. B* 93, 1 (2016), 14203. https://doi.org/10.1103/PhysRevB.93.014203

[8] Jed A Duersch, Meiyue Shao, Chao Yang, and Ming Gu. 2018. A robust and efficient implementation of LOBPCG. *SIAM J. Sci. Comput.* 40, 5 (2018), C655–C676. https://doi.org/10.1137/17M1129830

[9] Pieter Ghysels, Xiaoye Sherry Li, Christopher Gorman, and François-Henry Rouet. 2017. A robust parallel preconditioner for indefinite systems using hierarchical matrices and randomized sampling. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, Los Alamitos, CA, 897–906. https://doi.org/10.1109/IPDPS.2017.21

[10] Chunjing Jia, Yao Wang, Christian B Mendl, Brian Moritz, and Thomas Peter Devereaux. 2018. Paradeisos: A perfect hashing algorithm for many-body eigenvalue problems. *Comput. Phys. Commun.* 224 (2018), 81–89. https://doi.org/10.1016/j.cpc.2017.11.011

[11] Sonika Johri, Rahul Nandkishore, and R N Bhatt. 2015. Many-body localization in imperfectly isolated quantum systems. *Phys. Rev. Lett.* 114, 11 (2015), 117401. https://doi.org/10.1103/PhysRevLett.114.117401

[12] Andrew V Knyazev. 2001. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.* 23, 2 (2001), 517–541. https://doi.org/10.1137/S1064827500366124

[13] Thomas Kohlert, Sebastian Scherg, Xiao Li, Henrik P Lüschen, Sankar Das Sarma, Immanuel Bloch, and Monika Aidelsburger. 2019. Observation of many-body localization in a one-dimensional system with a single-particle mobility edge. *Phys. Rev. Lett.* 122, 17 (2019), 170403. https://doi.org/10.1103/PhysRevLett.122.170403

[14] David J Luitz, Nicolas Laflorencie, and Fabien Alet. 2015. Many-body localization edge in the random-field Heisenberg chain. *Phys. Rev. B* 91, 8 (2015), 81103. https://doi.org/10.1103/PhysRevB.91.081103

[15] Alexander Lukin, Matthew Rispoli, Robert Schittko, M Eric Tai, Adam M Kaufman, Soonwon Choi, Vedika Khemani, Julian Léonard, and Markus Greiner. 2019. Probing entanglement in a many-body–localized system. *Science* 364, 6437 (2019), 256–260. https://doi.org/10.1126/science.aau0818

[16] Vadim Oganesyan and David A Huse. 2007. Localization of interacting fermions at high temperature. *Phys. Rev. B* 75, 15 (2007), 155111. https://doi.org/10.1103/PhysRevB.75.155111

[17] Don N Page. 1993. Average entropy of a subsystem. *Phys. Rev. Lett.* 71, 9 (1993), 1291–1294. https://doi.org/10.1103/PhysRevLett.71.1291

[18] Arijeet Pal and David A Huse. 2010. Many-body localization phase transition. *Phys. Rev. B* 82, 17 (2010), 174411. https://doi.org/10.1103/PhysRevB.82.174411

[19] Francesca Pietracaprina, Nicolas Macé, David J Luitz, and Fabien Alet. 2018. Shift-invert diagonalization of large many-body localizing spin chains. *SciPost Phys.* 5, 5 (2018), 45. https://doi.org/10.21468/SciPostPhys.5.5.045

[20] Melven Röhrig-Zöllner, Jonas Thies, Moritz Kreutzer, Andreas Alvermann, Andreas Pieper, Achim Basermann, Georg Hager, Gerhard Wellein, and Holger Fehske. 2015. Increasing the performance of the Jacobi–Davidson method by blocking. *SIAM J. Sci. Comput.* 37, 6 (2015), C697–C722. https://doi.org/10.1137/140976017

[21] Michael Schreiber, Sean S Hodgman, Pranjal Bordia, Henrik P Lüschen, Mark H Fischer, Ronen Vosk, Ehud Altman, Ulrich Schneider, and Immanuel Bloch. 2015. Observation of many-body localization of interacting fermions in a quasirandom optical lattice. *Science* 349, 6250 (2015), 842–845. https://doi.org/10.1126/science.aaa7432

[22] Jacob Smith, Aaron Lee, Philip Richerme, Brian Neyenhuis, Paul W Hess, Philipp Hauke, Markus Heyl, David A Huse, and Christopher Monroe. 2016. Many-body localization in a quantum simulator with programmable random disorder. *Nat. Phys.* 12 (2016), 907. https://doi.org/10.1038/nphys3783

[23] Jan Šuntajs, Janez Bonča, Tomaž Prosen, and Lev Vidmar. 2019. *Quantum chaos challenges many-body localization.* Technical Report. arXiv:1905.06345

[24] LinWang Wang and Alex Zunger. 1994. Solving Schrödinger's equation around a desired energy: Application to silicon quantum dots. *J. Chem. Phys.* 100, 3 (1994), 2394–2397. https://doi.org/10.1063/1.466486