Massively parallel implementation and approaches to simulate quantum dynamics using Krylov subspace techniques

Marlon Brenes,[1] Vipin Kerala Varma,[1,2,3,4] Antonello Scardicchio,[1,5] and Ivan Girotto[1,6]

[1]The Abdus Salam ICTP, Strada Costiera 11, 34151 Trieste, Italy
[2] Initiative for the Theoretical Sciences, The Graduate Center, CUNY, New York, NY 10016, USA
[3] Department of Engineering Science and Physics, College of Staten Island, CUNY, Staten Island, NY 10314, USA
[4] Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA 15260, USA
[5] INFN, Sezione di Trieste, Via Valerio 2, 34127, Trieste, Italy
[6]University of Modena and Reggio Emilia, 41121 Modena, Italy

Abstract

We have developed an application and implemented parallel algorithms in order to provide a computational framework suitable for massively parallel supercomputers to study the dynamics of one-dimensional quantum systems. We use renowned parallel libraries such as PETSc/SLEPc combined with high-performance computing approaches in order to overcome the large memory requirements to be able to study systems whose subspace dimension is constituted by over 9 billion independent quantum states. Moreover, we provide descriptions on the parallel approach used for the three most important stages of the simulation: handling the Hilbert subspace basis, constructing a matrix representation for a generic Hamiltonian operator and the time evolution of the system by means of the Krylov subspace methods. We provide results to evaluate the overall performance of the application, as well as physical results from the dynamics of a quasi-disordered system under the Aubry-André model and a generic non-disordered model.

1 Introduction

In the last 20 years the study of how quantum systems reach (or do not) equilibrium has received a renovated interest. This is due mainly to experimental progresses in maintaining mesoscopic quantum systems isolated from the environment (i.e. from decoherence) being them cold atoms in optical traps [15], arrays of superconducting qubits [26] or impurities in crystals [18]. During the early years of quantum mechanics, it was established that equilibrium states of ergodic systems [10] must be effectively described with proper quantum statistical mechanics models. The modern view building on those early works is contained in the so-called "eigenstate thermalization hypothesis" [7]. However, the precise mechanism of how these states can be reached by local dynamics that follow microscopic laws is yet to be understood (see [8, 6] for a review) and counterexamples to widely hold conjectures are being found for example when quenched disorder is present in the system. Systems with quenched disorder might be constructed (and are indeed quite natural) to effectively behave like integrable systems [4, 20, 23, 21, 12] their time evolution constrained to the conservation of extensively many local observables. These counterexamples have spurred a sort of dissonance between a microscopic description based on Schrödinger equation and one that uses the classical ensembles from statistical mechanics, and have raised many questions, for example what are the nature and properties the dynamical quantum phase transitions.

Lacking a quantum computer, simulating unitary time evolution of quantum dynamical systems on a classical computer is, both in intellectual and computational terms, a very demanding task. Given the fact that studying quantum many-body systems out of equilibrium allows us to probe questions in the foundation of statistical mechanics and condensed matter theory, provide quantum simulators related to quantum computing as well as other quantum technologies; a platform to perform numerical operations and simulations involving these systems is indeed very important to current research. [6]

As entanglement is the main obstacle to simulating a quantum system on a classical computer, essentially, the techniques for calculating the time evolution of a quantum systems are of two kinds. The first kind assumes that entanglement will be small during the whole evolution and uses an approximate form of the wave function. These algorithms fall under the umbrella of density-matrix renormalization group evolution or tDMRG (see [22] for a review). The second kind does not make this assumption and handles from the very beginning the largest possible wave function which can fit given the computational resources. The first kind of algorithms is suited for evolving states close to the ground state. One is practically guaranteed that, for most models, the entanglement between a region A and a region B is bounded by the number of points at the boundary of A and B. However, as far as thermalization properties are concerned, close to the ground state is not the most interesting place to look at and, for this questions, tDMRG will quickly fail.

One is then led to consider the second category of algorithms, which store the entire wave function in memory without making any assumption on its entanglement structure to begin with. A typical approach in this category is then to perform full diagonalization of the Hamiltonian, to obtain eigenvalues and eigenvectors and use these to evolve the initial state for a given time t. However, for relatively large systems this practice is computationally problematic when it comes to actual computing times and memory requirements. A way out is the method of Krylov subspace techniques, of which a massively parallelized, effective implementation is the argument of this paper.

This paper is arranged as follows: Section 2 provides a brief background, we describe the implementation and approaches in Section 3, Section 4 provides Results and analysis and Section 5 is devoted to Performance analysis.

2 Background

In order to solve the quantum N body problem in the framework of a quantum lattice system, sparse matrix algorithms are usually applied to solve for the corresponding states of a given system. A translation of the considered many-particle Hamiltonian needs to be done, in the language of the second quantization, into a sparse Hermitian matrix. [9] Diverse models are under study to the further understanding of the many body localization phenomenon [16] and to evaluate real-time dynamics of lattice gauge theories [19] for example; where such a translation using sparse Hermitian matrices is common to study dynamic properties.

Models. — We focus our attention on a paradigmatic model for studying transport in one-dimensional system to illustrate the procedure of the parallel algorithms and implementation; namely, a system of one-dimensional *hardcore* bosons model with nearest neighbor hopping t and nearest neighbor repulsion V and an onsite quasidisordered potential of strength h, described by

$$\begin{aligned}\hat{H} = t\sum_{i=1}^{L-1}(\hat{c}_i^\dagger \hat{c}_{i+1} + H.c) \\ + V\sum_{i=1}^{L-1}(\hat{n}_i\hat{n}_{i+1}) + h\sum_{i=1}^{L-1}(\hat{n}_i \cos(2\pi\beta i + \phi)) \qquad (1)\end{aligned}$$

where $\hat{c}_i^\dagger$ is the bosonic creation operator, $\hat{c}_i$ is the bosonic destruction operator and $\hat{n}_i$ is the bosonic counting operator such that $\hat{n}_i \equiv \hat{c}_i^\dagger \hat{c}_i$; β is the inverse of the golden ratio given by $\frac{\sqrt{5}-1}{2}$ and ϕ is an arbitrary phase. In this work we restrict to lattices with sizes corresponding to Fibonacci numbers so that the disorder term completes a cycle along the length of the chain if β is chosen as the ratio of successive Fibonacci numbers i.e. $\beta = \dfrac{F_n}{F_{n+1}}$.

The two main limits of the above Hamiltonian that we consider are as follows: (i) The XXZ Hamiltonian with $h = 0, V \neq 0$ is a canonical integrable model for investigating transport and conservation laws in one-dimensional systems, with $V = 0$ corresponding to the case of free particles; (i) The Aubry-André model with $V = 0, h \neq 0$ is a *quasidisordered* model [1] which shows an Anderson delocalization-localization transition at all energy scales when $h = 2$.

In such a system the operation $\hat{H}\,|\psi\rangle$ returns a linear combination of other eigenstates in the Hilbert space basis, and so a Hamiltonian matrix can be constructed for the specific operator at hand. The result is a (usually very large) sparse Hermitian matrix. In this framework, an initial eigenstate can be prepared to study the behavior as a function of time of the system by means of the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t) \qquad (2)$$

with the corresponding solutions given by:

$$|\Psi_t\rangle = e^{-i\hat{H}t/\hbar}\,|\Psi_0\rangle \qquad (3)$$

Basis representation. — A proper representation of the basis vectors of the Hilbert subspace of dimension

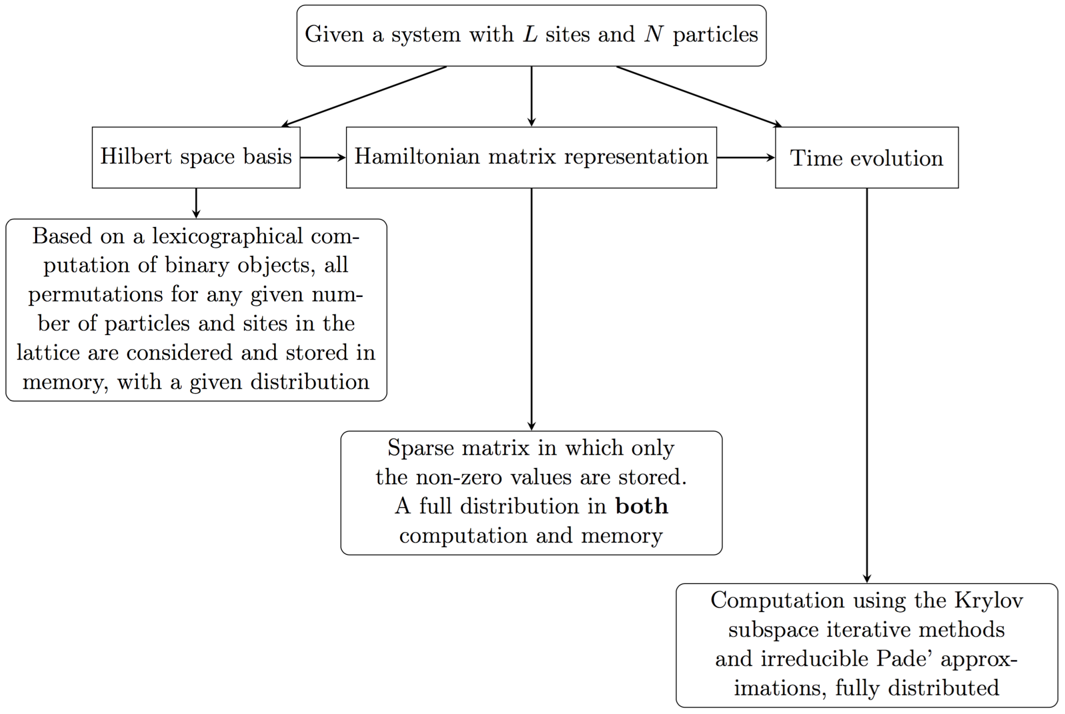
Given a system with L sites and N particles
Hilbert space basis
Hamiltonian matrix representation
Time evolution
Based on a lexicographical computation of binary objects, all permutations for any given number of particles and sites in the lattice are considered and stored in memory, with a given distribution
Sparse matrix in which only the non-zero values are stored. A full distribution in **both** computation and memory
Computation using the Krylov subspace iterative methods and irreducible Pade' approximations, fully distributed

Figure 1: Brief summary of the parallel design

$\mathcal{D}$ needs to be devised in order to perform operations on the computer and to create a matrix representation of the Hamiltonian operator. In particular, for the case of the hardcore bosons described before, the dimension of this space is given by $L!/N!(L-N)!$, where L is the linear dimension of the lattice and N is the number of particles.

An approach that can be used consists in assigning an integer value to each of the basis states of the subspace. In this representation, each of the states correspond to a value in a memory buffer that can be transversed by lookup algorithms. In that sense, the following is an example for the case of $L = 4$ and $N = 2$

$$\begin{aligned} |0011\rangle &\to 3 \\ |0101\rangle &\to 5 \\ |0110\rangle &\to 6 \\ |1001\rangle &\to 9 \\ |1010\rangle &\to 10 \\ |1100\rangle &\to 12 \end{aligned} \tag{4}$$

This is a very powerful mechanism to represent the basis, given that integer values are easier to work with than binary objects. In particular, if the basis is stored as a contiguous memory buffer effective lookup algorithms can be used to search for a specific element; even more so if the elements are sorted.[1]

Krylov subspace methods. — Applying the methodology described, the problem translates to evaluate the exponential of a large sparse matrix for the system. We can apply the technique of Krylov subspaces in order to **avoid** full diagonalisation. With this approach we approximate the solution to Equation 4 using power series. The optimal polynomial approximation to $|\Psi(t)\rangle$ from within the Krylov subspace,

$$\begin{aligned} \mathcal{K}_m = & span\{|\Psi_0\rangle, H|\Psi_0\rangle, \\ & H^2|\Psi_0\rangle \ldots, H^{m-1}|\Psi_0\rangle\} \end{aligned} \tag{5}$$

is obtained by an Arnoldi decomposition of the matrix $A_m = V_m^T H V_m$ where m is dimension of the subspace and previously defined. In order to attain convergence,

[1]For instance, a binary lookup could be used to search for an element of an array of size N with complexity $log(N)$, compared to the complexity of N given by an element-by-element lookup

Algorithm 1 Parallel distribution

```
procedure DISTRIBUTION(PetscInt &$nlocal$, PetscInt &$start$, PetscInt &$end$)
    $nlocal = basis\_size\_ \,/\, mpisize\_$
    PetscInt $rest = basis\_size\_ \,\%\, mpisize\_$
    if $rest$ && ($mpirank\_ < rest$) then
        $nlocal++$
    end if
    $start = mpirank\_ * nlocal$
    if $rest$ && ($mpirank\_ >= rest$) then
        $start \mathrel{+}= rest$
    end if
    $end = start + nlocal$
end procedure
```

a much smaller dimension of the Krylov subspace is required in comparison to the dimension of the Hilbert subspace. [24]
The solution is then given by:

$$|\Psi(t)\rangle \approx V_m exp(-itA_m)\,|e_1\rangle \tag{6}$$

where $|e_1\rangle$ is the first unit vector of the Krylov subspace. The much smaller matrix exponential is then evaluated using irreducible Padè approximations. [16] The algorithm to evaluate the numerical method has been extensively described by [24].

3 Implementation

Figure 1 shows a brief description of the workflow implemented.

We start with a an abstract 1D lattice quantum system described by the size of the grid, the number of particles present in the system and a given Hamiltonian operator, such as the ones presented in Equation 1 and Equation 2. The size of the subspace[2] is given by $L!/N!(L-N)!$, with a particular boundary conditions this describes the quantum system.[3] The following step is to create a representation of the Hilbert subspace, this can be done using integer-binary operations on the computer. The method we use for this section is a lexicographic computation of next bit permutations, this provides a fast way of computing all the possible combinations in a sorted manner, therefore avoiding the requirement of sorting algorithms for later lookup routines.

The construction of matrix representation of the Hamiltonian operator rests on this basis. This requires to apply the Hamiltonian operator to each of the states in the basis to get the correlation among each of the states; this translates into a sparse, Hermitian matrix that is used for the time evolution of a particular system. A sparse storage format is required in this stage, being that the expected sizes of the system subspace are very large. The preparation of the initial eigenstates needs to be done consistently with the format of the Hamiltonian matrix in terms of the parallel approach.

When it comes to the study of disordered systems, many layers of disorder can be introduced to the system in the form of randomness. One way to introduce disorder to the system is to introduce randomness to the parameters of the Hamiltonian for example, or to use a random initial eigenstate for each simulation. [28]

For the unitary time evolution of the system, there's no requirement to construct the Hamiltonian matrix for each timestep. One can just adjust the time parameter and evolve the system using the same Hamiltonian representation. This is done by means of the Krylov subspace methods in order to obtain an evolved state, once this is computed many different properties of the system can be computed in the form of quantum observables, which characterizes the behavior of the system as a function of time.

External libraries and dependencies. — The developed application relies on external libraries, namely: `C++ Boost ver 1.61.0` [5], `PETSc ver 3.7.3` [2] for parallel matrix and vector objects and `SLEPc ver 3.7.2` [11] for time evolution routines. `PETSc` was com-

[2]Composed by all the accessible states of the system
[3]It's easy to see that *half-filling*, i.e, $N = L/2$ gives the larger subspace for a given system

Algorithm 2 Next permutation of bits

```
1: procedure ConstructIntBasis(LLInt *$int\_basis$)
2:     LLInt $w$                                                ▷ Next permutation of bits
3:     LLInt $smallest$ = $smallest\_int()$                     ▷ Smallest int of the basis
4:     $int\_basis[0] = smallest$
5:     for LLInt $i = 1$ to $basis\_size\_ - 1$ do
6:         LLInt $t = (smallest \mid (smallest - 1)) + 1$
7:         $w = t \mid ((((t \,\&\, {-t} \,/\, (smallest \,\&\, {-smallest})) >> 1) - 1)$
8:         $int\_basis[i] = w$
9:         $smallest = w$
10:    end for
11: end procedure
```

piled using Intel MPI and Intel MKL [13] and built to support complex datatypes, 64-bit integers and indices, `FORTRAN` kernels and interfaces.

Basic parallelization. — For a fully parallel computation and distribution of objects across processing elements, a row distribution can be used to handle vector and matrix objects. Algorithm 1 shows a way in which this can be accomplished. The benefit of using this simple distribution is consistency in relation to external libraries (`PETSc`). The variables `mpisize_` and `mpirank_` are queried by means of MPI functionality and the parameters are used to decide the row sections of vector and matrix objects that each processing element will allocate and have access to. The variables `nlocal`, `start` and `end` refer, respectively, to the local number of rows and global row indices of each processing element. This distribution holds for both the construction of the Hamiltonian and the time evolution procedures, while the construction of the basis rests on different distributions as discussed in the following sections.

Basis representation. — A basis representation for the Hilbert subspace needs to be computed, given that the construction of the Hamiltonian operator is done by means of the basis. This requires to actually compute the integers that represent each of the states of the subspace. To do this, we used a contiguous section of memory of the required size. One could use standard C++ containers for this, but for specific design reasons related to memory management in later stages, we allocate memory using the regular mechanism by means of the `new` command and use raw pointers to access and modify elements. This was implemented as shown in Algorithm 2.

In such a way, the `int_basis` container gets filled with each of the possible permutations in a sorted manner. The `smallest_int()` function is a very simple routine that computes the smallest integer value of the bit representation corresponding for any given L and N. `Boost`'s component, `dynamic_bitset`, provides easy to use functionality to create binary objects out of each of the elements of this container and moving from a binary to integer representation. This functionality was used for the later construction of the Hamiltonian matrix.

Hamiltonian matrix construction. — Once a representation for the basis has been constructed, the Hamiltonian matrix can be computed in a distributed fashion - each processing element allocates, computes and holds the row sections of the matrix given by the distribution shown in Algorithm 1. This can be done by mapping the operations in Equation 1 and Equation 2 or any given Hamiltonian operator to bitwise operations and transformations and applying them to each of the states in the basis. A lookup routine is then used to obtain the indices of the corresponding matrix elements.

We proceed to sequentially compute the values of the basis by means of bit permutations, in such a way that the next permutation of a given `bitset` corresponds to the binary combination that provides the next integer value when translated into an integer representation. One of the benefits of doing this is not only it's performance, but the fact that the resulting integer representation of the basis is **sorted**.

In light of this, we can use a well-performing method to lookup entries of the memory buffer. For this particular case, a very good choice is to use a **binary lookup** and therefore using an algorithm with complexity $O(logN)$ instead of an element-by-element lookup with complexity $O(N)$.

We use `PETSc`'s `Mat` objects to handle Hamiltonian

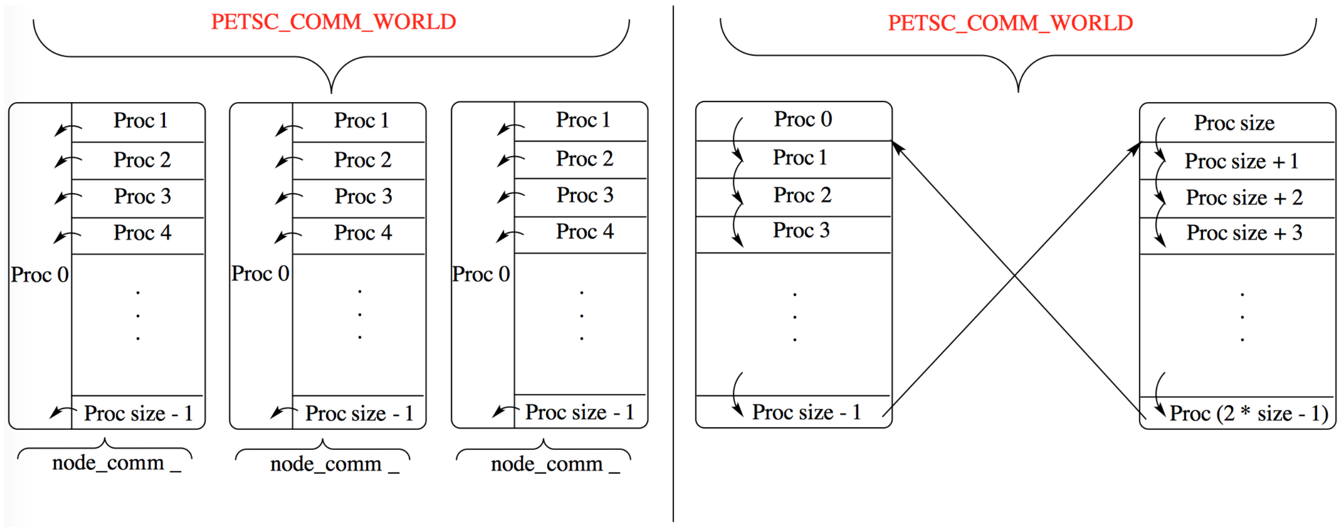

Figure 2: Visual representation of the `node communicator` (left) and `ring exchange` (right) parallel distribution arrangements for the Hilbert subspace basis representation

matrix representations. There are basically three different ways[2] in which a PETSc `Mat` object can be constructed:

1. Create an instance of the object without specifying preallocation details
2. Create an instance of the object providing estimated values of sizes for preallocation
3. Create an instance of the object providing the exact amount of elements in the diagonal and off-diagonal portions of the matrix with the parallel subdivision taken into account

Out of the three methods, the first one is the simplest but performs the worst. This is because of the *overhead* related to dynamically resizing memory sections. The second method performs well if a good estimation is provided, this usually requires allocating more memory than what it's actually required for the object.

Our goal is to optimize memory consumption, so the third method described above is the best for our purposes. This requires implementing another routine similar to the one used to find the Hamiltonian matrix elements, that computes the elements that each processing element contains in it's own section of the matrix, so that a **preallocation** step can be performed. This can also be performed in parallel in order to avoid compromising scalability.

The procedure introduces computational overhead, but the mechanism performs well enough so that we can use this in order to reduce memory consumption to a minimum, as shown in the Performance section of this paper.

Time evolution. — After constructing the matrix representation of the Hamiltonian operator in a distributed fashion and in consistency with the parallel distribution required to the PETSc and SLEPc routines, one can proceed to evaluate the time evolution of the system with an initial state. The initial state can be constructed in many different ways to study different behaviors of the system and should be represented as a vector distributed in parallel among processing elements, this can be easily done using PETSc. [3] For the time evolution procedure we can use SLEPc's routine related to the MFN component, which provides all the necessary framework with enough versatility to carry out the computation. [11]

Basis replication. — Table I presents a numerical calculation of memory consumption of the basis for given system sizes. It can be seen from Table I that for a large set of problem sizes[4], basis replication isn't really a problem to be concerned with memory-wise. Basis replication in this context means having a memory sec-

[4] $L = 28$ at half filling and smaller values of L, plus all the systems with subspace dimension smaller than this one

Algorithm 3 Construction of the basis in the `node communicator` and `ring exchange` arrangements

```
1:  procedure ConstructIntBasis(LLInt *$int\_basis$, PetscInt $nlocal$, PetscInt $start$)
2:      LLInt $w$                                          ▷ Next permutation of bits
3:      LLInt $first = first\_int(nlocal, start)$          ▷ Smallest int of the section of the basis
4:      $int\_basis[0] = first$
5:      for LLInt $i = 1$ to $nlocal - 1$ do
6:          LLInt $t$ = (first | (first −1)) +1
7:          $w = t$ | ((((t & -t / (first & -first)) >> 1) −1)
8:          $int\_basis[i] = w$
9:          $first = w$
10:     end for
11: end procedure
```

tion devoted to contain the integer values representing each of the states in the subspace per each computing element. In *Message Passing Interface* terms, each MPI process allocates and has access to the memory address of the entire basis.

Table I. Basis memory consumption for different problem sizes at half-filling.

System sizes	$\mathcal{D}$	Basis memory (GB)
$L = 28$, $N = 14$	4.01×10^7	0.320
$L = 30$, $N = 15$	1.55×10^8	1.25
$L = 32$, $N = 16$	6.01×10^8	4.8
$L = 34$, $N = 17$	2.33×10^9	18.7
$L = 36$, $N = 18$	9.08×10^9	72.6
$L = 38$, $N = 19$	3.53×10^{10}	282.8

Basis replication *has* to be avoided if one is interested in evaluating the dynamics of large systems, otherwise given the exponential increase of the size of the Hilbert subspace basis will quickly exhaust local memory resources. This implies that basis *distribution* across processing elements has to be done to carry out simulations of large systems. Here we present two different distribution arrangements and MPI communication patterns that can be used to overcome this problem.

3.1 Parallel distribution of the Hilbert subspace basis representation

`Node communicator` *approach.* — We focus our attention now on the first method that was used to overcome the basis replication problem. The paradigm consists in distributing the basis among all the processing elements, except for the first MPI process of each node, which allocates and holds the memory addresses of the entire basis. In this scenario, the entire memory required for the basis alone would be: 1 entire basis per computing node plus 1 entire basis distributed across the rest of the MPI processes. Computations required to construct the Hamiltonian matrix then require *intra-node* communications to find missing information. One of the benefits that are posed by this solution is the fact that the communication is being done inside of the node, most MPI implementations benefit from this using hardware locality directives. Figure 2 shows a visual representation of this arrangement.

We used this particular distribution for the construction of the Hamiltonian, however, the time evolution computation should use a global distribution by means of the global communicator (`PETSC_COMM_WORLD`) since that provides the best balance and compatibility with PETSc functionality.

The arrangement can be achieved by enabling a second MPI communicator. We called this second communicator `node_comm_`. As of the release of the MPI 3.0 standard there's a very natural way to accomplish this task by means of the MPI shared regions:

```
...
MPI_Comm node_comm_;
MPI_Comm_split_type(PETSC_COMM_WORLD,
  MPI_COMM_TYPE_SHARED,
     mpirank_, MPI_INFO_NULL, &node_comm_);
...
```

The `node_comm_` MPI communicator can be used to establish communication patterns of computing elements within a local node. Given that the first MPI process of each node contains the entire basis, communication can be performed in order to construct the matrix representation of the Hamiltonian operator using the data contained by this computing element.

This new distribution implies that the construction

Algorithm 4 Node communication pattern

```
 1: procedure NodeComm(...)
 2:     // Communication to rank 0 of every node to find size
 3:     if $node\_rank\_$ then
 4:         $MPI\_Send$(...)
 5:     else
 6:         for $i = 1$ to $node\_size\_$ do
 7:             $MPI\_Recv$(...)
 8:         end for
 9:     end if
10:     // Communication to rank 0 of node to find missing indices
11:     if $node\_rank\_$ then
12:         $MPI\_Send$(...missing info...)            ▷ Send unfound information
13:         $MPI\_Recv$(...required info...)           ▷ Required to finish the construction
14:     else                                           ▷ Rank 0 of every node
15:         for $i = 1$ to $node\_size\_$ do
16:             $MPI\_Recv$(...missing info of rank $i$)
17:             (...binary search...)
18:             $MPI\_Send$(...required info to rank $i$)
19:         end for
20:     end if
21:     (...complete construction with the updated data...)
22: end procedure
```

and computation of the basis, allocation details and Hamiltonian matrix has to be done consistently with the new arrangement. In particular this involves a careful indexing of memory locations that correspond to each of the elements of these objects. Algorithm 3 shows the approach used to compute the basis for both the `node communicator` and `ring exchange` approaches. It can be noticed that each MPI process will compute only it's own section of the basis. The `int_basis` buffer has size *nlocal*[5] given by the distribution in Algorithm 1. The method `first_int()` is a very simple routine that computes the first element in the basis for a given MPI process.

For the computation of the allocation details and the Hamiltonian matrix the indexing is changed accordingly. In this particular scenario, each MPI process will take ownership of a given subsection of rows in the matrix and compute the elements of the matrix by means of it's own basis section. Some elements of the matrix won't be able to be computed given that the local basis is incomplete, so this information is stored and communicated to the first MPI process of the local node in order to complete the procedure by means of the second MPI communicator. Algorithm 4 shows the approach for this particular section of the computation.

`Ring exchange` *approach.* — Depending on the available memory resources per node of the computational environment in which the application is executed, according to the estimates shown in Table I, allocating and constructing one entire basis per node can exhaust the memory resources of the system for large system sizes. Even if this could be done, there's still memory resources required for the actual Hamiltonian matrix and time evolution procedure. The node communicator version provides a good solution for a large range of system sizes, however, for systems that have a very large subspace dimension a fully distributed approach needs to be devised.

With the `ring exchange` approach: a full distribution of the basis across all the processing elements is achieved, communication is done in order to exchange sections of the basis and not the *unfound* elements of the Hamiltonian matrix (this is an important difference as computational load is more balanced in this setup). Moreover, the approach can be implemented by means of a single MPI communicator. Figure 2 shows a visual

[5]With the exception of the first MPI process of every node in the `node communicator` approach, for which $nlocal = basis\_size\_$

Algorithm 5 Ring communication pattern

```
1:  procedure RingComm(...)
2:      // Collective communication of global indices
3:      $gather\_nonlocal\_values\_(...)$;
4:      // Even when rest != 0, Proc 0 always gets the larger section
5:      // of the distribution, so we use this value for the ring exchange buffers
6:      $broadcast\_size\_of\_buffers\_(...)$;                              ▷ From 0 to all
7:      (...allocate ring exchange buffers, initialized to basis in a sorted fashion...)
8:      PetscMPIInt $next$ = ( $mpirank\_$ + 1 ) % $mpisize\_$;
9:      PetscMPIInt $prec$ = ( $mpirank\_$ + $mpisize\_$ - 1 ) % $mpisize\_$;
10:     for PetscMPIInt $exc = 0$ to $mpisize\_ - 2$ do
11:         $MPI\_Sendrecv\_replace\_(...)$;            ▷ Basis exchange using prec and next
12:         // source is required to find global indices from nonlocal values
13:         PetscMPIInt $source$ = $mod\_((prec - exc), mpisize\_)$;
14:         (...binary lookup of elements, if found, assign and multiply by $-1$)...
15:     end for
16:     (...flip all the signs: multiply by -1 all elements...)
17: end procedure
```

representation of the arrangement.

In this new setup then, each processing element exchanges sections of the basis in order to compute the elements of the Hamiltonian matrix. After *mpi_processes*-1 exchanges, all the elements have been computed and the Hamiltonian matrix is computed and distributed. With this setup, a *linear scaling* in memory is achieved which means that with increasing number of processing elements the amount of memory per MPI process required decreases linearly. However, *time scaling* is compromised, as increasing the number of processing elements will require more communication steps. Therefore, we expect that with this setup the time required to compute the Hamiltonian matrix will increase in relation to previous procedures.

The basis can be computed and distributed using Algorithm 3 using the corresponding global indices, while the computation of allocation details and the Hamiltonian matrix the indexing has to be changed accordingly. The major difference from the `node communicator` approach is the communication step.

Algorithm 5 shows the procedure used to perform the ring exchange of the basis in order to compute the Hamiltonian matrix. The first step is to to perform a collective communication so that every processing element holds the *nonlocal* values of the indices of each subsection of basis being communicated. This has to be done to keep track of the positions of the elements in the Hamiltonian matrix. Afterwards, since the number of processors is a generic parameter, different MPI processes may hold larger sections of the basis than others. To account for this, we use the larger size for the exchanging buffers with the remaining elements set to zero, as this won't affect the binary lookup given that no state in the basis is represented by the zero value.

The variable `source` identifies the MPI process that sends the section of the basis, this is required in order to align the indices with global values. The last step is to perform the ring exchange and the binary lookup procedure until all sections of the basis have been exchanged. In order to keep track of the already found elements, we set then to negative values and when the procedure is done we flip them back to positive entries. At the end of the procedure all the elements of the Hamiltonian matrix have been found and computed.

4 Results

Survival probability. — We can study the time evolution of the system by means of the survival probability. We start with the system prepared with a given initial state $|\Psi(0)\rangle$ at $t = 0$. The probability of finding the system in state $|\Psi(0)\rangle$ at time t is the so-called survival probability given by

$$F(t) = |A(t)|^2 \equiv |\langle\Psi(0)|\, e^{-iHt}\, |\Psi(0)\rangle|^2 \tag{7}$$

where $A(t)$ is the survival amplitude. We evaluated

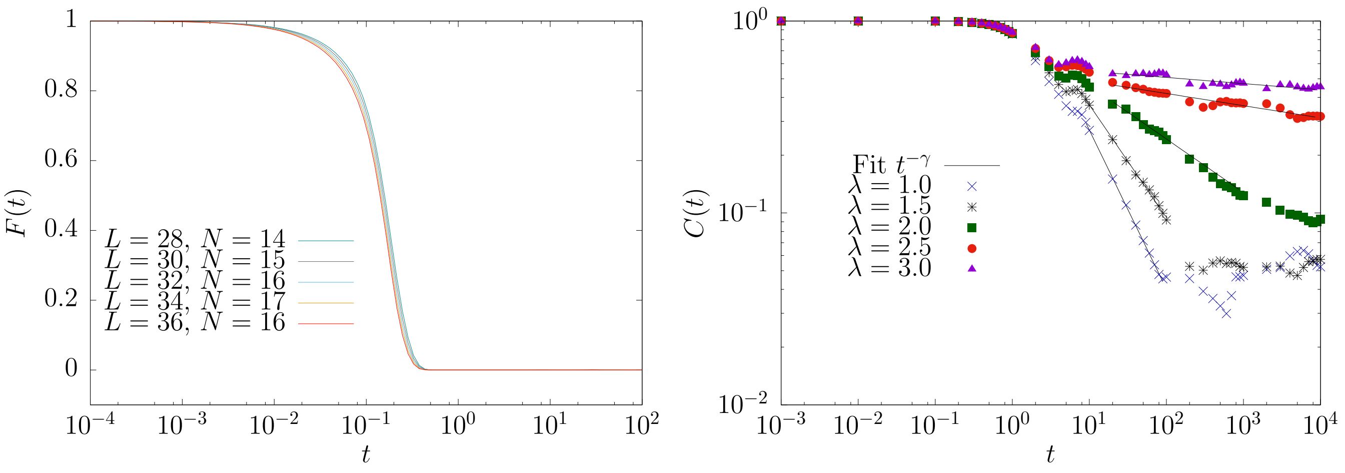

Figure 3: Survival probability $F(t)$ for different system sizes on a system with no disorder (left) and temporal autocorrelation function for the Aubry-André model non-interacting case with 1 particle ($L = 55$) using periodic boundary conditions for different values of λ (right)

this quantity by means of the Krylov subspace in systems of Hilbert space dimension of over 9 billion. The survival probability can be shown to be the Fourier transform of the local density of states (LDOS) corresponding to the initial state chosen: for an ergodically filled LDOS (expected for generic thermalizing or chaotic systems) the power-law decay of $F(t)$ is enhanced whereas this power-law can be suppressed due to multifractality or build-up of correlations among the eigenstates of the Hamiltonian. We employ the rate of power-law decay to measure the chaoticity or lack thereof in the systems.

Clean systems. — We first consider a clean hardcore bosonic chain with $h = 0$ in Eq. (1). Due to its integrability and lack of chaoticity the power-law is expected to be suppressed. Figure 3 exposes a very clean behavior of the survival probability given that Eq. (7) doesn't introduce any form of disorder to the system; for these simulations we used $t = 1$ and a weak interaction of $V = 0.2$, which is very close to the free fermionic system. The initial state used in these simulations is the Néel ordered state, which in our binary representation is the state given by (...010101). We find the power-law decay to be $\gamma \approx 0.97$ consistent with the fact that the model is integrable and nonergodic.

Disordered systems. — We evaluated the survival probability of a disordered system by means of the Aubry-André model with Hamiltonian operator given by Eq. (1) with periodic boundary conditions and gathered disorder averaged results over a random initial state. To produce cleaner results we measured a different form of the survival probability: the so-called *temporal autocorrelation function* which is given by [17]

$$C(t) = \frac{1}{t}\int_0^t |\langle\Psi(0)|\,|\Psi(t')\rangle|^2 dt' \tag{8}$$

The system is usually studied as a function of the parameters t, V and h in Equation 2. We can define $\lambda \equiv h/t$ and change this parameter for a fixed value of V. Figure 3 shows the behavior of the temporal autocorrelation function for five different regimes in the non-interacting case for a single particle: $\lambda < 2$ exposes the behavior of the delocalized states or extended states, $\lambda > 2$ shows the localized state behavior and $\lambda = 2$ is known to be a critical state, neither localized nor extended [17].

An interesting case of study is to evaluate the temporal evolution of the system described with the Aubry-André model for the interacting case, using higher densities. We proceeded in a similar fashion as we did for the single particle case and study the dynamics of the system using increasing densities $\rho \equiv N/L$.

Figure 4 shows a perspective of the temporal decay exponent as a function of both λ and the density ρ, it can be seen from this that increasing the density of system favors the decay towards the extended states. As

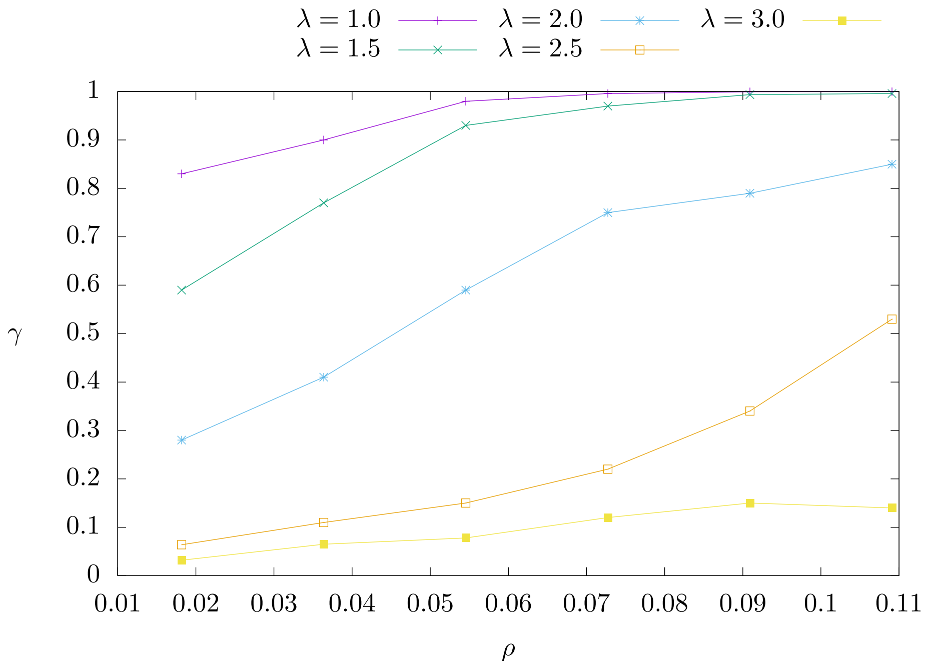

Figure 4: Temporal decay exponent γ for different densities ($\rho \equiv N/L$) and values of λ for the interacting Aubry-André model with periodic boundary conditions and $L = 55$

stated by [28], it can be said that increasing the density favors thermalization; although according to [28] the regime for which $0 < \gamma < 1$ no thermalization occurs at all. However we believe that for this system, because the presence of multifractality in the noninteracting limit already lead to slow temporal decay and $\gamma < 1$ even in the ergodic phase, as seen in the Fig. 4 and previously shown in Ref. [17], interactions cannot give $\gamma > 2$ required for ergodic filling of LDOS.

It can also be extracted from our analysis that $\gamma \leq 1$ for the λ values considered. This is an indicator that the powerlaw exponent extracted behaves and scales with similarity with the participation ratio PR

$$PR \propto \mathcal{D}^{\gamma} \tag{9}$$

where $\mathcal{D}$ is the dimension of the Hilbert subspace. When $\gamma \to 1$ this implies that $PR \propto \mathcal{D}$ indicative of full metallic (probably diffusive) behavior in one dimension. We may therefore, as an ad hoc hypothesis, posit that $\gamma < 1$ corresponds to subdiffusive dynamics.

To summarize, we can interpret the illustrated regimes of the system as follows:

- $\gamma \to 1$: As illustrated for increasing densities and $\lambda < 2.0$, this indicates the emergence of chaotic initial states with decreasing correlations among eigenstates; however full ergodic filling of LDOS is not present presumably due to multifractality already present in the single-particle limit which led to slow decay there [17]

- $\gamma < 1$: For $\lambda > 2.0$, indicates the presence of non-chaotic states with significant correlations among the eigenstates leading to very slow decay, again exacerbated by the multifractality in the noninteracting limit.

5 Performance analysis

System. — Table II shows the technical description of the system in which the application was tested and used for production runs.

Compilers and libraries. — The application has been compiled and tested using the C++ MPI wrappers of the Intel MPI compiler version 5.1.3. For underlying linear algebra operations we use Intel MKL version 11.3.3.

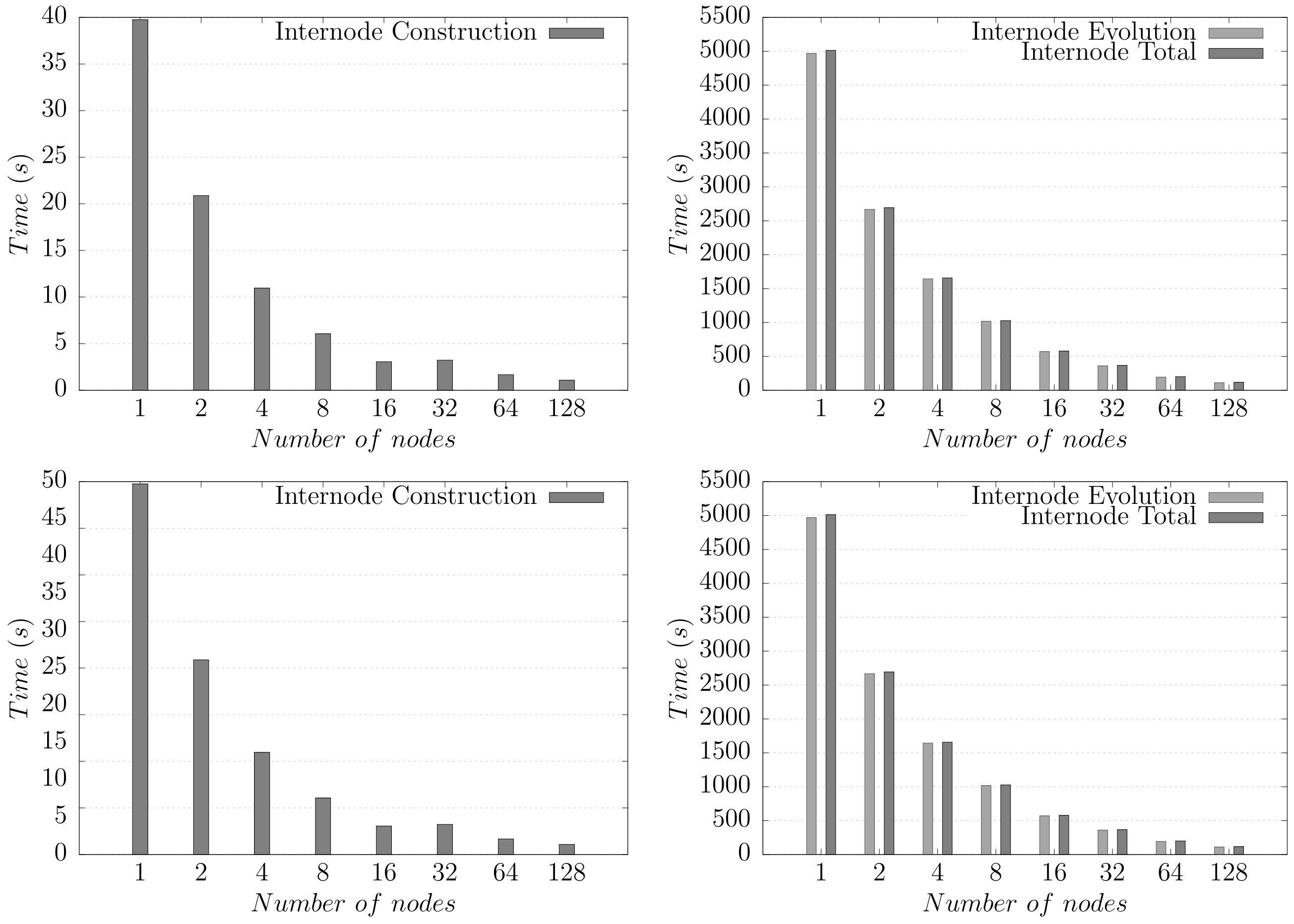

Figure 5: Strong scaling using a system with $L = 28$ at half-filling (subspace dimension of 40 116 600) up to $t = 100$ with a tolerance of 10^{-7} for a different amount of computing nodes: (TOP) using a replicated basis approach, (BOTTOM) using the `node communicator` approach, (LEFT) time required for allocation and computation of the Hamiltonian matrix and (RIGHT) time required for time-evolution procedure and overall walltime

The versions of the higher-end libraries used are: Boost v1.61.0, PETSc v3.7.3, SLEPc v3.7.2.

Performance results. — We started by evaluating the performance of the first instance of the application in a parallel environment. As was described in the Implementation section the first approach uses basis replication among all processing elements to construct the matrix representation of the Hamiltonian matrix, however, the computation and distribution of this object and the later time evolution step are performed in a fully distributed fashion.

Figure 5 shows the strong scaling behavior of the application in the two most important steps: the construction of the Hamiltonian object and the time evolution. As can be observed from Figure 5, the rest of the procedures such as computing the basis and initializing routines perform very well and account for only a negligible section of the computation time.

It can be seen that the time of the construction of the Hamiltonian object is very small compared to the actual time evolution procedure even for a relatively small physical time parameter ($t = 100$). This constituted our base for following development: one of our objectives were to keep the time to construct the Hamiltonian object small when compared to the time evolution, in order to avoid compromising scalability for a very large number of processing elements. Our results show a good scaling behavior even for large amount of processing elements (2048 MPI processes mapped to each core, as shown in Figure 5). An MPI approach was also required, given that for larger problem sizes ($L = 30$ at half filling and larger, for instance) can't be solved by means of a single computational node because of memory requirements and to enable the usage of more than one computational node for calculations.

Table II. System.

Description	Galileo-CINECA
Model	IBM NeXtScale
Architecture	Linux IB Cluster
Nodes	516
Processors	2 x 8-cores Intel Haswell
RAM	128 GB/node
Internal Network	IB with 4x QDR switches
Total on-board memory (RAM)	66 TB

Another important fact that can be extracted is that constructing and solving the system for the given parameters takes under 2 minutes overall with 128 nodes on the Galileo-CINECA machine. This is an important factor if for instance, one is interested to undertake disorder averages over random initial states or another random parameter of study.

`Node communicator` *approach.* — As was described in the previous section, basis replication is a big issue when it comes to large problem sizes. We developed the `node communicator` version with two objectives in mind: avoid basis replication and therefore have the possibility of studying larger system sizes and retain strong scaling behavior in the process. With this development we managed to achieve both as shown in Figure 5.

On the Galileo-CINECA machine we managed to solve a system with size up to $L = 34$ at half filling using this approach, although this version is also very useful on machines with less amount memory per node given that full replication of the basis is avoided and actual computation and scalability of the overall application is not compromised. The actual time evolution of the system is unchanged from the replicated basis approach so Figure 5 shows the same behavior as the previous version. In Figure 5 it can also be observed that the increase in time required to construct the Hamiltonian operator is very small compared to the time evolution of the system even for this relatively big system size, which was exactly our goal. Other different approaches, such as a systematic re-computation of sections of the basis to avoid storing this object in memory, proved to be inefficient.

Strong scaling behavior for this approach is not linear, but the results obtained are satisfactory when it comes to actually solving the problem for a large amount of processing elements. A *flat* behavior can be seen at the transition from 16 to 32 nodes, we believe this could be related to the network arrangement of the machine so that communication beyond 16 nodes could be using different switches in the interface (higher tree in the network arrangement). This behavior is present in both of the approaches used; so we can conclude it's unrelated to the parallel distribution, hence no further investigation is required.

`Ring exchange` *approach.* — The ring exchange version was designed with the purpose to open the possibility to even larger problem sizes. Recalling the description in the Implementation section, in this version no memory replication occurs in the application whatsoever. Memory requirements decrease linearly with increasing number of computing nodes, but *time scalability* is indeed compromised given that a larger amount of processing elements implicates more overall communications. So in perspective, a strong scaling view of the performance like the one shown for the previous developments is not a meaningful for this approach.

With this version, our goal was to have the ability to study even larger systems sizes by means of a full distribution of memory and computation overall. To demonstrate a perspective of the performance accomplished by this approach we show the time of computation required in the most important sections of the application with increasing problem size and processing elements in Figure 6.

It can be seen that the time to construct the Hamiltonian matrix object becomes important as the problem size increases, up to roughly 10% of the overall computation time for the larger case ($L = 36$ at half-filling). With this approach, most of the communication done during the construction of the Hamiltonian object occurs inside of the node which is beneficial in general terms, but when using a large number of processing elements the communication steps start to account for a considerable percentage of the overall execution time.

Even so, with this approach we were able to solve the $L = 36$ at half-filling system; a system that roughly has a subspace dimension of more than 9 billion elements.

Overall memory occupation. — By using the ap-

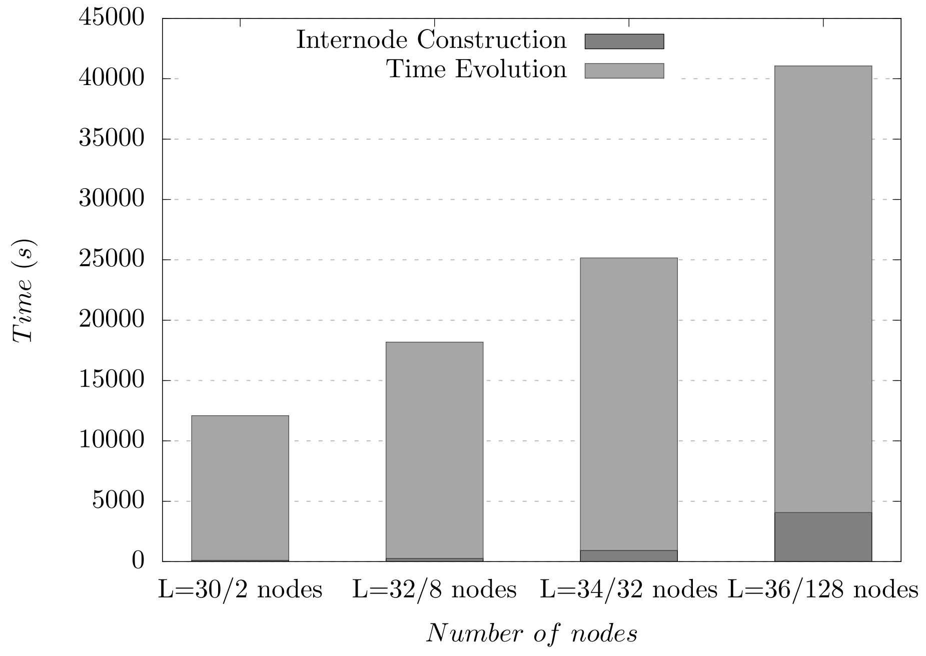

Figure 6: Running time of the application for the construction of Hamiltonian and time evolution using a different problem sizes at half-filling up to $t = 100$ with a tolerance of 10^{-7} for a different amount of computing nodes (`Ring exchange` approach)

proaches described in this paper we have managed to overcome the problem of basis replication by means of MPI distribution and communication patterns, however, a practical limit set by the size of the actual Hamiltonian matrix is still present.

Table III. Hamiltonian matrix and overall memory occupation at half-filling.

System sizes	$\mathcal{D}$	Matrix memory[6](GB)	Full occupation (TB)
$L = 28$	4.01×10^7	18	0.053
$L = 30$	1.55×10^8	75	0.220
$L = 32$	6.01×10^8	308	0.902
$L = 34$	2.33×10^9	1269	3.72
$L = 36$	9.08×10^9	5 227	15.3
$L = 38$	3.53×10^{10}	21 490	63.0

[6]Estimation

Table III shows an estimation of the memory required to store the matrix representation of the Hamiltonian operator and the overall memory occupation of the application.

The methodology of Krylov subspaces requires this matrix representation to be stored in memory to evaluate the projections on the subspace, such objects also are required to be stored internally by the library (`SLEPc`). The overall memory occupation presented in Table III during the time evolution procedure doesn't include the memory required by the basis given that the basis object can be deallocated before this routine. We have measured the full occupation using `PETSc`'s internal profiler. [2]

6 Conclusions

An application to study the dynamics of quantum correlated systems suitable to be executed on massively parallel supercomputers has been developed and tested in this work. We have used high-performing libraries in conjunction with distributed memory algorithms in order to study large quantum systems with subspace dimension of over 9 billion states with the computational resources available.

To check the validity of the results we have studied and presented the dynamics of known models. [17][28]

Different parallelization strategies have been implemented to gradually overrun memory scaling problem, but a practical limit is still present. As can be observed from Table III, evaluating the dynamics of a system with $L = 38$ at half-filling would required at least around 63 TB of memory distributed across many computational nodes, which corresponds roughly to the entire on-board memory of the machine used to run simulations (see Table II). Though this natural limit is still present and restricts the possibility to simulate even larger systems, our implementation has overcome the large memory requirements set by the basis to be able to simulate extremely large quantum systems.

Acknowledgements. — M.B acknowledges the Abdus Salam ICTP for it's support and funding under the MHPC program.

References

[1] S. Aubry and G. André. In: *Ann. Isr. Phys. Soc.* 3 (1980), p. 133.

[2] Satish Balay et al. *PETSc Users Manual.* Tech. rep. ANL-95/11 - Revision 3.7. Argonne National Laboratory, 2016. URL: `http://www.mcs.anl.gov/petsc`.

[3] Satish Balay et al. *PETSc Web page.* `http://www.mcs.anl.gov/petsc`. 2016. URL: `http://www.mcs.anl.gov/petsc`.

[4] DM Basko, IL Aleiner, and BL Altshuler. "Metal–insulator transition in a weakly interacting many-electron system with localized single-particle states". In: *Annals of physics* 321.5 (2006), pp. 1126–1205.

[5] Boost. *Boost C++ Libraries.* `http://www.boost.org/`. Last accessed Dec-2016. 2016.

[6] M. A. Cazadilla et al. "One dimensional bosons: From condensed matter systems to ultracold gases". In: *Rev. Mod. Phys.* 83 (4 Dec. 2011), pp. 1405–1466. URL: `http://link.aps.org/doi/10.1103/RevModPhys.83.1405`.

[7] J. M. Deutsch. "Quantum statistical mechanics in a closed system". In: *Phys. Rev. A* 43 (4 1991), pp. 2046–2049. URL: `http://link.aps.org/doi/10.1103/PhysRevA.43.2046`.

[8] J. Eisert, M. Friesdorf, and C. Gogolin. "Quantum many-body systems out of equilibrium". In: *Nat Phys* 11 (2 Feb. 2015). URL: `http://dx.doi.org/10.1038/nphys3215`.

[9] H. Fehske, R. Schneider, and A. Weisse. *Computational Many-Particle Physics.* Berlin Heidelberg, Germany: Springer, 2008.

[10] Sheldon Goldstein et al. "Normal typicality and von Neumann's quantum ergodic theorem". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences.* Vol. 466. 2123. The Royal Society. 2010, pp. 3203–3224.

[11] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems". In: *ACM Trans. Math. Software* 31.3 (2005), pp. 351–362.

[12] John Z Imbrie, Valentina Ros, and Antonello Scardicchio. "Review: Local Integrals of Motion in Many-Body Localized systems". In: *arXiv preprint arXiv:1609.08076* (2016).

[13] Intel. *Intel Math Kernel Library. Reference Manual.* Intel Corporation, 2009. ISBN: 630813-054US.

[14] Shankar Iyer et al. "Many-body localization in a quasiperiodic system". In: *Phys. Rev. B* 87 (13 Apr. 2013), p. 134202. URL: `http://link.aps.org/doi/10.1103/PhysRevB.87.134202`.

[15] Dieter Jaksch et al. "Cold bosonic atoms in optical lattices". In: *Physical Review Letters* 81.15 (1998), p. 3108.

[16] V. Kerala Varma et al. "Energy diffusion in the ergodic phase of a many body localizable spin chain". In: *ArXiv e-prints* (Nov. 2015). arXiv: `1511.09144 [cond-mat.dis-nn]`.

[17] R. Ketzmerick, G. Petschel, and T. Geisel. "Slow decay of temporal correlations in quantum systems with Cantor spectra". In: *Phys. Rev. Lett.* 69 (5 Aug. 1992), pp. 695–698. URL: `http://link.aps.org/doi/10.1103/PhysRevLett.69.695`.

[18] JR Maze et al. "Nanoscale magnetic sensing with an individual electronic spin in diamond". In: *Nature* 455.7213 (2008), pp. 644–647.

[19] C. Muschik et al. "U(1) Wilson lattice gauge theories in digital quantum simulators". In: *ArXiv e-prints* (Dec. 2016). arXiv: `1612.08653 [quant-ph]`.

[20] Rahul Nandkishore and David A Huse. “Many-body localization and thermalization in quantum statistical mechanics”. In: *Annu. Rev. Condens. Matter Phys.* 6.1 (2015), pp. 15–38.

[21] V Ros, M Müller, and A Scardicchio. “Integrals of motion in the many-body localized phase”. In: *Nuclear Physics B* 891 (2015), pp. 420–465.

[22] Ulrich Schollwöck et al. “Methods for time dependence in DMRG”. In: *AIP Conference Proceedings*. Vol. 816. 1. AIP. 2006, pp. 155–185.

[23] Maksym Serbyn, Z Papić, and Dmitry A Abanin. “Local conservation laws and the structure of the many-body localized states”. In: *Physical review letters* 111.12 (2013), p. 127201.

[24] R. B. Sidje. “Expokit. A Software Package for Computing Matrix Exponentials”. In: *ACM Trans. Math. Softw.* 24 (1 1998), pp. 130–156.

[25] Mark Srednicki. “Chaos and quantum thermalization”. In: *Physical Review E* 50.2 (1994), p. 888.

[26] Matthias Steffen et al. “Measurement of the entanglement of two superconducting qubits via state tomography”. In: *Science* 313.5792 (2006), pp. 1423–1425.

[27] M. Távora, E. J. Torres-Herrera, and L. F. Santos. “Inevitable power-law behavior of isolated many-body quantum systems and how it anticipates thermalization”. In: *Phys. Rev. A* 94.4, 041603 (Oct. 2016), p. 041603. arXiv: `1601.05807 [cond-mat.stat-mech]`.

[28] Marco Távora, E. J. Torres-Herrera, and Lea F. Santos. “Power-law decay exponents: A dynamical criterion for predicting thermalization”. In: *Phys. Rev. A* 95 (1 Jan. 2017), p. 013604. URL: `http://link.aps.org/doi/10.1103/PhysRevA.95.013604`.

[29] Marko Znidaric, Antonello Scardicchio, and Vipin Kerala Varma. “Diffusive and Subdiffusive Spin Transport in the Ergodic Phase of a Many-Body Localizable System”. In: *Phys. Rev. Lett.* 117 (4 2016), p. 040601. URL: `http://link.aps.org/doi/10.1103/PhysRevLett.117.040601`.