TensorMixedStates: a Julia library for simulating pure and mixed quantum states using matrix product states

Jérôme Houdayer⋆ and Grégoire Misguich†

Université Paris-Saclay, CNRS, CEA, Institut de Physique Théorique,
91191 Gif-sur-Yvette, France

⋆ jerome.houdayer@ipht.fr
† gregoire.misguich@ipht.fr

Abstract

We introduce TensorMixedStates, a Julia library built on top of ITensor which allows the simulation of quantum systems in presence of dissipation using matrix product states (MPS). It offers three key features: i) it implements the MPS representation for mixed states along with associated operations, in particular the time evolution according to a Lindblad equation or discrete time evolution using non-unitary gates (quantum channels), ii) it is based on ITensor, which has proven its effectiveness and which gives access to efficient low-level tensor manipulation as well state-of-the-art algorithms (like DMRG or TDVP), finally iii) it presents a user-friendly interface allowing writing sophisticated simulations for pure and mixed quantum states in a few lines of code.

Contents

1 Introduction

The field of open quantum many-body problems is a very active area of research in Physics [1]. In the last two decades, there has been huge experimental progress in the manipulation and in the control of quantum systems such as cold atoms [2], Rydberg atoms [3–5], trapped ions [6–8], coupled light-matter systems or superconducting circuits and processors [9, 10] to name a few. Quantum technologies and the development of devices that are able to perform quantum information tasks [11] has clearly been a major driving force in this domain. These systems are never perfectly isolated from their environment, and the presence of noise, dissipation and decoherence is often important [1, 12]. In some situations the presence of the environment can give rise to interesting new phenomena and new dynamical regimes. The environment can even be exploited to engineer useful quantum many-body states [13]. Simulating a quantum many-body problem on a classical computer is a notoriously difficult task because the computational cost is in general exponential in the number of constituents and open quantum systems are generally not simpler [14]. Nevertheless, numerical algorithms where the many-body states are represented (and compressed) using tensor networks have established themselves as among the most powerful for this type of problems [15, 16]. Among these methods, those based on matrix-product states (MPS) have proven to be very successful in many situations, and for low-dimensional systems in particular [17, 18]. In the field of quantum computing, calculations based on MPS have raised the bar concerning the performance that quantum processors must exceed in order to offer a quantum advantage [19–23]. In fact, an external environment tends to decrease the amount of entanglement among the degrees of freedom inside the system, and it often results in a decrease of correlations. This can be a favorable situation for tensor network representations which can exploit the reduced correlations to achieve a better compression of the state.

While there exist several powerful libraries for manipulating pure states with MPS (like ITensor [24] or TenPy [25]), the software offering for the simulation of open/dissipative systems with MPS is much more limited.[1] We attempt here to fill this gap by presenting the TensorMixedStates library [29] (TMS) which allows manipulating *mixed* many-body quantum states in the form of MPS. It is based on the ITensor [24] library in Julia and offers a solver for studying the time evolution of open quantum system described by a Lindblad master equation for the density operator [1, 30, 31]. It also permits constructing and manipulating density matrices using gates or user-defined quantum channels. Note that the library does not implement the unraveling of Lindblad master equations into ensembles of quantum trajectories.

The rest of the paper is organized as follows. We begin in Section 2 by recalling a few basic notions concerning quantum states and their MPS representation. The end of this section also summarizes the functionalities offered by TMS. Section 3 presents the main features of the TMS library. This section describes the general design of the library (3.1), the installation

[1] See for instance [26–28].

procedure (3.2), the way to define the Hilbert space (3.3), the states (3.4) and the operators (3.5). It also presents the main operations that can be performed on these objects: continuous time evolution or discrete gates (3.6) and measurements (3.7). Section (3.8) presents some functions which help monitoring the precision of the simulations, and Section (3.9) illustrates the high-level interface of TMS.

Section 4 illustrates the use of TMS with several physical models: a fermionic chain with dephasing in the bulk (4.1), a spin chain with boundary dissipation (4.2), a bosonic chain with incoherent particle injection (4.3), a fermionic chain with incoherent particle injection (4.4), a model describing the decoherence of a graph state, and a brick wall quantum circuit with noisy gates (4.6). Finally, Section 5 presents conclusions.

2 Context

In this section, we recall a few basic properties and a few notations concerning quantum (pure and mixed) states and their representations with MPS.

2.1 Pure and mixed quantum states

The (pure) states of a closed system form a Hilbert space $\mathcal{H}$, so that a state $|\psi\rangle \in \mathcal{H}$ is a vector. Given an operator acting on $\mathcal{H}$ there are essentially three basic operations that we may consider: i) measuring the expectation value of an observable O (with O hermitian)

$$\langle O\rangle = \langle\psi|O|\psi\rangle, \tag{1}$$

ii) doing some discrete evolution by applying a gate U (with U unitary)

$$|\psi\rangle = U|\psi_0\rangle, \tag{2}$$

or iii) doing continuous-time evolution with the Hamiltonian H (H hermitian)

$$\partial_t|\psi\rangle = -iH|\psi\rangle. \tag{3}$$

For an open system, this formulation is no longer sufficient, and a state must be represented as a density matrix ρ which must be Hermitian, positive semidefinite with unit trace [1]. When the state is pure, ρ is simply a projector

$$\rho = |\psi\rangle\langle\psi| \tag{4}$$

but for general mixed states we have $\mathrm{Tr}\left[\rho^2\right] < 1$. The three operations mentioned above for pure states become

$$\langle O\rangle = \mathrm{Tr}(O\rho), \tag{5}$$
$$\rho = U\rho_0 U^\dagger, \tag{6}$$
$$\partial_t\rho = -i[H,\rho]. \tag{7}$$

Where $[A,B] = AB - BA$ is the commutator. For a mixed state, a general discrete evolution is a linear and completely positive map (also called quantum operation, or quantum channel) and takes the form [32]

$$\rho = \sum_i E_i \rho_0 E_i^\dagger, \tag{8}$$

with Kraus operators $\{E_i\}$ satisfying $\sum_i E_i^\dagger E_i = \mathbb{1}$. In a continuous-time context, the evolution, if Markovian, can instead be modelled by the Gorini–Kossakowski–Sudarshan–Lindblad master equation [30,31]

$$\partial_t \rho = \mathcal{L}(\rho), \tag{9}$$

where $\mathcal{L}$ is the Lindbladian

$$\mathcal{L}(\rho) = -i[H,\rho] + \sum_k \left(L_k \rho L_k^\dagger - \frac{1}{2}\{L_k^\dagger L_k, \rho\} \right), \tag{10}$$

with no particular constraints on the L_k and $\{A,B\} = AB + BA$ is the anti-commutator.

2.2 Matrix product states and operators

In quantum many-body problems the dimension d of $\mathcal{H}$ grows exponentially with the system size and this severely limits the sizes accessible to numerically exact calculations. A possible option is then to use approximate representations of the states, and MPS [17] provide such approximate representations. Suppose $\mathcal{H}$ is a tensor product of N finite-dimensional local Hilbert spaces $\mathcal{H}_i$ of dimension d_i: $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$ (thus $d = d_1 d_2 \cdots d_N$). The system consists of N sites, each associated with a local Hilbert space of dimension d_i. In the pure case, any state $|\psi\rangle$ can be written

$$|\psi\rangle = \sum_{\{s\}} T_1^{s_1} T_2^{s_2} \cdots T_N^{s_N} |s_1 s_2 \cdots s_N\rangle, \tag{11}$$

where $T_i^{s_i}$ is a $\chi_{i-1} \times \chi_i$ matrix and s_i runs over all single site basis states of $\mathcal{H}_i$ (χ_0 and χ_N are set to 1). T_i as a whole can be seen as a 3-index tensor of dimension $\chi_{i-1} \times \chi_i \times d_i$ or alternatively as a $\chi_{i-1} \times \chi_i$ matrix whose elements are quantum states belonging to $\mathcal{H}_i$, in which case we can simply write.

$$|\psi\rangle = T_1 T_2 \cdots T_N, \tag{12}$$

hence the name matrix product state.

For large systems, the *exact* representation of a generic state as an MPS requires most of the bond dimensions χ_i to grow exponentially with N. MPS are particularly useful when there exist matrices T_i of size much smaller than d which provide a good approximation of the target state $|\psi\rangle$. In practice, we set a maximum bond dimension χ, and we numerically approximate the states of interest by a MPS with $\chi_i \leq \chi$. The larger is χ, the larger is the precision of the approximation.

To operate on a MPS, we can build a matrix-product operator (MPO)

$$O = O_1 O_2 \cdots O_N, \tag{13}$$

where the O_i are matrices whose elements are operators acting on $\mathcal{H}_i$, that is the O_i are tensors with four indices (two of which having dimension d_i) and we then have

$$O|\phi\rangle = (O_1 \cdot T_1)(O_2 \cdot T_2) \cdots (O_N \cdot T_N). \tag{14}$$

Simple operators, like single site operators, do not need to be represented as MPOs. They can be directly applied to the corresponding T_i, this is very useful to compute mean values of operators. All this can be nicely represented by tensor diagrams as shown on Fig. 1

What about mixed states? We can see the density matrix ρ of a mixed state as a vectorized state $|\rho\rangle\rangle$ in a larger space of dimension d^2 [1] and apply the same process as before and write

$$\rho = \sum_{\{s\},\{s'\}} R_1^{s_1,s'_1} R_2^{s_2,s'_2} \cdots R_N^{s_N,s'_N} |s_1 s_2 \cdots s_N\rangle\langle s'_1 s'_2 \cdots s_N|, \tag{15}$$

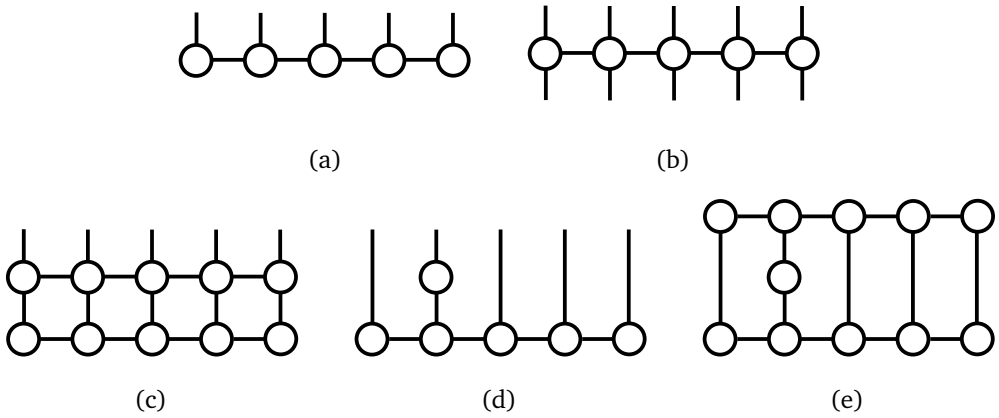

Figure 1: Tensor diagrams for pure states: tensors are represented by circles and indices by lines. A contraction is represented by a line (index) connecting two circles (tensors). (a) MPS with five sites. (b) MPO with five sites. (c) application of a MPO to a MPS. (d) Application of a single site operator. (e) Mean value of a single site operator (MPS represented with downward indices are complex conjugated).

where $R_i^{s_1,s_1'}$ is a $\chi_{i-1}\times\chi_i$ matrix and s_i and s_i' run over all single site basis states of $\mathcal{H}_i$. R_i as a whole is then a 4-index tensor of dimensions $\chi_{i-1}\times\chi_i\times d_i\times d_i$. In the vectorization process $\mathcal{H}_i\otimes\mathcal{H}_i$ is seen as the space of dimension d_i^2 of the operators acting on $\mathcal{H}_i$, and we can also write

$$|\rho\rangle\rangle = \sum_{\{o\}} R_1^{o_1} R_2^{o_2}\cdots R_N^{o_N}|o_1 o_2\cdots o_N\rangle\rangle, \tag{16}$$

where o_i runs over all single site operator basis of $\mathcal{H}_i$ [2] and R_i can be seen as a 3-index tensor of dimension $\chi_{i-1}\times\chi_i\times d_i^2$. We can finally see R_i as $\chi_{i-1}\times\chi_i$ matrix whose elements belong to $\mathcal{H}_i\otimes\mathcal{H}_i$ and simply write

$$|\rho\rangle\rangle = R_1 R_2\cdots R_N. \tag{17}$$

Note that, contrary to the matrix-product density operator representation [33], the representation of Eq. 15 does not guaranty that ρ is positive. In practice, this is not an issue as long as the bond dimension can be taken large enough to ensure a sufficient accuracy for the observables of interest.

For a pure state $|\psi\rangle$, we have $\rho=|\psi\rangle\langle\psi|$ and thus $R_i^{s_i,s_i'} = T_i^{s_i}\otimes T_i^{s_i'}\dagger$ or correspondingly $R_i = T_i\otimes T_i^\dagger$. Note that in this operation χ_i for R is the square of χ_i for T.

If one needs to represent an operator acting on a density matrix ρ, one may use the above strategy with an MPO acting on the vectorized state $|\rho\rangle\rangle$. The O_i are then matrices whose elements are operators acting on the "doubled" Hilbert space. The O_i can also be viewed as tensors with four indices, two of which having dimension $(d_i)^2$. Tensor diagrams for mixed states are shown on Fig. 2.

An important question is the size of the matrices that are required to represent accurately a given state as an MPS. For a pure state $|\psi\rangle$, it is well known that the dimension χ of the bond which separates the left region A from the right region B[3] has to be scaled as the exponential of the bipartite von Neumann entropy $S_{\text{vN}}^{A-B,|\psi\rangle}$ associated to the bipartition. In other words $\ln\chi\sim S_{\text{vN}}^{A-B}$. For a mixed state ρ and a given bipartition $A-B$ of the system, one can consider the (vectorized) pure state $|\rho\rangle\rangle$ and the von Neumann entropy $S_{\text{vN}}^{A-B,|\rho\rangle\rangle}$ associated

[2] In the simplest case where the system is made of qubits ($d_i=2$), the o_i can be chosen as the identity and the three Pauli matrices.

[3] In an MPS a given matrix index naturally separates the system in two subsystems.

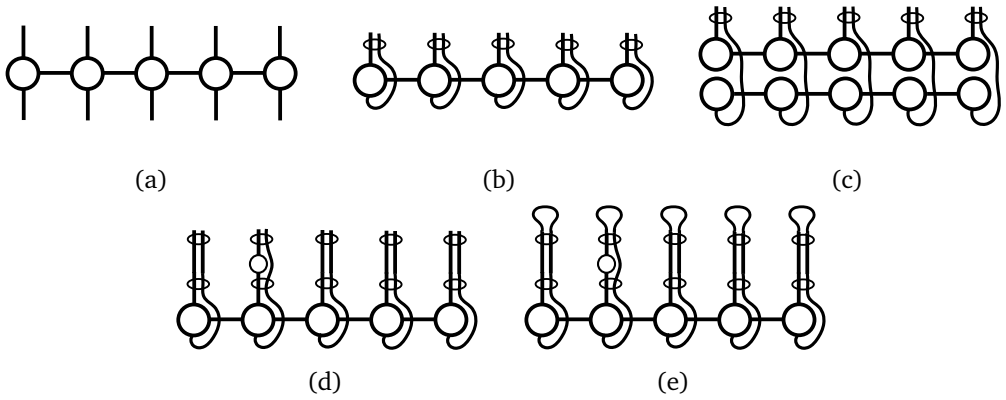

Figure 2: Tensor diagrams for mixed states: two lines surrounded by a circle represent combined indices, that is two indices combined as one. (a) Density matrix ρ with five sites represented as a MPO. (b) Vectorized density matrix $|\rho\rangle\rangle$ represented as a MPS. (c) Vectorized density matrix $|\rho\rangle\rangle$ of a pure state. (d) Application of a single site operator O. (e) Mean value $\mathrm{Tr}[\rho O]$ of a single site operator.

to the bipartition. The operator space entanglement entropy (OSEE) associated to the $A-B$ partition is, by definition, equal to this entanglement entropy [34]: $S^{A-B}_{\mathrm{OSEE}} = S^{A-B,|\rho\rangle\rangle}_{\mathrm{vN}}$. When approximating ρ by a MPS as in Eq. 15, the required bond dimension χ must then scale as the exponential of the OSEE, $\ln\chi \sim S^{A-B}_{\mathrm{OSEE}}$.

2.3 Functionalities offered by TensorMixedStates

The ITensor library provides tools for constructing and manipulating MPSs and MPOs. In particular, it provides the following four main operations: (i) creation of a MPS state representing a pure product state, (ii) creation of a MPO representing an operator acting on pure states, (iii) measurement of the mean value of an operator on a pure state, (iv) powerful algorithms for evolving a state with an operator: in particular applying the operator on the state (MPO / MPS contraction), computing the ground state of an operator (DMRG [17,35]), and solving a continuous equation of the form $\partial_t|\psi\rangle = O|\psi\rangle$ (TDVP [36]).

Unfortunately, ITensor has not been developed with mixed states in mind and while the evolution algorithms (point (iv)) are quite general and can be used in the mixed case, the other three components are not (points (i), (ii) and (iii)). Moreover, even if ITensor is largely extendable, using this expendability to accommodate the new needs was at best awkward, in particular it would have been complicated to use the operator framework of ITensor.

TMS thus rely on the powerful core algorithms of ITensor (tensor contractions, DMRG and TDVP) and reimplement from scratch all the rest to accommodate mixed state representions. Namely, it provides (i) the creation of a MPS representing a mixed product state, (ii) the creation of a MPO representing an operator acting on a mixed state, (iii) measurement of the mean value of an operator on a mixed state.

Recall that in the mixed-state setting there is a wider variety of operator actions to consider. In the pure-state case, an operator O is naturally applied to a state $|\psi\rangle$ by computing $O|\psi\rangle$. By contrast, when working with a density matrix ρ, an operator O can act in several distinct ways, namely $O\rho$, $O\rho O^\dagger$, $[O,\rho]$ and $\{O^\dagger O,\rho\}$. It is this richness that led us to develop a complete and flexible operator framework.

Finally, TMS also provides algorithms not present in ITensor, namely the W^{I} and W^{II} approximations [37] that build a MPO for approximately representing $\exp(tO)$ at small t. These

time-evolution algorithms are complementary to TDVP, in particular for non Hermitian evolutions where TDVP is sometimes less efficient (See for instance Sec. 4.4). TMS also provide an algorithm to compute the steady state of a Lindbladian $\mathcal{L}$, simply computing the ground state of $\mathcal{L}^\dagger\mathcal{L}$ with DMRG.

3 Features

3.1 Design choices

TMS is the successor of the Lindbladmpo library [38, 39]. Lindbladmpo was based on the C++ version of the ITensor library and since the ITensor library has migrated to the Julia language, Lindbladmpo could not benefit from the recent developments of ITensor. In addition, Lindbladmpo was limited to qubits (two-dimensional local Hilbert space on each site) while TMS is much more general.

In this context, we decided to create a more general and more flexible software to address the simulation of open quantum systems with non-unitary evolution. The choice of the Julia language was then natural to ease the interactions with ITensor. Moreover, using Julia helped us develop a more flexible and more user-friendly interface. Finally, the combined use of Julia and ITensor brings multithreading for free.

The main decision concerned the use of the operator and state framework of ITensor. As stressed in the previous section, the difficulty lies in the fact that, for a given operator, one has four different possible actions on ρ. It proved challenging to use ITensor's native structures for that, and we decided to write from scratch a new state and operator framework. This also required writing new functions for creating MPS and MPO from states and operators. While this amounted to a substantial and bookkeeping-heavy implementation effort, it did not involve any choice of nontrivial algorithm or any decision concerning approximation schemes. All algorithmic decisions are delegated to ITensor itself: TMS primarily acts as a wrapper that formats data appropriately for ITensor and forwards the user-specified parameters. Likewise for W^{I} and W^{II} approximations, we implemented the algorithms as described in [37, 40]. Finally, the steady-state computation routine is simply a three-line wrapper around an ITensor DMRG call.

We finally decided for a double interface: (i) a high-level interface allows designing state of state-of-the-art simulations in a few lines of code as demonstrated in Sec. 3.9. (ii) a low-level interface gives access to all the features of the library as we now explain.

3.2 Installation and usage

To use the TMS library, one must first add it to the Julia environment. This is done the usual way in Julia by

```
]add TensorMixedStates
```

Then, to use it in a script, one has to add a "using" clause at the top of the file

```
using TensorMixedStates
```

Note that the examples shown in this article, together with other examples are available with the source on GitHub [29]. Moreover, a full documentation is also available online (see the README.md file [29]).

3.3 Hilbert space

The first step in a quantum simulation consists in describing the Hilbert space. In our case, the Hilbert space must be a finite product over N local finite-dimensional Hilbert subspaces. Those local subspaces can be chosen independently among the "site types" proposed by TMS. At the moment, there are seven possible choices: qubit, boson, fermion, spin, electron, tJ and q-boson. This set is easily extendable (see TMS online documentation for details).

In TMS, the Hilbert space is described with a `System` object

```
# a system with 10 qubits
sys1 = System(10, Qubit())

# an hybrid system with 3 sites
sys2 = System([Qubit(), Boson(4), Fermion()])
```

Note that for bosons it is necessary to truncate the Fock space and the dimension d of the local Hilbert space must be provided as a parameter. In the example above, the second site has $d = 4$, which corresponds to a maximal occupation boson number equal to $d - 1 = 3$.

3.4 States and representations

The starting point for building states are single site states. For each kind of site, predefined states designated by their names are defined. For example, for qubits one has "Up" or "0", "Dn" or "1" and "+" and "-" and more (for a list of all predefined state names, see the online documentation). A state which is not predefined can be represented by a vector of complex numbers such as `[1, 0]`. For mixed representations, there is one predefined state for all site types called "FullyMixed" which corresponds to the infinite temperature state (with $\rho = \mathbb{1}/d$ proportional to the identity matrix). A general local mixed state can be represented by a matrix of complex numbers such as [[0.5 0];[0 0.5]].

In TMS, both pure and mixed representations are hold by `State` objects. All operations on `State` objects will work in the same way, with the same syntax independently of the nature, pure or mixed, of the state (as long as such operations make sense for this representation).

We can build a product state (i.e. $|\psi\rangle = |\psi_1\rangle \cdots |\psi_N\rangle$) with the `State` constructor parametrized by `Pure` or `Mixed` and two arguments: the system and a vector of single site states (either defined by their names or explicitly given as vectors or matrices), for the usual case where all single site states are the same, we can simply give that state. For example

```
# an all up pure state for a 10 qubit system
st1 = State{Pure}(sys1, "Up")

# a mixed state for a hybrid system
st2 = State{Mixed}(sys2, [[1, im]/sqrt(2), "1", "FullyMixed"])
```

As a shortcut, one can define both system and state in one call, for example

```
# an up and down state for a 5 qubit system
st1 = State{Pure}(5, Qubit(), ["Up", "Dn", "Up", "Dn", "Up"])

# an infinite temperature state for a hybrid system
st2 = State{Mixed}([Qubit(), Boson(4), Fermion()], "FullyMixed")
```

In this case the system can be retrieved from the state by accessing the `system` field of the state.

To build more complicated states, one can form linear combinations of states. For example, one can build the GHZ state on the 10 qubit system with

```
ghz = (State{Pure}(sys1,"Up") + State{Pure}(sys1,"Dn")) / sqrt(2)
```

Many functions are defined for simple tasks on `State` objects, for example to get some information on the state: `length` to get the number of sites, `maxlinkdim` to get the maximum bond dimension, `trace` and `trace2` to get $\mathrm{Tr}(\rho)$ and $\mathrm{Tr}(\rho^2)$ and so on. To obtain the mixed state representation of a pure state one can use

```
mixed_state = mix(pure_state)
```

3.5 Operators

In TMS, operators are a key feature. They allow the user to represent any kind of quantum operator to operate on the state. They come in two flavors, *generic* (like `X`) or *indexed* (like `X(3)`). The main difference is that indexed operators are applied on specific sites of the system whereas generic operators are not. For example, `Z` is the Pauli operator σ^z and `Z(3)` is the Pauli operator σ^z applied on site number 3.

For each site type several commonly used operators are predefined. For example, for the qubit space, `X, Y, and Z` are the Pauli operators, `Sp` is the raising operator S^+, `H` is the Hadamard gate, `Swap` is the Swap gate, `Id` is the identity operator. For the boson space, `A` is the annihilation operator, `dag(A)` is the creation operator and `N` is the number operator. See the online documentation for an exhaustive list of predefined operators.

Suppose that `A` and `B` are operators and that `x` is a real or complex number and `i` and `j` integers. Then the following operations are defined: `x*A`, `A/x`: multiplication by a scalar, `A+B`, `A-B`: sum of operators, `A*B`: product of operators, `A^x`: power, `exp(A)`: exponentiation, `dag(A)`: dagger, `A⊗B`, `tensor(A, B)`: tensorial product (the symbol ⊗ is usually obtained by typing `\otimes` in a Julia aware editor), `A(i)`, `A(i, j)`: indexation, `controlled(A)`: controlled gates for qubits (see below), `Dissipator(A)`: Lindblad dissipator operator (see below) and `Gate(A)`: the gate operator (see below).

Hamiltonians can thus be written in one line, for example

$$H = -J \sum_{i=1}^{n-1} X_i X_{i+1} + Y_i Y_{i+1}, \tag{18}$$

is build by

```
hamiltonian = -j * sum(X(i)X(i+1)+Y(i)Y(i+1) for i in 1:n-1).
```

Here we have used the julia `sum` function that adds the given iterator using + and the compact julia product syntax (we can omit "*" in some cases).

One can define Lindblad dissipators with the `Dissipator` function. For example, `Dissipator(Sp)` represents the jump operator toward up for a qubit and

```
dissipators = gamma * (Dissipator(Sp)(1) + Dissipator(Sp)(N))
```

is the sum of two jump operators. It can describe incoherent spin flips on the boundary sites of a qubit chain, at a rate γ. By convention the Linbladian is written

```
lindbladian = -im * hamiltonian + dissipators.
```

The `hamiltonian` term in the above Lindbladian definition represents the (super)operator $\rho \to [H, \rho]$ and each term L_k in `dissipators` represents the (super)operator $\rho \to L_k \rho L_k^\dagger - \frac{1}{2}\{L_k^\dagger L_k, \rho\}$. In the example above, the two dissipators correspond to $\rho \to \gamma\left(S_1^+ \rho S_1^- - \frac{1}{2}\{S_1^- S_1^+, \rho\}\right)$ and $\rho \to \gamma\left(S_n^+ \rho S_n^- - \frac{1}{2}\{S_n^- S_n^+, \rho\}\right)$.

One can define noisy gates (quantum channels) with the `Gate` function, for example

```
K = 0.9 * Gate(Id) + 0.1 * Gate(X)
```

defines an operator which acts as follows on a mixed state ρ:

$$K\rho = 0.9\rho + 0.1 X\rho X. \tag{19}$$

One can define more complex operators such as

$$R_{xy}(\phi) = \exp\left[-i\frac{\phi}{4}(X \otimes X + Y \otimes Y)\right], \tag{20}$$

by simply typing the following

```
Rxy(t) = exp(-im * t / 4 * (X ⊗ X + Y ⊗ Y)).
```

Another example is the Toffoli gate, which can be constructed by

```
toffoli = controlled(controlled(X))
```

Finally, one can also define new operators by giving their matrix with

```
Iswap = Operator{2}("Iswap", [1 0  0 0
                              0 0 im 0
                              0 im 0 0
                              0 0  0 1]).
```

In the instruction above `{2}` is the number of sites on which the operator acts.

3.6 Algorithms

There are four main algorithms that one can use in TMS. First, one can apply an operator O as a quantum gate: $|\psi\rangle \to O|\psi\rangle$ for pure states and $\rho \to O\rho O^\dagger$ for mixed states. This is implemented by

```
new_state = apply(my_gate, old_state; options...),
```

One can also perform time-evolutions. This can be done with a Schrödinger evolution $\partial_t|\psi\rangle = -iH|\psi\rangle$ for pure state, or a with Lindblad evolution for mixed states. The library provides two algorithms to do this, one using TDVP (called `tdvp`) [36] and one using the W^I or W^{II} (called `approx_W`) MPO approximation (see Zaletel *et al.* [37]). The syntax is as follows:

```
lindbladian = -im * hamiltonian + dissipators
new_state = tdvp(-im * hamiltonian, t, old_state; options...)
new_state = tdvp(lindbladian, t, old_state; options...)
new_state = approx_W(-im * hamiltonian, t, old_state; options...)
new_state = approx_W(lindbladian, t, old_state; options...)
```

where `t` is the integration time and the `options` allow one to set parameters such as the integration time step, the level of truncation, or the definition of intermediate measurements (via objects called *observers* similar to those of ITensor).

Note that `tdvp` and `approx_W` can be used with time-dependent Hamiltonians and/or dissipators and that the `approx_W` schemes are available up to the order 4 in the time step τ (leading to a Trotter error which scales as $\mathcal{O}(\tau^5)$ [40]).

Another algorithm is the computation of the ground state using DMRG [17, 35]. It is essentially the algorithm from the ITensor library:

```
energy, ground_state = dmrg(hamiltonian, start_state; options...)
```

Finally, it is also possible to use DMRG for mixed states in order to minimize the "square" $\mathcal{L}^\dagger\mathcal{L}$ of the Lindbladian. This allows one to compute directly the steady-state of the system without the need to perform a long time evolution.[4] This is achieved by

```
energy, s_state = steady_state(lindbladian, start_state; options...)
```

A more thorough explanations of the available options for the different algorithms can be found in the online documentation [29].

3.7 Measurements

Finally, we describe how to obtain the expectation value of an observable on a `State`. For example, one writes

```
measure(state, X(1))
```

A more complicated example would be

```
measure(state, 0.5X(1)Z(3)-im*X(2)Y(4))
```

One can also ask for the set of expectation values $\langle X(1)\rangle \cdots \langle X(N)\rangle$ of an operator X on all sites

```
measure(state, X)
```

or even a correlation matrix

```
measure(state, (X, Y))
```

One can also use predefined functions on `State` like `Trace` for the trace or `Linkdim` for the bond dimension (this set of functions is extendable by the user). One can also make several measures in a single call

```
measure(state,  [X(1), X, (X, Y), Purity, Linkdim])
```

3.8 Following computation precision

As with any simulation software, it is important to assess the precision of a computation. Here in TMS, the representation of mixed states does not guaranty that the density matrix stays Hermitian positive with trace one. While this allows more computational deviations to happen (due to truncations or Trotter errors for instance), it is possible to monitor such deviations. Even if we do not have access to the eigenvalues of the density matrix to check the positivity,

[4] The steady state is the eigenstate of $\mathcal{L}$ associated to the eigenvalue 0, hence it is the zero-energy ground state of the Hermitian operator $\mathcal{L}^\dagger\mathcal{L}$.

we can monitor its trace and its hermiticity. These quantities can in turn be used to adjust the parameters of the simulation (time step, truncation threshold and maximal bond dimension).

There are four predefined observables to do this: `Trace`, `TraceError`, `Hermiticity` and `HermiticityError`. The hermiticity is between 0 and 1, 0 for anti Hermitian and 1 for Hermitian. The error versions measure the deviation from the correct value (which is 1).

TMS also provides the `Trace2` observable which computes $\mathrm{Tr}(\rho^2)$, the purity of the state. Note that if the accumulated errors become really important the presence of spurious negative eigenvalues in ρ can lead to a purity that is apparently larger than 1.

Note that TMS works perfectly with non normalized density matrices, observables are then correctly normalized using the actual trace.

3.9 High-level interface

The high-level interface of the TMS library is accessed via the function `runTMS`. It can be used alone or together with the low-level interface for finer control. The goal of the high-level interface is to be able to design fully fledged simulation with minimal code. All the examples presented in the next section were created using this interface.

The principle is the following: define a sequence of actions to be applied on a state and pass it to `runTMS`. As a first example, consider the following complete script

```
using TensorMixedStates, .Fermions

hamiltonian(n) = -sum(dag(C)(i)C(i+1)+dag(C)(i+1)C(i) for i in 1:n-1)
dissipators(n, gamma) = sum(Dissipator(sqrt(4gamma) * N)(i) for i in 1:n)

sim_data(n, gamma, step) = SimData(
    name = "Fermion tight-binding chain with dephasing noise",
    phases = [
        CreateState(
            type = Mixed(),
            sytem = System(n, Fermion()),
            state = [ iseven(i) ? "Occ" : "Emp" for i in 1:n ]),
        Evolve(
            duration = 4,
            time_step = step,
            algo = Tdvp(),
            evolver = -im*hamiltonian(n) + dissipators(n, gamma),
            limits = Limits(cutoff = 1e-30, maxdim = 100),
            measures = [
                "density.dat" => N,
                "OSEE.dat" => EE(div(n, 2))
            ]
        )
    ]
)

runTMS(sim_data(40, 1., 0.05))
```

The first line brings our library in scope and the definitions for the fermion site type. The next two lines define the evolution operator (more details on the model in Sec. 4.1). Then we have a function definition for `sim_data` to build a `SimData` object corresponding to the parameters of the simulation. The `name` field in `SimData` sets the name of the directory

where the output will be stored. The `phases` field describes the simulation. Here there are two phases: first create a state, second compute the time evolution. The fields names are self-explanatory. The `measures` field describes the data files created by the simulation (here there are two files `density.dat` and `OSEE.dat`) and what quantities to be written in each of them (here the mean fermion occupancy measured on each site and the operator-space entanglement entropy (OSEE) at the middle point of the chain). Note that one can put more than one observable per file. The last line calls `runTMS` with the chosen parameters.

In this case, `runTMS` will create a directory with the given name, run the simulation and put the results in the corresponding files. In addition to the data files, it will also create a "log" file to follow the progress of the simulation and register information and warnings, it will also copy the program script to `prog.jl` for information and reproducibility.

In addition to `CreateState` and `Evolve` there are other available phases: `ToMixed` to turn a pure representation into a mixed representation, `Dmrg` to use the DMRG algorithm, `Gates` to apply gates, `PartialTrace` to trace out some sites and finally `SteadyState` to compute the steady state of a Lindbladian.

As already stated, more detailed information on the syntax and options can be found in the online documentation [29].

4 Examples

This section illustrates the use of the TMS library to study several dissipative quantum problems evolving according to a Lindblad equation. The examples are chosen because they have been the focus of recent studies in the literature and because an exact solution is available.[5] These solutions allow checking quantitatively the numerical results. The first example is a fermionic chain with dephasing noise (Sec. 4.1), the second example is a spin chain with boundary dissipation (Sec. 4.2), the third example is a one-dimensional bosonic model with an incoherent particle source in the center of the chain (Sec. 4.3) and the fourth example (Sec. 4.4) is the fermionic version of the previous bosonic model. The fifth example (Sec. 4.5) describes the dissipation of a complete-graph state in a qubit system. Finally, the model of Sec. 4.6 is a deep quantum circuit with unitary 2-qubit gates as well as dissipative channels which model qubit errors.

A runnable and well commented version of the code of these examples is available in the `examples/article` section of the repository on GitHub [29].

4.1 Fermion tight-binding chain with dephasing noise

We consider here the one-dimensional spinless fermion model studied in [41]. The initial state is a pure state where the even sites are occupied, and the odd ones are empty. The system then evolves under the action of a nearest-neighbor hopping Hamiltonian

$$H = -\sum_{i=1}^{N-1} \left(c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right) \tag{21}$$

as well as under the following "dephasing" jump operators:

$$L_i = \sqrt{4\gamma} n_i, \tag{22}$$

where $n_i = c_i^\dagger c_i$ is the fermion number operator on site i. Using the high-level interface, we combine the Hamiltonian and the jump operators into an `evolver` (see Sec. 3.9 for the full listing)

[5] Except for the model in Sec. 4.6.

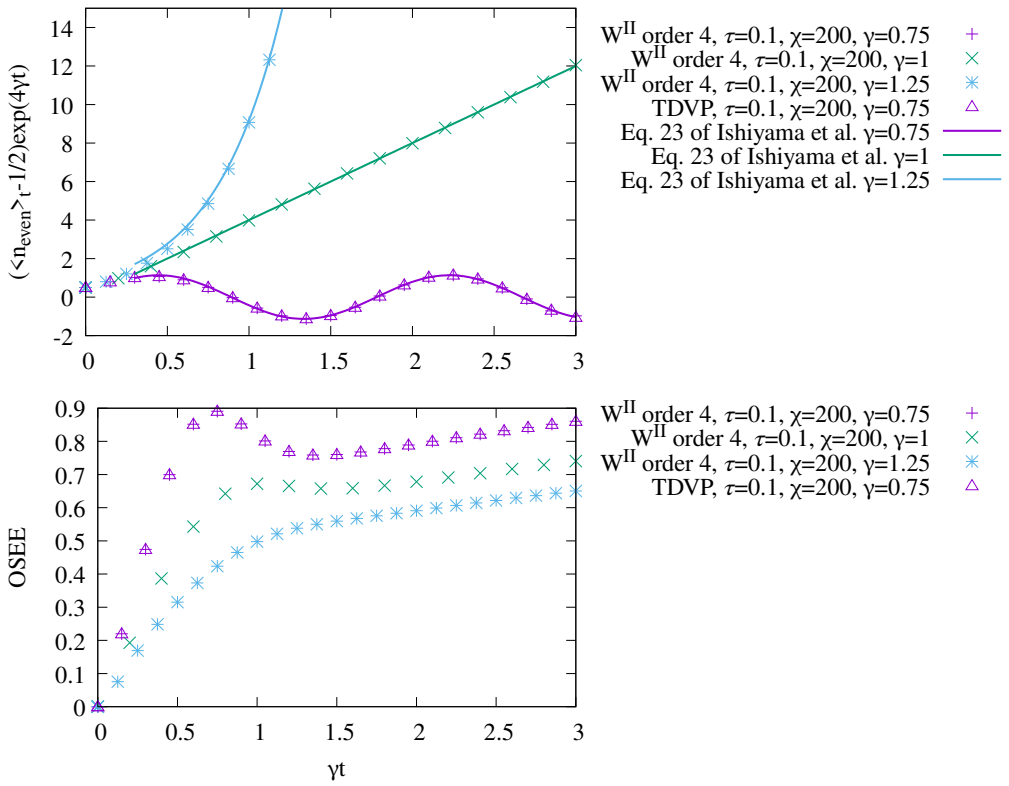

Figure 3: Top: fermion density $\langle n_i \rangle$(t) on even sites i as a function of time in a tight binding chain with dephasing noise and three different strengths of the noise ($\gamma = 0.75, 1.0, 1.25$). The density is evaluated by averaging over 4 sites in the center of the system. Solid lines indicate the exact asymptotic results of Ref. [41]. Bottom: OSEE as a function of time, computed for the bipartition in the middle of the chain. The simulations have been carried out with the W^{II} algorithm at order 4 and with TDVP (see legend).

```
hamiltonian(n) = -sum(dag(C)(i)C(i+1)+dag(C)(i+1)C(i) for i in 1:n-1)
dissipators(n, gamma) = sum(Dissipator(sqrt(4gamma)*N)(i) for i in 1:n)
```

Dissipative problems where the Hamiltonian and the Lindbladian are quadratic in the creation and annihilation operators can be solved exactly [42–45]. The present model was recently studied in the limit of an infinite chain [41]. The authors of this study demonstrated that the system displays oscillatory decay or over-damped decay, depending on the strength γ of the dissipation. In Fig. 3, numerical results for the fermion density, obtained with the present library, are compared to the exact asymptotic results derived from [41]. The fermion density converges to $1/2$ at long times. To magnify how such convergence occurs the vertical axis represents deviation from $1/2$ multiplied by an exponential factor $\exp(4\gamma t)$. We observe a good agreement between the simulations and the analytical behavior derived in [41].

The bottom panel of Fig. 3 represents the evolution of the OSEE [34, 46] associated with the central bipartition of the chain. The OSEE quantifies the total amount of correlations between the two subsystems. This quantity is very useful in the context of MPS since the bond dimension χ (on the bond associated to the bipartition) required to represent ρ faithfully is expected to obey a scaling of the form $\ln(\chi) \sim$ OSEE.

4.2 XX spin chain with boundary dissipation

We illustrate here the use of the TMS library to simulate the dynamics of an open spin-$\frac{1}{2}$ chain with Lindblad terms acting at its boundaries. The Hamiltonian is the so-called XX model

$$H = \sum_{i=1}^{N-1} \left(\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y\right), \tag{23}$$

and the dissipation is due to four Lindblad operators acting and both ends of the chain:

$$L_1 = \sqrt{\varepsilon_{\mathrm{L}} \frac{1+\mu_{\mathrm{L}}}{2}} \sigma_1^+, \quad L_3 = \sqrt{\varepsilon_{\mathrm{R}} \frac{1+\mu_{\mathrm{R}}}{2}} \sigma_N^+, \tag{24}$$

$$L_2 = \sqrt{\varepsilon_{\mathrm{L}} \frac{1-\mu_{\mathrm{L}}}{2}} \sigma_1^-, \quad L_4 = \sqrt{\varepsilon_{\mathrm{R}} \frac{1-\mu_{\mathrm{R}}}{2}} \sigma_N^-, . \tag{25}$$

$\sigma^{\pm} = (\sigma^x \pm i\sigma^y)/2$, $\varepsilon_{\mathrm{L,R}}$ are the strengths of the coupling between the spin chain and the reservoirs at both ends. $\mu_{\mathrm{L,R}}$ are the magnetization of each reservoir. Coding such model with TMS can be done by defining the following `evolver`:

```
hamiltonian(n) = sum(X(i)*X(i+1)+Y(i)*Y(i+1) for i in 1:n-1)
dissipators(n, eL, muL, eR, muR) =
  Dissipator(sqrt(eL*(1+muL)/2)*Sp)(1) +
  Dissipator(sqrt(eL*(1-muL)/2)*Sm)(1) +
  Dissipator(sqrt(eR*(1+muR)/2)*Sp)(n) +
  Dissipator(sqrt(eR*(1-muR)/2)*Sm)(n)
```

Via the Jordan-Wigner transformation the Hamiltonian above maps to a quadratic fermionic Hamiltonian and the Lindblad terms become linear in the fermionic creation and annihilation operators. This model is thus said to be quasi-free [45,47] and can be solved exactly. The dynamics of this model was studied in Ref. [48] (see also [42] for the exact steady-state). Fig. 4 displays the time evolution of two observables: the mean magnetization $\langle\sigma_1^x\rangle$ on the first spin and the mean spin current $\langle\sigma_1^x\sigma_2^y - \sigma_1^x\sigma_2^y\rangle$ on the first bond. The data is in agreement with the results of Ref. [48].

4.3 Free bosons with a localized source

We show here how the library can be used to simulate a dissipative system with bosonic degrees of freedom. The unitary part of the dynamics is generated by a free (quadratic) boson Hamiltonian on a chain:

$$H = \sum_{i=-N/2+1}^{N/2-1} (b_i^\dagger b_{i+1} + b_{i+1}^\dagger b_i) \tag{26}$$

The model contains a single Lindblad term which acts as a particle source at center (site $i = 0$) of the chain, at a rate parameterized by Γ:[6]

$$L_0 = \sqrt{2\Gamma}\, b_0^\dagger. \tag{27}$$

Coding such model can be done by defining the following `evolver`:

[6] Compared with Eq. 2 of Ref. [49], the factor 2 in the equation below comes from a different normalization used in their Lindblad equation.

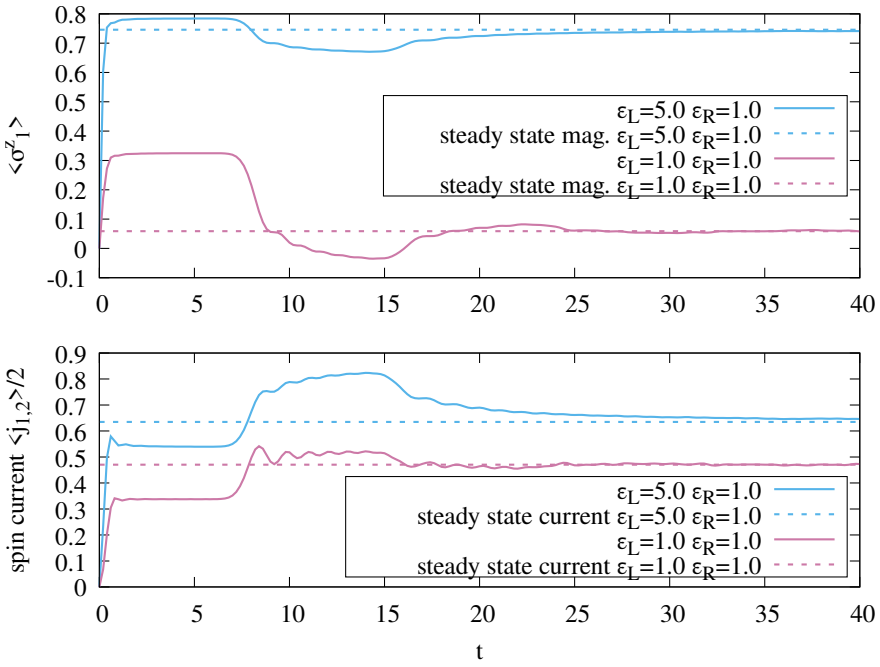

Figure 4: XX spin chain with boundary dissipation (Eqs. 23-25) Top: mean magnetization $\langle\sigma_1^x\rangle$ on the first spin as a function of time. Bottom: mean spin current $\langle\sigma_1^x\sigma_2^y - \sigma_1^x\sigma_2^y\rangle$ on the first bond. Physical parameters: infinite-temperature initial state, system size $N = 30$, $\mu_L = -\mu_R = 1.0$, $\varepsilon_R = 1.0$. Green curves: $\varepsilon_L = 5.0$, red curves: $\varepsilon_L = 1.0$. Simulation parameters: maximum bond dimension $\chi = 300$, time step $\tau = 0.1$, algorithm: W^{II} at order 4. These curves should be compared with Fig. 4 of [48].

```
hamiltonian(n) = sum(A(i)dag(A)(i+1)+dag(A)(i)A(i+1) for i in 1:n-1)
dissipators(n, Gamma) = Dissipator(sqrt(2Gamma) * dag(A))(div(n, 2))
```

This model is quasi-free and has been studied analytically by Krapivsky *et al.* [49] in the case where the chain is empty at $t = 0$. In dimension one, the model displays a phase transition separating a regime ($\Gamma < 2$) where the total number of bosons $N(t) = \sum_i \langle b_i^\dagger b_i \rangle$ grows quadratically and a regime ($\Gamma > 2$) where $N(t)$ grows exponentially. We illustrate here the use of the TMS library to study the small Γ regime. To perform the simulation of this model one has to specify the maximum boson occupation of the sites. To set the maximum occupation to 4 throughout the system and to start with an empty state in a mixed state representation, one can write:

```
CreateState(
  type = Mixed(),
  system = System(n, Boson(5))
  state =  "0"
)
```

In Fig. 5 the simulation results are compared with the exact solution of the model. The exact solution is obtained by solving a set of N^2 linear differential equations for the quantities $\langle b_i^\dagger b_j \rangle$. These equations are given in Eq. 10 of [49]. Two quantities are displayed in Fig. 5: the density profile $\langle b_i^\dagger b_i \rangle$ (top panel) at time $t = 1$ and $t = 5$, and the mean number of boson $N(t)$ (bottom panel). This simulation carried is out using the W^{II} algorithm at order 4 with a time step $\tau = 0.1$ and a maximum bond dimension $\chi = 200$, and it appears to accurately describe the dynamics up to time $t \simeq 5$. Due to the large local Hilbert space dimension of such a bosonic system is however not straightforward to obtain accurate results at longer times. For this reason, checking the asymptotic results derived analytically in [49] would require much longer simulations (the present calculation for the bosonic model, up to time $T = 5$, with $\chi = 200$ and $dim = 5$ already required 100 hours on CPU time with 8 threads running in parallel).

4.4 Free fermions with a localized source

The model considered in this section is the fermionic analog of the previous model. The Hamiltonian describes spinless fermions hopping on the chain

$$H = \sum_{i=-N/2+1}^{N/2-1} (c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i) \tag{28}$$

and the particle injection in the center of the chain is due to the following jump operators

$$L_0 = \sqrt{2\Gamma}\, c_0^\dagger. \tag{29}$$

As for the model of Sec. 4.3, the model is quasi-free, and it has been studied analytically [49]. The figure 6 displays the density profile at three different times, the time evolution of the OSEE (for a bipartition in the center of the system), and the error on the trace of $\rho(t)$. The density profiles $\langle n_i(t) \rangle$ are compared with the exact solution.[7]

The middle and the bottom panels of Fig. 6 allow to compare the precision of the TDVP and W^{II} and to see the influence of time step τ and maximum bond dimension χ. In this example

[7] This solution was obtained by solving numerically the set of N^2 differential equations describing the evolution of the two-point correlations $\mathrm{Tr}\left[\rho c_i^\dagger c_j\right]$, see Eqs. 17 of Ref. [50]. Up to signs these equations are very similar to those describing the dynamics of the 2-point correlations in the boson model of Sec. 4.3.

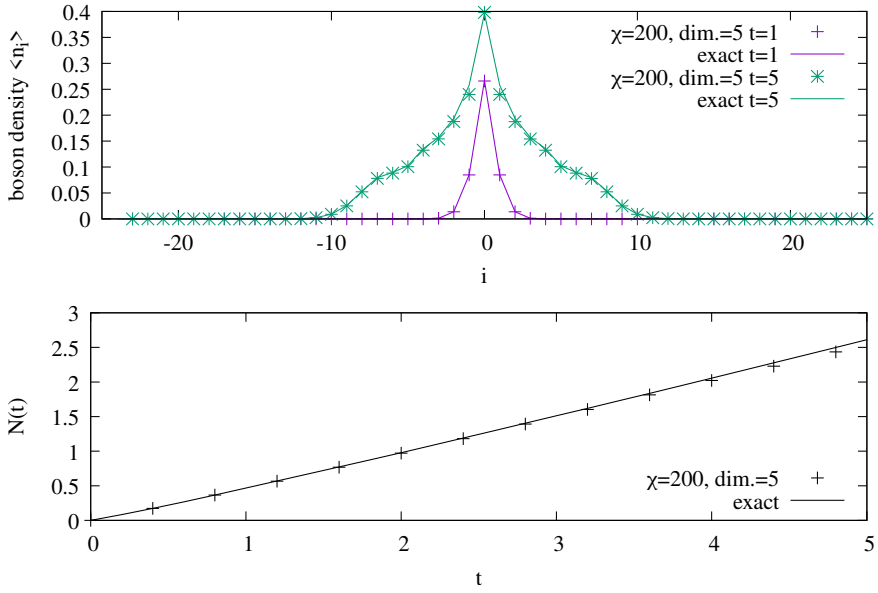

Figure 5: Free boson model with a localized source (Eqs. 26-27). Top: density profile $\langle b_i^\dagger b_i \rangle$ Bottom: mean total number of bosons $N(t)$, numerics versus exact result. Physical parameters: system size $N = 50$, $\Gamma = 0.2$. Simulation parameters: maximum bond dimension $\chi = 200$, local dimension: dim=5 (boson occupancy ≤ 4), time step $\tau = 0.1$, algorithm: W^{II} at order 4.

where the particle injection rate is $\Gamma = 0.2$ the all simulations are quantitatively accurate up to $t \simeq 5$. Beyond that time errors begin to be visible. Among the different simulations, the most accurate one is the one corresponding to $\chi = 200$ with TDVP (blue squares in Fig. 6). Using an even larger bond dimension would allow to describe the dynamics of the model at longer times.

Note that when an exact solution is not available, the error on the trace of ρ (deviations from $\mathrm{Tr}[\rho] = 1$) can be used to estimate at which time the simulations are no longer accurate enough. For this particular problem the algorithms W^{II} at order 4 and TDVP turn out to offer a similar precision. It should be noted however that the execution time is much longer in the case of TDVP: the simulation up to time $T = 8$ with W^{II} at order 4, $\tau = 0.2$ and $\chi = 200$ took 20 minutes on a CPU with 10 threads running in parallel, while the TDVP simulation with $\tau = 0.1$ (giving a similar precision as W^{II} with $\tau = 0.2$) took 340 minutes on the same processor.

4.5 Decoherence of a complete-graph state

The model presented in this section involves N qubits that are initialized in a complete graph state and which evolution is defined by a Lindblad equation only with dissipators (no Hamiltonian). For this model the time-dependence of numerous observables is known analytically, as discussed in Ref. [51].

A graph state [52, 53] is a pure and entangled state that is constructed by the application of controlled-Z (CZ) two-qubit gates to a product state:

$$|g_E\rangle = \prod_{(i,j)\in E} \mathrm{CZ}(i,j)|++\cdots+\rangle, \tag{30}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the product runs over the edges of a graph E. Here we are dealing with the complete graph case where all possible edges are present in E. The construction

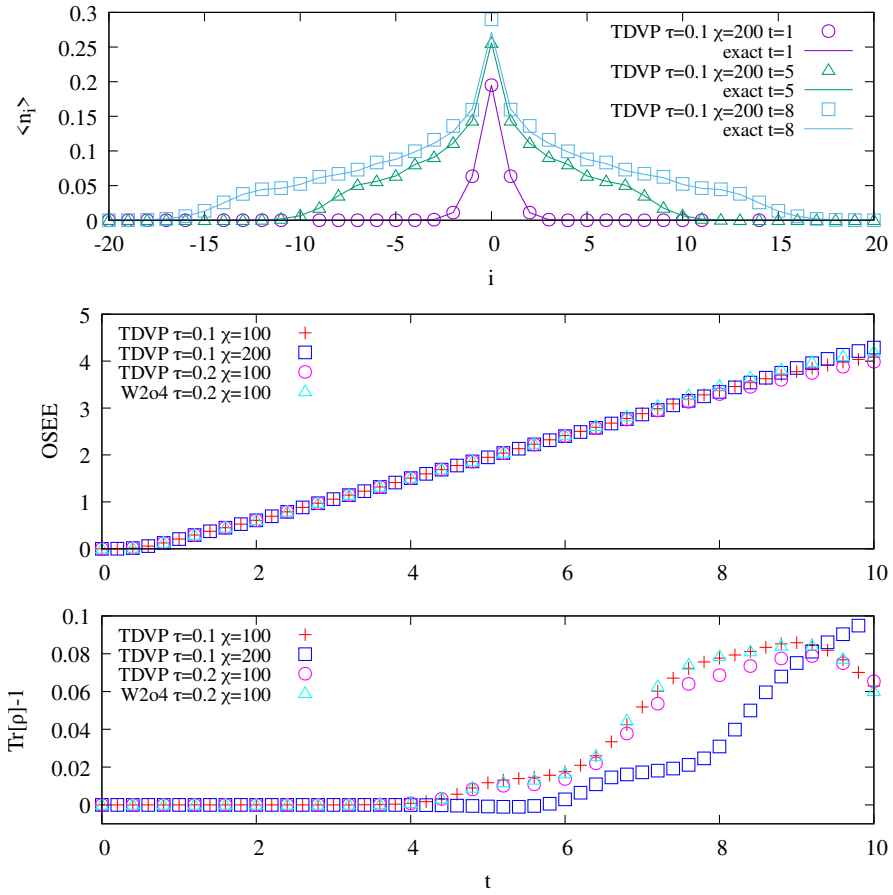

Figure 6: Free fermion model with a localized source (Eqs. 28-29). Top: density profile $\langle n_i(t)\rangle$ at three different times ($t = 1$, 5 and 8). The full lines are exact results and the symbols have been obtained with TDVP with Trotter time step $\tau = 0.1$, maximum bond dimension $\chi = 200$. System size: $N = 50$. At $t = 1$ and $t = 5$ the simulation reproduces almost perfectly the exact profiles. At time $t = 8$ some discrepancy starts to be visible in the center of the profile and at the injection site $i = 0$ in particular. Middle: linear growth of the OSEE taken in the middle bipartition. The different symbols correspond to simulations with different parameters or different algorithms (W^{II} at order 4 and TDVP). τ is the Trotter time step and χ the maximum bond dimension. Bottom: accumulated trace error as a function to time for different simulation parameters.

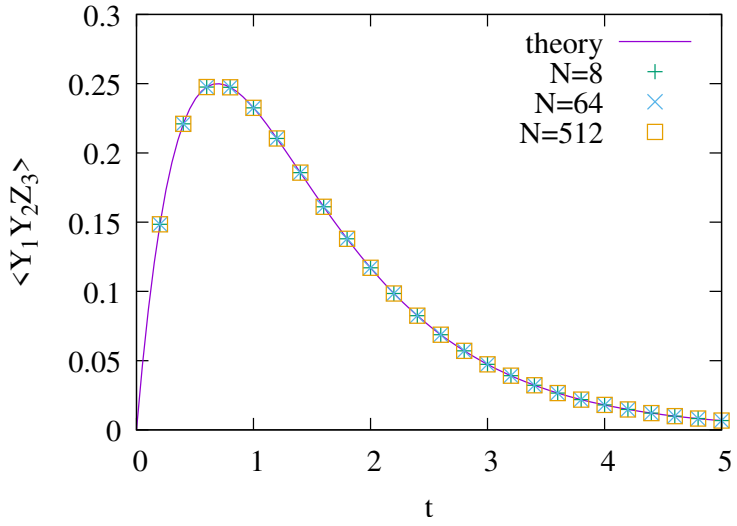

Figure 7: Time evolution of $\langle Y_1 Y_2 Z_3 \rangle$ in a model describing the decoherence of a complete-graph state for sizes 8, 64 and 512. The strength of the dissipation corresponds to $g_0 = 1.0$ in the notation of [51]. The solid line shows the exact result (Eq. 31 of [51]). Due to the permutation symmetry of this model the expectation value $\langle Y_i Y_j Z_k \rangle$ does not depend on i, j and k as long as they are different.

of such initial state (in pure representation) can be implemented as follows:

```
state = create_graph_state(complete_graph(n))
```

In the present model the dynamics is generated by the following dissipators

$$L_i = \sigma_i^+ \,,\; i = 1 \cdots N \tag{31}$$

which are operators bringing the individual spins toward the state $|\uparrow\rangle$. For this problem correlations turn out to be relatively low and the density matrix $\rho(t)$ can be represented by an MPS of bond dimension at most equal to 4 [51]. TMS is thus able to manage very large systems (here we went up to $N = 512$). Moreover, as the Lindbladian is only composed of single-site operators, the W^{II} approximation is exact and arbitrarily large time steps can be used without any Trotter error.[8] Comparisons of numerical results produced by TMS to exact results from Ref. [51] are shown on Fig. 7 for a 3-site observable $\langle Y_1 Y_2 Z_3 \rangle$.

4.6 Noisy quantum circuit

We present here an example which illustrates how the library can be used to perform calculations on quantum circuits. This example is different from the previous ones since it is not associated to a continuous-time evolution. The circuit we consider for this example has a brick wall structure and is similar to the circuits encountered when dealing with a Trotterized (discretized) Hamiltonian evolution.

Let us consider an even number of qubits N and a quantum circuit that is built from successive layers of unitary two-qubit gates as well as layers representing dissipative processes (or qubit errors). The first circuit layer, L_{XX}, is unitary and is defined as a product of (commuting) 2-qubit gates:

$$L_{XX} = U_{XX}^{\phi}(1,2) U_{XX}^{\phi}(3,4) \cdots U_{XX}^{\phi}(N-1,N) \tag{32}$$

[8] Note that since here the Lindbladian is a sum of single-site operators only, one could in principle construct the exact propagator $\exp(t\mathcal{L})$ as a product of local operators. The use of TMS with the W^{II} algorithm implements this automatically.

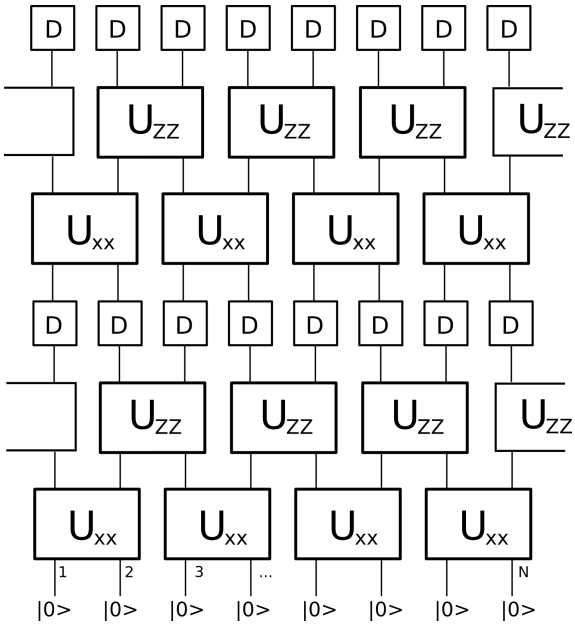

Figure 8: Brick wall quantum circuit made from layers of $U_{XX}^{\phi}(i,i+1)$ gates (Eq. 32), from layers of $U_{ZZ}^{\phi}(i,i+1)$ gates (Eq. 34) and from layers of depolarization gates (Eq. 36).

where $U_{XX}^{\phi}(i,j)$ acts on the qubits i and j and is defined by

$$U_{XX}^{\phi}(i,j) = \exp(i\phi X_i X_j). \tag{33}$$

The next layer, L_{ZZ}, is also defined by a product of (commuting) 2-qubit gates:

$$L_{ZZ} = U_{ZZ}^{\phi}(N,1)U_{XX}^{\phi}(2,3)\cdots U_{XX}^{\phi}(N-2,N-1) \tag{34}$$

where $U_{ZZ}^{\phi}(i,j)$ is defined by

$$U_{ZZ}^{\phi}(i,j) = \exp(i\phi Z_i Z_j). \tag{35}$$

The layers L_{XX} and L_{ZZ} do not commute with each other and create entanglement. After that we apply the gates that model qubit errors. Consider the depolarization channel:

$$D_{i,p} : \rho \to \left(1-\frac{3}{4}p\right)\rho + \frac{1}{4}pX_i\rho X_i + \frac{1}{4}pY_i\rho Y_i + \frac{1}{4}pZ_i\rho Z_i \tag{36}$$

where the parameter $0 \le p \le 1$ represents an error probability. The third layer of the circuit is the product of the depolarization channels for all qubits:

$$L_\epsilon = \prod_i D_{i,p} \tag{37}$$

Finally, the full circuit is a repetition $(L_\epsilon L_{ZZ} L_{XX})(L_\epsilon L_{ZZ} L_{XX})\cdots(L_\epsilon L_{ZZ} L_{XX})$ and the initial state is all qubits in state $|0\rangle$. The circuit as a brick wall structure, as illustrated in Fig. 8.

With TMS the needed operators are easily defined by

```
# Depolarization channel
DPL(p) = (1 - 0.75p) * Gate(Id) +
         0.25p * Gate(X) + 0.25p * Gate(Y) + 0.25p * Gate(Z)
# Ising coupling gates
Rxx(ϕ) = exp(-im * ϕ * X ⊗ X)
Rzz(ϕ) = exp(-im * ϕ * Z ⊗ Z)
```

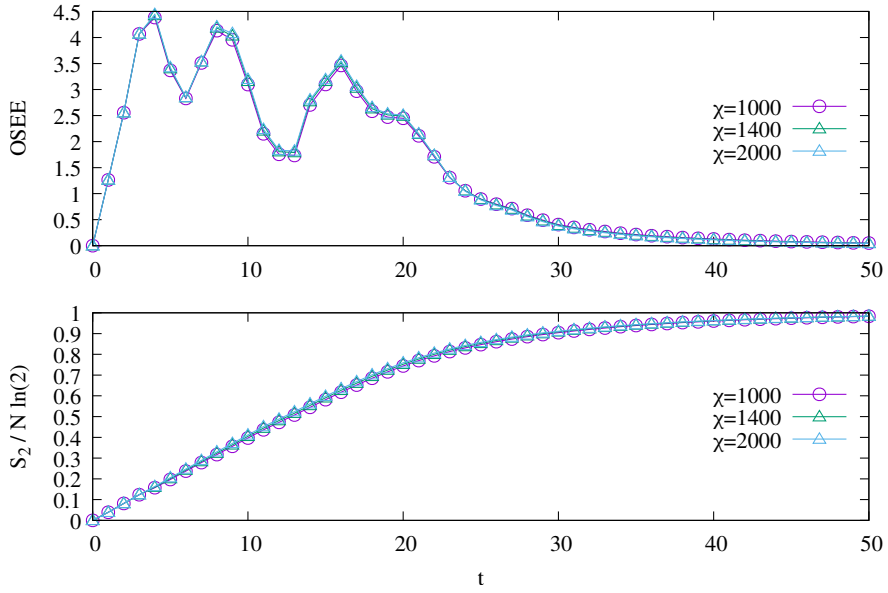

Figure 9: Circuit simulation. Top: OSEE as a function of the number t of layers ($L_\epsilon L_{ZZ} L_{XX}$ counts as one complete layer). Bottom panel: second Rényi entropy S_2 normalized by its maximum value $N \ln 2$. System size: $N = 20$. Error rate $p = 0.02$ and gate angle $\phi = 0.5$. Simulations with bond dimension $\chi = 1000$, 1400 and 2000.

And the gate sequence can be described by

```
[[Gates(
    name = "Applying exp(I*XX*ϕ) gates on qubits [1,2],[3,4],...",
    gates = prod(Rxx(ϕ)(2i-1,2i) for i in 1:div(n, 2)),
    limits = limits,
    ),
  Gates(
    name = "Applying exp(I*ZZ*ϕ) on qubits [N,1],[2,3],[4,5],...",
    gates = Rzz(ϕ)(1,n) *
            prod(Rzz(ϕ)(2i, 2i+1) for i in 1:(div(n,2))-1),
    limits = limits,
    ),
  Gates(
    name = "Depolarization channel on all qubits",
    final_measures = output(n),
    gates = prod(DPL(p)(i) for i in 1:n),
    limits = limits,
    )
] for _ in 1:steps]
```

The top panel of Fig. 9 represents the evolution of the OSEE for a partition in the center of the system as a function of the number of layers in the circuit. After an initial growth, due to the spread of correlations, the effect of the noise takes over when the number of layers become large. The state of the system then approaches an uncorrelated product state (with maximum Rényi-2 entropy). Due to the large amount of correlations generated by the initial layers of the circuit a relatively large bond dimension $\chi \sim 2000$ is required to get converged results.

5 Conclusion

We have presented TensorMixedStates, a Julia library for manipulating pure and mixed quantum states using matrix product state representations. This library allows in particular to apply unitary or non-unitary gates, as well as solving continuous evolution equations such as the Schrödinger or the Lindblad equation. Based on ITensor, this library gives access to state-of-the-art algorithms such as TDVP or DMRG and MPS compression. Moreover, the particularly flexible and user-friendly interface allows simulations to be set up in a few lines of code. We provided six examples to show the versatility and correctness of the software. Five of the examples involved solving the Lindblad equation: two on fermions, one on bosons and two on spin 1/2. The last example demonstrated the use of non-unitary gates in a noisy quantum circuit calculation.

In the near future, we intend to work on several further developments. These include in particular: (i) the use of conserved quantum number (QNS) capabilities of ITensor to improve the efficiency and precision of certain simulations by taking into account conserved quantities of operators, and (ii) use automated MPO compression to improve the performance.

Acknowledgements

We thank Haggai Landa for some previous collaborations on closely related topics. We also thank Pierre Cussenot for some discussions and feedbacks.

Funding information This work is supported by France 2030 under the French National Research Agency grant number ANR-22-QMET-0002 and by the PEPR integrated project EPiQ ANR-22-PETQ-0007.

References

[1] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems*, In *The Theory of Open Quantum Systems*, p. 0. Oxford University Press, ISBN 978-0-19-921390-0, doi:10.1093/acprof:oso/9780199213900.002.14006 (2007).

[2] I. Bloch, J. Dalibard and S. Nascimbène, *Quantum simulations with ultracold quantum gases*, Nature Physics **8**(4), 267 (2012), doi:10.1038/nphys2259.

[3] M. Saffman, T. G. Walker and K. Mølmer, *Quantum information with Rydberg atoms*, Rev.Mod.Phys. **82**(3), 2313 (2010), doi:10.1103/RevModPhys.82.2313.

[4] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić and M. D. Lukin, *Probing many-body dynamics on a 51-atom quantum simulator*, Nature **551**(7682), 579 (2017), doi:10.1038/nature24622.

[5] A. Browaeys and T. Lahaye, *Many-body physics with individually controlled Rydberg atoms*, Nature Physics **16**(2), 132 (2020), doi:10.1038/s41567-019-0733-z.

[6] R. Blatt and C. F. Roos, *Quantum simulations with trapped ions*, Nature Physics **8**(4), 277 (2012), doi:10.1038/nphys2252.

[7] C. D. Bruzewicz, J. Chiaverini, R. McConnell and J. M. Sage, *Trapped-Ion Quantum Computing: Progress and Challenges*, Applied Physics Reviews **6**(2), 021314 (2019), doi:10.1063/1.5088164, 1904.04178.

[8] C. Monroe, W. C. Campbell, L.-M. Duan, Z.-X. Gong, A. V. Gorshkov, P. W. Hess, R. Islam, K. Kim, N. M. Linke, G. Pagano, P. Richerme, C. Senko *et al.*, *Programmable quantum simulations of spin systems with trapped ions*, Reviews of Modern Physics **93**(2), 025001 (2021), doi:10.1103/RevModPhys.93.025001.

[9] M. H. Devoret and R. J. Schoelkopf, *Superconducting circuits for quantum information: an outlook*, Science **16**(5364), 1169 (2013), doi:10.1126/science.1231930.

[10] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen *et al.*, *Quantum supremacy using a programmable superconducting processor*, Nature **574**(7779), 505 (2019), doi:10.1038/s41586-019-1666-5.

[11] J. Preskill, *Quantum Computing in the NISQ era and beyond*, Quantum **2**, 79 (2018), doi:10.22331/q-2018-08-06-79.

[12] A. Morvan, B. Villalonga, X. Mi, S. Mandrà, A. Bengtsson, P. V. Klimov, Z. Chen, S. Hong, C. Erickson, I. K. Drozdov, J. Chau, G. Laun *et al.*, *Phase transitions in random circuit sampling*, Nature **634**(8033), 328 (2024), doi:10.1038/s41586-024-07998-6.

[13] R. Fazio, J. Keeling, L. Mazza and M. Schirò, *Many-Body Open Quantum Systems*, doi:10.48550/arXiv.2409.10300 (2024).

[14] H. Weimer, A. Kshetrimayum and R. Orús, *Simulation methods for open quantum many-body systems*, Rev. Mod. Phys. **93**(1), 015008 (2021), doi:10.1103/RevModPhys.93.015008.

[15] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics **349**, 117 (2014), doi:10.1016/j.aop.2014.06.013.

[16] R. Orús, *Tensor networks for complex quantum systems*, Nat Rev Phys **1**(9), 538 (2019), doi:10.1038/s42254-019-0086-7.

[17] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics **326**(1), 96 (2011), doi:10.1016/j.aop.2010.09.012.

[18] J. I. Cirac, D. Pérez-García, N. Schuch and F. Verstraete, *Matrix product states and projected entangled pair states: Concepts, symmetries, theorems*, Rev. Mod. Phys. **93**(4), 045003 (2021), doi:10.1103/RevModPhys.93.045003.

[19] Y. Zhou, E. M. Stoudenmire and X. Waintal, *What Limits the Simulation of Quantum Computers?*, Phys. Rev. X **10**(4), 041038 (2020), doi:10.1103/PhysRevX.10.041038.

[20] T. Ayral, T. Louvet, Y. Zhou, C. Lambert, E. M. Stoudenmire and X. Waintal, *Density-Matrix Renormalization Group Algorithm for Simulating Quantum Circuits with a Finite Fidelity*, PRX Quantum **4**(2), 020304 (2023), doi:10.1103/PRXQuantum.4.020304.

[21] T. Begušić, J. Gray and G. K.-L. Chan, *Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance*, Science Advances **10**(3), eadk4321 (2024), doi:10.1126/sciadv.adk4321.

[22] E. Stoudenmire and X. Waintal, *Opening the Black Box inside Grover's Algorithm*, Phys. Rev. X **14**(4), 041029 (2024), doi:10.1103/PhysRevX.14.041029.

[23] A. D. King, A. Nocera, M. M. Rams, J. Dziarmaga, R. Wiersema, W. Bernoudy, J. Raymond, N. Kaushal, N. Heinsdorf, R. Harris, K. Boothby, F. Altomare *et al.*, *Computational supremacy in quantum simulation*, doi:10.48550/arXiv.2403.00910 (2024).

[24] M. Fishman, S. R. White and E. M. Stoudenmire, *The ITensor Software Library for Tensor Network Calculations*, SciPost Phys. Codebases p. 4 (2022), doi:10.21468/SciPostPhysCodeb.4.

[25] J. Hauschild, J. Unfried, S. Anand, B. Andrews, M. Bintz, U. Borla, S. Divic, M. Drescher, J. Geiger, M. Hefel, K. Hémery, W. Kadow *et al.*, *Tensor network Python (TeNPy) version 1*, SciPost Phys. Codebases p. 41 (2024), doi:10.21468/SciPostPhysCodeb.41.

[26] J. Gray, *quimb: a python library for quantum information and many-body calculations*, Journal of Open Source Software **3**(29), 819 (2018), doi:10.21105/joss.00819.

[27] G. Torlai and M. Fishman, *PastaQ: A package for simulation, tomography and analysis of quantum computers*, github.com/GTorlai/PastaQ.jl (2020).

[28] T. Lacroix, B. Le Dé, A. Riva, A. J. Dunnett and A. W. Chin, *MPSDynamics.jl: Tensor network simulations for finite-temperature (non-Markovian) open quantum system dynamics*, The Journal of Chemical Physics **161**(8), 084116 (2024), doi:10.1063/5.0223107.

[29] *github.com/jerhoud/TensorMixedStates.jl*.

[30] G. Lindblad, *On the generators of quantum dynamical semigroups*, Commun. Math. Phys. **48**(2), 119 (1976), doi:10.1007/BF01608499.

[31] V. Gorini, A. Kossakowski and E. C. G. Sudarshan, *Completely positive dynamical semigroups of N-level systems*, J. Math. Phys. **17**(5), 821 (1976), doi:10.1063/1.522979.

[32] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, doi:10.1017/CBO9780511976667 (2010).

[33] F. Verstraete, J. J. García-Ripoll and J. I. Cirac, *Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems*, Phys. Rev. Lett. **93**(20), 207204 (2004), doi:10.1103/PhysRevLett.93.207204.

[34] T. Prosen and I. Pižorn, *Operator space entanglement entropy in a transverse Ising chain*, Phys. Rev. A **76**(3), 032316 (2007), doi:10.1103/PhysRevA.76.032316.

[35] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett. **69**(19), 2863 (1992), doi:10.1103/PhysRevLett.69.2863.

[36] M. Yang and S. R. White, *Time-dependent variational principle with ancillary Krylov subspace*, Phys. Rev. B **102**(9), 094315 (2020), doi:10.1103/PhysRevB.102.094315.

[37] M. P. Zaletel, R. S. K. Mong, C. Karrasch, J. E. Moore and F. Pollmann, *Time-evolving a matrix product state with long-ranged interactions*, Phys. Rev. B **91**(16), 165112 (2015), doi:10.1103/PhysRevB.91.165112.

[38] H. Landa and G. Misguich, *Nonlocal correlations in noisy multiqubit systems simulated using matrix product operators*, SciPost Phys. Core **6**, 037 (2023), doi:10.21468/SciPostPhysCore.6.2.037.

[39] *github.com/qiskit-community/lindbladmpo*.

[40] K. Bidzhiev and G. Misguich, *Out-of-equilibrium dynamics in a quantum impurity model: Numerics for particle transport and entanglement entropy*, Phys. Rev. B **96**(19), 195117 (2017), doi:10.1103/PhysRevB.96.195117.

[41] T. Ishiyama, F. Kazuya and T. Sasamoto, *Exact density profile in a tight-binding chain with dephasing noise*, doi:10.48550/arXiv.2501.07095 (2025).

[42] M. Žnidarič, *Exact solution for a diffusive nonequilibrium steady state of an open quantum chain*, J. Stat. Mech. **2010**(05), L05002 (2010), doi:10.1088/1742-5468/2010/05/L05002.

[43] V. Eisler, *Crossover between ballistic and diffusive transport: the quantum exclusion process*, J. Stat. Mech. **2011**(06), P06007 (2011), doi:10.1088/1742-5468/2011/06/P06007.

[44] B. Žunkovič, *Closed hierarchy of correlations in Markovian open quantum systems*, New J. Phys. **16**(1), 013042 (2014), doi:10.1088/1367-2630/16/1/013042.

[45] T. Barthel and Y. Zhang, *Solving quasi-free and quadratic Lindblad master equations for open fermionic and bosonic systems*, J. Stat. Mech. **2022**(11), 113101 (2022), doi:10.1088/1742-5468/ac8e5c.

[46] M. Žnidarič, T. Prosen and I. Pižorn, *Complexity of thermal states in quantum spin chains*, Phys. Rev. A **78**(2), 022103 (2008), doi:10.1103/PhysRevA.78.022103.

[47] T. Prosen, *Third quantization: a general method to solve master equations for quadratic open Fermi systems*, New J. Phys. **10**(4), 043026 (2008), doi:10.1088/1367-2630/10/4/043026.

[48] K. Yamanaka and T. Sasamoto, *Exact solution for the Lindbladian dynamics for the open XX spin chain with boundary dissipation*, SciPost Physics **14**(5), 112 (2023), doi:10.21468/SciPostPhys.14.5.112.

[49] P. L. Krapivsky, K. Mallick and D. Sels, *Free bosons with a localized source*, J. Stat. Mech. **2020**(6), 063101 (2020), doi:10.1088/1742-5468/ab8118.

[50] P. L. Krapivsky, K. Mallick and D. Sels, *Free fermions with a localized source*, J. Stat. Mech. **2019**(11), 113108 (2019), doi:10.1088/1742-5468/ab4e8e.

[51] J. Houdayer, H. Landa and G. Misguich, *A solvable model for graph state decoherence dynamics*, SciPost Physics Core **7**(1), 009 (2024), doi:10.21468/SciPostPhysCore.7.1.009.

[52] H. J. Briegel and R. Raussendorf, *Persistent entanglement in arrays of interacting particles*, Phys. Rev. Lett. **86**, 910 (2001), doi:10.1103/PhysRevLett.86.910.

[53] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest and H.-J. Briegel, *Entanglement in graph states and its applications*, In *Quantum Computers, Algorithms and Chaos*, pp. 115–218. IOS Press, doi:10.3254/978-1-61499-018-5-115 (2006).