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

Jérôme Houdayer, Grégoire Misguich

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

Abstract

We introduce TensorMixedStates, a Julia library build on top of ITensor which allows the simulation of quantum systems in presence of dissipation using matrix product states (MPS). It offers three main features: i) it implements the MPS representation for mixed states and associated operations, in particular the time evolution according to a Lindblad equation or using non-unitary gates, 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, TDVP, conserved quantum numbers and automated parallelization), finally iii) it presents a user-friendly interface allowing writing professional simulations for pure and mixed quantum states in a few lines of code.

Received Date
Accepted Date
Published Date

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, trapped ions, coupled light-matter systems or superconducting circuits to name a few. Quantum technologies and the development of devices that are able to perform some quantum information tasks 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. In some situations the presence of the environment can give rise to interesting new phenomena and new dynamical regime. The environment can even be exploited to engineer useful quantum many-body states [2]. 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 no simpler [3]. 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 [4, 5]. 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 [6]. 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 some quantum advantage [7–11]. In fact, an external environment tends to decrease the amount of entanglement among the degrees of freedom inside the system, and it often tends to decrease 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 libraries for manipulating pure states with MPS (like ITensor [12] or TenPy [13]), to our knowledge there is no general library for simulating *mixed* states with MPS. We attempt here to fill this gap and this paper presents the TensorMixedStates library [14] which allows manipulating mixed many-body quantum states in the form of MPS. It is based on the ITensor [12] 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, 15, 16].

2 Context

In this section, we recall well known facts about quantum states and their representations with MPS.

2.1 Pure and mixed quantum states

In quantum mechanics, the states of a closed system form a Hilbert space $\mathcal{H}$, so that a state $|\psi\rangle \in \mathcal{H}$. 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 matrix density ρ which must be hermitian, positive with unit trace [1]. When the state is pure, ρ is 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

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

For a mixed state a general discrete evolution is called a quantum map and takes the form [17]

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

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 Lindblad master equation [15, 16]

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

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.

2.2 Matrix product states and operators

In numerical simulations of quantum many-body problems, the dimension of $\mathcal{H}$ rapidly becomes a problem. We can either restrict ourselves to small systems or use approximate representations of the states. MPS [6] provide such an approximate representation. Suppose $\mathcal{H}$ is a finite tensor product of N finite-dimensional local Hilbert spaces $\mathcal{H}_i$ of dimension d_i. Thus, we have $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$. That is our system is composed of N sites i with d_i states each. In the pure case, any state $|\psi\rangle$ can be written (using simplified notations)

$$|\psi\rangle = T_1 T_2 \cdots T_N, \tag{11}$$

where T_i is a matrix whose elements are states of site i (i.e. they belong to $\mathcal{H}_i$ and have dimension d_i). Of course, the required T_i may in general have very large dimensions (as matrices). We will note these dimensions χ_{i-1} and χ_i (clearly $\chi_0 = \chi_N = 1$), they are called the *bond dimensions*. Said otherwise, the T_i are tensors with three indices of dimensions χ_{i-1}, χ_i and d_i. MPS are particularly useful when there exist matrices T_i of size much smaller than $\dim(\mathcal{H})$ which provide a good approximation of the target state $|\psi\rangle$. In practice, one sets a maximum bond dimension χ, and we approximate the states of interest by MPS with $\chi_i \leq \chi$. The larger χ, the larger is the precision of the approximation.

What about mixed states? Viewing the density matrix ρ as the element of a (larger) Hilbert space (so-called vectorization) allows to represent it also as an MPS. Such a vectorized form of a mixed state is denoted by $|\rho\rangle\rangle$, and we write

$$|\rho\rangle\rangle = R_1 R_2 \cdots R_N, \tag{12}$$

with the elements of R_i living in a larger local space at site i which dimension is d_i^2.[1] For a pure state, we have $R_i \sim Ti \otimes T_i^\dagger$.

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 in $\mathcal{H}_i$ (in the pure case), that is the O_i are tensors with four indices (two of which having dimension d_i) and we then have

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

2.3 The service provided by TensorMixedStates

The ITensor library provides tools to manipulate MPS and MPO. In particular, it contains powerful tools to create MPO from operators. For mixed states things however get more complicated. The main issue is the following: in the pure case, given one operator O and a state $|\psi\rangle$ there is essentially a single operation that needs to be performed, namely $|\psi\rangle \to O|\psi\rangle$. To perform this operation one possibility is to compute the MPO for O and the MPS for ψ and ITensor has been designed from the ground up to do this efficiently. But it was not designed with mixed states in mind. In the mixed case, from one operator and a state ρ one must be able to compute four new objects: $\mathrm{Tr}(O\rho)$, $O\rho O^\dagger$, $[O,\rho]$ and $\{O^\dagger O,\rho\}$.

In particular, it has been necessary to implement from scratch the functions to create the different required MPO and on this path we also had to change the way operators are managed. The TensorMixedStates library is automatically able to manage all the types of sites available in ITensor (including user-defined ones) and all the operators (including user-defined ones, multi-site operators and fermionic operators).

3 Features

3.1 Design choices

The TensorMixedStates library (TMS) is the successor of the Lindbladmpo library [19, 20]. The Lindbladmpo library was based on the C++ version of the ITensor library and is limited

[1] Note that, contrary to the matrix-product density operator representation [18], the representation of Eq. 12 does not guaranty that ρ is positive. In practice this is not an issue as long as the bond dimension is large enough to ensure a sufficient accuracy for the observables of interest.

to systems of qubits by design. As the ITensor library has migrated to the Julia language, the old C++ version lacks in features like the TDVP algorithm for time evolution [21].

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 the ITensor library. Moreover, using Julia helped us develop a more flexible and more user-friendly interface. Finally, the combined use of Julia and ITensor bring automated parallelization for free.

We thus decided for a double interface: (i) a high-level interface allows designing professional simulations in a few lines of code as demonstrated in section 3.8. (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. While the software is still in rapid development, we keep it to GitHub, so one must write at the julia prompt

```
]add https://github.com/jerhoud/TensorMixedStates.jl
```

later it will be available on the julia registry and one would add it in the usual way 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 [14].

3.3 Space

The first step in a quantum simulation consists in describing the Hilbert space. In our case, the space must be a finite product over N local finite-dimensional Hilbert spaces. Those local spaces can be chosen independently among the "site types" proposed by ITensor. At the moment there are six possible choices: spin-1/2 (or qubit), boson (or qudit), fermion, electron, spin one and tJ). Note that this set is extendable by the user (see ITensor documentation [22] 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", "Fermion"])
```

3.4 States and representations

In TMS, both pure and mixed states are represented by a `State` object. 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. $|\psi\rangle = |\psi_1\rangle \cdots |\psi_N\rangle$) for example by

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

# a mixed state for our hybrid system
st2 = State(Mixed, sys2, ["+", "1", "FullyMixed"])
```

The name of the local states (like "Up" or "1") are those defined by ITensor, this is extendable by the user. Note that "FullyMixed" corresponds to the maximally mixed state (thermal state at infinite temperature, $\rho = I_d/d$). A `State` object contains three fields: `state` for the MPS, `type` is `Pure` or `Mixed` and `system` is the `System` object.

To build more complicated states, one can form linear combinations of states, for example we could build the GHZ state on our 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 turn a pure representation into a mixed one we can write

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

3.5 Operators

In TMS, operators are objects of type `Operator`. For example, `Z` is the Pauli operator σ^z and `DZ` (also called `DPhase`) is the associated Lindblad dissipator (dephasing jump operator). From such operators, we can build, using Julia powerful syntax, any Hamiltonian, for example the XX spin chain

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

Quantum gates applying on more than one sites are also available: one can write for example

```
my_gate = H(1)H(2)CNOT(1,2)H(1)H(2).
```

One can also create its own operators, for example a dissipative gate (or quantum channel)

```
@create_mixed_gate(K, [1-(p+q+s), p, q, s], [I,X,Y,Z], p, q, s)
```

creates the operator `K` defined by

$$K|\rho\rangle\rangle = (1-(p+q+s))\rho + pX\rho X + qY\rho Y + sZ\rho Z, \tag{15}$$

which one can use to define gates

```
my_gate = K(1; p=0.1, q=0.2, s=0.3)
```

to apply to states as shown below.

3.6 Algorithms

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

```
new_state = apply(my_gate, old_state),
```

Second, we can perform some time evolution. That is integrate $\partial_t|\psi\rangle = -iH|\psi\rangle$ for pure state and the Lindblad equation for mixed states. There are two functions to do this, one using TDVP (called `tdvp`) [21] and one using the W^{I} or W^{II} (called `approx_W`) MPO approximation (see Zaletel *et al.* [23] and [24]). 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 many customizations, from setting the integration time step or the details of the algorithm like truncation, to defining observers for intermediate measurements.

Note that `tdvp` and `approx_W` can be used with time-dependent Hamiltonians and/or dissipators.

The last algorithm is the computation of the ground state using DMRG [6, 25]. At the moment, it is only useful for pure states and is essentially the algorithm from the ITensor library

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

In the near future, it will be possible to use DMRG for mixed states to minimize the square $\mathcal{L}^{\dagger}\mathcal{L}$ of the Lindbladian in order to compute directly the steady-state of the system without the need to perform a long time evolution.

3.7 Measurements

Finally, we need a way to make measurements on a `State`. For example, to obtain the expectation value of an observable, we can write

```
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 expectations 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))
```

We can also use some predefined function 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 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 in very few lines of code. All the examples of the next section were made in this mode.

The principle is the following: define a sequence of actions to be applied on a state and pass it to `runTMS`. For our first example, we have the following full script

```
using TensorMixedStates

hamiltonian(n) = -sum(Cdag(i)C(i+1)+Cdag(i+1)C(i) for i in 1:n-1)
dissipators(n, gamma) = sqrt(4gamma) * sum(DN(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, the next two define the evolution operator (more details on this 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 some time evolution. The fields names are self-explanatory. The `measures` field describes the data files created by the simulation (here 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: `SaveState` to save the state to disk in HDF5 format, `LoadState` to load from disk a state previously saved by SaveState, `ToMixed` to turn a pure representation into a mixed representation, `Dmrg` to use the DMRG algorithm and `Gates` to apply gates.

4 Examples

In this section, we illustrate the use of the TMS library to study four different dissipative quantum problems evolving according to a Lindblad equation. The examples are chosen because they have been studied in the recent literature and because some exact solution is available. These solutions allow checking quantitatively the numerical results. The first example is a fermionic chain with some 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.

Finally, Sec. 4.5 presents a study of a deep quantum circuit which involves unitary 2-qubit gates as well as some dissipative channels which model qubit errors.

4.1 Fermion tight-binding chain with dephasing noise

We consider here the one-dimensional spinless fermion model studied in [26]. 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

$$H = -\sum_{i=1}^{N-1} \left(c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right) \tag{16}$$

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

$$L_i = \sqrt{4\gamma} n_i, \tag{17}$$

where $n_i = c_i^\dagger c_i$ is the fermion number operator on site i. With the TMS library such dissipative model is very easy to define. Using the high-level interface the Hamiltonian and the jump operators are combined into an `evolver` (see Section 3.8 for the full listing)

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

Dissipative problems where the Hamiltonian and the Lindblad are quadratic in the creation and annihilation operators can be solved exactly [27–30]. The present model was recently studied in the limit of an infinite chain [26]. The authors of this study demonstrated that the system displays some oscillatory decay or some over-damped decay, depending on the strength γ of the dissipation. In Fig. 1 some numerical results for the fermion density, obtained with the present library, are compared to the exact asymptotic results derived in [26]. The fermion density converges to $1/2$ at long times and 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 [26].

The bottom panel of Fig. 1 represents the evolution of the OSEE [31, 32] associated to a bipartition of the chain in the center. 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.

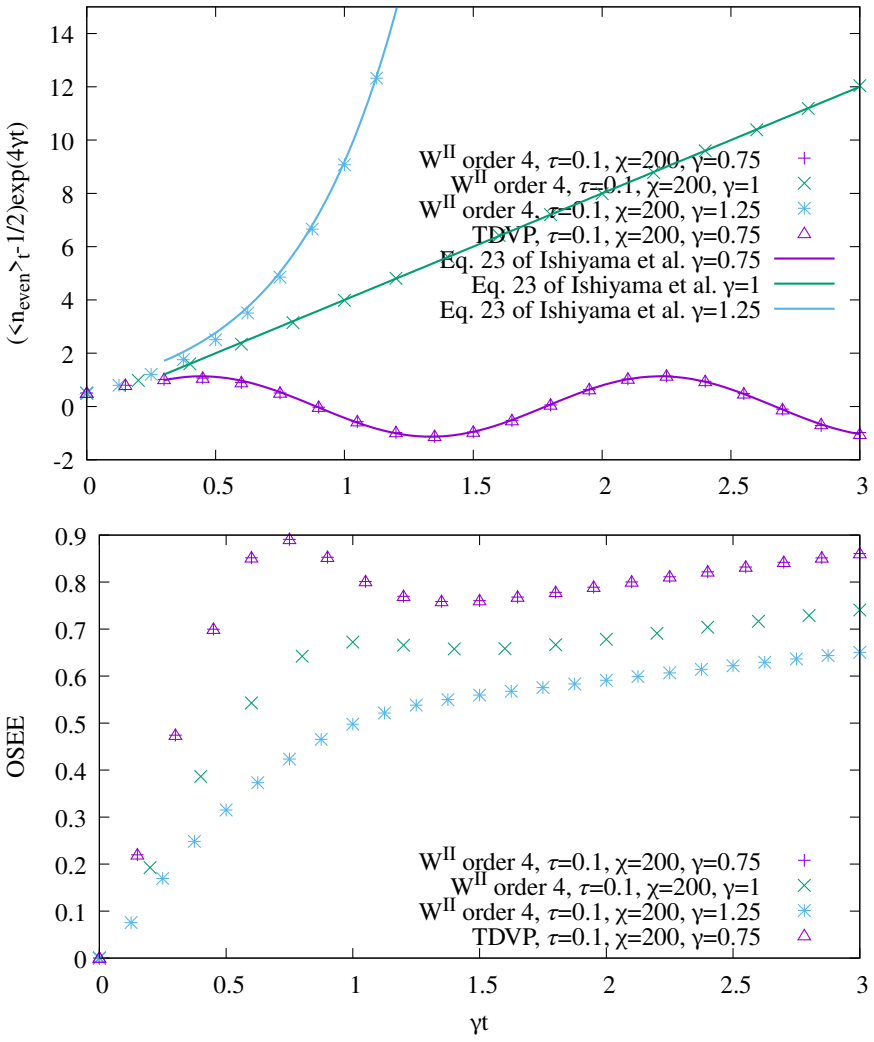

Figure 1: Top: fermion density $\langle n_i \rangle$(t) on even sites i as a function of time in a tight binding chain with dephasing noise and three different strength of the noise ($\gamma = 0.75, 1.0, 1.25$). The density is evaluated by averaging over 4 sites in the center of the system. Full lines: exact asymptotic results of Ref. [26]. Bottom: OSEE as a function of time, computed for the half of the chain. The simulations have been carried out with the W^{II} algorithm at order 4 and with TDVP (see legend).

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

$$H = \sum_{i=1}^{N-1} \left(\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y\right), \tag{18}$$

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

$$L_1 = \sqrt{\varepsilon_{\mathrm{L}} \frac{1+\mu_{\mathrm{L}}}{2}} \sigma_1^+, \quad L_3 = \sqrt{\varepsilon_{\mathrm{R}} \frac{1+\mu_{\mathrm{R}}}{2}} \sigma_N^+, \tag{19}$$

$$L_2 = \sqrt{\varepsilon_{\mathrm{L}} \frac{1-\mu_{\mathrm{L}}}{2}} \sigma_1^-, \quad L_4 = \sqrt{\varepsilon_{\mathrm{R}} \frac{1-\mu_{\mathrm{R}}}{2}} \sigma_N^-, . \tag{20}$$

$\sigma^{\pm} = (\sigma^x \pm i\sigma^y)/2$, $\varepsilon_{\mathrm{L,R}}$ 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) =
  sqrt(eL*(1+muL)/2)*DUp(1) + sqrt(eL*(1-muL)/2)*DDn(1) +
  sqrt(eR*(1+muR)/2)*DUp(n) + sqrt(eR*(1-muR)/2)*DDn(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 [30,33] and can be solved exactly. The dynamics of this model was solved exactly [34] (see also [27] for the exact steady-state). Fig. 2 displays the time evolution of two observables: the magnetization $\langle\sigma_1^x\rangle$ on the first spin and the spin current $\langle\sigma_1^x\sigma_2^y - \sigma_1^x\sigma_2^y\rangle$ on the first bond. The data are in agreement with the results of Ref. [34].

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:

$$H = \sum_{i=-N/2+1}^{N/2} (b_i^\dagger b_j + b_j^\dagger b_i) \tag{21}$$

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:[2]

$$L_0 = \sqrt{2\Gamma}\, b_0^\dagger. \tag{22}$$

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

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

[2] Compared with Eq. 2 of Ref. [35], the factor 2 in the equation below comes from their different normalization used in the Lindblad equation.

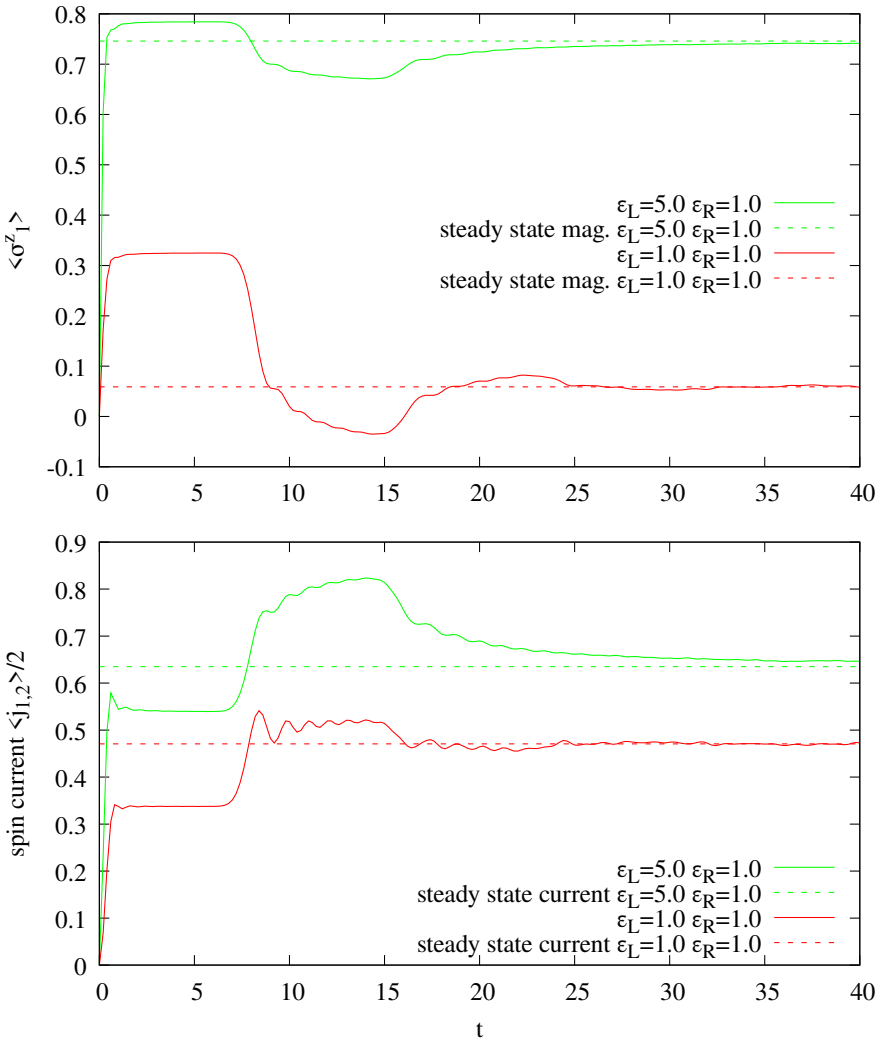

Figure 2: XX spin chain with boundary dissipation (Eqs. 18-20) Top: the magnetization $\langle\sigma_1^x\rangle$ on the first spin as a function of time. Bottom: spin current $\langle\sigma_1^x\sigma_2^y - \sigma_1^x\sigma_2^y\rangle$ on the first bond. Physical parameters: infinite-temperature initial 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 [34].

This model is quasi-free and has been studied analytically by Krapivsky *et al.* [35] 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 $N_{\text{tot}}(t) = \sum_i \langle b_i^\dagger b_i \rangle$ 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 some simulation one has to specify some maximum boson occupation of the sites. To set the maximum occupation to 5 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"; dim = 5),
  state =  "0"
)
```

In Fig. 3 the simulation results are compared with a numerically 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 [35]. Two quantities are displayed in Fig. 3: the density profile $\langle b_i^\dagger b_i \rangle$ (top panel) at time $t = 1$ and $t = 5$, and the evolution of $N(t)$, the mean number of boson (bottom panel). Up to time $t \simeq 5$ this simulation carried out using the W^{II} algorithm at order 4 with a time step $\tau = 0.1$ and a maximum bond dimension $\chi = 200$ appears to describe the dynamics quite accurately. Due to the large local Hilbert space dimension of such bosonic system is however not straightforward to obtain accurate results at longer times. For this reason, checking the asymptotic results derived analytically in [35] would require some relatively heavy simulations.

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

$$H = \sum_{i=-N/2+1}^{N/2} (c_i^\dagger c_j + c_j^\dagger c_i) \tag{23}$$

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

$$L_0 = \sqrt{2\Gamma}\, c_0^\dagger. \tag{24}$$

As for the model of Sec. 4.3, the model is quasi-free, and it has been studied analytically [35]. The figures 4 and 5 display several quantities: the density profile at three different times, the time evolution of the mean total number of fermions $N(t)$, the OSEE associated to a partition in the center of the system, the second Rényi entropy $S_2 = -\ln\left(\text{Tr}\left[\rho^2\right]\right)$, and the accumulated trace error. For the density profile $\langle n_i(t) \rangle$, $N(t)$ and for the entropy $S_2(t)$ the results are compared with the exact solution.[3]

Fig. 5 allows to compare the precision of the various algorithms (TDVP versus W^{II} at different orders) and for various values of the time step τ and maximum bond dimension χ. In this example where the particle injection rate is $\Gamma = 0.2$ the all simulations (except for W^{II} at order 1) 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

[3] This solution was obtained by solving numerically with Maple the set of N^2 differential equations describing the evolution of the two-point correlations $\text{Tr}\left[\rho c_i^\dagger c_j\right]$, see Eqs. 17 of Ref. [36]. Up to some signs these equations are very similar to those describing the dynamics of the 2-point correlations in the boson model of Sec. 4.3.

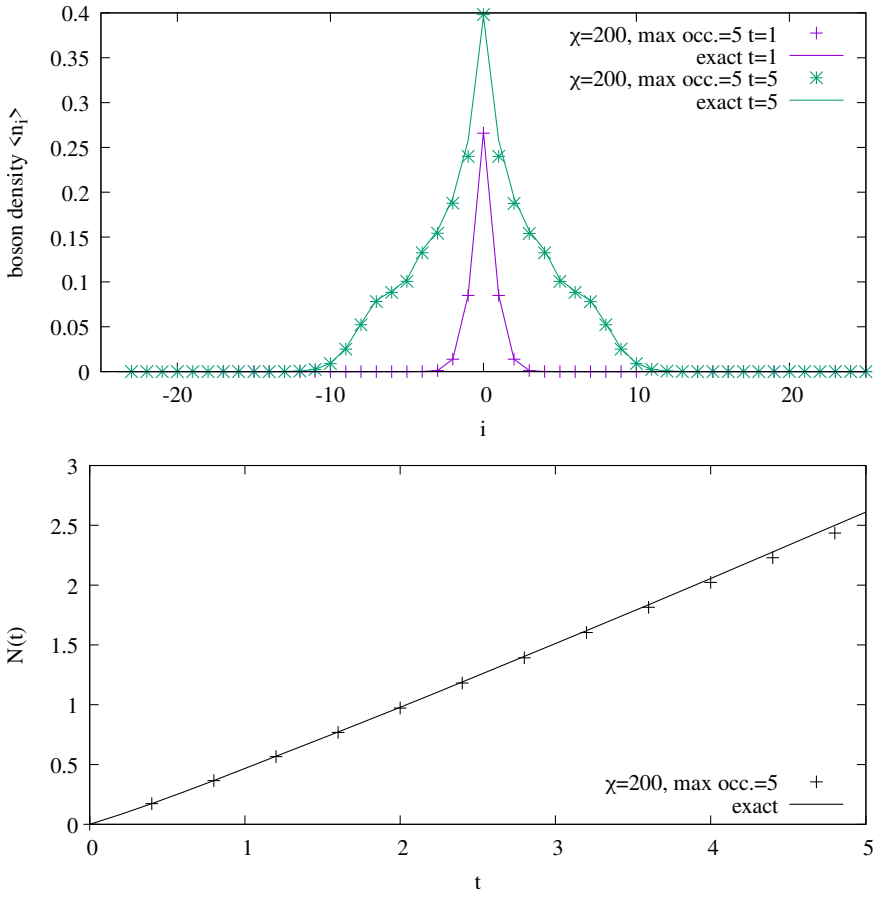

Figure 3: Free boson model with a localized source (Eqs. 21-22). Top: density profile $\langle b_i^\dagger b_i \rangle$ Bottom: total number of boson $N(t)$, numerics versus exact result. Physical parameters: system size $N = 50$, $\Gamma = 0.2$. Simulation parameters: maximum bond dimension $\chi = 200$, time step $\tau = 0.1$, algorithm: W^{II} at order 4.

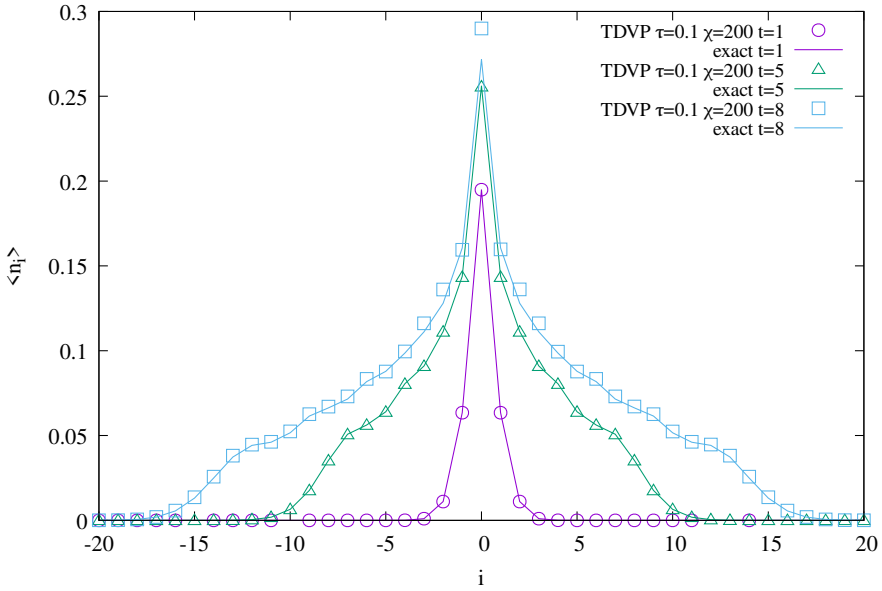

Figure 4: Free fermion model with a localized source (Eqs. 23-24): density profile $\langle n_i(t)\rangle$ at three different times ($t = 1$, 5 and 8). The full lines are exact results and the symbols have been obtained with TDVP with Trotter time step $\tau = 0.1$, maximum bond dimension $\chi = 200$. System size: $N = 50$. At $t = 1$ and $t = 5$ the simulation reproduces almost perfectly the exact profiles. However, 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.

TDVP (blue squares in Fig. 5). 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^{I} at order 4, W^{II} at order 4 and TDVP offer a similar precision. It should be noted however that the execution time is significantly longer in the case of TDVP.

4.5 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 brick wall structure and is similar to the circuits encountered when discretizing (Trotter) an Hamiltonian evolution.

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

$$L_{XX} = U^{\phi}_{XX}(1,2)U^{\phi}_{XX}(3,4)\cdots U^{\phi}_{XX}(N-1,N) \tag{25}$$

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

$$U^{\phi}_{XX}(i,j) = \exp(i\phi X_i X_j). \tag{26}$$

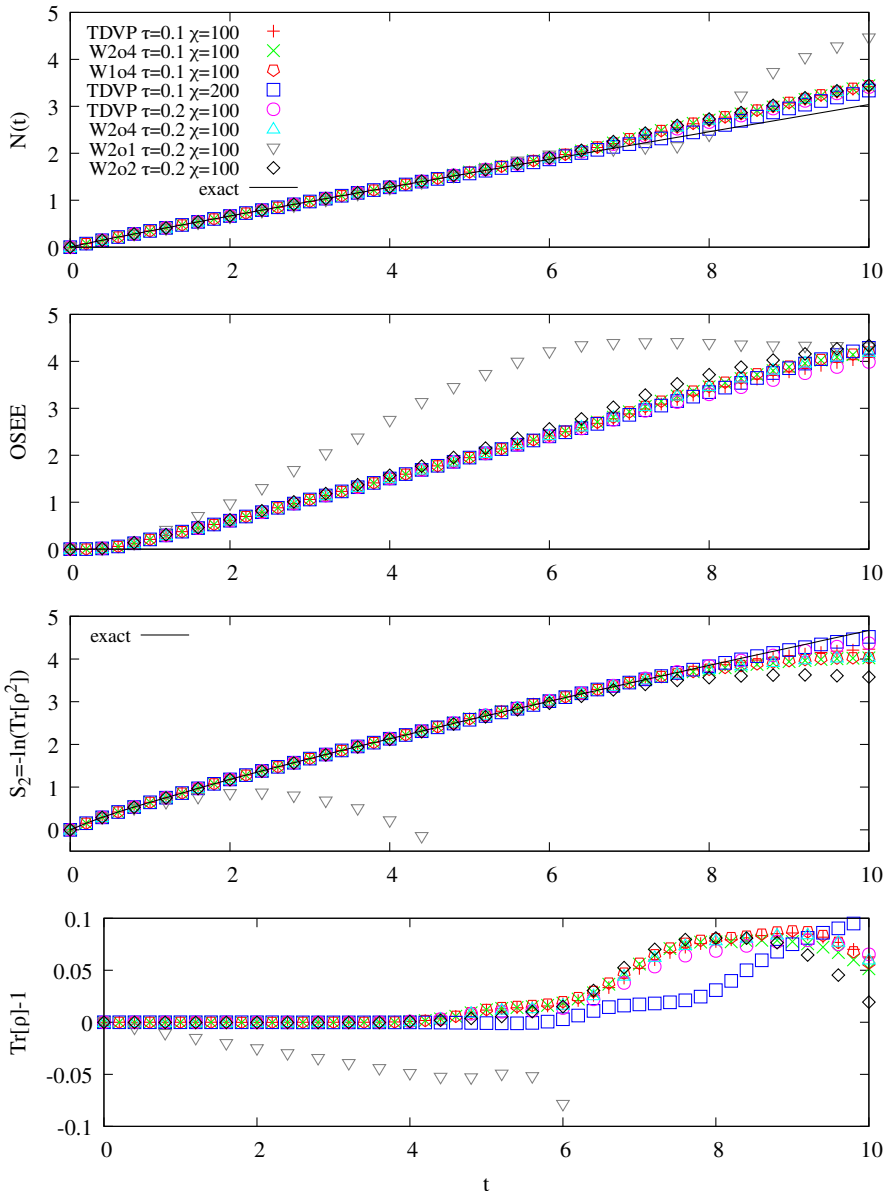

Figure 5: Free fermion model with a localized source (Eqs. 23-24). From top to bottom: (i) the time evolution of the mean total number of fermions $N(t)$, (ii) the operator-space entanglement entropy (OSEE) associated to a partition in the center of the system, (iii) the second Rényi entropy $S_2 = -\ln\left(\mathrm{Tr}\left[\rho^2\right]\right)$, and (iv) the accumulated trace error. In the legend w2o1 stands for W^{II} at order 1 and w2o4 stands for W^{II} at order 4. τ is the Trotter time step and χ the maximum bond dimension. System size: $N = 50$.

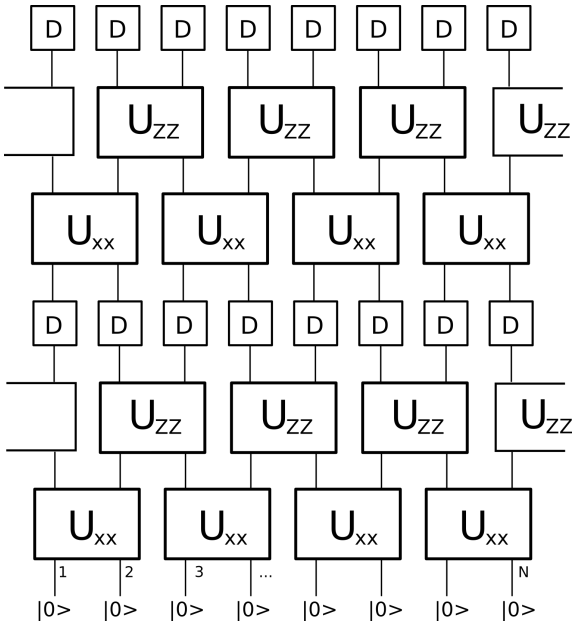

Figure 6: Brick wall quantum circuit made from layers of $U^{\phi}_{XX}(i,i+1)$ gates (Eq. 25), from layers of $U^{\phi}_{ZZ}(i,i+1)$ gates (Eq. 27) and from layers of depolarization gates (Eq. 29).

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

$$L_{ZZ} = U^{\phi}_{ZZ}(N,1)U^{\phi}_{XX}(2,3)\cdots U^{\phi}_{XX}(N-2,N-1) \tag{27}$$

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

$$U^{\phi}_{ZZ}(i,j) = \exp(i\phi Z_i Z_j). \tag{28}$$

The layers L_{XX} and L_{ZZ} do not commute with each other and create some entanglement. Next, we introduce gates which model qubit errors. This can be done via the following quantum channel (called depolarization channel):

$$D_{i,p} : \rho \to \left(1-\frac{3}{4}p\right)\rho + \frac{1}{4}pX_i\rho X_i + \frac{1}{4}pY_i\rho Y_i + \frac{1}{4}pZ_i\rho Z_i \tag{29}$$

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

$$L_{\epsilon} = \prod_i D_{i,p} \tag{30}$$

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 has all qubits in state $|0\rangle$. The circuit as a brick wall structure, as illustrated in Fig. 6.

With the TMS library the construction of the above gate sequence can be done with

```
[[Gates(
    name = "Applying exp(I*XX*phi) gates on qubits [1,2],[3,4],...",
    gates = prod(Rxx(2*i-1,2*i;phi) for i in 1:div(n, 2)),
    limits = limits,
    ),
  Gates(
```

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

The top panel of Fig. 7 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 S_2 entropy). Due to the large amount of correlation generated by the initial layers of the circuit a relatively large bond dimension $\chi \sim 2000$ is required to get some 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 usual Hamiltonian evolution 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 five examples to show the versatility and correctness of the software: four involving the Lindblad equation: two on fermions, one on bosons and one on spin 1/2. The last example demonstrates the use of non-unitary gates in a noisy quantum circuit calculation.

In the near future, we intend to work on further developments: (i) use the conserved quantum number capabilities of ITensor to improve the efficiency and precision of certain simulations, (ii) use DMRG on the square of the Lindbladian ($\mathcal{L}^\dagger\mathcal{L}$) to compute the steady-state of a dissipative system, (iii) use automated MPO compression to enhance performance in many cases.

Acknowledgements

We thank H. Landa for some previous collaborations on closely related topics.

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.

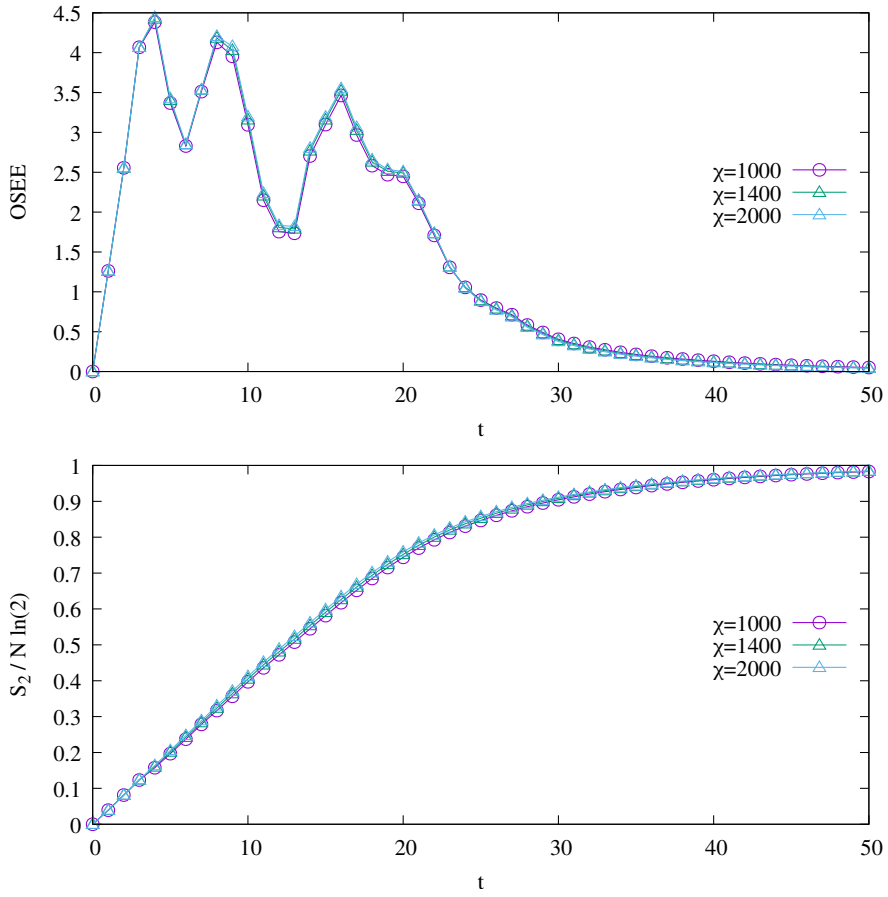

Figure 7: Circuit simulation. Top: OSEE as a function of the number t of layers ($L_\epsilon L_{ZZ} L_{XX}$ counts as one complete layer). Bottom panel: second Rényi entropy S_2 normalized by its maximum value $N \ln 2$. System size: $N = 20$. Error rate $p = 0.02$ and gate angle $\phi = 0.5$. Simulations with bond dimension $\chi = 1000$, 1400 and 2000.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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).

[12] 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.

[13] 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.

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

[22] *itensor.github.io/ITensors.jl/stable/examples/Physics.html*.

[23] 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.

[24] 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.

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

[26] 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).

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.