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 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, TDVP, quantum numbers conservation and automated parallelization), 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, 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, 7]. 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 [8–12]. 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 powerful libraries for manipulating pure states with MPS (like ITensor [13] or TenPy [14]), the software offering for the simulation of open/dissipative systems with MPS is much more limited.[1] We attempt here to fill this gap and this paper presents the TensorMixedStates library [18] (TMS) which allows manipulating *mixed* many-body quantum states in the form of MPS. It is based on the ITensor [13] 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, 19, 20]. It also permits constructing and manipulating density matrices using gates or user-defined quantum channels.

2 Context

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

[1] See for instance [15–17].

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 matrix density ρ which must be hermitian, positive 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

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

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 [21]

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

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 [19, 20]

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

where $\mathcal{L}$ is the Lindbladian (or Liouvillian)

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

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 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 [6] 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: $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$. 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 (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, they are called the *bond dimensions*.[2] 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, we set a maximum bond dimension χ, and we approximate the states of interest by MPS with $\chi_i \leq \chi$. The larger χ, the larger 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 $(d_i)^2$.[3] For a pure state, we have $R_i \sim T_i \otimes T_i^\dagger$.

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 Functionalities offered by TensorMixedState

The ITensor library provides tools for constructing and manipulating MPS and MPO. In particular, it contains powerful tools to create MPO from operators. For mixed states things however get more complicated since ITensor has not been designed to represent density matrices MPS as in Eq. 12. 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 developed with mixed states in mind. In the mixed case, from one operator O 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 TMS the functions to create the MPO associated to the objects above have been written from scratch, and on this path the way operators are managed has also been re-written. The TMS library is able to manage all the types of sites available in ITensor (qubits, fermions, bosons, etc.) and all the associated operators. Moreover, it is very easy to define new types of site or new operators, including multi-site operators, fermionic operators, Lindblad dissipators and noisy gates. Concerning time-evolution, TMS inherits from the powerful algorithms implemented in ITensor, such as the time-dependent variational principle (TDVP) [23].

[2] Clearly the first matrix needs to have $\chi_0 = 1$ and for the last one $\chi_N = 1$.

[3] Note that, contrary to the matrix-product density operator representation [22], 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 can be taken large enough to ensure a sufficient accuracy for the observables of interest.

3 Features

3.1 Design choices

TMS is the successor of the Lindbladmpo library [24, 25]. 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 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. 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 [18]. Moreover, a full documentation is also available online (see the README.md file [18]).

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 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 the dimension d of the local Hilbert space must be provided (truncation of the Fock space and maximal occupation number equal to $d-1$).

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 a 10 qubit system
st1 = State(Pure(), sys1, "Up")

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

The name of the local states (like "Up" or "1") are those predefined in TMS, 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 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 turn a pure representation into a mixed one we can write

```
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 many operators are predefined, for example, for qubit we have `X, Y, Z` the Pauli operators or `H` and `Swap` the Hadamard and Swap gates, for bosons we have `A` and `N` the destruction operator and number operator and so on. But where TMS really shines is for operator manipulation, here are a few examples.

Defining an Hamiltonian can be written in one line

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

or Lindblad dissipators with the `Dissipator` function, for example `Dissipator(Sp)` represents the jump operator toward up for qubit (`Sp` is the S^+ operator). By convention the Linbladian is written

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

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:

$$K\rho = 0.9\rho + 0.1X\rho X. \tag{15}$$

One can define more complex operators such as

$$R_{xy}(\phi) = \exp\left[-i\frac{\phi}{4}(X\otimes X + Y\otimes Y)\right], \tag{16}$$

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(cotrolled(X))
```

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

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

3.6 Algorithms

There are three main algorithms that one can use in TMS. First, one 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, one 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`) [23] and one using the W^{I} or W^{II} (called `approx_W`) MPO approximation (see Zaletel *et al.* [26]). 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 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 $\mathcal{O}(\tau^5)$ [27]).

The last algorithm is the computation of the ground state using DMRG [6,28]. It is essentially the algorithm from the ITensor library

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

Using the phase `SteadyState` it is also possible to use DMRG for mixed states in order 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.[4]

[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 $\mathcal{L}^\dagger\mathcal{L}$.

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 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))
```

One 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 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 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: `ToMixed` to turn a pure representation into a mixed representation, `Dmrg` to use the DMRG algorithm, `Gates` to apply gates and `PartialTrace` to trace out some of the sites. In the near future, we will add more phases like saving to file and loading from file.

4 Examples

In this section, we illustrate the use of the TMS library to study several different 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 some exact solution is available.[5] 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. 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 does not have a Lindblad dynamics, but it consists in a deep quantum circuit with unitary 2-qubit gates as well as dissipative channels which model some qubit errors.

4.1 Fermion tight-binding chain with dephasing noise

We consider here the one-dimensional spinless fermion model studied in [29]. The initial state is a pure state where the even sites are occupied, and the odd ones are empty. The system then

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

evolves under the action of a nearest-neighbor hopping Hamiltonian

$$H = -\sum_{i=1}^{N-1} \left(c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right) \tag{17}$$

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

$$L_i = \sqrt{4\gamma} n_i, \tag{18}$$

where $n_i = c_i^\dagger c_i$ is the fermion number operator on site i. 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(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 Lindblad are quadratic in the creation and annihilation operators can be solved exactly [30–33]. The present model was recently studied in the limit of an infinite chain [29]. 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 [29]. 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 [29].

The bottom panel of Fig. 1 represents the evolution of the OSEE [34, 35] 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.

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

$$H = \sum_{i=1}^{N-1} \left(\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y \right), \tag{19}$$

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

$$L_1 = \sqrt{\varepsilon_\mathrm{L} \frac{1+\mu_\mathrm{L}}{2}} \sigma_1^+, \quad L_3 = \sqrt{\varepsilon_\mathrm{R} \frac{1+\mu_\mathrm{R}}{2}} \sigma_N^+, \tag{20}$$

$$L_2 = \sqrt{\varepsilon_\mathrm{L} \frac{1-\mu_\mathrm{L}}{2}} \sigma_1^-, \quad L_4 = \sqrt{\varepsilon_\mathrm{R} \frac{1-\mu_\mathrm{R}}{2}} \sigma_N^-, . \tag{21}$$

$\sigma^\pm = (\sigma^x \pm i\sigma^y)/2$, $\varepsilon_{\mathrm{L,R}}$ 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) +
```

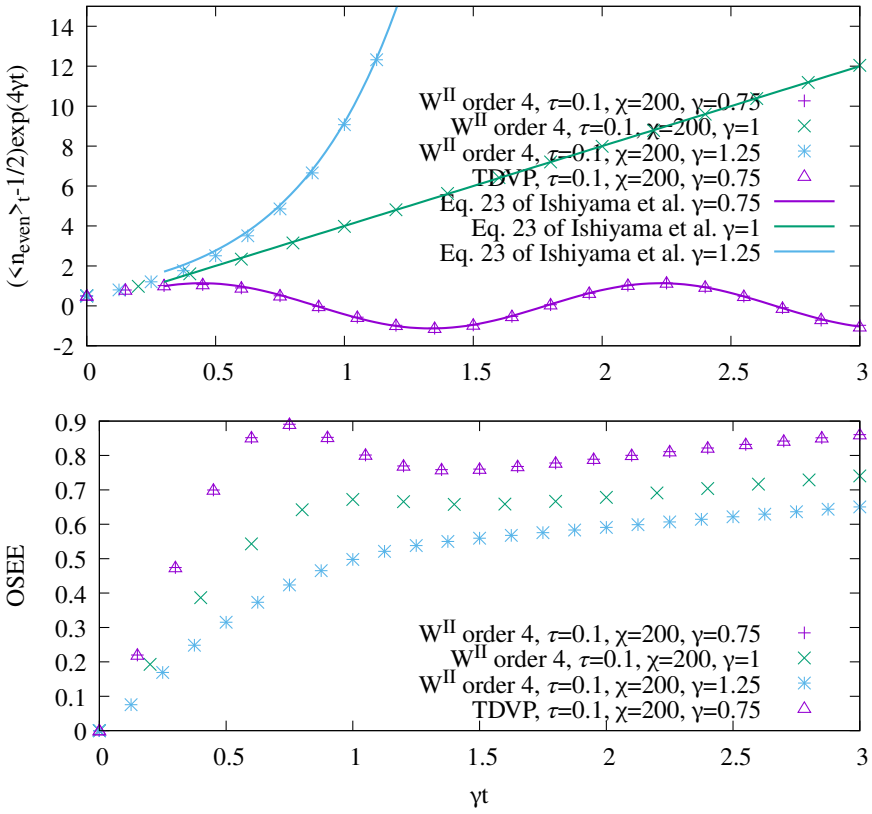

Figure 1: Top: fermion density $\langle n_i \rangle$(t) on even sites i as a function of time in a tight binding chain with dephasing noise and three different strength of the noise ($\gamma = 0.75, 1.0, 1.25$). The density is evaluated by averaging over 4 sites in the center of the system. Full lines: exact asymptotic results of Ref. [29]. 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).

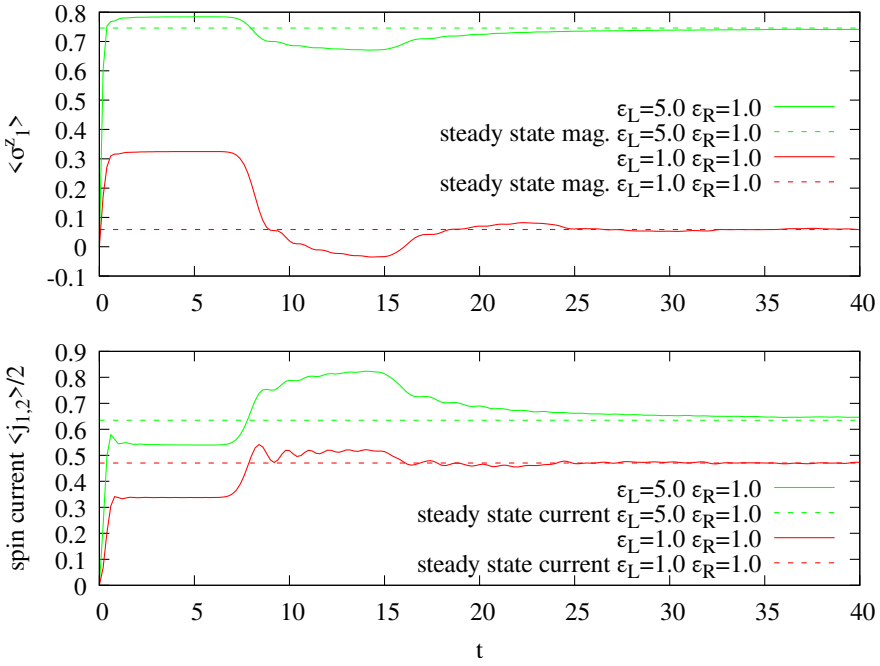

Figure 2: XX spin chain with boundary dissipation (Eqs. 19-21) Top: mean magnetization $\langle\sigma_1^x\rangle$ on the first spin as a function of time. Bottom: mean spin current $\langle\sigma_1^x\sigma_2^y - \sigma_1^x\sigma_2^y\rangle$ on the first bond. Physical parameters: infinite-temperature 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 [37].

```
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 [33,36] and can be solved exactly. The dynamics of this model was studied in Ref. [37] (see also [30] for the exact steady-state). Fig. 2 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 are in agreement with the results of Ref. [37].

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:

$$H = \sum_{i=-N/2+1}^{N/2-1} (b_i^\dagger b_{i+1} + b_{i+1}^\dagger b_i) \tag{22}$$

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]

$$L_0 = \sqrt{2\Gamma}\, b_0^\dagger. \tag{23}$$

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

```
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.* [38] 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 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. 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 [38]. 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 [38] 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

$$H = \sum_{i=-N/2+1}^{N/2-1} (c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i) \tag{24}$$

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

$$L_0 = \sqrt{2\Gamma}\, c_0^\dagger. \tag{25}$$

As for the model of Sec. 4.3, the model is quasi-free, and it has been studied analytically [38]. The figure 4 displays the density profile at three different time, the time evolution

[6] Compared with Eq. 2 of Ref. [38], the factor 2 in the equation below comes from a different normalization used in their Lindblad equation.

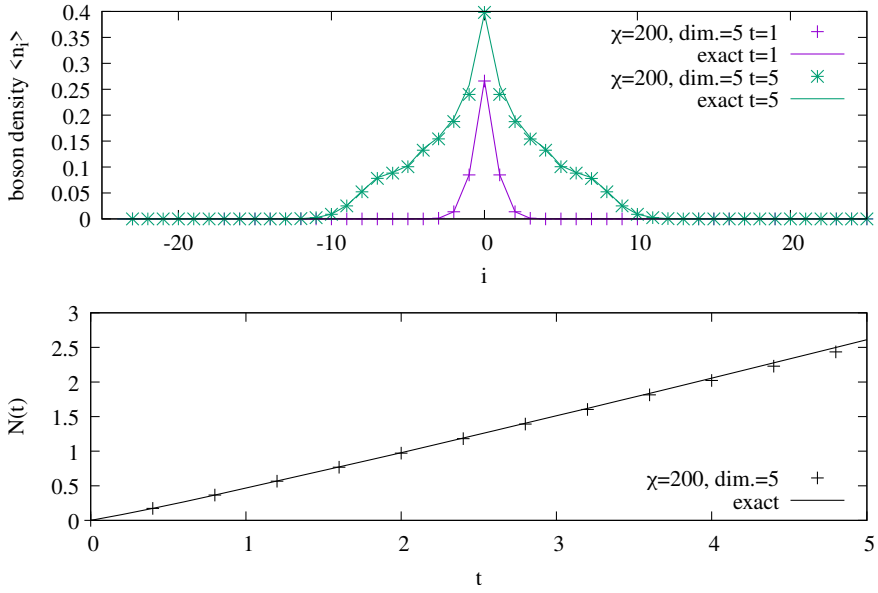

Figure 3: Free boson model with a localized source (Eqs. 22-23). Top: density profile $\langle b_i^\dagger b_i \rangle$ Bottom: total number of boson $N(t)$, numerics versus exact result. Physical parameters: system size $N = 50$, $\Gamma = 0.2$. Simulation parameters: maximum bond dimension $\chi = 200$, local dimension: dim=5 (boson occupancy ≤ 4), time step $\tau = 0.1$, algorithm: W^{II} at order 4.

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. 4 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 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. 4). 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 significantly longer in the case of TDVP.

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 with only dissipators (no Hamiltonian). For this model the time-dependence of numerous observables is known analytically, as discussed in Ref. [40].

A graph state [41, 42] is a pure entangled state that is constructed by the application of

[7] This solution was obtained by solving numerically with Maple 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. [39]. 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.

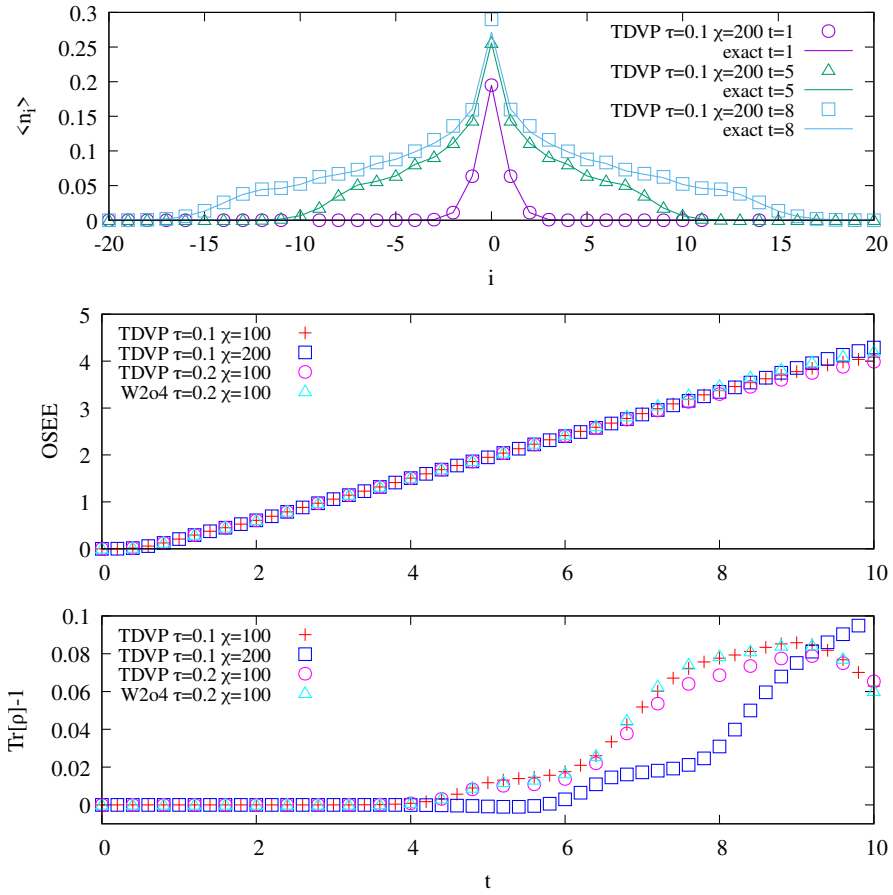

Figure 4: Free fermion model with a localized source (Eqs. 24-25). Top: density profile $\langle n_i(t)\rangle$ at three different times ($t = 1$, 5 and 8). The full lines are exact results and the symbols have been obtained with TDVP with Trotter time step $\tau = 0.1$, maximum bond dimension $\chi = 200$. 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 associated to a partition of the system in the center. 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.

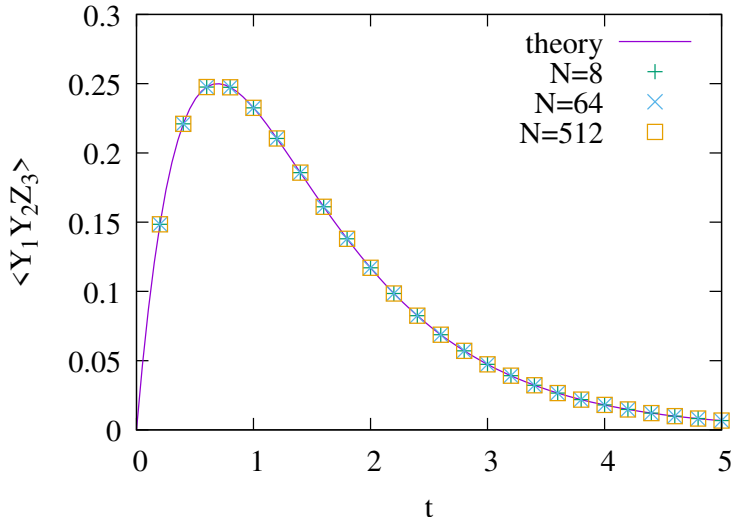

Figure 5: Time evolution of $\langle Y_1 Y_2 Z_3 \rangle$ in a model describing the decoherence of a complete-graph state for sizes 8, 64 and 512. The strength of the dissipation corresponds to $g_0 = 1.0$ in the notation of [40]. The solid line shows the exact result (Eq. 31 of [40]). Due to the permutation 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.

some controlled-Z (CZ) two-qubit gates on a product state:

$$|g_E\rangle = \prod_{(i,j)\in E} \mathrm{CZ}(i,j)|++\cdots+\rangle, \tag{26}$$

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 of such initial state (in a mixed form) can be implemented as follows:

```
create_graph_state(
  Mixed(),
  complete_graph(n),
)
```

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

$$L_i = \sigma_i^+ , \; i = 1 \cdots N \tag{27}$$

which are operators toward the state $|0\rangle$. 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 [40]. 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. Comparisons of numerical results produced by TMS to exact results from Ref. [40] are shown on Fig. 5 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 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:

$$L_{XX} = U_{XX}^{\phi}(1,2)U_{XX}^{\phi}(3,4)\cdots U_{XX}^{\phi}(N-1,N) \tag{28}$$

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

$$U_{XX}^{\phi}(i,j) = \exp(i\phi X_i X_j). \tag{29}$$

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

$$L_{ZZ} = U_{ZZ}^{\phi}(N,1)U_{XX}^{\phi}(2,3)\cdots U_{XX}^{\phi}(N-2,N-1) \tag{30}$$

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

$$U_{ZZ}^{\phi}(i,j) = \exp(i\phi Z_i Z_j). \tag{31}$$

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):

$$D_{i,p} : \rho \to \left(1-\frac{3}{4}p\right)\rho + \frac{1}{4}pX_i\rho X_i + \frac{1}{4}pY_i\rho Y_i + \frac{1}{4}pZ_i\rho Z_i \tag{32}$$

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:

$$L_\epsilon = \prod_i D_{i,p} \tag{33}$$

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 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)
```

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) *
```

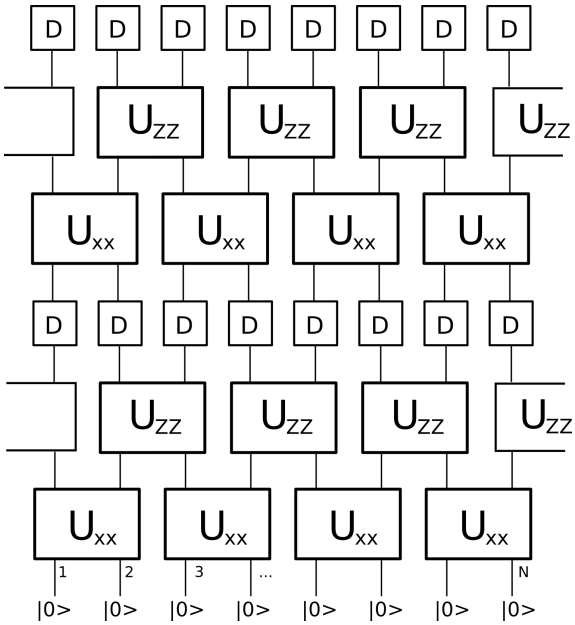

Figure 6: Brick wall quantum circuit made from layers of $U_{XX}^{\phi}(i,i+1)$ gates (Eq. 28), from layers of $U_{ZZ}^{\phi}(i,i+1)$ gates (Eq. 30) and from layers of depolarization gates (Eq. 32).

```
            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. 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 Rényi-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 six examples to show the versatility and correctness of the soft-

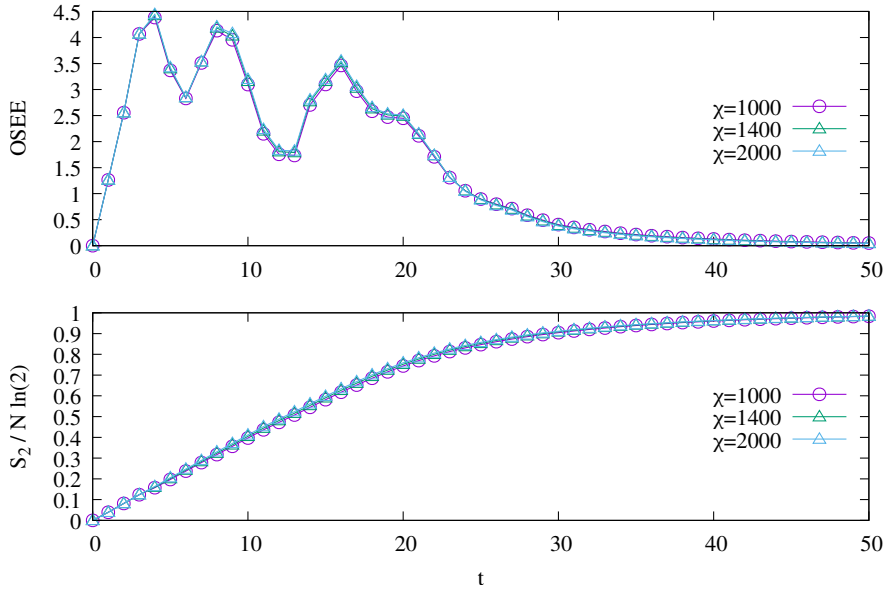

Figure 7: Circuit simulation. Top: OSEE as a function of the number t of layers ($L_\epsilon L_{ZZ} L_{XX}$ counts as one complete layer). Bottom panel: second Rényi entropy S_2 normalized by its maximum value $N \ln 2$. System size: $N = 20$. Error rate $p = 0.02$ and gate angle $\phi = 0.5$. Simulations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

ware: five involving the Lindblad equation: two on fermions, one on bosons and two 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 several further developments. These include in particular (i) the use of conserved quantum number capabilities of ITensor to improve the efficiency and precision of certain simulations, and (ii) use automated MPO compression to enhance 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] 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] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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).

[13] 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.

[14] 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.

[15] 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.

[16] G. Torlai and M. Fishman, *PastaQ: A package for simulation, tomography and analysis of quantum computers* (2020).

[17] 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.

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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).

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[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] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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).