Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Analysis of a density matrix renormalization group approach for transport in open quantum systems [☆]

Heitor P. Casagrande [a,b], Dario Poletti [a,c], Gabriel T. Landi [b,*]

[a] Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, 8 Somapah Road, 487372, Singapore
[b] Instituto de Física, Universidade de São Paulo, 05314-970, São Paulo, São Paulo, Brazil
[c] Engineering Product Development Pillar, Singapore University of Technology and Design, 8 Somapah Road, 487372, Singapore

ARTICLE INFO

Article history:
Available online 21 June 2021

Keywords:
Tensor networks
Open quantum systems
Density matrix renormalization group

ABSTRACT

Understanding the intricate properties of one-dimensional quantum systems coupled to multiple reservoirs poses a challenge to both analytical approaches and simulation techniques. Fortunately, density matrix renormalization group-based tools, which have been widely used in the study of closed systems, have also been recently extended to the treatment of open systems. We present an implementation of such method based on state-of-the-art matrix product state (MPS) and tensor network methods, that produces accurate results for a variety of combinations of parameters. Unlike most approaches, which use the time-evolution to reach the steady-state, we focus on an algorithm that is time-independent and focuses on recasting the problem in exactly the same language as the standard Density Matrix Renormalization Group (DMRG) algorithm, initially put forward in [1]. Hence, it can be readily exported to any of the available DMRG platforms. We show that this implementation is suited for studying thermal transport in one-dimensional systems. As a case study, we focus on the XXZ quantum spin chain and benchmark our results by comparing the spin current and magnetization profiles with analytical results. We then explore beyond what can be computed analytically. Our code is freely available on github at [2].

1. Introduction

Transport properties at the nanoscale may be significantly different from bulk materials, because of low-dimensionality, interactions and interference. For instance in quantum spin chains one can observe anomalous diffusion [3–5], negative differential conductance and rectification [6–8].

The study of many-body quantum systems, however, is in general very demanding. For instance, considering a pure state describing a chain of N spin-$1/2$ particles, one needs to take into account a Hilbert space of size 2^N. However, when aiming to study the transport properties of a system coupled to two different baths at its edges, as shown in Fig. 1, one needs also to find a way to model the effects of the bath, thus requiring to explore a space larger than 2^N. One approach to study open quantum, i.e. quantum systems in contact with an environment, is that of the Gorini-Kossakowski-Sudarshan-Linbdlad (GKSL) master equation, which is a linear equation describing the evolution of the density matrix of the system [9,10]. For this reason, to study transport in this framework, one needs to be able to explore a space of dimension 2^{2N}. An exact study of systems of this type becomes quickly too difficulty, e.g. for systems with $L \approx 12$ (see [11] for an exact diagonalization study with $L = 14$, which was only possible by considering symmetries of a particular class of boundary driven problems).

Fortunately, for the particular case of one-dimensional (1D) systems, a particular class of numerical methods has been developed starting from the seminal work [12], which outlined the density matrix renormalization group (DMRG) method that was later realized within the general framework of tensor networks [13]. Tensor networks are a particular form of variational ansatz to explore, at a polynomial cost, an otherwise exponentially large Hilbert space. The key advantage of this variational approach is that a large class of physically relevant ground states, e.g. Hamiltonians of 1D systems with finite range interactions, can be exactly described using tensor networks [13]. The key is that the bipartite entanglement entropy of the system should not grow linearly with the system size (a.k.a. volume law).

Tensor network methods are known to be useful also in the description of open quantum systems. In this case there is no analytical proof that the method will accurately describe the open quantum system. However, as we will see later in this manuscript,

[☆] The review of this paper was arranged by Prof. D.P. Landau.
[*] Corresponding author.
E-mail addresses: heitor_peres@mymail.sutd.edu.sg (H.P. Casagrande), dario_poletti@sutd.edu.sg (D. Poletti), gtlandi@if.usp.br (G.T. Landi).

https://doi.org/10.1016/j.cpc.2021.108060
0010-4655/© 2021 Elsevier B.V. All rights reserved.

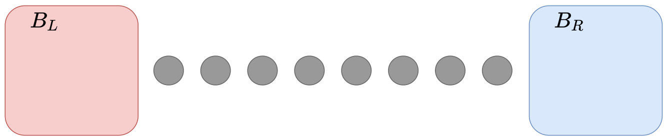


Fig. 1. Diagrammatic representation of a one-dimensional quantum spin chain coupled to two thermal baths B_L and B_R, one at each end. Here, each grey circle represent a spin.

the method can be very accurate for many physically relevant scenarios. Various approaches have been put forward to study many-body open quantum systems with tensor networks. A review which focuses on ensemble trajectories of stochastic wavefunctions [14], instead of evolving the density matrix, can be found in [15]. And a comparison of the trajectory method versus evolving directly the density matrix can be found here [16,17]. A short review of numerical methods to study many-body open quantum systems can be found in [18]. A particularly interesting approach was put forward in [1], in which the evaluation of the steady-state of a system was mapped into that of computing the ground state of an effective Hamiltonian with long range interactions. In this work we evaluate the performance of this approach when applied to boundary driven systems as the one shown in Fig. 1.

To make the paper self-contained, we first provide in Sec. 2 a detailed description of the model studied, with all the relevant equations and the different transport properties that can emerge. We follow this with a short description of a tensor network algorithm to compute the ground state for closed systems in Sec. 3. Then, in Sec. 4 we discuss in detail how one can map an open quantum system problem to a form conducive for tensor networks calculations. Next we describe how one can map the problem of finding the steady state of a many-body open quantum system to that of computing the ground state of an effective Hamiltonian in Sec. 5, where we also provide all details of an implementation using the ITensor library [19]. A series of numerical analyses benchmarking our code, comparing numerical and analytical results, and exploring the physics beyond what can be addressed analytically, is then discussed in Sec. 6. Conclusions are given in Sec. 7.

2. A boundary driven spin chain

One of the most widely studied examples of a open quantum spin chain is the XXZ model coupled to two local GKSL baths. The Hamiltonian for a 1D chain of N sites is given by

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

where σ_i^α are the Pauli matrices and J_i, Δ_i, h_i are parameters indicating respectively the tunneling between nearest sites, the anisotropy and a local magnetic field. In addition to the Hamiltonian dynamics, the system is also coupled to two baths at sites 1 and N, as described by a GKSL master equation [20]. The evolution of the system's density matrix ρ will then be given by

$$\frac{d\rho}{dt} = \mathcal{L}(\rho) := -i[H,\rho] + D_1(\rho) + D_N(\rho), \tag{2}$$

where

$$D_i(\rho) = \gamma_i f_i \mathcal{D}[\sigma_i^-](\rho) + \gamma_i(1-f_i)\mathcal{D}[\sigma_i^+](\rho), \qquad i = 1, N, \tag{3}$$

with $\mathcal{D}[L](\rho) = L\rho L^\dagger - \frac{1}{2}\{L^\dagger L, \rho\}$. Here $\gamma_i > 0$ represent the coupling strength to bath i and $f_i \in [0,1]$ represent the imbalance

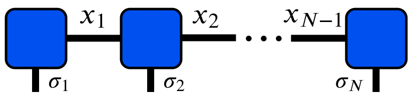


Fig. 2. Tensor network representation [Eq. (7)] of a high-rank tensor as a contraction of lower rank tensors.

between the baths. After a sufficient time has elapsed, the evolution of Eq. (2) will eventually reach a non-equilibrium steady-state (NESS) defined by

$$\mathcal{L}(\rho_{\text{ness}}) = 0. \tag{4}$$

In the vast majority of cases, this steady-state is also unique. Note also that, albeit a steady-state, the system will not be in equilibrium since there will be, in general, a steady current flow from one bath to the other.

The model described by Eqs. (1)-(3) presents remarkably rich physics. The most relevant observables to analyze are the local currents from site i to $i+1$

$$\mathcal{J}_i = 2J_i(\sigma_x^i\sigma_y^{i+1} - \sigma_y^i\sigma_x^{i+1}), \tag{5}$$

and the local magnetization σ_z^i. In the NESS, current conservation implies that $\langle\mathcal{J}_i\rangle$ will be independent of i (the current from $i-1 \to i$ is the same as that from $i \to i+1$). The physics is then characterized by the different transport properties of $\langle\mathcal{J}_i\rangle$. For large sizes, one usually has the scaling

$$\langle\mathcal{J}_i\rangle \sim \frac{1}{L^\alpha}, \tag{6}$$

where $\alpha > 0$ is an exponent characterizing the type of transport: ballistic for $\alpha = 0$, diffusive for $\alpha = 1$, superdiffusive for $\alpha \in [0,1]$, subdiffusive for $\alpha > 1$, and insulating when $\alpha \to \infty$. We mention in passing that, through a Jordan-Wigner transformation, the model in (1) can also be mapped into a chain of interaction fermions. In this case, the current (5) is interpreted instead as the particle current through the lattice.

3. Review of tensor network methods in closed quantum systems

The basic idea behind tensor networks is to decompose a high-rank tensor into a controlled product of lower rank tensors. Consider a generic rank-N tensor $\psi_{\sigma_1\ldots\sigma_N}$. A tensor network decomposition has the form

$$\psi_{\sigma_1\ldots\sigma_N} = \sum_{x_1,\ldots,x_{N-1}} A^{\sigma_1}_{x_1} A^{\sigma_2}_{x_1,x_2} \ldots A^{\sigma_N}_{x_{N-1}}, \tag{7}$$

which is shown diagrammatically in Fig. 2. This kind of expansion is relevant because quantum states of multipartite systems are naturally represented as a high-rank tensor. For instance, the state of a spin chain with N sites has the form

$$|\Psi\rangle = \sum_{\sigma_1,\ldots,\sigma_N} \psi_{\sigma_1,\ldots,\sigma_N} |\sigma_1,\ldots,\sigma_N\rangle, \tag{8}$$

where $\sigma_i = \pm 1$ are the eigenvalues of σ_i^z. Of course, while a decomposition of the form (7) is always possible, it is not necessarily advantageous. The advantages ultimately come from approximations that can be obtained by restricting the dimension of the internal indices x_i, called the bond dimension.

The tensor network decomposition (7) is used as the starting point for a variety of algorithms. The most notable is the Density Matrix Renormalization Group (DMRG) [12,21], a variational method to estimate the ground-state $|\psi_{\text{gs}}\rangle$ of one-dimensional

Hamiltonians, although the DMRG algorithm was not originally formulated in terms of tensor networks [12]. The idea is to solve the eigenvalue/eigenvector problem

$$H|\psi\rangle = E|\psi\rangle, \tag{9}$$

assuming that $|\psi\rangle$ is not an arbitrary quantum state, but rather a tensor network of the form (7) with a fixed maximum bond-dimension. Let us denote $|\psi_{\text{gs}}\rangle$ the lowest energy tensor network obtained from Eq. (9). This is to be contrasted with the true ground-state $|\Psi_{\text{gs}}\rangle$, which would be obtained if the full Hilbert space was used. According to the variational principle of quantum mechanics, the true ground-state energy $\mathcal{E}_{\text{gs}} = \langle\Psi_{\text{gs}}|H|\Psi_{\text{gs}}\rangle$ is always bounded by

$$E_{\text{gs}} := \langle\psi_{\text{gs}}|H|\psi_{\text{gs}}\rangle \geqslant \mathcal{E}_{\text{gs}}. \tag{10}$$

Hence, the energy associated to $|\psi_{\text{gs}}\rangle$ provides an upper bound on the true ground-state energy.

The search algorithm for the ground state is iterative. It proceeds by optimizing each tensor $A^{\sigma_i}_{x_{i-1},x_i}$ in Eq. (7) at a time, after which it moves to the next site (details can be found in Ref. [13]). Moving one site at a time through the chain, and then backwards, is usually referred to as a sweep. For a fixed maximum bond dimension, multiple sweeps can be employed to ensure convergence. After the algorithm has converged, the bond-dimension can be increased and the process can be restarted until the desired accuracy is met.

4. Hilbert space structures for open system dynamics

4.1. Vectorization

We now turn to the case of open quantum systems. For the purpose of concreteness, we shall focus on the problem described by Eqs. (1)-(3). The generalization to other types of Hamiltonians/dissipators is straightforward. The master equation (2) is still a linear equation in ρ. The difference is that the Liouvillian $\mathcal{L}(\rho)$ is now a superoperator, as it may act on ρ by means of matrix multiplications on both sides. This linearity can be made manifest by introducing a vectorization operation, also called Choi-Jamiolkowski's isomorphism [22,23], and described by

$$\text{vec}\big(|i\rangle\langle j|\big) = |j\rangle \otimes |i\rangle. \tag{11}$$

It thus converts an operator in Hilbert space, into a ket in a space whose size is the square of the initial one. Matrix-wise, this corresponds to stacking the columns of a matrix,

$$\text{vec}\begin{pmatrix} a & b \\ c & d\end{pmatrix} = \begin{pmatrix} a \\ c \\ b \\ d\end{pmatrix}. \tag{12}$$

Using vectorization, a general density matrix $\rho = \sum_{ij}\rho_{ij}|i\rangle\langle j|$ is converted into a ket

$$\text{vec}(\rho) = \sum_{ij}\rho_{ij}|j\rangle \otimes |i\rangle. \tag{13}$$

The fact that the size of the Hilbert space is squared reflects the fact that superoperators can act on both sides of a density matrix. Indeed, for any 3 matrices A, ρ and B, one may verify that

$$\text{vec}(A\rho B) = (B^{\text{T}} \otimes A)\text{vec}(\rho). \tag{14}$$

With this the master equation (2) can be converted into a linear matrix-vector equation

$$\frac{d}{dt}\text{vec}(\rho) = \hat{\mathcal{L}}\text{vec}(\rho), \tag{15}$$

where $\hat{\mathcal{L}}$ is now a matrix with entries

$$\hat{\mathcal{L}} = -i(I\otimes H - H^{\text{T}} \otimes I) + \hat{D}_1 + \hat{D}_N, \tag{16}$$

with $\hat{D}_i$ given by

$$\hat{D}_i = \gamma_i f_i \hat{\mathcal{D}}[\sigma_i^-] + \gamma_i(1-f_i)\hat{\mathcal{D}}[\sigma_i^+], \qquad i = 1, N, \tag{17}$$

and with

$$\hat{\mathcal{D}}[L] = L^* \otimes L - \frac{1}{2}\big[I \otimes (L^\dagger L) + (L^\dagger L)^{\text{T}} \otimes I\big]. \tag{18}$$

In all of the above expressions, I refers to the identity matrix with the appropriate dimension.

The above vectorization procedure is the starting point for most numerical algorithms dealing with quantum master equations of the form (2). The relaxation dynamics of (15) will simply be given by

$$\text{vec}(\rho_t) = e^{\hat{\mathcal{L}}t}\,\text{vec}(\rho_0), \tag{19}$$

Alternatively, one may look directly at the steady-state, Eq. (4), which now acquires the form

$$\hat{\mathcal{L}}\text{vec}(\rho_{\text{ness}}) = 0. \tag{20}$$

This equation makes explicit the fact that the NESS $\text{vec}(\rho_{\text{ness}})$ is simply the eigenvector of $\hat{\mathcal{L}}$ with eigenvalue 0. Stability implies that all eigenvalues of $\hat{\mathcal{L}}$ should have non-positive real parts. Moreover, when the steady-state is unique, there will be only a single eigenstate with eigenvalue 0. One should bear in mind, notwithstanding, that the normalization of the NESS is not the standard normalization for eigenvectors. Instead, density operators should be normalized as $\text{tr}(\rho) = 1$. This, in turn, can be viewed as the Hilbert-Schmidt inner product $\text{tr}(A^\dagger B)$ between two operators (in this case ρ and the identity matrix). Indeed, vectorization turns out to precisely convert Hilbert-Schmidt inner products into standard dot-products for the resulting vectors:

$$\text{tr}(A^\dagger B) = \text{vec}(A)^* \cdot \text{vec}(B). \tag{21}$$

Whence, the normalization condition becomes

$$\text{tr}(\rho) = \text{vec}(I)\cdot\text{vec}(\rho) = 1. \tag{22}$$

4.2. Vectorization for multipartite Hilbert spaces

The above recipe is not yet well suited for tensor network methods. The reason is that the vectorization procedure (11) generally changes the tensor ordering of the Hilbert space, which can have a significant impact on the numerics. To see this, we consider the spin chain problem in Eqs. (1)-(3). The density matrix for this system is described by the Matrix Product Operator (MPO)

$$\rho = \sum_{\substack{\sigma_1,\dots,\sigma_N\\ \sigma_1',\dots,\sigma_N'}} \rho_{\sigma_1,\dots,\sigma_N,\sigma_1',\dots,\sigma_N'}|\sigma_1\dots\sigma_N\rangle\langle\sigma_1'\dots\sigma_N'|, \tag{23}$$

where $|\sigma_1\dots\sigma_N\rangle = |\sigma_1\rangle\otimes\dots\otimes|\sigma_N\rangle$ (see Fig. 3). In what follows, the tensor product symbol will be omitted for clarity. That is, we will equivalently write this as $|\sigma_1\dots\sigma_N\rangle = |\sigma_1\rangle\dots|\sigma_N\rangle$. Naive vectorization, in terms of stacking columns [Eq. (12)] leads to

$$\text{vec}\big(|\sigma_1\dots\sigma_N\rangle\langle\sigma_1'\dots\sigma_N'|\big) = |\sigma_1'\rangle|\sigma_2'\rangle\dots|\sigma_N'\rangle|\sigma_1\rangle|\sigma_2\rangle\dots|\sigma_N\rangle. \tag{24}$$

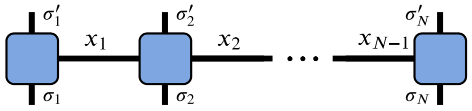


Fig. 3. Upon contracting over the horizontal, indices, one recovers ρ.

Whence, we see that it rearranges the Hilbert space so as to put all right-side indices σ_i' first, followed by all left-side indices. We shall refer to Eq. (24) as the $\boldsymbol{R^N L^N}$ **ordering**. The problem with this kind of structure, as we shall see below, is that it pushes indices pertaining to the same site, σ_i and σ_i', far away from each other.

Eq. (24) shows, in fact, that there is an arbitrariness in how to order the Hilbert space after a vectorization. The order in which the indices are placed is immaterial, provided that the operators acting on vec(ρ) are appropriately labeled to act on the correct site. For instance, a much more natural vectorization would be

$$\text{vec}\big(|\sigma_1 \dots \sigma_N\rangle\langle\sigma_1' \dots \sigma_N'|\big) = |\sigma_1'\rangle|\sigma_1\rangle \dots |\sigma_N'\rangle|\sigma_N\rangle, \tag{25}$$

which we shall refer to as the $\boldsymbol{(RL)^N}$ **ordering**. This ordering preserves the "real space" order of the original Hilbert space, bundling together left and right indices σ_i' and σ_i for each site. Other types of orderings may also be useful, depending on the problem in question. Ultimately, this will depend on the kinds of operators multiplying vec(ρ). As we shall see next, unitary and dissipative elements behave quite differently in this sense.

Let us begin by considering the unitary contribution. A typical Hamiltonian term for nearest-neighbor interactions has the form $H_1 = A_i A_{i+1}$, where A_i is an operator acting on site i. This Hamiltonian will act on the master equation as $[H_1, \rho]$. Thus, $H_1\rho$ will act on indices σ_i and σ_{i+1}, whereas ρH_1 will act on $\sigma_i'\sigma_{i+1}'$. The way this translates into the $\boldsymbol{R^N L^N}$ and $\boldsymbol{(RL)^N}$ orderings is illustrated in Figs. 4(a) and 5(a) respectively. As can be seen, in the $\boldsymbol{R^N L^N}$ ordering the Hamiltonian retains its nearest neighbor character, with two disconnected contributions acting on different parts of the Hilbert space. The $\boldsymbol{(RL)^N}$ ordering, on the other hand, leads to a second nearest neighbor interaction.

We now move on to the dissipative contributions. The special part is the first term in Eq. (18). A dissipator such as $\mathcal{D}[\sigma_1^-]$ for instance, has a contribution of the form $\sigma_1^-\rho\sigma_1^+$. This will act on indices σ_1 and σ_1'. The corresponding tensor structure for the two orderings will then be as shown in Figs. 4(b) and 5(b). As we now see, the $\boldsymbol{R^N L^N}$ ordering leads to a *highly* non-local Hilbert space structure, whereas in $\boldsymbol{(RL)^N}$ the structure is nearest-neighbor.

This analysis clearly shows why the $\boldsymbol{(RL)^N}$ ordering (Fig. 5) will in general fare better in a numerical calculation: even though the Hamiltonian is now a second nearest-neighbor interaction, the dissipator is only nearest-neighbor. This will be more advantageous than $\boldsymbol{R^N L^N}$ which has long-range interacting terms.

5. Implementation of the oDMRG algorithm

5.1. The $\mathcal{L}^\dagger\mathcal{L}$ method

Having established the Hilbert space structure, we are now in a position to implement the oDMRG algorithm introduced in [1]. The starting point is the steady-state equation (20), which shows that the NESS is the eigenstate of $\hat{\mathcal{L}}$ with eigenvalue 0. The problem with this equation is that $\hat{\mathcal{L}}$ is a non-Hermitian operator. To circumvent this, we consider instead the eigenvalue/eigenvector of $\hat{\mathcal{M}} = \hat{\mathcal{L}}^\dagger\hat{\mathcal{L}}$ [1]; that is, instead of (20) we solve

$$\hat{\mathcal{M}}\text{vec}(\rho_{\text{ness}}) = \hat{\mathcal{L}}^\dagger\hat{\mathcal{L}}\,\text{vec}(\rho_{\text{ness}}) = 0. \tag{26}$$

The operator $\hat{\mathcal{M}}$ has the same steady-state as $\hat{\mathcal{L}}$, but is Hermitian. Moreover, $\hat{\mathcal{M}}$ is by construction positive semi-definite, with exactly one zero eigenvalue (when the steady-state is unique) and all other eigenvalues strictly larger than zero.

For these reasons, Eq. (26) has now the exact same structure as the closed system eigenvalue problem Eq. (9): we need essentially to look for the ground-state of the effective Hamiltonian given by $\hat{\mathcal{M}}$, and the search for this groundstate is therefore entirely amenable to the closed DMRG algorithm. Eq. (26) also offers the additional advantage that the ground-state energy is known exactly, $E_{\text{gs}} = 0$. Hence, monitoring how the energy changes during the DMRG sweeps can be used as a way to probe the convergence of the algorithm (for an implementation of a DMRG-like code for the non-Hermitian superoperator $\hat{\mathcal{L}}$ see [24]).

For concreteness, we shall henceforth focus on the model in Eqs. (1)-(3) and choose the initial parameters such that $J_i = 1$, $\Delta_i = \Delta$ and $h_i = h$. The input parameters are then only the chain size N, together with γ, f_1, f_N, h and Δ.

One setback of the algorithm (26) is that even if the $\boldsymbol{(RL)^N}$ ordering is used for $\hat{\mathcal{L}}$, the tensor structure of $\hat{\mathcal{M}} = \hat{\mathcal{L}}^\dagger\hat{\mathcal{L}}$ will now be highly non-local. This should be expected because a steady state of local Hamiltonian and dissipative terms, in general does not need to follow an area-law, unlike the ground state of local Hamiltonians. However the non-locality of the terms may lead to an "entanglement-barrier" problem for the convergence of the algorithm. The reason is that, during convergence, the algorithm will pass through multiple, highly entangled, eigenstates of the operator $\hat{\mathcal{M}}$. As a side comment, note that these excited states are typically different from the eigenstates of $\hat{\mathcal{L}}$ because non-Hermitian operators have different left and right eigenvectors (the NESS is an eigenstate that both operators share). Another issue is that $\hat{\mathcal{M}}$ has usually a smaller gap, between the steady state and the first excited state, as compared to $\hat{\mathcal{L}}$, making the problem numerically harder to converge. These problems were investigated recently in Ref. [25], where the authors proposed additional approximations for making $\hat{\mathcal{M}}$ more local. In [25], however, the authors did not deal with boundary-driven transport problems. In our implementation we will consider the full $\hat{\mathcal{M}}$ operator, and as we show, for the boundary-driven problems we studied, good convergence rates were obtained without the need for these additional methods.

5.2. Positivity of the variational density matrix

Another side effect of using a closed-system algorithm for open quantum systems concerns the positivity of the numerically obtained density matrix vec(ρ). As discussed in Sec. 4.1, the vectorized density matrix is not normalized as a standard vector, but rather as in Eq. (22). This is at odds with the closed DMRG algorithm, which uses standard normalization. This, of course, can be readily fixed by appropriately renormalizing the output state. A more serious issue, however, concerns the positivity of the resulting density matrix: physical density matrices must be positive semi-definite. A physical tensor network variational state for vec(ρ) must therefore be one for which the corresponding "unvec'd" state is positive semi-definite. The set of tensor network states through which the system passes during the algorithm, however, is not restricted to this, but may very well contain also non-positive states. The set is also not convex, so that even if we were to start with a physical state, there is no guarantee that the algorithm remains in one. As a consequence, it is possible that the algorithm (26) converges to states which have low energies but are otherwise unphysical. This can be witnessed, for instance, by imaginary contributions to the expected values of observables.

We have found that this problem can be dramatically minimized by adopting the following procedure. First, we use as the starting guess for the tensor network state, a maximally mixed state, vec(I), an object which below we refer to as `Ivec`. Second, we start the process with very small bond dimensions, usually 2.

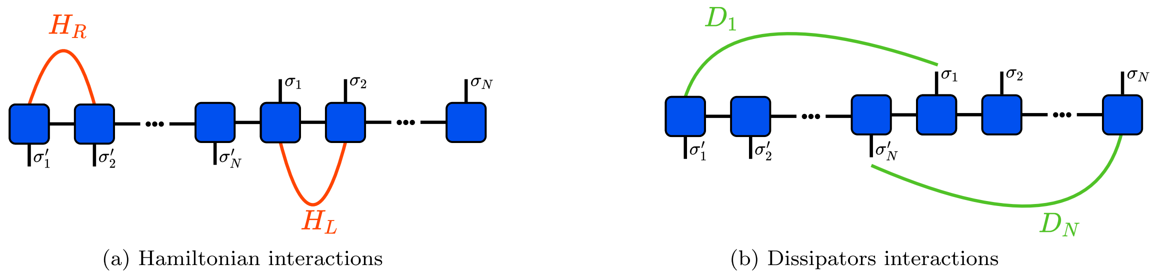


Fig. 4. Diagramatical depiction of the Hamiltonian and dissipative terms under the $\boldsymbol{R}^N\boldsymbol{L}^N$ formalism.

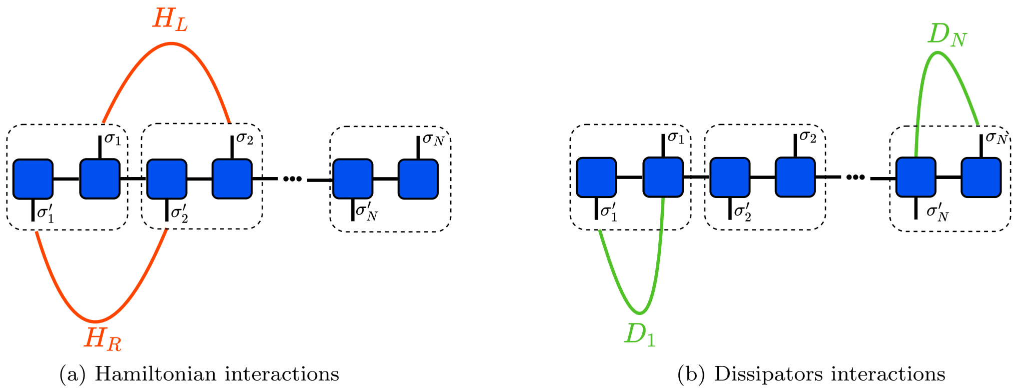


Fig. 5. Diagramatical depiction of the Hamiltonian and dissipative terms under the $(\boldsymbol{RL})^N$ formalism.

Such a small bond dimension allows for extremely fast computations, so that we allow for a large number of sweeps to ensure convergence. For such a small bond-dimension, the system is found to naturally converge to a physical state. We call this first phase the *warm-up*. Finally, and most importantly, we then proceed to increase the bond dimension in very small steps, usually in steps of 1 or 2 (allowing, of course, multiple sweeps for each bond dimension to ensure convergence). The reason why this works is because if the bond-dimension is too large, the algorithm will generally converge towards unphysical states. But by incrementing the bond dimension in small steps, one minimizes these disturbances, pushing the system towards the manifold of positive semi-definite states. We have no formal proof that this approach necessarily has to work. But in all scenarios we have tested, it was found to dramatically improve the results.

5.3. Numerical comparison between the $(\boldsymbol{RL})^N$ and $\boldsymbol{R}^N\boldsymbol{L}^N$ orderings

We implement the above steps using the iTensor library [19]. Initially, for the sake of comparison, we have implemented both the $(\boldsymbol{RL})^N$ and $\boldsymbol{R}^N\boldsymbol{L}^N$ orderings, and we provide in Fig. 6 a convergence test for both. This is done by monitoring the lowest eigenvalue of $\mathcal{M} = \mathcal{L}^\dagger\mathcal{L}$ which, as already discussed, is called "energy", in reference to Eq. (10). Recall that in our case the energy of the true steady-state is known to be identically zero. Thus, its magnitude serves as a quantifier of the convergence of the algorithm. Indeed, as the plot indicates, the $(\boldsymbol{RL})^N$ ordering is consistently more reliable, having a smoother convergence curve after each sweep, and requiring less sweeps to achieve better numerical results. In light of this, the discussion below will be centered on the $(\boldsymbol{RL})^N$ ordering.

5.4. Initialization

We now discuss the details of the implementation, focusing on the code available at [2]. Further details on the functions presented in this section can be found in A. The tensor class is called by

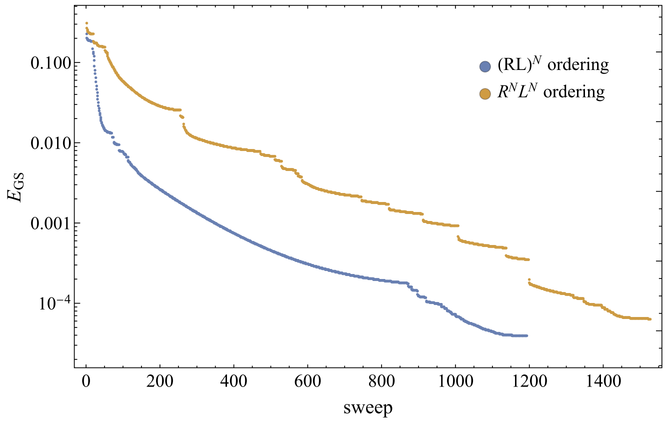


Fig. 6. Energy as a function of the number of sweeps, for the $(\boldsymbol{RL})^N$ and $\boldsymbol{R}^N\boldsymbol{L}^N$ orderings, with a fixed maximum bond dimension of 100. Values used: $N = 20, \gamma = 1, \Delta = 0.5, f_1 = 0.8, f_N = 0.2, h = 0$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

```
    auto sites = LRN(N);
    MPS rho = MPS(sites);
```

The resulting object `sites`, which is the output of the function `LRN`, contains all definitions of the Hilbert space structure, together with how the Pauli operators act on different indices for left- and right-multiplication.

All observables can then be constructed from the `sites` object. Here we focus on the currents (5) and the local magnetization σ_z^i. iTensor allows for a simple symbolic input for building operators, which we have adapted to include left and right multiplication. The resulting code is then implemented as

```
    for(int i=1; i<N; i++){
        auto aobs  = AutoMPO(sites);
        aobs +=  4.0,"SxL",i,"SyL",i+1;
        aobs += -4.0,"SyL",i,"SxL",i+1;
        obsCurrVec.push_back(MPO(aobs));
    }
    for(int i=1; i<=N; i++){
        auto aobs  = AutoMPO(sites);
        aobs += 2.0,"SzL",i;
        obsMagVec.push_back(MPO(aobs));
    }
```

An object such as ``SxL'', for instance, stands for the Pauli matrix acting on the left. Similarly, "SxL",i,"SyL",i+1 stand for the operator $\sigma_x^i \sigma_y^{i+1}$ acting on the left. The factors of 4 and 2 are simply because iTensor naturally loads spin operators $S_x^i = \sigma_x^i/2$, etc. The above routine constructs a list of MPOs, each representing the current in a given bond or the magnetization in a given site.

Next we construct the tensor network for the density matrix vec(ρ), which is the object that will be optimized in the algorithm. We also initialize it to the maximally mixed state vec(I) (normalization is not required and is done only when we compute the expectation values of observables). The code reads

```
    MPS rho;
    MakeIVEC(rho, N);
```

The function `MakeIVEC` constructs an MPO of the form vec(I). Lastly, we construct the matrix $\hat{\mathcal{M}} = \hat{\mathcal{L}}^\dagger \hat{\mathcal{L}}$:

```
   MPO LdL = LdLXXZConstruct(sites, Delta, f1, fL, gamma, h);
```

This function uses the same type of constructs used in iTensor to build Hamiltonians, but again taking care of proper left and right multiplications. It also uses a symbolic structure to construct the object in a way that is independent of the bond dimension being used. As a consequence, the resulting object `LdL` has no significant memory cost, irrespective of the size N. We also mention that while the above function focuses on a homogeneous chain (i.e. homogeneous Δ and h), it is trivial to extend it to the inhomogeneous case.

5.5. *Warm-up*

As discussed above, we perform a warm-up routine to improve the convergence to a physical tensor network. It performs multiple DMRG-sweeps with the lowest bond-dimension, to sharpen the initial parts of the simulation. The function receives the state `rho`, the MPO object `LdL`, the error threshold to stop the function, and an additional tag (which can be either "true" of "false") to manage the output of the function to the console.

```
   WarmUp(rho, LdL, 0.001, {"Quiet", true});
```

This function is a minor adaptation of iTensors built-in DMRG routine. The improvements brought about by the warm-up are significant, as shown in Fig. 7. The black horizontal line represents the sweep where the warm-up ends and the actual simulation begins (to be discussed in what follows). As can be seen, the reduction in energy during the warm-up is significant, even though the simulation time [Fig. 7(b)] is negligibly small.

5.6. *DMRG sweeps and final calculations*

After the initial warm-up routine, the DMRG procedure is then applied for increasing values of the bond-dimension parameter. The individual DMRG runs are called as

```
energyFin = dmrg(rho,LdL,sweeps});
```

which is just a call to the built-in DMRG function from iTensor. This can then be placed inside a loop, which compares the energy with the previous value; if the two fall within 10% of one another, the bond-dimension value is increased by a fixed amount. Both of these parameters, the threshold upon which one increases the bond dimension, and the amount of the bond dimension increase, can be easily altered by the user at the initial lines of the main routine. The algorithm can be run indefinitely, or the user may choose a stopping point, for instance, the maximum bond dimension, amount of sweeps, etc.

During each sweep, we calculate the spin current Eq. (5) throughout the chain, as well as the magnetization for each site. Both of these are done in a similar manner, contracting the previously loaded MPO of the relevant site with the tensor network object ρ. We take advantage of iTensor's optimized tensor network procedures. The value is then printed out. For example, the magnetization is computed with a loop going up to the size of the chain, for each site calculating

$$\frac{\langle I_{vec} | M_j | \rho \rangle}{\langle I_{vec} | \rho \rangle}, \tag{27}$$

where M_j is the magnetization MPO described in Sec. 5.4. This is done through the following excerpt:

```
for(int j=1;j<=N;j++){
    auto ev = overlapC(Ivec,obsMagVec[j-1],rho)/overlapC(Ivec,rho
     );
}
```

A similar procedure is done for the spin current.

```
for(int j=1; j<N; j++){
    auto current = overlapC(Ivec,obsCurrVec[j-1],rho)/overlapC(
     Ivec,rho);
}
```

In both cases we use the previously calculated vector of MPOs from Sec. 5.4. Of course, to optimize the code, one may also only compute the observables at the end of the process. Here we compute them at each step in order to monitor their convergence.

Finally, at the end of each sweep, we check if the energy has stabilized, and, if so, we increase the bond dimension value in order to advance the accuracy of the routine.

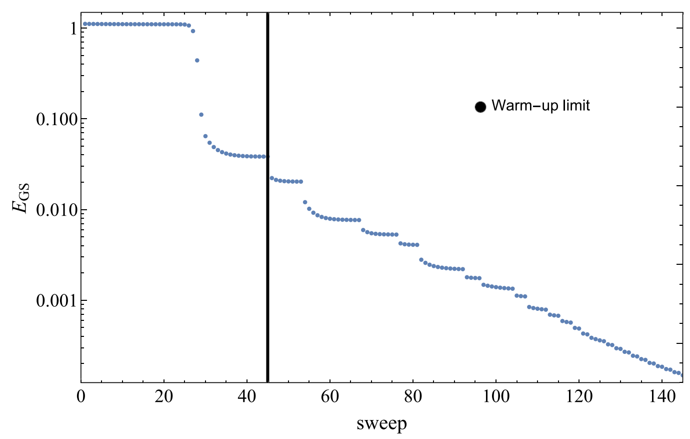
```
if ((energyIni-energyFin)/energyIni < sweepBDChangeThresholdValue
     ){
  bd += BDinc;
  sweeps.maxm() = bd;
}
```

Additional conditions can be easily implemented taking into account the simulation parameters, to finely tune the convergence of a specific set of parameters within a specific system.

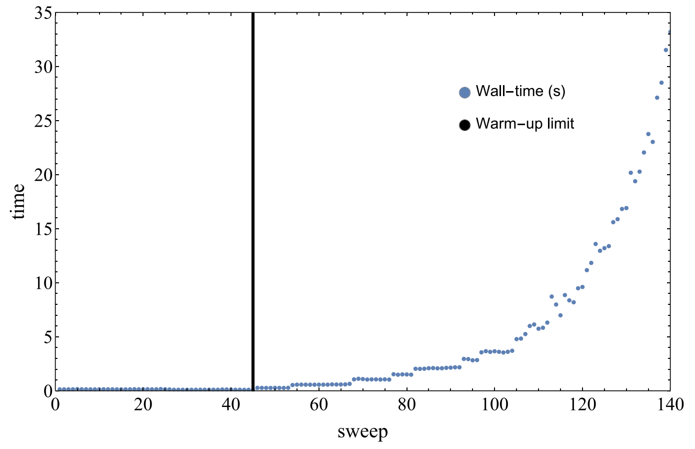
6. Results

6.1. *Benchmarking convergence*

Initially, we look at the convergence of the algorithm for different chain sizes. Illustrative results are shown in Fig. 8 for sizes up to $N = 50$. As can be seen, for small sizes the convergence is extremely fast. Increasing the size of the chain makes it so that more sweeps are necessary, but since the bond dimension of each sweep is increased in a slow, controlled manner, the convergence is possible even for larger sizes. For the particular case of $\Delta = 1$, $f_1 = 1$ and $f_N = 0$, the problem actually has an analytical solution in the form of a matrix product ansatz [3,26]. By looking at the average



(a) Ground-state energy guess after each DMRG sweep



(b) Wall-time for each sweep

Fig. 7. Plots of both ground state energy and time taken by sweep as a function of the number of sweeps. The black horizontal line represents the sweep where the warm-up ends and the actual simulation begins. Values used: $N = 10$, $\gamma = 1$, $f_1 = 1$, $f_L = 0$, $h = 0$, $\Delta = 1$.

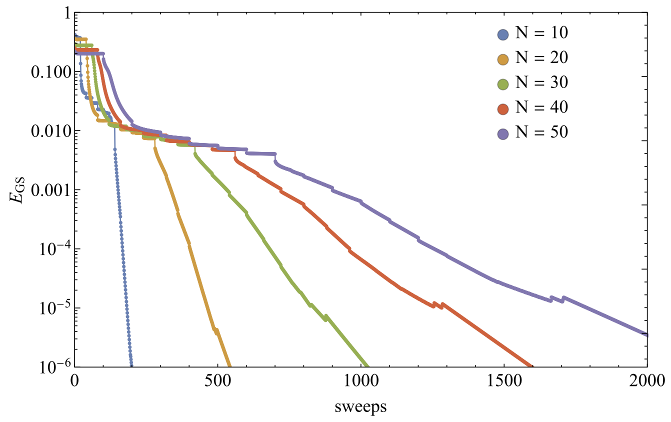


Fig. 8. Energy convergence for different chain lengths, after each sweep, for different system sizes N. Values used: $\gamma = 1$, $\Delta = 0.5$, $f_1 = 1$, $f_N = 0$, $h = 0$.

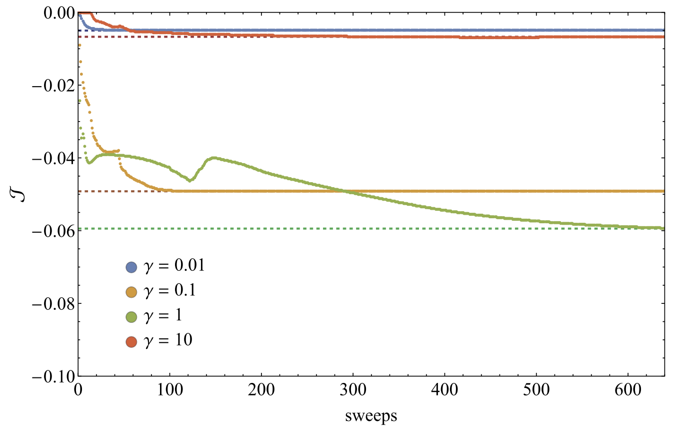


Fig. 9. Spin current convergence versus sweep number for different coupling values γ. Values used: $N = 25$, $\Delta = 1$, $f_1 = 1$, $f_N = 0$, $h = 0$. Dashed lines correspond to the analytical solutions in [3].

current after each sweep, we can therefore benchmark the algorithm to assure the convergence of the current. This is illustrated in Fig. 9. As can be seen, the convergence is generally slower for intermediate values of γ. Taking the analytical values available as references, we can see that the algorithm is working as intended. Additionally, all these simulations were made in the span of a couple days, with an average desktop: no broad computational power was required.

6.2. *Benchmarking the steady-state in comparison with analytical solutions*

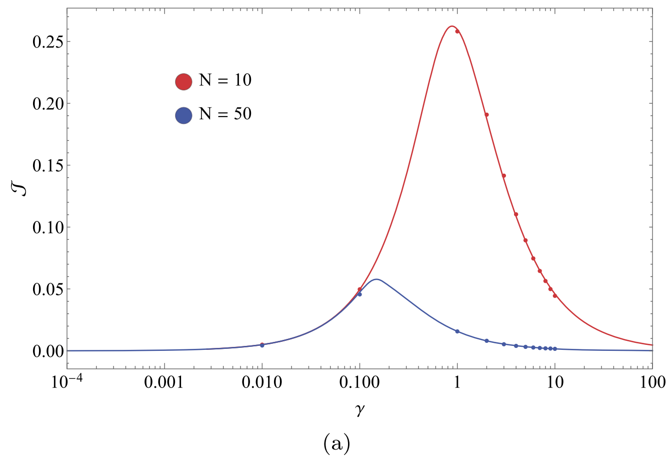
After assuring that the code is working, we can further our analysis by studying the current for different coupling values γ, for two chain sizes. The steady-state current as a function of γ is shown in Fig. 10(a), where it is compared with the analytical solution (solid lines). As can be seen, the agreement is extremely good. Similarly, in Fig. 10(b), where one can clearly see the change in transport type as L increases, from ballistic to subdiffusive [3]. Finally, in Fig. 11 we compare the magnetization profiles $\langle\sigma_z^i\rangle$ with the exact solutions, which again show perfect agreement.

6.3. *Regimes with no analytical solution*

Finally, we illustrate how our implementation can be used to explore situations which have no analytical solution and therefore rely *solely* on numerical methods. The current as a function of γ for different driving parameters f_i (see Eqs. (3), (17)) and $N = 10$ is illustrated in Fig. 12(a). As can be seen, changing f_i brings significant changes to the steady-state and indeed it vanishes when $f_1 - f_N = 0$. A similar analysis of the magnetization profile is presented in Fig. 12(b).

7. Conclusions

To summarize, in this paper we have detailed an implementation of a DMRG routine suited for dealing with open quantum chains. The implementation is based on the algorithm first presented in [1], and was implemented on the iTensor library [19]. The code is also freely available at [2]. The goal of the implementation is to convert the open problem into the language of traditional DMRG, for which many sophisticated routines have already been developed. A major advantage of this method is that it provides, without any overhead, a simple and effective quantifier of convergence, because the steady state corresponds to the zero energy eigenstate of an effective Hamiltonian. We have presented several benchmarks, analyzing both the convergence of the algorithm as well as comparing it with analytical predictions that are available for a limited choice of parameters. These analyses clearly show that our implementation is suitable for studying transport properties in one-dimensional quantum chains, and it can thus be used to study quantum transport phenomena such as interaction induced current rectification and negative differential conductance.



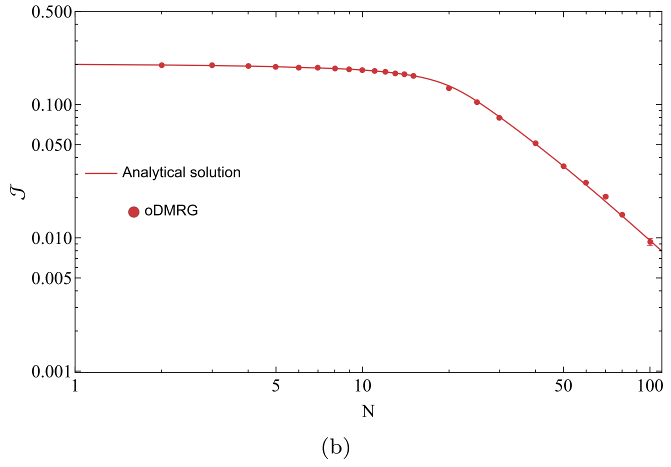
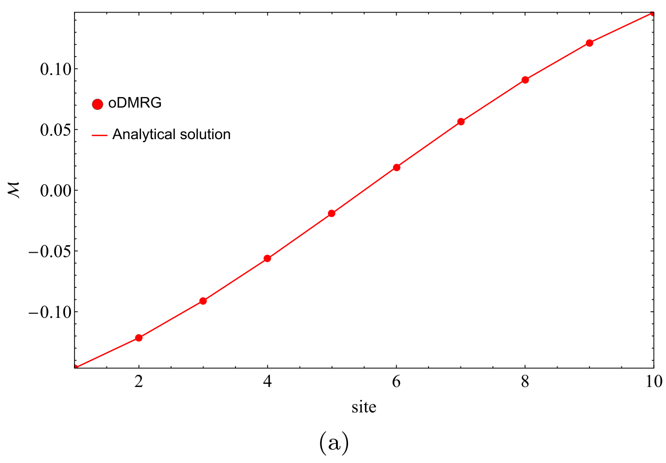


Fig. 10. (a) $\mathcal{J}$ vs. γ for two values of L. The solid line corresponds to the analytical solution from [3] and the dots correspond to the oDMRG simulations. (b) $\mathcal{J}$ vs. N for $\gamma = 0.5$. One can clearly see the change from ballistic to subdiffusive transport. Other parameters were $\Delta = 1$, $f_1 = 1$, $f_N = 0$, $h = 0$.



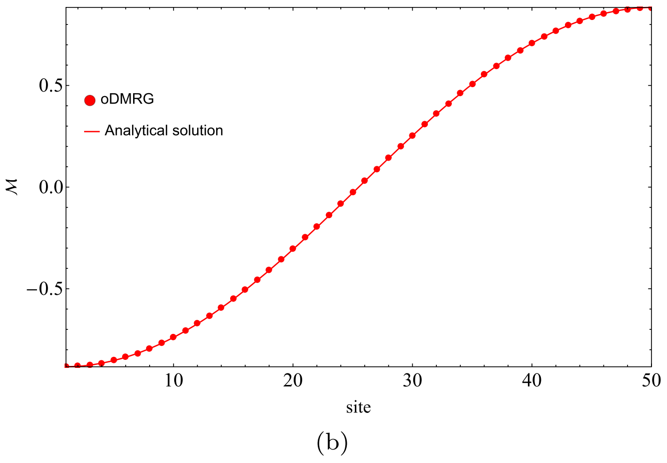
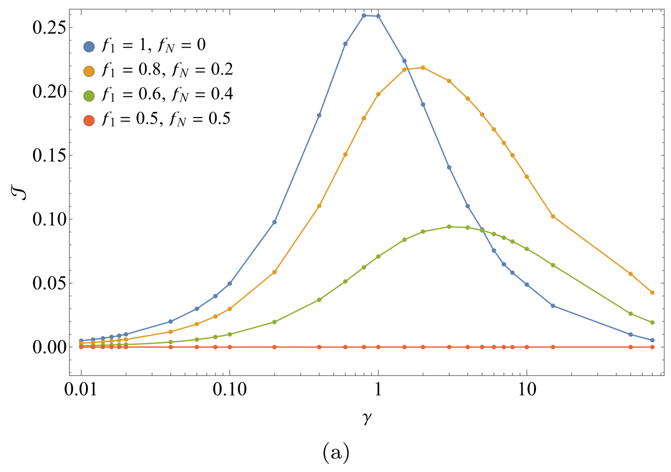


Fig. 11. Magnetization profile $\mathcal{M} = \langle \sigma_z^i \rangle$ as a function of the sites for $\gamma = 0.5$, for (a) $N = 10$ and (b) $N = 50$. The red curves are the exact solutions from [3]. Other parameters were $\Delta = 1$, $f_1 = 1$, $f_N = 0$, $h = 0$.



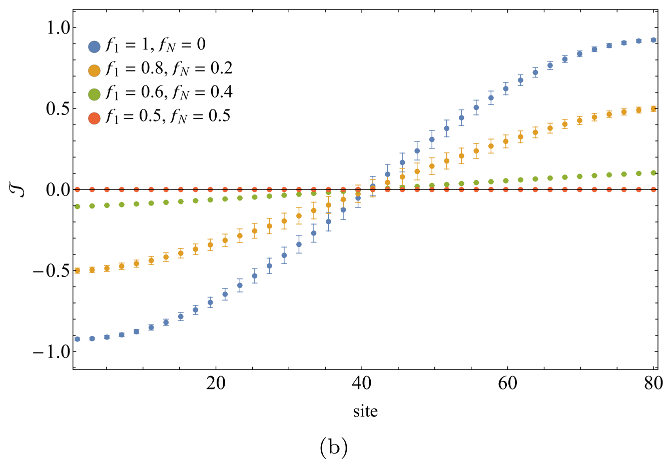


Fig. 12. (a) $\mathcal{J}$ vs. γ for $N = 10$ and different values of f_1, f_N. (b) Magnetization profile for $N = 80$. Other parameters were $\gamma = 0.5$, $\Delta = 1$, $h = 0$.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge fruitful discussion with J. Goold, M. Stoudenmire and L. Gregório. G. T. L. acknowledges the financial support of the São Paulo Funding Agency FAPESP (Grants No.

2017/50304-7, 2017/07973-5 and 2018/12813-0) and the Brazilian funding agency CNPq (Grant No INCT-IQ 246569/2014-0). DP acknowledges support from Ministry of Education - Singapore AcRF MOE Tier-II (project MOE2018-T2-2-142). GTL acknowledges the hospitality of apt44, where part of this work was developed. HPC acknowledges Douglas Casagrande for his support with some of the numerical simulations.

Appendix A. Available functions (and how to use them)

`LRN-sites.h`

This class is constructed with the $(LR)^N$ formalism in mind, and it therefore rearranges the indices of a N-sized tensor network accordingly. It defines the right (R) and left (L) versions of S_x, S_y, S_z, S_+, S_-, as well as combinations of those, which are used in the construction of more complex operators, such as $(S_-)R(S_-)L$ and $(S_+S_-)L$, and so on. It is fitting to be used by any new implementations, and its functionality is akin to the iTensor implemented `SpinHalf` class.

`MakeIVEC`

The MakeIVEC function sets the entries of a tensor network to match those of vec(I). It is called as

```
MakeIVEC(MPS &rho, int N)
```

The ordering is done in accordance with the indices of the input and the $(LR)^N$ formalism. This function is used as the initial guess state that is input into the simulation. This allows one to start with a valid physical state and also ensures that simulations can be re-done by starting from the same state. It is a void function, overwriting the values in `rho`.

`LdLXXZConstruct`

This function handles the creation of the Liouvillian MPO. It can receive either constant values of the parameters γ, h and Δ, or variable ones, which can be input into a vector. The same applies to the temperatures $f_1 \dots f_N$. It returns an MPO object, and is called using the following line.

```
LdLXXZConstruct(SiteSet &sites, double Delta, vector<int>
  dissipatorsVec,
     vector<double> dissipatorsTempValues, <double> gammaVec,
     vector<double> hVec)
```

`WarmUp`

The warm-up routine functions as a simple set of DMRG sweep with fixed bond-dimension. It has been explained in detail in Sec. 5.5. It is a void type function, which means it simply overwrites the tensor network object. It is called with the following line.

```
     Args& args = {"Quiet", false}, Args& argsDMRG = {"Quiet",
  true})
```

References

[1] J. Cui, J.I. Cirac, M.C. Bañuls, Phys. Rev. Lett. 114 (2015) 220601.
[2] ITensor-based oDMRG library (version 2.0), https://github.com/heitorc7/oDMRG, 2020.
[3] G.T. Landi, D. Karevski, Phys. Rev. B, Condens. Matter Mater. Phys. 91 (2015) 1–6.
[4] M. Žnidarič, Phys. Rev. Lett. 106 (2011) 220601.
[5] T. Prosen, M. Žnidarič, J. Stat. Mech. Theory Exp. (2009) 2009, P02035.
[6] G.T. Landi, E. Novais, M.J. de Oliveira, D. Karevski, Phys. Rev. E 90 (2014) 042142.
[7] L. Schuab, E. Pereira, G.T. Landi, Phys. Rev. E 94 (2016) 042122.
[8] V. Balachandran, G. Benenti, E. Pereira, G. Casati, D. Poletti, Phys. Rev. E 99 (2019).
[9] G. Lindblad, Commun. Math. Phys. 48 (1976) 119–130.
[10] V. Gorini, A. Kossakowski, E.C.G. Sudarshan, J. Math. Phys. 17 (1976) 821.
[11] C. Guo, D. Poletti, Phys. Rev. B 96 (2017) 165409.
[12] S.R. White, Phys. Rev. Lett. 69 (1992) 2863–2866.
[13] U. Schollwöck, Ann. Phys. 326 (2011) 96–192.
[14] K. Mølmer, Y. Castin, J. Dalibard, J. Opt. Soc. Am. B 10 (1993) 524–538.
[15] A. Daley, Adv. Phys. 63 (2014).
[16] L. Bonnes, A.M. Läuchli, Superoperators vs. trajectories for matrix product state simulations of open quantum system: A case study, 2014.
[17] A.J. Daley, C. Kollath, U. Schollwöck, G. Vidal, J. Stat. Mech. Theory Exp. (2004) 2004, P04005.
[18] H. Weimer, A. Kshetrimayum, R. Orús, Simulation methods for open quantum many-body systems, 2019.
[19] M. Fishman, S.R. White, E.M. Stoudenmire, The itensor software library for tensor network calculations, 2020.
[20] S. Maniscalco, Phys. Rev. A 75 (2007) 062103.
[21] S.R. White, Phys. Rev. B, Condens. Matter Mater. Phys. 72 (2005) 1–4.
[22] M.-D. Choi, Linear Algebra Appl. 10 (1975) 285–290.
[23] A. Jamiołkowski, Rep. Math. Phys. 3 (1972) 275–278.
[24] E. Mascarenhas, H. Flayac, V. Savona, Phys. Rev. A 92 (2015) 022116.
[25] A.A. Gangat, T. I, Y.J. Kao, Phys. Rev. Lett. 119 (2017) 010501.
[26] T. Prosen, Phys. Rev. Lett. 107 (2011) 137201.