Unfolding a quantum master equation into a system of real-valued equations: Computationally effective expansion over the basis of SU(N) generators

A. Liniov,[1] I. Meyerov,[2] E. Kozinov,[2] V. Volokitin,[2] I. Yusipov,[3] M. Ivanchenko,[3] and S. Denisov[4]
[1]*Software Engineering Department, Lobachevsky State University of Nizhny Novgorod, 603950 Nizhny Novgorod, Russia*
[2]*Mathematical Software and Supercomputing Technologies Department, Lobachevsky State University of Nizhny Novgorod, 603950 Nizhny Novgorod, Russia*
[3]*Department of Applied Mathematics, Lobachevsky State University of Nizhny Novgorod, 603950 Nizhny Novgorod, Russia*
[4]*Department of Computer Science, OsloMet - Oslo Metropolitan University, N-0130 Oslo, Norway*

(Received 15 January 2019; published 11 November 2019)

Dynamics of an open N-state quantum system is often modeled with a Markovian master equation describing the evolution of the system density operator. By using generators of SU(N) group as a basis, the density operator can be transformed into a real-valued "coherence-vector." A generator of the dissipative evolution, so-called "Lindbladian," can be expanded over the same basis and recast in the form of a real matrix. Together, these expansions result is a nonhomogeneous system of $N^2 - 1$ real-valued linear ordinary differential equations. Now one can, e.g., implement standard high-performance algorithms to integrate the system of equations forward in time while being sure in exact preservation of the trace (norm) and Hermiticity of the density operator. However, when performed in a straightforward way, the expansion turns to be an operation of the time complexity $O(N^{10})$. The complexity can be reduced when the number of dissipative operators is independent of N, which is often the case for physically meaningful models. Here we present an algorithm to transform quantum master equation into a system of real-valued differential equations and propagate it forward in time. By using a specific scalable model, we evaluate computational efficiency of the algorithm and demonstrate that it is possible to handle the model system with $N = 10^3$ states on a single node of a computer cluster.

DOI: 10.1103/PhysRevE.100.053305

I. INTRODUCTION

The most conventional approach to modeling of the dynamics of an open quantum system, i.e., a system interacting with its environment, is to use a Markovian master equation [1,2]. Such an equation describes the evolution of the system density operator ϱ, $\dot{\varrho} = \mathcal{L}(t)\varrho$, and its key ingredient is a time-dependent (in general) generator of evolution, *Lindbladian* $\mathcal{L}(t)$ [1].

To define a semigroup, which fulfill conditions of trace preservation and complete positivity, a generator (henceforth referred to as "*Lindbladian*") has to be of the so-called Gorini–Kossakowski–Sudarshan–Lindblad (GKSL) form [3–5],

$$\mathcal{L}(t)\varrho = \mathcal{L}_H(t)\varrho + \mathcal{L}_D(t)\varrho = -\mathrm{i}[H(t), \varrho] + \sum_{p=1}^{P} \gamma_p \mathcal{D}_p(\varrho),$$

$$\mathcal{D}_p(\varrho) = \frac{1}{2}([L_p, \rho L_p^\dagger] + [L_p\rho, L_p^\dagger]), \tag{1}$$

where $H(t)$ is a time-dependent Hamiltonian and the set of quantum dissipative operators, $\{L_p\}$, $p = 1, ..., P$, capture the action of the environment on the system (formally, any complex $N \times N$ matrix could be chosen as an operator L_p). Dissipative operators act via P "channels" with nonnegative rates γ_p. As a theoretical tool, the GSKL Eq. (1) is very popular in quantum optics [2], cavity optomechanics [6], and quantum electrodynamics [7,8]; it is also used in the context of ultracold atom physics [9,10].

When Lindbladian $\mathcal{L}$ is time-independent, the structure of the GSKL equation ensures the existence of an asymptotic state ϱ^A (at least one), which is a nontrivial zero eigen-element (kernel) of $\mathcal{L}$ [11]. When the Lindbladian is time periodic, $\mathcal{L}(t+T) = \mathcal{L}(t)$, Floquet theory [12] applies, and the asymptotic density operator is time-periodic with the same period T, $\varrho^A(t+T) = \varrho^A(t)$ [13]. In either case, the main challenge consists in explicit numerical evaluation of the matrix form of ϱ^A or $\varrho(t)$ [14].

Leaving aside recently developed tensor methods [15], which apply to lattice systems [16] only, there are three means to find ϱ^A or $\varrho(t)$ numerically. Here we only briefly list them (we refer the interested readers to the introduction of Ref. [17] for a more detailed discussion). First, one may use spectral methods (complete/partial diagonalization and different kinds of iterative algorithms [18]) to calculate ϱ^A as an eigen-element of $\mathcal{L}$. Next, one can propagate the system density operator forward in time, by numerically integrating the GKSL equation. Finally, one can unravel the GKSL Eq. (1) into a set of stochastic realizations, called "quantum trajectories" (QTs) [19–21], and thus transform the problem into a task of statistical sampling over QTs (which have to be propagated for a long time to approach ϱ^A [17]).

Here we address the second option; namely, we consider propagation of the density operator forward in time by numerically integrating the GKSL equation. This strategy was already implemented in a number of works; it is also included in such popular open-source package as QuTiP [22]. The first step in the realization of this idea is a vectorization of the density operator, based either on the straightforward row(column)-wise unfolding of the density matrix or usage of the matrix unit basis, $G_\beta \equiv G_{j,k} = |j\rangle\langle k|$, and some arbitrary bijection $\beta \Leftrightarrow (j,k)$. The vectorization renders the GKSL equation in a system of complex-valued linear differential equations, which is then propagated by using some standard high-order integrators [23]. Neither of the discussed vectorization accounts for (i) the norm conservation, $\mathrm{Tr}\varrho(t) = 1$, (ii) Hermiticty, $\varrho^\dagger(t) = \varrho(t)$, and (ii) nonnegativity, $\varrho(t) \geqslant 0$, of the density operator. At the same time, it well known that the first two conditions can be accounted explicitly [24] by using an orthonormal (Hilbert-Schmidt) basis of traceless Hermitian operators, which transform the GKSL equation into a set of *real*-valued linear differential equations [3,27–29]. For a single qubit this procedure is well-known as the Bloch-vector representation and it leads to the famous Bloch equations [1]. For an $N = 3$ system it can be realized by using eight Gell-Mann matrices [30]. For any $N > 3$ it can be performed [27,28] by using a complete set of infinitesimal generators of the SU(N) group [31], rendering density matrix in form of the so-called "coherence-vector" [27]. However, this strategy was never implemented in practice for $N > 4$, to the best of our knowledge. We guess that one of the main problems which prevents the usage SU(N) unfolding is its computational complexity (see Sec. V). This aspect has not been discussed in the literature; at the same time it is an interesting technical problem, for two reasons, "physical" and "computational" ones.

First, the coherence-vector representation allows for an alternative quantification of entanglement in multipartite systems; see, e.g., Refs. [32–34]. It also provides a tool to investigate a "geometry" of quantum states [35] by using the condition of positivity [36,37]. Second, by performing expansion over the SU(N) generators, a search for ϱ^A could be transformed into a linear programming problem [38]. This allows us to use a toolbox of parallel simplex methods, developed for large optimization problems, and implement them on a cluster or supercomputer [39], thus opening a way to large model systems. These reasons are our main motivation.

Here we present an implementation which realizes the expansion of a Lindbladian over the basis of SU(N) generators [40]. It is tailored to handle model systems with number P of dissipative channels which grows sub-linearly with N or just remains constant. The latter condition is not very limiting; in fact, many currently studied models fulfill it. One could think, e.g., of two "baths," L_1 and L_2, acting on the ends of an 1D spin chain [41] or of an ensemble of qubits interacting with a photonic mode in a leaky cavity [42].

The paper is organized as follows: In Sec. II we outline the idea of the expansion and present main definitions. In Sec. III we introduce a scalable model system. Section IV is devoted to the implementation of the algorithm on a cluster; its performance and scalability are analyzed in Sec. V. These results are summarized, together with an outline of further perspectives, in Sec. VI.

II. EXPANSION OF A LINDBLADIAN OVER THE BASIS OF SU(N) GENERATORS

In Refs. [27,28] the expansion is presented in detail; here we only summarize the results and introduce necessary definitions and notations.

The basis consists of $M = N^2 - 1$ $N \times N$ traceless Hermitian matrices, among which there are [43]

(1) $N(N-1)/2$ symmetric, $S^{(j,k)} = \frac{1}{\sqrt{2}}(G_{j,k} + G_{k,j})$, $1 \leqslant j < k \leqslant N$,

(2) $N(N-1)/2$ antisymmetric, $J^{(j,k)} = -\frac{\mathrm{i}}{\sqrt{2}}(G_{j,k} - G_{k,j})$, $1 \leqslant j < k \leqslant N$,

(3) and $N-1$ diagonal, $D^l = \frac{1}{\sqrt{l(l+1)}}(\sum_{k=1}^{l} G_{k,k} - lG_{l+1,l+1})$, $1 \leqslant l \leqslant N-1$,

which are forming a set $\bar{F} = \{F_i\}$, $i = 0, ..., M$. This set is complemented with the identity matrix, $F_0 = \mathbb{1}$. Now we have a basis which is orthonormalized with respect to the trace, $\mathrm{Tr}(F_iF_j) = \delta_{ij}$, and complete. Important are commutators and anticommutators of the basis elements,

$$[F_i, F_j] = \mathrm{i}\sum_{k=1}^{N^2-1} f_{ijk}F_k, \tag{2}$$

$$\{F_i, F_j\} = \frac{2}{N}F_0\delta_{ij} + \sum_{k=1}^{N^2-1} d_{ijk}F_k, \tag{3}$$

with f_{ijk} (d_{ijk}) being a real completely antisymmetric (symmetric), with respect to permutation of any pair of indices, tensor,

$$\begin{aligned} z_{ijk} &= f_{ijk} + \mathrm{i}d_{ijk}, && i,j,k = \overline{1, N^2-1},\\ f_{ijk} &= \mathrm{i}\mathrm{Tr}(F_k[F_i, F_j]), && i,j,k = \overline{1, N^2-1},\\ d_{ijk} &= \mathrm{Tr}(F_k\{F_i, F_j\}), && i,j,k = \overline{1, N^2-1}. \end{aligned} \tag{4}$$

A density operator can be expanded over this basis,

$$\rho = \frac{1}{N}F_0 + \sum_{i=1}^{N^2-1} v_iF_i, \tag{5}$$

where *coherence-vector* (also called "generalized Bloch vector" or simply "Bloch vector" [29]) $\bar{v} = (v_1, v_2, ..., v_M)$ consist of real-valued elements [28]. As a Hermitian operator, $H(t)$ can also be expanded, $H(t) = \sum_{i=1}^{N^2-1} h_i(t)F_i$ (without loss of generality, henceforth we assume the Hamiltonian to be traceless). The unitary part of the Lindbladian, $\mathcal{L}_H$, yields a $M \times M$ matrix Q, with elements

$$q_{sm} = \sum_{i=1}^{N^2-1} f_{ims}h_i, \tag{6}$$

which is skew-symmetric, $Q^T = -Q$, due to the antisymmetry of tensor f. Thus, the unitary part of the GKSL Eq. (1) transforms into $\dot{\bar{v}} = Q\bar{v}$.

Expansion of the dissipative part of the Lindbladan is more involved. In the original GKSL equation, this part can be rewritten in the following form [3]:

$$\mathcal{L}_D = \frac{1}{2}\sum_{j,k=1}^{N^2-1} a_{jk}([F_j, \rho F_k^\dagger] + [F_j\rho, F_k^\dagger]), \tag{7}$$

where complex $M \times M$ matrix $A = \{a_{jk}\}$ is positive semidefinite (at any instant of time), $A \geqslant 0$, and has rank P. It can be diagonalized, $\tilde{A} = SAS^\dagger = \mathrm{diag}\{\gamma_1, \gamma_2, ..., \gamma_P\}$, and dissipators can be expressed as $\bar{L} = S^\dagger\bar{F}$. By using spectral decomposition, $A = \sum_{p=1}^{P} \bar{l}_p\bar{l}_p^\dagger$, the dissipative part can be recast into

$$\mathcal{L}_D = \frac{1}{2}\sum_{p=1}^{P}\gamma_p \sum_{j,k=1}^{N^2-1} l_{p;j}l^*_{p;k}([F_j, \rho F_k^\dagger] + [F_j\rho, F_k^\dagger]). \tag{8}$$

To the equation for the coherence-vector $\mathcal{L}_D$ contributes with $M \times M$ matrix R and vector K,

$$\frac{dv(t)}{dt} = [Q(t) + R]v(t) + K, \tag{9}$$

with components

$$r_{sm} = -\frac{1}{2}\sum_{p=1}^{P}\gamma_p \sum_{i_1,i_2,j=1}^{N^2-1} l_{p;i_1}l^*_{p;i_2}(z_{i_1 jm}f_{i_2 js} + z^*_{i_2 jm}f_{i_1 js}),$$
$$m, s = \overline{1, N^2-1}, \tag{10}$$

$$k_s = \frac{\mathrm{i}}{N}\sum_{p=1}^{P}\gamma_p \sum_{i,j=1}^{N^2-1} l_{p;i}l^*_{p;j}f_{ijs}, \quad s = \overline{1, N^2-1}. \tag{11}$$

Summation over p in Eqs. (10) and (11) renders a trivial parallelization, because the computation of the sum for every dissipative operator L_d can be performed independently. so henceforth we restrict consideration to the case $P = 1$ (a single dissipative operator).

III. TEST-BED MODEL

As a test bed we use a model describing $N-1$ indistinguishable interacting bosons, which are hopping between the sites of a periodically modulated dimer. The model is described with a time-periodic Hamiltonian

$$H(t) = J(b_1^\dagger b_2 + b_1 b_2^\dagger) + \frac{2U}{(N-1)}\sum_{j=1}^{2} n_j(n_j - 1) + \varepsilon(t)(n_2 - n_1), \tag{12}$$

where b_j and $b_j^\dagger$ are the annihilation and creation operators of an atom at site j, while $n_j = b_j^\dagger b_j$ is the operator of number of particles on jth site, J is the tunneling amplitude, $U/(N-1)$ is the interaction strength (normalized by a number of bosons), and $\varepsilon(t)$ represents the modulation of the local potential. $\varepsilon(t)$ is chosen as $\varepsilon(t) = \varepsilon(t+T) = E + A\theta(t)$, where E is the stationary energy offset between the sites and A is the dynamic offset. This type of Hamiltonian has been studied theoretically [44–47] and was implemented in several experiments [48,49]. Two types of the driving are are popular: (i) piecewise constant periodic driving, $\theta(t) = 1$ for $0 \leqslant t < T/2$, $\theta(t) = -1$ for $T/2 \leqslant t < T$ and (ii) sinusoidal driving, $\theta(t) = \sin(t)$.

As a dissipative operator we use

$$L = \frac{\gamma}{N-1}(b_1^\dagger + b_2^\dagger)(b_1 - b_2). \tag{13}$$

This dissipative coupling tries to "synchronize" the dynamics on the sites by constantly recycling antisymmetric out-phase mode into symmetric in-phase one [9]. Since the jump operator is non-Hermitian, the asymptotic state is different (in general) from the maximally mixed state, $\varrho^A \neq \mathbb{1}/N$. This type of dissipation was realized in recent experiments with trapped ions [50].

As a result of modulations, the asymptotic state is characterized by a time-periodic density operator, $\varrho^A(t+T) = \varrho^A(t)$, so that the asymptotic state has to be specified over one period of the driving, $\varrho^A(t_s)$, $t_s = t \bmod T \in [0, T]$ [13].

Note that term $H^J = (b_1^\dagger b_2 + b_2^\dagger b_1)$ is represented by a tridiagonal matrix (in the Fock basis). The components $H^U = \frac{2U}{N-1}\sum_{j=1}^{2} n_j(n_j-1)$ and $H^E = \varepsilon(t)(n_2 - n_1)$ are diagonal matrices. Thus, the Hamiltonian is represented by a symmetric tridiagonal matrix. Dissipative operator L gives an antisymmetric tridiagonal matrix. Altoghether, this means that the resulting Lindbladian is very sparse.

IV. IMPLEMENTATION

The expansion described in Sec. II, together with the propagation, can be implemented in four steps; see Table I.

During the initialization step, the algorithm reads initial data, allocates memory, and initializes main data structures. During the second step, it prepares data for subsequent calculations. Namely, the coefficients of the expansion of the matrices H and L in the basis $\{F_i\}$, and the coefficients of the ODE system, Eq. (9), are calculated. In the third step, the ODE system is integrated up to time T. Finally, during the finalization step, the computed results are saved to files and memory is released.

The implementation of the expansion (step 2) seems to be straightforward but a brute force direct realization leads to a high time complexity and memory requirements, even in the case of sufficiently sparse Hamiltonians and dissipator matrix. Here we propose an implementation that allows to substantially reduce memory requirements and time complexity. This is achieved by taking into account sparsity patterns of the involved matrices and performing operations only with nonzero elements. In this section we estimate the implementation complexity and the amount of memory required for the general case of dense matrices (both of a Hamiltonian and dissipators), as well as for the application considered in Sec. III. Note that all operator matrices have size N and we use NZ to denote the number of nonzero elements in them.

A. Step 1: Initialization

The initialization of the Hamiltonian and the dissipator matrices requires $O(N^2)$ operations and $O(N^2)$ space in general case (for dense matrices). When dealing with the dimer model, we use the sparse matrix storage format CSR, which requires $O(N)$ operations for initialization and $O(N)$ space.

B. Step 2: Data preparation

First, we need to compute the coefficients h_i, l_i of the expansion of the matrices H and L in the basis $\{F_i\}$ (step 2.1). The coefficients of the elements of the basis $S^{(j,k)}$, $J^{(j,k)}$ are calculated for $O(1)$ operations, thanks to the form of the basis matrices, which gives the total time complexity $O(NZ_H + NZ_L)$. The coefficients for all D^l are calculated with $O(N)$ operations. Thus, the total time complexity is $O(NZ_H + NZ_L + N)$. We store the coefficients in two arrays. The first array contains the values and takes $O(NZ)$ space.

TABLE I. Main algorithm

Step	Substep
1. Initialization	1.1. Read the initial data from configuration files.
	1.2. Allocate and initialize memory.
2. Data preparation	2.1. Compute the coefficients h_i, l_i of the expansion of the matrices H and L in the basis $\{F_i\}$.
	2.2. Compute the coefficients f_{ijk}, d_{ijk}, z_{ijk} by Eqs. (4).
	2.3. Compute the coefficients q_{sm} by Eq. (6).
	2.4. Compute the coefficients k_s by Eq. (11).
	2.5. Compute the coefficients r_{sm} by Eq.(10).
	2.6. Compute the initial value $v(0)$.
3. ODE integration	3.1. Integrate the ODE (9), over time to $t = T$ by means of the Runge-Kutta method.
	3.2. Compute $\rho(T)$ by Eq. (5).
4. Finalization	4.1. Save the results.
	4.2. Release memory.

The second array represents the expansion and contains -1 for zero coefficients and indexes in the first array for nonzero ones. It takes $O(N^2)$ space. This allows quickly accessing the element and checking if it is equal to zero. In the case of dense matrices $NZ_H = NZ_L = N^2$, while for the dimer model $NZ_H = NZ_L = 3N - 2$.

Next, we need to compute the coefficients f_{ijk}, d_{ijk}, z_{ijk} (step 2.2). The number and values of these coefficients depend only on the size of the problem, N. Their direct calculation by Eq. (4) requires $2(N^2 - 1)^3$ matrix multiplications for the basis matrices $\{F_i\}$. Most multiplications require a fixed number of operations independent of N. Multiplication with the participation of the matrices $\{D^l\}$ require up to $O(N)$ operations. The total time complexity is therefore $O(N^6)$. It can be significantly reduced by taking into account the sparsity patterns of the matrices $\{F_i\}$. The main idea is to account for nonzero coefficients only. We found that it is possible to determine the set of nonzero coefficients analytically. Namely, the number of nonzero coefficients f_{ijk} is $NZ_F = 5N^3 - 9N^2 - 2N + 6$, the number of nonzero coefficients d_{ijk} is $NZ_D = 6N^3 - N(21N + 7)/2 + 1$, and the complexity of calculating each coefficient is $O(1)$. Thus, the overall time complexity is $O(N^3)$. It should be noted that, despite the apparent uniform distribution of $O(N^3)$ nonzero coefficients in tensors of size $N^2 \times N^2 \times N^2$, every set of two-dimensional sections of the tensor $\{d_{ijk}, i = \text{const}\}$, $\{d_{ijk}, j = \text{const}\}$, $\{d_{ijk}, k = \text{const}\}$ includes $O(N)$ two-dimensional sections, with $O(N^2)$ elements in each of them. It results in $O(N^3)$ elements in total in every such sub-tensor. The example of nonzero coefficients distribution for $N = 3$ is shown in Fig. 1).

When calculating the coefficients, we use the coordinate sparse matrix format, which requires $O(NZ_F + NZ_D)$ space. Next, we convert the tensors from the coordinate format to the CRS format. For this we employ the quicksort algorithm to dictionary sort triples of indices (i, j, k). The resulted complexity of this step is $O([NZ_F + NZ_D] \log[NZ_F + NZ_D]) \sim O(N^3 \log N)$ operations and it requires $O(N^3)$ space. The subsequent addition of the tensors $Z = F + D$ has the time complexity $O(NZ_F + NZ_D) \sim O(N^3)$ and requires another $O(N^3)$ space. Two points are of importance. First, it is possible to pre-compute the tensor Z and store it in a file. Second, we can get rid of memory allocation and element computation for the tensors D, F, and Z. Instead, elements of these tensors can be computed on fly, when needed. We use this approach to decrease memory consumption.

The next part of the algorithm (step 2.3) computes coefficients q_{sm}, Eq. (6). A straightforward implementation requires $O(N^6)$ operations. However, we again can significantly decrease the computational load thanks to sparsity of the tensor F. Computations can be done by using the following recipe:

(1) Represent the tensor F in the coordinate format (F') in which only nonzero $f'_{ijk} = f_{ijk}h_i$ are stored. It takes $O(N^3)$ time and $O(NZ_{F'})$ space. In the general case, $NZ_{F'}$ depends essentially on the form of the Hamiltonian. If there are nonzero elements on its main diagonal, then $NZ_{F'} \sim O(N^3)$, otherwise $O(N \cdot NZ_H)$.

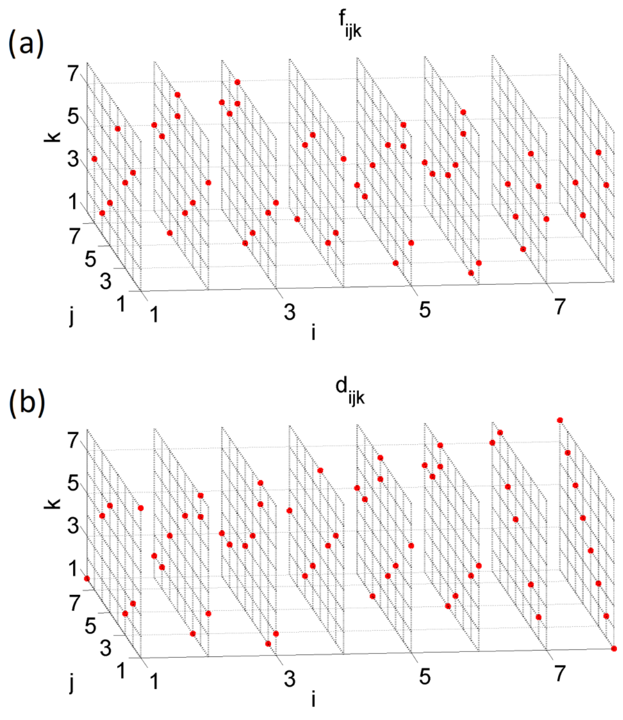

FIG. 1. Sparsity patterns of tensors $\{d_{ijk}\}$ (a) and $\{f_{ijk}\}$ (b) for $N = 3$. The red points indicate nonzero elements.

(2) Sort F' elements by a pair of indices (k, j). It can be done by using the counting sort (or radix sort) algorithm, which results in $O(N^3)$ scaling in time and $O(NZ_{F'})$ in space.

(3) Calculate sums $q_{sm} = \mathrm{Re}(\sum_{i=1}^{N^2-1} f_{ims}h_i)$ and form the Q matrix in the CRS format. The matrix can be filled in two steps. At the first, the number of nonzero elements in each row is computed. Next, the elements are calculated and indexes are written. All this can be done in $O(NZ_{F'})$ time and requires $O(NZ_{F'})$ space. The resulted time and space complexity of step 2.3 is $O(N^3)$. The number of nonzero elements in the resulting matrix Q depends essentially on the form of the Hamiltonian, but it does not exceed $O(N^3)$.

During the next step 2.4, we compute the coefficients k_s by Eq. (11).

A direct calculation of the coefficients requires $O(N^4)$ operations. This can be decreased if vector l is sparse. To do this, we convert the vector into the coordinate format, find all nonzero f_{ijs} for each nonzero $l_il_j^*$, and add $l_il_j^*f_{ijs}$ to corresponding k_s. It requires $O(NZ_L^2)$ time. The same result can be achieved by using the sparsity of the tensor F. Thus, we can just go through all nonzero elements of F, adding $l_il_j^*f_{ijs}$ to corresponding k_s. It requires $O(N^3)$ time. The choice of the algorithm is determined by the relation between N and NZ_L. We use the second option because it is independent of the input data. In any case, storing the vector k_s requires $O(N^2)$ space.

Next (step 2.5), we compute the coefficients r_{sm} by using Eq. (10). Again, a straightforward calculation of the coefficients—even for $P = 1$—requires $O(N^{10})$ operations, which is unacceptable. The following details should to be taken into account to reduce the scaling:

(1) Tensors F and Z are sparse and contain $O(N^3)$ elements each;

(2) Tensors F and Z are filled in such a way that their two-dimensional 'sections' (matrices) contain $O(N)$ to $O(N^2)$ elements;

(3) Vector l is sparse if the dissipator is sparse. For the dimer model this vector contains $NZ_L = 3N - 2$ nonzero elements ($N^2 - 1$ elements in the general case).

The corresponding algorithm reads:

(1) Convert tensors F and Z to the coordinate format (F' and Z' correspondingly). Both tensors store only nonzero elements $l_i^*f_{ijk}$ and l_iz_{ijk}. It can be done in $O(N^3)$ time and space.

If matrix L is dense, then tensors F' and Z' contain $NZ_{F'}$, $NZ_{Z'} \sim O(N^3)$ nonzero elements, and the two-dimensional sections of F' and Z' contain $O(N^2)$ nonzero elements.

Thanks to the sparsity of matrix L, the number of nonzero elements is much smaller in the dimer model. Namely, it is $O(N^2)$ for F' and Z' and $O(N)$ for their 2D sections.

(2) Sort elements of F' and Z' on the second and the third indexes. When using the counting sort or radix sort algorithm, it takes $O(NZ_{F'} + NZ_{Z'})$ time and space.

(3) Compute sums of elements with the same second and third indices. The results can be represented as matrices F'', Z'' in the coordinate format, ordered by the second and third coordinates. It requires $O(NZ_{F'} + NZ_{Z'})$ time and space.

TABLE II. Algorithm complexity

Algorithm step	Complexity for dense H and L		Complexity for the dimer model	
	Time	Space	Time	Space
1. Initialization	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N^2)$
2. Data preparation	$O(N^5 \log N)$	$O(N^4)$	$O(N^3 \log N)$	$O(N^3)$
3. ODE integration	$O(N^4)$	$O(N^2)$	$O(N^3)$	$O(N^2)$
4. Finalization	$O(N^2)$	–	$O(N^2)$	–

If the matrix L is dense, then $O(N)$ rows in the matrices F'' and Z'' contain $O(N^2)$ nonzero elements. The remaining $O(N^2)$ rows can contain $O(N)$ nonzero values. In the application considered in this paper all rows of the matrices F'' and Z'' contain no more than $O(N)$ nonzero elements.

(4) Store the matrix R as an array of the red-black trees where every row of the matrix is represented as a separate tree. For each $l = \overline{1, N^2 - 1}$ compute all products $l_{i_1}l_k^*(z_{i_1jm}f_{i_2js} + z_{i_2jm}^*f_{i_1js})$ and add the results to the corresponding elements of the matrix R.

It can be done in $O(N \cdot (N^2)^2 \cdot \log N) + O(N^2 \cdot N^2 \cdot \log N) \sim O(N^5 \log N)$ time and $O(N^4)$ space for a dense matrix and $O(N \cdot (N)^2 \log N)$ time and $O(N^3)$ space for the dimer model.

(5) Convert matrix R to the CRS format. It requires $O(N^4 \log N)$ time and $O(N^4)$ space in the general case, and $O(N^3 \log N)$ time and $O(N^3)$ space for the model problem.

Consequently, the step 2.5 requires $O(N^5 \log N)$ time and $O(N^4)$ space for the general case and $O(N^3 \log N)$ time and $O(N^3)$ space for the dimer model.

Finally, during step 2.6 we compute initial coherence-vector $v(0)$ and then initiate time propagation. For this purpose, we expand the initial state $\rho(0)$ in the F-basis, Eq. (5). It takes $O(N^2)$ time and $O(N^2)$ space (see explanations for step 2.1).

C. Step 3: ODE integration

During this step we integrate the linear real-valued ODE system, Eq. (9), over one period of modulations, T (step 3.1), and compute resulted $\rho(T)$ (step 3.2). The complexity of the ODE integration is determined by the method used for propagation and the number of nonzero elements in matrices Q and R (up to $O(N^4)$ elements for dense matrices). For example, the time complexity of one time step is $O(N^4)$ for the Runge-Kutta integration. However, the time complexity of one step is $O(N^3)$ for the dimer model, Sec. III. The integration of the corresponding ODE system by the forth-order Runge-Kutta method requires $O(N^2)$ additional space for storing intermediate results. The computation of $\varrho(T)$ has complexity $O(N^2)$, both in time and space.

D. Step 4: Finalization

During this step we save results to files and release memory. The time complexity is $O(N^2)$.

Resulted time and space complexity estimations are presented in Table II.

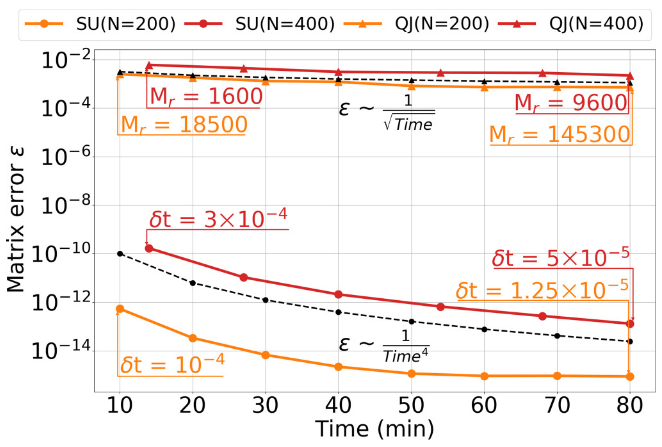

FIG. 2. Error in the calculated density matrix by the quantum jump method and the proposed method as a function of computation time. For $N = 200$, integration step $1E-4$ corresponds to a computation time of 18 500 trajectories using the quantum jump method and step $1.25E-5$ corresponds to 1 45 300 trajectories. For $N = 400$, these values are equal to $\delta t = 3E-4$ and 1600 trajectories, and $\delta t = 5E-5$ and 9600 trajectories, respectively. The initial state is $\varrho(0) = |0\rangle\langle 0|$.

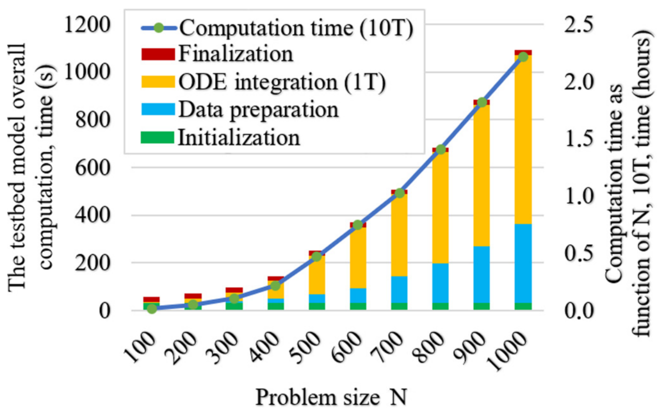

FIG. 3. Computation time as a function of N. The total computation time (line; right y axis) was measured for the propagation time $10T$. More detail analysis was performed for the propagation time T (bars; left y axis). To get estimates for longer propagation times, one has to scale linearly computation time of the ODE integration step.

V. PERFORMANCE ANALYSIS

For performance tests we use a node of the Lobachevsky supercomputer [51] with a 2×8-core Intel Xeon CPU E5-2660, 2.20 GHz, 128 GB RAM. The code was compiled with Intel C++ Compiler, Intel Math Kernel Library and Intel MPI from the Intel Parallel Studio XE suite of development tools.

To start, we compare the performance of the algorithm with the performance of a recently proposed implementation of quantum trajectory method [17]. The implementation has only one tunable parameter, the depth S, which defines the smallest step of propagation, $\delta t_{\min} = 2^{-S}\delta t_{\max}$, and thus the error in determining time of the next jump. The propagation in between jumps is numerically exact and performed by using nonunitary matrix operators (exponentinated effective Hamiltonians). The maximal time step $\delta t_{\max}$ is set to be equal to the mean value of inter-jump time (which is estimated during a warm-up phase). For further details we refer to Ref. [17].

We use the picewise constant periodic driving (see Sec. III) and propagate dimer from the pure initial state $\varrho(0) = |0\rangle\langle 0|$ (all bosons are seating on the right site). The system is propagated to $t = 10T$, by sampling over quantum trajectories and integrating the coherence vector, and then the obtained solutions, $\varrho_{qt}(10T)$ and $\varrho_{cv}(10T)$, are compared with the test solution $\varrho^*(10T)$ obtained by propagating Eq. (9) with the RK4 integrator and time step $dt = 10^{-6}$. The error is defined as the spectral norm of the difference matrix, $\epsilon = \|\varrho^*(10T) - \varrho_s(10T)\|$, $s \in \{'qt','cv'\}$. We estimated the total computation time (on the node) t_c needed to reach the accuracy ϵ; the results are presented on Fig. 2.

The error scaling in the case of the QT sampling is intuitive: spectral norm decays as $t_c^{-\frac{1}{2}}$, which is consistent with the linear scaling of the number M_r of realizations with the computation time. In the case of the coherence vector propagation, the error quickly saturates already for $dt = 2E-5$ (the corresponding computation time is 50 min for $N = 200$). However, there is a substantial difference even for a maximal computation time used in the tests; namely, it is near four order of magnitude. We conclude that, at least for a sparse Hamiltonian and a single jump operator, as in the test-bed model, the numerical propagation of the coherence vector is much more efficient, from the point of view of the time complexity, than the QT sampling.

Next we analyze scaling of computation times of different steps as functions of N. To do so we set propagation time to T. The results are shown in Fig. 3 (bars). First, it shows that the time of the preparation step, although significant, is substantially smaller than the time of ODE integration. Taking into account that the preparation step is performed once, while integration time scales linearly with the actual time of propagation, we conclude that it is the latter that determines the total computation time of the algorithm. It is evident that the initialization and finalization steps do not make a significant impact on the overall computational time.

Now we compare theoretical predictions obtained for $N = 10^2$ for the propagation time $10T$ upon the increase of the problem size to $N = 10^3$; see Fig. 4. Namely, we compare computation times of the most time consuming steps and the overall computation time with the theoretical predictions. To do so, we scale (as discussed above) the measured estimate and compare them with actual ones. First observation is that the relation obtained for the preparation step saturates to a constant value. The relation for the integration step slowly goes down with the increase of N; therefore, the estimated obtained early can be considered as an upper bound and the actual number of the operations during this step is less then expected. It is not a surprise if we recall that the matrix sparsity scales nontrivially with N. Finally, we analyze the scaling of the memory use. First we consider how it scales with the propagation time. The results of the analyses are presented with Fig. 5, where the memory used during different algorithm steps is shown as function of N. It peaks during

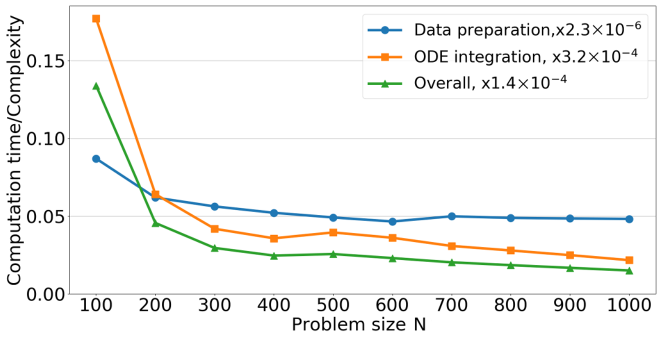

FIG. 4. Ratio between the actual computation time and the corresponding asymptotic predictions, Table II (time complexity for the dimer model) For a fixed N, the ratio is scaled to make the total time (to perform all steps) equal to one.

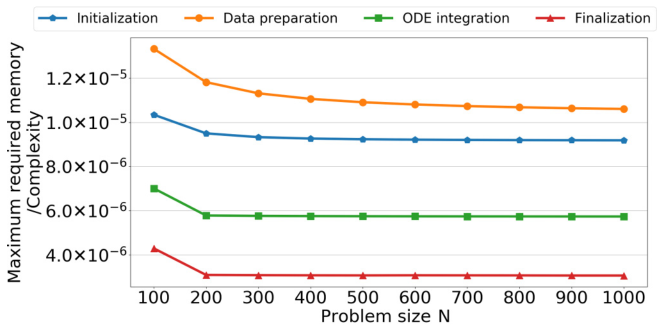

FIG. 6. Ratio between the maximal memory use measured in computation experiments and the corresponding asymptotic predictions, Table II (space complexity for the dimer model). For a fixed N, the ratio is scaled to make the total time (to perform all steps) equal to one.

the data preparation step, when matrix Q is calculated. It is noteworthy that the memory use is around 100 GB for $N = 10^3$; this already sets certain demands to the computational cluster. Now we compare the results of computation experiments with theoretical estimates. For this we tune the size of the model from $N = 10^2$ to 10^3 and calculate the ratio between the maximal memory use obtained in numerical experiments and estimates; see Fig. 6. Thus, the latter are confirmed.

VI. CONCLUSIONS

We have presented an algorithm to transform a quantum master equation in the Gorini–Kossakowski–Sudarshan–Lindblad form into a system of $N^2 - 1$ linear real-valued ordinary differential equations (ODEs). We included a propagation stage into an implementation of the algorithm so that the obtained system can be integrated forward in time. By using a test-bed model, we evaluated the performance of the implementation and demonstrated that it is possible to propagate a system with $N = 10^3$ states on a single node of a standard cluster.

A. Applications

We see our implementation as a computational tool to propagate large open models of general type, which may also include time-dependent Hamiltonians and dissipation rates. There is an interesting context to this which appeared very recently.

During last five years, random sampling of complex states was considered as a way to prove "quantum supremacy" [52]. Conventionally, such sampling is performed by constructing random quantum circuits and passing some trivial pure initial state (e.g., a product state) through them. A proposal to perform sampling of mixed quantum states, by using dissipative effects and time-modulated Hamiltonians, was presented in a recent work [53]. The consideration is based on the Lindbald equation, which described the action of a dissipative "circuit," and its unrevealing into ensemble of quantum trajectories [19–21]. The QT approach brings the issue of the variance around the relevant expectational values (e.g., diagonal elements of the density matrix) and the speed of the sampling should be estimated by restricting the variance to some threshold value ϵ and estimating the total number of trajectories needed to reach this threshold. As it follows from our results (though for a specific model, see Fig. 2), the numerical propagation of the density matrix, by encoding it into a coherence-vector, can be more efficient from the point of view of computational resources.

Parallelization is another potential resource to refine the implementation. Note that the calculations are performed in two stages, data preparation and integration of the obtained ODE system. At the same time, significant memory consumption during the data preparation stage is the main bottleneck limiting further increase of the model size. In this regard, the parallelization should help to satisfy memory requirements for models of larger size (we estimate it as $N \backsimeq 2000$–3000), by using the resources of several nodes. We do not expect a drastic reduction of the propagation time from the parallelization; yet it is not the key limiting factor in this case.

The asymptotic state of a model with a time-independent Lindbladian can potentially be calculated by using the method of linear programming [38]. Namely, the task can be reduced to minimization of the right-hand side of the ODE system, Eq. (9), by varying real-valued coherence vector υ. This provides with a possibility of futher parallelization since there is a toolbox of parallel methods designed to solve such problems [39].

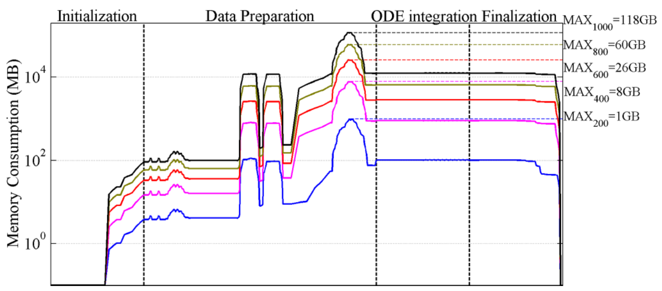

FIG. 5. Scaling of memory used during the different steps of the algorithm as functions of N, for $N = 200, 400, 600, 800, 1000$ (from bottom to top).

B. Alternatives

There are recent advances which serve alternatives to our approach. If a model many-body system is local, i.e., every local "body" (usually spin) is interacting, then both through Hamiltonian and dissipative operators, only with a small number of its neighbors, the corresponding Lindbladian is sparse. In this case, it is expediently to try tensor-based methods to propagate the system [15]. However, it could be that jump operators do not destroy long-range entanglement in the system (a special dissipation can even work in the opposite direction, see, e.g., Ref. [9]); in this case the tensor-based methods do not apply and our approach becomes of relevance.

In Ref. [54] it was proposed to calculate the asymptotic state of a Lindbladian $\mathcal{L}$ by transforming the problem into finding the ground state of a fictitious Hamiltonian $H_{\mathcal{L}} = \mathcal{L}^{\dagger}\mathcal{L}$. Hamiltonian $H_{\mathcal{L}}$ inherits this sparsity of $\mathcal{L}$, for the price of squaring the number of nonzero elements [55]. In this case, the matrix product state representation [56] may work well—if the ground state of $H_{\mathcal{L}}$ is characterized by a short-ranged entanglement. Normalization, $\mathrm{Tr}\varrho = 1$, and Hermeticity, $\varrho^{\dagger} = \varrho$, conditions can be built into the algorithm; however, similar to our approach, the positivity, $\varrho \geqslant 0$, is hard to control. It would be interesting, by using several models with sparse Lindbladians, to gauge performance of the algorithm from Ref. [54] vs our implementation.

When a Lindbladian is not sparse, the coherence vector approach may become advantageous—but in this case its efficiency has to be re-evaluated (especially versus QT approach). How dense a typical Lindbladian could be? It is not enough to have a model with a full Hamiltonian because it would only yield a Lindbladian with a block diagonal structure (with only $1/N$ fraction of nonzero entries). Physically meaningful jump operators are usually acting on a very small subspace of the total system Hilbert space each; e.g., in case of a spin network model, these operators act either individually on every spin or, at most, on pairs of neighboring spins. Moreover, jump operators are typically identical (though their rate can be different). Such jump operators can diminish the sparsity of the Lindbaladian but they cannot make if fully dense. In the case of physically relevant model, one would probably deal with an intermediate situation.

ACKNOWLEDGMENTS

The authors acknowledge support of the Russian Foundation for Basic Research Grant No. 18-37-00277 (Sec. V), President of Russian Federation Grant No. MD-6653.2018.2, and Ministry of Education and Science of the Russian Federation Research Assignment No. 1.5586.2017/BY. We acknowledge the use of computational resources provided by Lobachevsky University (Lobachevsky supercomputer), Moscow State University (Lomonosov supercomputer), Joint Supercomputer Center of the Russian Academy of Sciences (MVS-10P supercomputer), and Intel (Intel Endeavour supercomputer).

[1] H. -P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems* (Oxford University Press, Oxford, 2002).
[2] H. J. Carmichael, *An Open Systems Approach to Quantum Optics* (Springer, Berlin, 1993).
[3] V. Gorini, A. Kossakowski, and E. C. G. Sudarshan, J. Math. Phys. **17**, 821 (1976).
[4] G. Lindblad, Commun. Math. Phys. **48**, 119 (1976).
[5] D. Chruściński and S. Pascazio, Open Sys. Inf. Dyn. **24**, 1740001 (2017).
[6] M. Aspelmeyer, T. J. Kippenberg, and F. Marquardt, Rev. Mod. Phys. **86**, 1391 (2014).
[7] J. Jin, D. Rossini, R. Fazio, M. Leib, and M. J. Hartmann, Phys. Rev. Lett. **110**, 163605 (2013).
[8] M. Fitzpatrick, N. M. Sundaresan, A. C. Y. Li, J. Koch, and A. A. Houck, Phys. Rev. X **7**, 011016 (2017).
[9] S. Diehl, A. Micheli, A. Kantian, B. Krausa, H. P. Buchler, and P. Zoller, Nat. Phys. **4**, 878 (2008).
[10] M. Marcuzzi, J. Schick, B. Olmos, and I. Lesanovsky, J. Phys. A: Math. Theor. **47**, 482001 (2014).
[11] Depending on symmetry properties of a Lindbladian, there could be several asymptotic states; see V. V. Albert and L. Jiang, Phys. Rev. A **89**, 022118 (2014); V. V. Albert, B. Bradlyn, M. Fraas, and Liang Jiang, Phys. Rev. X **6**, 041031 (2016).
[12] V. A. Yakubovich and V. M. Starzhinskii, *Linear Differential Equations with Periodic Coefficients* (Wiley, New York, 1975).
[13] M. Hartmann, D. Poletti, M. Ivanchenko, S. Denisov, and P. Hänggi, New J. Phys. **19**, 083011 (2017).
[14] Note them when all dissipative operator are Hermitian, $L_p^{\dagger} = L_p$, the asymptotic density matrix is trivial normalized identity $\varrho^A = \frac{\mathbb{1}}{N}$ ("infinite temperature state").
[15] M. Zwolak and G. Vidal, Phys. Rev. Lett. **93**, 207205 (2004); A. H. Werner, D. Jaschke, P. Silvi, M. Kliesch, T. Calarco, J. Eisert, and S. Montangero, *ibid.* **116**, 237201 (2016).
[16] Lattice quantum systems are many-body systems of lattice topologies, i.e., with next-neighbor interaction between its particles or elements.
[17] V. Volokitin, A. Liniov, I. Meyerov, M. Hartmann, M. Ivanchenko, P. Hänggi, and S. Denisov, Phys. Rev. E **96**, 053313 (2017).
[18] P. D. Nation, J. R. Johansson, M. P. Blencowe, and A. J. Rimberg, Phys. Rev. E **91**, 013307 (2015).
[19] R. Dum, A. S. Parkins, P. Zoller, and C. W. Gardiner, Phys. Rev. A **46**, 4382 (1992).
[20] K. Mølmer, Y. Castin, and J. Dalibard, J. Opt. Soc. Am. B **10**, 524 (1993).
[21] M. B. Plenio and P. L. Knight, Rev. Mod. Phys. **70**, 101 (1998).
[22] R. Johansson, P. D. Nation, and F. Nori, Comput. Phys. Commun. **183**, 1760 (2012).
[23] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems* (John Wiley and Sons, Chichester, 1991).
[24] An interesting discussion on density matrix parametrization for spin systems, which accounts also for positivty, can be found in Refs. [25,26]. However, this parametrization is not discussed in

the context of evolution, neither unitary nor disisipative, so it is no clear how to propagate system when its state is encoded as proposed.

[25] N. Il'in, E. Shpagina, F. Uskov, and O. Lychkovskiy, J. Phys. A: Math. Theor. **51**, 085301 (2018).

[26] E. Shpagina, F. Uskov, N. Il'in, and O. Lychkovskiy, Stationary Schrödinger equation with density matrices instead of wave functions, arXiv:1812.03056 (2018).

[27] K. Lendi, J. Phys. A: Math. Gen. **20**, 15 (1987).

[28] R. Alicki and K. Lendi, *Quantum Dynamical Semigroups and Applications*, Lecture Notes in Physics, Vol. 286 (Springer, Berlin, 1987).

[29] G. Kimura, Phys. Lett. A **314**, 339 (2003).

[30] M. Gell-Mann, Phys. Rev. **125**, 1067 (1962).

[31] H. Georgi, *Lie Algebras in Particle Physics* (Addison Wesley Publishing Company, Boston MA, 1982).

[32] C.-S. Yu and H.-S. Song, Phys. Rev. A **72**, 022333 (2005).

[33] A. S. M. Hassan and P. S. Joag, Quantum Inf. Comput. **8**, 0773 (2008).

[34] T. Zhou, J. Cui, and G. L. Long, Phys. Rev. A **84**, 062105 (2011).

[35] I. Bengtsson and K. Życzkowski, *Geometry of Quantum States: An Introduction to Quantum Entanglement* (Cambridge University Press, Cambridge, 2006).

[36] M. S. Byrd and N. Khaneja, Phys. Rev. A **68**, 062322 (2003).

[37] G. Kimura and A. Kossakowski, Open Syst. Info. Dyn. **12**, 207 (2005).

[38] G. B. Dantzig and M. N. Thapa, *Linear Programming: Introduction* (Springer, NY, 2006).

[39] J. A. J. Hall, Comput. Manag. Sci. **7**, 139 (2010).

[40] M. Hammermesh, *Group Theory and Its Application to Physical Problems* (Addison-Wesley Publishing Company, Reading, MA, 1962)

[41] M. Žnidarič, A. Scardicchio, and V. K. Varma, Phys. Rev. Lett. **117**, 040601 (2016).

[42] M. Xu, D. A. Tieri, E. C. Fine, J. K. Thompson, and M. J. Holland, Phys. Rev. Lett. **113**, 154101 (2014).

[43] There are at least two different definitions of the basis based on different normalization conditions. Here we used the one from Refs. [27,28] (see Ref. [29] for the alternative definition).

[44] C. Weiss and N. Teichmann, Phys. Rev. Lett. **100**, 140408 (2008).

[45] A. Vardi and J. R. Anglin, Phys. Rev. Lett. **86**, 568 (2001).

[46] F. Trimborn, D. Witthaut, and S. Wimberger, J. Phys. B: At. Mol. Opt. Phys. **41**, 171001 (2008).

[47] D. Poletti, J.-S. Bernier, A. Georges, and C. Kollath, Phys. Rev. Lett. **109**, 045302 (2012).

[48] C. Gross, T. Zibold, E. Nicklas, J. Esteve, and M. K. Oberthaler, Nature **464**, 1165 (2010).

[49] J. Tomkovič, W. Muessel, H. Strobel, S. Löck, P. Schlagheck, R. Ketzmerick, and M. K. Oberthaler, Phys. Rev. A **95**, 011602(R) (2017).

[50] P. Schindler, M. Müller, D. Nigg, J. T. Barreiro, E. A. Martinez, M. Hennrich, T. Monz, S. Diehl, P. Zoller, and R. Blatt, Nat. Phys. **9**, 361 (2013).

[51] https://www.top500.org/site/50549.

[52] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, Nat. Phys. **14**, 595 (2018).

[53] E. Kapit, P. Roushan, C. Neill, S. Boixo, and V. Smelyanskiy, Entanglement and complexity of interacting qubits subject to asymmetric noise, arXiv:1905.01792 (2019).

[54] J. Cui, J. I. Cirac, and M. C. Bañuls, Phys. Rev. Lett. **114**, 220601 (2015).

[55] The test-bed model, Sec. III, is local (when written in the Fock basis) and therefore sparse.

[56] F. Verstraete and J. I. Cirac, Phys. Rev. B **73**, 094423 (2006); M. B. Hastings, J. Stat. Mech. (2007) P08024; U. Schollwöck, Ann. Phys. **326**, 96 (2011).