Classical simulation of quantum circuits using a multi-qubit Bloch vector representation of density matrices

Qunsheng Huang[1, *] and Christian B. Mendl[1, 2, †]

[1] *Technical University of Munich, Department of Informatics, Boltzmannstraße 3, 85748 Garching, Germany*
[2] *Technical University of Munich, Institute for Advanced Study, Lichtenbergstraße 2a, 85748 Garching, Germany*

(Dated: February 10, 2022)

In the Bloch sphere picture, one finds the coefficients for expanding a single-qubit density operator in terms of the identity and Pauli matrices. A generalization to n qubits via tensor products represents a density operator by a real vector of length 4^n, conceptually similar to a statevector. Here, we study this approach for the purpose of quantum circuit simulation, including noise processes. The tensor structure leads to computationally efficient algorithms for applying circuit gates and performing few-qubit quantum operations. In view of variational circuit optimization, we study "backpropagation" through a quantum circuit and gradient computation based on this representation, and generalize our analysis to the Lindblad equation for modeling the (non-unitary) time evolution of a density operator.

I. INTRODUCTION

Density operators are capable of describing (thermal) quantum ensembles and non-unitary noise processes [1]. In a textbook-type simulation on classical computers, one would store density operators as complex Hermitian matrices in memory, as is currently implemented in widely-used software libraries [2–4]. Here, we advocate and study an alternative approach, namely directly working with a tensorized Bloch vector representation, i.e., an expansion in terms of Pauli strings, see Eq. (3) below. As general insight, this form leads to equations analogous to statevector simulations for quantum circuits, where the multi-qubit Bloch vector assumes the role of the quantum state, and operations on density operators (like applying a unitary matrix by conjugation) become matrix-vector products. The data layout in memory is well suited for single- or two-qubit quantum gates due to the tensor structure, as compared to a literal implementation of matrix conjugations, which involves products from the left and right. As additional advantages, the Bloch representation involves only real-valued quantities, applying general quantum channels does not require a summation over Kraus operators, and gradient computation with respect to gate parameters (see Sect. V) becomes conceptually simpler.

Generalizations of the Bloch sphere representation for higher-level systems or multiple qubits have been investigated in various forms [5–8], and the observation that tensor products of Pauli matrices with real coefficients form a basis of Hermitian matrices can be considered common knowledge. Our main contributions here are efficient algorithms and practical details for quantum circuit simulation and variational optimization based on this representation. To clarify, the terms "density operator" and "density matrix" are used synonymously.

We have implemented the methods described in this work in a Julia software toolbox called *Qaintum* [9]. As demonstration, we perform parametric optimization of a density matrix via the variational quantum thermalizer (VQT) algorithm described in [10], see Sect. VI.

II. TENSORIZED BLOCH REPRESENTATION FOR MULTIPLE QUBITS

Let us recall the well-known Bloch sphere representation for density matrices: the Bloch vector $\vec{r} \in \mathbb{R}^3$ associated with a single-qubit density matrix ρ is defined via the relation

$$\rho = \frac{1}{2}(I_2 + \vec{r} \cdot \vec{\sigma}), \tag{1}$$

where $\vec{\sigma} = (X, Y, Z)$ is the Pauli vector and I_2 the 2×2 identity matrix. The property that ρ is positive semidefinite is equivalent to $\|\vec{r}\| \leq 1$ [1].

By setting $r_0 = 1$ and $\sigma_0 = I_2$, we can rewrite Eq. (1) as

$$\rho = \frac{1}{2} \sum_{j=0}^{3} r_j \sigma_j. \tag{2}$$

Slightly more generally, one observes that the vector space of Hermitian 2×2 matrices is isomorphic to $\mathbb{R}^4$. We can generalize this construction to an arbitrary number of qubits via tensor products of Pauli matrices: any n-qubit density matrix ρ has a unique representation as

$$\rho = \frac{1}{2^n} \sum_{j_{n-1}=0}^{3} \cdots \sum_{j_0=0}^{3} r_{j_{n-1},\ldots,j_0}\, \sigma_{j_{n-1}} \otimes \cdots \otimes \sigma_{j_0}. \tag{3}$$

We will denote the tensor $r \in (\mathbb{R}^4)^{\otimes n} \simeq \mathbb{R}^{4^n}$ as the multi-qubit Bloch vector associated with ρ; the condition $\mathrm{Tr}[\rho] = 1$ is then equivalent to $r_{0,\ldots,0} = 1$. For enumerating entries as in $r_{j_{n-1},\ldots,j_0}$, we adopt the convention that j_0 is the fastest-varying index.

* keefe.huang@tum.de
† christian.mendl@tum.de

Note that the Pauli strings in Eq. (3) form an orthonormal basis in the space of Hermitian $2^n \times 2^n$ matrices, with inner product $\langle A|B\rangle = \frac{1}{2^n}\mathrm{Tr}[AB]$.

III. UNITARY OPERATIONS

A unitary map U acting on a quantum state transforms its density matrix representation by conjugation:

$$\rho' = U\rho U^\dagger. \tag{4}$$

This holds in particular for quantum gates appearing in quantum circuits. Working directly with the Bloch vector representation of ρ in Eq. (3), the conjugation (4) becomes

$$r' = \mathsf{U} r \tag{5}$$

when interpreting the Bloch vector r indeed as a vector, $r \in \mathbb{R}^{4^n}$, with $\mathsf{U} \in \mathbb{R}^{4^n \times 4^n}$ an orthogonal matrix. For quantum circuits, which typically involve single- or two-qubit gates, we will provide details for efficient implementations of (5), without assembling the matrix U. In the following, the sans-serif styling (as in U) will denote the matrix associated with the Bloch representation in (5), given a complex unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$.

As a concrete example, consider the phase gate $S = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ acting on a single qubit. Since

$$SXS^\dagger = Y, \quad SYS^\dagger = -X, \quad SZS^\dagger = Z, \tag{6}$$

the conjugation by S in the Bloch representation (including component zero) reads

$$\mathsf{S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

We summarize the corresponding matrix representations related to the Bloch vector formulation for common quantum logic gates in Appendix A.

A. Application of quantum circuit gates

Let us first consider a single-qubit quantum gate $U^{(1)}$ acting on the ℓ-th qubit, $\ell \in \{0, \ldots, n-1\}$. The unitary matrix on the full n-qubit Hilbert space is thus

$$U = I_{2^{n-1-\ell}} \otimes U^{(1)} \otimes I_{2^\ell}, \tag{8}$$

with I_m denoting the $m \times m$ identity matrix. To efficiently apply this gate to a Bloch vector r, we first reshape r into a $4^{n-1-\ell} \times 4 \times 4^\ell$ tensor, denoted $\tilde{r}$. Then Eq. (5) can be concisely expressed as

$$\tilde{r}'_{:,j,:} = \sum_{k=0}^{3} \mathsf{U}^{(1)}_{jk}\, \tilde{r}_{:,k,:}, \quad j = 0, \ldots, 3, \tag{9}$$

where have used the slice index notation ":" to select all entries along a particular dimension. When specialized for common quantum gates, Eq. (9) can be implemented in a matrix-free form by expanding the sum and keeping only the non-zero terms of a particular $\mathsf{U}^{(1)}$, cf. Appendix A.

Next, consider a two-qubit gate $U^{(2)}$ acting on qubits ℓ_a and ℓ_b, with $\ell_\mathrm{a}, \ell_\mathrm{b} \in \{0, \ldots, n-1\}$, $\ell_\mathrm{a} < \ell_\mathrm{b}$. Now, we reshape r into a $4^{n-1-\ell_\mathrm{b}} \times 4 \times 4^{\ell_\mathrm{b}-\ell_\mathrm{a}-1} \times 4 \times 4^{\ell_\mathrm{a}}$ tensor, again denoted $\tilde{r}$. Eq. (5) then reads

$$\tilde{r}'_{:,j_\mathrm{b},:,j_\mathrm{a},:} = \sum_{k_\mathrm{a},k_\mathrm{b}=0}^{3} \mathsf{U}^{(2)}_{4j_\mathrm{b}+j_\mathrm{a},4k_\mathrm{b}+k_\mathrm{a}}\, \tilde{r}_{:,k_\mathrm{b},:,k_\mathrm{a},:} \tag{10}$$

for $j_\mathrm{a}, j_\mathrm{b} = 0, \ldots, 3$.

The scheme in Eqs. (9) and (10) is straightforwardly generalizable to gates acting on a larger number of qubits.

We remark that the matrix-vector form is easier to parallelize as compared to the matrix conjugations in Eq. (4), where the multiplications from the left and right would naturally be performed one after another. In terms of memory utilization, the Bloch representation requires the same amount of storage as the upper (or lower) triangular part of a Hermitian matrix, but has a more favorable data layout (for predicting memory access patterns) due to the tensor structure.

A short benchmark comparison is presented in Appendix C.

B. Controlled gates

Controlled gates turn out to be somewhat tedious to handle when working with density matrices in the Bloch vector representation. Let us first introduce the following "(anti-)symmetric" operations acting on the space of Hermitian matrices:

$$\mathcal{S}_G(\rho) = \frac{1}{2}\left(G\rho + \rho G^\dagger\right), \tag{11a}$$

$$\mathcal{A}_G(\rho) = \frac{i}{2}\left(G\rho - \rho G^\dagger\right), \tag{11b}$$

with G a complex matrix of compatible dimension. Bloch representations of $\mathcal{S}_U$, $\mathcal{A}_U$ for common quantum gates U are summarized in Appendix A.

Now, consider a unitary gate U controlled by a single qubit; this operation can be written as

$$CU = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U = I + |1\rangle\langle 1| \otimes (U - I). \tag{12}$$

For the scenario of k control qubits, U is active only if all of them are in the $|1\rangle$ state (in the computational basis representation). In terms of Eq. (12), this means generalizing $|1\rangle\langle 1|$ to $|1\cdots 1\rangle\langle 1\cdots 1|$ on the right, i.e.,

$$CU = I + (|1\rangle\langle 1|)^{\otimes k} \otimes (U - I). \tag{13}$$

U is always understood to act on the qubits following the control qubits. Application of CU to density matrices

leads to

$$\begin{aligned} CU\rho CU^\dagger &= \rho \\ &+ (|1\rangle\langle 1|)^{\otimes k} \otimes (U-I)\cdot\rho + \rho\cdot(|1\rangle\langle 1|)^{\otimes k}\otimes(U^\dagger - I) \\ &+ (|1\rangle\langle 1|)^{\otimes k} \otimes (U-I)\cdot\rho\cdot(|1\rangle\langle 1|)^{\otimes k}\otimes(U^\dagger - I) \\ &= \rho \\ &+ 2\mathcal{S}_{(|1\rangle\langle 1|)^{\otimes k}\otimes(U-I)}(\rho) \\ &+ (|1\rangle\langle 1|)^{\otimes k}\left(U\rho U^\dagger - 2\mathcal{S}_U(\rho) + \rho\right)(|1\rangle\langle 1|)^{\otimes k}. \end{aligned} \tag{14}$$

Regarding the last term in (14), note that the conjugations by $|1\rangle\langle 1|$, and evaluation of the expression $U\rho U^\dagger - 2\mathcal{S}_U(\rho) + \rho$, are linear operations acting on different qubits, and thus in particular commute.

In Appendix B we verify the following relations (given complex matrices F, G):

$$\mathcal{S}_{F\otimes G} = \mathcal{S}_F \otimes \mathcal{S}_G - \mathcal{A}_F \otimes \mathcal{A}_G, \tag{15a}$$
$$\mathcal{A}_{F\otimes G} = \mathcal{S}_F \otimes \mathcal{A}_G + \mathcal{A}_F \otimes \mathcal{S}_G. \tag{15b}$$

Recursive application allows us to evaluate $\mathcal{S}_{(|1\rangle\langle 1|)^{\otimes k}\otimes(U-I)}$ appearing in the penultimate line of Eq. (14). Specifically, to expand all combinations of tensor products, we first introduce the shorthand notation

$$\mathcal{E}_G^{(j)} = \begin{cases} \mathcal{S}_G, & j = 0 \\ \mathcal{A}_G, & j = 1 \end{cases} \tag{16}$$

Then, based on Eqs. (15),

$$\begin{aligned} &\mathcal{S}_{(|1\rangle\langle 1|)^{\otimes k}\otimes(U-I)} \\ &= \sum_{j_0,\ldots,j_{k-1}=0}^{1} (-1)^{\lceil (j_0+\cdots+j_{k-1})/2\rceil} \mathcal{E}_{|1\rangle\langle 1|}^{(j_{k-1})} \otimes \cdots \otimes \mathcal{E}_{|1\rangle\langle 1|}^{(j_0)} \\ &\qquad \otimes \mathcal{E}_{U-I}^{(j_0+\cdots+j_{k-1} \bmod 2)}, \end{aligned} \tag{17}$$

where $\lceil\cdot\rceil$ is the "ceil" function (rounding upwards, i.e., closest integer that is greater than or equal to the function argument). In particular, note that the sum in (17) consists of 2^k terms, each of which is a tensor product of linear operators acting on separate qubits. Regarding the last operator, also note that $\mathcal{S}_{U-I} = \mathcal{S}_U - \mathrm{id}$ and $\mathcal{A}_{U-I} = \mathcal{A}_U$, which immediately follows from the definitions (11).

In summary, we have expanded the conjugation by a controlled gate CU, such that operations on the control and target qubits (in the Bloch representation) can be performed sequentially, one control qubit at a time, using the techniques of Sect. III A. Namely, the conjugations by $|1\rangle\langle 1|$ in the last line of (14) can be applied one by one, and likewise for the penultimate line of (14): using the expansion in (17), one can first transform ρ by $\mathcal{E}_{|1\rangle\langle 1|}^{(j_{k-1})}$ (cf. the last row of Table I in the appendix), then the result by $\mathcal{E}_{|1\rangle\langle 1|}^{(j_{k-2})}$ etc. up to $\mathcal{E}_{|1\rangle\langle 1|}^{(j_0)}$, and finally by $\mathcal{E}_{U-I}^{(j_0+\cdots+j_{k-1} \bmod 2)}$.

IV. QUANTUM CHANNELS AND LINDBLAD EQUATION

In general, a quantum channel $\mathcal{E}$ acting on a density matrix ρ admits the following Kraus operator representation [1]:

$$\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger \tag{18}$$

with complex matrices E_k, which are denoted Kraus operators. Quantum channels generalize unitary transformations. Since they are likewise linear, we can still represent them in matrix-vector form, analogous to Eq. (5):

$$r' = \mathsf{E} r, \tag{19}$$

with $r \in \mathbb{R}^{4^n}$ the Bloch vector corresponding to ρ, and $\mathsf{E} \in \mathbb{R}^{4^n\times 4^n}$ a real-valued matrix describing the channel; see Table III for some concrete examples.

For the scenario of a quantum channel affecting one or few qubits within a many-qubit system, observe that the tensor product structure is again preserved by the Bloch representation. In particular, the formulas (9) and (10) for the efficient application are valid for quantum channels as well, after substituting $\mathsf{U}^{(m)}$ by the quantum channel analog $\mathsf{E}^{(m)}$.

An important special case is the time evolution of density matrices when including interactions with the environment, which includes, for example, dissipation. The time dynamics is governed by the following Gorini-Kossakowski-Sudarshan-Lindblad equation [11, 12] (in units of $\hbar = 1$):

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho = \mathcal{L}(\rho) = -i[H,\rho] + \sum_q \left(L_q \rho L_q^\dagger - \frac{1}{2}\left\{L_q^\dagger L_q, \rho\right\}\right), \tag{20}$$

with $[A,B] = AB - BA$, $\{A,B\} = AB + BA$, H the principal system Hamiltonian, and L_q the Lindblad operators. Note that we can use the definitions (11) to express $i[H,\rho] = 2\mathcal{A}_H(\rho)$ and $\frac{1}{2}\{L_q^\dagger L_q, \rho\} = \mathcal{S}_{L_q^\dagger L_q}(\rho)$.

Let L be the matrix corresponding to $\mathcal{L}$ in the Bloch representation, such that the Lindblad equation reads

$$\frac{\mathrm{d}}{\mathrm{d}t} r(t) = \mathsf{L} r(t). \tag{21}$$

In case L is time-independent, (21) has the formal solution

$$r(t) = \mathrm{e}^{\mathsf{L}t}\, r(0) \tag{22}$$

when starting from some initial state $r(0)$ at $t = 0$. We will revisit the Lindblad equation in the context of gradient computation at the end of the following section.

V. BACKPROPAGATION AND GRADIENT COMPUTATION

Let C be a real-valued "cost function" depending on the output state of a quantum channel $\mathcal{E}_{\mathrm{sys}}$, for example

$C = \mathrm{Tr}[M\mathcal{E}_{\rm sys}(\rho_{\rm in})]$, with $\rho_{\rm in}$ the input density matrix and M a measurement operator. For concreteness, we first consider the scenario that $\mathcal{E}_{\rm sys}$ describes a quantum circuit, such that $\mathcal{E}_{\rm sys}(\rho) = V\rho V^\dagger$, with V the overall unitary transformation effected by the circuit gates – the general case will be discussed later in this section. Our goal here is to compute the gradient of C with respect to individual parametrized gates in the circuit, which is an essential task for, e.g., variational circuit optimization. For that purpose, we perform a "backpropagation" pass through the quantum circuit, which originates from a recursive application of the chain rule for differentiation. Conceptually, in the framework of (classical) artificial neural networks with feedforward architecture, each quantum gate corresponds to a layer in such a network. The setup is sketched in Fig. 1, with the density matrix ρ describing an intermediate quantum state, and $\rho' = U(\theta)\rho U(\theta)^\dagger$ the next state after applying the parametrized gate $U(\theta)$.

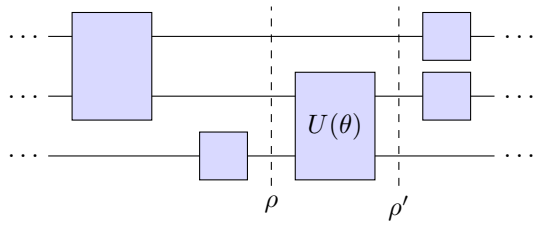


FIG. 1. Schematic excerpt of a parametrized quantum circuit, with intermediate states ρ and ρ'. The blue boxes represent unitary circuit gates.

In the following, we use the notation

$$\overline{a} = \frac{\partial C}{\partial a} \tag{23}$$

to denote the gradient of C with respect to some variable or parameter a (not to be confused with complex conjugation). We will only encounter real-valued quantities for gradient computation due to the Bloch representation.

Now consider Eq. (5): $r'_j = \sum_k \mathsf{U}_{jk} r_k$ for all j. Since C depends on U only via r', the gradient of C with respect to the entry U_{jk} obeys

$$\overline{\mathsf{U}_{jk}} = \frac{\partial C}{\partial r'_j}\frac{\partial r'_j}{\partial \mathsf{U}_{jk}} = \overline{r'_j}\, r_k. \tag{24}$$

In other words, $\overline{\mathsf{U}}$ is the outer product of $\overline{r'}$ and r:

$$\overline{\mathsf{U}} = \overline{r'} \otimes r. \tag{25}$$

To obtain the gradient with respect to a m-qubit gate $U^{(m)}$, we start from the relation (9). (The following derivation works analogously for $m \geq 2$; to simplify the notation, we only show the case $m = 1$ here.)

$$\overline{\mathsf{U}^{(1)}_{jk}} = \sum_{u,v} \frac{\partial C}{\partial \tilde{r}'_{u,j,v}}\frac{\partial \tilde{r}'_{u,j,v}}{\partial \mathsf{U}^{(1)}_{jk}} = \sum_{u,v} \overline{\tilde{r}'_{u,j,v}}\, \tilde{r}_{u,k,v}. \tag{26}$$

The sum on the right of (26) can be interpreted as tracing out the remaining qubits (which $U^{(1)}$ leaves invariant). For a general m-qubit gate acting on qubits $\ell_{\rm a}, \ell_{\rm b}, \ldots$, we thus arrive at the formula

$$\overline{\mathsf{U}^{(m)}} = \mathrm{Tr}_{0:n-1\setminus\{\ell_{\rm a},\ell_{\rm b},\ldots\}}\big[\overline{r'} \otimes r\big], \tag{27}$$

where the partial trace runs over the qubits which are unaffected by $\mathsf{U}^{(m)}$. To efficiently evaluate the partial trace in practice, one can form the entries of $\overline{r'}\otimes r$ "on the fly", without storing the outer product as a full matrix.

To complete the gradient computation with respect to unitary gates, let us consider the case that $\mathsf{U}^{(m)}$ depends on real parameters, and denote one such parameter by θ. Then

$$\overline{\theta} = \sum_{j,k} \overline{\mathsf{U}^{(m)}_{jk}}\,\frac{\partial}{\partial\theta}\mathsf{U}^{(m)}_{jk} = \mathrm{Tr}\Big[\overline{\mathsf{U}^{(m)}}^T \frac{\partial}{\partial\theta}\mathsf{U}^{(m)}\Big]. \tag{28}$$

Specialized to circuit gates, the entries of $\partial\mathsf{U}^{(m)}/\partial\theta$ can usually be evaluated analytically, and one can then efficiently implement the sum in (28) by only keeping the non-zero terms.

A step in the backpropagation requires the computation of $\overline{r}$ based on $\overline{r'}$. Again starting from (5), this is achieved by

$$\overline{r_k} = \sum_j \frac{\partial C}{\partial r'_j}\frac{\partial r'_j}{\partial r_k} = \sum_j \overline{r'_j}\,\mathsf{U}_{jk}, \tag{29}$$

which reads in matrix-vector notation

$$\overline{r} = \mathsf{U}^T\,\overline{r'}. \tag{30}$$

Since U is orthogonal, its transpose is also its inverse, thus (30) describes the application of the inverse quantum gate to $\overline{r'}$. Directly based on (5), the same relation holds for the Bloch vectors as well:

$$r = \mathsf{U}^T\, r'. \tag{31}$$

As has been noted before [13], one can recompute intermediate quantum states (in our case Bloch vectors) on the fly during the backward pass, which has the potential to significantly decrease computer memory requirements. (For comparison, classical neural networks typically keep the "activations" of intermediate layers in memory.) Moreover, we can reuse the techniques in Sect. III A for the backward pass.

For a general (parametrized) quantum channel $\mathcal{E}$ which maps $\rho' = \mathcal{E}(\rho)$ and is represented by Eq. (19), the formulas (25), (27), (28), (30) literally agree after substituting U and $\mathsf{U}^{(m)}$ by E and $\mathsf{E}^{(m)}$, respectively. Namely, the above derivation based on Eq. (5) likewise works when starting from Eq. (19). However, in the case when $\mathcal{E}$ is not invertible, it is (in general) infeasible to reconstruct ρ from ρ'; thus ρ must be kept in memory between the forward and backward pass.

For completeness, we remark that the backpropagation and gradient computation method described here is, in

particular, applicable to a composition of quantum channels $\mathcal{E}_{\text{sys}} = \mathcal{E}_1 \circ \mathcal{E}_2 \circ \ldots$, again based on the chain rule.

Finally, let us discuss gradient computation based on the Lindblad equation (20), which can be regarded as a special case of "trainable" differential equations [14, 15]. We start from the matrix-vector representation (21), to be solved in the time interval $t \in [0, t_\text{f}]$. We assume that the cost function C explicitly depends on the state $r(t_\text{f})$ at the final time point. Since $r(t_\text{f}) = \mathrm{e}^{\mathsf{L}(t_\text{f}-t)}\, r(t)$ (for time-independent L), it holds that

$$\overline{r}(t) = \mathrm{e}^{\mathsf{L}^T(t_\text{f}-t)}\, \overline{r}(t_\text{f}), \tag{32}$$

analogous to (30). Thus, $\overline{r}$ obeys the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{r}(t) = -\mathsf{L}^T\, \overline{r}(t), \tag{33}$$

which has to be solved backwards in time, with "initial condition" $\overline{r}(t_\text{f}) = \partial C/\partial r(t_\text{f})$. It turns out that (33) remains valid for time-dependent L as well. Namely, one can express (21) as ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t} r(t) = f(r(t), t) \tag{34}$$

with $f(r,t) = \mathsf{L}(t)\, r$, and then use that the "adjoint" $\overline{r}$ is governed by [15, 16]

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{r}(t)^T = -\overline{r}(t)^T \frac{\partial f(r(t),t)}{\partial r(t)}. \tag{35}$$

To relate (33) to the original Lindblad equation (20), let us define the dual $\mathcal{L}^*$ (acting on Hermitian matrices) via the condition

$$\mathrm{Tr}[\tau \mathcal{L}(\rho)] = \mathrm{Tr}[\mathcal{L}^*(\tau)\rho] \tag{36}$$

for all τ, ρ. Note that $\mathcal{L}^*$ describes time evolution in the Heisenberg picture and is the analogue of L^T, and thus Eq. (33) can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{\rho} = -\mathcal{L}^*(\overline{\rho}) = -i[H,\overline{\rho}] - \sum_q \left(L_q^\dagger \overline{\rho} L_q - \frac{1}{2}\left\{ L_q^\dagger L_q, \overline{\rho} \right\} \right). \tag{37}$$

To compute the gradient with respect to a time-independent L, we assume that L is parametrized by some variable $\theta \in \mathbb{R}$, and use the identity [17]

$$\frac{\partial}{\partial\theta}\, \mathrm{e}^{\mathsf{L}t} = \int_0^t \mathrm{e}^{\mathsf{L}(t-t')}\, \frac{\partial \mathsf{L}}{\partial\theta}\, \mathrm{e}^{\mathsf{L}t'}\, \mathrm{d}t'. \tag{38}$$

Then, by varying a single matrix entry,

$$\begin{aligned} \overline{\mathsf{L}_{jk}} &= \sum_\ell \frac{\partial C}{\partial r_\ell(t_\text{f})} \frac{\partial r_\ell(t_\text{f})}{\partial \mathsf{L}_{jk}} = \overline{r}(t_\text{f})^T \frac{\partial}{\partial \mathsf{L}_{jk}}\, \mathrm{e}^{\mathsf{L}t_\text{f}}\, r(0) \\ &= \overline{r}(t_\text{f})^T \int_0^{t_\text{f}} \mathrm{e}^{\mathsf{L}(t_\text{f}-t)} (e_j \otimes e_k)\, \mathrm{e}^{\mathsf{L}t}\, \mathrm{d}t\, r(0) \\ &= \int_0^{t_\text{f}} \overline{r_j}(t) r_k(t)\, \mathrm{d}t. \end{aligned} \tag{39}$$

Here $e_j \otimes e_k$ is the matrix with a single non-zero entry 1 at index (j,k), and we have used the relation (32). Thus, writing (39) in matrix notation,

$$\overline{\mathsf{L}} = \int_0^{t_\text{f}} \overline{r}(t) \otimes r(t)\, \mathrm{d}t, \tag{40}$$

which formally resembles Eq. (25).

Finally, let us discuss gradient computation in the scenario of a time-dependent L. In this setup, L additionally depends on some parameter $\theta \in \mathbb{R}$, and our goal is computing the gradient of C with respect to θ. We express the Lindblad equation (21) as

$$\frac{\mathrm{d}}{\mathrm{d}t} r(t) = f(r(t), t, \theta) \tag{41}$$

with $f(r,t,\theta) = \mathsf{L}(t,\theta)\, r$. Then, based on the derivation in [15], one obtains the following generalization of (39):

$$\overline{\theta} = \int_0^{t_\text{f}} \overline{r}(t)^T \frac{\partial f(r(t),t,\theta)}{\partial\theta}\, \mathrm{d}t = \int_0^{t_\text{f}} \overline{r}(t)^T \frac{\partial \mathsf{L}(t,\theta)}{\partial\theta} r(t)\, \mathrm{d}t. \tag{42}$$

Expressed in terms of $\mathcal{L}$, this equation reads

$$\overline{\theta} = \int_0^{t_\text{f}} \mathrm{Tr}\Big[\overline{\rho}(t) \frac{\partial}{\partial\theta} \mathcal{L}(\rho(t),t,\theta) \Big] \mathrm{d}t, \tag{43}$$

where we follow the convention that $\overline{r}(t)$ is related to $\overline{\rho}(t)$ as in (3) but without the 2^{-n} prefactor. Note that $\overline{\rho}(t)$ could be interpreted as measurement operator in Eq. (43)

VI. VQT APPLICATION EXAMPLE

In order to demonstrate the practical feasibility of our framework, we implement the variational quantum thermalizer (VQT) algorithm for a quantum Hamiltonian-based model (QHBM) [10] using the Qaintum software library. The code for the present example is available at [18]. The goal is to approximate a target thermal state

$$\sigma_\beta = \frac{1}{\mathcal{Z}_\beta}\, \mathrm{e}^{-\beta H}, \quad \mathcal{Z}_\beta = \mathrm{Tr}\big[\mathrm{e}^{-\beta H}\big], \tag{44}$$

given a known Hamiltonian H and inverse temperature β.

The ansatz density matrix starts from a "latent" diagonal density matrix ρ_θ (parametrized by a real vector θ), which is then conjugated by a unitary matrix U_ϕ (represented as a quantum circuit with parameters ϕ) [10]:

$$\rho_{\theta,\phi} = U_\phi\, \rho_\theta\, U_\phi^\dagger. \tag{45}$$

As in the prior work [10], we use

$$\rho_\theta = \bigotimes_{j=1}^n \begin{pmatrix} \frac{1}{2}(1+\cos(\theta_j)) & 0 \\ 0 & \frac{1}{2}(1-\cos(\theta_j)) \end{pmatrix} \tag{46}$$

for the latent density matrix. Regarding U_ϕ, the parameterized quantum circuit is a composition of several layers. Each layer in turn consists of two types of gates: a parametrized single-qubit rotation gate on site j defined as

$$R_{\phi_j} = \mathrm{e}^{i(\phi_j^1 X_j + \phi_j^2 Y_j + \phi_j^3 Z_j)}, \tag{47}$$

and a two-qubit entanglement gate

$$E_{\eta_j} = \mathrm{e}^{i(\eta_j^1 X_j X_{j+1} + \eta_j^2 Y_j Y_{j+1} + \eta_j^3 Z_j Z_{j+1})}. \tag{48}$$

As in [10], we use a sequential arrangement of qubits in the circuit with open boundary conditions, independent of the physical model. These entanglement gates are applied in a brick wall pattern. Fig. 2 shows a single such layer. For our experiments, we use three layers.

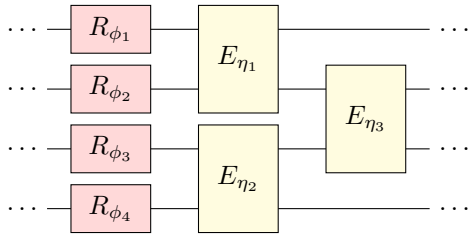


FIG. 2. A single parametrized layer of the circuit model.

The to-be minimized cost function of the optimization problem is based on the Kullback-Leibler divergence (relative entropy) [19] of the ansatz density matrix and the ground truth:

$$\begin{aligned}\mathcal{D}(\rho_{\theta,\phi} \parallel \sigma_\beta) &\equiv -\mathrm{Tr}[\rho_{\theta,\phi}\log(\rho_{\theta,\phi})] - \mathrm{Tr}[\rho_{\theta,\phi}\log(\sigma_\beta)] \\ &= -S(\rho_{\theta,\phi}) + \beta\,\mathrm{Tr}[H\rho_{\theta,\phi}] + \log(\mathcal{Z}_\beta),\end{aligned} \tag{49}$$

where S is the von Neumann entropy. Since β is fixed, the term $\log(Z_\beta)$ can be regarded as constant for the optimization with respect to $\rho_{\theta,\phi}$. One then arrives at the following cost function:

$$\mathcal{L}_{\theta,\phi} = -S(\rho_{\theta,\phi}) + \beta\,\mathrm{Tr}[H\rho_{\theta,\phi}]. \tag{50}$$

For the first experiment, we consider a Heisenberg Hamiltonian on a one-dimensional lattice:

$$H_{1\mathrm{D}} = -J\sum_{j=1}^{n-1}\vec{S}_j\cdot\vec{S}_{j+1} + \sum_{j=1}^{n}\left(gS_j^x + hS_j^z\right), \tag{51}$$

with $\vec{S} = \frac{1}{2}\vec{\sigma}$ on a 1D lattice with 4 qubits ($n = 4$). We kept the J, g and h parameters constant and examined how well the model performed when the temperature β is varied from $\beta = 0$ to $\beta = 20$ in 0.1 intervals. We utilized an AdaMax optimzer with learning rate 0.005 for faster convergence and ran the experiment 50 times with randomized initial parametric values (θ, ϕ) for each β.

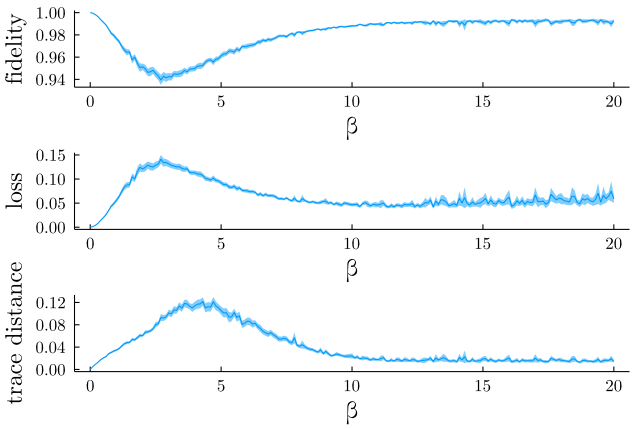


FIG. 3. Variation of (a) fidelity, (b) loss, and (c) trace distance after 500 optimization iterations over β values from $\beta = 0$ to $\beta = 20$ at 0.1 intervals for $H_{1\mathrm{D}}$ in (51) with $n = 4$, with chosen parameters $J = -1$, $g = 0.3$, $h = 0.2$. The shaded region indicates the 95% confidence interval.

The results are shown in Fig. 3. To ensure convergence, we ran each optimization for 500 iterations.

We reproduce the general behavior observed in the previous work [10] (a dip in the fidelity and spike in the loss between $\beta = 1$ and $\beta = 5$). As minor remark, in the worst case the minimum fidelity is ≈ 0.93 here, which is slightly higher than in the prior work. This may be explained by the larger number of optimization steps used here, or a differing gradient-based optimizer.

As next experiment, we apply the optimization procedure to a Heisenberg-type Hamiltonian on a two-dimensional lattice:

$$H_{2\mathrm{D}} = \sum_{\langle j,k\rangle_h} J_h\vec{S}_j\cdot\vec{S}_k + \sum_{\langle j,k\rangle_v} J_v\vec{S}_j\cdot\vec{S}_k, \tag{52}$$

where $\langle\cdot,\cdot\rangle_h$ and $\langle\cdot,\cdot\rangle_v$ indicate nearest neighbor pairs in the horizontal and vertical directions, respectively.

Fig. 4 visualizes the convergence of the numerical method with the number of optimization steps, averaged over 50 realizations with random initial parameters (θ, ϕ). One observes an unhampered, smooth convergence. The independence of the final values (after 300 iterations) of the initial random parameters indicates that the procedure is not trapped in local minima. The actual approximation metrics are shown in Fig. 5, for β between 0 and 20 in 0.1 intervals, indicating that the approximation worsens with increasing β. Note that the confidence interval remains quite small throughout the experiment; thus it is likely that the particular parametric ansatz (45) may not have sufficient expressibility for the two-dimensional case. Conversely, a more complex circuit or latent modular density matrix would be required for better results. This was suggested but not experimentally confirmed in the prior work [10].

In summary, the methods introduced in this work are implemented in the Qaintum software library, which of-

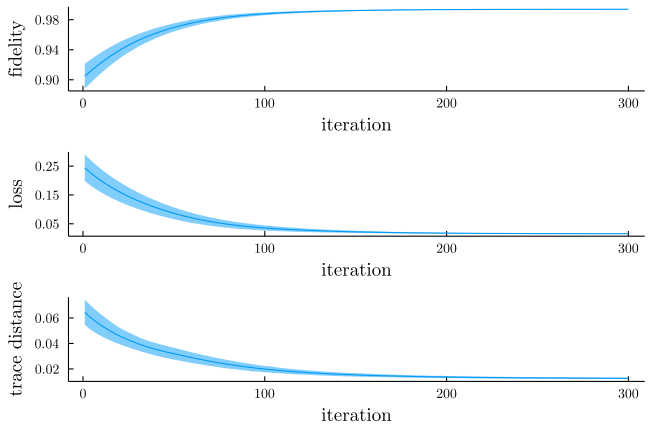


FIG. 4. Convergence of the VQT algorithm optimization procedure at $\beta = 0.5$ for the Heisenberg Hamiltonian (52) on a 2×2 lattice with parameters $J_h = 1$, $J_v = 0.6$. The shaded regions indicate 95% confidence intervals.

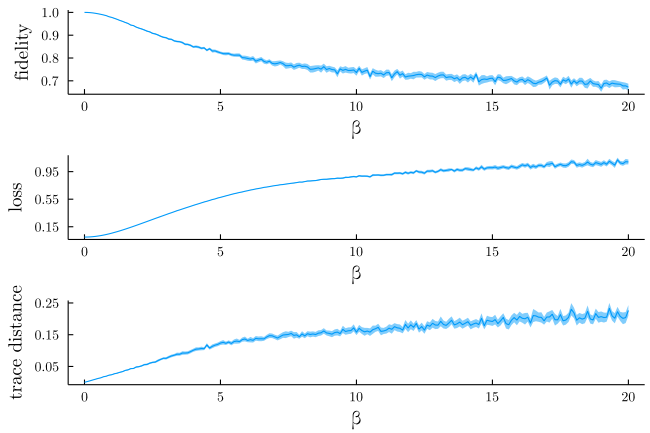


FIG. 5. Metrics for approximating the target thermal density σ_β in (44) as function of β, with 500 optimization iterations using the VQT algorithm with parameters as in Fig. 4. The shaded regions indicate 95% confidence intervals.

fers the functionality to construct a parametrized quantum circuit U_ϕ and apply it as conjugation (45), and then handles gradient computation internally to facilitate parameter optimization via the Flux [20] machine learning toolbox.

VII. CONCLUSIONS AND OUTLOOK

We have demonstrated several computational advantages of the Bloch representation, in particular in the context of variational circuit and quantum channel optimization for mixed states. Nevertheless, there are cases in which a conversion between a conventional matrix representation is still required. One scenario is the task of computing the eigenvalues of a density operator, when, for example, ensuring that it is positive semidefinite. It could be possible to adapt an implementation of, say, the QR iteration algorithm, which involves conjugations as in (4), to work directly with the Bloch vector representation, but established linear algebra software packages certainly expect a matrix as input. Another scenario for a matrix representation as starting point is a "pure state", i.e., a density operator of the form $\rho = |\psi\rangle\langle\psi|$ with $|\psi\rangle \in \mathbb{C}^{2^n}$ a statevector.

We remark that obtaining gradients as described in Sect. V is computationally more efficient than the parameter shift rule [21–23] in most cases; the latter is tailored to physical quantum computers, for which the intermediate quantum states are inaccessible. The parameter shift rule has the drawback that a circuit has to be run twice for each parameter. In our case, only a single backward pass through the circuit is necessary to obtain the gradients with respect to all gates. We have demonstrated the practical feasibility of this approach via the implementation of the VQT algorithm.

As an outlook, we want to draw the attention to tensor network methods as powerful tools for simulating density operators [24, 25]. An interesting project for future research could consist of approximating the multi-qubit Bloch vector by a real-valued matrix product state.

ACKNOWLEDGMENTS

We thank Frank Pollmann for helpful discussions, the Munich Center for Quantum Science and Technology for support, and the Leibniz Supercomputing Centre (LRZ) for providing computing resources.

Appendix A: Bloch representation of common quantum logic gates and channels

1. Single-qubit gates

Table I summarizes the Bloch representation of common single-qubit gates. Regarding the general rotation gate $R_{\vec{n}}(\theta)$, $\theta \in \mathbb{R}$ is the rotation angle, $\vec{n} \in \mathbb{R}^3$ is the unit vector specifying the rotation axis, $\vec{\sigma} = (X, Y, Z)$ the Pauli vector and $\mathrm{Rot}(\vec{n}, \theta)$ the matrix describing a classical three-dimensional rotation by angle θ. Its action on a vector $\vec{v} \in \mathbb{R}^3$ is given by Rodrigues' rotation formula:

$$\mathrm{Rot}(\vec{n},\theta)\vec{v} = \cos(\theta)\vec{v} + \sin(\theta)\,\vec{n} \times \vec{v} + (1 - \cos(\theta))(\vec{n} \cdot \vec{v})\,\vec{n}. \quad \text{(A1)}$$

See, e.g., [1] for a derivation that $R_{\vec{n}}(\theta)$ indeed translates to a classical rotation in the Bloch representation.

The projector $|1\rangle\langle 1|$ is not actually a unitary gate, but we include it here as ingredient of controlled gates.

symbol	U	U	Bloch repr. of $\mathcal{S}_U$	Bloch repr. of $\mathcal{A}_U$
X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}$	$\begin{pmatrix} & 1 & & \\ 1 & & & \\ & & 0 & \\ & & & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & & & \\ & 0 & & \\ & & & 1 \\ & & -1 & \end{pmatrix}$
Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$	$\begin{pmatrix} & 1 & & \\ 0 & & & \\ & & & 0 \\ & & 1 & \end{pmatrix}$	$\begin{pmatrix} 0 & & & \\ & & -1 & \\ & 0 & & \\ 1 & & & \end{pmatrix}$
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & 1 \end{pmatrix}$	$\begin{pmatrix} & & 1 & \\ & 0 & & \\ 0 & & & \\ & & & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & & & \\ & & 1 & \\ & -1 & & \\ & & & 0 \end{pmatrix}$
H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & & & 1 \\ & & -1 & \\ & 1 & & \end{pmatrix}$	$\begin{pmatrix} & \frac{1}{\sqrt{2}} & & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & & & \\ & & 0 & \\ \frac{1}{\sqrt{2}} & & & \end{pmatrix}$	$\begin{pmatrix} 0 & & & \\ & & \frac{1}{\sqrt{2}} & \\ & -\frac{1}{\sqrt{2}} & & \frac{1}{\sqrt{2}} \\ & & -\frac{1}{\sqrt{2}} & \end{pmatrix}$
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & & -1 & \\ & 1 & & \\ & & & 1 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} 1 & & & 1 \\ & 1 & -1 & \\ & 1 & 1 & \\ 1 & & & 1 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} -1 & & & 1 \\ & -1 & 1 & \\ & -1 & -1 & \\ 1 & & & -1 \end{pmatrix}$
phase shift	$\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\varphi} \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & \cos(\varphi) & -\sin(\varphi) & \\ & \sin(\varphi) & \cos(\varphi) & \\ & & & 1 \end{pmatrix}$	$\begin{pmatrix} \mathfrak{c}^2 & & & \mathfrak{s}^2 \\ & \mathfrak{c}^2 & -\mathfrak{cs} & \\ & \mathfrak{cs} & \mathfrak{c}^2 & \\ \mathfrak{s}^2 & & & \mathfrak{c}^2 \end{pmatrix}$	$\begin{pmatrix} -\mathfrak{cs} & & & \mathfrak{cs} \\ & -\mathfrak{cs} & \mathfrak{s}^2 & \\ & -\mathfrak{s}^2 & -\mathfrak{cs} & \\ \mathfrak{cs} & & & -\mathfrak{cs} \end{pmatrix}$
$R_{\mathrm{x}}(\theta)$	$\mathrm{e}^{-i\theta X/2}$	$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \cos(\theta) & -\sin(\theta) \\ & & \sin(\theta) & \cos(\theta) \end{pmatrix}$	$\begin{pmatrix} \mathfrak{c} & & & \\ & \mathfrak{c} & & \\ & & \mathfrak{c} & -\mathfrak{s} \\ & & \mathfrak{s} & \mathfrak{c} \end{pmatrix}$	$\begin{pmatrix} & \mathfrak{s} & & \\ \mathfrak{s} & & & \\ & & 0 & \\ & & & 0 \end{pmatrix}$
$R_{\mathrm{y}}(\theta)$	$\mathrm{e}^{-i\theta Y/2}$	$\begin{pmatrix} 1 & & & \\ & \cos(\theta) & & \sin(\theta) \\ & & 1 & \\ & -\sin(\theta) & & \cos(\theta) \end{pmatrix}$	$\begin{pmatrix} \mathfrak{c} & & & \\ & \mathfrak{c} & & \mathfrak{s} \\ & & \mathfrak{c} & \\ & -\mathfrak{s} & & \mathfrak{c} \end{pmatrix}$	$\begin{pmatrix} & & \mathfrak{s} & \\ & 0 & & \\ \mathfrak{s} & & & \\ & & & 0 \end{pmatrix}$
$R_{\mathrm{z}}(\theta)$	$\mathrm{e}^{-i\theta Z/2}$	$\begin{pmatrix} 1 & & & \\ & \cos(\theta) & -\sin(\theta) & \\ & \sin(\theta) & \cos(\theta) & \\ & & & 1 \end{pmatrix}$	$\begin{pmatrix} \mathfrak{c} & & & \\ & \mathfrak{c} & -\mathfrak{s} & \\ & \mathfrak{s} & \mathfrak{c} & \\ & & & \mathfrak{c} \end{pmatrix}$	$\begin{pmatrix} & & & \mathfrak{s} \\ & 0 & & \\ & & 0 & \\ \mathfrak{s} & & & \end{pmatrix}$
$R_{\vec{n}}(\theta)$	$\mathrm{e}^{-i\theta(\vec{n}\cdot\vec{\sigma})/2}$	$\begin{pmatrix} 1 & & \\ & & \\ & \mathrm{Rot}(\vec{n},\theta) & \end{pmatrix}$	$\begin{pmatrix} \mathfrak{c} & & & \\ & \mathfrak{c} & -\mathfrak{s}n_3 & \mathfrak{s}n_2 \\ & \mathfrak{s}n_3 & \mathfrak{c} & -\mathfrak{s}n_1 \\ & -\mathfrak{s}n_2 & \mathfrak{s}n_1 & \mathfrak{c} \end{pmatrix}$	$\begin{pmatrix} & \mathfrak{s}n_1 & \mathfrak{s}n_2 & \mathfrak{s}n_3 \\ \mathfrak{s}n_1 & & & \\ \mathfrak{s}n_2 & & & \\ \mathfrak{s}n_3 & & & \end{pmatrix}$
$\lvert 1\rangle\langle 1\rvert$	$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} 1 & & & -1 \\ & 0 & & \\ & & 0 & \\ -1 & & & 1 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} 1 & & & -1 \\ & 1 & & \\ & & 1 & \\ -1 & & & 1 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} 0 & & & \\ & & -1 & \\ & 1 & & \\ & & & 0 \end{pmatrix}$

TABLE I. Bloch representation of common single-qubit quantum gates and associated operators in (11), using shorthands $\mathfrak{c} = \cos(\theta/2)$ and $\mathfrak{s} = \sin(\theta/2)$ for the rotation gates, and likewise $\mathfrak{c} = \cos(\varphi/2)$ and $\mathfrak{s} = \sin(\varphi/2)$ for the phase shift gate.

2. Two-qubit gates

Table II shows the Bloch representation of selected two-qubit gates. For conciseness of notation, we use braket notation as $|jk\rangle = e_j \otimes e_k$ for $j,k = 0,\ldots,3$, with e_j the j-th unit vector of length 4.

3. Single-qubit quantum channels

Table III summarizes the Bloch representation of several single-qubit quantum channels [1]; the parameters p,γ,λ are from the interval $[0,1]$, and can be interpreted as probabilities.

Appendix B: Expansion of $\mathcal{S}_{F\otimes G}$ and $\mathcal{A}_{F\otimes G}$ and generalization to multiple tensor products

We first verify the relations in Eqs. (15). Given complex matrices $F \in \mathbb{C}^{m\times m}$ and $G \in \mathbb{C}^{n\times n}$, note that the linear operators $\mathcal{S}_{F\otimes G}$ and $\mathcal{A}_{F\otimes G}$ act on Hermitian matrices ρ of dimension $mn \times mn$. For any such ρ, one calculates

$$\begin{aligned}
&(\mathcal{S}_F \otimes \mathcal{S}_G)(\rho) - (\mathcal{A}_F \otimes \mathcal{A}_G)(\rho) \\
&= \frac{1}{2}\Big(F\otimes I \cdot \frac{1}{2}(I\otimes G\cdot\rho + \rho\cdot I\otimes G^\dagger) \\
&\quad + \frac{1}{2}(I\otimes G\cdot\rho + \rho\cdot I\otimes G^\dagger)\cdot F^\dagger\otimes I\Big) \\
&- \frac{i}{2}\Big(F\otimes I \cdot \frac{i}{2}(I\otimes G\cdot\rho - \rho\cdot I\otimes G^\dagger) \\
&\quad - \frac{i}{2}(I\otimes G\cdot\rho - \rho\cdot I\otimes G^\dagger)\cdot F^\dagger\otimes I\Big) \\
&= \frac{1}{2}(F\otimes G\cdot\rho + \rho\cdot F^\dagger\otimes G^\dagger) = \mathcal{S}_{F\otimes G}(\rho)
\end{aligned} \tag{B1}$$

and

$$\begin{aligned}
&(\mathcal{S}_F \otimes \mathcal{A}_G)(\rho) + (\mathcal{A}_F \otimes \mathcal{S}_G)(\rho) \\
&= \frac{1}{2}\Big(F\otimes I \cdot \frac{i}{2}(I\otimes G\cdot\rho - \rho\cdot I\otimes G^\dagger) \\
&\quad + \frac{i}{2}(I\otimes G\cdot\rho - \rho\cdot I\otimes G^\dagger)\cdot F^\dagger\otimes I\Big) \\
&+ \frac{i}{2}\Big(F\otimes I \cdot \frac{1}{2}(I\otimes G\cdot\rho + \rho\cdot I\otimes G^\dagger) \\
&\quad - \frac{1}{2}(I\otimes G\cdot\rho + \rho\cdot I\otimes G^\dagger)\cdot F^\dagger\otimes I\Big) \\
&= \frac{i}{2}(F\otimes G\cdot\rho - \rho\cdot F^\dagger\otimes G^\dagger) = \mathcal{A}_{F\otimes G}(\rho).
\end{aligned} \tag{B2}$$

Recursive application of (15) facilitates a generalization to multiple tensor products, i.e., an expansion of $\mathcal{S}_{G_{n-1}\otimes\cdots\otimes G_0}$ and $\mathcal{A}_{G_{n-1}\otimes\cdots\otimes G_0}$ for complex matrices $G_0,\ldots,G_{n-1}$. To arrive at a concise expression, observe that (15) formally resembles the product of two complex numbers, with $\mathcal{S}$ and $\mathcal{A}$ playing the roles of the real and imaginary parts, respectively. Following this analogy, let $z_j = x_j + iy_j$ with $x_j, y_j \in \mathbb{R}$ for $j = 0,\ldots,n-1$. Then the product of the z_j's in terms of real and imaginary parts is

$$z_{n-1}\cdots z_1 z_0 = \begin{pmatrix} x_{n-1} & -y_{n-1} \\ y_{n-1} & x_{n-1}\end{pmatrix} \cdots \begin{pmatrix} x_1 & -y_1 \\ y_1 & x_1\end{pmatrix}\begin{pmatrix} x_0 \\ y_0\end{pmatrix} \tag{B3}$$

when identifying $\mathbb{C}\simeq\mathbb{R}^2$. Thus likewise

$$\begin{aligned}
&\begin{pmatrix} \mathcal{S}_{G_{n-1}\otimes\cdots\otimes G_0} \\ \mathcal{A}_{G_{n-1}\otimes\cdots\otimes G_0}\end{pmatrix} \\
&= \begin{pmatrix} \mathcal{S}_{G_{n-1}} & -\mathcal{A}_{G_{n-1}} \\ \mathcal{A}_{G_{n-1}} & \mathcal{S}_{G_{n-1}}\end{pmatrix} \cdots \begin{pmatrix} \mathcal{S}_{G_1} & -\mathcal{A}_{G_1} \\ \mathcal{A}_{G_1} & \mathcal{S}_{G_1}\end{pmatrix}\begin{pmatrix} \mathcal{S}_{G_0} \\ \mathcal{A}_{G_0}\end{pmatrix},
\end{aligned} \tag{B4}$$

with $\mathcal{S}_{G_j}$ and $\mathcal{A}_{G_j}$ understood to act on the j-th qubit. We remark that (B4) is in fact a matrix product operator representation with virtual bond dimension 2.

Appendix C: Benchmarking

We compare the runtime of our Bloch representation for applying unitary quantum gates with a conventional conjugation of the density matrix, see Eq. (4). Due to the highly sparse nature of the gates depicted in Appendix A, we are able to implement Eq. (5) in a matrix-free manner using a single loop over the stored Bloch vector, which potentially offers a $\mathcal{O}(1)$ speedup. (The asymptotic computational complexity is linear in the number of Bloch vector entries for both versions.) A comparison of the application of single and two-qubit gates using our methodology, and an optimized in-place multiplication code using Julia's SparseArrays module (sparse CSC format for the gates and dense format for the density matrix) is shown in Fig. 6. Indeed one observes a constant speedup facilitated by the Bloch representation. We note some possible cache optimization issues for the case of three qubits, which could be remedied by proper chunking of the stored Bloch vectors. Single-qubit gates exhibit the largest runtime advantage, while the smaller speedup for controlled gates is likely due to the more involved expansion of such gates, see Sect. III B.

The benchmarking was performed on the cloud computing nodes offered by the Leibniz Supercomputing Centre; specifically for this study single-threaded on a Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz.

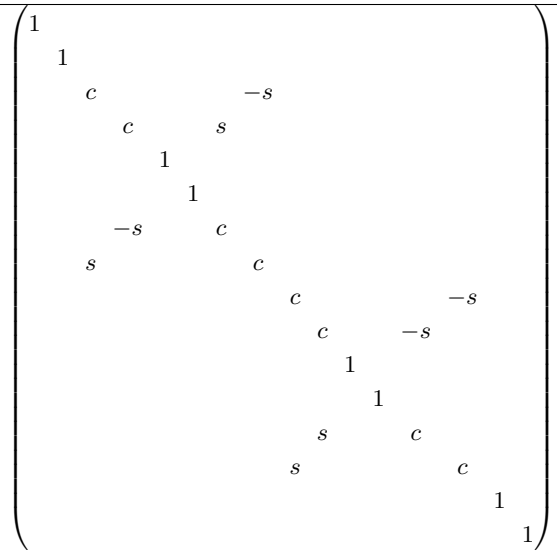
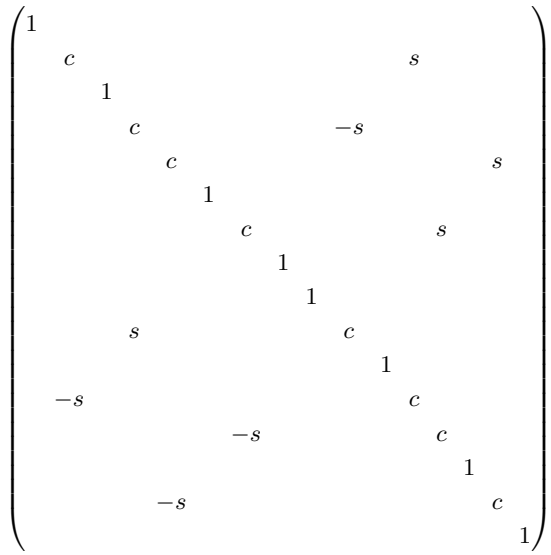
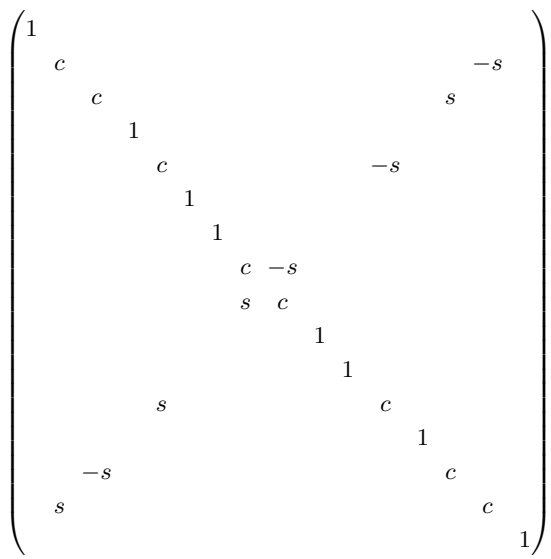
symbol	U	U	
$R_{\mathrm{xx}}(\theta)$	$\mathrm{e}^{-i\theta X\otimes X/2}$		$= I_{16}$ $-(1-c)\big(\lvert 02\rangle\langle 02\rvert+\lvert 20\rangle\langle 20\rvert+\lvert 03\rangle\langle 03\rvert+\lvert 30\rangle\langle 30\rvert+\lvert 12\rangle\langle 12\rvert+\lvert 21\rangle\langle 21\rvert+\lvert 13\rangle\langle 13\rvert+\lvert 31\rangle\langle 31\rvert\big)$ $+ s\big(\lvert 03\rangle\langle 12\rvert-\lvert 12\rangle\langle 03\rvert+\lvert 30\rangle\langle 21\rvert-\lvert 21\rangle\langle 30\rvert+\lvert 13\rangle\langle 02\rvert-\lvert 02\rangle\langle 13\rvert+\lvert 31\rangle\langle 20\rvert-\lvert 20\rangle\langle 31\rvert\big)$
$R_{\mathrm{yy}}(\theta)$	$\mathrm{e}^{-i\theta Y\otimes Y/2}$		$= I_{16}$ $-(1-c)\big(\lvert 01\rangle\langle 01\rvert+\lvert 10\rangle\langle 10\rvert+\lvert 03\rangle\langle 03\rvert+\lvert 30\rangle\langle 30\rvert+\lvert 12\rangle\langle 12\rvert+\lvert 21\rangle\langle 21\rvert+\lvert 23\rangle\langle 23\rvert+\lvert 32\rangle\langle 32\rvert\big)$ $+ s\big(\lvert 01\rangle\langle 23\rvert-\lvert 23\rangle\langle 01\rvert+\lvert 10\rangle\langle 32\rvert-\lvert 32\rangle\langle 10\rvert+\lvert 12\rangle\langle 30\rvert-\lvert 30\rangle\langle 12\rvert+\lvert 21\rangle\langle 03\rvert-\lvert 03\rangle\langle 21\rvert\big)$
$R_{\mathrm{zz}}(\theta)$	$\mathrm{e}^{-i\theta Z\otimes Z/2}$		$= I_{16}$ $-(1-c)\big(\lvert 01\rangle\langle 01\rvert+\lvert 10\rangle\langle 10\rvert+\lvert 02\rangle\langle 02\rvert+\lvert 20\rangle\langle 20\rvert+\lvert 13\rangle\langle 13\rvert+\lvert 31\rangle\langle 31\rvert+\lvert 23\rangle\langle 23\rvert+\lvert 32\rangle\langle 32\rvert\big)$ $+ s\big(\lvert 02\rangle\langle 31\rvert-\lvert 31\rangle\langle 02\rvert+\lvert 20\rangle\langle 13\rvert-\lvert 13\rangle\langle 20\rvert+\lvert 23\rangle\langle 10\rvert-\lvert 10\rangle\langle 23\rvert+\lvert 32\rangle\langle 01\rvert-\lvert 01\rangle\langle 32\rvert\big)$

TABLE II. Bloch representation of selected two-qubit quantum gates, using the shorthand notations $c = \cos(\theta)$ and $s = \sin(\theta)$.

channel	Kraus operators	Bloch repr. (19)
bit flip	$E_0 = \sqrt{p} I_2,$ $E_1 = \sqrt{1-p} X$	$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 2p-1 & \\ & & & 2p-1 \end{pmatrix}$
phase flip	$E_0 = \sqrt{p} I_2,$ $E_1 = \sqrt{1-p} Z$	$\begin{pmatrix} 1 & & & \\ & 2p-1 & & \\ & & 2p-1 & \\ & & & 1 \end{pmatrix}$
depolarizing channel	$E_0 = \sqrt{1-3p/4} I_2,$ $E_1 = \sqrt{p} X/2,$ $E_2 = \sqrt{p} Y/2,$ $E_3 = \sqrt{p} Z/2$	$\begin{pmatrix} 1 & & & \\ & 1-p & & \\ & & 1-p & \\ & & & 1-p \end{pmatrix}$
amplitude damping	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix},$ $E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & \sqrt{1-\gamma} & & \\ & & \sqrt{1-\gamma} & \\ \gamma & & & 1-\gamma \end{pmatrix}$
phase damping	$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{pmatrix},$ $E_1 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{pmatrix}$	$\begin{pmatrix} 1 & & & \\ & \sqrt{1-\lambda} & & \\ & & \sqrt{1-\lambda} & \\ & & & 1 \end{pmatrix}$

TABLE III. Bloch representation of several single-qubit quantum channels describing noise processes.

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).

[2] H. Abraham and et al., Qiskit: An open-source framework for quantum computing (2019).

[3] Cirq Developers, Cirq (2021).

[4] A. Li, O. Subasi, X. Yang, and S. Krishnamoorthy, Density matrix quantum circuit simulation via the BSP machine on modern GPU clusters, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2020).

[5] G. Kimura, The Bloch vector for N-level systems, Phys. Lett. A **314**, 339 (2003).

[6] R. A. Bertlmann and P. Krammer, Bloch vectors for qudits, J. Phys. A **41**, 235303 (2008).

[7] S. Jevtic, M. Pusey, D. Jennings, and T. Rudolph, Quantum steering ellipsoids, Phys. Rev. Lett. **113**, 020402 (2014).

[8] O. Gamel, Entangled Bloch spheres: Bloch matrix and two-qubit state space, Phys. Rev. A **93**, 062320 (2016).

[9] Qaintum (github.com/Qaintum) (2021).

[10] G. Verdon, J. Marks, S. Nanda, S. Leichenauer, and J. Hidary, Quantum Hamiltonian-based models and the variational quantum thermalizer algorithm, arXiv:1910.02071 (2019), arXiv:1910.02071 [quant-ph].

[11] V. Gorini, A. Kossakowski, and E. C. G. Sudarshan, Completely positive dynamical semigroups of N-level systems, J. Math. Phys. **17**, 821 (1976).

[12] G. Lindblad, On the generators of quantum dynamical semigroups, Commun. Math. Phys. **48**, 119 (1976).

[13] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, Yao.jl: Extensible, efficient framework for quantum algorithm design, Quantum **4**, 341 (2020).

[14] W. E, A proposal on machine learning via dynamical systems, Commun. Math. Stat. **5**, 1 (2017).

[15] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, Neural ordinary differential equations, in *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018) pp. 6572–6583.

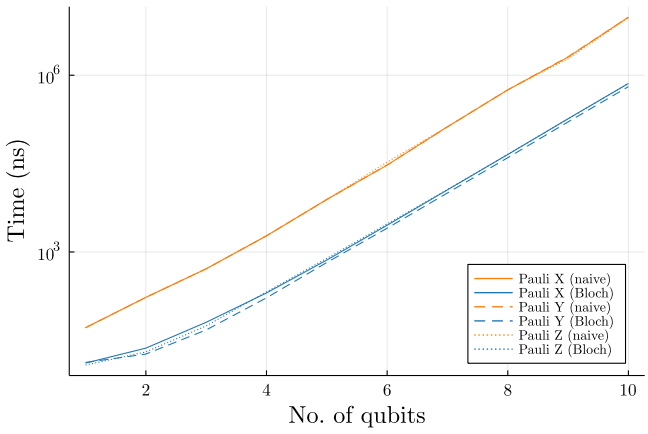
[16] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The mathematical theory of optimal processes* (Wiley, New York, 1962).

[17] R. M. Wilcox, Exponential operators and parameter differentiation in quantum physics, J. Math. Phys. **8**, 962 (1967).

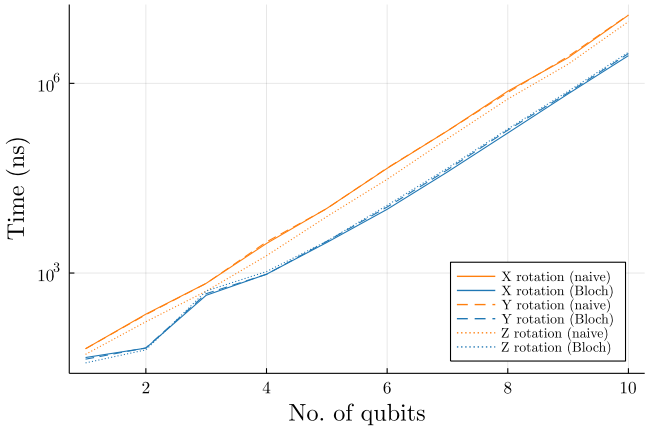
[18] Q. Huang and C. B. Mendl, https://github.com/cmendl/density-matrix-bloch-qcircuit (2021).

[19] S. Kullback and R. A. Leibler, On information and sufficiency, Ann. Math. Statist. **22**, 79 (1951).

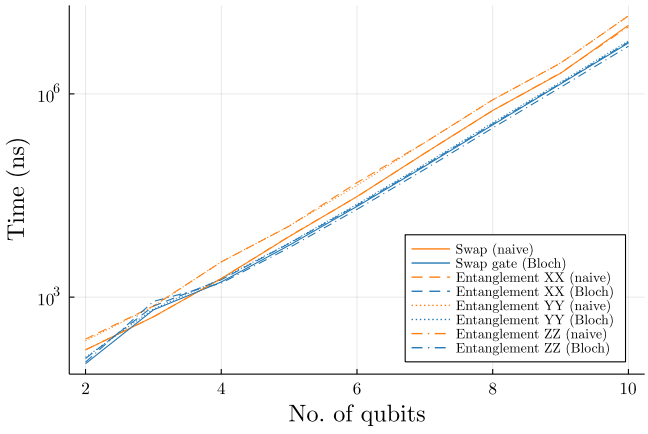
[20] Flux (fluxml.ai).



(a) Pauli gates



(b) rotation operators



(c) two-qubit gates

FIG. 6. Benchmark comparison of our Bloch vector methodology (blue) with a conventional implementation by matrix conjugations (orange), for the task of applying typical single- and two-qubit gates.

[21] J. Li, X. Yang, X. Peng, and C.-P. Sun, Hybrid quantum-classical approach to quantum optimal control, Phys. Rev. Lett. **118**, 150503 (2017).

[22] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[23] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[24] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, Matrix product density operators: Simulation of finite-temperature and dissipative systems, Phys. Rev. Lett. **93**, 207204 (2004).

[25] J. Hauschild, E. Leviatan, J. H. Bardarson, E. Altman, M. P. Zaletel, and F. Pollmann, Finding purifications with minimal entanglement, Phys. Rev. B **98**, 235163 (2018).