Exponentiation of parametric Hamiltonians via unitary interpolation

Michael Schilling [1,2] Francesco Preti [1,2] Matthias M. Müller [1] Tommaso Calarco [1,2,3] and Felix Motzoi [1]
[1]*Peter Grünberg Institute–Quantum Control (PGI-8), Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany*
[2]*Institute for Theoretical Physics, University of Cologne, Zülpicher Straße 77, 50937 Cologne, Germany*
[3]*Dipartimento di Fisica e Astronomia, Università di Bologna, 40127 Bologna, Italy*

(Received 5 June 2024; accepted 24 November 2024; published 16 December 2024)

The effort to generate matrix exponentials and associated differentials, required to determine the time evolution of quantum systems, frequently constrains the evaluation of problems in quantum control theory, variational circuit compilation, or Monte Carlo sampling. We introduce two ideas for the time-efficient approximation of matrix exponentials of linear multiparametric Hamiltonians. We modify the Suzuki-Trotter product formula from an approximation to an interpolation scheme to improve both accuracy and walltime. This allows us to achieve high fidelities within a single interpolation step, which can be computed directly from cached matrices. Furthermore, we define the interpolation on a grid of system parameters, and show that the interpolation infidelity converges with fourth-order accuracy in the number of interpolation bins.

DOI: 10.1103/PhysRevResearch.6.043278

I. INTRODUCTION

The repeated construction of unitaries of linear parametric Hamiltonians is a cornerstone task in various research domains of quantum mechanics. The ability to accurately and efficiently construct these unitaries enables us to predict and control the dynamics of quantum systems. This is vital for both theoretical insights and practical applications ranging from quantum simulation [1,2] to quantum optimal control [3] and quantum circuit compilation [4–6]. Due to the curse of dimensionality caused by the exponential growth of finite-dimensional representations of Hilbert spaces as a function of the problem size, even the generation of a few unitaries can be computationally taxing. Yet, the aforementioned applications require the construction of many such unitaries.

A common challenge is calculating the time evolution operator of time-dependent Hamiltonians. To address this, the typical strategy involves approximating these Hamiltonians using products of unitaries over sufficiently small time steps. Standard approaches to numerically compute these time steps include the Trotter-product formula [7], Dyson series [8], Magnus expansion [9,10], Fer expansion [11], and Runge-Kutta methods [12,13]. The case of quantum state propagation, is often tackled using Krylov subspace methods [14].

We consider Hamiltonians with a few linear parameters. Simulating such quantum systems classically requires the repeated construction of matrix exponentials (or their action onto states). Such problems commonly occur within the context of quantum optimal control theory [3], where we shape system parameters in time to achieve the high fidelity operations required in high-precision quantum technologies. Methods such as gradient ascent pulse engineering (GRAPE) [15–17], Krotov [18], dynamic optimization (DYNAMO) [19], chopped random-basis (CRAB) [20–22], and single optimization with multiple application (SOMA) [23] for generalized system parameter-dependent pulses, require the repeated construction of matrix exponentials to compute and optimize the time-propagation of quantum systems. Multiparameter Hamiltonians also find application in stochastic evolution [24] and other Monte Carlo methods [25,26].

In this work we introduce a matrix-product approach, which we call unitary interpolation (UI), which interpolates between unitaries to approximate matrix exponentials to arbitrary precision. We compare this new approach to existing schemes, namely exact matrix exponentiation via Hermitian eigenvalue decomposition, Trotterization, and in the case of state propagation, Krylov subspace method that uses Lanczos iterations [27]. Standard approaches to compute the time-evolution of unitaries (vectors) have a time complexity of $\mathcal{O}(d^3)$[1] ($\mathcal{O}(d^2)$), where d is the matrix dimension. Furthermore, the time complexity of such methods has a large constant prefactor compared to matrix-matrix (matrix-vector) multiplication. As such, matrix exponentiation typically constitutes a major bottleneck in the classical simulation of quantum systems. We show that our approach can reduce the computational cost of repeated matrix exponentiation compared to Trotter-based approaches and eigenvalue decomposition. Our method allows for the inclusion of multiple system parameters at the cost of requiring more precomputations. We also consider the time evolution of quantum states,

[1]This can be reduced to $\mathcal{O}(d^{2.8})$ for matrix multiplication using the Strassen algorithm, which is not often used in practice as its advantages are limited to large matrices $d > 1000$.

and find further speedups with respect to a Krylov subspace method that uses Lanczos iterations.

The paper is structured as follows: in Sec. II we define the unitary evolution of a time-dependent Hamiltonian with continuous linear parameters. In Sec. III the interpolation between two arbitrary unitaries via an interpolation operator is introduced. This method is applied to problems with only one time-dependent parameter in Sec. IV. We then generalize the interpolation scheme to problems with multiple time-dependent parameters in Sec. V. Furthermore, in Sec. VI we discuss the special case of state evolution using UI and in Sec. VII we discuss best practices for caching.

II. UNITARIES OF LINEAR PARAMETRIC HAMILTONIANS

We consider the problem of computing unitaries $\hat{U}(\mathbf{c}) \in \mathrm{U}(d)$ generated by linear parametric d-dimensional Hamiltonians $\hat{H}(\mathbf{c})$,

$$\hat{U}(\mathbf{c}) = \exp(-i\hat{H}(\mathbf{c})) \quad \text{with} \quad \hat{H}(\mathbf{c}) = \hat{H}_0 + \mathbf{c}^T\hat{\mathbf{H}}, \tag{1}$$

where $\mathbf{c} = (c_1, \cdots, c_n)^T$ is a vector of n real amplitude-bound parameters $c_p \in [c_{p,\min}, c_{p,\max}]$, $\hat{H}_0$ a d-dimensional drift Hamiltonian, and $\hat{\mathbf{H}} = (\hat{H}_1, \cdots, \hat{H}_n)^T$ a n-dimensional vector of d-dimensional parameter Hamiltonians. To keep the presentation general, we removed references to time in this definition. The absorption of time into the Hamiltonian is considered in the discussion of the special case in Sec. II A.

Let us first consider two types of problems in which these kinds of Hamiltonians are commonly encountered.

A. Time evolution via time-slicing

The time evolution $\hat{U}(0, T)$ of quantum systems with time-dependent Hamiltonians,

$$\tilde{H}(t) = \tilde{H}_0 + \sum_{p=1}^{n} \tilde{c}_p(t)\tilde{H}_p, \tag{2}$$

is represented by a product of smaller time steps,

$$\hat{U}(0, T) = \prod_{j=0}^{m-1} \hat{U}(t_j, t_{j+1}), \tag{3}$$

with $t_j = j\Delta t$ and $\Delta t = \frac{T}{m}$. Each of the time steps $[t_j, t_{j+1})$ is approximated by a unitary $\hat{U}(t_j, t_{j+1}) \approx \tilde{U}(\mathbf{c}_j)$ generated from a time-independent linear parametric Hamiltonian $\hat{H}(\mathbf{c}_j)$ of the form given in Eq. (1). In particular, the time-independent "surrogate" Hamiltonian $\hat{H}(\mathbf{c}_j)$ can be constructed from a truncated Magnus expansion [9,10]. The first $\tilde{n}$ terms of the expansion are given by the average parameter values in the interval

$$c_p = \frac{1}{\Delta t}\int_{t_j}^{t_{j+1}} \tilde{c}_p(t')dt' \tag{4}$$

and the normalized Hamiltonian elements $\hat{H}_p = \tilde{H}_p\Delta t$. Higher-order terms are given by the nested commutators of the original Hamiltonian elements $\hat{H}_p$ and higher-order integrals of the parameters $\tilde{c}_p(t)$. An instructive introduction can be found in Ref. [28].

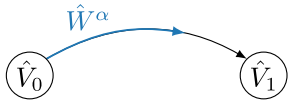

FIG. 1. Interpolation between unitaries $\hat{V}_0$ and $\hat{V}_1$ via the interpolation operation $\hat{W}^\alpha$ with $\alpha \in [0, 1]$ (here $\alpha = 3/4$), which transforms $\hat{V}_0$ into $\hat{V}_1$ for $\alpha = 1$.

B. Computation via variational quantum circuits

The classical simulation of variational quantum circuits requires the sequential multiplication of a given number l of unitaries

$$\hat{U}(\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(l)}) = \prod_{i=0}^{l-1} \hat{U}^{(i)}(\boldsymbol{c}^{(i)}), \tag{5}$$

where each $\hat{U}^{(i)}$ is a quantum gate with parameters $\boldsymbol{c}^{(i)}$ acting on one or more qubits. When dealing with n-qubit collective gates, the computation of their unitary matrix, which needs to be evaluated at multiple parameter values—e.g., in the context of variational quantum algorithms [29]—is usually computationally expensive, unless an analytical decomposition is available [23]. As parametric quantum gates can be represented in the form of Eq. (1), they can benefit from improved techniques to efficiently compute such unitaries $\hat{U}^{(i)}(\mathbf{c}^{(i)})$, and the (analytical) gradients with respect to the parameters $\frac{\mathrm{d}\hat{U}^{(i)}(\mathbf{c}^{(i)})}{\mathrm{d}c_p}$.

To accelerate such computations, we develop a technique to interpolate between unitaries, and to create a cache of matrices from which we can construct such interpolations efficiently. We assess the accuracy of the approximation via the average gate infidelity $I_{\mathrm{avg}} = \frac{d}{d+1} - \frac{|\mathrm{Tr}(\hat{U}_{\mathrm{approx}}\hat{U}^\dagger_{\mathrm{exact}})|^2}{d(d+1)}$ between $\hat{U}_{\mathrm{exact}}$—the exact unitary—and $\hat{U}_{\mathrm{approx}}$—the approximated unitary; see also Appendix C. We then perform an analogous assessment for quantum states.

III. INTERPOLATION BETWEEN TWO UNITARIES

Any two unitary operators $\hat{V}_0, \hat{V}_1 \in \mathbb{C}^{d\times d}$ can be written in terms of generators $\hat{A}$ and $\hat{B}$ as $\hat{V}_0 = \exp(-i\hat{A})$ and $\hat{V}_1 = \exp(-i(\hat{A} + \hat{B}))$, where $\hat{B}$ is the difference generator. We wish to efficiently construct operators of the form $\exp(-i(\hat{A} + \alpha\hat{B}))$, where $\hat{A}$ and $\hat{B}$ are Hermitian operators and $\alpha \in [0, 1]$.

To this end we can construct an interpolation between the unitaries, from the following product formula of the unitaries:

$$\hat{V}(\alpha) = \underbrace{(\hat{V}_1\hat{V}_0^\dagger)^\alpha}_{=\hat{W}^\alpha}\hat{V}_0. \tag{6}$$

We call $\hat{W} = (\hat{V}_1\hat{V}_0^\dagger)$ the interpolation operator and $\alpha \in (0, 1)$ the interpolation parameter; see Fig. 1. We find $\hat{V}(\alpha)$ to be an *interpolation*, as it reproduces the original operators on the edges of the open interpolation interval:

$$\hat{V}(0) = \hat{V}_0 \quad \text{with} \quad \hat{V}(1) = \hat{V}_1. \tag{7}$$

The interpolation $\hat{V}(\alpha)$ is furthermore a *unitary* operator, as any product of unitaries is itself unitary, hence the name UI.

A. BCH expansion of the interpolation

The first-order BCH expansion [30] is constructed from the sum of the generators (using a sign change for the Hermitian conjugate), revealing

$$\hat{V}(\alpha) = \exp(-i(\hat{A} + \alpha\hat{B}) + \cdots). \tag{8}$$

To first order, this corresponds to the desired operator $\hat{V}(\alpha) = \exp(-i(\hat{A} + \alpha\hat{B}))$. Deviations from the desired form are given by higher-order terms in the BCH expansion. We can investigate these terms by studying the properties of the interpolation operator $\hat{W}$.

The BCH expansion of the interpolation operator $\hat{W}$ exponentiated with the interpolation parameter α is given by

$$\begin{aligned}\hat{W}^\alpha &= (\hat{V}_1\hat{V}_0^\dagger)^\alpha = (\exp(-i(\hat{A}+\hat{B}))\exp(i\hat{A}))^\alpha \\ &= \exp\left(\alpha(-i\hat{B} - \frac{1}{2}[\hat{A},\hat{B}] + \cdots)\right) \\ &= \exp\left(\alpha\sum_i \hat{C}_i\right),\end{aligned} \tag{9}$$

where we generically denoted the nested commutators of $\hat{A}$ and $\hat{B}$ as $\hat{C}_i$, e.g., $\hat{C}_1 = -i\hat{B}$, $\hat{C}_2 = -\frac{1}{2}[\hat{A},\hat{B}]$, $\hat{C}_3 = \frac{i}{12}[\hat{B},[\hat{A},\hat{B}]]$, and so on. The nested commutators $\hat{C}_i$ are at least linear in $\hat{B}$, as commutators built from only one operator ($\hat{A}$) would vanish. Furthermore, every term is linear in α. Hence, for every term the order of $\hat{B}$ is at least as large as the order of α.

To study the properties of the interpolation scheme—see Eq. (6)—we perform a similar generic BCH expansion for the undesired higher-order terms,

$$\hat{V}(\alpha) = \exp\left(\overbrace{-i(\hat{A}+\alpha\hat{B})}^{\text{Desired Terms}} + \overbrace{\sum_i p_i(\alpha)\hat{D}_i}^{\text{Undersired Terms}}\right). \tag{10}$$

The undesired terms are represented by a sum of polynomials $p_i(\alpha)$ and corresponding nested commutators $\hat{D}_i$ of the generators $\hat{A}$ and $\hat{B}$, e.g., $\hat{D}_1 = [\hat{B},[\hat{A},\hat{B}]]$. The operators $\hat{D}_i$ can be constructed from nested commutators of $\hat{A}$ and the $\hat{C}_i$—see Eq. (9). There are usually multiple different nested commutators that generate the same operator $\hat{D}_i$. For example, $\hat{D}_1$ is constructed both via $[\hat{C}_1,[\hat{A},\hat{C}_1]]$ and directly from $\hat{C}_3$. The construction of $\hat{D}_1$ via $\hat{C}_3$ contains only one $\hat{C}$ term and therefore contributes with only a linear α to the polynomial $p_1(\alpha)$, whereas the construction via $[\hat{C}_1,[\hat{A},\hat{C}_1]]$ contains two $\hat{C}$ terms and contributes a quadratic term in α to $p_1(\alpha)$.

Each of the undesired terms must vanish for $\alpha = 0$ and $\alpha = 1$, hence the polynomials $p_i(\alpha)$ must be at least quadratic in α (because it has two roots at $\alpha = 0$ and $\alpha = 1$), meaning that one of the constructions of $\hat{D}_i$ must contain at least two $\hat{C}$ terms. Therefore, every term $\hat{D}_i$ is at least quadratic in $\hat{B}$. The average infidelity I_{avg} is quadratic in the undesired terms, and scales with the fourth power of the difference operator $\hat{B}$.

B. Construction from cached matrices

To repeatedly and quickly construct the interpolations given in Eq. (6), we decompose the interpolation operator $\hat{W}$ via Schur decomposition

$$\hat{W} = (\hat{V}_1\hat{V}_0^\dagger) = \hat{L}\exp(-i\hat{E})\hat{L}^\dagger \tag{11}$$

into the eigenvector matrix $\hat{L}$ and the diagonal eigenvalue matrix $\exp(-i\hat{E})$, which we cache as a vector $\mathbf{E} = \text{diag}(\hat{E})$. Furthermore, we cache the product $\hat{R} = \hat{L}^\dagger\hat{V}_0$, so that we can rewrite the interpolation as the product

$$\hat{V}(\alpha) = \hat{L}\exp(-i\hat{E}\alpha)\hat{R}. \tag{12}$$

The matrix exponential in the middle can be performed as a row-wise scaling operation on the right-side matrix $\hat{R}$, so that the relevant complexity reduces to a single matrix multiplication.

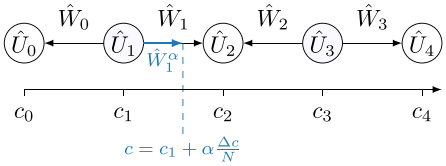

FIG. 2. The interpolation of a unitary using a single system parameter c and multiple interpolation bins. The interpolation is performed from the odd-indexed unitaries ($\hat{U}_1$, $\hat{U}_3$) toward the adjacent even indexed unitaries ($\hat{U}_0$, $\hat{U}_2$, $\hat{U}_4$), resulting in an odd-centered interpolation grid.

IV. UNITARY INTERPOLATION OF A SINGLE-SYSTEM PARAMETER

The UI method allows us to approximate unitaries of the form $\exp(-i(\hat{A} + \alpha\hat{B}))$ constructed from Hermitian operators $\hat{A}$ and $\hat{B}$. Most importantly, we find that the corrections scale with the fourth power of the difference operator $\hat{B}$, which we use to reduce the infidelity of the interpolation. We now apply the approach to a Hamiltonian with a single parameter and reduce the size of $\hat{B}$ to suppress the contributions of the undesired terms.

Let us consider the unitary $\hat{U}(c)$ generated by a Hamiltonian $H(c)$ with a single-bounded parameter $c \in [c_{\min}, c_{\max}]$,

$$\hat{U}(c) = \exp(-i\hat{H}(c)) \quad \text{with} \quad \hat{H}(c) = \hat{H}_0 + c\hat{H}_1. \tag{13}$$

To repeatedly compute unitaries $\hat{U}(c)$ for different values of c, we separate the parameter interval into a suitably[2] chosen number N of (equidistant) interpolation intervals/bins—see Fig. 2. The bins are separated by the interpolation of $N+1$ grid vertices c_i for $i \in \{0,\cdots,N\}$:

$$c_i = c_{\min} + i\frac{\Delta c}{N} \quad \text{with} \quad \Delta c = c_{\max} - c_{\min}, \tag{14}$$

for which we compute the corresponding grid unitaries,

$$\hat{U}_i = \hat{U}(c_i) = \exp(-i\hat{H}(c_i)). \tag{15}$$

To perform the UI for a system parameter c, we decompose it into a grid vertex c_i and an interpolation parameter $\alpha(c)$,

$$c = c_i + \alpha(c)\frac{\Delta c}{N}, \tag{16}$$

[2]We discuss optimal binning in Appendix F.

where $\frac{\Delta c}{N}$ is the width of a single interpolation bin. The interpolation is performed between adjacent grid unitaries $\hat{U}_i$ and $\hat{U}_{i\pm1}$, starting from the closest odd-indexed unitary $\hat{U}_i$ with $i \bmod 2 = 1$ toward the closest even-indexed unitary $\hat{U}_j$ with $j = i + \mathrm{sign}(\alpha(c))$, where $\alpha(c) = (c - c_i)\frac{N}{\Delta c} \in (-1, 1)$. The interpolation $\hat{U}_{\mathrm{UI}}(c)$ is then constructed using a bin-specific interpolation operator,

$$\hat{U}_{\mathrm{UI}}(c) = \underbrace{(\hat{U}_j \hat{U}_i^\dagger)^{|\alpha(c)|}}_{=\hat{W}_k} \hat{U}_i. \tag{17}$$

The interpolation operator $\hat{W}_k$ is uniquely indexed by the minimum of the indices involved, i.e., $k = \min(i, j)$. Here, we take the absolute value of the interpolation parameter $|\alpha(c)|$, which by definition can be negative now, as the interpolation is performed from the odd-indexed unitaries toward the even-indexed unitaries.

A. Accuracy analysis

In this subsection we investigate the accuracy of the single parametric UI and comparing it to Trotterizations. Numerical results are presented in the corresponding analysis for the n-parametric generalization in Sec. V E. Using the separation of the system parameter c into a grid vertex c_i and interpolation parameter $\alpha(c)$—see Eq. (14)—we can separate the target Hamiltonian given in Eq. (13),

$$\hat{H}(c) = \underbrace{(\hat{H}_0 + c_i \hat{H}_1)}_{=\hat{A}_i} + \alpha(c) \underbrace{\frac{\Delta c}{N} \hat{H}_1}_{=\hat{B}}. \tag{18}$$

Here we use the operators $\hat{A}_i$ and $\hat{B}$ corresponding to the (interval specific) generators of our analysis in Sec. III. The binning allows us to rescale the difference operator $\hat{B}$ to achieve our target infidelity. We show in Sec. III that the infidelity scales with the fourth power of the difference operator $\hat{B}$, so that the infidelity of our binned interpolation is proportional to $\mathcal{O}(\frac{\Delta c^4}{N^4})$. As a consistency check we give a symbolically derived BCH expansion in Appendix F 3 a.

By comparison, the infidelity of N_T Suzuki-Trotter steps [7]—see Eq. (G1)—scales quadratically $\mathcal{O}(\frac{c^2}{N_T^2})$ both in the number of steps N_T and the parameter amplitude c—see the BCH expansion in Appendix G 1. Similarly to the N bins of the UI, the number of Trotter steps N_T also rescales the difference operator $\hat{B} = \frac{\Delta c}{N}\hat{H}_1$.

We can improve on this ansatz by performing N_S symmetric Trotter steps (also known as split steps) [31]. In this case, the infidelity scales as $\mathcal{O}(\frac{c^2}{N_S^4})$—see Eq. (G2)—quadratically in the parameter amplitude c, but fourth order in the number of steps N_S.

As a side remark, the symmetrization approach used in the symmetric Trotter method, which removes all odd-ordered BCH correction terms of N_S, does not translate to significant improvements in the UI. The symmetric (split-step) UI, derived in Appendix H, still has fourth-order correction terms in the infidelity as the odd-ordered BCH correction terms are already zero. Thus, this method improves the infidelity only by a constant factor. Attempts to increase the order of convergence with an interpolation should instead rely on increasing the number of interpolation roots (i.e., points at which the interpolation is exact)—see also the argument in Sec. III.

TABLE I. Convergence orders of the infidelity for different methods. The expansion was performed in orders of $\mathcal{O}(\frac{(\Delta c)^r}{N^s})$, the Table lists the exponents r and s for the leading-order terms. In Trotterization schemes time is split into N steps. Similarly, in UI parameter amplitudes are split into N intervals (bins).

Method	r for $(\Delta c)^r$	s for N^s
Trotter	2	2
Sym Trotter	2	4
UI	4	4
Sym UI	4	4

An overview of the leading-order scaling of the different methods, both in the generators and the infidelity, is given in Table I.

In Fig. 3 we show the infidelity as a function of the system parameter c for a single parameter problem. We sample 100 random $d = 16$-dimensional Hamiltonians generated using the approach in Appendix C 2, with standard deviation of the eigenvalues $\mathrm{std}(\hat{E}_p) = \pi/2$ and the maximum parameter amplitude $c_{p,\max} = 0.025$. We compare $N \in [1, 2, 4, 8]$ interpolation bins and a single Trotter and symmetric Trotter step. A single (optimized) Trotter step requires same number of computational steps as the UI. Every doubling in the number of bins leads to a reduction of the infidelity by a factor of $2^4 = 16$.

B. Construction from cached matrices

Similar to the caching proposed in Sec. III B, we can cache bin-dependent matrices $\hat{L}_k$, $\hat{R}_k$, and $\hat{E}_k$ of the Schur

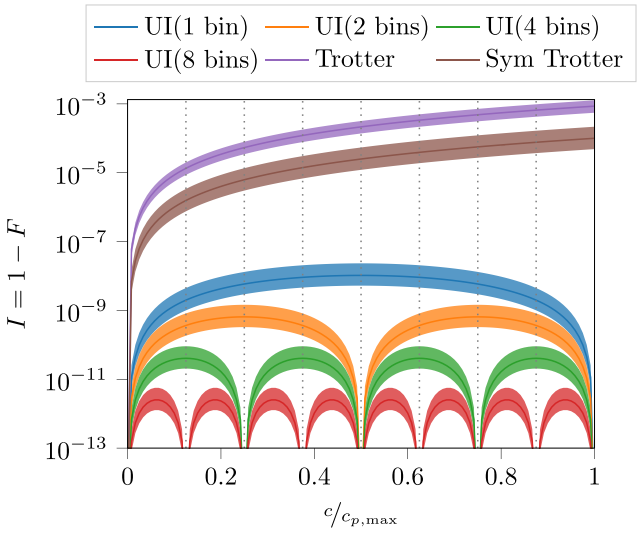

FIG. 3. The target-unitary infidelity of a single parameter problem using (symmetric) Trotterization and unitary interpolation with 1, 2, 4, or 8 bins. The plot shows one standard deviation of the infidelity sampled using 100 random $d = 16$-dimensional Hamiltonians with $\mathrm{std}(\hat{E}_p) = \pi/2$ and the maximum parameter amplitude $c_{p,\max} = 0.025$. A single UI interval improves the infidelity of the symmetric Trotterization by more than *three orders of magnitude*.

decomposition of the interpolation operators $\hat{W}_k$,

$$\hat{W}_k = (\hat{U}_j \hat{U}_i^\dagger) = \hat{L}_k \exp(-i\hat{E}_k)\hat{L}_k^\dagger, \tag{19}$$

where $k = \min(i, j)$ is the minimum of the involved grid indices, i being the odd index and $j = i \pm 1$ the even index. Furthermore, we define $\hat{R}_k = \hat{L}_k^\dagger \hat{U}_i$, so that the interpolation can be calculated via

$$\hat{U}_{\mathrm{UI}}(c) = \hat{L}_k \exp(-i\hat{E}_k|\alpha(c)|)\hat{R}_k. \tag{20}$$

The product of the diagonal exponential $\exp(-i\hat{E}_k|\alpha(c)|)$ with the right side matrix $\hat{R}_k$ can be performed as a row-wise scaling operation, which has complexity $\mathcal{O}(d^2)$. The relevant complexity then reduces to a single matrix multiplication.

C. Gradients

Gradient-based optimization techniques, e.g., standard gradient descent [32], noise propagation [33], as well as exact [34] and approximate second-order optimizers, such as Adam [35] and L-BFGS [36], require the gradients of the objective function with respect to the system parameters. Our interpolation scheme allows for the efficient calculation of the gradient of the unitaries with respect to the system parameter,

$$\frac{\mathrm{d}\hat{U}_{\mathrm{UI}}(c)}{\mathrm{d}c} = -\mathrm{sign}(\alpha)\hat{L}_k \frac{i\hat{E}_k N}{\Delta c} \exp(-i\hat{E}_k|\alpha(c)|)\hat{R}_k. \tag{21}$$

This removes the need for either (semi)automatic differentiation [37,38] or analytical gradients from diagonalization [17,39,40].

V. UNITARY INTERPOLATION OF MULTIPLE PARAMETERS

We generalize the one-parameter UI to interpolate unitaries of linear Hamiltonians with n system parameters—see Eq. (1). First, we generalize the one-parameter interpolation operators to the n-parameter case in Sec. V A. Afterward, we construct a regular tiling of n-dimensional parameter spaces and generalize the odd-indexed grid to an odd-summed lattice in Sec. V C. Eventually, we discuss the resulting n-parameter interpolation scheme in Sec. V D.

A. Displacement operators in n-parameter space

Let us now consider $n+1$ unitaries in $U(d)$, a seed unitary $\hat{V}_0 = \exp(-i\hat{A})$ and n unitaries $U_i = \exp(-i(\hat{A} + \hat{B}_i))$, with the difference operator $\hat{B}_i$ being the difference between the generators of $\hat{V}_0$ and $\hat{V}_i$. We construct n one-parameter interpolation operators $\hat{W}^{(i)}$, $i = 1, \ldots, n$ to displace the seed unitary $\hat{V}_0$ to the unitaries $\hat{V}_i$,

$$(\hat{W}^{(i)})^{\alpha_i} = (\hat{V}_i \hat{V}_0^\dagger)^{\alpha_i}. \tag{22}$$

The n-parameter interpolation operator $\hat{\mathcal{W}}$ is constructed as a product of the one-parameter interpolation operators,

$$\hat{\mathcal{W}}(\boldsymbol{\alpha}) = \prod_{i=1}^{n} (\hat{W}^{(i)})^{\alpha_i}, \tag{23}$$

with the interpolation parameter vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$. The action of this interpolation operator $\hat{\mathcal{W}}(\boldsymbol{\alpha})$ on the seed unitary $\hat{V}_0$ corresponds to an n-parameter interpolation,

$$\hat{V}(\boldsymbol{\alpha}) = \hat{\mathcal{W}}(\boldsymbol{\alpha})\hat{V}_0 = \left(\prod_{i=1}^{n} (\hat{V}_i \hat{V}_0^\dagger)^{\alpha_i}\right)\hat{V}_0, \tag{24}$$

which reproduces the unitaries used in its construction:

$$\hat{V}(\mathbf{0}) = \hat{V}_0 \quad \text{and} \quad \hat{V}(\mathbf{e}_i) = \hat{V}_i, \tag{25}$$

where $\mathbf{e}_i$ is the unit vector in the direction i. The first-order BCH expansion of the interpolation given in Eq. (24) can again be calculated from the sum of the generators, weighted by the interpolation parameters α_i, so that we can write

$$\hat{V}(\boldsymbol{\alpha}) = \exp\left(\underbrace{-i(\hat{A} + \boldsymbol{\alpha}^T \mathbf{B})}_{\text{Desired Terms}} + \sum_j p_j(\boldsymbol{\alpha})\hat{D}_j\right). \tag{26}$$

The first-order expansion reveals a linear equation of the interpolation parameters $\boldsymbol{\alpha}$ and the difference operators $\mathbf{B} = (\hat{B}_1, \ldots, \hat{B}_n)^T$, which we can use to approximate unitaries of n-parameter Hamiltonians defined in Eq. (1). These are the desired terms. The higher-order (undesired) terms are represented by a sum of polynomials $p_j(\boldsymbol{\alpha})$—see also Eq. (10)—and commutators $\hat{D}_j$ of the generators $\hat{A}$ and $\hat{B}_i$, i.e., $\hat{D}_1 = [\hat{B}_1, \hat{B}_2]$.

The undesired terms must vanish for $\boldsymbol{\alpha} = \mathbf{0}$ and whenever $\boldsymbol{\alpha} = \mathbf{e}_i$ for all i. This (again) requires the polynomials $p_j(\boldsymbol{\alpha})$ to be at least quadratic in the α_i, i.e., $\alpha_i\alpha_j$, to satisfy the $n+1$ roots.[3] In addition, using the fact that every term in the interpolation operators defined in Eq. (9) is linear in α_i and at least linear in $\hat{B}_i$, it must follow that the commutators $\hat{D}_j$ must be at least quadratic in the $\hat{B}_i$—i.e., $[\hat{B}_i, \hat{B}_j]$—as it was the case for the one-parameter UI.

B. Tiling of the n-dimensional parameter space

To construct the UI of a linear n-parameter Hamiltonian as the one given in Eq. (1), up to a target infidelity, for amplitude-constrained system parameters $c_p \in [c_{p,\min}, c_{p,\max}]$, we divide the n-dimensional parameter space V into a regular tiling of interpolation cells $V_{\mathbf{i}}$, to uniquely associate every point $\mathbf{c} \in V$ with an interpolation cell $V_{\mathbf{i}}$.

We divide the system parameter (hyper)volume into a simple cubic (hyper)lattice of $\mathbf{N} = (N_1, \ldots, N_n)$ bins[4] along the n parameter directions and the lattice-site parameter amplitudes

$$\mathbf{c_i} = \left(c_{1i_1}, \ldots, c_{ni_n}\right)^T \quad \text{with} \quad c_{p,i_p} = c_{p,\min} + i_p \frac{\Delta c_p}{N_p}. \tag{27}$$

These amplitudes are given by the lattice-site indices $\mathbf{i} = (i_1, \ldots, i_n)$, where $i_p \in \{0, \ldots, N_p\}$ and the lattice sizes $\Delta c_p = c_{p,\max} - c_{p,\min}$. Analogously we define the lattice-site unitaries $\hat{U}_{\mathbf{i}}$

$$\hat{U}_{\mathbf{i}} = \exp\left(-i\left(\hat{H}_0 + \mathbf{c_i}^T \hat{\mathbf{H}}\right)\right), \tag{28}$$

[3] Proof. Any linear polynomial of degree n can be written as $p(\boldsymbol{\alpha}) = \sum_{i=1}^{n} c_i\alpha_i + c_0$. From the seed root $p(\mathbf{0}) = 0$, it follows $c_i = 0$. From the n roots $p(\hat{e}_i) = 0$, it follows $c_i = 0$ for all i. So that only the trivial linear polynomial remains.

[4] Chosen to achieve a target infidelity, we discuss an approach for binning in Appendix F.

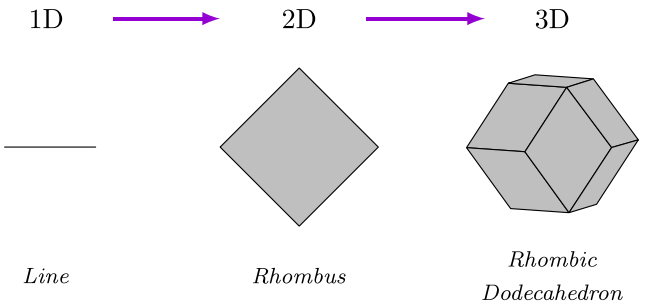

FIG. 4. Examples of interpolation lines, areas (rhombus), and volumes (rhombic dodecahedron) for one, two, and three parameter UI.

using the lattice-site parameter amplitudes $\mathbf{c_i}$. The interpolation operators are then defined from adjacent lattice-site unitaries $\hat{U}_\mathbf{i}$ and $\hat{U}_{\mathbf{i}+\mathbf{e}_p}$, with $\mathbf{e}_p$ pointing in the pth direction.

C. Voronoi cells

The interpolation (hyper)volumes are constructed as a regular Voronoi grid [41]. To construct the Voronoi grid, we choose the odd-summed lattice sites $\mathbf{c_i}$ with $\{\mathbf{i} \mid (\sum_{k=1}^{n} i_k) \bmod 2 = 1\}$ as seed points for the Voronoi cells $V_\mathbf{i}$. All points $\mathbf{c}$ that are closer to a seed point $\mathbf{c_i}$ than any other odd-summed grid points $\mathbf{c_{j\neq i}}$ are part of the Voronoi cell $V_\mathbf{i}$,

$$V_\mathbf{i} = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x} - \mathbf{c_i}\| = \min_\mathbf{j} \|\mathbf{x} - \mathbf{c_j}\|\}. \tag{29}$$

For points with two or more odd-summed lattice sites at equal distance, any choice between the neighbors can be made. The specific method we employ is discussed in Appendix A. In 1D the Voronoi cells divide the parameter space into line segments, in 2D the areas are given by rhombi and in 3D the volumes are described by rhombic dodecahedra; see Fig. 4.

D. Unitary interpolation

To approximate a unitary $\hat{U}(\mathbf{c})$, we first decompose the system parameters $\mathbf{c}$ into the odd-summed lattice site $\mathbf{c_i}$ (the seed of the corresponding Voronoi cell) and interpolation parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$, via

$$\mathbf{c} = \mathbf{c_i} + \boldsymbol{\alpha}\frac{\Delta\mathbf{c}}{\mathbf{N}}. \tag{30}$$

The interpolation parameter term $\boldsymbol{\alpha}\frac{\Delta\mathbf{c}}{\mathbf{N}}$ describes the relative location with respect to the seed point $\mathbf{c_i}$ of the corresponding Voronoi cell.

The interpolation is achieved via the appropriate interpolation operation $\hat{\mathcal{W}}_\mathbf{i}(\boldsymbol{\alpha})$,

$$\hat{\mathcal{W}}_\mathbf{i}(\boldsymbol{\alpha}) = \prod_{p=1}^{n} \left(\hat{U}_{\mathbf{i}+bs_p\mathbf{e}_p}\hat{U}_\mathbf{i}^\dagger\right)^{|\alpha_p|}, \tag{31}$$

where we use the signed index shift vector $\mathbf{s}_p = \mathrm{sign}(\alpha_p)\mathbf{e}_p$ for more clarity. The sign of the interpolation parameter α_p determines the direction in which we interpolate from the seed point $\mathbf{c_i}$ (up or down the grid). The product is used from the right to the left $\prod_{i=1}^{s} x_i = x_s \cdots x_2 x_1$. The interpolation is then

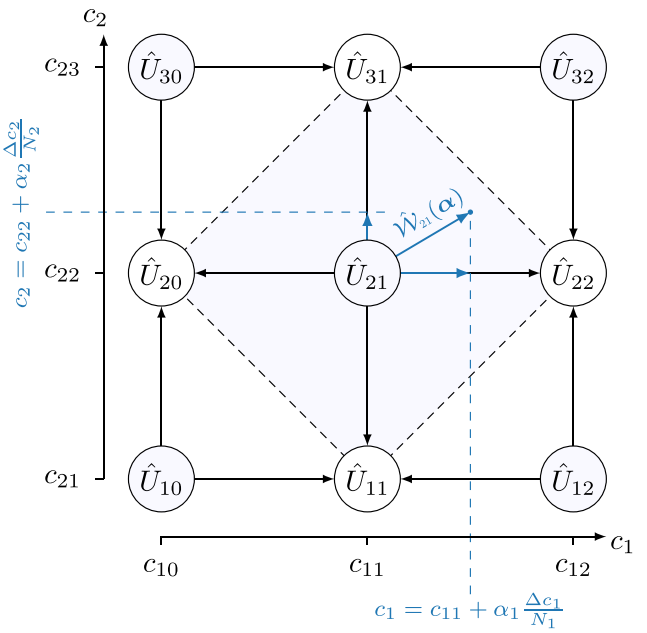

FIG. 5. A 2×2 Grid *segment* extracted from a two-parameter interpolation grid. The arrows point from the seed points along the approximate trajectory achieved by the one-parameter displacements. The index vectors are expanded into their two components, i.e., $\mathbf{i} = (2, 1) \to 21$. The interpolation is carried out from $\hat{U}_{21}$ via the interpolation operator $\hat{D}_{21}(\alpha)$.

constructed from

$$\hat{U}_{\mathrm{UI}}(\mathbf{c}) = \hat{\mathcal{W}}_\mathbf{i}(\boldsymbol{\alpha})\hat{U}_\mathbf{i} = \left(\prod_{p=1}^{n} \left(\hat{U}_{\mathbf{i}+bs_p}\hat{U}_\mathbf{i}^\dagger\right)^{|\alpha_p|}\right)\hat{U}_\mathbf{i}, \tag{32}$$

The interpolation procedure that starts at a seed point and ends at a point within its Voronoi cell is shown in Fig. 5 for a two-parameter interpolation grid.

E. Accuracy and performance

The decomposition given in Eq. (30) carries over the Hamiltonian

$$\hat{H}(c) = \underbrace{\hat{H}_0 + \mathbf{c_i}^T\hat{\mathbf{H}}}_{=\hat{A}_\mathbf{i}} + \boldsymbol{\alpha}^T \underbrace{\frac{\Delta\mathbf{c}}{\mathbf{N}}\hat{\mathbf{H}}}_{=\hat{\mathbf{B}}} \tag{33}$$

and is conveniently separated into the seed Hamiltonian $A_\mathbf{i}$ and the difference operator $\hat{\mathbf{B}}$, which scales with the bin sizes $\frac{\Delta\mathbf{c}}{\mathbf{N}}$. Our analysis in Sec. V A proved that in the same way as the one-parameter interpolation, the infidelity scales with the fourth power of the difference operator $\hat{\mathbf{B}}$. Using Eq. (33) we conclude, that the infidelity is proportional to $\mathcal{O}(\frac{\Delta\mathbf{c}^4}{\mathbf{N}^4})$ and that the results in Table I remain valid. The BCH expansion for n-parameter interpolations is given in Appendix F 3 b.

The leading-order correction is maximal at $\alpha_p = 1/2\,\forall p \in \{1, \ldots, n\}$. In Fig. 6 we show the infidelity as a function of the system parameters for a 5×5 grid. We compare the behavior of the UI and its symmetrization (symmetric UI), which is derived in Appendix H, with that of Trotterization (Trotter) and symmetric Trotterization (symmetric Trotter).

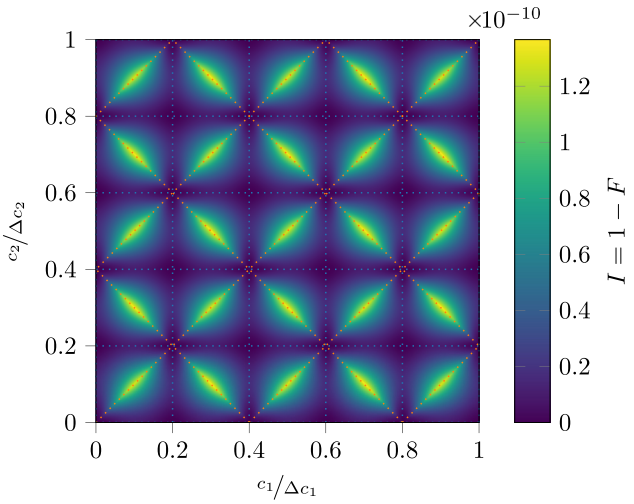

FIG. 6. Infidelity map of a two-parameter UI using 5×5 bins and a random Hamiltonian, see Appendix C 2, with $\mathrm{std}(\hat{E}_p) = \pi/2 \,\forall p \in \{1, \ldots, n\}$ and $\Delta c_p = 0.025$. The UI achieves an infidelity of $I(\mathbf{c}) < 1.37 \times 10^{-10}$.

The corresponding numerical results are shown in Fig. 7 and discussed below.

The infidelities in Fig. 7(a) and 7(b) converge according to the predictions in Table I. In (b) we can see that the UI, symmetric UI, and symmetric Trotterization are shifted by a constant factor in the number of steps (bins) N. This factor strongly depends on the parameter amplitudes $c_{p,\max}$ (with $c_{p,\min} = 0$). In Fig. 7(a) we show the infidelity as a function of the parameter amplitudes $c_{p,\max}$, highlighting the quadratic improvement of the UI methods over the Trotterizations for smaller parameter amplitudes. The symmetric UI improves on the UI only by a constant prefactor, which in our examples led to a fourfold reduction in the infidelity, which will generally not make up for the additional computational cost. Therefore, we generally recommend using the UI over the symmetric UI.

In Fig. 7(c) the infidelity of a single step (bin) is shown as a function of the number of system parameters n. Finally in Fig. 7(d) we show the walltime required to achieve a target infidelity I_{tar} for a 16-dimensional Hilbert space. The UI can achieve machine precision [Fig. 7(d)] without incurring additional computational costs beyond the initial caching and the computations of a single UI step. Trotterization, however, accumulates initial numerical errors (gray dashed line) through the repeated squaring of the initial step in the computation of $N = 2^j$ steps. Furthermore, the repeated squaring of (symmetric) Trotter steps add to the computational cost. We find an extrapolated speedup of three times for the UI over the symmetrized Trotterization in the construction of unitaries, when extrapolating the fidelity improvements of the Trotterization methods. Importantly, as was shown in Fig. 7(b), such fidelities cannot be achieved by Trotter due to the accumulation of numerical errors by repeated squaring. In Fig. 8 we compare the computational time as a function of the Hilbert space dimension for (symmetric) UI and (symmetric) Trotter (using a single step), with (Hermitian) eigenvalue decomposition-based exponentiation; see Appendix D. All methods show roughly a $\mathcal{O}(d^3)$ complexity, with roughly a 10-fold speedup for the matrix product approaches. Greater speedups are expected for GPU implementations. For a single step the Trotter approach and UI show similar walltimes, as they require exactly the same number of matrix multiplications.

F. Construction from cached matrices

The interpolation given in Eq. (32) can be computed from cached matrices. Using Eq. (20), we can decompose the one-parameter interpolation operators via Schur decomposition into eigenvector and diagonal eigenvalue matrices $\hat{V}_{\mathbf{i},\mathbf{i}\pm\mathbf{e}_p}$ and $\hat{E}_{\mathbf{i},\mathbf{i}\pm\mathbf{e}_p}$. The decomposition then reads

$$\left(\hat{U}_{\mathbf{i}+\mathbf{s}_p}\hat{U}_{\mathbf{i}}^{\dagger}\right) = \hat{V}_{\mathbf{i},\mathbf{s}_p} \exp\left(-i\hat{E}_{\mathbf{i},\mathbf{s}_p}\right)\hat{V}_{\mathbf{i},\mathbf{s}_p}^{\dagger}. \tag{34}$$

The diagonal eigenvector matrices $\hat{E}_{\mathbf{i},\mathbf{s}_p}$ are stored as vectors. From the eigenvector matrices we define the cached matrices

$$\hat{L}_{\mathbf{i},\mathbf{s}_n} = \hat{V}_{\mathbf{i},\mathbf{s}_n}, \quad \hat{R}_{\mathbf{i},\mathbf{s}_1} = \hat{V}_{\mathbf{i},\mathbf{s}_1}^{\dagger}\hat{U}_{\mathbf{i}}, \hat{C}_{\mathbf{i},\mathbf{s}_{m+1},\mathbf{s}_m} = \hat{V}_{\mathbf{i},\mathbf{s}_{m+1}}^{\dagger}\hat{V}_{\mathbf{i},\mathbf{s}_m}. \tag{35}$$

Using these cached matrices, we can construct the UI via

$$\hat{U}_{\mathrm{UI}}(\boldsymbol{\alpha}) = \hat{L}_{\mathbf{i}+\mathbf{s}_n} \exp\left(-i\hat{E}_{\mathbf{i}+\mathbf{s}_n}|\alpha_n|\right) \times \left(\prod_{p=1}^{n-1} \hat{C}_{\mathbf{i},\mathbf{s}_{p+1},\mathbf{s}_p} \exp\left(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p}|\alpha_p|\right)\right)\hat{R}_{\mathbf{i}+\mathbf{s}_1}. \tag{36}$$

The multiplication of the diagonal exponentials $\exp(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p}|\alpha_p|)$ can again be performed as a row-wise scaling operation with negligible computational cost, effectively reducing the computational cost to n matrix multiplications.

G. Gradients

In Sec. IV C we discussed the utility of calculating the gradients of the unitaries with respect to the system parameters. These gradients are efficiently calculated from the cached matrices. For the derivative $\frac{\partial}{\partial c_p}$, we replace the energy exponential $\exp(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p}|\alpha_p|)$ in Eq. (36) with its derivative $-\frac{i\hat{E}_{\mathbf{i}+\mathbf{s}_p}N_p}{\Delta c_p}\exp(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p}|\alpha_p|)$.

To avoid unnecessary matrix operations in the computation of the derivatives, we adapt the differentiation trick from Ref. [15]. Let $\hat{U}(\mathbf{c}) = (\prod_{p=1}^{n}\hat{C}_p\hat{A}_p(\alpha_p))\hat{C}_0$ be the matrix product of an n-parameter interpolation, where we neglect the grid indices for readability. We replace $\hat{L}$ with $\hat{C}_n$, $\hat{R}$ with $\hat{C}_0$, and the diagonal-matrix exponential terms with $\hat{A}_p(\alpha_p)$. The gradients can be calculated efficiently via the chain rule, by first defininning $\hat{P}_0 = \hat{U}(\mathbf{c})\hat{A}_0^{\dagger}(c_0)\hat{C}_0^{\dagger}$, $\hat{Q}_0 = \hat{C}_0$, and then iteratively starting from $i = 1$ to $i = N$ calculating $\hat{P}_i = \hat{P}_{i-1}(\hat{A}_i(c_i)\hat{C}_i)^{\dagger}$, then $\frac{\mathrm{d}\hat{U}(\mathbf{c})}{\mathrm{d}c_i} = \hat{P}_i\frac{\mathrm{d}\hat{A}_i}{\mathrm{d}c_i}\hat{Q}_{i-1}$ and then updating $\hat{Q}_i = (\hat{C}_i\hat{A}_i(c_i))\hat{Q}_{i-1}$.

This requires only three additional matrix multiplications per derivative, and compares favorably with diagonalization-based differentiation, which requires four additional matrix multiplications per system parameter; see Eq. (10) in Ref. [34].

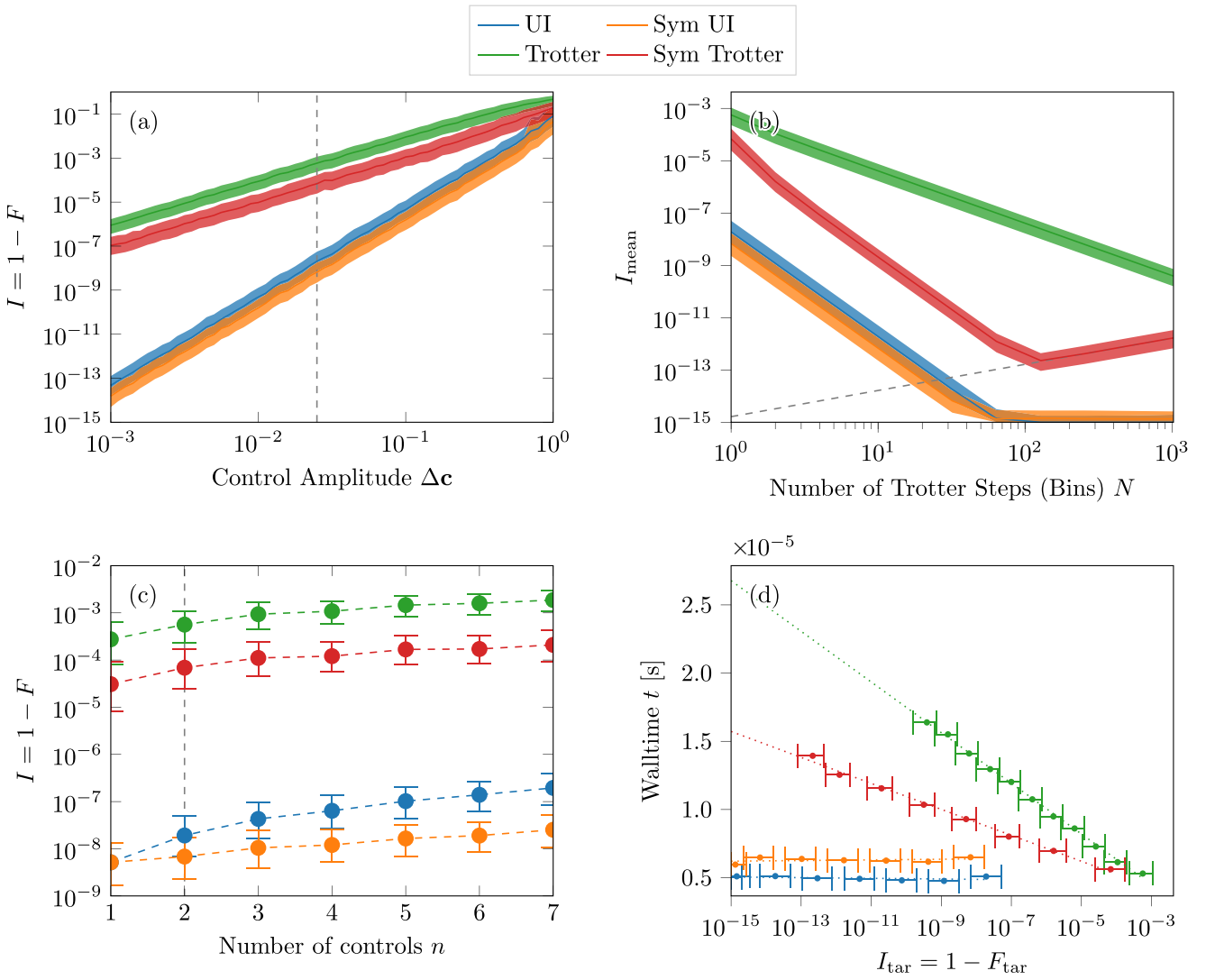

FIG. 7. Comparison of (symmetric) UI and (symmetric) Trotterization methods for the construction of unitaries. The infidelity I is shown as a function of (a) the maximum parameter amplitudes $c_{p,\max}$, (b) the number of Trotter steps or bins N, and (c) the number of system parameters n. The walltime t needed to achieve a target infidelity I_{tar} is shown in panel (d) for different numbers of Trotter steps and bins, using powers of two, i.e., 1, 2,..., 2^k Trotter steps/bins. Results are sampled 100 times for every datapoint from $d = 16$-dimensional random Hamiltonians, constructed via the procedure in Appendix C 2, with $\text{std}(\hat{E}_p) = \pi/2 \, \forall p \in \{1, \ldots, n\}$. By default the unvaried parameters are set to the maximal parameter amplitude of $c_{p,\max} = 0.025$, $n = 2$ system parameters and a single $N = 1$ Trotter step (interpolation bin). These default values are highlighted by the vertical gray dashed lines when varied in a plot.

VI. STATE EVOLUTION

Tasks such as state transfer optimization [15] or the optimization of subsystem dynamics [16] do not require the full unitary but only its action on a wave function $|\psi\rangle = \hat{U} \, |\psi_0\rangle$ or a (sampled) set of wave functions.

While this does not impact the caching routine, it allows us to replace matrix-matrix operations with time complexity $\mathcal{O}(d^3)$ with matrix-vector operations with time complexity $\mathcal{O}(d^2)$ once the cache is constructed. Trotterization techniques can benefit from the improved scaling as well, but at the cost of performing Trotter steps sequentially. The repeated squaring approach explored in Sec. V E, that allowed the construction of exponentially many Trotter steps $T_1^{2^j} = T_j$ via j iterative steps $T_{j+1} = T_j^2$ is no longer possible. Furthermore, the divide and conquer method used in the eigendecomposition of the Hamiltonian can not profit from this, and is replaced by direct computation of the action of the matrix exponential onto the wave function via Lanczos iterations [43].

To compare this approach with the other methods, we adeveloped a CYTHON [44] version of the *scipy* implementation [42]. To improve the performance, we reuse the computation of the hyperparameters of the Lanczos algorithm to reach a target accuracy between different exponentiations, further speeding up the procedure with respect to the scipy implementation.

In Fig. 9 we compare the time to reach a target infidelity for the different methods. The UI achieves machine precision *one order of magnitude faster* than the Lanczos algorithm, which already outperforms Trotter-based methods.

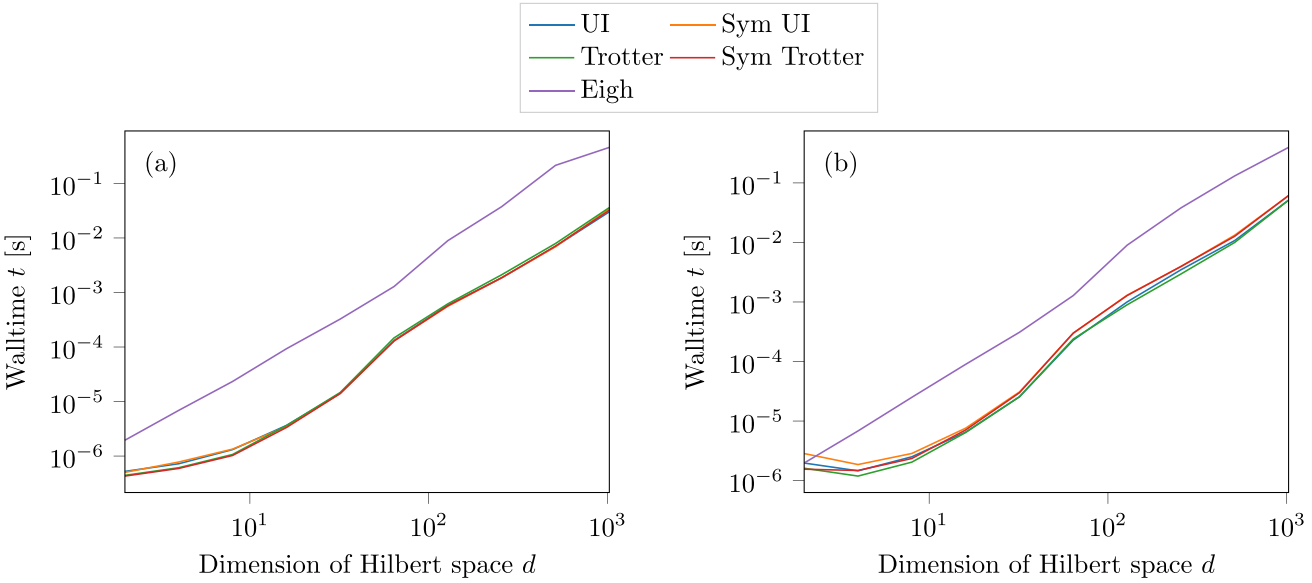

FIG. 8. Walltime required to compute a unitary as a function of the Hilbert space dimension d. We compare eigh [42], *a single* (symmetric) Trotter step, or the (symmetric) unitary interpolation for (a) single parameter and (b) two-parameter problems, using the system parameters from Fig. 7. Both eigh and (symmetric) unitary interpolation achieve machine precision in this time, whereas the fidelity of a single Trotter step is about 10^{-4}, as explored in Fig. 7.

VII. CACHING

The UI replaces the repeated diagonalization of Hamiltonians with the initial creation of an interpolation cache. This cache scales as $\mathcal{O}(\prod_{i=1}^{n} N_i)$, where N_i is the number of interpolation bins along the ith parameter direction (a precise formula for the cache size is derived in Appendix E).

To reduce cache size while maintaining accuracy, we optimize the cache size. As a constraint, we require that the accuracy of the constructed unitaries is below a target infidelity I_{tar}. The optimization routine is developed in Appendix F. Our approach consists of three main steps:

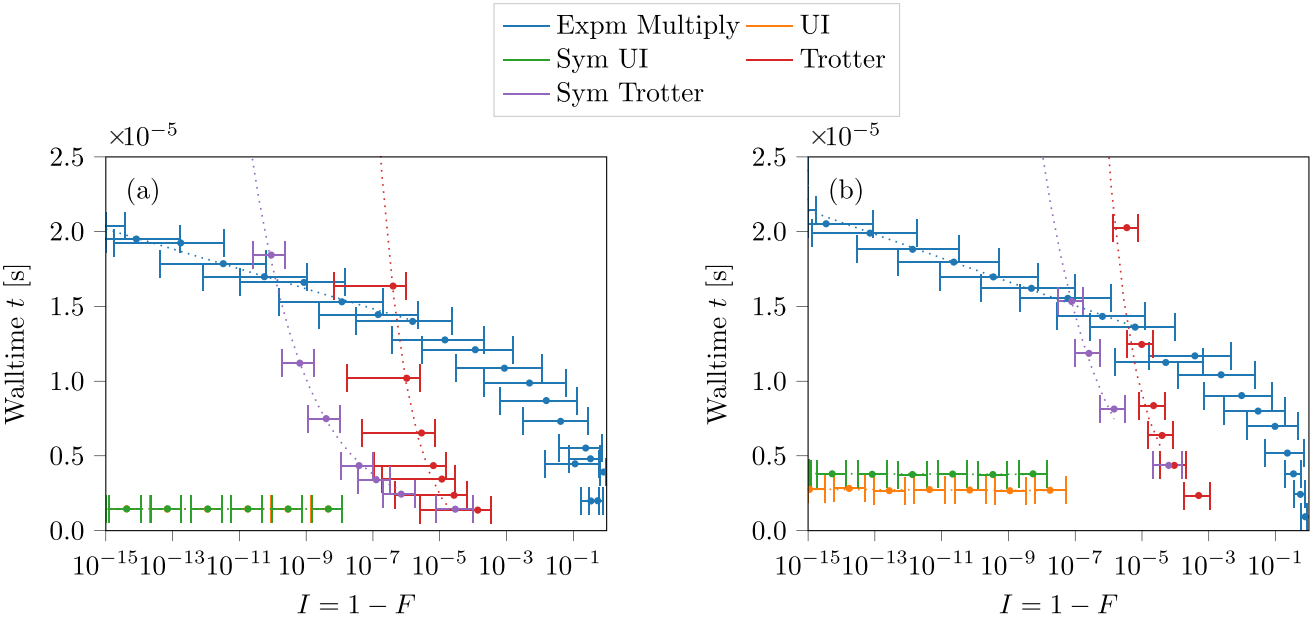

FIG. 9. Required walltime for *state evolution* to reach a target infidelity for (a) single parametrer and (b) two-parameter problems, using the system parameters from Fig. 7. The Trotter-based methods perform worse, because every step has to be executed. Expm multiply [42] computes the action of the Hamiltonian onto the wave function using the Krylov subspace method with Lanczos iterations [14]. We find the UI to be about 10 times faster than the Krylov subspace method.

TABLE II. Required cache size C as a function of system parameters n, the maximum parameter amplitudes $c_{p,\max}$ (and $c_{p,\min} = 0$), the error of the optimization in determining the optimal binning, and the accuracy of the infidelity estimation. Cache is supposed to achieve $I_{\text{tar}} = 10^{-12}$. The worst estimation with a relative Error >1% is highlighted in red.

n	Parameter amplitudes	Cache size	Rel. err. cache optimum	Rel. acc. infidelity
1	$[2.5]10^{-2}$	9.6 ± 1.7	0 ± 0	$(1.7 \pm 1.3)10^{-3}$
2	$[2.5,\ 1]10^{-2}$	$(1 \pm 0.3)10^{2}$	$(1.9 \pm 6.8)10^{-3}$	$(8.7 \pm 8)10^{-2}$
2	$[2.5,\ 2.5]10^{-2}$	$(2.4 \pm 0.7)10^{2}$	$(9.3 \pm 44.1)10^{-4}$	$(2.10 \pm 2.6)10^{-2}$
3	$[2.5,\ 1,\ 1]10^{-2}$	$(1.3 \pm 0.5)10^{3}$	$(4.6 \pm 12.3)10^{-3}$	$(8.9 \pm 6.5)10^{-2}$
3	$[2.5,\ 2.5,\ 2.5]10^{-2}$	$(7.8 \pm 2.6)10^{3}$	$(1.1 \pm 3.7)10^{-3}$	$(7.4 \pm 5.4)10^{-2}$
4	$[2.5,\ 1,\ 1,\ 1]10^{-2}$	$(2 \pm 0.9)10^{4}$	$(?)(2.6 \pm 4.3)10^{-2}$	$(8.10 \pm 7.6)10^{-2}$
4	$[2.5,\ 2.5,\ 2.5,\ 2.5]10^{-2}$	$(3.1 \pm 1.3)10^{5}$	$(5.8 \pm 12.4)10^{-3}$	$(9.4 \pm 6.4)10^{-2}$

(1) Estimate the maximum infidelity $I(\mathbf{N})$ as an inverse-square relationship of the binning configuration $\mathbf{N}$

$$I(\mathbf{N}) \approx \frac{1}{d+1} \sum_{i \leqslant j}^{n} \frac{1}{N_i^2 N_j^2} T_{ij}, \tag{37}$$

where $T_{ij} = 2\text{Re}[\text{Tr}(\hat{A}_{ij}^2)]$ are bidirectional error terms constructed via Eq. (F4). We compare an interpolation using a test binning (a single interpolation cell and $N_i = 1$ for all i)[5] with the exact diagonalization at the halfway point along the ith and jth axis.

The approximation is based on our previous results for the convergence order from Sec. V E and can be derived from the n-dimensional Taylor expansion of the infidelity in orders of $\frac{1}{N_i}$ for $i \in \{1, \ldots, n\}$. The cache optimization is performed using Eq. (37) for the infidelity. The test binning has to be performed only once, to estimate the parameters T_{ij}.

(2) We increase the cache size to reach the target infidelity I_{tar}, by iteratively increasing the number of bins along the axis j that has the highest ratio of infidelity decrease to cache size increase; see Eq. (F6).

(3) We this solution via two additional optimization procedures:

(i) Using a local search, we check if any neighboring positions have lower cache size while maintaining an estimated infidelity below I_{tar}. We repeat this until no further improvement is possible.

(ii) Using a diagonal search, we decrease the binning test-wise in one direction, we can find the first binning in a second direction, so that the infidelity is below I_{tar} and except the resulting binning, if the resulting cache size is smaller than before. Again, we repeat this procedure until no further improvement is possible.

We compare the quality of the bin optimization with results from exhaustive search, in Table II, sampled 100 times for different parameter amplitude combinations, $c_{p,\max} \in \{0.25, 1.0\}$ for Hamiltonians with $\text{std}(\hat{E}_p) = \pi/2$, and number of parameters n. This demonstrates that we can estimate the infidelity with less than 10% error, and find bin optima typically within 1% of the optimal binning.

[5]Test binnings with other values of N_i can be performed by rescaling Δc_i accordingly.

VIII. CONCLUSION

In this work we derived a new framework for the repeated computation of unitaries $\hat{U}(\mathbf{c})$ generated by parametric Hamiltonians $\hat{H}(\mathbf{c})$ with varying system parameters $\mathbf{c}$. We used two key insights to speedup such computations. First, UI guarantees fourth-order convergence in system parameters c_p and interpolation bins N_i. Second, the interpolation can be performed on a grid, which allows us to (in principle) reach arbitrary numerical precision. The interpolation requires the same amount of matrix multiplications as a single Trotter step, but with the benefit of only requiring one step to achieve machine precision, whereas Trotterization needs to be repeated to achieve desired target accuracies.

We optimized the binning to further reduce the time and space needed for caching matrices as this remains the main bottleneck especially for systems with many system parameters or large variation in the parameter amplitudes $\Delta c_p = c_{p,\max} - c_{p,\min}$. A further limitation is the reliance on dense operators. It remains unclear whether the interpolation can bring additional speedup for sparse operators.

All methods were implemented in Cython (C compiled for use in python) [44] and are available at Ref. [45]. The code can be installed via pip using: pip install unipolator.

ACKNOWLEDGMENTS

This work was funded by the Federal Ministry of Education and Research (BMBF) within the framework programme "Quantum technologies—From basic research to market" (Project QSolid, Grant No. 13N16149, Project Spinning, Grant No. 13N16210), by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1–390534769 and by HORIZON-CL4-2022-QUANTUM-01-SGA Project under Grant No. 101113946 OpenSuperQPlus100.

APPENDIX A: FINDING THE ASSOCIATED VORONOI CELL

To determine the interpolation cell, in which a point $\mathbf{c}$ lies, we need to find the closest odd-summed lattice site $\mathbf{i}_c$. We do

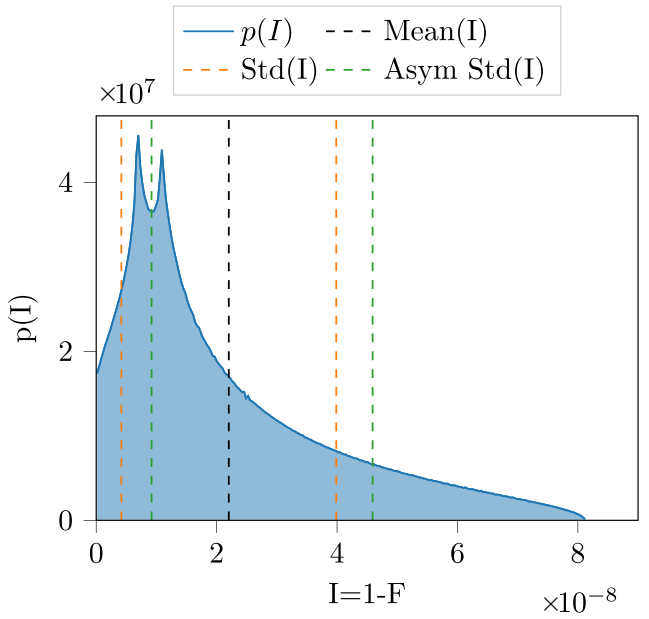

FIG. 10. The infidelity distribution of a two-dimensional UI, its mean, and the bounds of both the classical and asymmetric standard deviation. The values below the mean are clustered more strongly in two peaks, whereas the values above the mean are more spread out, with very few large values contributing the most to the standard deviation.

this by first finding the closest lattice site $\tilde{\mathbf{i}} = (\tilde{i}_1, \ldots, \tilde{i}_n)$ with

$$\tilde{i}_p = \left\lfloor \frac{N(c_p - c_{p,\min}) + 0.5}{\Delta c_p} \right\rfloor, \quad p = 1, \ldots, n. \tag{A1}$$

If the lattice site is odd-summed, i.e., $\sum_{p=1}^{n} \tilde{i}_p \mod 2 = 1$, then $\mathbf{i}_c = \tilde{\mathbf{i}}$, otherwise we need to find the closest odd-summed lattice site $\mathbf{i}$. To do so, we calculate the relative location $\tilde{\boldsymbol{\alpha}}$ between $\mathbf{c}$ and the closest lattice site

$$\tilde{\boldsymbol{\alpha}} = \mathbf{c} - \mathbf{c}_{\tilde{\mathbf{i}}}, \quad \text{where} \quad \mathbf{c}_{\tilde{\mathbf{i}}} = \mathbf{c}_{\min} + \tilde{\mathbf{i}} \frac{\Delta \mathbf{c}}{\mathbf{N}}. \tag{A2}$$

The interpolation cell $\mathbf{i}$ is then found by moving in the direction of the largest component $m = \text{argmax}_p |\tilde{\alpha}_p|$ of $\tilde{\boldsymbol{\alpha}}$, so that

$$\mathbf{i} = \tilde{\mathbf{i}} + \text{sign}(\tilde{\alpha}_m)\mathbf{e}_m. \tag{A3}$$

In case there are multiple largest components, we choose the one with the smallest index m.

APPENDIX B: ASYMMETRIC STANDARD DEVIATIONS

The infidelity distribution of the UI (but also of the Trotterization schemes) as a function of the system parameters is not normally distributed; see Fig. 10. As a result the use of a standard deviation can be misleading. Furthermore, as the average infidelity approaches zero (with increasing numbers of bins or Trotter steps), the standard deviation can be larger than the average, which would suggest values outside of the possible range $I \in [0, 1]$. We therefore decided to use an asymmetric standard deviation, with separately calculated standard deviations for values below and above the average infidelity. We calculated this standard deviation using the following formula:

$$\text{std}_{\min}(\mathbf{I}) = \sqrt{\frac{\sum_{I_i \leqslant I_{\text{avg}}} (I_i - I_{\text{avg}})^2}{\left(\sum_{I \leqslant I_{\text{avg}}} 1\right) - 1}},$$

$$\text{std}_{\max}(\mathbf{I}) = \sqrt{\frac{\sum_{I_i \geqslant I_{\text{avg}}} (I_i - I_{\text{avg}})^2}{\left(\sum_{I \geqslant I_{\text{avg}}} 1\right) - 1}}. \tag{B1}$$

The asymmetric standard deviations are used throughout this paper and are either given by error bars or by a transparent area around a line plot.

APPENDIX C: ESTIMATING AVERAGE INFIDELITIES

In this paper we construct the average infidelity of the interpolation (approximation) with respect to the exact unitary. For this purpose we average the infidelity over

(1) all possible states (analytically);

(2) sampled Hamiltonians with a gaussian eigenvalue distribution (numerically);

(3) the parameters within the interpolation/approximation hypervolume (semianalytical).

In the following sections we explain the procedures for each of those averaging procedures. For completeness we also estimate the maximum infidelity over the space of interpolation/approximation parameters.

1. Averaging infidelity over possible states

The fidelity $F(|\Phi\rangle)$ for an initial state Φ, between an approximated time evolution $\hat{U}_{\text{ap}}$ and the exact evolution $\hat{U}_{\text{ex}}$ is given by the probability that the approximated final state $\hat{U}_{\text{ap}}|\Phi\rangle$ collapses onto the exact final state $\hat{U}_{\text{ex}}|\Phi\rangle$ when measured in an eigenstate of the final state:

$$F(|\Phi\rangle) = |\langle\Phi|\hat{U}^{\dagger}_{\text{exact}}\hat{U}_{\text{approx}}|\Phi\rangle|^2. \tag{C1}$$

Using the overlap operator $\hat{M} = \hat{U}^{\dagger}_{\text{exact}}\hat{U}_{\text{approx}}$ the fidelity can be written in the short form:

$$F(|\Phi\rangle) = |\langle\Phi|\hat{M}|\Phi\rangle|^2. \tag{C2}$$

Equivalently the infidelity is defined as $I(|\Phi\rangle) = 1 - F(|\Phi\rangle)$, and corresponds to the rate of error for a measurement in the eigenbasis of $\hat{U}_{\text{exact}}$. Integrating the infidelity $F(|\Phi\rangle)$ over the space of all normalized wave vectors $|\Phi\rangle$ we use

$$I_{\text{av}} \overset{\text{def}}{=} 1 - \int |\langle\Phi|\hat{M}\,|\Phi\rangle\,|^2 dS^{2d-2} = \frac{d}{d+1} - \frac{|\text{Tr}(\hat{M})|^2}{d(d+1)} \tag{C3}$$

for the average infidelity, as derived in Eq. (4.4) in Ref. [46] and more approachably derived again in Ref. [47]. where d refers to the $d \times d$-dimensional overlap operator $\hat{M} \in \mathbb{C}^{d\times d}$.

2. Sampling random Hamiltonians

We average the infidelities using random Hamiltonians $\hat{H}$ of Hilbert space dimension d with normally distributed random eigenvalues $\hat{E}_i = \mathcal{N}(0, \sigma^2)$, as this emulates the

eigenvalue distribution of large collections of particles (Stirling limit). Starting from these randomly selected eigenvalues, we construct the Hamiltonians via the transformation $\hat{H} = \hat{U}\,\mathrm{diag}(\hat{E})\hat{U}^\dagger$, where $\hat{U}$ is a random unitary matrix (generated from a Haar measure).

a. Generating Haar measure random unitaries

We generate s-dimensional random unitaries $\hat{U}$ (from a Haar measure) using the following procedure:

(1) generate a random complex matrix $\hat{X}$ of size $d \times d$ with real and imaginary parts chosen using normally distributed pseudo random numbers (using the PCG-64 algorithm [48]) $\hat{X}_{i,j} = \mathcal{N}(0,1) + i\mathcal{N}(0,1)$;

(2) extract a unitary matrix $\hat{U}$ from $\hat{X}$ using the QR decomposition (Gram-Schmidt orthogonalization and normalization) $\hat{U}, \hat{R} = \mathrm{QR}(\hat{X})$.

3. Averaging infidelity over the interpolation/approximation volume

We estimate the average infidelity over the interpolation/approximation by fitting a set of basis functions $u_{\mathrm{UI}}(\boldsymbol{\alpha}, \mathbf{j})$ to samples of the infidelity within the interpolation/approximation volume. For each basis function we cache the average infidelity, which we calculate analytically. The average infidelity is then given by the weighted sum of the basis function averages, where the weights are the coefficients of the basis function expansion, determined via the regression procedure.

a. Basis functions

We choose n-dimensional polynomial basis functions of degrees 1 to m,

$$u(\boldsymbol{\alpha}, \mathbf{j}) = \prod_{i=1}^{n} \alpha_i^{j_i} \quad \text{with} \quad 1 \leqslant \sum_{i=1}^{n} j_i \leqslant m, \tag{C4}$$

using the fact that at $\boldsymbol{\alpha} = 0$ both approximation and interpolation is exact [$I_{\mathrm{av}}(\boldsymbol{\alpha} = 0) = 0$]. We then approximate the infidelity as a linear combination of the basis functions

$$I(\boldsymbol{\alpha}) \approx \sum_{\mathbf{j} \in \mathbb{N}^n}^{1 \leqslant \|\mathbf{j}\| \leqslant m} c_{\mathbf{j}} u(\boldsymbol{\alpha}, \mathbf{j}). \tag{C5}$$

The coefficients $c_{\mathbf{j}}$ are estimated by linear regression of sample values of the infidelity at randomly chosen points in the interpolation volume. There are $\binom{m+n}{n} - 1$ basis functions of degrees $1 \leqslant \|\mathbf{j}\| \leqslant m$, which we can estimate accurately by sampling $\binom{m+n}{n-1} - 1$ points. Furthermore, we can use the fact that in the case of the UI the infidelity returns to zero at the corners $I(\boldsymbol{\alpha} = \hat{e}_k) = 0$, where $\hat{e}_k$ for $k = 1, \ldots, n$ are the unit vectors in the kth direction. This reduces the number of required sample points by n.

b. Average infidelities

We calculate the average infidelity by integrating over the interpolation/approximation volume. This is accomplished by calculating and caching the average infidelity of the basis functions $\overline{u}(\mathbf{j})$ over the interpolation volume, and reusing the regression coefficients $c_{\mathbf{j}}$:

$$I_{\mathrm{av}} \approx \sum_{\mathbf{j} \in \mathbb{N}^n}^{1 \leqslant \|\mathbf{j}\| \leqslant m} c_{\mathbf{j}} \overline{u}(\mathbf{j}). \tag{C6}$$

The integration of the basis functions has to be performed separately for UI and Trotter techniques, due to the different interpolation volumes. We focus on the positive quadrant $\alpha_i \leqslant 0\,\forall i \in [1, n]$, as all other quadrants are structurally equivalent.

Average infidelity of Trotterizations. For Trotterizations the interpolation volume is given by a hypercube, which allows for the separation of the n-dimensions in the integral

$$\begin{aligned} \overline{u}_{\mathrm{Tr}}(\mathbf{j}) &= \frac{1}{\mathcal{V}_{\mathrm{Tr}}} \int_{\mathcal{V}_{\mathrm{Tr}}} u_{\mathrm{Tr}}(\boldsymbol{\alpha}, \mathbf{j}) d\boldsymbol{\alpha} \\ &= \prod_{i=1}^{n} \int_0^1 \alpha_i^{j_i} d\alpha_i = \prod_{i=1}^{n} \frac{1}{j_i + 1}. \end{aligned} \tag{C7}$$

Average infidelity of the unitary interpolation. In the case of the UI the solution is not as straightforward, as we need to integrate over the interpolation cells. Using Eq. (30), i.e., the definition of the Voronoi cells, Eq. (29) can be restated as

$$\begin{aligned} &\alpha_p \in (-1, 1) \quad \forall \quad p \in \{1, \ldots, n\}, \\ &|\alpha_p| \geqslant 1 - |\alpha_q| \quad \forall \quad q \neq p \in \{1, \ldots, n\}. \end{aligned} \tag{C8}$$

For every direction the integral is limited by the largest component $\alpha_{\max}$, so that all other components α_i satisfy $\alpha_i \leqslant \min(\alpha_{\max}, 1 - \alpha_{\max})$. Therefore, we split the integral into n cases, one for each parameter, so that that parameter is the largest component in $\boldsymbol{\alpha}$:

$$\begin{aligned} \overline{u}_{\mathrm{UI}}(\mathbf{j}) &= \frac{1}{\mathcal{V}_{\mathrm{UI}}} \int_{\mathcal{V}_{\mathrm{UI}}} u(\boldsymbol{\alpha}, \mathbf{j}) d\boldsymbol{\alpha} \\ &= \frac{1}{\mathcal{V}_{\mathrm{UI}}} \sum_{i=1}^{n} \int_0^1 \alpha_i^{j_i} \left(\prod_{h \neq i}^{n} \int_0^{\min(\alpha_i, 1-\alpha_i)} \alpha_h^{j_h} d\alpha_h \right) d\alpha_i, \end{aligned} \tag{C9}$$

with the hypervolume $\mathcal{V}_{\mathrm{UI}} = 2^{n-1}$. We split up this integral further into the two cases $\alpha_i \in [0, 1/2]$ and $\alpha_i \in (1/2, 1]$, to remove the min function in the integral bounds:

$$\begin{aligned} \overline{u}_{\mathrm{UI}}(\mathbf{j}) &= \frac{1}{\mathcal{V}_{\mathrm{UI}}} \sum_{i=1}^{n} \int_0^{1/2} \alpha_i^{j_i} \left(\prod_{h \neq i}^{n} \int_0^{\alpha_i} \alpha_h^{j_h} d\alpha_h \right) d\alpha_i \\ &\quad + \frac{1}{\mathcal{V}_{\mathrm{UI}}} \sum_{i=1}^{n} \int_{1/2}^1 \alpha_i^{j_i} \left(\prod_{h \neq i}^{n} \int_0^{1-\alpha_i} \alpha_h^{j_h} d\alpha_h \right) d\alpha_i \\ &= \frac{1}{\mathcal{V}_{\mathrm{UI}}} \sum_{i=1}^{n} \int_0^{1/2} \alpha_i^{j_i} \left(\prod_{h \neq i}^{n} \frac{\alpha_i^{j_h+1}}{j_h + 1} \right) d\alpha_i \\ &\quad + \frac{1}{\mathcal{V}_{\mathrm{UI}}} \sum_{i=1}^{n} \int_{1/2}^1 \alpha_i^{j_i} \left(\prod_{h \neq i}^{n} \frac{(1-\alpha_i)^{j_h+1}}{j_h + 1} \right) d\alpha_i. \end{aligned} \tag{C10}$$

This expression can be expanded into a polynomial sum and can be evaluated quickly for every basis function (in our case via vector operations).

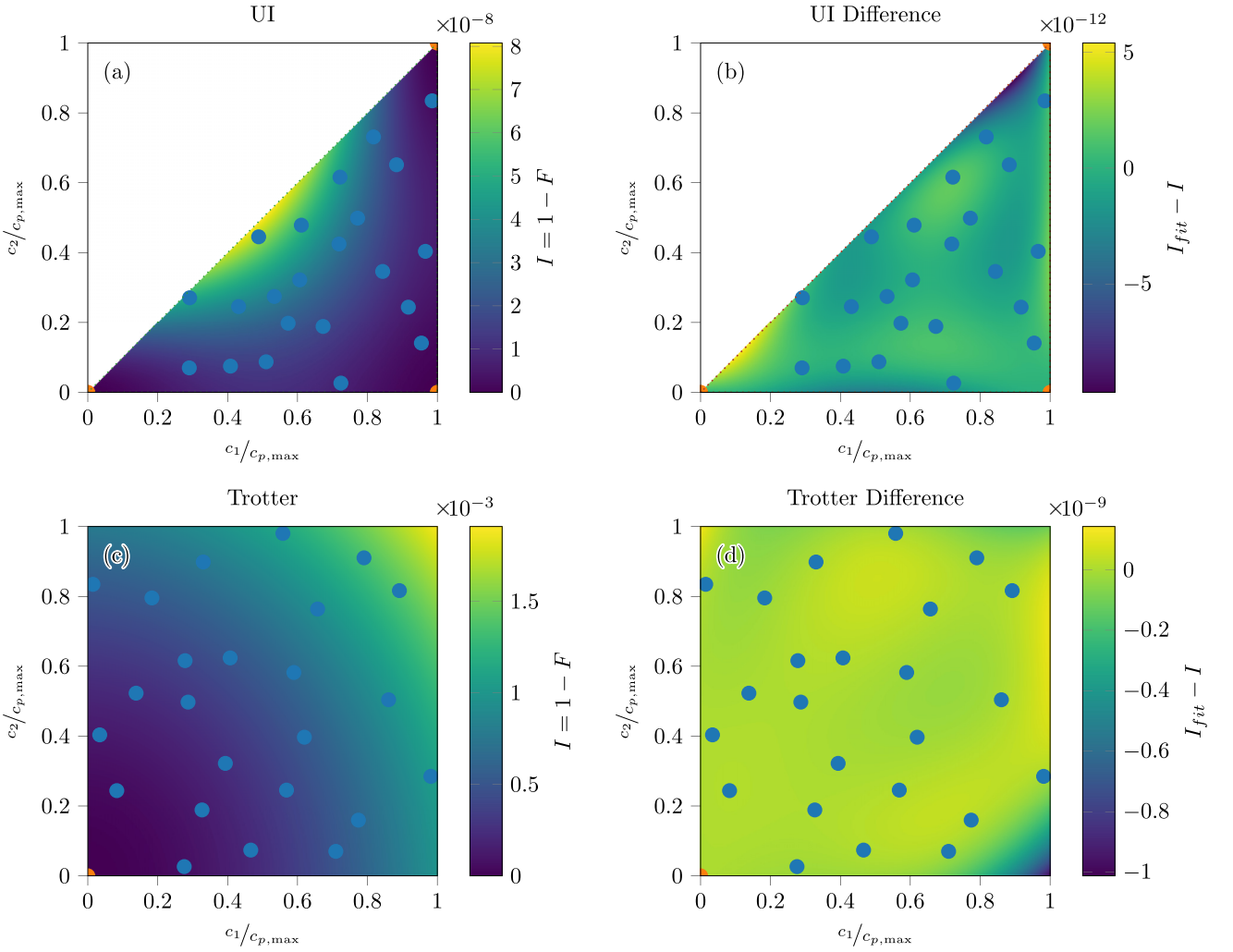

FIG. 11. Infidelity maps of two-parameter UI (a) and Trotterization (c) with sample points at a point ratio of 2.0 in blue and boundary condition points in orange with known vanishing infidelity, from which we construct an approximation to the infidelity map using second-order Taylor expansion (neglecting zeroth-order terms). The difference between the estimation and the true infidelity map for unitary interpolation and Trotterization are shown in panels (b) and (d), respectively. We find a relative accuracy of $<10^{-4}$ for the unitary interpolation and $<10^{-7}$ for the Trotterization. The fit works extremely well close to the sample points used in the estimation, the decrease in accuracy can be seen in the top-right corner of panel (b) and bottom-right corner of panel (d).

c. Standard deviations

To evaluate the asymmetric standard deviations from the average infidelity, we sample the fitted infidelity function at 100 randomly chosen sample points. This allows a fast evaluation of the standard deviation, as the contribution of every basis function can be cached for every sample point, thus avoiding the evaluation of the matrix exponentials.

d. Quality of the basis function estimates

We test the quality of the basis function approach to approximate the average infidelity using two times as many sample points as fit parameters. In Fig. 11 we show the infidelity maps of the two-dimensional UI and Trotterization and the deviations from the true infidelity as a function of the system parameters. We find low relative errors of $<10^{-4}$ for the UI and $<10^{-7}$ for the Trotterization across the space of system parameters with high accuracies close to the sample points used in the estimation. We found that using just as many sample points as fit parameters is sufficient to obtain a good approximation as long as the infidelities do not approach the double precision limit of 10^{-16}, we avert this problem via the choice of two times as many sample points as fit parameters. In Fig. 12 we explore the behaviour of the fit method for UI, symmetric UI, Trotterization, and symmetric Trotterization as a function of number of system parameters. For The UI-based approaches the accuracy remains stable, whereas for Trotterization it decreases from 10^{-7} to 10^{-4} for two and seven system parameters, respectively.

APPENDIX D: (HERMITIAN) EIGENVALUE DECOMPOSITION

In Fig. 8 we compare the matrix product approaches (Trotterizations and UI) to eigenvalue decomposition-based

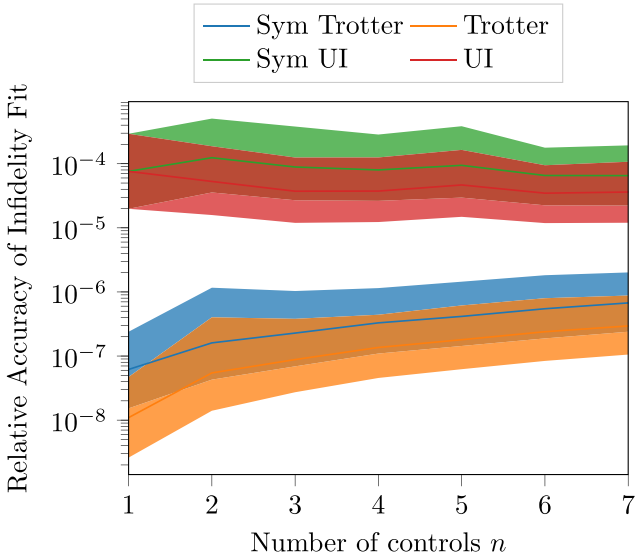

FIG. 12. The relative accuracy of the fit using a point ratio of 2.0 for Trotterization and unitary intepolation techniques as a function of the number of system parameters. The relative accuracy is computed for 100 systems with 100 sample points each. At each of the sample points $i \in \{1, \ldots, 100\}$ we calculate the difference $d_i = |I_{i,\text{fit}} - I_{i,\text{true}}|$ between the true infidelity at the point and the prediction prediction using the fit technique. The relative accuracy at each point is calculated via $r_i = \frac{d_i}{\text{mean}(I_{\text{true}})}$. Importantly, we find relative accuracies beyond 10^{-3} and higher relative accuracies in estimating the infidelity for less accurate methods.

exponentiation. These are calculated via

$$\hat{U} = \exp(-i\hat{H}) = \hat{V}\exp(-i\hat{E})\hat{V}^\dagger, \tag{D1}$$

with eigenvalues $\mathbf{E} = \text{diag}(\hat{E})$ and eigenvector matrices $\hat{V}$ determined for $\hat{H}$ via the Lapack function `zheevd` [49] for the eigenvalue decomposition of Hermitian matrices, which utilizes the optimized divide-and-conquer method.

APPENDIX E: CACHE SIZE

We want to analyze the computational demand of creating the cache and the memory requirements of the cache. We can count the number of cached matrices from the grid. For every edge $\mathbf{i} \pm \mathbf{s}_1$ along the first parameter direction, we store one $\hat{L}_{\mathbf{i}\pm\mathbf{s}_1}$ matrix, likewise we store one $\hat{R}_{\mathbf{i}\pm\mathbf{s}_n}$ matrix for every edge $\mathbf{i} \pm \mathbf{s}_n$ along the last parameter direction, leading to

$$\#\hat{L} = N_1 \prod_{i=2}^{n}(N_i + 1) \quad \text{and} \quad \#\hat{R} = N_n \prod_{i=1}^{n-1}(N_i + 1). \tag{E1}$$

For every grid point $\mathbf{i}$ we store as many $\hat{C}_{\mathbf{i},\mathbf{s}_p,\mathbf{s}_{p+1}}$ matrices as there are quadrants next to it, so that the number of cached matrices is given by the number of odd-summed grid points around a (hyper)cubic unit cell 2^{n-1} times the number of quadrants $\prod_{i=1}^{n} N_i$

$$\#\hat{C} = 2^{n-1} \prod_{i=1}^{n} N_i. \tag{E2}$$

The overall number of cached matrices is then given by

$$\begin{aligned} C(\mathbf{N}) &= \#\hat{C} + \#\hat{L} + \#\hat{R} \\ &= 2^{n-1} \prod_{i=1}^{n} N_i + N_1 \prod_{i=2}^{n}(N_i + 1) + N_n \prod_{i=1}^{n-1}(N_i + 1). \end{aligned} \tag{E3}$$

In addition we store one eigenvalue vector $\hat{E}_{\mathbf{i},\mathbf{s}_p}$ for every edge $\mathbf{i} \to \mathbf{i} + \mathbf{s}_p$ of the grid, leading to a cummulative $\#\hat{E} = \prod_{i=1}^{n} N_i$. The cache is constructed by

(1) computing the grid point unitaries $\hat{U}_\mathbf{i}$ for all grid points $\mathbf{i}$;

(2) applying an eigendecomposition to the product of the edges vertex unitaries for every edge $\mathbf{i} + \mathbf{s}_p$ of the grid as defined in Eq. (35) to obtain the eigenenergies $\hat{E}$ and eigenvectors $\hat{V}$;

(3) constructing the cached matrices $\hat{L}$, $\hat{C}$, and $\hat{R}$ as defined in Eq. (30).

APPENDIX F: OPTIMAL BINNING

In practice we wish to approximate our unitaries with guaranteed target precision I_{tar}, while requiring the least amount of caching. To facilitate this we approximate the maximum infidelity of different binnings $\mathbf{N}$ as a polynomial function of the bin widths $\Delta c_i(N_i) = \frac{c_{i,\max} - c_{i,\min}}{N_i}$, using constants derived from a test binning (for example, a $N_i = 1\,\forall i \in \{1, \ldots, n\}$). We then proceed to develop an optimization routine.

We expand the average infidelity given in Eq. (C3) of the UI into a Taylor expansion, using Eq. (F16). The leading-order corrections in the exponent of the overlap operators $\hat{M}$ are of the type

$$\begin{aligned} \hat{M} &= \hat{U}^\dagger_{\text{exact}}(\alpha)\hat{U}_{\text{UI}}(\alpha, \mathbf{N}) \\ &= \exp\left(\sum_{i,j=1}^{n} \frac{1}{N_i N_j} \alpha_i \alpha_j \hat{A}_{ij} + \mathcal{O}(N^{-3}) \right), \end{aligned} \tag{F1}$$

where the $\hat{A}_{ij}$ are linear combinations of commutator terms involving a single $\hat{H}_i$, $\hat{H}_j$ and different numbers of $\hat{H}_0$.[6] To estimate the maximum infidelity of the interpolation, we set $\alpha_i = \frac{1}{2}\,\forall i \in \{1, \ldots, n\}$. The trace $\text{Tr}(\hat{A}_{ij}) = 0$ must be zero, as all terms in $\hat{A}_{ij}$ are constructed from commutators, and commutators are traceless. Therefore in leading-order Taylor expansion the trace is given by

$$\text{Tr}(\hat{M}) = d + \sum_{\substack{i<j \\ k<l}}^{n} \frac{1}{16 N_i N_j N_k N_l} \text{Tr}(\hat{A}_{ij}\hat{A}_{kl}) + \mathcal{O}(N^{-6}). \tag{F2}$$

In Appendix F 2 we show that the traces $\text{Tr}(\hat{A}_{ij}\hat{A}_{kl})$ are dominated by terms with $i, j = k, l$, while terms $i, j \neq k, l$ are suppressed. We therefore neglect these terms in our expansion of the trace of $\hat{M}$:

$$\text{Tr}(\hat{M}) = d + \sum_{1 \leqslant i < j}^{n} \frac{1}{16 N_i^2 N_j^2} \text{Tr}\left(\hat{A}_{ij}^2\right) + \mathcal{O}(\Delta c^6). \tag{F3}$$

[6] The 1D case represents an exception, where $i = j = 1$ and the commutators involved have two $\hat{H}_i$ and at least one $\hat{H}_0$ term.

Lacking an efficient way to compute the terms $\hat{A}_{ij}$ or even just generally deriving the commutator terms, we instead approximate at the point of maximum infidelity using

$$\hat{A}_{ij} \approx \hat{U}^{\dagger}_{\text{exact}}(\alpha_i = \alpha_j = \tfrac{1}{2}, \alpha_{k\neq i,j} = 0)\hat{U}_{\text{UI}}(\alpha_i = \alpha_j = \tfrac{1}{2}, \alpha_{k\neq i,j} = 0, N_i = 1)) - \mathbb{I} - \begin{cases} 0 & i = j \\ (\hat{A}_{ii} + \hat{A}_{jj}) & i \neq j \end{cases}. \tag{F4}$$

Here we remove other terms depending only on i or j from the approximation, as they are also included in the unitaries but depend on $\frac{1}{N_i^4}$. We cache the traces $T_{ij} = 2\text{Re}[\text{Tr}(\hat{A}_{ij}^2)]$ once at the beginning of the procedure, to then repeatedly approximate the infidelity as

$$I(\mathbf{N}) = \frac{1}{d+1}\sum_{i\leqslant j}^{n} \frac{1}{N_i^2 N_j^2} T_{ij} + \mathcal{O}\left(\frac{1}{N^6}\right), \tag{F5}$$

to optimize the number of bins $\mathbf{N}$.

1. Optimization of binning

The bin optimization is divided into three steps. First, we increase the number of bins $\mathbf{N}$ to achieve an estimated infidelity below the designated threshold $I(\mathbf{N}) < I_{\text{tar}}$. We then decrease the number of bins in two steps, to minimize the number of bins $C(\mathbf{N})$ while keeping the infidelity below the target infidelity.

a. Increasing cache size to reach a target infidelity

We start with an initial binning $N_i^{(0)} = 1\,\forall i \in \{1, \ldots, n\}$. In every iteration of the optimization, we increase the bin count by one ($N_i^{(k+1)} = N_i^{(k)} + 1$) for one-parameter direction i, and keep all other directions constant $N_h^{(k+1)} = N_h^{(k)} \forall h \neq i$. For every direction $j \in \{1, \ldots, n\}$, we calculate the ratio $\mathcal{L}_j$ of the infidelity decrease to the cache size increase,

$$\mathcal{L}_j(\mathbf{N}^{(k)}) = \frac{I(\mathbf{N}^{(k)}) - I(\mathbf{N}^{(k)} + \mathbf{e}_j)}{C(\mathbf{N}^{(k)} + \mathbf{e}_j) - C(\mathbf{N}^{(k)})}, \tag{F6}$$

and choose the direction i with the largest ratio,

$$i = \text{argmax}_j \mathcal{L}_j(\mathbf{N}), \tag{F7}$$

for the increase of $N_i + 1$. The procedure ends once the estimated infidelity is below the target infidelity $I(\mathbf{N}^{(k)}) < I_{\text{tar}}$.

b. Decreasing cache size while staying below a target infidelity

In every optimization step we decrease the bin number by one in one direction $\mathbf{N}_i^{(k+1)} = \mathbf{N}_i^{(k)} - 1$, where we choose the direction i with the largest decrease in cache size, which still keeps the infidelity below the target infidelity,

$$i = \text{argmin}_i C(\mathbf{N} - \mathbf{e}_i) \quad \forall i : I(\mathbf{N} - \mathbf{e}_i) < I_{\text{tar}}. \tag{F8}$$

This procedure ends, when the current binning $\mathbf{N}$ is the smallest binning, which still keeps the infidelity below the target infidelity $I(\mathbf{N}) < I_{\text{tar}}$. In our tests, this procedure consistently found optima that were within 1% of the optimal binning, as determined by exhaustive search, while reducing the number of bins by 50%.

c. Further decrease in cache size

We further refine the previous step, by utilizing a constrained analytical solution to find the number of bins in direction m, so that the estimated infidelity $I(\mathbf{N})$ is equal to the target infidelity I_{tar}, while at the same time keeping the other binnings $N_{j\neq m}$ constant. Using Eq. (F5) and substituting both $s_i = \frac{1}{N_i^2}$ and $\frac{T_{ij}}{d(d+1))} = P_{ij}$, we can rewrite the target infidelity condition as:

$$\begin{aligned} 0 &= I(\mathbf{s}) - I_{\text{tar}} \\ &= \sum_{i<j}^{n} s_i s_j P_{ij} + \sum_{i=1}^{n} s_i^2 P_{ii} - I_{\text{tar}} \\ &= \underbrace{\sum_{\substack{i<j \\ i\neq m\neq j}}^{n} s_i s_j P_{ij} + \sum_{i\neq m}^{n} s_i^2 P_{ii} - I_{\text{tar}}}_{=I_{\text{min}} - I_{\text{tar}} = I_0} + s_m \underbrace{\sum_{j\neq m}^{n} s_j P_{mj}}_{=Q_m} + s_m^2 P_{mm}. \end{aligned} \tag{F9}$$

We solve this quadratic equation so that

$$s_{m,\pm} = -\frac{Q_m}{2P_{mm}} \pm \sqrt{\left(\frac{Q_m}{2P_{mm}}\right)^2 - \frac{I_0}{P_{mm}}}. \tag{F10}$$

This equation only has relevant solutions if $I_{\text{min}} < I_{\text{tar}} \rightarrow I_0 < 0$. We are furthermore interested in the smallest positive solution of $s_{m,\pm}$, which we call $s_{m,\text{min}}$. If such a solution exists, then determine $N_m = \frac{1}{s_{m,\text{min}}^2}$.

We can now optimize the binning via test-wise reduction of one of the bin numbers by one $N_i^{(k+1)} = N_i^{(k)} - 1$ and then calculating the optimal bin number $N_m^{(k+1)}(N_{i\neq m}^{(k+1)})$ for each of the other directions m. We then repeat this procedure for every test-wise reduction of one bin number, and choose the reduction combination (i, m), which leads to the largest decrease in cache size, while keeping the infidelity below the target infidelity. We repeat this procedure until no reductions can be found.

2. Traces of products of traceless operators

We consider the traceless operators A_{ij} and A_{kl} from Appendix F. We show that terms with $ij \neq kl$ are (statistically) significantly smaller than terms with $ij = kl$. To show this, let us expand the operators into the generalized Pauli matrices (n-qubit Pauli matrices):[7]

$$\hat{A}_{ij} = \sum_{\mu=0}^{4^n-1} a_\mu \hat{\sigma}_\mu, \quad \text{and} \quad \hat{A}_{kl} = \sum_{\nu=0}^{4^n-1} b_\nu \hat{\sigma}_\nu. \tag{F11}$$

[7]For systems that cannot be decomposed into qubits, we can simply add dimensions with zero valued entries in $\hat{A}_{ij}$ and $\hat{A}_{kl}$.

TABLE III. The accuracy of the approximation in Eq. (F5) using randomly generated and trace-orthogonalized Hamiltonians $\hat{H}_p$. The relative accuracy $\frac{I_{\text{est}}-I_{\text{exact}}}{I_{\text{exact}}}$ is estimated for different numbers of parameter terms $n \in [2, 3, 4]$ and Hilbert space dimension $s \in \{4, 16\}$, each sampled 100 times according to Appendix C 2. In one- and two-parameter dimensions the relative error is limited by machine precision. For larger n the relative errors are about one order of magnitude smaller than the values. The trace-orthogonalazied Hamiltonians behave similar to the unorthogonalized ones.

n	s	Rel. acc.	Rel. acc. (orthogonalized)
1	4	$(6.59 \pm 49.48)10^{-6}$	$(1.65 \pm 8.79)10^{-6}$
1	16	$(7.87 \pm 9.54)10^{-9}$	$(1.06 \pm 1.50)10^{-8}$
2	4	$(3.31 \pm 4.82)10^{-8}$	$(4.67 \pm 8.05)10^{-8}$
2	16	$(1.92 \pm 0.95)10^{-8}$	$(1.64 \pm 0.95)10^{-8}$
3	4	$(2.71 \pm 2.75)10^{-1}$	$(2.25 \pm 2.24)10^{-1}$
3	16	$(6.42 \pm 4.72)10^{-2}$	$(7.24 \pm 5.47)10^{-2}$
4	4	$(3.37 \pm 3.04)10^{-1}$	$(2.81 \pm 2.55)10^{-1}$
4	16	$(9.74 \pm 7.69)10^{-2}$	$(8.32 \pm 5.91)10^{-2}$

Here the particular indexing of the Pauli matrices is not important, as long as the same indexing is used for both operators. We find that the trace of the product of the two operators is given by

$$\mathrm{Tr}(\hat{A}_{ij}\hat{A}_{kl}) = \sum_{\mu,\nu=0}^{4^n-1} a_\mu b_\nu \mathrm{Tr}(\hat{\sigma}_\mu\hat{\sigma}_\nu) = \sum_{\mu=0}^{4^n-1} a_\mu b_\mu \mathrm{Tr}\big(\hat{\sigma}_\mu^2\big), \quad \text{(F12)}$$

where we used the fact that the trace of the product of two Pauli matrices is zero $\mathrm{Tr}(\hat{\sigma}_\mu\hat{\sigma}_\nu) = d\delta_{\mu,\nu}$. We therefore conclude that

$$\mathrm{Tr}(\hat{A}_{ij}\hat{A}_{kl}) = 2d\|\mathbf{a}\|\|\mathbf{b}\|\cos(\theta). \quad \text{(F13)}$$

For diagonal terms $ij = kl$ we have $\cos(\theta) = 1$, so that $\mathrm{Tr}(A_{ij}A_{ij}) = d\|\mathbf{a}\|^2$, while for off-diagonal terms $ij \neq kl$ the result will depend on the angle between the matrices. We assume that the Pauli vectors $\mathbf{a}$ and $\mathbf{b}$ are randomly distributed In Table III we show this to also be a valid approximation for trace-orthogonal Hamiltonian terms $\mathrm{Tr}(\hat{H}_i\hat{H}_j) = 0\,\forall i \neq j$. The cosine of the angle $\cos(\theta)$ between two random vectors $\mathbf{a}$ and $\mathbf{b}$ isotropically sampled from the unit (hyper)sphere, has the following n-dimensional isotropic distribution [50]:

$$p(\cos\theta) = \frac{\Gamma\left(\frac{n}{2}\right)}{\sqrt{\pi}\,\Gamma\left(\frac{n-1}{2}\right)}(1-\cos^2\theta)^{\frac{n-3}{2}} \quad \text{for } -1 \leqslant \cos\theta \leqslant 1, \quad \text{(F14)}$$

with expectation value $\mathbb{E}[\cos(\theta)] = 0$ and variance $\mathrm{Var}[\cos(\theta)] = \mathbb{E}[\cos(\theta)^2] = \frac{1}{d^2}$. We therefore conclude that the trace of the off-diagonal terms will be $\mathbb{E}[\mathrm{Tr}(A_{ij}A_{kl})] = 0$ with variance $\mathrm{Var}[\mathrm{Tr}(A_{ij}A_{kl})] = \frac{1}{d^2}\|\mathbf{a}\|\|\mathbf{b}\|$—see Eq. (F13).

3. BCH expansion of unitary interpolations

a. One-parametric unitary interpolation

The BCH expansion of the one-parametric UI to third order reveals the leading undesired terms

$$\hat{U}_{\mathrm{UI}}(\alpha) = \exp\left(-i\left(\overbrace{\hat{H}_0 + \left(c_0 + \alpha\frac{\Delta c}{N}\right)\hat{H}_1}^{\text{Desired Terms}} - \overbrace{\frac{\Delta c^2}{12N^2}(\alpha^2-\alpha)[[\hat{H}_0,\hat{H}_1],\hat{H}_1] + \mathcal{O}\left(\frac{\Delta c^2}{N^2}\right)}^{\text{Undesired Terms}}\right)\right). \quad \text{(F15)}$$

b. Multiparametric unitary interpolation

Similarly the BCH expansion for the multiparametric UI reveals

$$\hat{U}_{\mathrm{UI}}(\boldsymbol{\alpha}) = \exp\left(-i\left(\overbrace{\hat{H}_0 + \sum_{m=1}^{n}\frac{c_k}{N_k}\hat{H}_m}^{\text{Exact Hamiltonian}}\right) + \overbrace{\frac{1}{2}\sum_{1\leqslant p<q}^{n}\alpha_p\alpha_q\frac{\Delta c_p\Delta c_q}{N_pN_q}[\hat{H}_p,\hat{H}_q] + \mathcal{O}\left(\frac{\Delta c^2}{N^2}\right)}^{\text{Corrections}}\right). \quad \text{(F16)}$$

APPENDIX G: BCH EXPANSIONS OF TROTTERIZATIONS

1. Suzuki-Trotter

The BCH expansion of a Hamiltonian as the one defined in Eq. (1) into N Trotter steps, of an n-parameter problem, reveals

$$\begin{aligned}\hat{U}_{\mathrm{Tr}}(\mathbf{c}) &= \left(\left(\prod_{p=1}^{n}\exp\left(-i\frac{\hat{H}_p}{N}c_p\right)\right)\exp\left(-i\frac{\hat{H}_0}{N}\right)\right)^N \\ &= \exp\left(-i\left(\hat{H}_0 + \sum_{p=1}^{n}c_p\hat{H}_p\right) - \frac{1}{2N}\left(\sum_{p=1}^{n}c_p[\hat{H}_0,\hat{H}_p] + \sum_{1\leqslant p<q}^{n}c_pc_q[\hat{H}_p,\hat{H}_q]\right) + \mathcal{O}\left(\frac{c}{N}\right)\right)\right). \quad \text{(G1)}\end{aligned}$$

Each of the parametrized exponentials can be eigendecomposed (as in the UI) as $\hat{V}_i\hat{E}_i\hat{E}_i^\dagger = \hat{H}_i$. The product of the adjacent eigenvector matrices in the construction of the expression in Eq. (G1) can be precomputed $\hat{C}_i = \hat{V}_{i+1}^\dagger\hat{V}_i$ and cached together with the eigenvalues $\hat{E}_i$. Hence, a single Trotter step requires the same number of matrix multiplications as the UI. $N = 2^j$ Trotter steps can be computed from a single Trotter step $\hat{U}_{\mathrm{Tr}}$ by repeated squaring $\hat{U}_{\mathrm{Tr}}^{2^j} = \hat{U}_{\mathrm{Tr}}^{2^{j-1}}\hat{U}_{\mathrm{Tr}}^{2^{j-1}}$.

2. Symmetric Trotter (Strang or split-step)

The symmetrized split-step variant of the Trotter method is given by

$$\begin{aligned}\hat{U}_{\mathrm{St}}(c) &= \left(\left(\prod_{p=1}^{n}\exp\left(-i\frac{\hat{H}_p}{2N}c_p\right)\right)\exp\left(-i\frac{\hat{H}_0}{N}\right)\left(\coprod_{p=1}^{n}\exp\left(-i\frac{hH_p}{2N}c_p\right)\right)\right)^N\\ &= \exp\Bigg(-i\Bigg(\hat{H}_0+\sum_{p=1}^{n}c_p\hat{H}_p+\sum_{p=1}^{n}\frac{c_p}{12N^2}\Big([[\hat{H}_0,\hat{H}_p],\hat{H}_0]+\frac{c_p}{2}[[\hat{H}_0,\hat{H}_p],\hat{H}_p]\Big)\\ &\quad+\sum_{1\leqslant p<q}^{n}\frac{c_pc_q}{12N^2}\Big(\frac{c_q}{2}[[\hat{H}_p,\hat{H}_q],\hat{H}_q]+c_p[[\hat{H}_p,\hat{H}_q],\hat{H}_p]\Big)\\ &\quad+\sum_{0\leqslant p<q<r}^{n}\frac{c_pc_qc_r}{12N^2}([[\hat{H}_p,\hat{H}_r],\hat{H}_q]+[[\hat{H}_q,\hat{H}_r],\hat{H}_p])+\mathcal{O}\left(\frac{c}{N^2}\right)\Bigg)\Bigg),\end{aligned}\tag{G2}$$

where the coproduct sign $\coprod_{p=1}^{n}$ denotes the reversed order of the product. This adds $n-1$ additional matrix multiplication over Trotterization and UI for the construction of a single symmetric Trotter step, while the repeated squaring remains the same.

APPENDIX H: SYMMETRIC UNITARY INTERPOLATION

In the main part of the manuscript we also compared the UI with a symmetrized version of the UI. As the additional accuracy provided by this method did not justify the additional computational cost, we did not include its derivation in the main text. Here we quickly introduce the method. We use the indexing and interpolation parameters from Sec. V D. We can generalize the UI in analogy to the split step approach by symmetrization. This requires half-step unitaries on the interpolation grid:

$$\tilde{U}_{\frac{\mathbf{i}}{2}} = \exp\left(-\frac{i}{2}\left(\hat{H}_0+\sum_{p=1}^{n}\left(c_{p,\min}+i_p\frac{\Delta c_p}{N_p}\right)\hat{H}_p\right)\right).\tag{H1}$$

The interpolation is then constructed as

$$\hat{U}_{\mathrm{UI},S}(\mathbf{c},\boldsymbol{\alpha}(\mathbf{c})) = \tilde{U}_{\frac{\mathbf{i}}{2}}\left(\prod_{p=2}^{n}\left(\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger\tilde{U}_{\frac{\mathbf{i}+\mathbf{s}_p}{2}}\right)^{|\alpha_p|}\right)\left(\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger\tilde{U}_{\frac{\mathbf{i}+\mathbf{s}_1}{2}}\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger\right)^{|\alpha_1|}\left(\coprod_{p=2}^{n}\left(\tilde{U}_{\frac{\mathbf{i}+\mathbf{s}_p}{2}}\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger\right)^{|\alpha_p|}\right)\tilde{U}_{\frac{\mathbf{i}}{2}}.\tag{H2}$$

Here the $\coprod$ symbol denotes the product of the matrices in the reverse order (from the right side). The lattice indices $\mathbf{i}$ and interpolation parameters $\boldsymbol{\alpha}$ are calculated analogously to the case of the UI in Eq. (32). This product can be constructed from cached matrices which are derived from the logarithmic decomposition of the products $(\tilde{U}_{\frac{\mathbf{i}+\mathbf{s}_p}{2}}\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger) = \hat{L}_{\mathbf{i},\mathbf{s}_p}\exp(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p})\hat{L}_{\mathbf{i},\mathbf{s}_p}^\dagger$ and their counterparts $(\tilde{U}_{\frac{\mathbf{i}+\mathbf{s}_p}{2}}\tilde{U}_{\frac{\mathbf{i}}{2}}^\dagger) = \hat{R}_{\mathbf{i},\mathbf{s}_p}^\dagger\exp(-i\hat{E}_{\mathbf{i}+\mathbf{s}_p})\hat{R}_{\mathbf{i},\mathbf{s}_p}$. As a result, we have $\hat{C}^{(L)}_{\mathbf{i},\mathbf{s}_{p+1},\mathbf{s}_p} = \hat{L}^\dagger_{\mathbf{i},\mathbf{s}_{p+1}}\hat{L}_{\mathbf{i},\mathbf{s}_p}$ and $\hat{C}^{(R)}_{\mathbf{i},\mathbf{s}_{p+1},\mathbf{s}_p} = \hat{R}_{\mathbf{i},\mathbf{s}_p}\hat{R}^\dagger_{\mathbf{i},\mathbf{s}_{p+1}}$, respectively, doubling the number of central cached matrices in Eq. (E2).

By repeatedly applying the BCH expansion, we have

$$\begin{aligned}\hat{U}_{\mathrm{UI},S}(\mathbf{c},\boldsymbol{\alpha}(\mathbf{c})) = \exp\Bigg(-i\Bigg(&\hat{H}_0-\sum_{p=1}^{n}c_p\hat{H}_p-\frac{1}{48}\sum_{p=1}^{n}\frac{\Delta c_p^2}{N_p^2}(|\alpha_p|^2-|\alpha_p|)[[\hat{H}_0,\hat{H}_p],\hat{H}_p]\\ &-\frac{1}{24}\sum_{1\leqslant p<q}^{n}|\alpha_p||\alpha_q|\frac{\Delta c_p\Delta c_q}{N_pN_q}(2[[\hat{H}_0,\hat{H}_p],\hat{H}_q]-[[\hat{H}_0,\hat{H}_q],\hat{H}_p])\\ &-\frac{1}{48}\sum_{1\leqslant p<q}^{n}\left(d_q\Delta\frac{c_p^2}{N_p^2}(|\alpha_p|-|\alpha_p|^2)-2|\alpha_q|\frac{\Delta c_q}{N_q}\left(d_p|\alpha_p|\frac{\Delta c_p}{N_p}+2\frac{(|\alpha_p|\Delta c_p)^2}{N_p^2}\right)\right)[[\hat{H}_p,\hat{H}_q],\hat{H}_p]\\ &+\frac{1}{48}\sum_{1\leqslant p<q}^{n}\left(d_p\Delta\frac{c_q^2}{N_q^2}(|\alpha_q|-|\alpha_q|^2)+2|\alpha_p|\frac{\Delta c_p}{N_p}\left(2d_q|\alpha_q|\frac{\Delta c_q}{N_q}+\frac{(|\alpha_q|\Delta c_q)^2}{N_q^2}\right)\right)[[\hat{H}_p,\hat{H}_q],\hat{H}_q]\end{aligned}$$

$$
\begin{aligned}
&- \frac{1}{12} \sum_{1\leqslant p<q<r}^{n} |\alpha_r| \frac{\Delta c_r}{N_r} \left(|\alpha_p| \frac{\Delta c_p}{N_p} d_q - d_p |\alpha_q| \frac{\Delta c_q}{N_q} - \alpha_p \frac{\Delta c_p}{N_p} |\alpha_q| \frac{\Delta c_q}{N_q} \right) [[\hat{H}_p, \hat{H}_q], \hat{H}_r] \\
&+ \frac{1}{24} \sum_{1\leqslant p<q<r}^{n} |\alpha_q| \frac{\Delta c_q}{N_q} \left(d_p |\alpha_r| \frac{\Delta c_r}{N_r} + 2 |\alpha_p| \frac{\Delta c_p}{N_p} d_r \right) [[\hat{H}_p, \hat{H}_r], \hat{H}_q] \\
&- \frac{1}{24} \sum_{1\leqslant p<q<r}^{n} |\alpha_p| \frac{\Delta c_p}{N_p} \left(d_q |\alpha_r| \frac{\Delta c_r}{N_r} - |\alpha_q| \frac{\Delta c_q}{N_q} d_r \right) [[\hat{H}_q, \hat{H}_r], \hat{H}_p] + \mathcal{O}\left(\frac{\Delta c^2}{N^2} \right) \Bigg) \Bigg),
\end{aligned}
\tag{H3}
$$

where we used $d_p = c_{p,\min} + i_p \frac{\Delta c_p}{N_p}$. As in the case of symmetric Trotterization, we obtain a method with leading-order infidelity $I \propto \mathcal{O}(\frac{1}{N^2})$. The new approximation only outperforms UI by a constant ratio, due to the presence of better coefficients in the expansion.

[1] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[2] I. M. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, Rev. Mod. Phys. **86**, 153 (2014).

[3] C. P. Koch, U. Boscain, T. Calarco, G. Dirr, S. Filipp, S. J. Glaser, R. Kosloff, S. Montangero, T. Schulte-Herbrüggen, D. Sugny, and F. K. Wilhelm, Quantum optimal control in quantum technologies: Strategic report on current status, visions and goals for research in europe, EPJ Quantum Technol. **9**, 19 (2022).

[4] E. A. Martinez, C. A. Muschik, P. Schindler, D. Nigg, A. Erhard, M. Heyl, P. Hauke, M. Dalmonte, T. Monz, P. Zoller, and R. Blatt, Real-time dynamics of lattice gauge theories with a few-qubit quantum computer, Nature (London) **534**, 516 (2016).

[5] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, Quantum-assisted quantum compiling, Quantum **3**, 140 (2019).

[6] F. Preti, M. Schilling, S. Jerbi *et al.*, Hybrid discrete-continuous compilation of trapped-ion quantum circuits with deep reinforcement learning, Quantum **8**, 1343 (2024).

[7] H. F. Trotter, On the product of semi-groups of operators, Proc. Am. Math. Soc. **10**, 545 (1959).

[8] F. J. Dyson, The *s* matrix in quantum electrodynamics, Phys. Rev. **75**, 1736 (1949).

[9] W. Magnus, On the exponential solution of differential equations for a linear operator, Commun. Pure Appl. Math. **7**, 649 (1954).

[10] M. Dalgaard and F. Motzoi, Fast, high precision dynamics in quantum optimal control theory, J. Phys. B: At., Mol. Opt. Phys. **55**, 085501 (2022).

[11] F. Fer, Résolution de l'équation matricielle $dU/dt = pU$ par produit infini d'exponentielles matricielles, Bull. Acad. r. Belg **44**, 818 (1958).

[12] C. Runge, Ueber die numerische auflösung von differentialgleichungen, Math. Ann. **46**, 167 (1895).

[13] W. Kutta, Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, Zeit. Math. Phys. **46**, 435 (1901).

[14] M. Hochbruck and C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. **34**, 1911 (1997).

[15] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms, J. Magn. Reson. **172**, 296 (2005).

[16] F. Motzoi, J. M. Gambetta, S. T. Merkel, and F. K. Wilhelm, Optimal control methods for rapidly time-varying Hamiltonians, Phys. Rev. A **84**, 022307 (2011).

[17] P. de Fouquieres, S. Schirmer, S. Glaser, and I. Kuprov, Second order gradient ascent pulse engineering, J. Magn. Reson. **212**, 412 (2011).

[18] O. V. Morzhin and A. N. Pechen, Krotov method for optimal control of closed quantum systems, Russian Math. Surveys **74**, 851 (2019).

[19] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework, Phys. Rev. A **84**, 022305 (2011).

[20] T. Caneva, T. Calarco, and S. Montangero, Chopped random-basis quantum optimization, Phys. Rev. A **84**, 022326 (2011).

[21] N. Rach, M. M. Müller, T. Calarco, and S. Montangero, Dressing the chopped-random-basis optimization: A bandwidth-limited access to the trap-free landscape, Phys. Rev. A **92**, 062343 (2015).

[22] M. M. Müller, R. S. Said, F. Jelezko, T. Calarco, and S. Montangero, One decade of quantum optimal control in the chopped random basis, Rep. Prog. Phys. **85**, 076001 (2022).

[23] F. Preti, T. Calarco, and F. Motzoi, Continuous quantum gate sets and pulse-class meta-optimization, PRX Quantum **3**, 040311 (2022).

[24] E. Onorati, O. Buerschaper, M. Kliesch, W. Brown, A. H. Werner, and J. Eisert, Mixing properties of stochastic quantum Hamiltonians, Commun. Math. Phys. **355**, 905 (2017).

[25] D. Ceperley and B. Alder, Quantum Monte Carlo, Science **231**, 555 (1986).

[26] B. M. Austin, D. Y. Zubarev, and W. A. Lester Jr., Quantum Monte Carlo and related approaches, Chem. Rev. **112**, 263 (2012).

[27] Y. Saad, *Numerical Methods for Large Eigenvalue Problems* (Manchester University Press, Manchester, UK, 1992), p. 346.

[28] A. Brinkmann, Introduction to average Hamiltonian theory. I. Basics, Concepts Magn. Reson. A **45A**, e21414 (2016).

[29] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[30] B. C. Hall, The Baker–Campbell–Hausdorff formula, *Lie Groups, Lie Algebras, and Representations* (Springer, New York, NY, 2003), pp. 63–90.

[31] G. Strang, On the construction and comparison of difference schemes, SIAM J. Numer. Anal. **5**, 506 (1968).

[32] R. Fletcher, *Practical Methods of Optimization* (Wiley & Sons, New York, NY, 2013), p. 248.

[33] M. Dalgaard, C. A. Weidner, and F. Motzoi, Dynamical uncertainty propagation with noisy quantum parameters, Phys. Rev. Lett. **128**, 150503 (2022).

[34] M. Dalgaard, F. Motzoi, Jesper Hasseriis Mohr Jensen, and J. Sherson, Hessian-based optimization of constrained quantum control, Phys. Rev. A **102**, 042612 (2020).

[35] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[36] J. Nocedal, Updating quasi-newton matrices with limited storage, Math. Comput. **35**, 773 (1980).

[37] M. H. Goerz, S. C. Carrasco, and V. S. Malinovsky, Quantum optimal control via semi-automatic differentiation, Quantum **6**, 871 (2022).

[38] N. Leung, M. Abdelhafez, J. Koch, and D. Schuster, Speedup for quantum optimal control from automatic differentiation based on graphics processing units, Phys. Rev. A **95**, 042318 (2017).

[39] I. Kuprov and C. T. Rodgers, Derivatives of spin dynamics simulations, J. Chem. Phys. **131**, 234108 (2009).

[40] A. S. Lewis and H. S. Sendov, Twice differentiable spectral functions, SIAM J. Matrix Anal. Appl. **23**, 368 (2001).

[41] G. Voronoi, Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs., J. Reine Angew. Math. **1908**, 198 (1908).

[42] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey *et al.*, SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nat. Methods **17**, 261 (2020).

[43] A. H. Al-Mohy and N. J. Higham, Computing the action of the matrix exponential, with an application to exponential integrators, SIAM J. Sci. Comput. **33**, 488 (2011).

[44] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, Cython: The best of both worlds, Comput. Sci. Eng. **13**, 31 (2011).

[45] https://github.com/Ntropic/unipolator.

[46] C. Dankert, Efficient simulation of random quantum states and operators, arXiv:quant-ph/0512217.

[47] L. H. Pedersen, N. M. Møller, and K. Mølmer, Fidelity of quantum operations, Phys. Lett. A **367**, 47 (2007).

[48] M. E. O'Neill, *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*, Technical Report HMC-CS-2014-0905 (Harvey Mudd College, Claremont, CA, 2014).

[49] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK User Guide*, 3rd ed. (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999).

[50] T. Cai, J. Fan, and T. Jiang, Distributions of angles in random packing on spheres, arXiv:1306.0256.