

Lie-algebra-assisted quantum simulation and quantum optimal control via high-order Magnus expansions

R. F. dos Santos  and S. J. J. M. F. Kokkelmans 

Department of Applied Physics, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands



(Received 24 December 2025; accepted 20 August 2026; published 8 September 2026)

The evolution of a quantum system under time-dependent driving exhibits phenomena that are absent in its stationary counterpart. However, the high dimensionality and noncommutative nature of quantum dynamics make this a challenging problem. The Magnus expansion provides an analytic framework to approximate the effective dynamics on short timescales, but computing high-order terms with existing methods is computationally expensive. We introduce a scalable approach for Hamiltonians consisting of a time-independent drift term and a controllable term, in which the high-order Magnus expansion is reduced to a polynomial expression in the duration and the parameters of the time-dependent control function. For a fixed Hamiltonian model, the demanding algebraic calculation is performed once and stored in dynamical coefficients, after which the effective Hamiltonian can be evaluated numerically several orders of magnitude faster than previous techniques. We first demonstrate the method as a proof of principle for quantum simulation, verifying the expected high-order error scaling and benchmarking the numerical efficiency of the polynomial representation. We then apply the same analytically differentiable framework to a state-of-the-art quantum optimal-control problem, designing continuous control pulses for multiqubit phase gates on a neutral-atom Rydberg platform. Our results show that this method provides a practical route toward optimal control of challenging driven quantum systems.

DOI: [10.1103/jmnrn-t4jw](https://doi.org/10.1103/jmnrn-t4jw)

I. INTRODUCTION

Quantum dynamics underpins the development of emerging technologies, including quantum computation [1,2], sensing [3–5], and communication [6], as well as serving as a test bed for fundamental physics [7–10]. Devices exploiting quantum phenomena are now beginning to appear, with quantum computers being a prominent example. These versatile platforms enable experimental validation of quantum information theory, encompassing topics such as quantum control [11–13], entanglement generation [14], ground-state preparation [15,16], and quantum error correction [17–20]. Yet, in the current noisy intermediate-scale quantum (NISQ) era, available qubits remain scarce and highly susceptible to environmental noise, limiting practical advantages over classical approaches [21,22]. Consequently, the design of efficient quantum operations from inherently complex quantum dynamics is critical for advancing quantum computing and overcoming the challenges of this era.

Neutral-atom quantum computing platforms are especially interesting as a toolbox for studying complex quantum dynamics [23–27]. Since the interatomic interactions have a strong dependence on distance, different geometries serve as test beds to simulate quantum materials and study the associated properties [16,26,27]. Coherent rearrangeability

through optical means is another prominent feature, allowing for flexible quantum circuits and quantum error correction demonstrations [20,28].

Harnessing the intricate dynamics of quantum systems demands operations that are both precise and coherence preserving. Fundamental tasks in quantum computing—such as state preparation and gate synthesis—rely on engineered control pulses to shape the underlying physical evolution. Yet, the noncommutative nature of quantum dynamics often precludes closed-form analytical solutions, making simulation and optimization necessary for designing and understanding these operations on real devices. Lie groups and algebras have been found to be effective tools for describing and studying the noncommutative nature of quantum dynamics [11,29–35].

The Magnus expansion (ME) is an exponential representation of the solution operator of the quantum system that emerges from the study of Lie algebras [36,37]. It is a series expansion that approximates the solution operator of the time-dependent quantum system. This solution operator is represented as a constant Hamiltonian, often referred to as the effective Hamiltonian, and it evolves the quantum system the same way as if evolved through the time-dependent dynamics. The Magnus expansion is also often used as a numerical integrator due to its convenient property of guaranteeing state unitarity, even at low orders [36]. However, evaluating the expansion using standard techniques is unfeasible in practice, as they rely on high-order time integrals of commutators of the time-dependent Hamiltonian. The majority of studies that make use of the Magnus expansion limit themselves to at most order four, due to the numerical limitations [38,39].

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

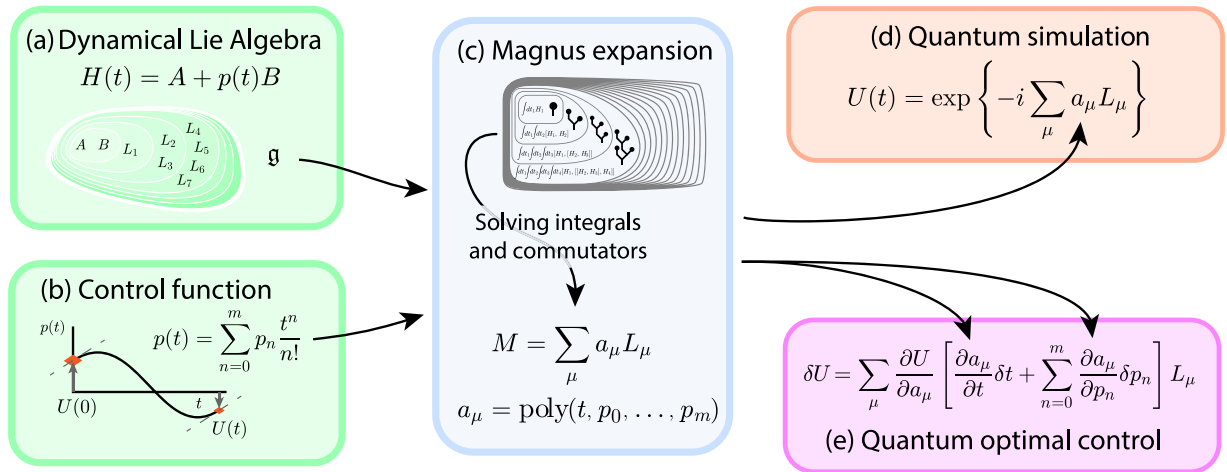


FIG. 1. In this work we propose a method to efficiently evaluate the Magnus expansion. (a) Diagram of the generation of the dynamical Lie algebra that stems from the single-control Hamiltonian terms A and B . (b) The control function is represented as a polynomial with maximum degree m , which allow us to solve high-order integrals analytically. (c) The high-order integrals and commutators from the Magnus expansion are simplified into a polynomial expression. We use $\text{poly}(x_1, \dots, x_n)$ to denote a generic polynomial with complex-valued coefficients and corresponding monomials in the $(x_1, \dots, x_n)$ variables. (d) This enables high-order simulation of continuous-control quantum systems. (e) The analytically differentiable expression enables quantum optimal-control applications.

One of the first studies that used the ME as a numerical integrator was performed in Ref. [40]. It validates the error scaling of the ME approach by comparing with typical Runge-Kutta integrators, reaching expansion order up to four. More recent works are able to approximate the evolution under the Magnus expansion using commutator-free methods, achieving at most order-eight methods [38,41,42]. Finally, the authors in Ref. [43] were able to find a method to efficiently express the continuous Baker-Campbell-Hausdorff formula [44,45], a particular instance of the Magnus expansion, in terms of the chained integrals up to order eight. However, they rely on intricate analytic integral expressions, the computation of which does not scale favorably for higher orders, and a systematic way to evaluate them is not provided.

The study of Floquet systems revolves around interpreting the dynamics of a time-periodic Hamiltonian as an effective (constant) Hamiltonian [46–50]. Consequently, the Magnus expansion naturally becomes a fundamental tool in this analysis, being called in this setting the Floquet-Magnus expansion [37]. The study of Floquet dynamics is crucial for the development of quantum technologies as it is often present in the study of quantum materials [51–53]. However, the complexity of the ME limits these studies to only a few orders, generally up to four.

The central message of this work is that the difficult part of the high-order Magnus expansion can be separated from the repeated numerical evaluation of commutators of the control Hamiltonian. For a fixed Hamiltonian model, the Lie-algebraic and integral structure is computed once and stored in a set of dynamical coefficients. Thereafter, evaluating the effective Hamiltonian reduces to evaluating a polynomial in the pulse duration and control parameters. We first use quantum simulation as a proof of principle to verify the accuracy, scaling, and numerical efficiency of this construction.

Next we show that the same differentiable representation can be used directly in a demanding optimal-control problem. Since the resulting expressions are analytically differentiable,

they can be readily incorporated into optimal quantum control algorithms. Related work has employed the Magnus expansion in control applications [54,55]. These approaches typically consider a single propagation step and rely on low-order truncations of the Magnus series, thereby restricting the class of dynamics that can be accessed. In contrast, the framework developed here allows for the systematic inclusion of higher-order terms and multiple Magnus propagation steps, extending the applicability of Magnus-based control methods to more complex dynamical regimes. It corresponds to an extension of gradient-ascent pulse engineering (GRAPE) [56] for spline-shaped pulses, using the truncated Magnus expansion. In this extension we optimize the pulse duration directly with the fidelity. This work opens the way for alternative approaches in pulse design for multiqubit controls using Lie-algebraic techniques.

The article is structured as follows: In Sec. II we introduce the Magnus expansion. As a precursor to the method, we introduce the dynamical Lie algebra and consider the control function to be of polynomial form, corresponding to Figs. 1(a) and 1(b), respectively. This leads to the main result of this work in Sec. III, displayed in the central box [Fig. 1(c)] in the diagram. In Sec. IV, we look at the application of our method to quantum simulation [Fig. 1(d)], study the error scaling with increasing orders, and argue for the high scalability of the method through numerical benchmarking. In Sec. V, we showcase an optimal control application for a Rydberg-atom-based system, displayed in Fig. 1(e). Finally, in Sec. VI we conclude and discuss an outlook for future work.

II. THE MAGNUS EXPANSION

Coherent quantum systems evolve according to the time-dependent Schrödinger equation [57],

$$\frac{d}{dt}|\psi(t)\rangle = -iH(t)|\psi(t)\rangle, \quad (1)$$

where the Hamiltonian $H(t)$ propagates the state of the system. The unitary state propagator that evolves the state from t_0 to t_f is

$$U(t_f, t_0) = \mathcal{T} \exp \left\{ -i \int_{t_0}^{t_f} ds H(s) \right\}, \quad (2)$$

where $\mathcal{T}$ is the time-ordering operator [58]. In this representation, $U(t_f, t_0)$ is the propagator, encoding the evolution of an initial state $|\psi_0\rangle = |\psi(t_0)\rangle$ into $|\psi(t_f)\rangle = U(t_f, t_0)|\psi_0\rangle$.

The unitary $U(t_f, t_0)$ can be interpreted as the propagator of a constant-in-time evolution of an effective Hamiltonian $H_E(t_f, t_0)$:

$$U(t_f, t_0) = \exp\{-i H_E(t_f, t_0)(t_f - t_0)\}. \quad (3)$$

The ME provides a constructive approach for calculating

$$H_E(t_f, t_0) = \frac{1}{t_f - t_0} \lim_{k_M \rightarrow \infty} M^{(k_M)}(t_f, t_0), \quad (4)$$

as long as the corresponding series converges [36]. The series expansion takes the form

$$M^{(k_M)}(t_f, t_0) := \sum_{k=1}^{k_M} M_k(t_f, t_0), \quad (5)$$

with

$$M_1(t_f, t_0) := \int_{t_0}^{t_f} dt_1 H(t_1), \quad (6)$$

$$M_n(t_f, t_0) := (-i)^{n-1} \sum_{j=1}^{n-1} \frac{B_j}{j!} \int_{t_0}^{t_f} dt_1 S_n^{(j)}(t_1), \quad n \geq 2, \quad (7)$$

where B_j is the j th Bernoulli number [59]. The $S_n^{(j)}(t)$ operators are defined recursively as

$$S_n^{(1)}(t) := [M_{n-1}(t, t_0), H(t)], \quad (8)$$

$$S_n^{(n-1)}(t) := \text{ad}_{M_1(t)}^{n-1}(H(t)), \quad (9)$$

$$S_n^{(j)}(t) := \sum_{m=1}^{n-j} [M_m(t, t_0), S_{n-m}^{(j-1)}(t)], \quad 2 \leq j \leq n-1, \quad (10)$$

with the adjoint operator $\text{ad}_A^1(B) = [A, B]$, and $\text{ad}_A^n(B) = [A, \text{ad}_A^{n-1}(B)]$. For illustrative purposes, the first three orders of the Magnus expansion are shown below as time integrals of commutators of $H(t)$ for varying values of t , where $M_k := M_k(t_f, t_0)$:

$$M_1 = \int_{t_0}^{t_f} dt_1 H(t_1), \quad (11)$$

$$M_2 = -\frac{i}{2} \int_{t_0}^{t_f} dt_1 \int_{t_0}^{t_1} dt_2 [H(t_1), H(t_2)], \quad (12)$$

$$M_3 = -\frac{1}{6} \int_{t_0}^{t_f} dt_1 \int_{t_0}^{t_1} dt_2 \int_{t_0}^{t_2} dt_3 ([H(t_1), [H(t_2), H(t_3)]] \\ + [H(t_3), [H(t_2), H(t_1)]]) \quad (13)$$

The Magnus expansion approximates the effective Hamiltonian $H_E(t_f, t_0)$ as long as the series converges. A necessary

and sufficient condition for convergence is given by

$$\int_0^t ds \|H(s - t_0)\|_2 < r_C = \pi, \quad (14)$$

where $\|\cdot\|_2$ is the operator spectral norm and r_C is the convergence radius of the Magnus expansion [60]. For a given time-dependent Hamiltonian, the convergence time is the largest time, denoted t_* , where the above inequality holds. Furthermore, the ME is time symmetric, i.e.,

$$M(t_f, t_0) = -M(t_0, t_f), \quad (15)$$

which is an important property for numerical integrators, as it guarantees the preservation of geometric quantities, such as state unitarity. It is often considered $t := t_f, t_0 = 0$, thus writing $M(t) := M(t_f, t_0)$. In numerical studies, the series must be truncated at some order k_M :

$$M^{(k_M)}(t) := \sum_{k=1}^{k_M} M_k(t). \quad (16)$$

Due to the time symmetry of the series, the truncation error at even orders $k_M = 2s, s \in \mathbb{N}$, is given by

$$\|M^{(2s)}(t) - H_E(t)\| = O(t^{2s+3}). \quad (17)$$

For more details on the Magnus expansion, we refer the reader to the review paper in Ref. [36].

III. POLYNOMIAL EXPRESSION FOR THE MAGNUS EXPANSION

We consider the case of Hamiltonian control models consisting of a time-independent, noncontrollable term A and a single controllable term B ,

$$H(t) = A + p(t)B, \quad (18)$$

where both A and B are Hermitian operators. This family of Hamiltonians is often considered in quantum simulation or control problems, with the physical systems ranging from neutral atoms [25] to superconducting circuits [61].

We aim to find the effective Hamiltonian $H_E(t)$ that is generated from the time-dependent dynamics of $H(t)$. The Magnus expansion provides a way to do this for $t < t_*$, but it requires the calculation of high-order integrals involving commutators of $H(t)$, similar to the ones seen in Eqs. (11)–(13). The direct calculation is not feasible in practice, and so it is necessary to find alternative ways to obtain H_E .

The control function $p(t)$ is represented as a polynomial with maximum degree m ,

$$p(t) := \sum_{n=0}^m p_n \frac{t^n}{n!}, \quad (19)$$

where p_γ are real-valued coefficients of the polynomial $p(t)$ and $\vec{p} = (p_0, \dots, p_m)$ is its vectorial representation. This finite representation of the control function as a polynomial in time is useful as it allows an analytical expression for the high-order time integrals involved in the Magnus expansion. As long as the order of the polynomial, m , is greater than the truncation order of the ME, the dynamics resulting from an arbitrary control function and from its polynomial approximation will be practically equal. This allows us to accurately

represent all realistic control functions. This is represented in Fig. 1(b).

The next elements of the calculation are the high-order commutators. The effective Hamiltonian generated from the time-dependent dynamics of $H(t)$ belongs to the dynamical Lie algebra which is generated from the operators A and B [30]. The dynamical Lie algebra $\mathfrak{g}$ is a vector space that is closed under the Lie bracket $[A, B] = AB - BA$, with

$$\mathfrak{g} := i \text{span}_{\mathbb{R}} \langle \{iA, iB\} \rangle_{\text{Lie}} = \text{span}_{\mathbb{R}} \{L_{\mu}\}_{\mu=1}^{\dim(\mathfrak{g})}. \quad (20)$$

The Lie closure $\langle \cdot \rangle_{\text{Lie}}$ is the set of elements obtained by repeatedly applying the Lie bracket until no new linearly independent elements are obtained [30]. The set $\{L_{\mu}\}_{\mu=1}^{\dim(\mathfrak{g})}$ is a basis of independent Hermitian elements of the Lie algebra $\mathfrak{g}$. We denote $\mathfrak{g}_n$ the dynamical Lie algebra that is generated after applying the Lie bracket at most n times. In the truncated Magnus expansion with order k_M , the largest algebra required is $\mathfrak{g}_{k_M}$. Often, the size of this dynamical Lie algebra is substantially smaller than $\dim(\mathfrak{g})$, which is advantageous for the efficiency of the algorithm proposed in this work. On the Lie algebra we can define a structure constant f_{ijk} that describes the relation between its elements:

$$[L_i, L_j] = \sum_k f_{ijk} L_k, \quad L_k \in \mathfrak{g}_n, \quad \forall L_i, L_j \in \mathfrak{g}_n. \quad (21)$$

Thus, when we write the Hamiltonian $H(t)$ in the basis of the elements of this Lie algebra, we know *a priori* all of the commutation relations that are required by the ME. This component of the calculation corresponds to Fig. 1(a).

The final step of this process is to find out which integrals of commutators have to be solved for a given expansion order. We resort to the proposal from Ref. [40] due to its systematic nature. Those authors represent integrals of commutators of $H(t)$ as binary trees, and the Magnus expansion as a weighted series of binary trees. This approach is explained in detail in Appendix A.

Together, these steps yield an explicit expression of the Magnus expansion, where we can quickly obtain the effective Hamiltonian. It is expressed as an element of the dynamical Lie algebra,

$$M^{(k_M)}(t, \vec{p}) = \sum_{\mu} a_{\mu}^{(k_M)}(t, \vec{p}) L_{\mu}, \quad (22)$$

with truncation order k_M , propagation time t , and control functions represented with $\vec{p}$. The Lie-algebra coefficients $a_{\mu} := a_{\mu}^{(k_M)}(t, \vec{p})$ are polynomials in the evolution time t and in the control function coefficients

$$a_{\mu} := \sum_{k=1}^{k_M} \sum_{q=0}^k \sum_{\gamma_1, \dots, \gamma_q=0}^m \mathbf{T}_{\mu, \vec{\gamma}}^{(k, q)} t^k \prod_{i=1}^q (p_{\gamma_i} t^{\gamma_i}), \quad (23)$$

where the index γ_i is related to the control coefficient p_{γ_i} , and to the dynamical coefficient $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, q)} := \mathbf{T}_{\mu, \gamma_1, \dots, \gamma_q}^{(k, q)}$, with $\vec{\gamma} = (\gamma_1, \dots, \gamma_q)$. The derivation of this expression is detailed in Appendix A, showing the formula for the coefficients $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, q)}$. The number of summation terms in this expression can be improved by setting a time truncation order Γ , that is, by neglecting terms in the sum that are smaller than $O(t^{\Gamma})$. The γ_i indices represent terms that are equivalent up to permutation;

thus, we accumulate the dynamical coefficients that represent the same term into a single term, reducing the total number of elements in the sum in Eq. (23).

The dynamical coefficients $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, q)}$ encapsulate the time-dependent dynamics that are generated from $H(t)$, up to a certain order k_M . The calculation of these coefficients is quite involved, as explained before, but only needs to be done once for a given control model (A and B). In fact, a given dynamical coefficient contains contributions from many integrals of commutators that result in the same Lie-algebra element L_{μ} and which correlate the same monomial in $\vec{p}$. In essence, this efficiently compresses the complexity of the ME calculation into these dynamical coefficients, instead of the convoluted integrals of commutators previously mentioned in Sec. II. The remaining complexity in the calculation is related to the degrees of freedom given by the evolution time t and control coefficients $\vec{p}$. So, once the dynamical coefficients are obtained, calculating the ME is reduced to a much faster polynomial evaluation.

With the polynomial form of the ME, it is practical to obtain the effective Hamiltonian for high approximation order. We will show in the following section that this is in fact numerically efficient. The polynomial form of Eq. (23) also allows for analytic derivatives; we will later showcase its use in optimal control in Sec. V.

IV. QUANTUM SIMULATION

For the purpose of quantum simulation we will consider two specific Hamiltonian models: "sparse," with nearest-neighbor interactions, and "dense," with long-range interactions. Recall also the single-control Hamiltonian format from Eq. (18).

(1) The sparse model is a one-dimensional transverse field Ising model with n qubits, where the generators of the dynamics are

$$A := \sum_{i=1}^{n-1} Z_i Z_{i+1}, \quad B := \sum_{i=1}^n X_i. \quad (24)$$

(2) The dense model is a transverse field Ising model with long-range interactions,

$$A := \sum_{i=1}^n \sum_{j>i}^n J_{ij} Z_i Z_j, \quad B := \sum_{i=1}^n X_i, \quad (25)$$

where $J_{ij} = 1/|i - j|$.

These models have been selected to allow comparison between the evolution method's performance in the regimes where the dynamics are simple (sparse and/or low n) versus when they are complex (dense and/or large n). We expect the performance of the method to scale with the size of the dynamical Lie algebra.

A. Numerical efficiency

Generating the dynamical coefficients $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, q)}$ and evaluating the polynomials $a_{\mu}(t, \vec{p})$ does not involve states or operators of the full system size. Instead, its numerical complexity scales directly with the complexity of the dynamics of the quantum system, obtained from the Lie algebra, resulting in

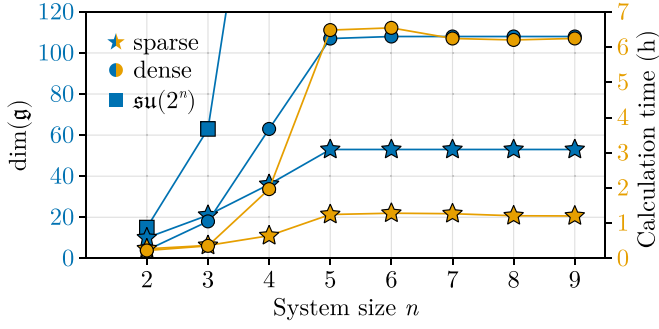


FIG. 2. Left axis (blue): Scaling of the size of the Lie algebra with system size, for the two models and $\mathfrak{su}(n)$, using $k_M = 10$ and $\Gamma = 12$. Right axis (yellow): Scaling of the calculation time, in hours, of the dynamical coefficients $T_{\mu,\gamma}^{(k,q)}$ as system size increases.

a scalable method. Moreover, for the generation of the coefficients, we do not require the full Lie algebra, and may only generate it up to the commutation depth k_M that corresponds to the ME truncation order. Once the Lie algebra and its structure constants are generated, all of the relevant properties of the dynamics of the quantum system are accounted for, within the convergence radius of the Magnus expansion.

There are two numerical timescales associated with this method:

- (1) The calculation time is the time to compute the structure constant of the dynamical Lie algebra, the integrals and commutators, resulting in the dynamical coefficients $T_{\mu,\gamma}^{(k,p)}$.
- (2) The evaluation time is the time to evaluate the polynomial expression of the Magnus expansion, using the precomputed dynamical coefficients in the expression of Eq. (23).

In Fig. 2, we see a strong correlation between the Lie-algebra size $\dim(g)$ and the calculation time. The size of the dynamical Lie algebra, with limited commutation depth, scales weakly with the number of qubits, and thus generating the dynamical coefficients takes comparable time for different system sizes [62].

In Fig. 3, we show benchmarks on the evaluation time. Evaluating the ME polynomials is fast, ranging from few microseconds at small orders to about 100 μ s in the larger orders, which is four orders of magnitude faster than the work in Ref. [43]. Moreover, the evaluation time remains nearly

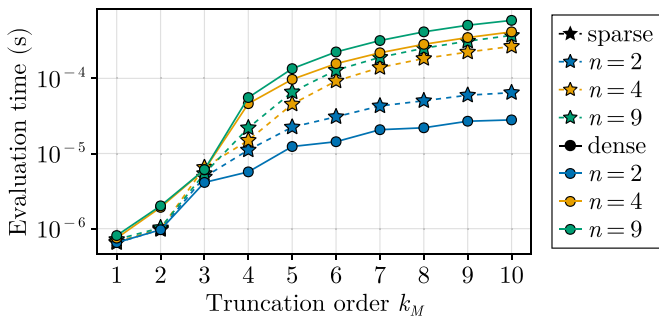


FIG. 3. Median evaluation time of the Magnus expansion polynomial, using the expression shown in Eq. (23), with truncation order k_M and $\Gamma = 12$, for both models and various system sizes n .

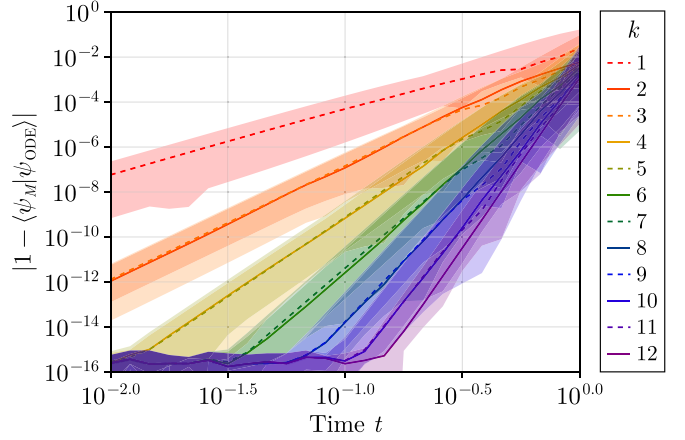


FIG. 4. Comparison of state errors at different orders, using a single step of the Magnus expansion. The control coefficients p_γ are sampled from a uniform distribution $p_\gamma \in [-1, 1]$. The sparse model with $n = 3$ and time truncation $\Gamma = 14$ is used. The average of 20 samples (random initial state and random controls) is plotted in a line, and the minimum and maximum samples are plotted as a band of the respective color. A plateau at around 2×10^{-16} shows up due to floating-point error.

constant with increasing system size across all considered models.

Access to higher Magnus expansion orders allows for coarser time discretization without compromising accuracy. Consequently, the number of state-propagation steps—and hence the main computational bottleneck—is significantly reduced. This makes the method especially suitable for quantum control applications that demand fast and accurate propagation.

B. Error scaling and time symmetry

When truncating the Magnus expansion at even orders $k_M = 2s$, the error relative to the true effective Hamiltonian is $O(t^{2s+3})$ [see Eq. (17)]. The ME is used as a *single-step* propagator in a Hamiltonian simulation setting to obtain the truncation error. This is done by comparing the propagated state when evolving under the effective Hamiltonian,

$$|\psi_M\rangle := \exp\{-iM(t, \vec{p})\}|\psi_I\rangle, \quad (26)$$

versus a state propagated with an ordinary differential equation (ODE) method $|\psi_{ODE}\rangle$ with sufficient intermediate time steps [63], which we consider to be an exact state up to machine precision [64–66]. The initial state is sampled from the uniform Haar distribution [67]. The results of this comparison for increasing truncation orders are shown in Fig. 4. The log-log plot reveals polynomial scaling for single-step propagation, increasing for each *even* truncation order.

At each truncation order we fit the error metric to obtain the corresponding apparent power law k_{app} , considering $\varepsilon_M > 10^{-15}$ to suppress floating-point errors. These are shown in Fig. 5 for both models, for different system sizes and Γ 's. In Ref. [36], it is shown that $k_{app} = k_M + 3$ for even-valued k_M due to the time symmetry of the expansion. Our numerical tests suggest that this scaling is preserved with our method, as long as $\Gamma \geq k_M + 2$. In contrast, in the specific example

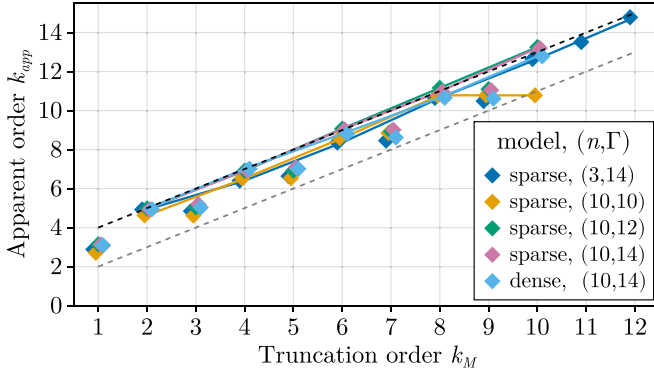


FIG. 5. Scaling laws for sparse and dense models with $n = 10$, where for the sparse model we also compare the order scaling with Γ . Dashed lines indicate the time-symmetry threshold at $k_{\text{app}} = k_M + 3$ (black) and the threshold $k_{\text{app}} = k_M + 1$ (gray). Solid lines connect markers of the color-corresponding model for even-valued k_M . Most models considered follow the time-symmetry scaling $k_{\text{app}} \approx k_M + 3$.

shown in Fig. 5 with the sparse model, $n = 10$ and $\Gamma = 10$, we see that the apparent order does not follow the time scaling expectation at $k_M = 10$.

V. OPTIMAL CONTROL

Having established in Sec. IV that the polynomial Magnus-operator representation provides fast and accurate propagation, we now use the same representation in a substantially more demanding setting. The purpose of this section is to benchmark the control framework developed in the previous sections on a realistic and challenging quantum optimal-control problem. We focus on the design of fast, high-fidelity multiqubit gates, which are a key building block for scalable quantum computing platforms. In contrast to sequences of single- and two-qubit gates, directly implemented multiqubit operations can reduce circuit depth and mitigate accumulated control noise.

Neutral-atom platforms provide a particularly relevant setting for such gates, as multiple atoms can interact simultaneously in a geometry-dependent manner. The quantum optimal control of multiqubit gates in these systems is therefore an active area of research [28,68–73]. At the same time, these problems pose substantial numerical challenges due to increasing pulse durations, high-dimensional control landscapes, and the computational cost of time-dependent simulations.

The design of multiqubit gates becomes progressively more challenging as the number of qubits increases, due to correspondingly longer pulse durations, a growing number of control parameters, and extended simulation times. The key advantage of our method is that the effective Hamiltonian and its derivatives are available analytically as polynomial functions of the control parameters, allowing continuous pulses and pulse durations to be optimized directly. This gives the freedom to include pulse-duration minimization constraints directly as Lagrange multipliers during minimization [74]. Our method corresponds to an extension of GRAPE for continuous pulses.

In a given quantum control optimization problem, we aim to design a pulse that implements the target gate G_{tg} , such that $U(\vec{x}) \simeq G_{\text{tg}}$, where $\vec{x}$ is the vector of tunable control parameters that define the pulse and $U(\vec{x})$ is the unitary propagator of the pulse. The optimization of the state trajectory consists of determining $\vec{x}$ that minimizes a cost function $J[\vec{x}]$. In this work we use

$$J[\vec{x}] := \frac{1}{2} \sum_{r=1}^R (1 - \text{Re}\langle \phi_r | U(\vec{x}) | \psi_r \rangle), \quad (27)$$

where $|\psi_r\rangle$ and $|\phi_r\rangle$ are the initial state and target state, respectively, for the trajectory indexed by r , and R is the number of trajectories that are being optimized. A suitable set of initial states $|\psi_r\rangle$ must be chosen for the design of a target gate G_{tg} . This set may be a complete basis of the Hilbert space, but often it is possible to use a much smaller set while still addressing all of the symmetries present in the dynamics. Once these are established, the set of target states is uniquely determined: $|\phi_r\rangle = G_{\text{tg}}|\psi_r\rangle$.

We choose the problem of designing parametrized multi-qubit gates for neutral atoms to showcase the advantages of the method. Namely, we aim to show that we can take advantage of the time-adaptive property of the method to efficiently design gates, by using previous pulse shapes as initial guesses for gates with similar parameters.

A. Hermite spline control pulse

The convergence time of the Magnus expansion is often less than the time necessary to design a useful gate. Therefore, we concatenate multiple ME propagation segments to form a longer pulse. Each ME segment is comprised of a polynomial-shape control function, in contrast with GRAPE, where the propagation consists of a sequence of constant step functions. Several options exist for selecting both the representation of the control pulse and its parameters. In this work, we represent and optimize the control pulse as an Hermite spline [75]. This representation ensures continuity for the whole control function, and can be easily converted into the underlying polynomial representation with which we have described the ME.

An Hermite spline is a function with class C^L that consists of a sequence of segments that split the control function into S time intervals as such

$$d(t) := \begin{cases} d_1(t), & t \in [0, t_1] \\ \vdots \\ d_s(t), & t \in [t_{s-1}, t_s] \\ \vdots \\ d_S(t), & t \in [t_{S-1}, t_S], \end{cases} \quad (28)$$

with the time at the end of the segment, $t_s := t_{s-1} + \Delta t_s$, with $t_0 = 0$. These are defined through the segment time intervals Δt_s . The function $d_s(t)$ is a polynomial given by

$$d_s(t) := \sum_{n=0}^{2L+1} d_n^{(s)} \frac{t^n}{n!}. \quad (29)$$

As we will later see, our gate sequence consists of S gates, one per polynomial segment, where the gate s is given by the

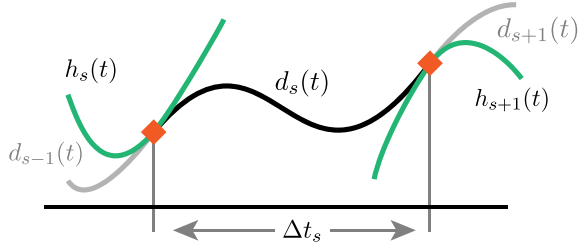


FIG. 6. Segment s of an Hermite spline. The Hermite nodes are shown as orange diamonds, and delimit a time interval with duration Δt_s . The Hermite polynomials $h_s(t)$ and $h_{s+1}(t)$ at these nodes are represented as green lines. The black line that connects the orange markers is the polynomial that defines the shape of the spline on this interval.

propagation of the Magnus expansion, $\exp\{-iM(\Delta t_s, \vec{d}^{(s)})\}$, with $\vec{d}^{(s)} = (d_0^{(s)}, \dots, d_{2L+1}^{(s)})$. At the start and end of the segment s there exist Hermite nodes that define a polynomial of degree L . These polynomials are denoted $h_s(t)$ and $h_{s+1}(t)$, respectively. The construction of an Hermite spline involves determining the polynomial $d_s(t)$ whose l th-order derivatives match the ones at the start and end of the segment, ensuring that $d(t)$ is of class C^L :

$$\left. \frac{d^l}{dt^l} d_s(t) \right|_{t=t_s} = \left. \frac{d^l}{dt^l} h_s(t) \right|_{t=t_s}, \quad (30)$$

$$\left. \frac{d^l}{dt^l} d_s(t) \right|_{t=t_{s+1}} = \left. \frac{d^l}{dt^l} h_{s+1}(t) \right|_{t=t_{s+1}}, \quad (31)$$

with $l = 0, \dots, L$. The polynomial function $d_s(t)$ is of polynomial degree $2L + 1$, since the number of constraints on the segment is $2L + 2$, becoming fully determined with $\vec{c}_s = (\Delta t_s, \vec{d}^{(s)})$. An illustration of one of these segments is shown in Fig. 6. An Hermite spline with class C^L and S segments is fully determined with $S + 1$ nodes $h_s(t)$ and S time intervals Δt_s . Each node has $L + 1$ parameters. Thus, the number of parameters for such a spline is $(S + 1)(L + 1) + S$. These are the Hermite spline parameters $\vec{h}$, in contrast with the segment polynomial coefficients $\vec{c} = (\vec{c}_1, \dots, \vec{c}_S)$ that define the polynomial function $d_s(t)$ for each of the segments $s = 1, \dots, S$.

The Hermite spline parameters $\vec{h}$ are the controls that are used on the numerical optimizer. Since the ME is expressed in terms of the segment polynomial coefficients $\vec{c}$, the Hermite spline parameters $\vec{h}$ are transformed into the segment polynomial coefficients $\vec{c}$ through Eqs. (30) and (31). The Jacobian of this transformation is required for gradient-based optimization. The mathematical details of this transformation are discussed in Appendix B.

B. Rydberg atoms

The design problem of the multicontrolled phase gate $C_k P(\phi)$ for Rydberg atoms has been tackled in multiple works [28,68,70–72], with k being the number of control qubits. Here, we focus on the parametrized variant

$$C_k P(\phi) := (\mathbb{I} - (|1\rangle\langle 1|)^{\otimes n}) e^{i\phi} + (|1\rangle\langle 1|)^{\otimes n}, \quad (32)$$

where $n = k + 1$ is the number of atoms. Then, the target gate for the quantum optimal-control problem is $G_{\text{tg}} = C_k P(\phi)$. The typical Hamiltonian considered when studying Rydberg atoms is

$$H(t) = \frac{\Omega(t)}{2} \sum_{i=1}^n X_i - \Delta(t) \sum_{i=1}^n |r\rangle_i \langle r|_i + \sum_{i>j} V_{ij} |rr\rangle_{ij} \langle rr|_{ij}, \quad (33)$$

with

$$X_i := |0\rangle_i \langle 0|_i + |r\rangle_i \langle 1|_i + |1\rangle_i \langle r|_i, \quad (34)$$

and where $\Omega(t)$ is the Rabi frequency of a global rotation, and $\Delta(t)$ is the corresponding off-resonance detuning of the transition [76]. The states $|0\rangle$ and $|1\rangle$ are used for single-qubit operations and storage due to their longevity. The state $|r\rangle$ is the Rydberg state, which allows for entanglement generation between neighboring atoms. These states interact through the van der Waals interatomic potential V_{ij} , which scales with $1/R_{ij}^6$, where R_{ij} is the distance between atoms i and j [25].

When designing gates for this platform it is common to consider the regime where $V_{ij} \gg \Omega$, called the blockade regime. In the perfect blockade scenario there is a symmetric, all-to-all blockade interaction, which we can represent with a simpler Hamiltonian control model [68]:

$$H(t)/\Omega = \frac{1}{2} \sum_{i=1}^n X_i Q_{[i]} + d(t) \frac{1}{2} \sum_{i=1}^n Z_i, \quad (35)$$

where

$$Q_{[i]} := \prod_{\substack{j=1 \\ j \neq i}}^n Q_j, \quad Q_j := |0\rangle_j \langle 0|_j + |1\rangle_j \langle 1|_j, \quad (36)$$

$$Z_i := |0\rangle_i \langle 0|_i + |1\rangle_i \langle 1|_i - |r\rangle_i \langle r|_i. \quad (37)$$

The Rabi frequency is kept constant, $\Omega(t) = \Omega$, and the detuning control function $\Delta(t)$ is rescaled accordingly: $d(t) = \Delta(t)/\Omega$. The control function $d(t)$ takes the form of an Hermite spline, which corresponds to the control pulse that is to be optimized. We focus only on a subset of the Hilbert space of states

$$|\psi_i\rangle := |1\rangle^{\otimes i} |0\rangle^{\otimes (n-i)}, \quad i = 0, \dots, n, \quad (38)$$

as the system Hamiltonian remains invariant under atomic permutations, faithfully representing the full Hilbert space.

In gate design for Rydberg atoms it is common to consider the availability of single-qubit gates that can act before and/or after an entanglement-generating interaction [16,70]. Then, the parametrized gate sequence

$$U(\vec{h}, \theta) := \prod_{i=1}^n R_{Z_i}(\theta) \prod_{s=1}^S D_s(\vec{c}_s(\vec{h}_s)) \quad (39)$$

is used for the design of the target gate, where $\vec{h}$ are the Hermite spline parameters as described in Sec. V A, S is the number of segments, and θ is the rotation angle for the single-qubit rotations,

$$R_{Z_i}(\theta) = \mathbb{I} \cos(\theta/2) - i(\mathbb{I} - 2|1\rangle_i \langle 1|_i) \sin(\theta/2) \quad (40)$$

that act on atom i , and

$$D_s(\vec{c}_s) := \exp\{-iM^{(k_M)}(\Delta t_s, \vec{d}^{(s)})\} \quad (41)$$

is the unitary propagator obtained from the truncated Magnus expansion from Eq. (22).

The numerical optimization requires the derivatives of the propagator $\nabla_{\vec{h}}U(\vec{h}, \theta)$ and $\partial_{\theta}U(\vec{h}, \theta)$. The method to obtain these is shown in Appendix B.

C. Numerical optimization

For numerical optimization of the cost function $J[\vec{h}, \theta]$ we have used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm with box constraints on the time intervals [77]. The pulse duration is constrained to $T \in [T_{\min}, T_{\max}]$ by imposing constraints on the time intervals of the Hermite spline, $T_{\min}/S \leq \Delta t_s \leq T_{\max}/S$. During the optimization the truncation error of the ME is estimated with

$$\varepsilon_M^2 := \sum_{\mu} (a_{\mu}^{(k_M)})^2 \|L_{\mu}\|^2 \geq \|M_{k_M}\|^2, \quad (42)$$

where $\|L_{\mu}\|$ is the ℓ_1 operator norm of the Lie-algebra element L_{μ} , computed once before optimization. We set a threshold ε_* where, whenever

$$\sum_{s=1}^S \varepsilon_M^{(s)} > \varepsilon_*, \quad (43)$$

the optimization is stopped in order to increase the number of nodes in the spline, until $\varepsilon_M \leq 0.1\varepsilon_*$, with $\varepsilon_M^{(s)}$ the truncation error at segment s . This ensures the truncation error remains bounded while the spline stays nearly unchanged. The downside is that optimization has to be interrupted as the number of Hermite nodes increases, changing $\vec{h}$. The optimization converges successfully once $J \leq \varepsilon_M$. The duration of the pulse is improved by including a Lagrange multiplier term,

$$J_T := \lambda_T \sum_{s=1}^S \Delta t_s = \lambda_T T, \quad (44)$$

in the cost function, with λ_T tuned heuristically.

In Fig. 7, an example is shown of how the control pulse changes throughout the optimization, with the controlled-controlled-Z (C2Z) gate as the target. The control pulse is an Hermite spline with Hermite order $L = 1$. The bounds $\Omega T \in [4.0, 18.0]$ are applied in this optimization. We have observed that with lower $T_{\max}$ bounds, the optimizer requires more iterations to find an adequate control pulse, and often with $J > 10^{-3}$. The optimization starts with $d(t)$ being an Hermite spline approximation of $0.1 \cos(2\pi\Omega t/3)$ with 21 segments and with duration $\Omega T = 9.0$. As the optimization progresses, the total truncation error of the ME surpasses the threshold, leading to a resampling of the spline of the control pulse, increasing the number of nodes and the number of control parameters. At iteration 180, the cost function J decreases substantially and remains at about 10^{-6} for the remainder of the optimization. This plateau is likely due to errors in the gradient of the propagator. During this portion of the optimization there are spikes in the cost function. These coincide with the instances where the Hermite spline is resampled to ensure that the truncation error of the ME is

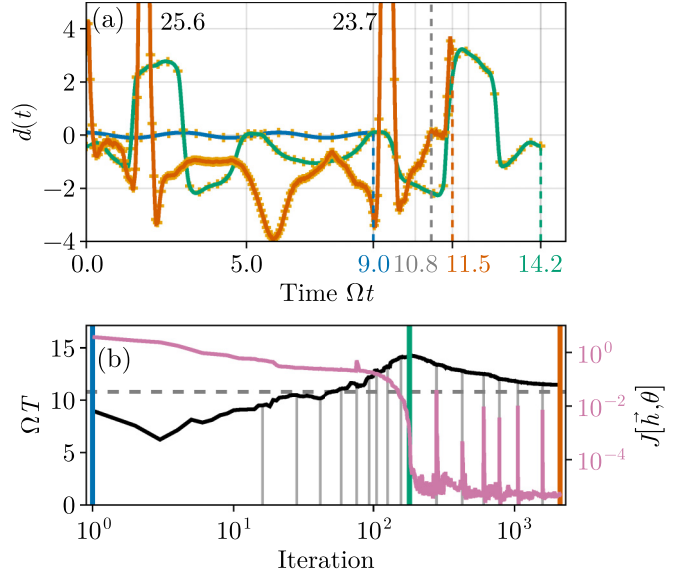


FIG. 7. Optimization of the c2z gate, with $(T_{\min}, T_{\max}, \lambda_T, \epsilon_M) = (4.0, 18.0, 10^{-3}, 10^{-6})$. (a) Three pulses are shown for iterations 0, 180, and 2106, with respective colors blue, green, and orange. The orange pulse has two marked high-value peaks which fall outside of the plotting region. The yellow + symbols show where the Hermite nodes are placed for that spline. (b) A black line shows how the dimensionless pulse duration ΩT progresses along the optimization. Vertical solid gray lines indicate where the threshold on Eq. (43) is triggered. The cost function $J[\vec{h}, \theta]$ is plotted in pink.

bounded. From there onward, the pulse duration is gradually improved through the Lagrange multiplier term J_T , leading to an increase of the sharpness of the peaks of the control pulse. The final pulse duration is $\Omega T = 11.5$, which is 6% slower than the time-optimal pulse with $\Omega T = 10.8$ [28]. This difference likely arises because we optimize the laser detuning $\Delta(t)$ instead of the phase [78]. Explicit values related to the marked pulses for their duration ΩT , number of segments, S , the number of pulse parameters, $\dim(\vec{h})$, and cost function $J[\vec{h}, \theta]$ are shown in Table I. As the optimization progresses, and the control pulse becomes more detailed, and the number of parameters increases accordingly. It is comparable with the 400 discrete time steps used in the time-optimality study in Ref. [68].

We direct our attention now to the optimization of control pulses for the $C_4P(\phi)$ gate family. This is inherently a more challenging problem due to the increased number of qubits. Prior work on time-optimal pulses has, to the best of our

TABLE I. Parameters and properties of several control pulses for realizing the C2Z gate.

Iteration	ΩT	S	$\dim(\vec{h})$	$J[\vec{h}, \theta]$
0	9.00	21	65	3.74
180	14.2	58	176	9.84×10^{-4}
500	12.6	95	287	5.05×10^{-6}
1000	11.8	123	371	5.64×10^{-6}
2106	11.5	208	626	4.91×10^{-6}

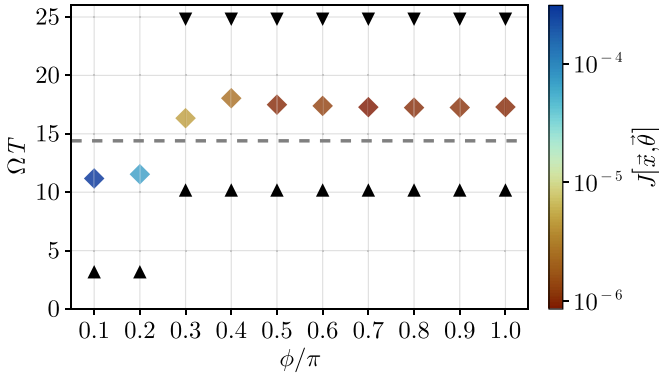


FIG. 8. Pulse durations and costs for the different angles. The colors of the markers indicate the respective costs, related to the color bar on the right. The size of the markers at a given ϕ/π showcases the progress of the pulse duration and cost along the iterations. Smaller markers indicate earlier iterations. The gray line indicates the duration of the time-optimal pulse at $\phi = \pi$ from Ref. [28]. For $\phi < \pi$, we expect lower optimal times. The upper and lower triangle markers indicate the imposed pulse duration bounds $T_{\max}$ and $T_{\min}$, respectively.

knowledge, only considered a single member of this family: the $C_4Z = C_4P(\pi)$ gate [28]. Beyond the increased computational cost of simulation and gradient evaluation, the control pulses themselves exhibit greater structural complexity, requiring more spline segments to be represented accurately. This results in a higher-dimensional optimization landscape.

Each round of optimization proceeds similarly to the C_2Z case. Our strategy is to optimize the gate family in increments of $\Delta\phi = \pi/10$, using warm starting: the optimized pulse for $C_4P(n\Delta\phi)$ serves as the initial guess for $C_4P((n+1)\Delta\phi)$. We begin by finding a pulse for the $C_4P(\Delta\phi)$ gate with $\varepsilon_* = 10^{-4}$, tuning $T_{\min}$ and $T_{\max}$ heuristically. At this stage we set $\lambda_T = 0$, as pulse duration is not yet a concern. Once $J \leq \varepsilon_*$, we refine by tightening to $\varepsilon_* = 5 \times 10^{-6}$ and enabling duration optimization with $\lambda_T = 10^{-4}$. This refinement procedure is then repeated for each subsequent gate up to $C_4P(\pi) = C_4Z$. For $\phi/\pi \geq 0.3$, we update the duration constraints to improve optimizer convergence. The optimization results are shown in Fig. 8. The C_4Z pulse obtained is approximately 20% longer than the time-optimal result of Ref. [28], and this difference we attribute again to optimizing the laser detuning instead of its phase. The remaining gates in the interval $\phi/\pi \in [0.3, 1.0]$ exhibit similar durations, a trend also observed for two- and three-qubit controlled-phase gates [68,71].

VI. DISCUSSION AND OUTLOOK

We demonstrated that the high-order Magnus expansion can now be used in regimes where the involved computational cost had made it impractical. We express the solution operator from the Magnus expansion in polynomial form for Hamiltonians consisting of a time-independent drift and a single control term. This is achieved by systematically solving the high-order integrals of commutators required by the Magnus expansion. This polynomial form condenses the dynamics into a set of coefficients, shifting the computational complexity of the Magnus expansion to the control degrees of

freedom alone. This method enables numerical calculation of the Magnus expansion in hundreds of microseconds, up to 12th order, while being 4 orders of magnitude faster than previous approaches. Naturally, this application is well suited for tasks involving simulation of time-dependent quantum systems in classical computers, but can also be an important analytical tool for the study of these systems.

We further demonstrate the method's applicability to quantum optimal control by designing multiqubit gates for Rydberg-atom systems. The approach enables optimization of continuous control pulses represented as splines while maintaining a similar number of parameters when compared with the state of the art. As the pulse duration is optimized jointly with the pulse shape, suitable solutions require less fine-tuning compared to standard GRAPE-based techniques. Using this framework, we successfully design pulses for the five-qubit phase gate family, leveraging the adaptive-time capabilities of our method.

The julia code to generate the dynamical coefficients and evaluate the respective Magnus expansion polynomial and its derivatives can be found in Ref. [79]. It also contains the code for the optimal quantum control methods adopted in this work.

ACKNOWLEDGMENTS

We thank R. d. Keijzer, J. Postema, J. van de Kraats, M. Mohan, P. Kersbergen, and E. Akaturk for fruitful discussions. This research was financially supported by the Dutch Ministry of Economic Affairs and Climate Policy (EZK), as part of the QDNL program, and the Horizon Europe program HORIZON-CL4-2021-DIGITAL-EMERGING-01-30 via Project No. 101070144 (EuRyQa).

DATA AVAILABILITY

The data that support the findings of this article are openly available [79].

APPENDIX A: REPRESENTING THE MAGNUS EXPANSION AS A POLYNOMIAL

The Magnus expansion can be expressed as a sum of integrals of commutators represented as binary trees, proposed by Ref. [40]. The k th-order term of the ME is

$$M_k(t) = (-i)^{k-1} \sum_{\tau \in \mathbb{T}_{k-1}} \alpha(\tau) I_\tau[t, H(t)], \quad (\text{A1})$$

with the chained-commutator-integral functional taking the form

$$I_\tau[t, H(t)] = \int_0^t dt_1 \int_0^{t_1} dt_2 \cdots \int_0^{t_{\tau_1-1}} dt_i \cdots \times \int_0^{t_{\tau_{k-1}}} dt_k [H(t_1) \cdots H(t_k)]_\tau. \quad (\text{A2})$$

The set of binary trees with k nodes, $\mathbb{T}_k$, represents the integrals and commutators of order k . To each binary tree τ there is a weight $\alpha(\tau)$. Each tree represents the integration limits $t_{\tau_i} \in \{t, t_1, \dots, t_{i-1}\}$ and the order in the chain of

commutators $[H(t_1) \cdots H(t_k)]_\tau$. The specifics of how to construct the chained-integral functional $I_\tau[t, H(t)]$ is given below.

1. Binary trees as integrals of commutators

The set of binary trees is generated recursively according to

$$\mathbb{T}_k := \left\{ \begin{array}{c} \tau_1 \\ \diagup \quad \diagdown \\ \tau_2 \end{array} : \tau_1 \in \mathbb{T}_{m_1}, \tau_2 \in \mathbb{T}_{m_2}, m_1 + m_2 = k - 1 \right\}, \quad (\text{A3})$$

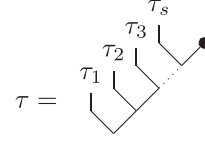
where the trivial case is $\mathbb{T}_0 = \{\bullet\}$. Then the first few sets of trees are

$$\mathbb{T}_1 = \left\{ \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array} \right\}, \quad (\text{A4})$$

$$\mathbb{T}_2 = \left\{ \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}, \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} \right\}, \quad (\text{A5})$$

$$\mathbb{T}_3 = \left\{ \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \quad \bullet \end{array}, \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \quad \bullet \end{array}, \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \quad \bullet \end{array}, \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \quad \bullet \end{array} \right\}. \quad (\text{A6})$$

We can identify any tree τ in a unique way:



where τ_l are the subtrees of τ . For each of the trees in $\mathbb{T}_k$ there is a corresponding weight $\alpha(\tau)$ on the ME given by

$$\alpha(\tau) = \frac{B_s}{s!} \prod_{l=1}^s \alpha(\tau_l), \quad (\text{A7})$$

where the trivial tree has weight $\alpha(\bullet) = 1$ and B_s is the s th Bernoulli number [59]. Each tree represents the integral and commutation of the Hamiltonian evaluated at different times. This is done recursively according to

$$H_\tau(t) = \left[\int_0^t ds H_{\tau_1}(s), H_{\tau_2}(t) \right] \quad \text{with} \quad \tau = \begin{array}{c} \tau_1 \\ \diagup \quad \diagdown \\ \tau_2 \end{array}. \quad (\text{A8})$$

The chained-commutator integral from Eq. (A2) then corresponds to the integration of Eq. (A8):

$$I_\tau[t, H(t)] := \int_0^t dt_1 H_\tau(t_1). \quad (\text{A9})$$

For illustrative purposes, we follow with a few examples in sequence. The single tree in $\mathbb{T}_1$ represents the integral

$$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array} \sim I_{\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}}[t, H(t)] = \int_0^t dt_1 H_{\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}}(t_1) = \int_0^t dt_1 \left[\int_0^{t_1} dt_2 H(t_2), H(t_1) \right] = \int_0^t dt_1 \int_0^{t_1} dt_2 [H(t_2), H(t_1)]. \quad (\text{A10})$$

which is the second term of the ME. Then, following to the second element of $\mathbb{T}_2$,

$$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} \sim \int_0^t dt_1 H_{\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}}(t_1) = \int_0^t dt_1 \left[\int_0^{t_1} dt_2 H(t_2), H_{\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}}(t_1) \right] = \int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_1} dt_3 [H(t_2), [H(t_3), H(t_1)]]]. \quad (\text{A11})$$

Another more complex case that shows up at fourth order is illustrated here:

$$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \quad \bullet \end{array} \sim \int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \int_0^{t_2} dt_4 [[H(t_3), [H(t_4), H(t_2)]], H(t_1)]. \quad (\text{A12})$$

Each vertical line represents time integration, and each bifurcation corresponds to a commutation. The commutators at the root of the tree are computed from the leaves, shown as $\bullet$, downward. The tree also determines the integration limits. Each leaf on the tree is uniquely associated with an integration time t_i .

2. Separating the integrals from the commutators

The high-order terms of ME involve an increasingly complex series of integrals and commutators. Evaluating $H(t)$ independently on these integrals becomes numerically unfeasible if no special treatment is taken. We have a single-control Hamiltonian

$$H(t) = A + d(t)B, \quad (\text{A13})$$

where $d(t)$ is a polynomial in time t , which allows us solve the high-order commutators independently of the integrals. The core of the method involves generating a basis for the Lie algebra generated by $\{A, B\}$, plus approximations neglecting higher powers in the time t . To help illustrate the process, consider the third-order commutator that is related to the second tree in $\mathbb{T}_2$:

$$\begin{aligned} [H(t_1) \cdots H(t_3)]_{\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}} &= [H(t_1), [H(t_2), H(t_3)]] \\ &= [A, [A, d(t_3)B]] + [A, [d(t_2)B, A]] \\ &\quad + [d(t_1)B, [A, d(t_3)B]] \\ &\quad + [d(t_1)B, [d(t_2)B, A]]. \end{aligned} \quad (\text{A14})$$

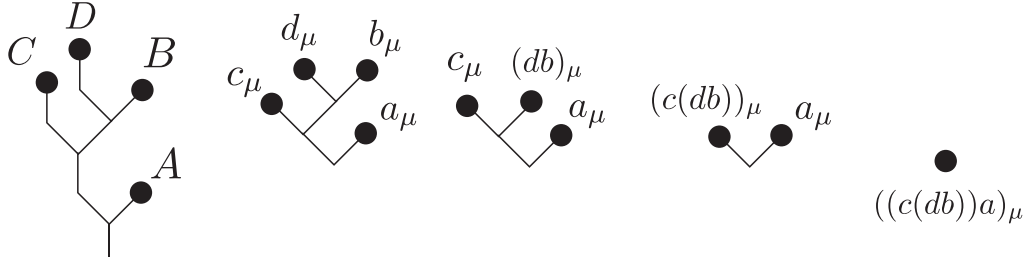


FIG. 9. Example of evaluating a chained integral defined over a binary tree. The operators A, B, C , and D are placed at the leaves, which are represented in the Lie algebra with $A = \sum_{\mu} a_{\mu} L_{\mu}$ (equivalently for B, C , and D). We turn it into the Lie-algebra representation at the second step. Here we also remove vertical branches as they represent integration, which is unnecessary for time-independent operators. The commutation is evaluated in topological order, and occurs between the members at each end of the branch. Since every operator here belongs to the Lie algebra, their transformation under commutation is easily described with the structure tensor.

In this calculation we distribute the sums for A and B that are inside of the chained commutator. In each entry of the commutator one can place either A or B , and through distribution they are placed in every combination. For a k th-order chained commutator, there are 2^k options for placement of A, B , although in some of them, the commutation result is zero due to self-commutation ($[A, A] = [B, B] = 0$). Depending on which entry the operator B is placed, the control $d(t_{\circ})$ is evaluated at a different time index, as illustrated above. Then, we adopt the notation

$$[(A), (d(t_{\circ})B)]_{\tau, i} \quad (\text{A15})$$

for the element i of a distributed sum, with the commutator structure given from the tree τ . Here are examples of the representation of the terms shown in the calculation above, with $i = i_k \cdots i_1 \in \{0, \dots, 2^k - 1\}$ represented in base 2:

$$[(A), (d(t_{\circ})B)]_{\tau, 001_2} = [A, [A, d(t_3)B]], \quad (\text{A16})$$

$$[(A), (d(t_{\circ})B)]_{\tau, 010_2} = [A, [d(t_2)B, A]], \quad (\text{A17})$$

$$[(A), (d(t_{\circ})B)]_{\tau, 101_2} = [d(t_1)B, [A, d(t_3)B]], \quad (\text{A18})$$

$$[(A), (d(t_{\circ})B)]_{\tau, 110_2} = [d(t_1)B, [d(t_2)B, A]]. \quad (\text{A19})$$

Then, a generic commutator chain that comes from a tree τ is given by

$$\begin{aligned} [(A), (d(t_{\circ})B)]_{\tau} &= \sum_{i_k=0}^1 \cdots \sum_{i_1=0}^1 [(A), (d(t_{\circ})B)]_{\tau, i_k \cdots i_1} \\ &= \sum_{i=0}^{2^k-1} [(A), (d(t_{\circ})B)]_{\tau, i}, \end{aligned} \quad (\text{A20})$$

where some of these terms evaluate to zero due to self-commutation. The algorithm to evaluate each term of this sum is explained in Fig. 9. The Lie algebra is expressible only in relation to the commutators of A and B , and so we need to separate the time dependence that originates from $d(t_{\circ})$. Due to the linearity of the commutator, the chain can be written as the following sum:

$$[H(t_1) \cdots H(t_k)]_{\tau} = \sum_{i=0}^{2^k-1} \left(\prod_{j \in J_i} d(t_j) \right) [(A), (B)]_{\tau, i}, \quad (\text{A21})$$

where the product over the set $J_i = \{j \in [k] \mid i_j = 1\}$ can be read as *all* $j \in \{1, \dots, k\}$ such that the j th digit of i in base 2 is 1. The commutator chain $[(A), (B)]_{\tau, i}$ can then be efficiently determined by the Lie-algebra structure tensor, which we will describe below. The time-dependent term that is the product of $d(t_{\circ})$ will be integrated under the chained integral defined from the tree τ .

3. Generating Lie algebras

The chain of commutators $[(A), (B)]_{\tau, i}$ can be efficiently computed *a priori* by generating an orthogonal basis of operators via the Gram-Schmidt process. The Lie-algebra set $\mathfrak{g}_k$ of order k contains all of the independent operators that are generated from the generator set $G = \{A, B\}$ up to commutation order k . The first-order set $\mathfrak{g}_1 = \{L_1, L_2\}$, where

$$L_1 = A, \quad L_2 = B - (\text{proj}_{L_1} B), \quad (\text{A22})$$

and where $\text{proj}_{L_1} B$ is the projector of operator B into L_1 . When B and L_1 are matrices, the projection is computed on the vectorized representations of B and L_1 . The set $\mathfrak{g}_k$ is generated recursively as follows:

$$\begin{aligned} \mathfrak{g}_{k+1} &= \{L_a : L_a = [g, L_b] - \text{proj}_{\{L_1, \dots, L_{a-1}\}}[g, L_b], \\ &L_b \in \mathfrak{g}_k, g \in G \setminus \{0\}, \end{aligned} \quad (\text{A23})$$

where $\text{proj}_{\{L_1, \dots, L_{a-1}\}}[g, L_b]$ is the projection of $[g, L_b]$ into the orthogonal space generated by $\{L_1, \dots, L_{a-1}\}$. The full Lie algebra generated by G is then the union of all of the sets of each order $\mathfrak{g} = \bigcup_{m=1}^M \mathfrak{g}_m$, where the recursive depth is at most M , where $\mathfrak{g}_M = \emptyset$. We say that if a term $Q \in \mathfrak{g}_k$, then Q shows up at commutation depth k on the Lie algebra. A cutoff is implemented when calculating this numerically, neglecting terms L_a where $\|L_a\|_1 < \varepsilon_L \min(\|A\|_1, \|B\|_1)$, with $\varepsilon_L = 10^{-5}$. For a finite number of qubits, the number of elements in $\mathfrak{g}$ is also finite, and may correspond to the full Hilbert space of operators.

The structure constant f_{ijk} of the Lie algebra describes all of the necessary relations between its elements:

$$[L_i, L_j] = \sum_{L_k \in \mathfrak{g}} f_{ijk} L_k, \quad \forall L_i, L_j \in \mathfrak{g}, \quad (\text{A24})$$

which is enough to compute all of the commutation relations that show up in ME. The calculation of f_{ijk} takes $O(\dim(\mathfrak{g})^2)$

commutator evaluations. Each of those is projected back to the Lie algebra $\mathfrak{g}$ using the Gram-Schmidt process. Any chain of commutators can be represented as an element of the Lie algebra:

$$[(A), (B)]_{\tau, i} = \sum_{L_\mu \in \mathfrak{g}} \beta_{\mu, i\tau} L_\mu. \quad (\text{A25})$$

See Appendix A 5 for an illustration of this calculation.

4. Polynomial representation

Now with the Lie-algebra relations, the integral over the polynomial function needs to be determined. For convenience,

we rewrite the sequence of integrals defined over the tree $\tau \in \mathbb{T}_{k-1}$ as

$$\begin{aligned} \int_0^t dt_{[\tau]} f(t_1, \dots, t_k) &:= \int_0^t dt_1 \int_0^{t_1} dt_2 \cdots \int_0^{t_{\tau_{i-1}}} dt_i \cdots \\ &\times \int_0^{t_{\tau_{k-1}}} dt_k f(t_1, \dots, t_k), \end{aligned} \quad (\text{A26})$$

with k being the number of leaves in the tree τ and t_{τ_i} being the same ones as defined above. Equation (A1) can be rewritten using Eqs. (A21) and (A25), yielding the ME at order k :

$$M_k(t) = (-i)^{k-1} \sum_{\mu} \left[\sum_{\tau \in \mathbb{T}_{k-1}} \alpha(\tau) \sum_{i=0}^{2^k-1} \beta_{\mu, i, \tau} \left(\int_0^t dt_{[\tau]} \prod_{j \in J_i} d(t_j) \right) \right] L_\mu. \quad (\text{A27})$$

Since we have written the control function

$$d(t) = \sum_{a=0}^m \frac{d_a}{a!} t^a \quad (\text{A28})$$

as a polynomial, the integral term becomes

$$\begin{aligned} \int_0^t dt_{[\tau]} \prod_{j \in J_i} d(t_j) &= \int_0^t dt_{[\tau]} \prod_{j \in J_i} \left(\sum_{a_j=0}^m \frac{d_{a_j}}{a_j!} (t_j)^{a_j} \right) = \sum_{a_{j_1}=0}^m \cdots \sum_{a_{j_{p_i}}=0}^m \left(\prod_{j \in J_i} d_{a_j} \right) \left(\int_0^t dt_{[\tau]} \prod_{j \in J_i} \frac{(t_j)^{a_j}}{a_j!} \right) \\ &= \sum_{(a_j)_{j \in J_i} \in \{0, \dots, m\}^{J_i}} \left(\prod_{j \in J_i} d_{a_j} \right) \left(\int_0^t dt_{[\tau]} \prod_{j \in J_i} \frac{(t_j)^{a_j}}{a_j!} \right), \end{aligned} \quad (\text{A29})$$

where a_j are the summation indices of the polynomial corresponding to $d(t_j)$, $J_i = \{j_1, \dots, j_{p_i}\}$ and $p_i = \dim(J_i)$. The product of sums,

$$\prod_{j \in J_i} \left(\sum_{a_j=0}^m f(a_j) \right) = \sum_{(a_j)_{j \in J_i} \in \{0, \dots, m\}^{J_i}} \prod_{j \in J_i} f(a_j), \quad (\text{A30})$$

has been rewritten as a multidimensional sum. For example, if $i = 101_2$, then $J_i = \{1, 3\}$, and the sum becomes

$$\sum_{(a_j)_{j \in \{1,3\}} \in \{0, \dots, m\}^{\{1,3\}}} \prod_{j \in \{1,3\}} f(a_j) = \sum_{a_1=0}^m \sum_{a_3=0}^m f(a_1) f(a_3). \quad (\text{A31})$$

We can rearrange the sum in the i and a_j by grouping together the product of the polynomial coefficients d_a with the same number of products between d_{a_j} . That is, the sum transforms

$$\sum_{i=0}^{2^k-1} \sum_{(a_j)_{j \in J_i} \in \{0, \dots, m\}^{J_i}} b_i \left(\prod_{j \in J_i} f(a_j) \right) g(i, (a_j)_{j \in J_i}) = \sum_{p=0}^k \sum_{\gamma_1=0}^m \cdots \sum_{\gamma_p=0}^m \left[\left(\prod_{p=1}^p f(\gamma_p) \right) \sum_{0 < q_1 < \dots < q_p \leq k} b_{q_1, \dots, q_p} g'(q_1, \dots, q_p, \gamma_1, \dots, \gamma_p) \right], \quad (\text{A32})$$

where we have the change of indices $\{q_1, \dots, q_p\} = J_i$, $\{\gamma_1, \dots, \gamma_p\} = (a_j)_{j \in J_i}$, and $p = \dim(J_i)$, and $g'(\cdot)$ returns the same as $g(\cdot)$ according to this index change,

$$g'(q_1, \dots, q_p, \gamma_1, \dots, \gamma_p) := g(i(q_1, \dots, q_p), \{\gamma_1, \dots, \gamma_p\}), \quad (\text{A33})$$

with $i(q_1, \dots, q_p)$ written in the binary representation

$$i(q_1, \dots, q_p) := i_k \dots i_1, \quad i_j(q_1, \dots, q_p) := \begin{cases} 1, & j \in \{q_1, \dots, q_p\} \\ 0, & \text{otherwise.} \end{cases} \quad (\text{A34})$$

In essence, these q 's indicate the position of the 1's in $i_{\text{base } 2}$. To give a simple example with $k = 3$, this sum corresponds to the following grouping of the terms:

$$\begin{aligned}
 & b_{000_2} g(0, \{\}) + b_{001_2} f(a_1) g(1, \{a_1\}) + b_{010_2} f(a_2) g(2, \{a_2\}) + b_{011_2} f(a_1) f(a_2) g(3, \{a_1, a_2\}) \\
 & + b_{100_2} f(a_3) g(4, \{a_3\}) + b_{101_2} f(a_1) f(a_3) g(5, \{a_1, a_3\}) + b_{110_2} f(a_2) f(a_3) g(6, \{a_2, a_3\}) \\
 & + b_{111_2} f(a_1) f(a_2) f(a_3) g(7, \{a_1, a_2, a_3\}) \\
 & = [b_{000_2} g(0, \{\})] \quad (p = 0) \\
 & + [b_{001_2} f(a_1) g(1, \{a_1\}) + b_{010_2} f(a_2) g(2, \{a_2\}) + b_{100_2} f(a_3) g(4, \{a_3\})] \quad (p = 1) \\
 & + [b_{011_2} f(a_1) f(a_2) g(3, \{a_1, a_2\}) + b_{101_2} f(a_1) f(a_3) g(5, \{a_1, a_3\}) + b_{110_2} f(a_2) f(a_3) g(6, \{a_2, a_3\})] \quad (p = 2) \\
 & + [b_{111_2} f(a_1) f(a_2) f(a_3) g(7, \{a_1, a_2, a_3\})], \quad (p = 3)
 \end{aligned} \tag{A35}$$

where the sums in a_1 , a_2 , and a_3 have been neglected for simplicity. The inner sum in Eq. (A27) can then be rewritten as the following:

$$\sum_{i=0}^{2^k-1} \beta_{\mu, i, \tau} \sum_{(a_j)_{j \in J_i}} \left(\prod_{j \in J_i} d_{a_j} \right) \int_0^t dt_{[\tau]} \prod_{j \in J_i} \frac{(t_j)^{a_j}}{a_j!} = \sum_{p=0}^k \sum_{\gamma_1, \dots, \gamma_p=0}^m t^k \left(\prod_{l=1}^p d_{\gamma_l} t^{\gamma_l} \right) \left[\sum_{0 < q_1 < \dots < q_p \leq k} \beta_{\mu, q_1 \dots q_p, \tau} \int_0^1 dx_{[\tau]} \prod_{l=1}^p \frac{(x_{q_l})^{\gamma_l}}{\gamma_l!} \right], \tag{A36}$$

where we have applied the change of integration variable $dt_i = t dx_i$, $i = 1, \dots, k$. Notice the reindexing from the binary representation of i to the q 's. The term within the square brackets is a coefficient that depends only on $\mu, \gamma_1 \dots \gamma_p, \tau$ for a specific tree τ , independent of the control function $d(t)$. Since $\beta_{\mu, \alpha_1 \dots \alpha_p, \tau}$ is a coefficient in this sum, this object also includes the properties of the Lie algebra. Including now the sum over the trees of order k results in the formula for the dynamical coefficient:

$$\mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)} = \frac{(-i)^{k-1}}{\gamma!} \sum_{\tau \in \mathbb{T}_{k-1}} \alpha(\tau) \sum_{0 < q_1 < \dots < q_p \leq k} \left(\beta_{\mu, q_1 \dots q_p, \tau} \int_0^1 dx_{[\tau]} \prod_{l=1}^p (x_{q_l})^{\gamma_l} \right). \tag{A37}$$

The method to systematically solve the integral term is described in Appendix A 6. The Magnus expansion term of order k then becomes

$$M_k(t, \vec{d}) = \sum_{\mu} \left[\sum_{p=0}^k \sum_{\gamma_1, \dots, \gamma_p} \mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)} t^k \left(\prod_{l=1}^p d_{\gamma_l} t^{\gamma_l} \right) \right] L_{\mu}, \tag{A38}$$

where the term within square brackets is simply a coefficient for the Lie-algebra element L_{μ} . One can consider $t < 1$ and neglect terms smaller than $O(t^{\Gamma})$; i.e., we truncate at chosen time order Γ . Thus, the tensor $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)}$ is computed for all γ 's such that $k + \sum_{i=1}^p \gamma_i \leq \Gamma$. This is a reasonable approximation as long as $\Gamma > m, k_M$. Furthermore, since the evaluation of the polynomial is independent of the ordering of d_{γ_i} , the sum over $\gamma_1, \dots, \gamma_p$ can be simplified to a sum where $\gamma_i \leq \gamma_{i+1}$ with $i = 1, \dots, p-1$, resulting in the dynamical coefficients $\mathbf{S}_{\mu, \vec{\gamma}}^{(k, p)}$. This yields an expression with a similar form,

$$a_{\mu} = \sum_{k=1}^{k_M} \sum_{p=0}^k \sum_{\vec{\gamma} \in \mathcal{G}_{\Gamma}^{(k, p)}} \mathbf{S}_{\mu, \vec{\gamma}}^{(k, p)} t^k \prod_{i=1}^p (d_{\gamma_i} t^{\gamma_i}), \tag{A39}$$

where the dynamical coefficient $\mathbf{S}_{\mu, \vec{\gamma}}^{(k, p)}$ is constructed from a sum over the permutations in $\vec{\gamma}$ of $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)}$, and

$$\mathcal{G}_{\Gamma}^{(k, p)} := \left\{ (\gamma_1, \dots, \gamma_p) \mid \gamma_i \in \mathbb{N}_0; \gamma_i \leq \gamma_{i+1}; k + \sum_{i=1}^p \gamma_i \leq \Gamma \right\}. \tag{A40}$$

5. Lie Algebra commutators over a tree

In this section we want to define the identity in Eq. (A25). After generating the independent Lie algebra and the corresponding structure tensor [Eqs. (A23) and (A24)], we can evaluate the recursive commutators defined over the trees in a similar fashion as with the chained integrals. Consider two operators A and B which belong to the Lie algebra $\mathfrak{g}$:

$$A = \sum_{L_{\mu} \in \mathfrak{g}} a_{\mu} L_{\mu}, \quad B = \sum_{L_{\mu} \in \mathfrak{g}} b_{\mu} L_{\mu}. \tag{A41}$$

Their commutation is easily evaluated using the Lie-algebra structure tensor:

$$\begin{aligned}
 [A, B] &= \sum_{\mu, \nu} a_{\mu} b_{\nu} [L_{\mu}, L_{\nu}] = \sum_{\alpha} \left(\sum_{\mu, \nu} f_{\mu\nu\alpha} a_{\mu} b_{\nu} \right) L_{\alpha} \\
 &= \sum_{\alpha} (ab)_{\alpha} L_{\alpha}.
 \end{aligned} \tag{A42}$$

Since the result belongs to the same algebra, this becomes a simple tensor contraction when evaluated numerically. An example of evaluating the recursive commutators defined over a tree is shown in Fig. 9.

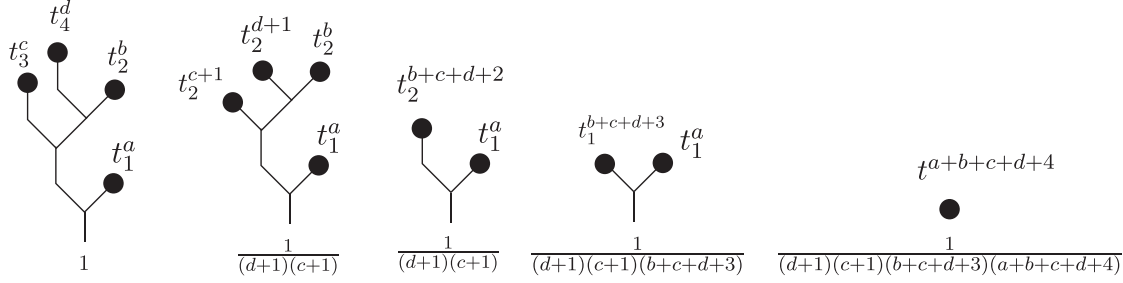


FIG. 10. Example of solving a chained integral defined from a binary tree. Powers of the integration time are placed at the leaves, and these are worked out downward, in topological order. In the first step, two integrals are resolved, introducing a coefficient, shown at the root of the tree. The powers are aggregated when there is a branch before computing the following integral. The end result is always t to the power of the sum of the leaf powers plus the number of leaves (identical to the order of the tree) with a coefficient which depends on the topological transversal of the binary tree.

6. Integration of polynomials over a tree

In this section we describe how the polynomial integrals

$$\int_0^t dt_{[\tau]} \prod_{l=1}^p (t_{q_l})^{\gamma_l} \quad (\text{A43})$$

can be computed systematically. When calculating a chained integral defined with a tree, we need to label each of the leaves of the tree, from 1 to k . Then the indices q_1 to q_p define where the powers $(t_{q_l})^{\gamma_l}$ go on the leaves of the tree. To solve the integral, we start from the leaves of the tree and work downward, in topological order, until the root of the tree is reached. Going down a vertical branch corresponds to evaluating the integral (see example in Fig. 10). Consider that q is the label of a leaf which is connected to a vertical branch, and p is the label of the leaf at the root of the vertical branch. Then the integral can be directly evaluated:

$$\int_0^{t_p} dt_q (t_q)^{\gamma_q} = \frac{1}{\gamma_q + 1} (t_p)^{\gamma_q + 1}. \quad (\text{A44})$$

This term would be embedded in the integral of leaf p ,

$$\frac{1}{\gamma_q + 1} \int_0^{t_a} dt_p (t_p)^{\gamma_q + 1} (t_p)^\alpha / \beta, \quad (\text{A45})$$

where a is the label of the leaf at the root of vertical branch p , α is an integer, and β a real coefficient computed from other branches. It is apparent that this new integral can be evaluated in a similar fashion, incrementing the power and updating the coefficient.

APPENDIX B: OPTIMAL CONTROL OF CONTINUOUS FUNCTIONS

1. Control derivatives of the Magnus expansion

The expressions for the derivatives are polynomials represented as tensor contractions, and are easily obtained from the original set of coefficients, $\mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)}$. The time derivative of a_μ is given by

$$\frac{\partial a_\mu}{\partial t} = \sum_{k=1}^{k_M} \sum_{p=0}^k \sum_{\gamma_1, \dots, \gamma_p} \tilde{\mathbf{T}}_{\mu, \vec{\gamma}}^{(k, p)} t^{k-1} \prod_{q=1}^p d_{\gamma_q} t^{\gamma_q}, \quad (\text{B1})$$

where

$$\tilde{\mathbf{T}}_{\mu, \vec{\gamma}}^{(k, p)} := \left(k + \sum_q \gamma_q \right) \mathbf{T}_{\mu, \vec{\gamma}}^{(k, p)}. \quad (\text{B2})$$

The derivative with respect to the control polynomial coefficients $\vec{d}$ is

$$\frac{\partial a_\mu}{\partial d_\alpha} = \sum_{k=1}^{k_M} \sum_{p=1}^k \sum_{\gamma_1, \dots, \gamma_{p-1}} \mathbf{D}_{\mu, \vec{\gamma}}^{(\alpha, k, p)} t^{k+\alpha} \prod_{q=1}^{p-1} d_{\gamma_q} t^{\gamma_q}, \quad (\text{B3})$$

where

$$\begin{aligned} \mathbf{D}_{\mu, \vec{\gamma}}^{(\alpha, k, p)} &:= \mathbf{T}_{\mu, (\alpha, \gamma_1, \dots, \gamma_{p-1})}^{(k, p)} + \mathbf{T}_{\mu, (\gamma_1, \alpha, \gamma_2, \dots, \gamma_{p-1})}^{(k, p)} \\ &+ \dots + \mathbf{T}_{\mu, (\gamma_1, \dots, \gamma_{p-1}, \alpha)}^{(k, p)} \end{aligned} \quad (\text{B4})$$

are the coefficients for the control derivative of the dynamical coefficients.

2. Pulse sequence derivatives

The pulse sequence $U(\vec{c})$ with the set of controls $\vec{c}$ is composed of S segments,

$$U(\vec{c}) = \prod_{s=1}^S U_s(\vec{c}), \quad (\text{B5})$$

where U_s is the propagator at segment s and $\vec{c}$ is a set of tunable parameters. The gradients of $U_s(\vec{c})$ are easy to evaluate when these are generated by Pauli operators. When the propagator $U_s(\vec{c})$ is generated by a dynamical evolution with singular control [Eq. (18)], we can use the derivatives of the ME we obtained above [Eqs. (B1) and (B3)]. More specifically, consider that U_s is generated from the evolution of a time-independent operator, similar to an effective Hamiltonian,

$$U_s(\vec{c}_s) = e^{-iH_s(\vec{c}_s)}, \quad (\text{B6})$$

where its derivative is

$$\partial_{\vec{c}_s} U_s = -iU_s(\vec{c}_s) \sum_{k=0}^{k_D \rightarrow \infty} \frac{i^k}{(k+1)!} \text{ad}_H^k(\partial_{\vec{c}_s} H_s), \quad (\text{B7})$$

with the adjoint action $\text{ad}_A^1(B) = [A, B]$ and $\text{ad}_A^{k+1}(B) = [A, \text{ad}_A^k(B)]$, for any operators A, B . This effective Hamiltonian $H_s(\vec{c}_s)$ can be approximated by the truncated ME when $t < t_*$, and so $H \simeq M^{(k_M)} =: M$. We consider that the derivative is accurately approximated when $k_D \geq k_M$, and thus

choose $k_D = k_M$. The derivatives $\partial_{\vec{c}} M$ have been obtained in Eqs. (B1) and (B3) when $\vec{c} = (t, d_0, \dots, d_m)$ describes a single control function $d(t)$ expressed as a polynomial in time. Since $\partial_{\vec{c}} M$ are expressed in terms of the Lie-algebra elements, evaluating the adjoint action can be done using only the structure constant f_{ijk} instead of the full Hilbert space.

3. Transformation to the Hermite spline

The control pulse $d(t)$ to be designed is manipulated using Hermite spline nodes. Consider that we have S spline segments. Then, there are $S + 1$ nodes placed at times $(0, t_1, \dots, t_S)$, where t_S is the total duration of the pulse. Each node

$$h^{(s)}(t - t_s) = \sum_{l=0}^L h_l^{(s)} \frac{(t - t_s)^l}{l!} \quad (\text{B8})$$

is a polynomial that matches the value and L first derivatives of $d(t_s)$:

$$h_l^{(s)} = \left. \frac{d^l}{dt^l} d(t) \right|_{t=t_s} \quad (l = 0, \dots, L). \quad (\text{B9})$$

The shape of the control pulse is fully parametrized with

$$\vec{h} = (\Delta t'_1, \dots, \Delta t'_S, h_0^{(0)}, \dots, h_L^{(S)}). \quad (\text{B10})$$

The segment duration is tuned with the time intervals $\Delta t'_s = t_s - t_{s-1}$, $s = 1, \dots, S$. To each of these segments, indexed by $s = 1, \dots, S$, there is a polynomial

$$d_s(t) = \sum_{n=0}^m d_n^{(s)} \frac{t^n}{n!}, \quad t \in [0, \Delta t_s], \quad (\text{B11})$$

that matches its derivatives at the past node $h_{s-1}(t)$ and the derivatives at future node $h_s(t)$, where $m = 2L + 1$. This leads to the constraints

$$\Delta t'_s = \Delta t_s \quad (s = 1, \dots, S), \quad (\text{B12})$$

$$h_l^{(s-1)} = d_l^{(s)} \quad (s = 1, \dots, S; l = 0, \dots, L), \quad (\text{B13})$$

$$h_l^{(s)} = \sum_{n=l}^m d_n^{(s)} \frac{(\Delta t_s)^{n-l}}{(n-l)!} \quad (s = 1, \dots, S; l = 0, \dots, L), \quad (\text{B14})$$

where Eqs. (B13) and (B14) are solved by constructing a linear system of equations. We can evaluate the ME and its derivatives using the segment duration and polynomial coefficients at each segment $\vec{c}_s(\vec{h}) = (\Delta t_s, d_0^{(s)}, \dots, d_m^{(s)})(\vec{h})$. These must be transformed into the polynomial coefficients $\vec{c}(\vec{h}) = (\vec{c}_1, \dots, \vec{c}_S)(\vec{h})$ so that M and $\partial_{\vec{c}} M$ can be evaluated at each segment. This is a change of coordinates $\vec{c}(\vec{h})$, where the Jacobian of the transformation is $\partial \vec{c} / \partial \vec{h}$ and is obtained from the constraints in Eqs. (B13) and (B14), such that the derivatives $\partial_{\vec{c}} U$,

$$\delta U = \left(\frac{\partial U}{\partial \vec{c}} \right)^T \cdot \delta \vec{c} = (\delta \vec{c})^T \cdot \left(\frac{\partial U}{\partial \vec{c}} \right), \quad (\text{B15})$$

are transformed into $\partial_{\vec{h}} U$. Since $\delta \vec{c} = (\partial \vec{c} / \partial \vec{h}) \cdot \delta \vec{h}$, we get

$$\delta U = (\delta \vec{h})^T \cdot \left(\frac{\partial \vec{c}}{\partial \vec{h}} \right)^T \cdot \left(\frac{\partial U}{\partial \vec{c}} \right) \Leftrightarrow \frac{\partial U}{\partial \vec{h}} = \left(\frac{\partial \vec{c}}{\partial \vec{h}} \right)^T \left(\frac{\partial U}{\partial \vec{c}} \right). \quad (\text{B16})$$

The variational of the constraint equations

$$\delta(\Delta t'_s) = \delta(\Delta t_s) \quad (s = 1, \dots, S), \quad (\text{B17})$$

$$\delta h_s^{(s-1)} = \delta d_l^{(s)} \quad (s = 1, \dots, S; l = 0, \dots, L), \quad (\text{B18})$$

$$\delta h_l^{(s)} = \sum_{n=l}^m \frac{(\Delta t_s)^{n-l}}{(n-l)!} \delta d_n^{(s)} + \left(\sum_{n=l+1}^m d_n^{(s)} \frac{(\Delta t_s)^{n-l-1}}{(n-l-1)!} \right) \delta(\Delta t_s) \quad (s = 1, \dots, S; l = 0, \dots, L), \quad (\text{B19})$$

is used for the calculation of the Jacobian $\partial \vec{c} / \partial \vec{h}$. This set of equations can be written as $A \delta \vec{h} = B \delta \vec{c}$ with A, B matrices obtained from these equations. Then, the Jacobian is simply

$$\frac{\partial \vec{c}}{\partial \vec{h}} = B^{-1} A, \quad (\text{B20})$$

where B^{-1} is the pseudoinverse of B .

-
- [1] A. Miessen, P. J. Ollitrault, and I. Tavernelli, Quantum algorithms for quantum dynamics: A performance study on the spin-boson model, *Phys. Rev. Res.* **3**, 043212 (2021).
 - [2] X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J. L. O'Brien, J. B. Wang, and J. C. F. Matthews, Efficient quantum walk on a quantum processor, *Nat. Commun.* **7**, 11511 (2016).
 - [3] V. Montenegro, C. Mukhopadhyay, R. Yousefjani, S. Sarkar, U. Mishra, M. G. A. Paris, and A. Bayat, Review: Quantum metrology and sensing with many-body systems, *Phys. Rep.* **1134**, 1 (2025).
 - [4] N. Aslam *et al.*, Quantum sensors for biomedical applications, *Nat. Rev. Phys.* **5**, 157 (2023).
 - [5] Z. Zou, J. Gong, and W. Chen, Quantum dynamics can push sensing performance to the limit, *Phys. Rev. Lett.* (2025).
 - [6] R. Ramya, P. Kumar, D. Dhanasekaran, R. S. Kumar, and S. A. Sharavan, A review of quantum communication and information networks with advanced cryptographic applications using machine learning, deep learning techniques, *Frankl. Open* **10**, 100223 (2025).
 - [7] C. Bauer *et al.*, Quantum simulation of fundamental particles and forces, *Nat. Rev. Phys.* **5**, 420 (2023).
 - [8] R. van der Meer *et al.*, Experimental simulation of loop quantum gravity on a photonic chip, *Nat. Quantum Inf.* **9**, 32 (2022).
 - [9] D. Gonzalez-Cuadra, M. Hamdan, T. V. Zache, B. Braverman, M. Kornjaca, A. Lukin, S. H. Cantu, F. Liu, S.-T. Wang, A. Keesling, M. D. Lukin, P. Zoller, and A. Bylinskii, Observation of string breaking on a $(2 + 1)$ D Rydberg quantum simulator, *Nature (London)* **642**, 321 (2024).

- [10] A. N. Ciavarella and C. W. Bauer, Quantum simulation of SU(3) lattice Yang-Mills theory at leading order in large- N_c expansion, *Phys. Rev. Lett.* **133**, 111901 (2024).
- [11] Q. Ansel, E. Dionis, F. Arrouas, B. Peaudecerf, S. Guérin, D. Guéry-Odelin, and D. Sugny, Introduction to theoretical and experimental aspects of quantum optimal control, *J. Phys. B: At. Mol. Opt. Phys.* **57**, 133001 (2024).
- [12] T. S. Mahesh, P. Batra, and M. H. Ram, Quantum optimal control: Practical aspects and diverse methods, [arXiv:2205.15574](https://arxiv.org/abs/2205.15574).
- [13] Q.-M. Chen, H. Rabitz, and R.-B. Wu, Quantum optimal control without arbitrary waveform generators, *Phys. Rev. Appl.* **20**, 064016 (2023).
- [14] J. F. Kam, H. Kang, C. D. Hill, G. J. Mooney, and L. C. L. Hollenberg, Characterization of entanglement on superconducting quantum computers of up to 414 qubits, *Phys. Rev. Res.* **6**, 033155 (2024).
- [15] Y. Dong, L. Lin, and Y. Tong, Ground-state preparation and energy estimation on early fault-tolerant quantum computers via quantum eigenvalue transformation of unitary matrices, *PRX Quantum* **3**, 040305 (2022).
- [16] N. Maskara *et al.*, Programmable simulations of molecules and materials with reconfigurable quantum processors, *Nat. Phys.* **21**, 289 (2023).
- [17] K. Wang *et al.*, Demonstration of low-overhead quantum error correction codes, *Nat. Phys.* **22**, 308 (2025).
- [18] R. Acharya *et al.*, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2024).
- [19] C. Poole, T. M. Graham, M. A. Perlin, M. Otten, and M. Saffman, Architecture for fast implementation of quantum low-density parity-check codes with optimized Rydberg gates, *Phys. Rev. A* **111**, 022433 (2025).
- [20] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
- [21] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [22] S. Chen, J. Cotler, H.-Y. Huang, and J. Li, The complexity of NISQ, *Nat. Commun.* **14**, 6001 (2023).
- [23] M. Saffman, Quantum computing with atomic qubits and Rydberg interactions: Progress and challenges, *J. Phys. B: At. Mol. Opt. Phys.* **49**, 202001 (2016).
- [24] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, Quantum computing with neutral atoms, *Quantum* **4**, 327 (2020).
- [25] M. Morgado and S. Whitlock, Quantum simulation and computing with Rydberg-interacting qubits, *AVS Quantum Sci.* **3**, 023501 (2021).
- [26] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye, A. M. Läuchli, and A. Browaeys, Quantum simulation of 2D antiferromagnets with hundreds of Rydberg atoms, *Nature (London)* **595**, 233 (2021).
- [27] S. Euchner and I. Lesanovsky, Rydberg atom arrays as quantum simulators for molecular dynamics, *Phys. Rev. Res.* **7**, L042009 (2025).
- [28] C. Evered, E. Lawrence, H. R. Bapat, A. Deller, J. Laurer, A. Paz, J. Zeiher, M. A. Norcia, and J. Bernien, High-fidelity parallel entangling gates on a neutral-atom quantum computer, *Nature (London)* **622**, 268 (2023).
- [29] G. Dirr and U. Helmke, Lie theory for quantum control, *GAMM Mitteilungen*, **31**, 59 (2008).
- [30] M. L. Goh, M. Larocca, L. Cincio, M. Cerezo, and F. Sauvage, Lie-algebraic classical simulations for quantum computing, *Phys. Rev. Res.* **7**, 033266 (2025).
- [31] J. Wei and E. Norman, Lie algebraic solution of linear differential equations, *J. Math. Phys.* **4**, 575 (1963).
- [32] B. C. Hall, *Lie Groups, Lie Algebras, and Representations*, Graduate Texts in Mathematics Vol. 222 (Springer International Publishing, Cham, 2015).
- [33] R. Wiersema, E. Kökcü, A. F. Kemper, and B. N. Bakalov, Classification of dynamical Lie algebras of 2-local spin systems on linear, circular and fully connected topologies, *npj Quantum Inf.* **10**, 110 (2024).
- [34] S. Qvarfort and I. Pikovski, Solving quantum dynamics with a Lie-algebra decoupling method, *PRX Quantum* **6**, 010201 (2025).
- [35] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. O. Marrero, M. Larocca, and M. Cerezo, A Lie algebraic theory of barren plateaus for deep parameterized quantum circuits, *Nat. Commun.* **15**, 7172 (2024).
- [36] S. Blanes, F. Casas, J. A. Oteo, and J. Ros, The Magnus expansion and some of its applications, *Phys. Rep.* **470**, 151 (2009).
- [37] K. F. Milfeld and R. E. Wyatt, Study extension and application of Floquet theory for quantum molecular systems in an oscillating field, *Phys. Rev. A* **27**, 72 (1983).
- [38] N. Auer, L. Einkemmer, P. Kandolf, and A. Ostermann, Magnus integrators on multicore CPUs and GPUs, *Comput. Phys. Commun.* **228**, 115 (2018).
- [39] A. Chakraborty, T. L. Patti, B. Khailany, A. N. Jordan, and A. Anandkumar, GPU-accelerated effective Hamiltonian calculator, *Quantum* **9**, 1946 (2025).
- [40] C. J. Budd and S. P. Nørsett, On the solution of linear differential equations in Lie groups, *Philos. Trans. R. Soc. London Ser. A* **357**, 983 (1999).
- [41] A. Alvermann and H. Fehske, High-order commutator-free exponential time-propagation of driven quantum systems, *J. Comput. Phys.* **230**, 5930 (2011).
- [42] P. A. M. Casares, M. S. Zini, and J. M. Arrazola, Quantum simulation of time-dependent Hamiltonians via commutator-free quasi-Magnus operators, *Quantum* **8**, 1567 (2024).
- [43] A. Arnal, F. Casas, and C. Chiralt, An efficient procedure to compute the continuous Baker–Campbell–Hausdorff formula, *Appl. Math. Comput.* **507**, 129563 (2025).
- [44] E. B. Dynkin, Calculation of the coefficients in the Campbell–Hausdorff formula, *Dokl. Akad. Nauk SSSR (N. S.)* **57**, 323 (1947).
- [45] R. Achilles and A. Bonfiglioli, The early proofs of the theorem of Campbell, Baker, Hausdorff, and Dynkin, *Arch. Hist. Exact Sci.* **66**, 295 (2012).
- [46] V. Gritsev and A. Polkovnikov, Integrable Floquet dynamics, *SciPost Phys.* **2**, 021 (2017).
- [47] J. Wu, J.-L. Wu, F.-Q. Guo, B.-B. Liu, S.-L. Su, X.-K. Song, L. Ye, and D. Wang, Quantum computation via Floquet tailored Rydberg interactions, *npj Quantum Inf.* **11**, 118 (2025).
- [48] M. Rodríguez-Vega, M. Vogl, and G. A. Fiete, Low-frequency and Moiré–Floquet engineering: A review, *Ann. Phys.* **435**, 168434 (2021).
- [49] M. Kalinowski, N. Maskara, and M. D. Lukin, Non-Abelian Floquet spin liquids in a digital Rydberg simulator, *Phys. Rev. X* **13**, 031008 (2023).

- [50] C. Weitenberg and J. Simonet, Tailoring quantum gases by Floquet engineering, *Nat. Phys.* **17**, 1342 (2021).
- [51] F. Wilczek, Quantum time crystals, *Phys. Rev. Lett.* **109**, 160401 (2012).
- [52] H. Liu, H. Cao, and S. Meng, Floquet engineering of topological states in realistic quantum materials via light-matter interactions, *Prog. Surf. Sci.* **98**, 100705 (2023).
- [53] T. Eckstein, R. Mansuroglu, P. Czarnik, J.-X. Zhu, M. J. Hartmann, L. Cincio, A. T. Sornborger, and Z. Holmes, Large-scale simulations of Floquet physics on near-term quantum computers, *npj Quantum Inf.* **10**, 84 (2024).
- [54] D. I. Bondar, L. B. Gaggioli, G. Korpas, J. Marecek, J. Vala, and K. Jacobs, Globally optimal control of quantum dynamics, *Phys. Rev. Res.* **7**, 043202 (2025).
- [55] L. B. Gaggioli, D. I. Bondar, J. Vala, R. Ovsiannikov, and J. Mareček, Unitary gate synthesis via polynomial optimization, *arXiv:2508.01356*.
- [56] 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).
- [57] In units where $\hbar = 1$.
- [58] *Ordered exponential*, Wikipedia (unpublished).
- [59] J. Bernoulli, *Ars Conjectandi* (Thurneisen, Basel, 1713).
- [60] P. C. Moan and J. Niesen, Convergence of the Magnus series, *Found. Comput. Math.* **8**, 291 (2008).
- [61] J. L. Allen, R. Kosut, J. Joo, P. Leek, and E. Ginossar, Optimal control of two qubits via a single cavity drive in circuit quantum electrodynamics, *Phys. Rev. A* **95**, 042325 (2017).
- [62] All of the time-sensitive benchmarking has been done using CPU multithreading, on a system with $2 \times$ Intel(R) Xeon(R) Gold 6240.
- [63] We use Verner's ninth-order method with 100 time steps, enough to keep the propagation error under floating-point accuracy.
- [64] J. H. Verner, Numerically optimal Runge–Kutta pairs with interpolants, *Numer. Algorithms* **53**, 383 (2010).
- [65] C. Rackauckas and Q. Nie, Differentialequations.jl—A performant and feature-rich ecosystem for solving differential equations in Julia, *J. Open Res. Software* **5**, 15 (2017).
- [66] J. M. Dominy and D. Leermakers, *et al.*, QUANTUMPROPAGATORS.JL: A Julia package for quantum state propagation, GitHub, 2025, <https://github.com/QuantumControl/QuantumPropagators.jl>.
- [67] E. S. Meckes, *The Random Matrix Theory of the Classical Compact Groups* (Cambridge University Press, Cambridge, UK, 2019).
- [68] S. Jandura and G. Pupillo, Time-optimal two- and three-qubit gates for Rydberg atoms, *Quantum*, **6**, 712 (2022).
- [69] M. Mohan, R. de Keijzer, and S. Kokkelmans, Robust control and optimal Rydberg states for neutral atom two-qubit gates, *Phys. Rev. Res.* **5**, 033052 (2023).
- [70] C. Fromenteil, R. Tricarico, F. Cesa, and H. Pichler, Hamilton-Jacobi-Bellman equations for Rydberg-blockade processes, *Phys. Rev. Res.* **6**, 033333 (2024).
- [71] M. Mohan, J. de Hond, and S. Kokkelmans, Parametrized multiqubit gates for neutral-atom quantum platforms, *Phys. Rev. Appl.* **23**, 054074 (2025).
- [72] A. Delakouras, G. Doultisinos, and D. Petrosyan, Multi-qubit Rydberg gates between distant atoms, *Quantum* **10**, 1990 (2025).
- [73] R. de Keijzer, L. Visser, O. Tse, and S. Kokkelmans, Consensus-based qubit configuration optimization for variational algorithms on neutral atom quantum systems, *npj Quantum Inf.* **11**, 186 (2025).
- [74] J.-L. Lagrange, *Mécanique Analytique* (La Veuve Desaint, 1788).
- [75] J. Fageot, S. Aziznejad, M. Unser, and V. Uhlmann, Support and approximation properties of Hermite splines, *arXiv:1902.02565*.
- [76] The inclusion of the $|0\rangle\langle 0|$ projector in the qutrit X operator is, such that its eigenspectrum is the same as the qubit X operator, keeping the property that $X^2 = \mathbb{I}$.
- [77] M. Mogensen and P. E. Farrell, OPTIM.JL: A mathematical optimization package for Julia, [software] GitHub, 2020, <https://github.com/JuliaNLSolvers/Optim.jl>.
- [78] The comparison has its limitations. We are solving for a different optimization problem. An intuition for this distinction comes from the fact that the derivative of the phase of the laser is its detuning, and as such any discontinuity of the phase leads to a Dirac δ function.
- [79] R. F. dos Santos, Magnus Tensor, 2025, <https://gitlab.tue.nl/20235021/magnustensor>.