Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework

S. Machnes,[1,2] U. Sander,[3] S. J. Glaser,[3] P. de Fouquières,[4] A. Gruslys,[4] S. Schirmer,[4] and T. Schulte-Herbrüggen[3,*]

[1]*Quantum Group, Department of Physics, Tel-Aviv University, Tel Aviv 69978, Israel*

[2]*Institute for Theoretical Physics, University of Ulm, D-89069 Ulm, Germany*

[3]*Department of Chemistry, Technical University of Munich (TUM), D-85747 Garching, Germany*

[4]*Department of Applied Mathematics and Theoretical Physics, University of Cambridge, CB3 0WA, United Kingdom*

(Received 7 December 2010; published 3 August 2011)

For paving the way to novel applications in quantum simulation, computation, and technology, increasingly large quantum systems have to be steered with high precision. It is a typical task amenable to numerical optimal control to turn the time course of pulses, i.e., piecewise constant control amplitudes, iteratively into an optimized shape. Here, we present a comparative study of optimal-control algorithms for a wide range of finite-dimensional applications. We focus on the most commonly used algorithms: GRAPE methods which update all controls concurrently, and Krotov-type methods which do so sequentially. Guidelines for their use are given and open research questions are pointed out. Moreover, we introduce a unifying algorithmic framework, DYNAMO (dynamic optimization platform), designed to provide the quantum-technology community with a convenient MATLAB-based tool set for optimal control. In addition, it gives researchers in optimal-control techniques a framework for benchmarking and comparing newly proposed algorithms with the state of the art. It allows a mix-and-match approach with various types of gradients, update and step-size methods as well as subspace choices. Open-source code including examples is made available at http://qlib.info.

DOI: 10.1103/PhysRevA.84.022305 PACS number(s): 03.67.Lx, 02.30.Yy, 02.60.Pn, 07.05.Dz

I. INTRODUCTION

For unlocking the inherent quantum treasures of future quantum technology, it is essential to steer experimental quantum dynamical systems in a fast, accurate, and robust way [1,2]. While the accuracy demands in quantum computation (the "error-correction threshold") may seem daunting at the moment, quantum simulation is far less sensitive.

In practice, using coherent superpositions as a resource is often tantamount to protecting quantum systems against relaxation without compromising accuracy. In order to tackle these challenging quantum engineering tasks, optimal control algorithms are establishing themselves as indispensable tools. They have matured from principles [3] and early implementations [4–6] via spectroscopic applications [7–9] to advanced numerical algorithms [10,11] for state-to-state transfer and quantum-gate synthesis [12] alike.

In engineering high-end quantum experiments, progress has been made in many areas including cold atoms in optical lattice potentials [13,14], trapped ions [15–21], and superconducting qubits [22,23] to name just a few. To back these advances, optimal control among numerical tools have become increasingly important, see, e.g., [24] for a recent review. For instance, near time-optimal control may take pioneering realizations of solid-state qubits which are promising candidates for a computation platform [25], from their fidelity limit to the decoherence limit [26]. More recently, open systems governed by a Markovian master equation have been addressed [27], and even smaller non-Markovian subsystems can be tackled if they can be embedded into a larger system that in turn interacts in a Markovian way with its environment [28]. Taking the concept of decoherence-free subspaces [29,30] to more realistic scenarios, avoiding decoherence in encoded subspaces [31] complements recent approaches of dynamic error correction [32,33]. Along these lines, quantum control is anticipated to contribute significantly to bridging the gap between quantum principles demonstrated in pioneering experiments and high-end quantum engineering [1,2].

A. Scope and focus

The schemes used to locate the optimal-control sequence within the space of possible sequences are varied. The values taken by the system controls over time may be parametrized by piecewise constant control amplitudes in the time domain or in frequency space [34], by splines or other methods. For specific aspects of the toolbox of quantum control, see, e.g., [11,12,16,26,28,31,35–45], while a recent review can be found in [46]. Here, we concentrate on piecewise constant controls in the time domain. For this parametrization of the control space, there are two well-established optimal-control approaches: Krotov-type methods [36,37,47,48] which update all controls within a single time slice once before proceeding on to the next time slice (cycling back to the first slice when done), and GRAPE-type methods [11] which update all controls in all time slices concurrently. Here we refer to the former as *sequential-update schemes* and to the latter as *concurrent-update schemes*.

Sequential methods have mainly been applied to provide control fields in (infinite-dimensional) systems of atomic and molecular optics characterized by energy potentials [36,37,49,50], while concurrent methods have mostly been applied to (finite-dimensional) qubit systems of spin nature [11,12], or to Josephson elements [26,28], ion traps [51,52], or two-dimensional-cavity grids in quantum electrodynamics

*tosh@tum.de

[53]. Here we compare sequential vs concurrent algorithms in finite-dimensional systems.

Both methods require a mechanism to control the selection of the next point to sample. For sequential-update methods, which perform a single or a few iterations per parameter subspace choice, first-order methods are most often used; yet for algorithms repeatedly modifying the same wide segment of parameter space at every iteration, second-order methods, such as the well-established one by Broyden-Fletcher-Goldfarb-Shanno (BFGS) [54], seem better suited. These choices, however, are by no means the final word and are the subject of on-going research.

Controlling quantum systems via algorithms on classical computers naturally comes with unfavorable scaling. Thus it is essential to optimize the code by minimizing the number of operations on matrices which scale with the system size, and by parallelizing computation on high-performance clusters. While elements of the latter have been accomplished [55], here we focus on the former.

To this end, we present a unifying programming framework, the DYNAMO platform, allowing us to combine different methods of subspace selection, gradient calculation, update controls, step-size controls, etc. The framework allows us to benchmark the various methods on a wide range of problems in common usage, allowing future research to quickly compare proposed methods to the current state of the art. It also makes significant strides toward minimizing the number of matrix operations required for serial, concurrent, and generalized hybrid schemes. The full MATLAB code of the platform is provided to the community at [56]. We benchmark Krotov-type algorithms and GRAPE algorithms over a selection of scenarios, giving the user of control techniques guidelines as to which algorithm is appropriate for which problem.

The paper is organized as follows. In Sec. II we provide a generalized algorithmic framework embracing the established algorithms GRAPE and Krotov as limiting cases. Section III shows how the formal treatment applies to concrete standard settings of optimizing state transfer and gate synthesis in closed and open quantum systems. In Sec. IV we compare the computational performance of concurrent vs sequential update algorithms for a number of typical test problems of synthesizing gates or cluster states. Computational performance is discussed in terms of costly multiplications and exponentials of matrices.

Section V provides an outlook on emerging guidelines as to which type of problem asks for which flavor of algorithm in order not to waste computation time. Finally, we point at a list of open research questions in the pursuit of which DYNAMO is anticipated to prove useful.

II. ALGORITHMIC SETTINGS

Most of the quantum-control problems boil down to a single general form, namely, steering a dynamic system following an internal drift under additional external controls, so as to maximize a given figure of merit. Because the underlying equation of motion is taken to be linear both in the drift as well as in the control terms, dynamic systems of this form are known as *bilinear control systems* (Σ)

$$\dot{X}(t) = -\left(A + \sum_{j=1}^{m} u_j(t) B_j\right) X(t), \tag{1}$$

with "state" $X(t) \in \mathbb{C}^N$, drift $A \in \mathrm{Mat}_N\mathbb{C}$, controls $B_j \in \mathrm{Mat}_N\mathbb{C}$, and control amplitudes $u_j(t) \in \mathbb{R}$. Defining the $A_u(t) := A + \sum_{j=1}^{m} u_j(t) B_j$ as generators, the formal solution reads

$$X(t) = \mathbb{T} \exp\left\{-\int_0^t d\tau A_u(\tau)\right\} X(0), \tag{2}$$

where $\mathbb{T}$ denotes Dyson's time ordering operator. In this work, the pattern of a bilinear control system will turn out to serve as a convenient unifying frame for applications in closed and open quantum systems, which thus can be looked upon as a variation of a theme.

A. Closed quantum systems

Throughout this work we study systems that are *fully controllable* [57–63], i.e., those in which—neglecting relaxation—every unitary gate can be realized. Finally, unless specified otherwise, we allow for unbounded control amplitudes.

Closed quantum systems are defined by the system Hamiltonian H_d as the only *drift term*, while the "switchable" *control Hamiltonians* H_j express external manipulations in terms of the quantum system itself, where each control Hamiltonian can be steered in time by its (here piecewise constant) *control amplitudes* $u_j(t)$. Thus one obtains a bilinear control system in terms of the controlled Schrödinger equations

$$|\dot{\psi}(t)\rangle = -i\left(H_d + \sum_{j=1}^{m} u_j(t) H_j\right) |\psi(t)\rangle, \tag{3}$$

$$\dot{U}(t) = -i\left(H_d + \sum_{j=1}^{m} u_j(t) H_j\right) U(t), \tag{4}$$

where the second identity can be envisaged as lifting the first one to an operator equation. For brevity we henceforth concatenate all Hamiltonian components and write

$$H_u(t) := H_d + \sum_{j=1}^{m} u_j(t) H_j. \tag{5}$$

Usually one wishes to absorb unobservable global phases by taking density-operator representations of states $\rho(t)$. Their time evolution is brought about by unitary conjugation $\widehat{U}(\cdot) := U(\cdot)U^\dagger \equiv \mathrm{Ad}_U(\cdot)$ generated by commutation with the Hamiltonian $\widehat{H}_u(\cdot) := [H_u, (\cdot)] \equiv \mathrm{ad}_{H_u}(\cdot)$. So in the projective representation in Liouville space, Eqs. (3) and (4) take the form

$$\dot{\rho}(t) = -i\,\widehat{H}_u \rho(t), \tag{6}$$

$$\frac{d}{dt}\widehat{U}(t) = -i\,\widehat{H}_u \widehat{U}(t). \tag{7}$$

It is now easy to accommodate dissipation to this setting.

B. Open quantum systems

Markovian relaxation can readily be introduced on the level of the equation of motion by the operator Γ, which may, e.g., take the standard form according to Gorini-Kossakowski-Sudarshan known as GKS-Lindblad form. Then the respective controlled master equations for state transfer and its lift for gate synthesis read

$$\dot{\rho}(t) = -(i\widehat{H}_u + \Gamma)\rho(t), \tag{8}$$

$$\dot{F}(t) = -(i\widehat{H}_u + \Gamma)F(t). \tag{9}$$

Here F denotes a *quantum map* in the general linear group $\mathrm{GL}(N^2)$. It is a linear image over all basis states of the Liouville space representing the open system, where henceforth $N := 2^n$ for an n-qubit system. Note that only in the case of $[\widehat{H}_u, \Gamma] = 0$ does the map $F(t)$ boil down to a mere contraction of the unitary conjugation $\widehat{U}(t)$. In the generic case, it is the intricate interplay of the respective coherent ($i\widehat{H}_u$) and incoherent (Γ) part of the time evolution [64] that ultimately entails the need for relaxation-optimized control based on the full knowledge of the master equation (9).

These scenarios are also summarized in Table I.

C. Figures of merit

In this work, we treat quality functions only depending on the final state $X(T)$ of the system without taking into account running costs, which, however, is no principal limitation.[1]

[1]Depending on the state of the system $X(t)$ over a time interval $[0,T]$ and on the control amplitudes $u(t)$, a general quality function may be formulated to take the form
$f := f_T(X(T),T) + \int_0^T f_0(X(t),t,u(t))dt$,
where $f_T(X(T),T)$ is the component solely depending on the final state of the system $X(T)$ and independent of the control amplitudes, while $f_0(X(t),t,u(t))$ collects the *running costs* usually depending on the amplitudes $u(t)$. In optimal-control and variational calculus, the general case ($f_T \neq 0, f_0 \neq 0$) is known as *problem of Bolza*, while the special case of zero running costs ($f_T \neq 0, f_0 = 0$) is termed *problem of Mayer*, and ($f_T = 0, f_0 \neq 0$) defines the *problem of Lagrange*. Henceforth, we will not take into account any running costs (thereby also allowing for unbounded control amplitudes). Thus here all our problems take the form of Mayer. On the other hand, many applications of Krotov-type algorithms have included explicit running costs [36,37,49,50] to solve problems of the Bolza form, which are also amenable to GRAPE (as has been shown in [11]). Yet, though well known, it should be pointed out again that a Bolza problem can always be transformed into a Mayer problem, and ultimately all three types of problems are equivalent [65], which can even be traced back to the precontrol era in the calculus of variations [66]. The implications for convergence of the respective algorithms are treated in detail in [72].

TABLE I. Bilinear quantum-control systems.

Setting and task $\dot{X}(t) = -(A + \sum_j u_j(t)B_j)X(t)$		Drift A	Controls B_j
Closed systems:			
Pure-state transfer	$X(t) = \lvert\psi(t)\rangle$	iH_0	iH_j
Gate synthesis I	$X(t) = U(t)$	iH_0	iH_j
State transfer	$X(t) = \rho(t)$	$i\widehat{H}_0$	$i\widehat{H}_j$
Gate synthesis II	$X(t) = \widehat{U}(t)$	$i\widehat{H}_0$	$i\widehat{H}_j$
Open systems:			
State transfer	$X(t) = \rho(t)$	$i\widehat{H}_0 + \Gamma$	$i\widehat{H}_j$
Map synthesis	$X(t) = F(t)$	$i\widehat{H}_0 + \Gamma$	$i\widehat{H}_j$

No matter whether the $X(t)$ in Eq. (1) denote states or gates, a common natural figure of merit is the projection onto the target in terms of the overlap

$$g = \frac{1}{\|X_{\text{target}}\|_2}\mathrm{tr}\{X_{\text{target}}^\dagger X(T)\}. \tag{10}$$

Depending on the setting of interest, one may choose as the actual figure of merit $f_{\text{SU}} := \mathrm{Re}\, g$ or $f_{\text{PSU}} := |g|$.

More precisely, observe there are two scenarios for realizing quantum gates or modules $U(T) \in \mathrm{SU}(N)$ with maximum trace fidelities: Let

$$g := \frac{1}{N}\mathrm{tr}\{U_{\text{target}}^\dagger U(T)\} \tag{11}$$

define the normalized overlap of the generated gate $U(T)$ with the target. Then the quality function

$$f_{\text{SU}} := \frac{1}{N}\mathrm{Re}\,\mathrm{tr}\{U_{\text{target}}^\dagger U(T)\} = \mathrm{Re}\, g \tag{12}$$

covers the case where overall global phases are respected, whereas if a global phase is immaterial [12], another quality function f_{PSU} applies, whose square reads

$$f_{\text{PSU}}^2 := \frac{1}{N^2}\mathrm{Re}\,\mathrm{tr}\{\widehat{U}_{\text{target}}^\dagger \widehat{U}(T)\} = |g|^2. \tag{13}$$

The latter identity is most easily seen [12] in the so-called vec representation [67] of ρ, where $\widehat{U} = \bar{U} \otimes U \in \mathrm{PSU}(N)$ (with $\bar{U}$ as the complex conjugate) recalling that the projective unitary group is $\mathrm{PSU}(N) = \frac{\mathrm{U}(N)}{\mathrm{U}(1)} = \frac{\mathrm{SU}(N)}{\mathbb{Z}_N}$. Now observe that $\mathrm{tr}\{(\bar{U} \otimes U)(\bar{V} \otimes V)\} = \mathrm{tr}\{\bar{U}\bar{V} \otimes UV\} = |\mathrm{tr}\{UV\}|^2$.

D. Core of the numerical algorithms: Concurrent and sequential

Since the equations of motion for closed and open quantum systems as well as the natural overlap-based quality functions are of common form, we adopt the unified frame for the numerical algorithms to find optimal steerings $\{u_j(t)\}$. To this end, we describe first- and second-order methods to iteratively update the set of control amplitudes in a unified way for bilinear control problems.

1. Discretizing time evolution

For algorithmic purposes, one discretizes the time evolution. To this end, the *control terms* B_j are switched by piecewise constant *control amplitudes* $u_j(t_k) \in \mathcal{U} \subseteq \mathbb{R}$ with $t_k \in [0,T]$, where T is a fixed final time and $\mathcal{U}$ denotes some subset of admissible control amplitudes. For simplicity, we henceforth assume equal discretized time spacing $\Delta t := t_k - t_{k-1}$ for all time slices $k = 1,2,\ldots,M$. So $T = M\Delta t$. Then the total generator (i.e., Hamiltonian or Lindbladian)

governing the evolution in the time interval $(t_{k-1},t_k]$ shall be labeled by its final time t_k as

$$A_u(t_k) := A + \sum_j u_j(t_k) B_j, \tag{14}$$

generating the propagator

$$X_k := e^{-\Delta t A_u(t_k)}, \tag{15}$$

which governs the controlled time evolution in the time slice $(t_{k-1},t_k]$. Next, we define as boundary conditions $X(0) := X_0$ and $X_{M+1} := X_{\text{target}}$. They specify the problem and are therefore discussed in more detail in Sec. III, Table II. A typical problem is unitary gate synthesis, where $X_0 \equiv \mathbb{I}$ and $X_{\text{target}} \equiv U_{\text{target}}$, whereas in a pure-state transfer $X_0 \equiv |\psi_0\rangle$ and $X_{\text{target}} \equiv |\psi\rangle_{\text{target}}$. In any case, the state of the system is given by the discretized evolution

$$X(t_k) = X_{k:0} := X_k X_{k-1} \cdots X_1 X_0. \tag{16}$$

Likewise, the state of the adjoint system also known as the *co-state* $\Lambda^\dagger(t_k)$ results from the backward propagation of $X_{M+1} \equiv X_{\text{target}}$

$$\begin{aligned}\Lambda^\dagger(t_k) &:= X^\dagger_{\text{target}} X_M X_{M-1} \cdots X_{k+1} \\ &= X^\dagger_{M+1} X_M X_{M-1} \cdots X_{k+1} =: X_{M+1:k+1},\end{aligned} \tag{17}$$

which is needed to evaluate the figure of merit here taken to be

$$f := \frac{1}{N} |\text{tr}\{\Lambda^\dagger(t_k) X(t_k)\}| = |\text{tr}\{X^\dagger_{\text{target}} X(T)\}| \; \forall k \tag{18}$$

as the (normalized) projection of the final state under controlled discretized time evolution up to time T onto the target state.

2. *Algorithmic steps*

With the above stipulations, one may readily characterize the core algorithm by the following steps, also illustrated in Fig. 1 and the flowchart in Fig. 2.

(0) Set initial control amplitudes $u_j^{(0)}(t_k) \in \mathcal{U} \subseteq \mathbb{R}$ for all times t_k with $k \in \mathcal{T}^{(0)} := \{1,2,\ldots,M\}$ then set counters $r = 0$, $q = 0$, $s = 1$; fix s_{limit} and f'.

(1) *Outer loop start*, enumerated by q: Unless $r = q = 0$, choose a selection of time slices, i.e., a subspace, $\mathcal{T}^{(q)}$, on which to perform the next stage of the search, which will update only $u_j^{(r)}(t_k^{(q)})$ for $t^{(q)}_{k\in\{1\cdots M^{(q)}\}} \in \mathcal{T}^{(q)}$.

(2) *Inner loop,* enumerated by s: Take one or more gradient-based steps within the subspace. Depending on subspacechoice, number of matrix operations may be reduced as compared to the naive implementation of the algorithm.

(3) *Exponentiate:* $X_k^{(r)} = e^{\Delta t A_u^{(r)}(t_k^{(q)})}$ for all $k \in \mathcal{T}^{(q)}$ with $A_u^{(r)}(t_k^{(q)}) := A + \sum_j u_j^{(r)}(t_k^{(q)}) B_j$.

(4) Compute goal function at some $k = \kappa$:

(5) *Forward propagation:* $X^{(r)}_{\kappa:0} := X_k^{(r)} X_{k-1}^{(r)} \cdots X_1^{(r)} X_0$

(6) *Backward propagation:* $\Lambda^{(r)\dagger}_{M+1:\kappa+1} := X^\dagger_{\text{target}} X_M^{(r)} X_{M-1}^{(r)} \cdots X_{k+1}^{(r)}$

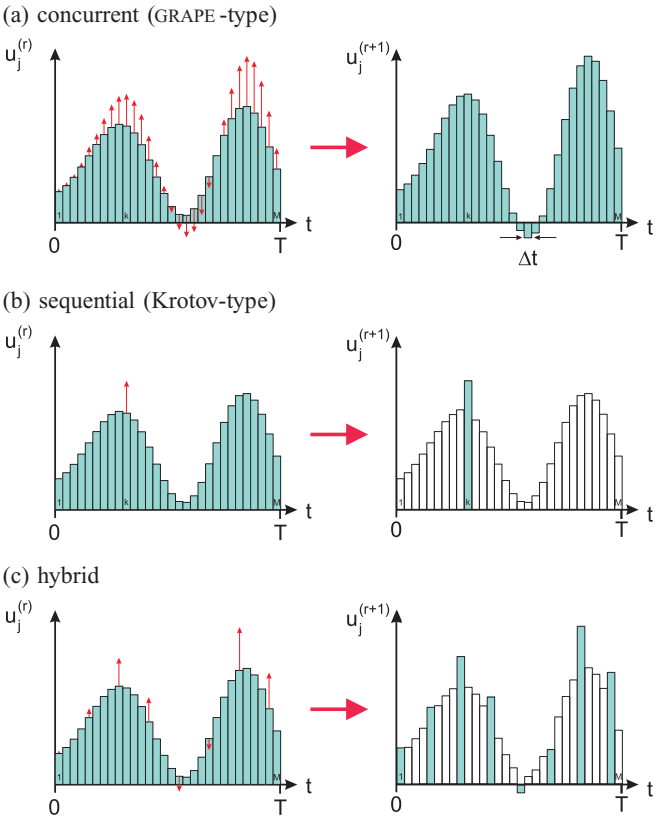

FIG. 1. (Color online) Overview of the update schemes of gradient-based optimal-control algorithms in terms of the set of time slices $\mathcal{T}^q = \{k_{1^q}, k_2^{(q)}, \ldots, k^{(q)}_{M^{(q)}}\}$ for which the control amplitudes are concurrently updated in each iteration. Subspacesare enumerated by q, gradient-based steps within each subspaceby s, and r is the global step counter. In GRAPE (a) all the M piecewise constant control amplitudes are updated at every step, so $\mathcal{T}^{(1)} = \{1,2,\ldots,M\}$ for the single iteration $q \equiv 1$. Sequential-update schemes (b) update a single time slice once, in the degenerate inner-loop $s \equiv 1$, before moving to the subsequent time slice in the outer loop, q; therefore here $\mathcal{T}^{(q)} = \{q \bmod M\}$. Hybrid versions (c) follow the same lines: for instance, they are devised to update a (sparse or block) subset of p different time slicesbefore moving to the next (disjoint) set of time slices.

(7) *Evaluate current fidelity:* $f^{(r)} = \frac{1}{N} |\text{tr}\{\Lambda^{(r)\dagger}_{M+1:\kappa+1} X^{(r)}_{\kappa:0}\}| = \frac{1}{N} |\text{tr}\{X^\dagger_{\text{tar}} X^{(r)}_{M:0}\}|$ for some k.

(8) If $f^{(r)} \geqslant 1 - \varepsilon_{\text{threshold}}$, done: go to step 13.

(9) Else, *calculate gradients* $\frac{\partial f^{(r)}[X^{(r)}(t_k^{(q)})]}{\partial u_j(t_k^{(q)})}$ for all $k \in \mathcal{T}^{(q)}$.

(10) *Gradient-based update* step: $u_j^{(r)}(t_k^{(q)}) \mapsto u_j^{(r+1)}(t_k^{(q)})$ for all $k \in \mathcal{T}^{(q)}$ by a method of choice (e.g., Newton, quasi-Newton, BFGS or L-BFGS, conjugate gradient, etc.).

(11) If $s < s_{\text{limit}}$ and $||\frac{\partial f_k^{(r)}}{\partial u_j}|| < f'_{\text{limit}}$ $\forall k \in \mathcal{T}_k^{(r)}$, then set and $s \longrightarrow s+1$, $r \longrightarrow r+1$ and return to step 3.

(12) $q \longrightarrow q+1$. Choose a new subspace $\mathcal{T}^{(q)}$ and return to step 2.

(13) *Output:* Final control vectors $\{u_j^{(r)}(t_k) | k = 1,2,\ldots,M\}$ for all controls j, final quality $f^{(r)}$, final state $X^{(r)}(T)$, and diagnostic output.

(14) *Terminate.*

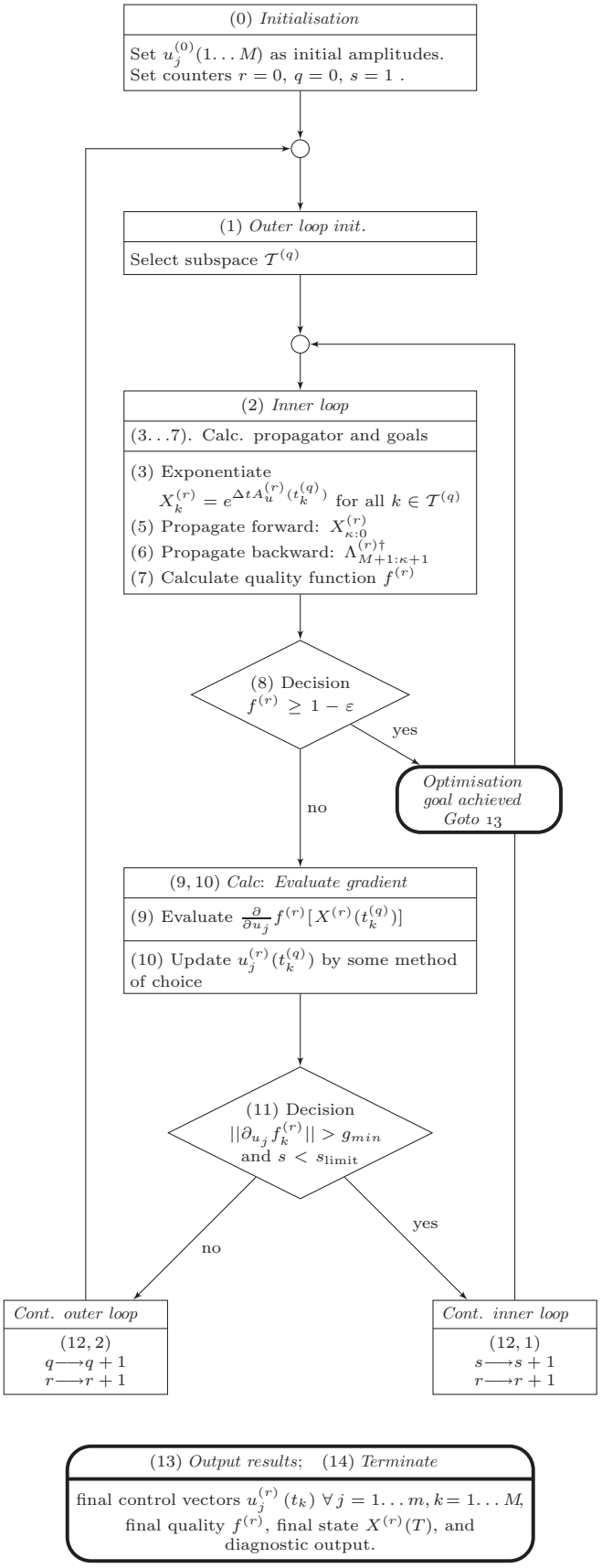

FIG. 2. Flow diagram for the generalized DYNAMO optimal-control search embracing standard GRAPE and Krotov methods as limiting special cases.

Having set the frame, one may now readily compare the Krotov and GRAPE approaches. In Krotov-type algorithms, we make use of a sequential-update scheme, where $\mathcal{T}^{(q)} = \{q \bmod M\}$ and $s_{\text{limit}} = 1$, implying the inner loop is degenerate, as only a single step is performed per subspace selection, giving $s \equiv 1, r = s$. With GRAPE, a concurrent-update scheme, $\mathcal{T}^{(q)} = \{1 \cdots M\}$, i.e., the entire parameter set is updated in each step of the inner loop, implying $q \equiv 1$, $r = s$ and the outer loop is degenerate. The above construction naturally invites hybrids: algorithms where the subspace size is arbitrary in the $1 \cdots M$ range and where the size of the subspace to be updated in each step q as well as the number of steps within each subspace, s, can vary dynamically with iteration, depending, e.g., on the magnitude of the gradient and the distance from the goal fidelity. This is a subject of ongoing research.

E. Overview of the DYNAMO package and its programming modules

DYNAMO provides a flexible framework for optimal-control algorithms with the purpose of allowing (i) quick and easy optimization for a given problem using the existing set of optimal-control search methods as well as (ii) flexible environment for development of and research into new algorithms.

For the first use case, the design goal is to make optimal-control techniques available to a broad audience, which is eased as DYNAMO is implemented in MATLAB. Thus to generate an optimized control sequence to a specific problem, one only needs modify one of the provided examples, specifying the drift and control Hamiltonians of interest, choose GRAPE, Krotov, or one of the other hybrid algorithms provided, and wait for the calculation to complete. Wall time, CPU time, gradient-size, and iteration-number constraints may also be imposed.

For the second use case—developing optimal-control algorithms—DYNAMO provides a flexible framework allowing researchers to focus on aspects of immediate interest, allowing DYNAMO to handle all other issues, as well as providing facilities for benchmarking and comparing performance of the new algorithms to the current cadre of methods.

1. Why a modular programming framework?

The explorative findings underlying this work make a strong case for setting up a programming framework in a modular way. They can be summarized as follows:

(a) There is no universal single optimal-control algorithm that serves all types of tasks at a time. For quantum computation, unitary gate synthesis, or state-to-state transfer of (non-) pure states require accuracies beyond the error-correction threshold, while for spectroscopy, improving the robustness of controls for state-to-state transfer may well come at the expense of lower maximal fidelities.

(b) Consequently, for a programming framework to be universal, it has to have a modular structure allowing one to switch between different update schemes (sequential, concurrent, and hybrid) with task-adapted parameter settings.

(c) In particular, the different update schemes have to be matched with the higher-order gradient module (conjugate gradients, Newton, quasi-Newton). For instance, with increasing dimension the inverse Hessian for a Newton-type algorithm

becomes computationally too costly to be still calculated exactly as one may easily afford to do in low dimensions. Rather, it is highly advantageous to approximate the inverse Hessian and the gradient *iteratively* by making use of previous runs within the same inner loop (see flow diagram to Fig. 1 in Fig. 2). This captures the spirit of the well-established limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) approach [54,68,69]. The pros of L-BFGS, however, are rather incompatible with restricting the number of inner loops to $s_{\max} = 1$ as is often done in sequential approaches. Therefore in turn, gradient modules scaling favorably with problem dimension may ask for matched update schemes.

(d) It is a common misconception to extrapolate from very few iterations needed for convergence in low dimensions that the same algorithmic setting will also perform best in high-dimensional problems. Actually, effective CPU time and number of iterations needed for convergence are far from being one-to-one. The same feature may be illustrated by recent results in the entirely different field of tensor approximation, where again in low dimensions, exact Newton methods outperform any other by the number of iterations as well as by CPU time, while in higher dimensions, exact Newton steps cannot be calculated at all (see Figs. 11.2–11.4 in Ref. [70]).

It is for these reasons we discuss the key steps of the algorithmic framework in terms of their constituent modules.

2. Gradient-based update modules

Here we describe the second-order and first-order control-update modules used by the respective algorithms.

Second-order (quasi-) Newton methods. The array of piecewise constant control amplitudes (in the rth iteration), $\{u_j^{(r)}(t_k^{(q)}) | j = 1,2,\ldots,m \text{ and } k = 1,2,\ldots,M^{(q)}\}$ are concatenated to a control vector written $|u^{(r)}\rangle$ for convenience (in slight abuse of notation). Thus the standard Newton update takes the form

$$|u^{(r+1)}\rangle = |u^{(r)}\rangle + \alpha_r \mathcal{H}_r^{-1} |\mathrm{grad}\, f^{(r)}\rangle. \tag{19}$$

Here α_r is again a step size and $\mathcal{H}_r^{-1}$ denotes the inverse Hessian, where $|\mathrm{grad}\, f^{(r)}\rangle$ is the gradient vector. For brevity we also introduce short-hand notation for the respective differences of control vectors and gradient vectors:

$$|x_r\rangle := |u^{(r+1)}\rangle - |u^{(r)}\rangle,$$
$$|y_r\rangle := |\mathrm{grad}\, f^{(r+1)}\rangle - |\mathrm{grad}\, f^{(r)}\rangle.$$

Now, in the BFGS standard algorithmic scheme [54], the inverse Hessian is conveniently approximated by making use of previous iterations via

$$\mathcal{H}_{r+1}^{-1} = V_r^t \mathcal{H}_r^{-1} V_r + \pi_r |x_r\rangle\langle x_r|, \tag{20}$$

with the definitions

$$\pi_r := \langle y_r | x_r \rangle^{-1}, \quad V_r := \mathbb{I} - \pi_r |y_r\rangle\langle x_r|.$$

By its recursive construction, BFGS (i) introduces time nonlocal information into the optimization procedure as soon as the inverse Hessian has off-diagonal components and (ii) perfectly matches *concurrent updates* within the inner loop: using second-order information makes up for its high initialization costs by iterating over the same subspace of controls throughout the optimization. Note that the MATLAB routine fminunc uses the standard BFGS scheme, while the routine fminconc uses the standard limited-memory variant L-BFGS [54,68,69,71]. Another advantage of the BFGS scheme is that the approximate Hessian is by construction positive definite, allowing straightforward Newton updates.

In contrast, for *sequential updates*, BFGS is obviously far from being the method of choice, because sequential updates iterate over a changing subset of controls. In principle, direct calculation of the Hessian is possible. However, this is relatively expensive and the local Hessian is not guaranteed to be positive definite, necessitating the need for more complex trust-region Newton updates. A detailed analysis of optimal strategies for sequential-update methods is necessary and is presented in [72]. Preliminary numerical data (see Sec. IV D) suggest that the gain from such higher-order methods for sequential-update schemes is limited and not sufficient to offset the increased computational costs per iteration in general. Thus we shall restrict ourselves here to sequential updates based on first-order gradient information.

First-order gradient ascent. The simplest case of a gradient-based sequential-update algorithm amounts to steepest ascent in the control vector, whose elements follow

$$u_j^{(r+1)}\left(t_k^{(q)}\right) = u_j^{(r)}\left(t_k^{(q)}\right) + \alpha_r \frac{\partial f^{(r)}}{\partial u_j\left(t_k^{(q)}\right)}, \tag{21}$$

where α_r is an appropriate step size or search length. For gate optimization problems of the type considered here it can be shown that sequential gradient update with suitable step-size control can match the performance of higher-order methods such as sequential Newton updates while avoiding the computational overhead of the latter [72]. Although choosing a small constant α_r ensures convergence (to a critical point of the target function) this is usually a bad choice. We can achieve much better performance with a simple heuristic based on a quadratic model $\alpha_r(2-\alpha_r)$ of f along the gradient direction in the step-size parameter α_r. Our step-size control is based on trying to ensure that the actual gain in the fidelity $\Delta f = f(\alpha_r) - f(0)$ is at least 2/3 of the maximum gain achievable based on the current quadratic model. Thus, we start with an initial guess for α_r, evaluate $\Delta f(\alpha_r)$ and use the quadratic model to estimate the optimal step size $\alpha_*(r)$. If the current α_r is less than 2/3 of the optimum step size then we increase α_r by a small factor, e.g., 1.01; if α_r is greater than 4/3 of the estimated optimal $\alpha_*(r)$ then we decrease α_r by a small factor, e.g., 0.99. Instead of applying the change in α_r immediately, i.e., for the current time step, which would require reevaluating the fidelity, we apply it only in the next time step to give

$$\alpha_{r+1} = \begin{cases} 1.01\alpha_r & \text{for} \quad \alpha_r < \frac{2}{3}\alpha_*(r) \\ 0.99\alpha_r & \text{for} \quad \alpha_r > \frac{4}{3}\alpha_*(r). \\ \alpha_r & \text{else} \end{cases} \tag{22}$$

For a sequential update with many time steps, avoiding the computational overhead of multiple fidelity evaluations is usually preferable to the small gain achieved by continually adjusting the step size α_r at the current time step. This deferred application of the step-size change is justified in our case as for unitary gate optimization problems of the type

considered here, as α_r usually quickly converges to an optimal (problem-specific) value and only varies very little after this initial adjustment period, regardless of the initial α_r [72].

As mentioned above, this step-size control scheme for sequential update comes close to a direct implementation of trust-region Newton (see Fig. 6 in Sec. IV D), a detailed analysis of which is given in [72].

3. Gradient modules

Exact gradients. In the module used for most of the subsequent comparisons, exact gradients to the exponential maps of total Hamiltonians with piecewise constant control amplitudes over the time interval Δt are to be evaluated. Here we use exact gradients as known from various applications [31,73]. Their foundations were elaborated in [74,75], so here we give a brief sketch along the lines of [73,74] (leaving more involved scenarios beyond piecewise constant controls to be dwelled upon elsewhere). For

$$X := \exp\{-i\,\Delta t\,H_u\} = \exp\left\{-i\,\Delta t(H_d + \sum_j u_j H_j)\right\}, \tag{23}$$

the derivative invokes the spectral theorem to take the form

$$\left\langle \lambda_l \left| \frac{\partial X}{\partial u_j} \lambda_m \right\rangle = \begin{cases} -i\,\Delta t\,\langle\lambda_l|H_j|\lambda_m\rangle e^{-i\,\Delta t\,\lambda_l} & \text{if} \quad \lambda_l = \lambda_m, \\ -i\,\Delta t\,\langle\lambda_l|H_j|\lambda_m\rangle \frac{e^{-i\,\Delta t\lambda_l} - e^{-i\,\Delta t\lambda_m}}{-i\,\Delta t(\lambda_l - \lambda_m)} & \text{if} \quad \lambda_l \neq \lambda_m, \end{cases} \tag{24}$$

where in the second identity we have deliberately kept the factor $-i\,\Delta t$ for clarity. Thus the derivative is given elementwise in the orthonormal eigenbasis $\{|\lambda_i\rangle\}$ to the real eigenvalues $\{\lambda_i\}$ of the Hamiltonian H_u. Details are straightforward, yet lengthy, and are thus relegated to Appendix A.

Approximate gradients. In Ref. [11] we took an approximation to be valid as long as the respective digitization time slices were small enough in the sense $\Delta t \ll 1/||H_u||_2$ with H_u as in Eq. (23)

$$\frac{\partial X}{\partial u_j} \approx -i\,\Delta t\,H_j e^{-i\,\Delta t\,H_u}. \tag{25}$$

This approximation can be envisaged as replacing the average value brought about by the time integral over the duration $\Delta t = t_k - t_{k-1}$, which in the above eigenbasis takes the form

$$\begin{aligned} &\left\langle \lambda_l \left| \frac{\partial X}{\partial u_j} \lambda_m \right\rangle \right. \\ &\quad = -i \int_{t_{k-1}}^{t_k} d\tau e^{-i\lambda_l(t_k-\tau)} \langle\lambda_l|H_j|\lambda_m\rangle e^{-i\lambda_m(\tau - t_{k-1})}, \\ &\quad \approx -i\,\Delta t\,\langle\lambda_l|H_j|\lambda_m\rangle e^{-i\lambda_m \Delta t}, \end{aligned} \tag{26}$$

by the value of the integrand at the right-hand side of the time interval $\tau \in [t_{k-1}, t_k]$. Clearly, this approximation ceases to be exact as soon as the time evolution $U(t_k, t_{k-1}) = e^{-i\,\Delta t\,H_u}$ fails to commute with H_j. Generically this is the case and the error scales with $|\lambda_l - \lambda_m|\Delta t$.

Finite differences. These provide another standard alternative, which may be favorable particularly in the case of pure-state transfer, see [76].

4. Exponentiation module

Matrix exponentials are a notorious problem in computer science [77,78]. Generically, the standard MATLAB module takes the matrix exponential via the Padé approximation, while in special cases (like the Hermitian one pursued throughout this paper) the eigendecomposition is used.[2]

From evaluating exact gradients (see above) the eigendecomposition of the Hamiltonian is already available. Though in itself the eigendecomposition typically comes at slightly higher computational overhead than the Padé matrix exponential, this additional computational cost is outweighed by the advantage that evaluating the matrix exponential now becomes trivial by exponentiation of the eigenvalues and a matrix multiplication.

Thus as long as the eigendecompositions are available, the matrix exponentials essentially come for free. Since in the sequential-update algorithm, the gradient needed for the exponential in time slice k requires an update in time slice $k-1$, the exponential occurs in the *inner loop* of the algorithm, while obviously the concurrent-update algorithm takes its exponentials only in the *outer loops*. The total number of exponentials required by the two algorithms are basically the same.

5. Reducing the number of matrix operations

As described above, the search for an optimal-control sequence proceeds on two levels: an outer loop choosing the time slices to be updated (a decision which may imply choice of gradient-based step method, as well as other control parameters), and an inner loop which computes gradients and advances the search point. With DYNAMO, significant effort has been made to optimize the overall number of matrix operations.

For a general hybrid scheme, where $\mathcal{T}^{(q)}$ is a subset of time slices $\{t_1^{(q)} \ldots t_{M^{(q)}}^{(q)}\}$, the approach is as follows. Given time slices $X_1, \ldots, X_M$, of which in hybrid update schemes we select for updating any general set $X_{t_1}, \ldots, X_{t_p}$, we can collapse multiple consecutive nonupdating X into a single effective Y. For example, consider $X_1, \ldots, X_{10}$ of which we update X_2, X_5, and X_6. Before proceeding with the inner loop, we generate concatenated products $Y_1, \ldots, Y_4$ such that $Y_1 = X_1$, $Y_2 = X_4 X_3$, and $Y_3 = X_{10} X_9 X_8 X_7$. Now the heart of the expression to optimize for is $Y_3 X_6 X_5 Y_2 X_2 Y_1$.

As a result, computation of forward and backward propagators can be done with the minimal number of matrix multiplications. Matrix exponentiation is also minimized by way of caching and making use of the fact that for some gradient computation schemes, eigen decomposition is required, thus allowing for light-weight exponentiation.

Moreover, the DYNAMO platform isolates the problem of minimizing matrix operations to a specific module, which is aware of which H_u, X, and Λ are needed for the next step, compares these with the time slices which have been updated, and attempts to provide the needed data with the minimal number of operations. And while for some hybrid

[2] In view of future optimization, however, note that our parallelized C++ version of GRAPE already uses faster methods based on Chebychev polynomials as described in [79,80].

update schemes the current number of operations performed in the outer loop is not strictly optimal in all cases, optimality is reached for Krotov, GRAPE, and schemes which update consecutive blocks of time slices.

6. *Modularization approach in* DYNAMO

To allow flexibility in the design and implementation of new optimal-control techniques, the framework is modularized by way of function pointers, allowing, e.g., the second-order search method to receive a pointer to a function which calculates the gradient, which in turn may receive a pointer to a function which calculates the exponential. The crossover algorithm described in Fig. 4, e.g., is implemented by a search method receiving as input two search-method modules and a crossover condition, which is used as a termination condition for the first search method. The first-order hybrids described in Fig. 8 are similarly implemented by a blockwise subspace selection function (generalization of the sequential vs concurrent selection schemes) receiving a pointer to the search function to be used within each block. DYNAMO is provided with many such examples.

If one is exploring, e.g., second-order search methods appropriate for serial update schemes, one only needs to write the update-rule function. DYNAMO will provide both the high-level subspace-selection logic and the low-level bookkeeping that is entrusted with tracking which controls have been updated. When given a demand for gradients, propagators, or the value function, it performs the needed calculations while minimizing the number of matrix operations. Moreover, once a new algorithm is found, DYNAMO makes it easy both to compare its performance to that of the many schemes already provided as examples and to do so for a wide set of problems described in this paper. Thus DYNAMO serves as a valuable benchmarking tool for current and future algorithms.

III. STANDARD SCENARIOS FOR QUANTUM APPLICATIONS

We have discussed the versatile features of the framework embracing all standard scenarios of bilinear quantum control problems listed in Table I. Here we give the (few!) necessary adaptations for applying our algorithms to such a broad variety of paradigmatic applications, while our test suite is confined to unitary gate synthesis and cluster-state preparation in closed quantum systems.

A. Closed quantum systems

The most frequent standard tasks for optimal control of closed systems comprise different ways of gate synthesis as well as state transfer of pure or nonpure quantum states. More precisely, sorted for convenient development from the general case, they amount to the following tasks:

Task 1. unitary gate synthesis up to a global phase
Task 2. unitary gate synthesis with fixed global phase
Task 3. state transfer among pure-state vectors
Task 4. state transfer among density operators.

TABLE II. Boundary conditions for standard scenarios. State of the system: evolution of initial state as $(X(t_k)) = X_{k:0} := X_k X_{k-1} \cdots X_1 \mathbf{X}_0$, with propagators $X_\nu = e^{\Delta t\left(A+\sum_j u_j(t_\nu)B_j\right)}$ for $\nu = 1,\ldots,k$ and with A, B_j as defined in Table I.

Conditions	Initial X_0	Final X_{M+1}
Closed systems:		
Pure-state transfer	$\|\psi_0\rangle$	$\|\psi\rangle_{\text{target}}$
Gate synthesis I	$\mathbb{1}_N$	U_{target}
State transfer	ρ_0	ρ_{target}
Gate synthesis II	$\mathbb{1}_{N^2}$	$\widehat{U}_{\text{target}}$
Open systems:		
State transfer	ρ_0	ρ_{target}
Map synthesis	$\mathbb{1}_{N^2}$	F_{target}

As will be shown, all of them can be treated by common propagators that are of the form

$$\begin{aligned} X_k &= \exp\{-i\,\Delta t\, H_u(t_k)\}, \\ &= \exp\left\{-i\,\Delta t\left(H_d + \sum_j u_j(t_k) H_j\right)\right\}. \end{aligned} \tag{27}$$

Algorithmically, this is very convenient, because then the specifics of the problem just enter via the boundary conditions as given in Table II; clearly, the data type of the state evolving in time via the propagators X_k is induced by the initial state being a vector or a matrix represented in Hilbert space or (formally) in Liouville space.

Indeed, for seeing interrelations, it is helpful to formally consider some problems in Liouville space, before breaking them down to a Hilbert-space representation for all practical purposes, which is obviously feasible in any closed system.

Task 1. Projective phase-independent gate synthesis. In Table II the target projective gate $\widehat{U}_{\text{target}}$ can be taken in the phase-independent superoperator representation $\widehat{X} := \bar{X} \otimes X$ to transform the quality function

$$\begin{aligned} f_{\text{PSU}}^2 &= \frac{1}{N^2}\text{Re tr}\{\widehat{U}_{\text{target}}^\dagger \widehat{X}(T)\} \\ &= \frac{1}{N^2}\text{Re tr}\left\{\left(U_{\text{target}}^t \bar{X}_T\right) \otimes \left(U_{\text{target}}^\dagger X_T\right)\right\} \\ &= \frac{1}{N^2}|\text{tr}\{U_{\text{tar}}^\dagger X_T\}|^2, \quad \text{so} \\ f_{\text{PSU}} &= \frac{1}{N}|\text{tr}\{U_{\text{tar}}^\dagger X_T\}| = \frac{1}{N}|\text{tr}\{\Lambda_{M+1:k+1}^\dagger X_{k:0}\}|, \end{aligned} \tag{28}$$

where the last identity recalls the forward and backward propagations $X(t_k) := X_k X_{k-1} \cdots X_2 X_1 X_0$ and $\Lambda^\dagger(t_k) := U_{\text{target}}^\dagger X_M X_{M-1} \cdots X_{k+2} X_{k+1}$.

So with the overlap $g := \frac{1}{N}\text{tr}\{\Lambda_{M+1:k+1}^\dagger U_{k:0}\}$ of Eq. (11), the derivative of the squared fidelity with respect to the control amplitude $u_j(t_k)$ becomes

$$\frac{\partial f_{\text{PSU}}^2(X(t_k))}{\partial u_j} = \frac{2}{N}\text{Re tr}\left\{g^* \Lambda_{M+1:k+1}^\dagger \left(\frac{\partial X_k}{\partial u_j}\right) X_{k-1:0}\right\}, \tag{29}$$

where $\frac{\partial X_k}{\partial u_j}$ is given by Eq. (24). The term g^* arises via $f^2(u) = |g(u)|^2$, so that by $\frac{\partial f^2}{\partial u} = 2|g(u)|[\frac{\partial}{\partial u}|g(u)|]$ one gets [for $|g(u)| \neq 0$] $\frac{\partial f}{\partial u} = \frac{\partial}{\partial u}|g(u)| = \frac{1}{2|g(u)|}\frac{\partial f^2}{\partial u}$ to arrive at

$$\frac{\partial f_{\mathrm{PSU}}(X(t_k))}{\partial u_j} = \frac{1}{N}\mathrm{Re\,tr}\left\{e^{-i\phi_g}\Lambda^\dagger_{M+1:k+1}\left(\frac{\partial X_k}{\partial u_j}\right)X_{k-1:0}\right\}, \tag{30}$$

where $e^{-i\phi_g} := g^*/|g|$ uses the polar form $g = |g|e^{+i\phi_g}$ for a numerically favorable formulation.

Thus, in closed systems, the superoperator representation is never used in the algorithm explicitly, yet it is instructive to apply upon derivation, because task 2 now follows immediately.

Task 2. Phase-dependent gate synthesis. In Table II the target gate U_{target} now directly enters the quality function

$$f_{\mathrm{SU}} = \frac{1}{N}\mathrm{Re\,tr}\{U^\dagger_{\mathrm{tar}}X_T\} = \frac{1}{N}\,\mathrm{Re\,tr}\{\Lambda^\dagger_{M+1:k+1}X_{k:0}\}. \tag{31}$$

So the derivative of the fidelity with respect to the control amplitude $u_j(t_k)$ with reference to $\frac{\partial X_k}{\partial u_j}$ of Eq. (24) reads

$$\frac{\partial f_{\mathrm{SU}}(X(t_k))}{\partial u_j} = \frac{1}{N}\mathrm{Re\,tr}\left\{\Lambda^\dagger_{M+1:k+1}\left(\frac{\partial X_k}{\partial u_j}\right)X_{k-1:0}\right\}. \tag{32}$$

It is in entirely analogous to Eq. (30).

Actually, this problem can be envisaged as the lifted operator version of the pure-state transfer in the subsequent task 3, which again thus follows immediately as a special case.

Task 3. Transfer between pure-state vectors. Target state and propagated initial state from Table II, $|\psi\rangle_{\mathrm{target}}$, $X(T)|\psi_0\rangle$ form the scalar product in the quality function

$$f = \frac{1}{N}\mathrm{Re}\langle\psi_{\mathrm{target}}|X_T\rangle = \frac{1}{N}\mathrm{Re}[\mathrm{tr}]\{\Lambda^\dagger_{M+1:k+1}X_{k:0}\}, \tag{33}$$

where the latter identity treats the propagated column vector $X_{k:1}|X_0\rangle$ as $N \times 1$ matrix $X_{k:0}$ and likewise the back-propagated final state $\langle\psi_{\mathrm{tar}}|(X_{M:k+1})^\dagger$ as $1 \times N$ matrix $\Lambda^\dagger_{M+1:k+1}$ so the trace can be ommited. Hence the derivative of the fidelity with respect to the control amplitude $u_j(t_k)$ remains

$$\frac{\partial f_{SU}(X(t_k))}{\partial u_j} = \frac{1}{N}\mathrm{Re}\,[\mathrm{tr}]\left\{\Lambda^\dagger_{M+1:k+1}\left(\frac{\partial X_k}{\partial u_j}\right)X_{k-1:0}\right\}, \tag{34}$$

with $\frac{\partial X_k}{\partial u_j}$ of Eq. (24).

Task 4. State transfer between density operators. The quality function normalized with respect to the (squared) norm of the target state $c := ||\rho_{\mathrm{tar}}||_2^2$ reads

$$\begin{aligned} f &= \frac{1}{c}\mathrm{Re\,tr}\{X^\dagger_{M+1}\mathrm{Ad}_{X_T}(X_0)\} \\ &\equiv \frac{1}{c}\mathrm{Re\,tr}\{X^\dagger_{M+1}X_TX_0X^\dagger_T\} \\ &= \frac{1}{c}\mathrm{Re\,tr}\{X^\dagger_{M+1}X_MX_{M-1}\cdots X_k\cdots X_2X_1X_0 \\ &\quad \times X^\dagger_1X^\dagger_2\cdots X^\dagger_k\cdots X^\dagger_{M-1}X^\dagger_M\}. \end{aligned} \tag{35}$$

Hence the derivative of the quality function with respect to the control amplitude $u_j(t_k)$ takes the somewhat lengthy form

$$\begin{aligned} \frac{\partial f(X(t_k))}{\partial u_j} = \frac{1}{c}\mathrm{Re}\Bigg\{&\mathrm{tr}\left[X^\dagger_{M+1}X_M\cdots\left(\frac{\partial X_k}{\partial u_j}\right)\cdots X_2X_1X_0\right. \\ &\left.\times X^\dagger_1X^\dagger_2\cdots X^\dagger_k\cdots X^\dagger_M\right] \\ &+\mathrm{tr}\left[X^\dagger_{M+1}X_M\cdots X_k\cdots X_2X_1X_0\right. \\ &\left.\times X^\dagger_1X^\dagger_2\cdots\left(\frac{\partial X^\dagger_k}{\partial u_j}\right)\cdots X^\dagger_M\right]\Bigg\}, \end{aligned} \tag{36}$$

where the exact gradient $\frac{\partial X_k}{\partial u_j}$ again follows Eq. (24).

Notice that task 1 can be envisaged as the lifted operator analog to task 4 if phase-independent projective representations $|\psi_\nu\rangle\langle\psi_\nu|$ of pure states $|\psi_\nu\rangle$ are to be transferred.

B. Open quantum systems

Task 5. Quantum map synthesis in Markovian systems. The superoperator $\widehat{H}_u(t_k)$ to the Hamiltonian above can readily be augmented by the relaxation operator Γ. Thus one obtains the generator to the quantum map

$$X_k = \exp\{-\Delta t[i\widehat{H}_u(t_k) + \Gamma(t_k)]\}, \tag{37}$$

following the Markovian equation of motion

$$\dot{X}(t) = -(i\widehat{H}_u + \Gamma)X(t). \tag{38}$$

By the (super)operators $X(t_k) := X_kX_{k-1}\cdots X_1X_0$ and $\Lambda^\dagger(t_k) := F^\dagger_{\mathrm{target}}X_M \times X_{M-1}\cdots X_{k+2}X_{k+1}$, the derivative of the trace fidelity at fixed final time T

$$f = \frac{1}{N^2}\mathrm{Re\,tr}\{F^\dagger_{\mathrm{target}}X(T)\} = \frac{1}{N^2}\mathrm{Re\,tr}\{\Lambda^\dagger(t_k)X(t_k)\}$$

with respect to the control amplitude $u_j(t_k)$ formally reads

$$\frac{\partial f}{\partial u_j(t_k)} = \frac{1}{N^2}\mathrm{Re\,tr}\left\{\Lambda^\dagger(t_k)\left(\frac{\partial X_k}{\partial u_j(t_k)}\right)X(t_{k-1})\right\}. \tag{39}$$

Since in general Γ and $i\widehat{H}_u$ do not commute, the semigroup generator $(i\widehat{H}_u + \Gamma)$ is not normal, so taking the exact gradient as in Eq. (24) via the spectral decomposition has to be replaced by other methods. There are two convenient alternatives, (i) approximating the gradient for sufficiently small $\Delta t \ll 1/||i\widehat{H}_u + \Gamma||_2$ by

$$\frac{\partial X_k}{\partial u_j(t_k)} \approx -\Delta t\left(i\widehat{H}_{u_j} + \frac{\partial\Gamma(u_j(t_k))}{\partial u_j(t_k)}\right)X_k \tag{40}$$

or (ii) via finite differences.

This standard task devised for Markovian systems [27] can readily be adapted to address also non-Markovian systems, provided the latter can be embedded into a (numerically manageable) larger system that in turn interacts with its environment in a Markovian way [28].

Task 6. State transfer in open Markovian systems. This problem can readily be solved as a special case of task 5 when envisaged as the vector version of it. To this end it is convenient to resort to the so-called vec notation [81] of a matrix M as the column vector $\mathrm{vec}(M)$ collecting all columns of M.

Now, identifying $X_0 := \mathrm{vec}(\rho_0)$ and $X^\dagger_{\mathrm{target}} := \mathrm{vec}^t(\rho^\dagger_{\mathrm{target}})$ one obtains the propagated initial state $X(t_k) := X_k X_{k-1} \cdots X_1 X_0$ and $\Lambda^\dagger(t_k) := X^\dagger_{\mathrm{target}} X_M X_{M-1} \cdots X_{k+2} X_{k+1}$ as the back-propagated target state. In analogy to task 3, they take the form of $N^2 \times 1$ and $1 \times N^2$ vectors, respectively. Thus the derivative of the trace fidelity at fixed final time T

$$f = \frac{1}{N}\mathrm{Re}[\mathrm{tr}]\{X^\dagger_{\mathrm{target}} X(T)\} = \frac{1}{N}\mathrm{Re}[\mathrm{tr}]\{\Lambda^\dagger(t_k) X(t_k)\}$$

with respect to the control amplitude $u_j(t_k)$ reads

$$\frac{\partial f}{\partial u_j(t_k)} = \frac{1}{N}\mathrm{Re}\ [\mathrm{tr}]\left\{\Lambda^\dagger(t_k)\left(\frac{\partial X_k}{\partial u_j(t_k)}\right) X(t_{k-1})\right\}, \qquad (41)$$

where for $\frac{\partial X_k}{\partial u_j(t_k)}$ the same gradient approximations apply as in task 5.

For the sake of completeness, Appendix B gives all the key steps of the standard tasks 1–6 in a nutshell.

IV. RESULTS ON UPDATE SCHEMES: CONCURRENT AND SEQUENTIAL

A. Specification of test problems

We studied the 23 systems listed in Table III as test problems for our optimization algorithms. This test suite includes spin chains, a cluster state system whose effective Hamiltonian represents a C_4 graph, a nitrogen-vacancy (N-V) center system and two driven spin-j systems with $j = 3,6$. Attempting to cover many systems of practical importance (spin chains, cluster-state preparation, N-V centers) with a range of coupling topologies and control schemes, the study includes large sets of parameters like system size, final time, number of time slices, and target gates. We therefore anticipate that our suite of test problems will provide good guidelines for choosing an appropriate algorithm in many practical cases.

1. Spin chains with individual local controls

Explorative problems 1–12 are Ising-ZZ spin chains of various length in which the spins are addressable by individual x and y controls. The Hamiltonians for these systems take the following form:

$$H_d = \frac{J}{2}\sum_{k=1}^{n-1} \sigma_k^z \sigma_{k+1}^z, \qquad (42)$$

$$H_j^{x,y} = \frac{1}{2}\sigma_j^{x,y}, \qquad (43)$$

where $J = 1$, $n = 1, \ldots, 5$, and $j = 1, \ldots, n$.

In example 1 we also consider linear cross-talk (e.g., via off-resonant excitation), leading to the control Hamiltonians

$$H_{1,2} = \alpha_{1,2}\sigma_1^x + \alpha_{2,1}\sigma_2^x, \qquad (44)$$

$$H_{3,4} = \beta_{2,1}\sigma_1^y + \beta_{1,2}\sigma_2^y, \qquad (45)$$

TABLE III. Specification of test problems. The notation *ABC* means the spin chain consists of three spins that are addressable each by an individual set of x and y controls. We write *A*0 for a locally controllable spin *A* which is coupled to a neighbor 0 not accessible by any control field.

Problem	Quantum system	Matrix dimensions	No. of time slices	Final time	Target gate
1	*AB* Ising-ZZ chain	4	30	2	CNOT
2	*AB* Ising-ZZ chain	4	40	2	CNOT
3	*AB* Ising-ZZ chain	4	128	3	CNOT
4	*AB* Ising-ZZ chain	4	64	4	CNOT
5	*ABC* Ising-ZZ chain	8	120	6	QFT
6	*ABC* Ising-ZZ chain	8	140	7	QFT
7	*ABCD* Ising-ZZ chain	16	128	10	QFT
8	*ABCD* Ising-ZZ chain	16	128	12	QFT
9	*ABCD* Ising-ZZ chain	16	64	20	QFT
10	*ABCDE* Ising-ZZ chain	32	300	15	QFT
11	*ABCDE* Ising-ZZ chain	32	300	20	QFT
12	*ABCDE* Ising-ZZ chain	32	64	25	QFT
13	C_4 Graph-ZZ	16	128	7	U_{CS}
14	C_4 Graph-ZZ	16	128	12	U_{CS}
15	NV center	4	40	2	CNOT
16	NV center	4	64	5	CNOT
17	*AAAAA* Ising-ZZ chain	32	1000	125	QFT
18	*AAAAA* Ising-ZZ chain	32	1000	150	QFT
19	*AAAAA* Heisenberg-XXX chain	32	300	30	QFT
20	*A*00 Heisenberg-XXX chain	8	64	15	rand U
21	*AB*00 Heisenberg-XXX chain	16	128	40	rand U
22	Driven spin-6 system	13	100	15	rand U
23	Driven spin-3 system	7	50	5	rand U

where u_k are independent control fields and α_k and β_k are crosstalk coefficients. We chose $\alpha_1 = \beta_2 = 1$ and $\alpha_2 = \beta_1 = 0.1$.

2. Cluster state preparation in completely coupled spin networks

The effective Hamiltonian of test problems 13 and 14,

$$H_{\mathrm{CS}} = \frac{J}{2}\left(\sigma_1^z\sigma_2^z + \sigma_2^z\sigma_3^z + \sigma_3^z\sigma_4^z + \sigma_4^z\sigma_1^z\right), \tag{46}$$

represents a C_4 graph of Ising-ZZ coupled qubits which can be used for cluster state preparation according to [82]. The underlying physical system is a completely Ising-coupled set of four ions that each represents a locally addressable qubit:

$$H_d = \frac{J}{2}\sum_{k=1}^{3}\sum_{l=k+1}^{4}\sigma_k^z\sigma_l^z, \tag{47}$$

$$H_j^{x,y} = \frac{1}{2}\sigma_j^{x,y} \quad (j = 1,\ldots,4). \tag{48}$$

Again, the coupling constant J was set to 1. The following unitary was chosen as a target gate, which applied to the state $|\psi_1\rangle = [(|0\rangle + |1\rangle)/\sqrt{2}]^{\otimes 4}$ generates a cluster state

$$U_G = \exp\left(-i\frac{\pi}{2}H_{\mathrm{CS}}\right). \tag{49}$$

3. NV center in isotopically engineered diamond

In test problems 15 and 16 we optimized for a controlled-NOT (CNOT) gate on two strongly coupled nuclear spins at a NV center in diamond as described in [83]. In the eigenbasis of the coupled system, after a transformation into the rotating frame, the Hamiltonians are of the form

$$H_d = \mathrm{diag}(E_1,E_2,E_3,E_4) + \omega_c\mathrm{diag}(1,0,0,-1), \tag{50}$$

$$H_1 = \tfrac{1}{2}\left(\mu_{12}\sigma_{12}^x + \mu_{13}\sigma_{13}^x + \mu_{24}\sigma_{24}^x + \mu_{34}\sigma_{3,4}^x\right), \tag{51}$$

$$H_2 = \tfrac{1}{2}\left(\mu_{12}\sigma_{12}^y + \mu_{13}\sigma_{13}^y + \mu_{24}\sigma_{24}^y + \mu_{34}\sigma_{34}^y\right). \tag{52}$$

Here $E_1,\ldots,E_4$ are the energy levels, ω_c is the carrier frequency of the driving field, and $\mu_{\alpha,\beta}$ is the relative dipole moment of the transition between levels α and β. We chose the following values for our optimizations: $\{E_1,E_2,E_3,E_4\} = 2\pi\{-134.825,-4.725,4.275,135.275\}$ MHz, $\omega_c = 2\pi \times 135$ MHz, $\{\mu_{12},\mu_{13},\mu_{24},\mu_{34}\} = \{1,1/3.5,1/1.4,1/1.8\}$ in accordance with [83].

4. Special applications of spin chains

Test problems 17 and 18 are modified five-qubit Ising chains extended by a local Stark-shift term being added in the drift Hamiltonian H_d resembling a gradient. The control consists of simultaneous x and y rotations on all spins

$$H_d = \frac{J}{2}\sum_{i=1}^{4}\sigma_i^z\sigma_{i+1}^z - (i+2)\sigma_i^z \tag{53}$$

$$H_1 = \frac{1}{2}\sum_{i=1}^{5}\sigma_i^x, \quad H_2 = \frac{1}{2}\sum_{i=1}^{5}\sigma_i^y. \tag{54}$$

Problem 19 is a Heisenberg-XXX coupled chain of five spins extended by global permanent fields inducing simultaneous x rotations on all spins:

$$H_d = \frac{J}{2}\sum_{i=1}^{4}\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z - 10\sigma_i^x. \tag{55}$$

Control is exerted by switchable local Stark-shift terms,

$$H_i = \sigma_i^z \qquad (i = 1,\ldots,5). \tag{56}$$

Spin chains may be put to good use as quantum wires [43,84–87]. The idea is to control just the input end of the chain using the remainder to passively transfer this input to the other end of the chain. To embrace such applications, in problems 20 and 21, the spins are coupled by an isotropic Heisenberg-XXX interaction and the chains are subject to x and y controls only at one end (at one or two spins, respectively):

$$H_d = \frac{J}{2}\sum_{i=1}^{n-1}\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z, \tag{57}$$

$$H_{1,2} = \tfrac{1}{2}\sigma_1^{x,y}, \quad H_{3,4} = \tfrac{1}{2}\sigma_2^{x,y}. \tag{58}$$

Here $J = 1$ and $n = 3,4$. Restricting the controls in this way makes the systems harder to steer and thus raises the bar for numerical optimization.

5. Spin-3 and spin-6 systems

As an example beyond spin-1/2 systems, in test problems 22 and 23 we consider a Hamiltonian of the following form [88]

$$H_u = J_z^2 + u_1J_z + u_2J_x, \tag{59}$$

where the J_i are angular momentum operators in spin-j representation. The J_z^2 term represents the drift Hamiltonian and the other two terms function as controls. We chose $j = 6$ for problem 22 and $j = 3$ for problem 23.

B. Test details

As shown in Table III, we optimized each test system for one of four quantum gates: a CNOT, a quantum Fourier transformation (QFT), a random unitary, or a unitary for cluster state preparation according to Sec. IV A 2. Random unitary gates generated according to the Haar measure [89] are meant to be numerically more demanding than the other gates. The final times T were always chosen sufficiently long to ensure the respective problem is solvable with full fidelity (hence the times should not be mistaken as underlying time-optimal solutions). All results were averaged over 20 runs with different initial pulse sequences (control vectors), i.e. randomly generated vectors with a mean value of $\mathrm{mean}(u_{\mathrm{ini}}) = 0$ and a standard deviation of $\mathrm{std}(u_{\mathrm{ini}}) = 1$ in units of $1/J$ unless specified otherwise [as in Table V, where $\mathrm{std}(u_{\mathrm{ini}}) = 10$ to study the influence of the initial conditions]. The maximum number of loops was set to 3000 for the concurrent-update scheme and to 300 000 for the sequential update. All systems were optimized with a target fidelity of $f_{\mathrm{target}} = 1$–10^{-4}. As an additional stopping criterion the change of the function value from one iteration to the next (concurrent update) or between the last iteration and the

TABLE IV. Test results obtained from 20 unconstrained optimizations (fminunc in MATLAB) for each problem of Table III using the sequential- or concurrent-update algorithm. Small initial pulse amplitudes were used [mean(u_{ini}) = 0, std(u_{ini}) = 1].

Problem	Algorithm	Final fidelity (**mean** | min | max)	Wall time (min) (**mean** | min | max)	No. eigendecs/1000 (**mean** | min | max)	No. matrix mults/1000 (**mean** | min | max)
1	Conc.	**0.9999** | 0.9999 | 1.0000	**0.02** | 0.01 | 0.03	**2.02** | 1.35 | 2.94	**38** | 25 | 56
	seq.	**0.9999** | 0.9999 | 0.9999	**0.19** | 0.13 | 0.34	**6.29** | 4.36 | 11.68	**88** | 61 | 163
2	Conc.	**0.9999** | 0.9999 | 1.0000	**0.05** | 0.03 | 0.08	**2.68** | 1.76 | 4.44	**50** | 33 | 84
	seq.	**0.9999** | 0.9999 | 0.9999	**0.16** | 0.11 | 0.27	**5.43** | 3.80 | 9.04	**76** | 53 | 126
3	Conc.	**0.9999** | 0.9999 | 1.0000	**0.05** | 0.04 | 0.08	**4.61** | 3.46 | 7.04	**85** | 63 | 132
	seq.	**0.9999** | 0.9999 | 0.9999	**0.07** | 0.05 | 0.12	**2.29** | 1.56 | 4.12	**32** | 22 | 57
4	Conc.	**0.9999** | 0.9999 | 1.0000	**0.02** | 0.01 | 0.02	**1.70** | 1.28 | 2.43	**31** | 23 | 45
	seq.	**0.9999** | 0.9999 | 0.9999	**0.05** | 0.03 | 0.11	**1.72** | 1.08 | 3.84	**24** | 15 | 54
5	Conc.	**0.9978** | 0.9973 | 0.9990	**7.19** | 5.84 | 7.86	**362** | 310 | 367	**9774** | 8364 | 9917
	seq.	**0.9973** | 0.9918 | 0.9986	**34** | 22 | 58	**1976** | 1320 | 3292	**35542** | 23729 | 59209
6	Conc.	**0.9999** | 0.9999 | 0.9999	**0.85** | 0.34 | 2.21	**35** | 17 | 76	**954** | 450 | 2050
	seq.	**0.9999** | 0.9999 | 0.9999	**5.14** | 1.14 | 18.72	**310** | 68 | 1143	**5574** | 1216 | 20554
7	Conc.	**0.9970** | 0.9886 | 0.9999	**9.42** | 2.32 | 18.48	**229** | 63 | 391	**8028** | 2210 | 13679
	seq.	**0.9945** | 0.9825 | 0.9999	**242** | 50 | 491	**3975** | 1242 | 7753	**87385** | 27313 | 170455
8	Conc.	**0.9999** | 0.9999 | 0.9999	**2.90** | 0.83 | 10.39	**72** | 21 | 275	**2530** | 735 | 9627
	seq.	**0.9999** | 0.9999 | 0.9999	**16.11** | 2.36 | 65.72	**500** | 89 | 2223	**11002** | 1953 | 48876
9	Conc.	**0.9999** | 0.9999 | 0.9999	**0.61** | 0.33 | 0.92	**20** | 11 | 30	**685** | 381 | 1052
	seq.	**0.9999** | 0.9999 | 0.9999	**4.83** | 1.91 | 7.81	**161** | 63 | 259	**3536** | 1375 | 5696
10	Conc.	**0.9982** | 0.9740 | 0.9999	**376** | 12 | 918	**435** | 82 | 917	**18694** | 3510 | 39442
	seq.	**0.9959** | 0.9661 | 0.9999	**2591** | 244 | 8458	**13123** | 1312 | 40136	**341107** | 34116 | 1043279
11	Conc.	**0.9999** | 0.9991 | 0.9999	**148** | 11 | 1236	**189** | 71 | 919	**8114** | 3045 | 39519
	seq.	**0.9998** | 0.9988 | 0.9999	**786** | 72 | 4817	**4041** | 427 | 17767	**105031** | 11097 | 461821
12	Conc.	**0.9996** | 0.9974 | 0.9999	**62.22** | 4.48 | 286.60	**89** | 22 | 192	**3818** | 942 | 8276
	seq.	**0.9994** | 0.9956 | 0.9999	**284** | 56 | 1842	**987** | 245 | 4563	**25637** | 6360 | 118491
13	Conc.	**0.9989** | 0.9936 | 0.9999	**5.20** | 1.52 | 14.89	**138** | 41 | 390	**4833** | 1434 | 13634
	seq.	**0.9759** | 0.9373 | 0.9999	**129.00** | 6.39 | 439.92	**4174** | 215 | 15103	**91773** | 4719 | 332029
14	Conc.	**0.9999** | 0.9999 | 0.9999	**1.45** | 0.70 | 2.62	**35** | 19 | 60	**1235** | 677 | 2089
	seq.	**0.9999** | 0.9999 | 0.9999	**6.47** | 1.76 | 16.06	**219** | 59 | 547	**4813** | 1292 | 12033
15	Conc.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.00 | 0.01	**0.90** | 0.64 | 1.24	**9.57** | 6.67 | 13.30
	seq.	**0.9999** | 0.9999 | 1.0000	**0.02** | 0.01 | 0.04	**1.76** | 0.72 | 3.28	**17.53** | 7.16 | 32.64
16	Conc.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.00 | 0.01	**0.80** | 0.70 | 1.28	**8.19** | 7.13 | 13.48
	seq.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.01 | 0.02	**0.67** | 0.51 | 1.54	**6.64** | 5.10 | 15.31
17	Conc.	**0.9999** | 0.9999 | 0.9999	**160** | 27 | 357	**684** | 616 | 773	**7516** | 6767 | 8495
	seq.	**0.9999** | 0.9999 | 0.9999	**2582** | 1411 | 4638	**16577** | 11490 | 27082	**165733** | 114877 | 270766
18	Conc.	**0.9999** | 0.9999 | 0.9999	**88** | 13 | 220	**394** | 286 | 620	**4330** | 3137 | 6811
	seq.	**0.9999** | 0.9999 | 0.9999	**492** | 247 | 1520	**2954** | 2434 | 3985	**29535** | 24335 | 39842
19	Conc.	**0.9999** | 0.9999 | 0.9999	**45.85** | 8.49 | 213.95	**170** | 103 | 264	**3896** | 2354 | 6060
	seq.	**0.9999** | 0.9999 | 0.9999	**128** | 76 | 217	**1124** | 809 | 1490	**17978** | 12945 | 23822
20	Conc.	**0.9999** | 0.9999 | 0.9999	**0.06** | 0.04 | 0.08	**6.92** | 4.80 | 9.47	**76** | 52 | 104
	seq.	**0.9999** | 0.9999 | 0.9999	**0.45** | 0.21 | 1.02	**26** | 15 | 43	**258** | 148 | 431
21	Conc.	**0.9999** | 0.9999 | 0.9999	**0.18** | 0.16 | 0.20	**8.56** | 7.81 | 9.60	**161** | 146 | 180
	seq.	**0.9999** | 0.9999 | 0.9999	**1.26** | 0.82 | 2.76	**39** | 29 | 57	**551** | 410 | 804
22	Conc.	**0.9999** | 0.9999 | 0.9999	**0.96** | 0.47 | 2.02	**68** | 42 | 105	**750** | 459 | 1154
	seq.[a]	**0.9998** | 0.9994 | 0.9999	**407** | 112 | 732	**21692** | 6473 | 30000	**216483** | 64599 | 299399
23	Conc.	**0.9999** | 0.9999 | 0.9999	**0.60** | 0.24 | 1.64	**53** | 25 | 141	**588** | 279 | 1559
	seq.	**0.9951** | 0.9797 | 0.9995	**39.03** | 9.39 | 111.74	**2992** | 744 | 7163	**29796** | 7408 | 71343

[a]Here the stopping conditions were changed for Fig. 3, so the data are no longer comparable to Tables V and VI.

TABLE V. Same as Table IV, but with higher initial pulse amplitudes: mean (u_{ini}) = 0, std (u_{ini}) = 10.

Problem	Algorithm	Final fidelity (**mean** | min | max)	Wall time (min) (**mean** | min | max)	No. eigendecs/1000 (**mean** | min | max)	No. matrix mults/1000 (**mean** | min | max)
1	Conc.	**0.9999** | 0.9999 | 0.9999	**0.04** | 0.02 | 0.06	**4.25** | 2.61 | 6.60	**80** | 49 | 125
	Seq.	**0.9999** | 0.9998 | 0.9999	**1.54** | 0.43 | 3.78	**118** | 33 | 289	**1645** | 459 | 4023
2	Conc.	**0.9999** | 0.9999 | 0.9999	**0.05** | 0.02 | 0.08	**5.39** | 2.76 | 8.56	**102** | 52 | 162
	Seq.	**0.9999** | 0.9999 | 0.9999	**1.38** | 0.42 | 3.28	**109** | 33 | 261	**1520** | 464 | 3635
3	Conc.	**0.9999** | 0.9999 | 1.0000	**0.07** | 0.05 | 0.10	**6.06** | 4.35 | 8.45	**113** | 80 | 158
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.22** | 0.12 | 0.49	**17.29** | 9.73 | 38.53	**242** | 136 | 539
4	Conc.	**0.9999** | 0.9999 | 1.0000	**0.02** | 0.01 | 0.03	**2.50** | 1.60 | 3.39	**46** | 29 | 63
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.18** | 0.05 | 0.54	**13.79** | 4.22 | 42.18	**193** | 59 | 589
5	Conc.	**0.9976** | 0.9959 | 0.9986	**6.97** | 6.24 | 7.32	**364** | 362 | 370	**9839** | 9784 | 9995
	Seq.	**0.9969** | 0.9952 | 0.9983	**73** | 51 | 105	**4246** | 2954 | 6021	**76349** | 53121 | 108271
6	Conc.	**0.9999** | 0.9999 | 0.9999	**2.23** | 1.36 | 3.82	**105** | 60 | 180	**2842** | 1623 | 4860
	Seq.	**0.9999** | 0.9999 | 0.9999	**16** | 11 | 32	**935** | 614 | 1866	**16823** | 11049 | 33567
7	Conc.	**0.9893** | 0.9366 | 0.9999	**15** | 14 | 16	**386** | 385 | 387	**13499** | 13469 | 13563
	Seq.	**0.9928** | 0.9444 | 0.9993	**325** | 129 | 730	**8053** | 4190 | 16630	**177047** | 92125 | 365595
8	Conc.	**0.9998** | 0.9984 | 0.9999	**9.86** | 4.27 | 15.02	**257** | 110 | 386	**9000** | 3832 | 13495
	Seq.	**0.9990** | 0.9851 | 0.9999	**158** | 43 | 645	**3511** | 1400 | 13269	**77189** | 30788 | 291716
9	Conc.	**0.9999** | 0.9999 | 0.9999	**2.81** | 1.80 | 4.62	**87** | 57 | 142	**3056** | 2007 | 4982
	Seq.	**0.9996** | 0.9995 | 0.9998	**41** | 22 | 104	**1057** | 693 | 2033	**23223** | 15223 | 44653
10	Conc.	**0.9978** | 0.9834 | 0.9999	**638** | 91 | 1566	**798** | 402 | 944	**34315** | 17276 | 40590
	Seq.	**0.9995** | 0.9974 | 0.9999	**3213** | 529 | 8282	**16045** | 6914 | 33844	**417076** | 179721 | 879708
11	Conc.	**0.9999** | 0.9998 | 0.9999	**449** | 37 | 1165	**613** | 335 | 906	**26342** | 14386 | 38939
	Seq.	**0.9999** | 0.9998 | 0.9999	**1557** | 863 | 2935	**8408** | 4895 | 14865	**218564** | 127232 | 386383
12	Conc.	**0.9984** | 0.9948 | 0.9999	**197** | 16 | 416	**196** | 192 | 202	**8426** | 8273 | 8678
	Seq.	**0.9974** | 0.9911 | 0.9990	**883** | 255 | 2060	**4320** | 1994 | 9522	**112192** | 51780 | 247266
13	Conc.	**0.9999** | 0.9999 | 0.9999	**2.65** | 1.74 | 4.43	**64** | 47 | 107	**2247** | 1636 | 3729
	Seq.	**0.9999** | 0.9999 | 0.9999	**16.31** | 9.63 | 31.65	**520** | 310 | 1040	**11427** | 6804 | 22872
14	Conc.	**0.9999** | 0.9999 | 0.9999	**1.48** | 1.09 | 1.94	**40** | 33 | 48	**1405** | 1152 | 1676
	Seq.	**0.9999** | 0.9999 | 0.9999	**5.25** | 3.90 | 6.50	**166** | 126 | 207	**3654** | 2772 | 4550
15	Conc.	**0.9999** | 0.9999 | 1.0000	**0.00** | 0.00 | 0.01	**0.55** | 0.40 | 0.76	**5.70** | 4.02 | 8.00
	Seq.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.01 | 0.02	**0.75** | 0.52 | 1.60	**7.44** | 5.17 | 15.92
16	Conc.	**0.9999** | 0.9999 | 1.0000	**0.00** | 0.00 | 0.01	**0.76** | 0.58 | 1.22	**7.73** | 5.71 | 12.77
	Seq.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.01 | 0.02	**0.58** | 0.45 | 1.15	**5.77** | 4.47 | 11.48
17	Conc.	**0.9999** | 0.9999 | 0.9999	**162** | 28 | 320	**616** | 536 | 750	**6768** | 5887 | 8242
	Seq.	**0.9999** | 0.9999 | 0.9999	**1346** | 763 | 2603	**9238** | 7502 | 13230	**92361** | 75005 | 132274
18	Conc.	**0.9999** | 0.9999 | 0.9999	**118** | 24 | 309	**522** | 400 | 652	**5736** | 4391 | 7163
	Seq.	**0.9999** | 0.9999 | 0.9999	**897** | 428 | 1152	**6788** | 5799 | 8207	**67863** | 57978 | 82054
19	Conc.	**0.9999** | 0.9999 | 0.9999	**41.77** | 5.48 | 120.52	**65** | 58 | 71	**1481** | 1332 | 1636
	Seq.	**0.9999** | 0.9999 | 0.9999	**59** | 32 | 89	**460** | 398 | 547	**7354** | 6362 | 8747
20	Conc.	**0.9998** | 0.9991 | 0.9999	**1.91** | 0.51 | 6.98	**139** | 47 | 202	**1529** | 517 | 2225
	Seq.	**0.9980** | 0.9879 | 0.9996	**64** | 29 | 103	**3508** | 2098 | 6647	**34972** | 20910 | 66265
21	Conc.	**0.9999** | 0.9999 | 0.9999	**1.50** | 1.18 | 2.07	**70** | 53 | 96	**1328** | 1005 | 1818
	Seq.	**0.9998** | 0.9998 | 0.9999	**118** | 84 | 164	**4269** | 2860 | 5791	**59697** | 39992 | 80980
22	Conc.	**0.9999** | 0.9999 | 0.9999	**0.58** | 0.32 | 0.90	**51** | 29 | 74	**563** | 317 | 820
	Seq.	**0.9995** | 0.9982 | 0.9998	**81** | 35 | 137	**4128** | 1981 | 6114	**41194** | 19771 | 61017
23	Conc.	**0.9999** | 0.9999 | 0.9999	**0.06** | 0.05 | 0.07	**7.58** | 6.20 | 9.85	**83** | 68 | 108
	Seq.	**0.9999** | 0.9999 | 0.9999	**1.20** | 0.59 | 2.24	**93** | 46 | 171	**925** | 458 | 1702

TABLE VI. Test results obtained from 20 constrained optimizations (fmincon in MATLAB) for each problem of Table III using the sequential- or the concurrent-update algorithm. Small initial pulse amplitudes were used: mean(u_{ini}) = 0, std(u_{ini}) = 1.

Problem	Algorithm	Final fidelity (**mean** | min | max)	Wall time (min) (**mean** | min | max)	No. eigendecs/1000 (**mean** | min | max)	No. matrix mults/1000 (**mean** | min | max)
1	Conc.	**0.9999** | 0.9999 | 1.0000	**0.11** | 0.02 | 0.26	**1.43** | 1.23 | 1.68	**27** | 23 | 31
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.10** | 0.04 | 0.22	**6.17** | 2.70 | 10.92	**86** | 38 | 152
2	Conc.	**0.9999** | 0.9999 | 0.9999	**0.05** | 0.03 | 0.10	**2.10** | 1.64 | 2.52	**39** | 31 | 47
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.08** | 0.05 | 0.09	**5.74** | 4.04 | 6.96	**80** | 56 | 97
3	Conc.	**0.9999** | 0.9999 | 1.0000	**0.09** | 0.08 | 0.13	**4.49** | 3.71 | 5.50	**83** | 68 | 102
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.08** | 0.03 | 0.15	**5.13** | 2.05 | 8.06	**72** | 29 | 113
4	Conc.	**0.9999** | 0.9999 | 1.0000	**0.03** | 0.02 | 0.04	**1.60** | 1.34 | 1.98	**29** | 24 | 37
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.03** | 0.01 | 0.04	**1.89** | 1.02 | 2.75	**26** | 14 | 38
5	Conc.	**0.9877** | 0.9322 | 0.9990	**44** | 24 | 67	**364** | 361 | 368	**9828** | 9759 | 9947
	Seq.	**0.9973** | 0.9918 | 0.9986	**34** | 22 | 61	**1976** | 1320 | 3292	**35542** | 23729 | 59209
6	Conc.	**0.9999** | 0.9999 | 0.9999	**1.87** | 0.73 | 3.61	**24** | 18 | 40	**650** | 473 | 1074
	Seq.	**0.9999** | 0.9999 | 0.9999	**5.34** | 1.15 | 19.60	**310** | 68 | 1143	**5574** | 1216 | 20554
7	Conc.	**0.9958** | 0.9808 | 0.9999	**29.82** | 5.74 | 69.99	**244** | 69 | 390	**8552** | 2411 | 13634
	Seq.	**0.9945** | 0.9825 | 0.9999	**123** | 39 | 244	**3978** | 1242 | 7749	**87443** | 27313 | 170360
8	Conc.	**0.9999** | 0.9999 | 0.9999	**5.39** | 2.08 | 20.83	**49** | 29 | 198	**1697** | 995 | 6925
	Seq.	**0.9999** | 0.9999 | 0.9999	**15.19** | 2.66 | 67.58	**500** | 89 | 2223	**11002** | 1953 | 48876
9	Conc.	**0.9999** | 0.9999 | 0.9999	**1.21** | 0.69 | 1.68	**14.94** | 8.19 | 21.06	**521** | 285 | 735
	Seq.	**0.9999** | 0.9999 | 0.9999	**4.90** | 1.91 | 7.99	**161** | 63 | 259	**3536** | 1375 | 5696
10	Conc.	**0.9998** | 0.9985 | 0.9999	**281** | 21 | 1753	**355** | 94 | 904	**15251** | 4052 | 38874
	Seq.	**0.9991** | 0.9864 | 0.9999	**1942** | 122 | 12208	**11549** | 988 | 84232	**300198** | 25679 | 2189468
11	Conc.	**0.9999** | 0.9999 | 0.9999	**153.76** | 7.80 | 1104.29	**144** | 68 | 566	**6182** | 2890 | 24346
	Seq.	**0.9998** | 0.9988 | 0.9999	**1141** | 86 | 9848	**6990** | 427 | 74922	**181706** | 11097 | 1947465
12	Conc.	**0.9999** | 0.9992 | 0.9999	**76.27** | 5.76 | 948.99	**70** | 22 | 194	**3017** | 936 | 8331
	Seq.	**0.9997** | 0.9970 | 0.9999	**1108** | 39 | 5566	**5054** | 245 | 19200	**131246** | 6360 | 498598
13	Conc.	**0.9844** | 0.9102 | 0.9999	**13.91** | 2.49 | 75.62	**120** | 27 | 398	**4189** | 950 | 13926
	Seq.	**0.9759** | 0.9373 | 0.9999	**128.25** | 6.64 | 454.86	**4174** | 215 | 15103	**91773** | 4719 | 332029
14	Conc.	**0.9973** | 0.9867 | 0.9999	**7.47** | 2.06 | 14.20	**49** | 29 | 80	**1720** | 1000 | 2801
	Seq.	**0.9999** | 0.9999 | 0.9999	**6.58** | 1.77 | 16.61	**219** | 59 | 547	**4813** | 1292 | 12033
15	Conc.	**0.9999** | 0.9999 | 1.0000	**0.05** | 0.02 | 0.10	**0.71** | 0.52 | 0.92	**7.49** | 5.35 | 9.77
	Seq.	**0.9999** | 0.9999 | 1.0000	**0.02** | 0.01 | 0.04	**1.76** | 0.72 | 3.28	**17.53** | 7.16 | 32.64
16	Conc.	**0.9999** | 0.9999 | 1.0000	**0.04** | 0.01 | 0.07	**0.96** | 0.77 | 1.34	**9.99** | 7.83 | 14.19
	Seq.	**0.9999** | 0.9999 | 1.0000	**0.01** | 0.01 | 0.02	**0.67** | 0.51 | 1.54	**6.64** | 5.10 | 15.31
17	Conc.	**0.9999** | 0.9999 | 0.9999	**531** | 58 | 1443	**1224** | 1032 | 1551	**13454** | 11344 | 17054
	Seq.	**0.9999** | 0.9999 | 0.9999	**2284** | 1054 | 3898	**16774** | 11490 | 27294	**167710** | 114877 | 272885
18	Conc.	**0.9999** | 0.9999 | 0.9999	**157** | 26 | 754	**574** | 530 | 655	**6300** | 5821 | 7196
	Seq.	**0.9999** | 0.9999 | 0.9999	**386** | 175 | 690	**2953** | 2434 | 3985	**29524** | 24335 | 39842
19	Conc.	**0.9999** | 0.9999 | 0.9999	**105** | 16 | 335	**166** | 141 | 186	**3807** | 3244 | 4273
	Seq.	**0.9999** | 0.9999 | 0.9999	**143** | 64 | 328	**1130** | 996 | 1465	**18064** | 15925 | 23433
20	Conc.	**0.9999** | 0.9999 | 0.9999	**0.53** | 0.12 | 1.14	**5.15** | 4.16 | 6.91	**56** | 45 | 76
	Seq.	**0.9999** | 0.9999 | 0.9999	**0.45** | 0.23 | 0.77	**30** | 16 | 51	**302** | 160 | 511
21	Conc.	**0.9999** | 0.9999 | 0.9999	**0.49** | 0.36 | 0.89	**9.53** | 9.09 | 10.37	**179** | 171 | 195
	Seq.	**0.9999** | 0.9999 | 0.9999	**1.39** | 0.79 | 3.80	**39** | 29 | 57	**551** | 410 | 804
22	Conc.	**0.9999** | 0.9999 | 0.9999	**5.34** | 2.39 | 8.03	**131** | 93 | 193	**1444** | 1026 | 2128
	Seq.	**0.9991** | 0.9983 | 0.9995	**108** | 59 | 386	**4702** | 3317 | 6780	**46924** | 33106 | 67669
23	Conc.	**0.9999** | 0.9999 | 0.9999	**2.26** | 0.42 | 9.57	**38** | 13 | 83	**420** | 148 | 913
	Seq.	**0.9951** | 0.9797 | 0.9995	**37.38** | 9.40 | 97.52	**2991** | 744 | 7163	**29786** | 7408 | 71343

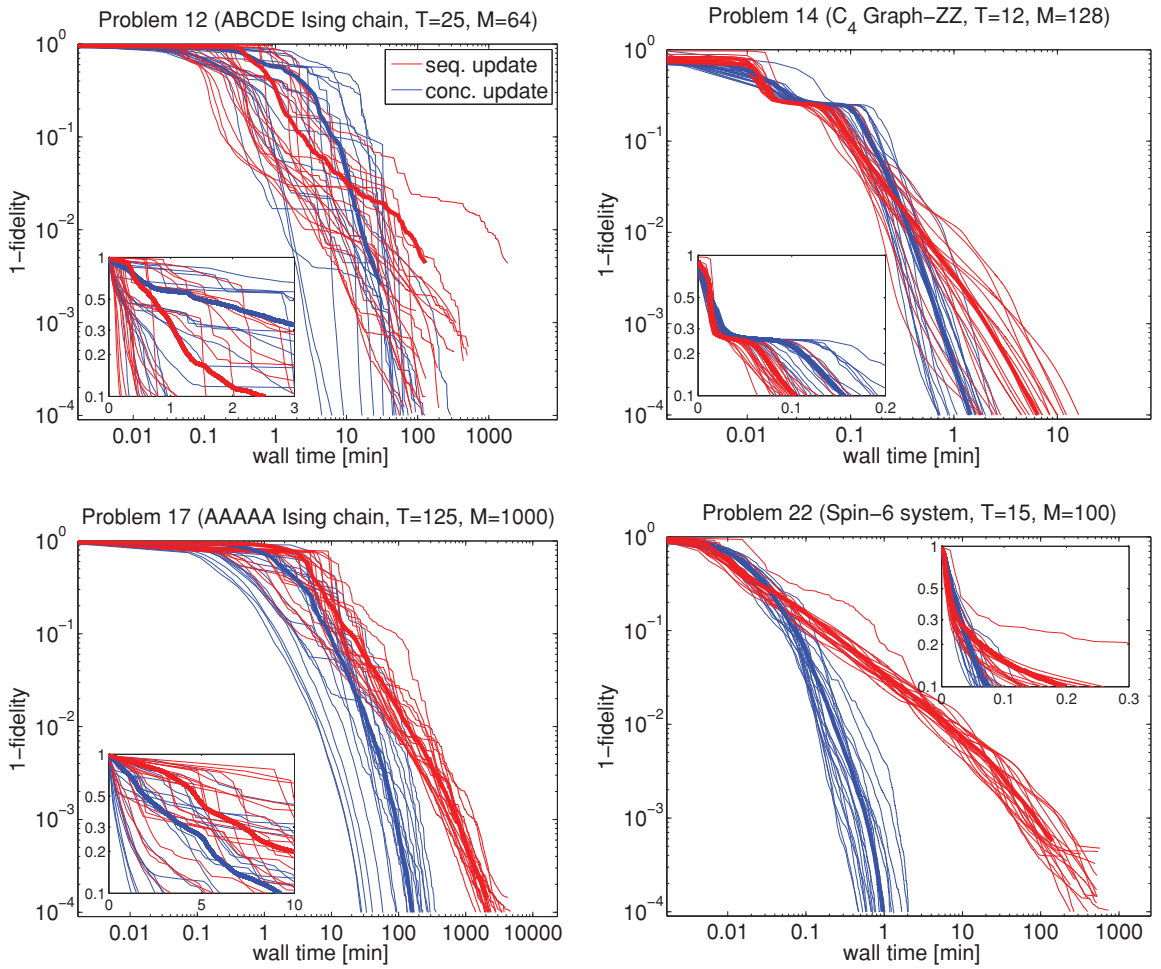

FIG. 3. (Color) Optimization results for problems 12, 14, 17, and 22 shown in doubly logarithmic plots; each optimization is run with 20 random initial conditions; the trace of mean values is given in boldface. The blue (concurrent) and red (sequential) lines depict the deviation of the quality from the maximum of 1 as a function of the wall time. Each line represents one optimization. The insets show the initial behaviors and crossing points in a log-linear scale. For the sequential-update algorithm in problem 22 (last panel), the thresholds for the change in the control and function values have to be lowered [to 10^{-10} instead of the standard 10^{-8} (see test conditions)] for reaching qualities comparable to the ones the concurrent scheme arrives at under standard conditions. (Note the altered thresholds apply as well to the data listed in Table IV).

average of the previous M iterations (sequential update) was introduced. The threshold value in this case was set to 10^{-8}. For the concurrent-update algorithm, the optimization stopped when the smallest change in the control vector was below 10^{-8}. We measured the wall times of our optimizations to give a measure for the actual running time from start to completion (including, e.g., memory loads and communication processes) instead of only measuring the time spent on the CPU. The optimizations were carried out under MATLAB R2009b (64-bit, single-thread mode) on an AMD Opteron dual-core CPU at 2.6 GHz with 8 GB of random access memory (RAM). (The DYNAMO hybrids ran later with an extension to 32 GB of RAM under features of MATLAB R2010b). The wall time was measured using the tic and toc commands in MATLAB. Pure Krotov vs GRAPE comparisons (Tables IV–VI) were carried out on separate optimized MATLAB implementations thus avoiding any overhead (e.g., loops and checks) required for more flexibility in DYNAMO, where the hybrids (Figs. 8 and 9) were run.

C. Test results and discussion

From the full set of data presented in Table IV, Fig. 3 selects a number of representatives for further illustration. Note the following results First, in most of the problems, sequential- and concurrent-update algorithms reach similar final fidelities, the target set to $1–10^{-4}$ being in the order of a conservative estimate for the error-correction threshold [90]. Out of the total of 23 test problems, this target is met within the limits of iterations specified above except in problems 5, 7, 10, 12, and 13. Only in problem 23 does the sequential-update algorithm yield average residual errors (1-fidelity) up to two orders of magnitude higher than in the concurrent optimization. Remarkably enough, the average running times differ substantially in most of the test problems, with the concurrent-update algorithm being faster. Only in problems 3, 4, 15, and 16 are the final wall times similar. Note that in all but the very easy problems 3, 4, and 16, the sequential algorithm needs a larger total number of matrix multiplications and eigendecompositions. In particular, due to the slower

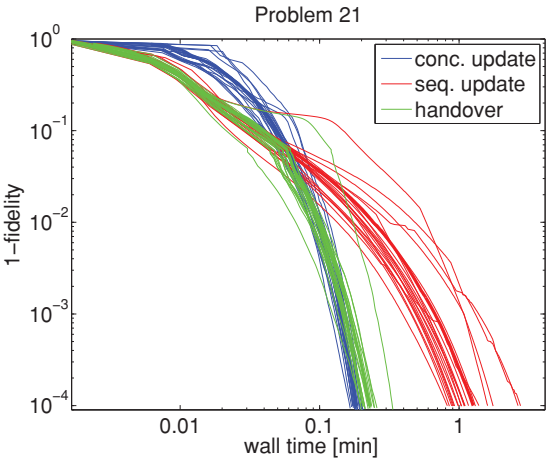

FIG. 4. (Color) Example of a handover (green) from a sequential- (red) to a concurrent-update (blue) scheme. The sequential algorithm is run up to a handover quality of 0.93, where the resulting pulse sequence is then used as input to the concurrent algorithm for optimization up to the target quality. This type of handover is supported by the modular structure of DYNAMO.

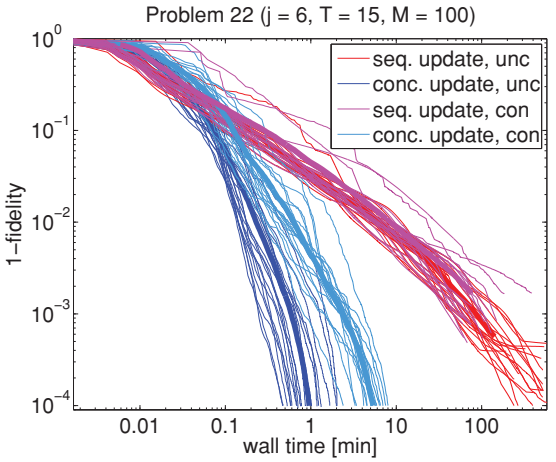

FIG. 5. (Color) Comparison of unconstrained and (loosely) constrained optimizations. The concurrent-update algorithm uses the standard MATLAB-toolbox functions fminunc and fmincon, with the latter being slower than the former, because it may switch between different internal routines. The sequential-update algorithm uses a very basic cutoff method for respecting the constraints, which shows little effect on the performance.

convergence near the critical points, the sequential-update scheme requires more iterations in order to reach the target fidelity of $1–10^{-4}$ thus resulting in a greater number of matrix multiplications and eigendecompositions.

In many problems (3, 5, 6, 8, 9, 11, 12, 14, 18, 19, 21, and 22), we observe a crossing point in time course of the fidelity of the two algorithms. The sequential-update algorithm is overtaken by the concurrent-update scheme between a quality of 0.9 and 0.99 (see, e.g., problem 21 in Fig. 3). Therefore, exploiting the modular framework of the programming package to dynamically change from a sequential- to a concurrent-update scheme at a medium fidelity can be advantageous. This is exemplified in the (constrained) optimization shown in Fig. 4. Here the sequential method is typically faster at the beginning of the optimization, whereas the concurrent method overtakes at higher fidelities near the end of the optimization. Moreover with regard to dispersion of the final wall times required to achieve the target fidelity, in problems 5, 7, 10, 11, 12, 13, and 23 the sequential-update algorithm shows a larger standard deviation, thus indicating higher sensitivity to the initial controls.

Also on a more general scale, we emphasize that the run times may strongly depend on the *choice of initial conditions*. Results for larger initial pulse amplitudes with a higher standard deviation can be found in Table V. Increasing mean value and standard deviation of the initial random control-amplitude vectors typically translates into longer run times. This effect is more pronounced for sequential- than for concurrent-update algorithms. Consequently, the performance differences between the two algorithms may increase and crossing or handover points may change as well.

Finally, as shown in Fig. 5, the performance of the concurrent-update scheme also differs between constrained and unconstrained optimization, i.e., between the standard MATLAB subroutines fmincon and fminunc (see MATLAB documentation). In contrast, the sequential-update algorithm uses the same set of routines for both types of optimization, where a basic cutoff method for respecting the constraints has almost no effect, as also illustrated by Fig. 5.

D. Preliminaries on trust-region Newton methods for sequential-update algorithms

Figure 6 shows that the sequential-update method with first-order gradient information used in this work already achieves a quality gain per iteration that comes closest to the one obtained by a direct implementation of a trust-region Newton method. However, as is analyzed in detail on a larger scale in [72], the small initial advantage *per iteration* of the latter against the former is outweighed CPU-timewise by more costly calculations, which is why we have used the first-order gradients for comparison.

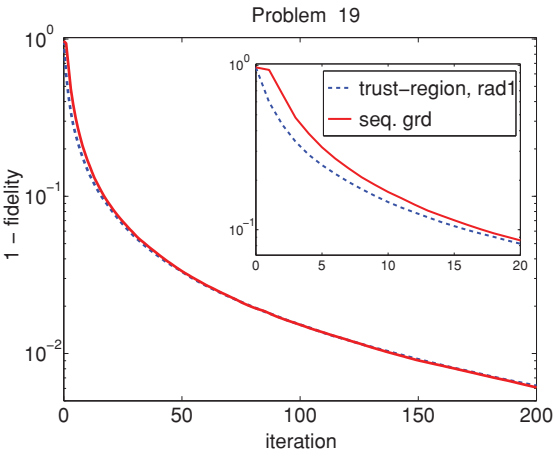

FIG. 6. (Color) Comparison of sequential-update methods with first-order gradient information (red solid track) and with a direct implementation of a trust-region Newton method (blue dotted track) showing that per iteration the gains are similar, in particular in the long run. The curves represent averages over 100 trajectories with random initial conditions.

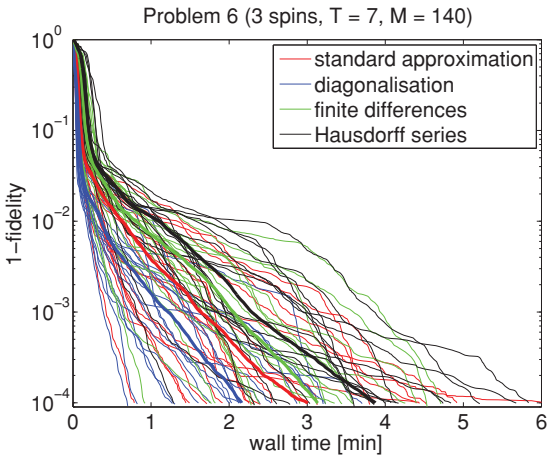

FIG. 7. (Color) Comparison of four different methods for computing gradients in 20 unitary optimizations of problem 6. Apart from the standard approximation, all methods compute exact gradients. By making use of the spectral decomposition, diagonalization the total Hamiltonian to give exact parameter derivatives [73–75] is the fastest among these methods, because by the eigendecomposition the matrix exponential can be settled as well (i.e., in the same go). In the case of optimizing controls for (pure) state-to-state transfer, the standard approximation can be shown to be competitive.

E. Comparing gradient methods

We compare the performance of four different methods to compute gradients for the concurrent algorithm: in addition to the standard approximation and the exact procedure described in Sec. II E 2, we follow Ref. [91] and study a Taylor series to compute the exponential and a Hausdorff series to compute the gradient, while the fourth method is standard finite differences. Note that Hausdorff series and finite differences can be taken to a numerical precision exceeding that of the standard approximation.

An example of the performance results found for these four methods is given in Fig. 7, where we optimize controls for a QFT on the four-spin system of problem 6. Unitary optimizations on other systems yield similar results, with the diagonalization being the fastest methods in all cases. For state-to-state transfer (pure states), however, the standard approximation performs well enough as to be competitive with exact gradients by diagonalization. Note that for unitary gate synthesis of generic gates, one cannot use sparse-matrix techniques, for which the Hausdorff series is expected to work much faster as demonstrated in the software package SPINACH [92].

F. Hybrid schemes

Using DYNAMO, we have just begun to explore the multitude of possible hybrid schemes. Here we present first-order (Fig. 8) as well as second-order (Fig. 9) schemes, where the hybrids are taken with respect to sequential vs concurrent subspace selection. More precisely, this amounts to an outer-loop subspace-selection scheme which picks consecutive blocks of n time slices to be updated in the inner loop using either a first-order or a second-order method update scheme, each allowing us to take at most s_{limit} steps within each block. The results of these explorations, as applied to the two-spin case of problems 2 and 21 (see Table III and Sec. IV A 1 with the same initial conditions as in Table IV), are depicted in Fig. 8 for first-order gradient update and in Fig. 9 for second-order BFGS update. They provide illuminating guidelines for further investigation, as the Krotov method taking a single time slice ($n = 1$) sequentially after the other for a single update step ($s_{\mathrm{limit}} = 1$) may not always be the best performing use of the first-order update scheme.

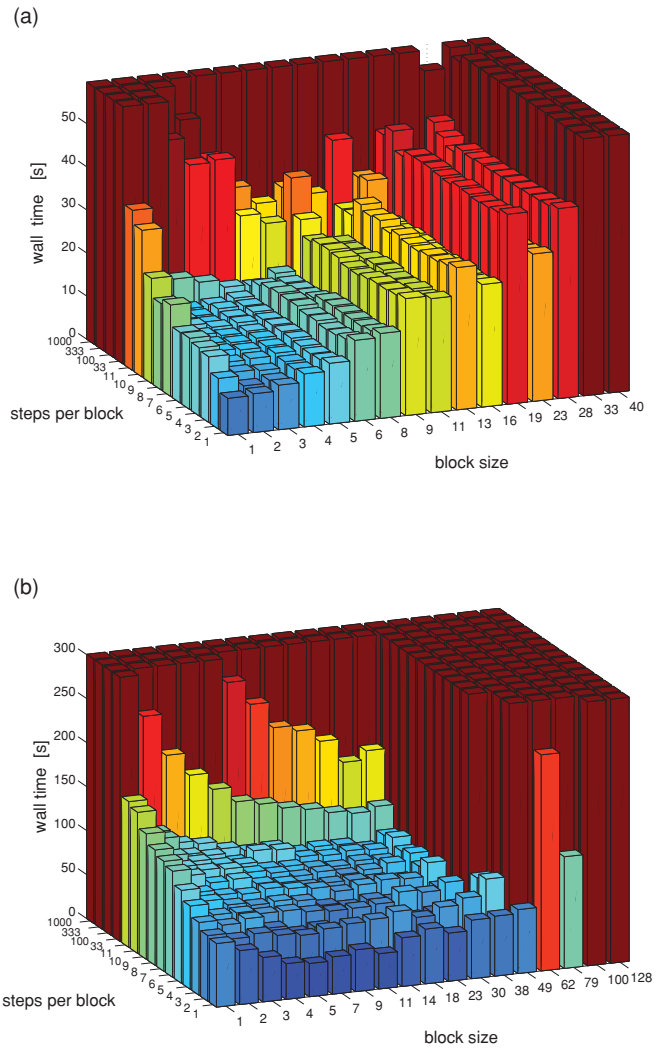

FIG. 8. (Color) Performance of generalizing the first-order-gradient sequential scheme to updating blocks of joint time slicesand allowing for multiple iteration steps within each block ($s_{\mathrm{limit}} > 1$), as applied to test problem (a) 2 and (b) 21 (see Table III and Sec. IV A 1). Original Krotov modifies one time slicein a single iteration ($s = 1$) before moving to the subsequent time sliceto be updated: this special case is shown in the lower corner of the plot, while the upper right is the first-order variant of GRAPE (in a suboptimal setting, since the step-size handling is taken over from the one optimized for Krotov). Wall times represent the average over 42 runs with random initial control vectors [again with mean $(u_{\mathrm{ini}}) = 0$ and std $(u_{\mathrm{ini}}) = 1$ in units of $1/J$]; times are cut off at 60 (300) s. Note that in problem 2 the hybrid first-order versions are not faster than the original Krotov, while in problem 21 it pays to concurrently update four or five time slots by a single step before moving on to the next set of time slots. Note that in other cases also the first-order concurrent update can be fastest, see Fig. 10.

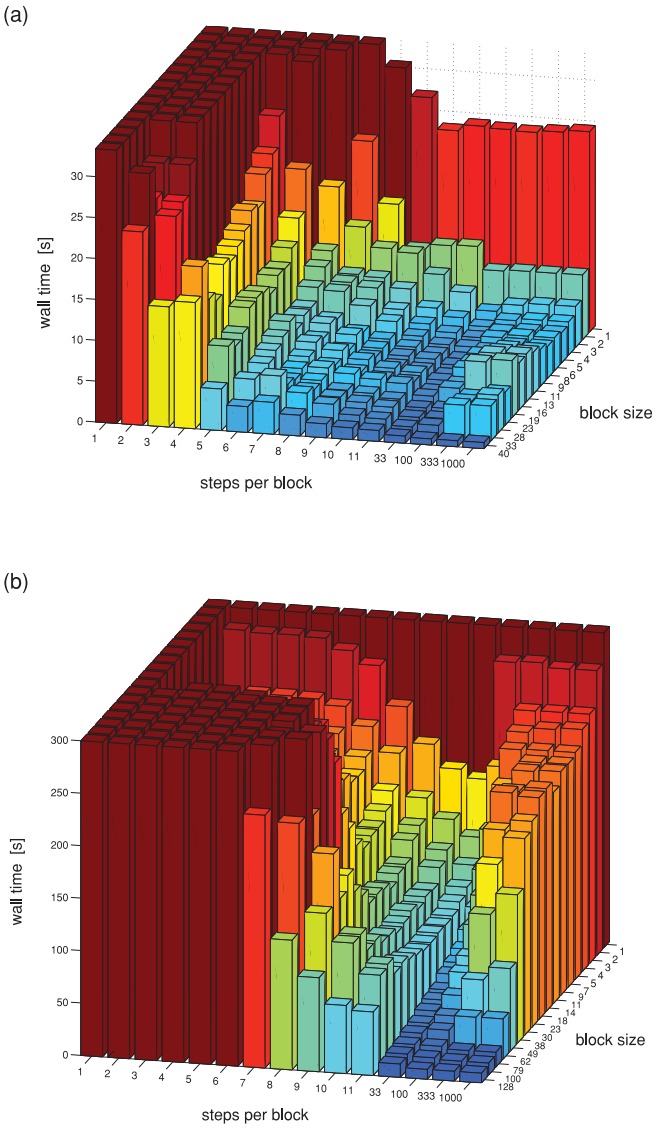

FIG. 9. (Color) Performance of generalizing the second-order (BFGS) concurrent scheme to updating blocks of joint time slicesand allowing fewer iteration steps within each block (s_{limit}), as applied to problems (a) 2 and (b) 21 (see Table III and Sec. IV A 1). Original GRAPE modifies all time slices in each iteration: this special case is shown in the lower right corner of the plot, while the upper left corner is the crude second-order variant of Krotov (for the sake of comparison here in the unrecommended setting of BFGS). It is part of the obvious no-go area of single iterations ($s = 1$) on a single time slice ($n = 1$), or just few, shown for completeness. Wall times represent the average over 42 runs with random initial control vectors again with mean $(u_{\text{ini}}) = 0$ and std $(u_{\text{ini}}) = 1$ in units of $1/J$; times are cut off at 35 (300) s.

On the other hand, in second-order BFGS methods the GRAPE scheme with totally concurrent update cannot be accelerated by allowing for smaller blocks of concurrent update in the sense of a "compromise toward Krotov"; rather it is an optimum within a broader array of similarly performing schemes. This is remarkable, while the incompatibility of BFGS with sequential-update rules is to be expected on the grounds of the discussion above.

Further explorative numerical results on first-order hybrids between sequential and concurrent update as compared to the second-order concurrent update can be found in Fig. 10. They show that in simpler problems the first-order sequential update (as in Krotov) is faster than the (highly suboptimal) first-order variants of hybrid or concurrent update, while in more complicated problems already the first-order variant of concurrent update is slightly faster. In any case, all first-order methods are finally outperformed by second-order concurrent update (as in GRAPE-BFGS).

Clearly, these explorative results are by no means the last word on the subject. Rather they are meant to invite further studies over a wider selection of problems. But even at this early stage we can state that there are hints that hybrid methods hold a yet untapped potential, and follow-up work is warranted.

V. CONCLUSIONS AND OUTLOOK

We have provided a unifying modular programming framework, DYNAMO, for numerically addressing bilinear quantum-control systems. It allows benchmarking, comparing, and optimizing numerical algorithms that constructively approximate optimal quantum controls. Drawing from the modular structure, we have compared the performance of gradient-based algorithms with sequential update of the time slices in the control vector (Krotov-type) versus algorithms with concurrent update (GRAPE-type) with focus on synthesizing unitary quantum gates with high fidelity. For computing gradients, exact methods using the eigendecomposition have on average proven superior to gradient approximations by finite differences, series expansions, or time averages.

When it comes to implementing second-order schemes, the different construction of sequential update and concurrent update translates into different performance: in contrast to the former, recursive concurrent updates match particularly well with quasi-Newton methods and their iterative approximation of the (inverse) Hessian as in standard BFGS implementations. Currently, however, there seems to be no standard Newton-type second-order routine that would match with sequential update in a computationally fast and efficient way such as to significantly outperform our implementation of first-order methods. Finding such a routine is rather an open research problem. At this stage, we have employed efficient implementations, i.e., first-order gradient ascent for sequential update and a second-order concurrent update (GRAPE-BFGS). As expected from second-order vs first-order methods, at higher fidelities (here typically 90–99%), GRAPE-BFGS overtakes Krotov. For reaching a fidelity in unitary gate synthesis of $1–10^{-4}$, GRAPE-BFGS is faster, in a number of instances even by more than one order of magnitude on average. Yet at lower qualities the computational speeds are not that different and sequential update typically has a (small) advantage.

By its flexibility, the DYNAMO framework answers a range of needs, reaching from quantum-information processing to coherent spectroscopy. For the primary focus of this study, namely, gate synthesis with high fidelities beyond the error-correction threshold of some 10^{-4} [90], fidelity requirements significantly differ from pulse engineering for state transfer, where often for the sake of robustness over a broad range of experimental parameters, some fidelity

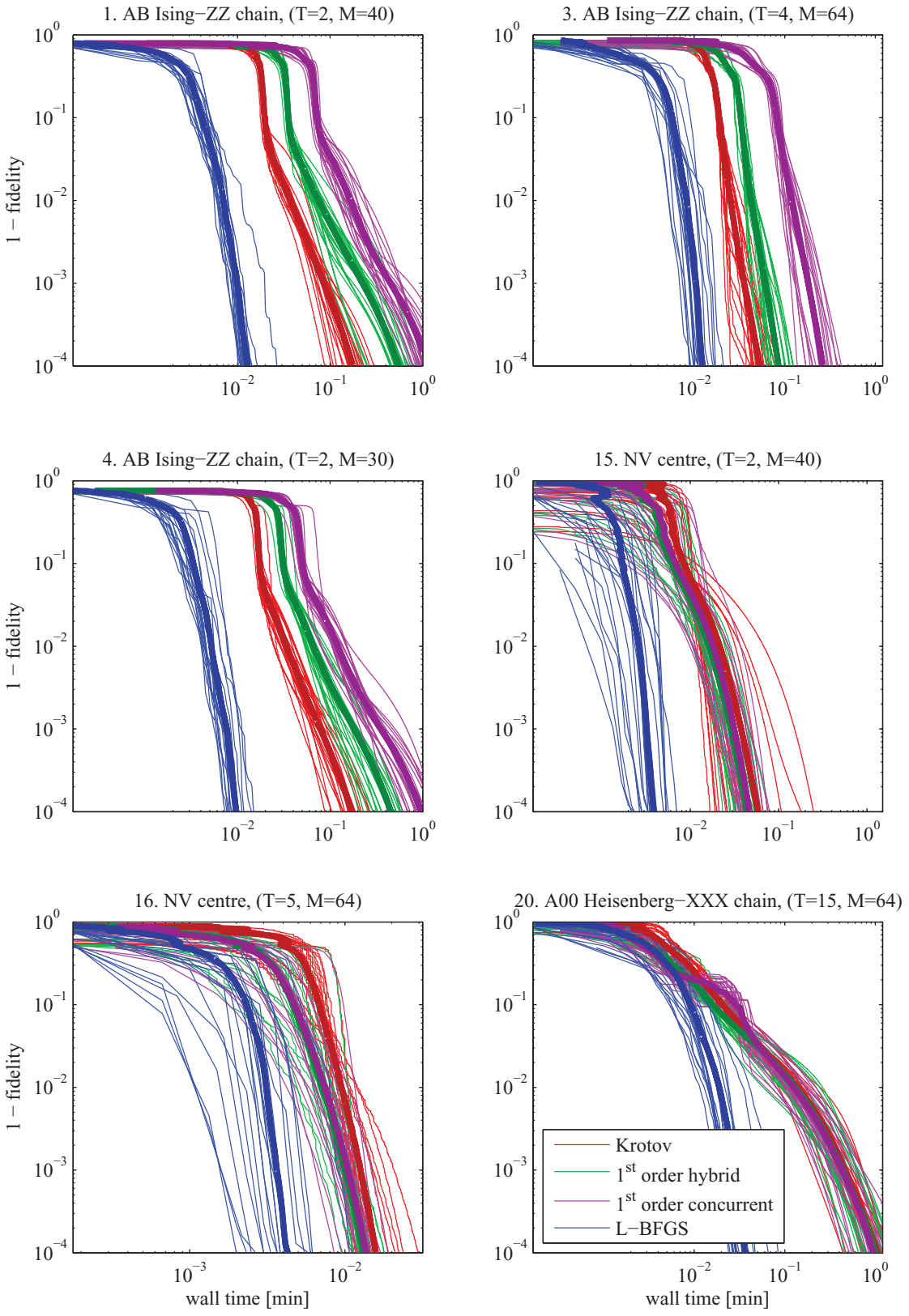

FIG. 10. (Color) Performance of a broader variety of *first-order* schemes compared to the L-BFGS concurrent update. The red traces show the plain Krotov sequential first-order update, while the first-order concurrent update is given in magenta and a first-order hybrid (with block size of 5) is given in green. For comparison, the *second-order* L-BFGS concurrent update is shown in blue. The test examples are again taken as before (see Tab. III and Sec. IV A 1) with mean $(u_{\mathrm{ini}}) = 0$ and std $(u_{\mathrm{ini}}) = 1$ in units of $1/J$. Note that in the (simpler) problems 1, 3, and 4, the original Krotov performs fastest among all the first-order methods, while in problems 15, 16, and 20 the variance within each method comes much closer to the variance among the methods so that in problem 16 the first-order concurrent scheme outperforms the sequential one.

(say 5%) may readily be sacrificed. Thus for optimizing robustness, sequential-update schemes are potentially advantageous, while for gate synthesis sequential methods can be a good start, but for reaching high fidelities, we recommend changing to concurrent-update schemes. More precisely, since DYNAMO allows the efficient handover from one scheme to the other, this is our state-of-the-art recommendation. Ongoing and future comparisons are expected to profit from this framework, e.g., when trying update modules with nonlinear conjugate gradients [93,94].

1. Research Perspectives

We have presented a first step toward establishing a useful tool set for quantum optimal control. It is meant to provide the platform for future improvements and follow-up studies, e.g., along the following lines:

Further types of applications. We have focused on the synthesis of high-fidelity unitary quantum gates in closed systems. Yet, follow-up comparisons should extend to open systems or to spectroscopic state transfer, where it is to be anticipated that different demand of fidelity may lead to different algorithmic recommendations.

Initial conditions. Currently there is no systematic way how to choose good initial control vectors in a problem-adapted way. Scaling of initial conditions has been shown to translate into computational speeds differing significantly (i.e., up to an order of magnitude). Yet, good guidelines for selecting initial controls are still sought for.

Second-order methods for sequential update. As has been mentioned, we have indications that sequential-update methods are most efficient when matched to first-order gradient procedures. This issue is the subject of follow-up work.

Hybrid algorithms. We have focused on the two extremes of the update scheme spectrum: the sequential and the fully concurrent. Hybrid schemes which intelligently select the subset of time slicesto update at each iteration and dynamically decide on the number of steps and appropriate gradient-based stepping methodology for the inner loop may even achieve better results than the established two extremes. The success, however, depends on developing alternatives to BFGS matching with sequential-update schemes.

Control parametrization methods. We have looked exclusively at piecewise-constant discretization of the control function in the time domain. Although frequency-domain methods also exist (e.g., [34]), there is both ample space to develop further methods and the need for comparative benchmarking.

Algorithms for superexpensive goal functions. For many-body quantum systems, ascertaining the time-evolved state of the system requires extremely costly computational resources. The algorithms described in this manuscript all require some method of ascertaining the gradient, by finite differences if no other approach is available. Such requirements, however, are badly adapted to super-expensive goal functions. Further research to discover new search algorithms excelling in such cases is required.

ACKNOWLEDGMENTS

We wish to acknowledge useful discussions at the Kavli Institute; in particular, we are indebted to exchange within the informal optimal-control comparison group hosted and supported by Tommaso Calarco through the EU project ACUTE. We thank Seth Merkel and Frank Wilhelm for suggesting that we test higher spin-j systems and Ilya Kuprov for helpful discussions. This work was supported by the Bavarian Ph.D. Programme of Excellence QCCC, by the EU projects QAP, Q-ESSENCE, exchange with COQUIT, by Deutsche Forschungsgemeinschaft, DFG, in SFB 631. S.S. gratefully acknowledges the EPSRC ARF grant EP/DO7192X/1. P.d.F. is supported by EPSRC and Hitachi (CASE/CNA/07/47). S.M. wishes to thank the EU project CORNER and the Humboldt Foundation. The calculations were carried out mostly on the Linux cluster of Leibniz Rechenzentrum of the Bavarian Academy of Sciences.

APPENDIX A: EXACT GRADIENTS [Eq. (24)]

For deriving the gradient expression in Eq. (24), we follow [73,74]. Note that by

$$\frac{\partial X}{\partial u_j} = \frac{\partial}{\partial u}\exp\left\{-i\Delta t\left[H_d + (u_j+u)H_j + \sum_{v\neq j} u_v H_v\right]\right\}\Bigg|_{u=0}$$
$$= \frac{\partial}{\partial u}\exp\{-i\Delta t(H_u + uH_j)\}\Big|_{u=0} \tag{A1}$$

one may invoke the spectral theorem in a standard way and calculate matrix functions via the eigendecomposition. For an arbitrary pair of Hermitian (noncommuting) matrices A, B and $x \in \mathbb{R}$, take $\{|\lambda_v\rangle\}$ as the orthonormal eigenvectors to the eigenvalues $\{\lambda_v\}$ of A to obtain the following straightforward yet lengthy series of identities:

$$\begin{aligned}
D &= \langle\lambda_l|\frac{\partial}{\partial x}e^{A+xB}|\lambda_m\rangle\Big|_{x=0} \\
&= \langle\lambda_l|\frac{\partial}{\partial x}\sum_{n=0}^{\infty}\frac{1}{n!}(A+xB)^n|\lambda_m\rangle\Big|_{x=0} \\
&= \langle\lambda_l|\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{q=1}^{n}(A+xB)^{q-1}B(A+xB)^{n-q}|\lambda_m\rangle\Big|_{x=0} \\
&= \langle\lambda_l|\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{q=1}^{n}A^{q-1}BA^{n-q}|\lambda_m\rangle \\
&= \sum_{n=0}^{\infty}\frac{1}{n!}\sum_{q=1}^{n}\lambda_l^{q-1}\langle\lambda_l|B|\lambda_m\rangle\lambda_m^{n-q} \\
&= \langle\lambda_l|B|\lambda_m\rangle\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{q=1}^{n}\lambda_l^{q-1}\lambda_m^{n-q}
\end{aligned} \tag{A2}$$

already explaining the case $\lambda_l = \lambda_m$, while for $\lambda_l \neq \lambda_m$ we have

$$D = \langle\lambda_l|B|\lambda_m\rangle\sum_{n=0}^{\infty}\frac{1}{n!}\lambda_m^{n-1}\sum_{q=1}^{n}\left(\frac{\lambda_l}{\lambda_m}\right)^{q-1}$$
$$= \langle\lambda_l|B|\lambda_m\rangle\sum_{n=0}^{\infty}\frac{1}{n!}\lambda_m^{n-1}\frac{(\lambda_l/\lambda_m)^n - 1}{(\lambda_l/\lambda_m) - 1}$$

$$= \langle\lambda_l|B|\lambda_m\rangle \sum_{n=0}^{\infty} \frac{1}{n!} \frac{\lambda_l^n - \lambda_m^n}{\lambda_l - \lambda_m}$$
$$= \langle\lambda_l|B|\lambda_m\rangle \frac{e^{\lambda_l} - e^{\lambda_m}}{\lambda_l - \lambda_m}. \quad \text{(A3)}$$

An analogous result holds for skew-Hermitian iA, iB. So substituting $A \mapsto -i\Delta t H_u$ and $xB \mapsto -i\Delta t u H_j$ as well as $\lambda_\nu \mapsto -i\Delta t \lambda_\nu$ for $\nu = l,m$ while keeping the eigenvectors $|\lambda_\nu\rangle$ readily recovers Eq. (24). Note that we have explicitly made use of the orthogonality of eigenvectors to different eigenvalues in Hermitian (or more generally *normal*) matrices A, B. Hence in generic open quantum systems with a non-normal Lindbladian, there is no such simple extension for calculating exact gradients.

APPENDIX B: STANDARD SETTINGS IN A NUTSHELL

For convenience, here we give the details for six standard tasks of optimizing state transfer or gate synthesis. The individual steps give the key elements of the core algorithm in Sec. II D and its representation as a flow chart.

Task 1. Approximate unitary target gate up to global phase in closed systems. Define boundary conditions $X_0 := \mathbb{1}$, $X_{M+1} := U_{\text{target}}$; fix final time T and digitization M so that $T = M\Delta t$.

(a) Set initial control amplitudes $u_j^{(0)}(t_k) \in \mathcal{U} \subseteq \mathbb{R}$ for all times t_k with $k \in \mathcal{T}^{(0)} := \{1,2,\ldots,M\}$.
(b) Exponentiate $U_k = e^{-i\Delta t H(t_k)}$ for all $k \in \mathcal{T}^{(r)}$ with $H_k := H_d + \sum_j u_j(t_k) H_j$.
(c) Calculate forward propagation $U_{k:0} := U_k U_{k-1} \cdots U_1 U_0$.
(d) Calculate back propagation $\Lambda^\dagger_{M+1:k+1} := U_{\text{tar}}^\dagger U_M U_{M-1} \cdots U_{k+1}$.
(e) Evaluate fidelity $f = |g|$, where $g := \frac{1}{N}\text{tr}\{\Lambda^\dagger_{M+1:k+1} U_{k:0}\} = \frac{1}{N}\text{tr}\{U_{\text{tar}}^\dagger U_{M:0}\}$ and stop if $f \geqslant 1 - \varepsilon_{\text{threshold}}$ or iteration $r > r_{\text{limit}}$.
(f) Evaluate gradients for all $k \in \mathcal{T}^{(r)}$ $\frac{\partial f(U(t_k))}{u_j} = \frac{1}{N}\,\text{Re}\,\text{tr}\{e^{-i\phi_g}\Lambda^\dagger_{M+1:k+1}(\frac{\partial U_k}{\partial u_j})U_{k-1:0}\}$ with $\frac{\partial U_k}{\partial u_j}$ of Eq. (24) and $e^{-i\phi_g} := g^*/|g|$.
(g) Update amplitudes for all $k \in \mathcal{T}^{(r)}$ by quasi-Newton $u_j^{(r+1)}(t_k) = u_j^{(r)}(t_k) + \alpha_k \mathcal{H}_k^{-1} \frac{\partial f(X(t_k))}{\partial u_j}$ or other methods (as in the text).
(h) While $\frac{\partial f_k}{\partial u_j} > f'_{\text{limit}}$ for some $k \in \mathcal{T}^{(r)}$ reiterate; else reiterate with new set $\mathcal{T}^{(r+1)}$.

Comments. Algorithmic scheme for synthesizing a unitary gate $U(T)$ such as to optimize the gate fidelity $f := |\frac{1}{N}\text{tr}\{U_{\text{target}}^\dagger U(T)\}|$. This setting automatically absorbs global phase factors as immaterial: tracking for minimal times T_* to realize U_{target} up to an undetermined phase automatically gives a $e^{i\phi_*}U_{\text{target}} \in \text{SU}(N)$ of fastest realization.

Task 2. Approximate unitary target gate sensitive to global phase in closed systems. Fix boundary conditions $X_0 := \mathbb{1}$, $X_{M+1} := e^{i\phi}U_{\text{target}}$ by choosing global phase to ensure $\det(e^{i\phi}U_{\text{target}}) = +1$ so that $e^{i\phi}U_{\text{target}} \in \text{SU}(N)$; there are N such choices [12]; fix final time T and digitization M so $T = M\Delta t$.

(a) through (d) as in task 1.
(e) Evaluate fidelity $f = \frac{1}{N}\,\text{Re}\,\text{tr}\{\Lambda^\dagger_{M+1:k+1}U_{k:0}\} = \frac{1}{N}\,\text{Re}\,\text{tr}\{e^{-i\phi}U_{\text{tar}}^\dagger U_{M:0}\}$ and stop if $f \geqslant 1 - \varepsilon_{\text{threshold}}$ or iteration $r > r_{\text{limit}}$.
(f) Evaluate gradients for all $k \in \mathcal{T}^{(r)}$ $\frac{\partial f(U(t_k))}{u_j} = \frac{1}{N}\,\text{Re}\,\text{tr}\{\Lambda^\dagger_{M+1:k+1}(\frac{\partial U_k}{\partial u_j})U_{k-1:0}\}$ with $\frac{\partial U_k}{\partial u_j}$ of Eq. (24).
(g) and (h) as in task 1.

Comments. Algorithmic scheme for synthesizing a unitary gate $U(T)$ in closed quantum systems such as to optimize the gate fidelity $f := \frac{1}{N}\,\text{Re}\,\text{tr}\{e^{-i\phi}U_{\text{target}}^\dagger U(T)\}$. This setting is sensitive to global phases ϕ that have to be specified in advance. *Warning:* whenever drift and control Hamiltonians operate on different time scales, the minimal time T_* required to realize $e^{i\phi}U_{\text{target}} \in \text{SU}(N)$ will typically and significantly depend on ϕ as demonstrated in [12], a problem eliminated by task 1.

Task 3. Optimize state transfer between pure-state vectors. Define boundary conditions $X_0 := |\psi_0\rangle$, $X_{M+1} := |\psi\rangle_{\text{target}}$; fix final time T and digitization M so that $T = M\Delta t$.

(a) Set initial control amplitudes $u_j^{(0)}(t_k) \in \mathcal{U} \subseteq \mathbb{R}$.
(b) Exponentiate $U_k = e^{-i\Delta t H(t_k)}$ for all $k \in \mathcal{T}^{(r)}$ with $H_k := H_d + \sum_j u_j(t_k) H_j$.
(c) Calculate forward propagation $|\psi_0(t_k)\rangle := U_k U_{k-1} \cdots U_1 |\psi_0\rangle$.
(d) Calculate back propagation $\langle\psi_{\text{tar}}(t_k)| := \langle\psi_{\text{tar}}|U_M U_{M-1} \cdots U_{k+1}$.
(e) Evaluate fidelity f, where $f := \text{Re}\langle\psi_{\text{tar}}(t_k)|\psi_0(t_k)\rangle = \text{Re}\langle\psi_{\text{tar}}|(U_M \cdots U_k \cdots U_1|\psi_0\rangle)$ and stop if $f \geqslant 1 - \varepsilon_{\text{threshold}}$ or iteration $r > r_{\text{limit}}$.
(f) Evaluate gradients for all $k \in \mathcal{T}^{(r)}$ $\frac{\partial f(U(t_k))}{u_j} = \text{Re}\langle\psi_{\text{tar}}|(U_M \cdots U_{k+1}(\frac{\partial U_k}{\partial u_j})U_{k-1} \cdots U_1|\psi_0\rangle)$ again with $\frac{\partial U_k}{\partial u_j}$ of Eq. (24);
(g) Update amplitudes for all $k \in \mathcal{T}^{(r)}$ by quasi-Newton $u_j^{(r+1)}(t_k) = u_j^{(r)}(t_k) + \alpha_k \mathcal{H}_k^{-1} \frac{\partial f(X(t_k))}{\partial u_j}$ or other methods (as in the text).
(h) While $\frac{\partial f_k}{\partial u_j} > f'_{\text{limit}}$ for some $k \in \mathcal{T}^{(r)}$ reiterate; else reiterate with new set $\mathcal{T}^{(r+1)}$.

Comments. Algorithmic scheme for optimizing (pure) state-to-state transfer in closed quantum systems. This setting is sensitive to global phases ϕ in $e^{i\phi}|\psi\rangle$ that have to be specified in advance.

Task 4. Optimize state transfer between density operators in closed systems. Define boundary conditions $X_0 := \rho_0$, $X_{M+1} := \rho_{\text{target}}$; fix final time T and digitization M so that $T = M\Delta t$.

(a) and (b) as in tasks 1–3.
(c) Calculate forward propagation $\rho_0(t_k) := U_k U_{k-1} \cdots U_1 \rho_0 U_1^\dagger \cdots U_{k-1}^\dagger U_k^\dagger$.
(d) Calculate back propagation $\rho_{\text{tar}}^\dagger(t_k) := U_{k+1}^\dagger \cdots U_{M-1}^\dagger U_M^\dagger \rho_{\text{tar}}^\dagger U_M U_{M-1} \cdots U_{k+1}$.
(e) Evaluate fidelity f with normalization $c := ||\rho_{\text{tar}}||_2^2$ $f := \frac{1}{c}\,\text{Re}\,\text{tr}\{\rho_{\text{tar}}^\dagger(t_k)\rho_0(t_k)\}$ and stop if $f \geqslant 1 - \varepsilon_{\text{threshold}}$ or iteration $r > r_{\text{limit}}$.

(f) Evaluate gradients for all $k \in \mathcal{T}^{(r)}$
$\frac{\partial f(U(t_k))}{u_j} = \frac{1}{c}\mathrm{Re}\{\mathrm{tr}[\rho_{\mathrm{tar}}^{\dagger}(t_k)(\frac{\partial U_k}{\partial u_j})\rho_0(t_{k-1})U_k^{\dagger}] +$
$\mathrm{tr}\,[\rho_{\mathrm{tar}}^{\dagger}(t_k)U_k\rho_0(t_{k-1})(\frac{\partial U_k^{\dagger}}{\partial u_j})]\}$
with $\frac{\partial U_k}{\partial u_j}$ of Eq. (24).

(g) and (h) as in tasks 1–3.

Comments. Algorithmic scheme for optimizing state-to-state transfer of density operators in closed quantum systems.

Task 5. Approximate unitary target gate by quantum map in open Markovian systems. Define boundary conditions $X_0 := \mathbb{I}$, $X_{M+1} := \widehat{U}_{\mathrm{target}}$; fix final time T and digitization M so that $T = M\Delta t$.

(a) Set initial control amplitudes $u_j^{(0)}(t_k) \in \mathcal{U} \subseteq \mathbb{R}$ for all times t_k with $k \in \mathcal{T}^{(0)} := \{1,2,\ldots,M\}$.

(b) Exponentiate $X_k = e^{-i\Delta t\widehat{H}(t_k)+\Gamma}$ for all $k \in \mathcal{T}^{(r)}$ with $\widehat{H}_k := \widehat{H}_0 + \sum_j u_j(t_k)\widehat{H}_j$.

(c) Calculate forward propagation
$X_{k:0} := X_k X_{k-1}\cdots X_1(X_0 = \mathbb{I})$.

(d) Calculate back propagation
$\Lambda_{M+1:k+1}^{\dagger} := \widehat{U}_{\mathrm{tar}}^{\dagger}X_M X_{M-1}\cdots X_{k+1}$,

(e) Evaluate fidelity
$f = \frac{1}{N^2}\,\mathrm{Re}\,\mathrm{tr}\{\Lambda_{M+1:k+1}^{\dagger}X_{k:0}\} = \frac{1}{N^2}\,\mathrm{Re}\,\mathrm{tr}\{\widehat{U}_{\mathrm{tar}}^{\dagger}X_{M:0}\}$
and stop if $f \geqslant 1 - \varepsilon_{\mathrm{threshold}}$ or iteration $r > r_{\mathrm{limit}}$.
$\frac{\partial f(X(t_k))}{\partial u_j} \approx \frac{-\Delta t}{N^2}\,\mathrm{Re}\,\mathrm{tr}\{\Lambda_{M+1:k+1}^{\dagger}(i\widehat{H}_{u_j} + \frac{\partial\Gamma}{\partial u_j})X_{k:0}\}$.

(f) Update amplitudes for all $k \in \mathcal{T}^{(r)}$ by quasi-Newton
$u_j^{(r+1)}(t_k) = u_j^{(r)}(t_k) + \alpha_k\mathcal{H}_k^{-1}\frac{\partial f(X(t_k))}{\partial u_j}$ or other methods (as in the text).

(g) While $||\frac{\partial f_k}{\partial u_j}|| > f'_{\mathrm{limit}}$ for some $k \in \mathcal{T}^{(r)}$ reiterate;
else reiterate with new set $\mathcal{T}^{(r+1)}$.

Comments. General algorithmic scheme for synthesizing quantum maps $X(T)$ at fixed time T with optimized gate fidelity $f := \frac{1}{N^2}\,\mathrm{Re}\,\mathrm{tr}\{\widehat{U}_{\mathrm{target}}X(T)\}$ in open dissipative quantum systems. $X(t)$ denotes Markovian quantum maps generated in step 1.

Task 6. Optimize state transfer between density operators in open systems. Define boundary conditions by vectors in Liouville space $X_0 := \mathrm{vec}(\rho_0)$ and $X_{\mathrm{tar}}^{\dagger} := \mathrm{vec}^t(\rho_{\mathrm{target}}^{\dagger})$; fix final time T and digitization M so that $T = M\Delta t$.

(a) and (b) as in task 5.

(c) Calculate forward propagation
$X_{k:0} := X_k X_{k-1}\cdots X_1\mathrm{vec}(\rho_0)$.

(d) Calculate back propagation
$\Lambda_{M+1:k+1}^{\dagger} := \mathrm{vec}^t(\rho_{\mathrm{tar}}^{\dagger})X_M X_{M-1}\cdots X_{k+1}$.

(e) Evaluate fidelity
$f = \frac{1}{N}\mathrm{Re}[\mathrm{tr}]\{\Lambda_{M+1:k+1}^{\dagger}X_{k:0}\} = \frac{1}{N}\mathrm{Re}[\mathrm{tr}]\{\widehat{X}_{\mathrm{tar}}^{\dagger}X_{M:0}\}$
and stop if $f \geqslant 1 - \varepsilon_{\mathrm{threshold}}$ or iteration $r > r_{\mathrm{limit}}$.

(f) Approximate gradients for all $k \in \mathcal{T}^{(r)}$
$\frac{\partial f(X(t_k))}{\partial u_j} \approx \frac{-\Delta t}{N}\,\mathrm{Re}\,\mathrm{tr}\{\Lambda_{M+1:k+1}^{\dagger}(i\widehat{H}_{u_j} + \frac{\partial\Gamma}{\partial u_j})X_{k:0}\}$.

(g) and (h) as in task 5.

Comments. Algorithmic scheme for optimizing state transfer between density operators in open Markovian quantum systems, where the representation in Liouville space is required. So task 6 can be seen as the rank-1 version of task 5.

[1] J. P. Dowling and G. Milburn, Philos. Trans. R. Soc. London A **361**, 1655 (2003).

[2] H. M. Wiseman and G. J. Milburn, *Quantum Measurement and Control* (Cambridge University Press, Cambridge, England, 2009).

[3] A. G. Butkovskiy and Y. I. Samoilenko, *Control of Quantum-Mechanical Processes and Systems* (Kluwer, Dordrecht, 1990), see also the translations from Russian originals: Autom. Remote Control (USSR) **40**, 485 (1979), **40**, 629 (1979), as well as Dokl. Akad. Nauk. USSR **250**, 22 (1980).

[4] D. J. Tannor and S. A. Rice, J. Chem. Phys. **83**, 5013 (1985).

[5] A. Peirce, M. Dahleh, and H. Rabitz, Phys. Rev. A **37**, 4950 (1988).

[6] M. Dahleh, A. P. Peirce, and H. Rabitz, Phys. Rev. A **42**, 1065 (1990).

[7] N. Khaneja, B. Luy, and S. J. Glaser, Proc. Natl. Acad. Sci. USA **100**, 13162 (2003).

[8] R. Xu, Y. J. Yan, Y. Ohtsuki, Y. Fujimura, and H. Rabitz, J. Chem. Phys. **120**, 6600 (2004).

[9] H. Jirari and W. Pötz, Phys. Rev. A **74**, 022306 (2006).

[10] V. F. Krotov, *Global Methods in Optimal Control* (Marcel Dekker, New York, 1996).

[11] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, J. Magn. Reson. **172**, 296 (2005).

[12] T. Schulte-Herbrüggen, A. K. Spörl, N. Khaneja, and S. J. Glaser, Phys. Rev. A **72**, 042331 (2005).

[13] M. Greiner, O. Mandel, T. Esslinger, T. W. Hänsch, and I. Bloch, Nature (London) **415**, 39 (2002).

[14] I. Bloch, J. Dalibard, and W. Zwerger, Rev. Mod. Phys. **80**, 885 (2008).

[15] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, Rev. Mod. Phys. **75**, 281 (2003).

[16] J. J. García-Ripoll, P. Zoller, and J. I. Cirac, Phys. Rev. Lett. **91**, 157901 (2003).

[17] J. J. García-Ripoll, P. Zoller, and J. I. Cirac, Phys. Rev. A **71**, 062309 (2005).

[18] U. Dorner, T. Calarco, P. Zoller, A. Browaeys, and P. Grangier, J. Opt. B **7**, S341 (2005).

[19] R. Blatt and D. Wineland, Nature (London) **453**, 1008 (2008).

[20] M. Johanning, A. F. Varón, and C. Wunderlich, J. Phys. B **42**, 154009 (2009).

[21] C. Wunderlich, Nature **463**, 37 (2010).

[22] M. Hofheinz *et al.*, Nature (London) **459**, 546 (2009).

[23] L. DiCarlo *et al.*, Nature (London) **460**, 240 (2009).

[24] K. Singer, U. Poschinger, M. Murphy, P. Ivanov, F. Ziesel, T. Calarco, and F. Schmidt-Kaler, Rev. Mod. Phys. **82**, 2609 (2010).

[25] J. Clarke and F. Wilhelm, Nature (London) **453**, 1031 (2008).

[26] A. K. Spörl, T. Schulte-Herbrüggen, S. J. Glaser, V. Bergholm, M. J. Storcz, J. Ferber, and F. K. Wilhelm, Phys. Rev. A **75**, 012302 (2007).

[27] T. Schulte-Herbrüggen, A. Spörl, N. Khaneja, and S. J. Glaser, J. Phys. B **44**, 154013 (2011).

[28] P. Rebentrost, I. Serban, T. Schulte-Herbrüggen, and F. K. Wilhelm, Phys. Rev. Lett. **102**, 090401 (2009).

[29] P. Zanardi and M. Rasetti, Phys. Rev. Lett. **79**, 3306 (1997).
[30] J. Kempe, D. Bacon, D. A. Lidar, and K. B. Whaley, Phys. Rev. A **63**, 042307 (2001).
[31] R. Nigmatullin and S. G. Schirmer, New J. Phys. **11**, 105032 (2009), [http://dx.doi.org/10.1088/1367-2630/11/10/105032].
[32] K. Khodjasteh and L. Viola, Phys. Rev. Lett. **102**, 080501 (2009).
[33] K. Khodjasteh and L. Viola, Phys. Rev. A **80**, 032314 (2009).
[34] P. Doria, T. Calarco, and S. Montangero, Phys. Rev. Lett. **106**, 190501 (2011).
[35] S. Lloyd, Phys. Rev. A **62**, 022108 (2000).
[36] J. P. Palao and R. Kosloff, Phys. Rev. Lett. **89**, 188301 (2002).
[37] J. P. Palao and R. Kosloff, Phys. Rev. A **68**, 062308 (2003).
[38] Y. Ohtsuki, G. Turinici, and H. Rabitz, J. Chem. Phys. **120**, 5509 (2004).
[39] N. Ganesan and T.-J. Tarn, in Proceedings of the 44th IEEE CDC-ECC, 2005, p. 427.
[40] S. E. Sklarz and D. J. Tannor, Chem. Phys. **322**, 87 (2006).
[41] M. Möttönen, R. de Sousa, J. Zhang, and K. B. Whaley, Phys. Rev. A **73**, 022332 (2006).
[42] M. Grace, C. Brif, H. Rabitz, I. Walmsley, R. Kosut, and D. Lidar, J. Phys. B **40**, S103 (2007).
[43] S. G. Schirmer and P. J. Pemberton-Ross, Phys. Rev. A **80**, 030301 (2009).
[44] S. G. Schirmer, J. Mod. Opt. **56**, 831 (2009).
[45] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, Phys. Rev. Lett. **103**, 110501 (2009).
[46] D. D'Alessandro, *Introduction to Quantum Control and Dynamics* (Chapman & Hall/CRC, Boca Raton, 2008).
[47] V. F. Krotov and I. N. Feldman, Eng. Cybern. **21**, 123 (1983) [Izv. Akad. Nauk. SSSR Tekh. Kibern. **1983**(2), 162 (1983)].
[48] A. I. Konnov and V. F. Krotov, Autom. Remote Control **60**, 1427 (1999) [Avtom. Telemekh. **1999**(10) 77 (1999)].
[49] C. Koch and R. Kosloff, Phys. Rev. A **81**, 063426 (2010).
[50] M. Ndong and C. P. Koch, Phys. Rev. A **82**, 043437 (2010).
[51] N. Timoney, V. Elman, S. J. Glaser, C. Weiss, M. Johanning, W. Neuhauser, and C. Wunderlich, Phys. Rev. A **77**, 052334 (2008).
[52] V. Nebendahl, H. Häffner, and C. F. Roos, Phys. Rev. A **79**, 012312 (2009).
[53] R. Fisher, F. Helmer, S. J. Glaser, F. Marquardt, and T. Schulte-Herbrüggen, Phys. Rev. B **81**, 085328 (2010).
[54] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. (Springer, New York, 2006).
[55] T. Gradl, A. K. Spörl, T. Huckle, S. J. Glaser, and T. Schulte-Herbrüggen, Lect. Notes Comput. Sci. **4128**, 751 (2006)
[56] [http://qlib.info].
[57] H. Sussmann and V. Jurdjevic, J. Diff. Equat. **12**, 95 (1972).
[58] V. Ramakrishna and H. Rabitz, Phys. Rev. A **54**, 1715 (1996).
[59] S. Lloyd, Science **273**, 1073 (1996).
[60] T. Schulte-Herbrüggen, Ph.D. thesis, Diss-ETH 12752, Zürich, 1998.
[61] S. G. Schirmer, H. Fu, and A. I. Solomon, Phys. Rev. A **63**, 063410 (2001).
[62] S. G. Schirmer, I. H. C. Pullen, and A. I. Solomon, J. Phys. A. **35**, 2327 (2002).
[63] F. Albertini and D. D'Alessandro, IEEE Trans. Automat. Control **48**, 1399 (2003).
[64] G. Dirr, U. Helmke, I. Kurniawan, and T. Schulte-Herbrüggen, Rep. Math. Phys. **64**, 93 (2009).
[65] L. D. Berkovitz, Am. Math. Mon. **83**, 225 (1976).
[66] E. B. Bliss, *Lectures on the Calculus of Variations* (University of Chicago, Chicago, 1946).
[67] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis* (Cambridge University Press, Cambridge, England, 1991).
[68] J. Nocedal, Math. Comput. **35**, 773 (1980).
[69] R. H. Byrd, P. Lu, and R. B. Schnabel, Math. Program. **63**, 129 (1994).
[70] B. Savas and L. H. Lim, SIAM J. Sci. Comput. **32**, 3352 (2010).
[71] R. H. Byrd, P. Lu, and J. Nocedal, SIAM J. Sci. Comput. **16**, 1190 (1995).
[72] S. G. Schirmer and P. de Fouquières, New J. Phys. **13**, 073029 (2011), [http://dx.doi.org/10.1088/1367-2630/13/7/073029].
[73] T. Levante, T. Bremi, and R. R. Ernst, J. Magn. Reson. Ser. A **121**, 167 (1996).
[74] K. Aizu, J. Math. Phys. **4**, 762 (1963).
[75] R. M. Wilcox, J. Math. Phys. **8**, 962 (1967).
[76] U. Sander, Ph.D. Thesis, Technical University of Munich, 2010.
[77] C. Moler and C. van Loan, SIAM Rev. **20**, 801 (1978).
[78] C. Moler and C. van Loan, SIAM Rev. **45**, 3 (2003).
[79] K. Waldherr, Diploma thesis, Technical University of Munich, 2007.
[80] T. Schulte-Herbrüggen, A. K. Spörl, K. Waldherr, T. Gradl, S. J. Glaser, and T. Huckle, in *High-Performance Computing in Science and Engineering, Garching 2007* (Springer, Berlin, 2008), p. 517.
[81] R. A. Horn and C. R. Johnson, *Matrix Analysis* (Cambridge University Press, Cambridge, England, 1987).
[82] H. Wunderlich, C. Wunderlich, K. Singer, and F. Schmidt-Kaler, Phys. Rev. A **79**, 052324 (2009).
[83] P. Neumann, N. Mizuochi, F. Rempp, P. Hemmer, H. Watanabe, S. Yamasaki, V. Jacques, T. Gaebel, F. Jelezko, and J. Wrachtrup, Science **320**, 1326 (2008).
[84] S. Bose, Phys. Rev. Lett. **91**, 207901 (2003).
[85] D. Burgarth and S. Bose, New J. Phys. **7**, 135 (2005).
[86] S. Bose, Contemp. Phys. **48**, 13 (2007).
[87] A. Kay, Int. J. Quant. Inf. **8**, 641 (2010).
[88] I. H. Deutsch and P. S. Jessen, Opt. Commun. **283**, 681 (2010).
[89] F. Mezzadri, Notices Amer. Math. Soc. **54**, 592 (2007), [http://www.ams.org/notices/200705/fea-mezzadri-web.pdf].
[90] E. Knill, Nature (London) **434**, 39 (2005).
[91] I. Kuprov and C. T. Rodgers, J. Chem. Phys. **131**, 234108 (2009).
[92] I. Kuprov, J. Magn. Reson. **89**, 241 (2007).
[93] W. W. Hager and H. Zhang, SIAM J. Optim. **16**, 170 (2005).
[94] W. W. Hager and H. Zhang, Pac. J. Optim. **2**, 35 (2006), [http://www.ybook.co.jp/online/pjoe/vol2/v2nlp35.html].