Quantum machine learning with glow for episodic tasks and decision games

Jens Clausen and Hans J. Briegel
Institut für Theoretische Physik, Universität Innsbruck, Technikerstraße 21a, A-6020 Innsbruck, Austria

(Received 5 July 2017; published 5 February 2018)

We consider a general class of models, where a reinforcement learning (RL) agent learns from cyclic interactions with an external environment via classical signals. Perceptual inputs are encoded as quantum states, which are subsequently transformed by a quantum channel representing the agent's memory, while the outcomes of measurements performed at the channel's output determine the agent's actions. The learning takes place via stepwise modifications of the channel properties. They are described by an update rule that is inspired by the projective simulation (PS) model and equipped with a glow mechanism that allows for a backpropagation of policy changes, analogous to the eligibility traces in RL and edge glow in PS. In this way, the model combines features of PS with the ability for generalization, offered by its physical embodiment as a quantum system. We apply the agent to various setups of an invasion game and a grid world, which serve as elementary model tasks allowing a direct comparison with a basic classical PS agent.

DOI: 10.1103/PhysRevA.97.022303

I. INTRODUCTION

If we consider the development of new technologies as a collective learning process, we can distinguish between different interlaced processes. While basic research focuses on exploration characterized by a search for potential alternatives to established methods, the more promising an approach appears, the more likely it becomes subject to subsequent exploitation, where it is optimized and matured with the ultimate hope to supersede what is available. Examples of explorative activity are early efforts to solve problems by artificial intelligence (AI), such as inventing unconventional heuristic techniques. Interest in AI has recently been renewed [1,2], which may be a consequence of new approaches to computation [3–6] as well as improved capacities of classical computing and networking. The present work aims at drawing a connection between the recently suggested scheme of PS [7,8] and quantum control theory [9–12].

We consider a class of schemes, where a quantum agent learns from cyclic interactions with an external environment via classical signals. The learning can be considered as an *internal* quantum navigation process of the agent's "hardware" or "substrate" that forms its memory of past experience. For notational convenience, we describe the memory operation as a unitary $\hat{U}$ involving (information-carrying and other) controllable and uncontrollable degrees of freedom (such as a "bath"), where the latter are not necessarily identical with the environment, on which the agent operates. While conceptually the memory may hence be seen as an open quantum system [13], the numerical examples considered in the present work are restricted to closed-system dynamics. This navigation of agent memory $\hat{U}$ must be distinguished from the evolution of quantum states in which, following external or internal stimulus, the memory is excited [7,14]. Learning as an internal navigation process corresponds to the colloquial notion of a learner desiring to quickly "make progress" rather than "marking time." For the agent's internal dynamics, we talk of a navigation process rather than a navigation problem that is to be solved, since ultimately the agent responds to its environment that is generally unknown and subject to unpredictable changes.

While the proposed PS model is characterized by an episodic and compositional memory (ECM), we here ignore the clip network aspect and restrict attention to a parameter update that is motivated from the basic scheme [7,8], which we apply to simple learning tasks involving an agent equipped with a quantum memory. We specifically reconsider some of the examples discussed in Refs. [7,15,16] in order to investigate to what extent the results can be reproduced. A comprehensive comparative numerical study of applications to computer games has been carried out in Ref. [17]. In contrast to the classical scheme, where the parameters are weights in a clip network, we here refrain from ascribing a particular role, they could play (e.g., in a quantum walk picture mentioned in Ref. [7]). Here, the parameters are simply controls, although in our examples they are defined as interaction strengths in a stack of layers constituting the agent memory $\hat{U}$. This choice of construction is, however, not essential for the main principle. From the viewpoint of RL [18,19], classical PS considers a type of problems, where an "agent" (embodied decision maker or "controller") learns from interaction with an environment (controlled system or "plant") to achieve a goal. The learning consists in developing a (generally stochastic) rule, the agent's "policy," of how to act depending on the situation it faces, with the goal to accumulate "reward" granted by the environment. In RL, the environment is anything outside of control of this decision making. The reward could describe, for example, pleasure or pain felt by an individual. It is generated within the individual's body but is beyond its control, and therefore is considered as originating in the *agent's* environment.

Another aspect is embodiment. A historical example is application-specific classical optical computing with a 4F-optical correlator. A more recent effort is neuromorphic computing, which aims at a very-large-scale integration

(VLSI)–based physical implementation of neural networks, whose simulation with a conventional computer architecture is inefficient. This becomes even more crucial for quantum systems, which may be implemented as superconducting solid-state devices, trapped ions or atoms, or waveguide-confined optical fields. Given the availability of a controllable quantum system, it is hence tempting to transform quantum-state-encoded sensory input and select actions based on measurement outcomes. While the parameter update is typically done by some standard linear temporal difference (TD) rule, the selection of actions is in classical algorithms governed by a separate stochastic rule that tries to balance exploration and exploitation. This stochastic rule is described in terms of a policy function that determines how the probabilities for choosing the respective actions depend on the value functions in RL, edge strengths in PS, or controls in direct policy approaches. Examples are the ε greedy" and the "softmax" rules. The quantum measurement here serves as a direct physical realization of an action selection, whose uncertainty allows us to incorporate both exploration and exploitation [20]. In our context, the resulting measurement-based (and hence effectively quadratic) policy forms an intermediate between the linear stochastic function used in Ref. [7] and the exponential softmax function applied in Ref. [15]. A measurement-based policy moreover can be tailored on demand by the way in which classical input is encoded as a quantum state. One could, e.g., apply mixtures of a pure state and a maximally mixed state to mimic an ε-greedy policy function, or one could use thermal input states to mimic an exponential function. In contrast to the value-function-based RL, our approach amounts to a direct policy search, where the agent-environment interaction employs a general state preparation $\rightarrow$ transformation $\rightarrow$ measurement scheme that reflects the kinematic structure of quantum mechanics.

This work is organized as follows. Section II serves to provide some first intuition into the connection between learning and navigation through unitaries. It may be skipped in the first instance, but becomes relevant later in the example Fig. 7 discussed in Sec. V A 2. While Sec. III provides a detailed description of the agent framework, Sec. IV defines the learning rule and introduces the glow parameter. Section V discusses two examples, namely an invasion game in Sec. V A, where the environment provides a feedback at each cycle, and a grid world in Sec. V B, where feedback is delayed to the end of a multicycle episode to illustrate the role of glow. Section VI briefly outlines how features such as internal loops mentioned in Sec. III may be employed to empirically estimate the gradient or instead use finite difference alternatives. A summary and outlook is given in Sec. VII. Additional technical details can be found in two appendixes.

II. RL AS A REWARD-DRIVEN NAVIGATION OF THE AGENT MEMORY

Consider the specific task of mapping input states $|s_i\rangle$ by means of a controllable unitary $\hat{U}$ to outputs $|a_i\rangle$. Under the (restrictive) assumption, that for each input there is exactly one correct output, the task is to learn this output from interaction with an environment. In our context, the $|s_i\rangle$ ($|a_i\rangle$) are regarded as encoded percepts (actions), while $\hat{U}$ acts as memory of the learned information and can finally accomplish the mapping as an embodied "ad hoc" computer or an "oracle," which is similar to adjusting a unitary as a means to solve a given task [21].

Consider (i) the case where there is only one possible input state $|s\rangle$. If the task is the navigation of the output state $\hat{\varrho} = \hat{U}|s\rangle\langle s|\hat{U}^\dagger$ by means of $\hat{U}$ to a desired destination state $|a\rangle$, a learning agent has to realize the maximization of the conditional probability $p(a|s) = \langle a|\hat{\varrho}|a\rangle$ by tuning $\hat{U}$. The intuition behind this is that p is bounded and if $\hat{U}(\boldsymbol{h})$ depends analytically on some control vector $\boldsymbol{h}$, the gradient with respect to $\boldsymbol{h}$ must vanish at the maximum of p. To give a simple example, we assume that $\hat{U}(t)$ depends (rather than on $\boldsymbol{h}$) on a single real parameter t in a continuous and differentiable way such that it obeys the Schrödinger equation $\frac{d}{dt}\hat{U} = -i\hat{H}\hat{U}$ with the state boundary conditions $\hat{\varrho}(0) = |s\rangle\langle s|$ and $\hat{\varrho}(t_\mathrm{F}) = |a\rangle\langle a|$. This gives

$$\frac{dp(t)}{dt} = 2\mathrm{Re}\langle a|\Big(\frac{d}{dt}\hat{U}\Big)|s\rangle\langle s|\hat{U}^\dagger|a\rangle \tag{1}$$

$$= 2\mathrm{Im}\langle a|\hat{H}\hat{\varrho}(t)|a\rangle, \tag{2}$$

so that indeed

$$\frac{dp(t)}{dt}|_{t=t_\mathrm{F}} = 2\mathrm{Im}\langle a|\hat{H}|a\rangle = 0. \tag{3}$$

Any algorithm that results in a $\hat{U}$ such that p approaches 1 accomplishes this task.

Assume now that (ii) we are required to transform a given orthonormal basis (ONB) $\{|s_i\rangle\}$ into another given ONB $\{|a_i\rangle\}$ of a vector space of same dimension, but we are not told which state is to be transformed into which other state. We could build a quantum device that implements some unitary $\hat{U}_\mathrm{T}$ such that $|a_i\rangle = \hat{U}_\mathrm{T}|s_i\rangle$. Preparing the system in state $|s_i\rangle$ and measuring in the second basis gives outcome $|a_i\rangle$. One may consider the problem as a (trivial) learning task, namely that of an identical mapping of the state indices i. However, if we do not know from the beginning what kind of mapping the solution is, we have to learn it. In our quantum device, we would tune $\hat{U}$ until it gives the desired measurement statistics. Inspired by Ref. [7], we call this task "invasion game." To solve it, we initialize the device in states $|s_i\rangle$ chosen randomly from the given ONB, while the measurement is done in the second ONB formed by the $|a_i\rangle$. The algorithm will drive $\hat{U}$ to some unitary $\hat{U}_2'\hat{U}_\mathrm{T}\hat{U}_1$, where $\hat{U}_1$ ($\hat{U}_2'$) are undetermined unitaries which are diagonal in the basis $\{|s_i\rangle\}$ ($\{|a_i\rangle\}$).

If (iii) the percept states are random, this phase freedom is removed up to a global phase. In the simplest case, we draw the initial states of the device from an "overcomplete" basis, where the set of all possible states is linearly dependent. For a n-level system, this can be accomplished by (randomly) choosing n SU(n) unitaries. During each state initialization, we then take one $\hat{U}_\mathrm{R}$ from this set, a random $|s_i\rangle$ from our first ONB, and then prepare the device in a state $\hat{U}_\mathrm{R}|s_i\rangle$. Consequently, the measurement is done in a transformed basis formed by the $\hat{U}_\mathrm{T}\hat{U}_\mathrm{R}\hat{U}_\mathrm{T}^{-1}|a_i\rangle$ rather than the $|a_i\rangle$ themselves.

In this sense, the navigation of (i) a given input state, (ii) a given ONB, and (iii) random states can be described as a navigation of unitaries $\hat{U}$ with a varying amount of freedom. While formally, all three cases (i)–(iii) can be considered as

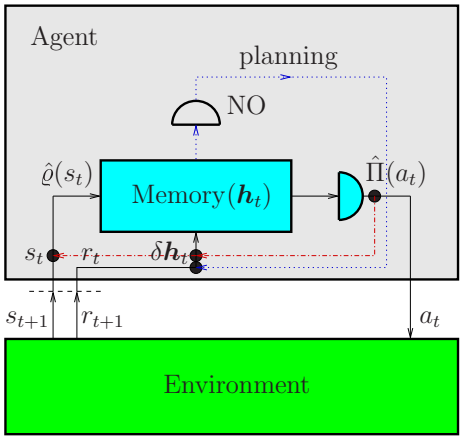

FIG. 1. Agent-environment interaction as a feedback scheme. The s are percepts, which initialize the agent's memory in a quantum state $\hat{\varrho}(s)$. Choice of an action a is made by a measurement process as described by a given POVM $\hat{\Pi}$. Depending on the rewards r given by the environment, the memory is updated at the end of each cycle t. The memory can also be modified by internal loops based on a numerical objective NO (dotted line) or measurements (dash-dotted line).

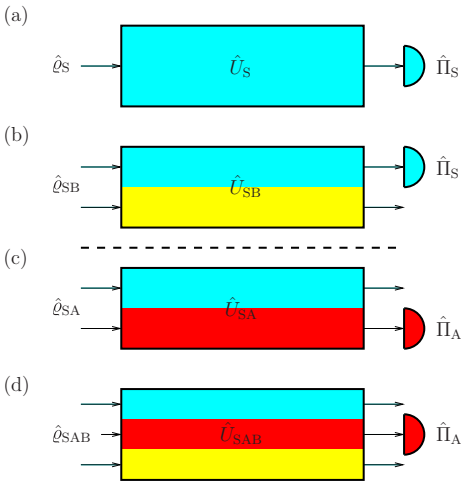

FIG. 2. Examples for the agent's memory as shown in Fig. 1. (a) Unitary evolution of the percept states, (b) open system evolution due to interaction with a bath B, (c) composite memory with coupled subsystems for percept (S) and action (A) variables, (d) extends panel (c) to an open system evolution analogous to panel (b) extending panel (a). Further ancilla systems may be added (not shown), to account for, e.g., emotion degrees of freedom introduced in Ref. [7].

special cases of a navigation of $\hat{U}(\boldsymbol{h})$ to a point (B11), where a percept statistics-based fidelity (B9) becomes maximum, practically they can be accomplished in RL by means of the mentioned reward signal, independently of the availability of analytic solutions. In what follows, we consider $\hat{U}$ as a memory of an RL agent that solves tasks arising from its interaction with an environment.

III. A CYBERNETIC PERSPECTIVE

The scheme is depicted in Fig. 1. The agent is equipped with some quantum channel that acts as its memory whose properties can be modified by control parameters denoted by a vector $\boldsymbol{h}$. Examples of memory structures are listed in Fig. 2. In Fig. 2 and in what follows, we refer to the memory operation by means of some unitary $\hat{U}$ for notational simplicity. Since any quantum process can be treated as unitary on an enlarged space, this is not a conceptual restriction. The agent interacts with an external environment in discrete cycles t. At the beginning of a cycle, the agent receives (via sensors) some percept s, which it encodes as a quantum state $\hat{\varrho}(s)$, in which its memory is prepared. After transformation of $\hat{\varrho}(s)$ by the memory channel, a quantum measurement is performed, where we assume for simplicity that the positive-operator-valued measure (POVM) $\{\hat{\Pi}\}$ describing this measurement is fixed. Depending on the outcome of this measurement, an action a is selected and performed on the environment (via actuators), which completes the cycle. The environment reacts with a new percept and a reward r, which are perceived by the agent during the following cycle. Depending on the reward, some adjustments are made on the control parameters, which modify the properties of the memory channel (i.e., its "hardware"). This feedback loop is adapted from the classical schemes in Refs. [18] and [19], where the percepts s in Fig. 1 correspond to the states in Ref. [18]. The agent's interaction with the environment is here considered classical in the sense that percepts, actions, and rewards are classical signals. The environment itself is not specified; it could represent, e.g., an experiment performed on a quantum system. Note that the environment in Fig. 1 is not to be confused with the bath in Fig. 2, which affects the memory channel but is not considered part of the agent's "habitat."

In addition to the external loop, we may also equip the agent with two types of internal feedback loops, which allow the agent to undertake what corresponds to "planning steps" in RL and "reflection" in PS. One type is similar to the external loop in that it involves state initializations and measurements on the memory channel, but exploits that percepts, actions, and rewards can be recorded and reproduced as a consequence of their classicality. The second type of internal loop does not involve state evolutions but requires some mathematical model of the memory channel itself, which is used to directly calculate a numerical objective (NO), whose value is used to alter the control parameters. Figure 1 does not imply that all of these loops need to be simultaneously present; they are rather thought of either subprocesses within an overall agent scheme or possible modes of its operation. The numerical examples in this work will exclusively apply the external loop.

All three loops involve a parameter update $\delta\boldsymbol{h}$. In a "first-order" update, $\delta\boldsymbol{h}$ is proportional to some quantity that depends on the gradient $\nabla\hat{U}$ of $\hat{U}$ with respect to $\boldsymbol{h}$. This gradient can either be computed directly from a memory model $\hat{U}(\boldsymbol{h})$ (i.e., from some symbolic expression of $\nabla\hat{U}$ if available) or estimated from measurements. These "measurements" can be physical (POVM in Fig. 1) or numerical (NO in Fig. 1). For the estimation, one varies the components of $\boldsymbol{h}$ by a small amount and records the changes in the measured POVM or computed NO. Here are some elementary examples: (1a) A simulation of an external loop with a given model-based (i.e., analytic) $\nabla\hat{U}$ is performed in Sec. V A 1 (Fig. 4) for the case Fig. 2(c), in

Sec. V A 2 (Figs. 6 and 7) for the case Fig. 2(a), and in Sec. V B (Figs. 10 and 11) for the case Fig. 2(c). (1b) A simulation of an external loop with a POVM measurement-based $\nabla\hat{U}$ is carried out in Ref. [22] (Fig. 6) for the case Fig. 2(b). (2) An NO-based internal loop with a model-based $\nabla\hat{U}$ is considered in Ref. [23] for the case Fig. 2(b) and in Ref. [22] (Figs. 2–4) for the case Fig. 2(a). (3) The POVM-based internal loop in Fig. 1 can be used to estimate $\nabla\hat{U}$ in the absence of a model $\hat{U}(\boldsymbol{h})$ of the agent memory. To this end, one of the agent's possibilities consists in inserting a number of internal cycles between each external cycle, where it repeatedly prepares its memory in the latest percept state and observes how a variation $\delta\boldsymbol{h}$ affects the measurement statistics. A discussion of this will be given in Sec. VI. Beyond these examples, all three loops can be interlaced with each other in various ways, analogous to the wealth of approaches reviewed in Ref. [18].

IV. UPDATE RULE IN PARAMETER SPACE

For the cyclewise update of the control parameters $\boldsymbol{h}$ of the memory channel $\hat{U}$, we apply a rule

$$\boldsymbol{h}' = \boldsymbol{h} + \alpha r \sum_{k=0}^{t-1} (1-\eta)^k \boldsymbol{D}_{t-k}. \tag{4}$$

The number of components h_k can range from one ($\boldsymbol{h}$ scalar) to infinity ($\boldsymbol{h}$ may represent a function or a vector of functions), and the h_k can be assumed to be real valued without loss of generality. In Refs. [7,8], the components of $\boldsymbol{h}$ are the edge strengths of a directed graph representing a network of clips (the graph's vertices). While these clips are considered sequences of remembered percepts and actions, the network itself abstracts from the clip's internal contents. Our view of $\boldsymbol{h}$ as a control vector is one further simplification and generalization that may allow for but does not require the view of the memory as a network.

In (4), $\boldsymbol{h}$ and $\boldsymbol{h}'$ are the control vectors before and after the update at cycle t, respectively. $\alpha \geqslant 0$ is a (typically small) learning rate, and r is the reward given at cycle t. $\boldsymbol{D}_t$ is a difference vector. While some options for finite difference choices of $\boldsymbol{D}$ are outlined in Sec. VI, in all numerical examples within this work we restrict to the case, where $\boldsymbol{D}_t = \nabla_t$ is a short-hand notation for the gradient

$$\nabla_t = \nabla\, p(a|s)_t = 2\mathrm{Re}\langle \hat{U}^\dagger \hat{\Pi}(a) \nabla \hat{U}\rangle_t, \tag{5}$$

$$p(a|s) = \mathrm{Tr}[\hat{U}\hat{\varrho}(s)\hat{U}^\dagger\hat{\Pi}(a)] = \langle \hat{U}^\dagger\hat{\Pi}(a)\hat{U}\rangle, \tag{6}$$

with components $\frac{\partial p(a|s)}{\partial h_k}$ at cycle t. $p(a|s)$ is the probability of the obtained measurement outcome a under the condition of the respective cycle's percept state $\hat{\varrho}(s)$, where $\langle\cdots\rangle \equiv \mathrm{Tr}[\hat{\varrho}(s)\cdots]$ denotes the expectation value with respect to this state, and $\hat{\Pi}(a)$ is the member of the POVM that corresponds to measurement outcome a. The latter determines the action performed by the agent, and we use the same symbol for both. $(1-\eta)$ describes a backward-discount rate, which we have defined via a parameter $\eta \in [0,1]$ to allow comparison with the glow mechanism introduced in Ref. [8]. As mentioned above, the unitary transformation of the respective percept states $\hat{\varrho}(s_t)$ by the memory $\hat{U} = \hat{U}(\boldsymbol{h}_t)$ at cycle t in (6) refers in general to a larger (dilated) space. The dynamical semigroup of CPT maps proposed in Ref. [7] is included and can be recovered by referring to Fig. 2(d) [or alternatively Fig. 2(b)] and the assumption that

$$\hat{\varrho}(s_t) \quad (\equiv \hat{\varrho}_{\mathrm{SAB}}) = \hat{\varrho}_{\mathrm{SA}}(s_t) \otimes \hat{\varrho}_{\mathrm{B}}, \tag{7}$$

$$\mathrm{Tr}_{\mathrm{B}}[\hat{U}\hat{\varrho}(s_t)\hat{U}^\dagger] = \mathrm{e}^{\mathcal{L}_{\mathrm{SA}}\Delta T_t^{(\mathrm{mem})}}\hat{\varrho}_{\mathrm{SA}}(s_t), \tag{8}$$

where the physical memory evolution time $\Delta T_t^{(\mathrm{mem})}$ may depend on the cycle t for a chosen parametrization $\hat{U}(\boldsymbol{h})$ and must be distinguished from the agent response time that can additionally be affected by the potential involvement of internal loops in Fig. 1. The superoperator $\mathcal{L} = \mathcal{L}_{\mathrm{SA}}$, whose effect on $\hat{\varrho} = \hat{\varrho}_{\mathrm{SA}}$ is defined as a sum

$$\mathcal{L}\hat{\varrho} = -\mathrm{i}[\hat{H},\hat{\varrho}] + L\hat{\varrho}, \tag{9}$$

generates in Ref. [7] a quantum walk and is given by a Hamiltonian $\hat{H} = \sum_{\{j,k\}\in E}\lambda_{jk}(\hat{c}_{kj} + \hat{c}_{kj}^\dagger) + \sum_{j\in V}\epsilon_j\hat{c}_{jj}$ and a Lindbladian $L\hat{\varrho} = \sum_{\{j,k\}\in E}\kappa_{jk}(\hat{c}_{kj}\hat{\varrho}\hat{c}_{kj}^\dagger - \frac{1}{2}\{\hat{c}_{kj}^\dagger\hat{c}_{kj},\hat{\varrho}\})$, with $\hat{c}_{kj} = |c_k\rangle\langle c_j|$ performing transitions between clip states $|c_l\rangle \in \mathcal{H}_{\mathrm{SA}}$ along a graph $G = (V,E)$ consisting of a set V of vertices and a set E of edges. Since on the one hand we here do not intend to necessarily represent $\hat{U}$ by a clip network and on the other hand we do not want to exclude from the outset situations involving time-dependent or non-Markovian bath effects, we use the dilated $\hat{U}$ for simplicity instead. The set of all probabilities $p(a|s)$ in (6), i.e, the whole conditional distribution, then defines the agent's policy.

A reward given by the environment at time t raises the question of the extent to which decisions made by the agent in the past have contributed to this respective reward. A heuristic method is to attribute all past decisions, but to a lesser degree the further the decision lies in the past. (Considering the agent's life as a trajectory of subsequent percepts and actions, we could imagine the latest event trailing a decaying tail behind.) A detailed description of this idea is presented in Ref. [18] in the form of eligibility traces, which can be implemented as accumulating, replacing, clearing, or dutch [24] traces. In the context of PS, a similar idea has been introduced as an afterglow mechanism that can be implemented as edge or clip glow [8,15]. In stochastic gradient descent, a related technique is the momentum method [25]. In our context (4), we realize it by updating the control vector by a backward-discounted sum of gradients ∇_{t-k} referring to percepts and actions involved in cycles that happened k steps in the past. A sole update by the present gradient is included as limit $\eta = 1$, for which (4) reduces to $\boldsymbol{h}' = \boldsymbol{h} + \alpha r \nabla\, p(a|s)$. This special case is sufficient for the invasion game, which we will consider in Sec. V, because at each cycle, the environment provides a feedback on the correctness of the agent's decision by means of a nonzero reward. After that, we apply the general update (4) to a grid world task, where the agent's goal cannot be achieved by a single action, and where the long-term consequences of its individual decisions cannot be foreseen by the agent.

We conclude this section with some comments on how the update rule (4) is related to other existing approaches to RL. Since the latter is an umbrella term for problems that can be described as agent-environment interactions characterized by percepts, actions, and rewards, let us focus on the computational aspects and ignore the quantum implementation

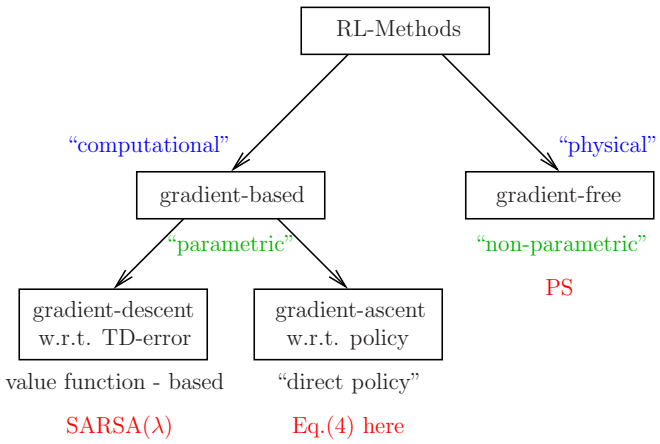

FIG. 3. Relation between the scheme discussed in this work and other existing approaches to RL problems. While the agent learning as a navigation of the memory "hardware" $\hat{U}$ governed by (4) is a computational gradient-based concept (unless a direct physical mechanism can be found for this purpose), the action selection is physical in the sense that it employs a generic preparation, transformation, and measurement of quantum states.

described in Sec. III. As summarized in Fig. 3, we may distinguish (a) between gradient-based methods, which mathematically compute a gradient of a quantity of interest relating to a given set of parameters, and gradient-free methods such as PS, which do not explicitly rely on a gradient. Apart from this, we may distinguish (b) between "parametric" methods, which apply a fixed set of parameters and methods, for which parameters (like edge strengths in PS) can be identified, but their number may not be fixed (since, e.g., in PS, the graph topology of the clip network may be subject to change), and hence we here refer to them as "nonparametric." Apart from this, we may distinguish (c) whether the action selection is done "computationally" (i.e., by an algorithm running on a general purpose computer) or by an *ad hoc* "physical" process (such as diffusion within a PS clip network). To keep things simple, in Fig. 3 we have roughly identified the parametric and computational with the gradient-based approaches: standard RL methods such as Q-learning or SARSA (for details see e.g., [18]) base their actions on running estimates of value functions (expected discounted sums of future rewards the agent is trying to maximize). At each cycle, the agent then adjusts its parameters' gradient descent with respect to a TD error (the mismatch between the respective reward expected by the agent and the one actually received). In contrast, the update rule (4) is gradient ascent with respect to the conditional probabilities $p(a|s)$, which are increased if the actions are rewarded. Since the $p(a|s)$ form the agent policy, (4) can be considered a "direct policy" method. "Tabular" methods (not separately highlighted in Fig. 3) are a special parametric case, in which the parameters are the running estimates themselves and which are explicitly stored by the algorithm as a table.

V. EXAMPLES

A. Invasion game

In what follows, we consider a simple invasion game as treated in Ref. [7]. An attacker randomly chooses one out of two possible symbols $\{\Leftarrow, \Rightarrow\}$ which signals the direction in which it intends to move. The chosen symbol may represent, e.g., a head turn and is visible to the defender, whose task is to learn to move in the same direction, which is required to block the attacker. We approach this learning task as an external loop in Fig. 1 with a closed system (i.e., bathless) memory [cases (a) and (c) in Fig. 2], described within a four-dimensional Hilbert space. The control parameters are updated according to (4) in the absence of gradient glow ($\eta = 1$). The update is done with an analytic $\nabla\hat{U}$ as described in Appendix A 2, where the memory consists of alternating layers, $\hat{U} = \cdots e^{-ih_3\hat{H}^{(1)}} e^{-ih_2\hat{H}^{(2)}} e^{-ih_1\hat{H}^{(1)}}$, with a given number of controls $h_1, \ldots, h_n$. At the beginning of the first cycle, the memory is initialized as identity. For the two Hamiltonians $\hat{H}^{(1)}$ and $\hat{H}^{(2)}$, we distinguish (I) a general case, where $\hat{H}^{(1)}$ and $\hat{H}^{(2)}$ are two given (randomly generated) four-rowed Hamiltonians acting on the total Hilbert space and (II) a more specialized case, in which they have the form

$$\hat{H}^{(1)} = \hat{H}_{\mathrm{S}}^{(1)} \otimes \hat{I}_{\mathrm{A}} + \hat{I}_{\mathrm{S}} \otimes \hat{H}_{\mathrm{A}}^{(1)}, \tag{10}$$

$$\hat{H}^{(2)} = \hat{H}_{\mathrm{S}}^{(2)} \otimes \hat{H}_{\mathrm{A}}^{(2)}, \tag{11}$$

where $\hat{H}_{\mathrm{S}}^{(1)}$, $\hat{H}_{\mathrm{A}}^{(1)}$, $\hat{H}_{\mathrm{S}}^{(2)}$, $\hat{H}_{\mathrm{A}}^{(2)}$ are four given (randomly generated) two-rowed Hamiltonians acting on the percept (S) and action (A) subsystems, respectively, with $\hat{I}$ denoting the identity. The latter case (II) refers to a physical implementation of Fig. 2(c) as a bath-mediated interaction of the S and A subsystems that is obtained from the setup Fig. 2(d) by eliminating the bath [22]. It has been included here to demonstrate that this special structure as considered in Ref. [22] may be applied in the present context, but this is not mandatory. While the Hamiltonians have been chosen in both cases (I) and (II) at random to avoid shifting focus toward a specific physical realization, in an experimental setup, the respective laboratory Hamiltonians will take their place (assuming that they generate universal gates in the sense of Ref. [26], which is almost surely the case for a random choice).

1. *Two percepts → two actions*

We start with a basic version of the game with two possible percepts (the two symbols shown by the attacker) and two possible actions (the two moves of the defender). For each percept, there is hence exactly one correct action, which is to be identified. The memory applied is shown in Fig. 2(c), and the different input states are

$$\hat{\varrho} = \hat{\varrho}_{\mathrm{S}} \otimes \hat{\varrho}_{\mathrm{A}}, \quad \hat{\varrho}_{\mathrm{S}} = |s\rangle\langle s|, \tag{12}$$

$$\hat{\varrho}_{\mathrm{A}} = p_{\mathrm{coh}}|\varphi\rangle\langle\varphi| + (1 - p_{\mathrm{coh}})\frac{1}{d_{\mathrm{A}}}\hat{I}_{\mathrm{A}}, \tag{13}$$

$$|\varphi\rangle = \frac{1}{\sqrt{d_{\mathrm{A}}}} \sum_a |a\rangle, \tag{14}$$

where $d_{\mathrm{A}} = \dim\mathcal{H}_{\mathrm{A}} = 2$ is given by the number of actions. $|s\rangle$ and $|a\rangle$ can both be one of the two orthonormal states $|0\rangle$ or $|1\rangle$ of the S and A subsystem, respectively. The POVM consists of the elements

$$\hat{\Pi}(a) = \hat{I}_{\mathrm{S}} \otimes |a\rangle_{\mathrm{A}}\langle a|. \tag{15}$$

Choosing the correct (wrong) action [i.e., $a = s$ ($a \neq s$) in (15) and (12)] returns a reward of $r = +1$ ($r = -1$).

Figure 4 shows the average reward $\bar{r}$ received at each cycle, where the averaging is performed over an ensemble of 10^3 independent agents. $(\bar{r}+1)/2$ is hence an estimate of the defender's probability to block an attack. Referring to pure states in (13), Fig. 4(a) shows the increase of learning speed with the number of controls. Significant learning progress begins only after some initial period of stagnation. From the viewpoint of control theory, the identity, in which the memory is initialized, may lie on a "near-flat ground" (valley), which must first be left before progress can be made [27]. Asymptotically, perfect blocking can be achieved once the memory becomes controllable, i.e., if the number of controls equals (or exceeds) the number of group generators. Figure 4(b) demonstrates the need of a pure input state $\hat{\varrho}_{\mathrm{A}}$ in (13) of the action subsystem A rather than an incoherent mixture. After the agent in Fig. 4(b) had some time to adapt to the attacker, the meaning of the symbols is suddenly interchanged, and the agent must now learn to move in the opposite direction. This relearning differs from the preceding learning period in the absence of the mentioned initial stagnation phase, which supports the above hypothesis of the proposed valley, the agent has left during the initial learning. This plot is motivated by Fig. 5 in Ref. [7] describing classical PS. While Figs. 4(a) and 4(b) refer to case (I) described before (10), Fig. 4(c) refers to the restricted case (II), which appears to impede learning. In the simulations of the following Sec. V A 2, which all refer to case (II), this is resolved by applying a negative reward that is 10 times larger for each wrong action. This demonstrates the flexibility in approaching RL problems offered by the freedom to allocate rewards in a suitable way.

2. *Four percepts → four or two actions*

We now consider a version with four percepts, referring to an attacker presenting each of its two symbols in two colors at random. Since we want to keep the Hilbert space dimension unchanged (rather than doubling it by adding the color category) for better comparison of the effect of the number of controls on the learning curve, we must apply a memory as shown in Fig. 2(a). The four percepts are encoded as tensor products of orthonormal projectors

$$\hat{\varrho}_{jk} = |j\rangle\langle j| \otimes |k\rangle\langle k|, \tag{16}$$

where $j = 0,1$ ($k = 0,1$) refers to the symbol (color). The POVM operators are the four projectors

$$\hat{\Pi}_{jk} = \hat{U}_{\mathrm{T}} \hat{\varrho}_{jk} \hat{U}_{\mathrm{T}}^{\dagger}, \tag{17}$$

where $\hat{U}_{\mathrm{T}}$ is a given (randomly generated) four-rowed target unitary acting on the total system. The memory in Fig. 2(a) is hence still composed of two subsystems referring to the two percept categories "symbol" and "color," but both subsystem's initial state depends on the respective percept, and both are measured afterward. The differences between the setup discussed in the previous Sec. V A 1 and the two setups discussed in the present Sec. V A 2 are summarized in Fig. 5.

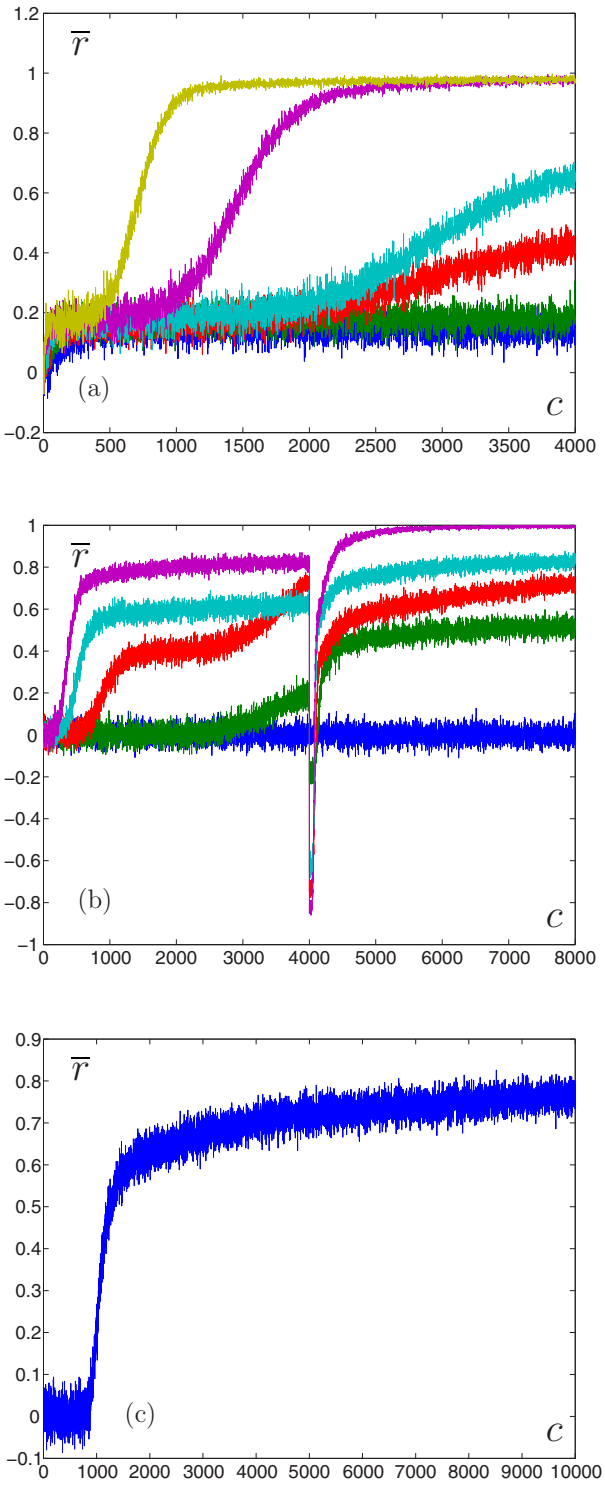

FIG. 4. Average reward $\bar{r}$ as a function of the number of cycles c for an invasion game (two symbols, two moves), learning rate $\alpha = 10^{-3}$, with a reward of $+1$ (-1) for a correct (false) move, averaged over 10^3 agents. The input states and POVM are given by Eqs. (12) and (15), respectively. (a) The six graphs from bottom to top correspond to 1, 2, 3, 4, 8, and 16 controls, respectively, with $p_{\mathrm{coh}} = 1$ in (13). (b) 16 controls, where after 4×10^3 cycles the meaning of the symbols is reversed. The five graphs from bottom to top correspond in Eq. (13) to $p_{\mathrm{coh}} = 0$, 0.25, 0.5, 0.75, and 1, respectively. Panel (c) shows the case of 32 controls and $p_{\mathrm{coh}} = 1$, but refers to case (II) described by (10) and (11), whereas Figs. 4(a) and 4(b) refer to case (I).

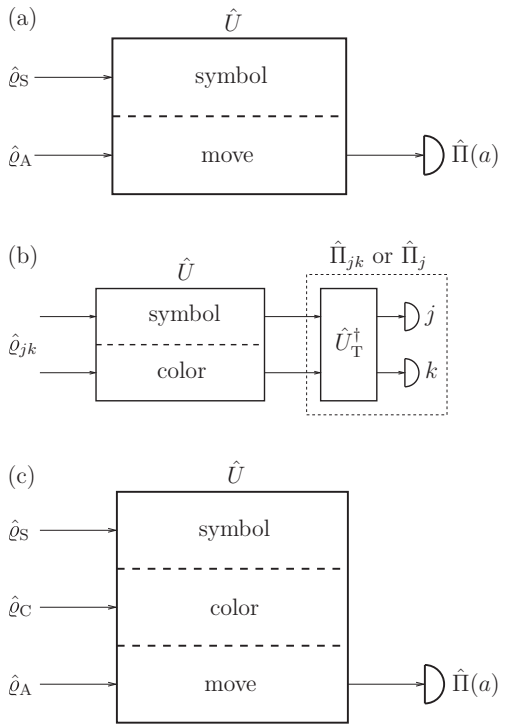

FIG. 5. Setups for the invasion game as investigated numerically in Sec. V A. Setup (a) involves two percepts (symbols) and two actions (moves) as discussed in Sec. V A 1 (Fig. 4). Setup (b) involves four percepts consisting of two two-colored symbols and two measurements yielding four different outcomes described by $(j,k = 0,1)$ as discussed in Sec. V A 2 and determining either four [Figs. 6(a) and 7] or—if outcome k is ignored—two [Fig. 6(b)] possible actions (moves). While setup (a) refers to Fig. 2(c), setup (b) must refer to Fig. 2(a), if we want to keep the same Hilbert space dimension of four for both setups, which allows better comparison of the effect of the number of controls on the learning curve. Setup (c) involves a continuum of percepts consisting of two arbitrary-colored symbols and two actions (moves) as discussed in Sec. V A 3 (Fig. 8). In setup (c), separate subsystems are used for all three categories, hence it refers to Fig. 2(c), and the Hilbert space dimension becomes 8.

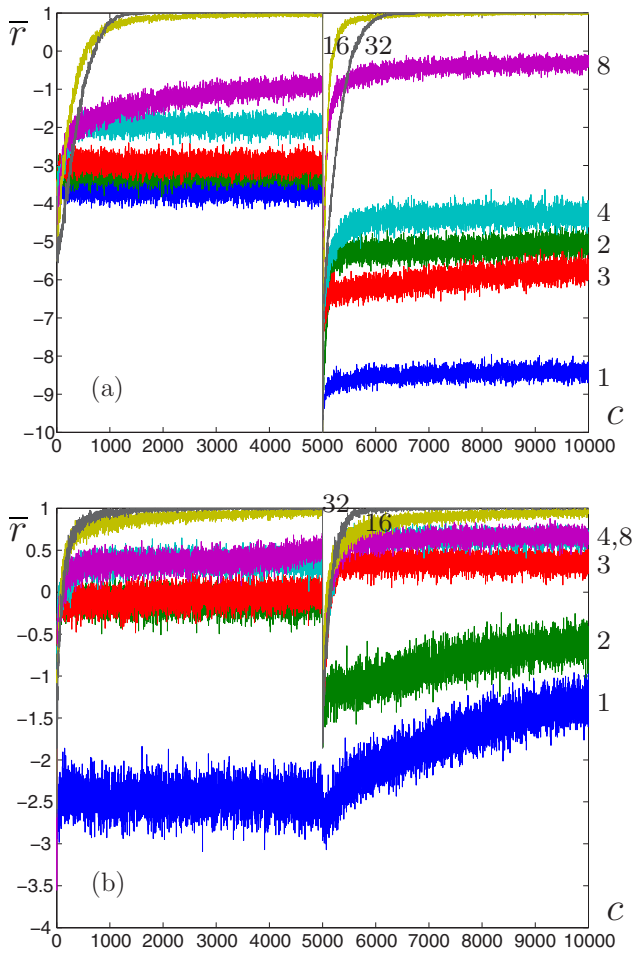

FIG. 6. Average reward $\overline{r}$ as a function of the number of cycles c for an invasion game (two symbols in two colors), learning rate $\alpha = 10^{-2}$, with a reward of $+1$ (-10) for a correct (false) move, averaged over 10^3 agents. The input states and POVM are given by Eqs. (16) and (17), respectively. The graphs correspond to 1, 2, 3, 4, 8, 16, and 32 controls, as marked on the right. (a) Out of four possible moves, the defender must learn the correct one for each symbol and color. After 5×10^3 cycles, the meanings of the symbols as well as the colors are reversed. (b) Out of two possible moves, the defender must learn the correct one for each symbol, whereas the color is irrelevant. For the first 5×10^3 cycles, only symbols in a single color are presented, whereas for the remaining cycles, they are shown randomly in both colors.

Figure 6 shows the average reward $\overline{r}$ received at each cycle, where the averaging is performed over an ensemble of 10^3 independent agents, analogous to Fig. 4. Note that in this Sec. V A 2, all figures refer to case (II) described by (10) and (11), where S and A now denote symbol and color, respectively. To account for this [cf. the comments on Fig. 4(c) above], a reward of $r = -10$ (instead of -1) is now given for a wrong action. The estimate of the defender's probability to block an attack is hence now $(\overline{r} + 10)/11$.

In Fig. 6(a), the defender can choose between four moves, where for each percept, there is exactly one correct action [i.e., detecting $\hat{\Pi}_{jk}$ $(\hat{\Pi}_{j'k'} \neq \hat{\Pi}_{jk})$ for $\hat{\varrho}_{jk}$ in (17) and (16) returns a reward of $r = +1$ $(r = -10)$]. After 5×10^3 cycles, symbol j and color k are read as symbol $1 - j$ and color $1 - k$, respectively, similar to the manipulations in Fig. 5 in Ref. [7]. In Fig. 6(b), the defender can choose between two moves, where for each symbol (relevant category), there is exactly one correct action, irrespective of its color (irrelevant category) [i.e., detecting $\hat{\Pi}_j = \sum_{k=0}^{1} \hat{\Pi}_{jk}$ $(\hat{\Pi}_{j'} \neq \hat{\Pi}_j)$ for $\hat{\varrho}_{jk}$ in (17) and (16) returns a reward of $r = +1$ $(r = -10)$]. The second color is added only after 5×10^3 cycles, analogous to Fig. 6 in Ref. [7]. [Note that the mentioned initial stagnation phase in Fig. 4 is not visible in Fig. 6, which is attributed to the choice of parameters (rewards), accelerating the initial learning.]

Figures 4(b) and 6 are all motivated by Figs. 5 and 6 in Ref. [7] and confirm that the agent's adaptation to changing environments is recovered in our quantum control context. In addition, Figs. 4(a) and 6 show the behavior of an underactuated memory, where the number of controls is insufficient for its full controllability. Since a $U(n)$ matrix is determined by n^2 real parameters, and a global phase can be disregarded

(so that we can restrict to SU(n) matrices), $n^2 - 1$ controls are sufficient, i.e., 15 for our invasion game, as mentioned above.

In (17), the measurements are made in a basis rotated by a randomly given unitary $\hat{U}_{\mathrm{T}}$, which serves two purposes. On the one hand, it is required to ensure that the agent starts at the beginning of the first cycle with a policy that does not give exclusive preference to certain actions that follow from symmetries of the (identity-) initialized memory. This is a flaw of Fig. 2(a) and can be overcome by using Fig. 2(c) instead (cf. a more detailed discussion in the grid world example below). On the other hand, $\hat{U}_{\mathrm{T}}$ serves as a given target in our discussion, Sec. II, where we consider the agent learning as a navigation of its memory $\hat{U}$; cf. also Fig. 5(b). Figure 7 compares the case, where the agent is always fed with percept states drawn from one single ONB defined via (16) with the case, where the percept states are drawn randomly, i.e., taking $\hat{U}_{\mathrm{R}}\hat{\varrho}_{jk}\hat{U}_{\mathrm{R}}^{\dagger}$ with a random unitary $\hat{U}_{\mathrm{R}}$ as explained in Sec. II instead of (16). Note that in Fig. 7, we generate a new random $\hat{U}_{\mathrm{R}}$ at each cycle, although a fixed set of $\dim\mathcal{H}$ (four in our case) such $\hat{U}_{\mathrm{R}}$ is sufficient as mentioned in Sec. II. Fidelity F and squared distance D are defined in (B2) and (B1), where $\hat{U}$ represents the agent memory and $\hat{U}_{\mathrm{T}}$ is the target unitary. Each cycle's update constitutes a single navigation step in the unitary group [$U(4)$ in our example]. If, for a single ONB, after a number of cycles, the average reward has approached unity, $\hat{U}$ has reached a close neighborhood of any unitary of the form $\hat{U}_2'\hat{U}_{\mathrm{T}}\hat{U}_1$, where $\hat{U}_2' = \hat{U}_{\mathrm{T}}\hat{U}_2\hat{U}_{\mathrm{T}}^{\dagger}$ with $\hat{U}_1$ and $\hat{U}_2$ being undetermined four-rowed unitary matrices diagonal in the common eigenbasis of the $\hat{\varrho}_{jk}$ (i.e., the "computational basis"). Figure 7(a) shows that for a solution of the invasion game, a fixed ONB is sufficient. Drawing the percept states randomly, so that the set of all percept states is linearly dependent, does not affect the agent's ability to achieve perfect blocking efficiency but does slow down the learning process. The single ONB case allows for a larger set of $\hat{U} = \hat{U}_2'\hat{U}_{\mathrm{T}}\hat{U}_1$ with respect to $\hat{U}_{\mathrm{T}}$, as becomes evident in Fig. 7(b), so that navigation of $\hat{U}$ from the identity to a member of this set takes less time (as measured in cycles). The only freedom left in the case of multiple ONBs is a global phase of $\hat{U}$, which remains undefined: navigation of $\hat{U}$ toward $\hat{U}_{\mathrm{T}}$ with respect to the squared Euclidean distance D is not required for the learning tasks discussed, as evidenced by Fig. 7(c).

3. *Never-ending-color scenario*

In Sec. V A 2, we considered the case, where the symbols are presented in two different colors, as depicted in Fig. 5(b). The original motivation for introducing colors as an additional percept category was to demonstrate the agent's ability to learn that they are irrelevant [7]. In contrast, [16] present a "never-ending-color scenario," where at each cycle, the respective symbol is presented in a new color. It is shown that while the basic PS agent is in this case unable to learn at all, it becomes able to *generalize* (abstract) from the colors, if it is enhanced by a wild-card mechanism. The latter consists in adding an additional (wild card "#") value to each percept category and inserting between the input layer of percept clips and the output layer of action clips hidden layers of wildcard percept clips, in which some of the percept categories attain the wild-card value. The creation of these wild-card clips follows predefined deterministic rules, and the transitions from percept to action clips take then place via the hidden layers. (The notion of layers in the general ECM clip network Fig. 2 in Ref. [7] follows from restricting to clips of length $L = 1$).

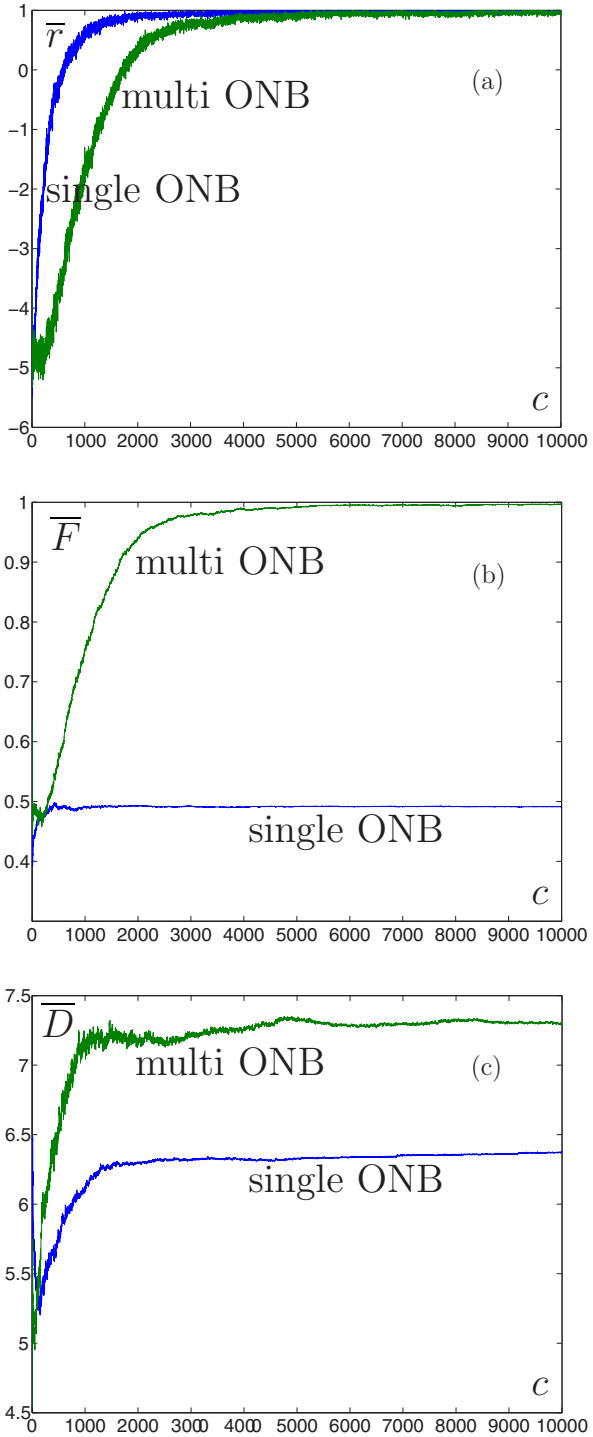

FIG. 7. (a) Reward r, (b) fidelity F defined in (B2), and (c) squared distance D defined in (B1), as a function of the number of cycles c, where the overline denotes the ensemble average over 10^3 agents for the setup as in Fig. 6(a) (i.e., four percepts and four actions) with 16 controls but without the reversal of meaning. The initial memory states are drawn from either a single or multiple orthonormal bases.

Since the use of wild-card clips is an integrated mechanism within PS (inspired by learning classifier systems), the question is raised how similar ideas could be implemented in our context. For a memory, Fig. 2(c), we could, e.g., attribute one of the levels (such as the respective ground state) of the quantum system of each percept category S_i to the wild-card-level $|\#\rangle_i$, so that the percept space $\mathcal{H}_S = \otimes\mathcal{H}_{S_i}$ is enlarged to $\mathcal{H}_S = \otimes(\mathcal{H}_{S_i} \oplus \mathcal{H}_{\#_i})$, where the $\mathcal{H}_{\#_i}$ are one dimensional.

Instead of this, let us simply make use of the *built-in* generalization capacity of a quantum agent resulting from its coding of percepts as quantum states, which is much in the sense of Sec. 8 in Ref. [18], where the percepts can be arbitrarily real valued rather than being drawn from a countable or finite value set. Consider the setup shown in Fig. 5(c), whose percept system includes a symbol and a color category and refers to a memory structure, Fig. 2(c). To allow for infinite colors, we could apply a color quantum system with infinite levels $\mathcal{H}_\infty$ (such as an oscillator-type system), which is initialized at each cycle in a new state drawn from a fixed ONB (such as a higher number state for an oscillator-type system). While such a scheme becomes more challenging to control, because the control vector $\boldsymbol{h}$ has an infinite number of components [we may replace it with a continuous control function $h(t)$], it still ignores the fact that colors (as most percept categories in general) are not countable. With this in mind, we can take the notion of colors literally and, to put it simply, code them in some appropriate color space such as RGB, where three parameters correspond to the red, green, and blue signals of the agent's eye sensors. This suggests encoding a color as a mixed state of a two-level system, which is also given by three real-valued parameters (determining its location in the Bloch ball). The generalization from two colors to all RGB colors then corresponds to the generalization from a classical to a quantum bit. In our setup, it is hence sufficient to apply a two-level system for the color category and initialize it at each cycle in a randomly chosen *mixed* state $\hat{\varrho}_C$ (for never-ending colors) rather than a (pure) state randomly drawn from a single ONB (for two colors), whereas no changes are required on the agent's memory configuration itself. Figure 8 demonstrates the learning process. Similar to Fig. 7, random initialization slows down the learning process, so that we restrict to a single agent in Fig. 8, rather than an ensemble average. As illustrated in Fig. 8(a), the agent's response becomes near deterministic after about 10^6 cycles, irrespective of the color. Figure 8(b) illustrates in the example of the Euclidean length of the control vector $|\boldsymbol{h}| = \sqrt{\boldsymbol{h}^{\mathrm{T}} \cdot \boldsymbol{h}}$, that the navigation, which starts at $\boldsymbol{h}_0 = \boldsymbol{0}$, eventually comes to rest. While the random $\hat{\varrho}_C$ are drawn such that a positive probability is attributed to every volume element in the Bloch ball, we did not care about drawing them with a uniform probability density, since mapping of an RGB space of color (as a perceptual property) to the Bloch ball is not uniquely defined.

The ability to learn to distinguish between relevant S_{rel} and an arbitrary number of irrelevant percept categories S_{irr} as discussed in Ref. [16] is of particular relevance for a quantum agent, where the irrelevant percept categories can be understood as adopting the role of a bath as shown in Figs. 2(b) and 2(d). Here, a formal solution consists in a decoupled $\hat{U}_{\mathrm{SA}} = \hat{U}_{\mathrm{S_{rel}A}} \otimes \hat{U}_{\mathrm{S_{irr}}}$.

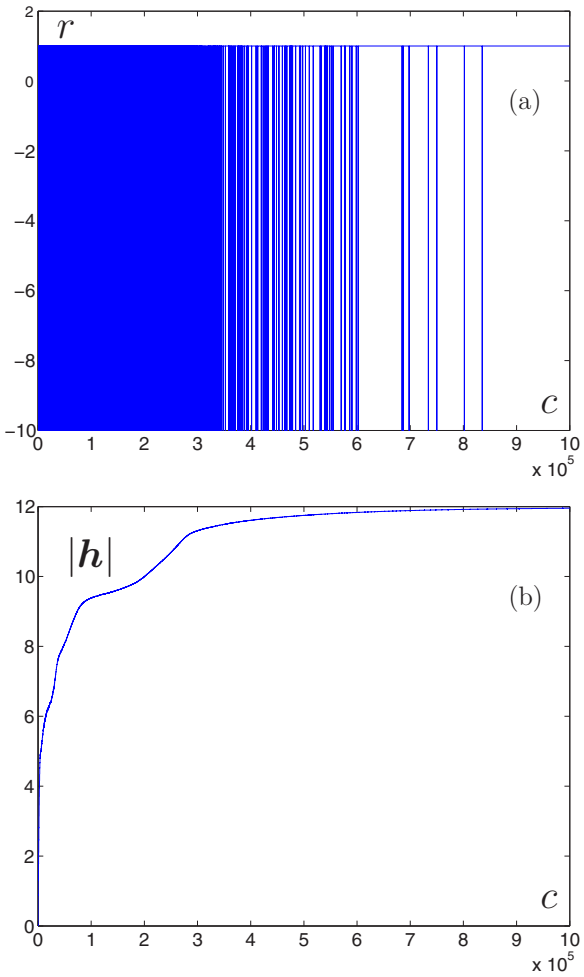

FIG. 8. (a) Reward r and (b) length $|\boldsymbol{h}|$ of the control vector as a function of the number of cycles c for a single agent, Fig. 5(c), playing an invasion game with two symbols, presented in a continuum of never-ending colors, and two moves. A reward of $+1$ (-10) is given for each correct (false) move. The agent applies 64 controls with a learning rate of $\alpha = 10^{-2}$ in an alternating layer scheme, Appendix A 2, defined by two (Schmidt-orthonormalized) eight-rowed random Hamiltonians.

B. Grid world

In what follows, we consider an arrangement of eight grid cells as shown in Fig. 9. The agent's task is to find the shortest route to a goal cell G, where at each step, only moves to an adjacent cell in four directions (left, right, up, down) are allowed. If the agent hits a boundary of the grid or the black cell, which is considered an obstacle, its location remains unchanged.

This external classical navigation task constitutes a learning problem, because situations (present location) must be mapped to decisions (direction to go). The agent only perceives whether or not it has arrived at the goal cell. It has no access to a "bird's perspective," which would allow immediate exact location of

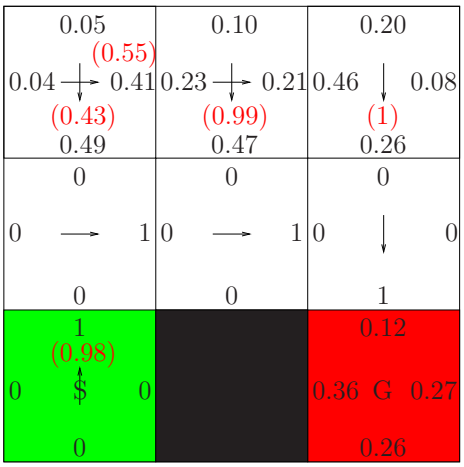

FIG. 9. (3×3)-grid world with an obstacle (black cell). The arrows show the optimal policy for the shortest path to G, and the numbers present the policy numerically obtained with the agent Fig. 10(d) after 10^4 episodes. The red numbers in parentheses are the relevant different values obtained after 10^5 episodes, if the agent starts each episode from a random cell, rather than always at S.

the goal. It also has no access to a measure of goal distance or fidelity (as in the case of the internal NO-based loop regarding its own quantum memory in Fig. 1), which prevents the use of external gradient information that could be obtained by testing the nearest neighbourhood of the present location. One can thus distinguish two objectives: (a) locating the goal and (b) finding a shortest route to it. This task constitutes a RL-type problem, whose composite "two-objective" structure is approached by nesting iterations. The individual action selections, i.e., choices of moves, correspond to the cycles in Fig. 1. Sequences of cycles form episodes, which are terminated only once objective (a) has been solved. Objective (b) is solved by sequences of episodes, which allow the agent to gradually solve objective (a) more efficiently and find an optimal policy. In Fig. 9, the policy consists of a set of four probabilities for each cell, with which a corresponding move should be made from there. The optimal policy corresponding to the shortest route to G is indicated by the arrows in Fig. 9.

This grid world extends the above decision game in two aspects: (a) The optimal policy is in contrast to the decision game not deterministic, as indicated by the double arrows in the upper left and middle cells in Fig. 9. (b) Depending on where the agent starts, more than a single move is required to reach G in general, preventing the agent from obtaining immediate environmental feedback on the correctness of each individual move it makes. This second aspect leads to the mentioned notion of episodes. In what follows, we always place the agent at a fixed start cell S at the beginning of each episode, which is sufficient for learning the shortest path from S to G. While in the invasion game, episodes and cycles are synonyms, here an episode is longer than a single cycle, since at least four moves are required to reach G from S.

When designing the agent's memory structure in the sense of Fig. 2, we must take into account that the unitarity of the state transformation $\hat{U}_S$ in Fig. 2(a) places restrictions on the percept encodings and the action measurements, since $\hat{U}_S$ maps an ONB into another one. If we encode in Fig. 9 each cell location as a member of a given ONB in an eight-dimensional system Hilbert space $\mathcal{H}_8$ and perform a naive symmetric $\mathcal{H}_8 = \mathcal{H}_2 \oplus \mathcal{H}_2 \oplus \mathcal{H}_2 \oplus \mathcal{H}_2$ measurement for action selection, where the four two-dimensional subspaces correspond to right, down, left, and up moves, we cannot properly ascribe the upper left and upper middle cells, because the right and downward pointing actions are already ascribed to the remaining four white cells. One may either try to construct a learning algorithm that exploits the fact that the two mentioned cells are off the optimal path from S to G so that the agent quickly ceases to visit them or construct a new POVM such as a $\mathcal{H}_8 = \mathcal{H}_1 \oplus \mathcal{H}_3 \oplus \mathcal{H}_3 \oplus \mathcal{H}_1$ measurement, where two three-dimensional subspaces correspond to right and down, and two one-dimensional subspaces correspond to left and up moves. These possibilities require insight into the specifics of this problem and are not generalizable. In addition to that, Fig. 2(a) requires initialization of the agent memory in a random unitary to ensure it starts with a policy that does not give exclusive preference to certain actions that follow from symmetries of the initial $\hat{U}_S$ (such as the identity $\hat{U}_S = \hat{I}_S$). If we want to avoid invoking a bath as in Fig. 2(b), we hence must resort to Fig. 2(c), which here implies a factorization $\mathcal{H} = \mathcal{H}_S \otimes \mathcal{H}_A$ of $\mathcal{H}$ into an eight-dimensional $\mathcal{H}_S$ for encoding the grid cells and a four-dimensional $\mathcal{H}_A$ for encoding the four actions. If we encode the cells and actions as members of some ONB in S and A, then initializing the agent's memory as identity, $\hat{U}_{SA} = \hat{I}_{SA}$, and the initial action states as in (14) ensures that the agent starts at the beginning of the first episode with a policy that assigns the same probability to all possible actions.

In Figs. 10 and 11, we investigate the episode length which is defined as the number of cycles per episode. Rather than performing an ensemble average, we consider individual agents. These agents are described by (4) with a learning rate of $\alpha = 10^{-1}$ and varying amounts of gradient glow ($\eta \leqslant 1$). The number of episodes equals the number of times the agent is allowed to restart from S, whereas the time passed equals the sum of episode lengths. The episode length can be infinite but not smaller than four, the length of the shortest path from S to G.

Figure 10 shows evolutions of episode lengths with the number of episodes, where we have set a maximum of 10^4 episodes. As explained, each episode starts at S and ends only when G has been reached.

Figure 10(f) shows for comparison the lengths of 10^4 random walks through the grid of an agent whose learning has been disabled by always setting the reward to 0. The average number of 54.1 steps to reach G from S is shown in Figs. 10 and 11 as a dashed line for comparison. In Figs. 10(a)–10(e), a positive reward of $r = 1$ is given for hitting G. While in Fig. 10(a), the reward is always zero before G has been hit, in Fig. 10(c) hitting a boundary is punished with a negative reward of $r = -10$, which slightly improves the agent's performance. [Note that all plots are specific to the respective learning rate (here $\alpha = 10^{-1}$), which has been chosen by hand to observe an improvement within our 10^4 episode window and at the same time minimizing the risk of oversized learning steps. While in general, the learning rate is gradually decreased (cf. the

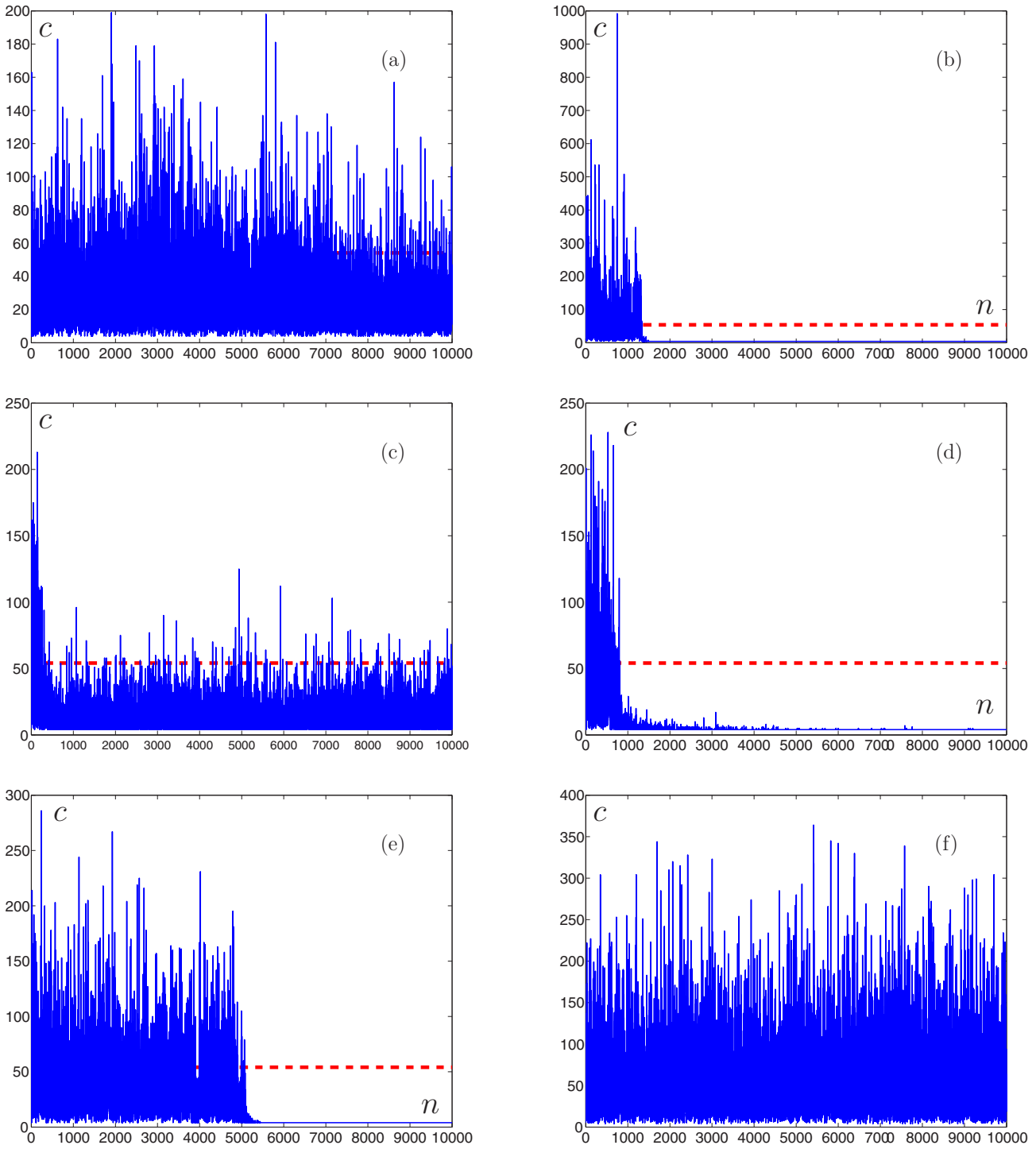

FIG. 10. Episode lengths (number of cycles c per episode) as a function of the number of episodes n for the (3×3)-grid world as shown in Fig. 9. The plots illustrate the effect of gradient glow for single histories of individual agents. (a) The agent receives a reward $r = 1$ when it has found the goal, without gradient glow ($\eta = 1$). (b) As in panel (a), but with gradient glow enabled ($\eta = 0.01$). (c) In addition to receiving a reward $r = 1$ when it has found the goal, the agent is punished with a reward $r = -10$, when it has hit a boundary, without gradient glow ($\eta = 1$). (d) As in panel (c), but with gradient glow enabled ($\eta = 0.7$). (e) As in panel (d), but with gradient glow prolonged further ($\eta = 0.5$). (f) Learning is disabled by always setting the reward to 0. The agent performs a random walk through the grid with average length 54.1 which is included as dashed line in Figs. 10 and 11.

conditions Eq. (2.8) in Ref. [18] to ensure convergence), this is not strictly necessary. In our numerical examples, we have kept α constant for simplicity. Implementation of a dynamic adaptation of the learning rate as was done in Refs. [23] and [22] in the present context is left for future work.] The transitions Figs. 10(a→b) and Fig. 10(c→d) show the effect of enabling gradient glow, i.e., $(\eta = 1) \rightarrow (\eta < 1)$ in (4). Gradient glow provides a mechanism of gradual backpropagation of the policy

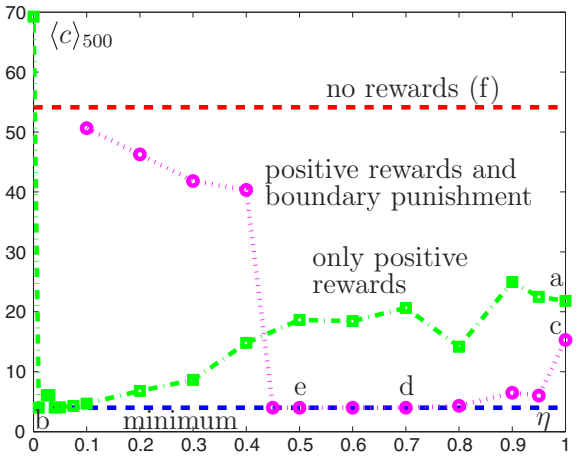

FIG. 11. Episode lengths c averaged over the last 500 episodes in Fig. 10, i.e., episodes 9.5×10^3 to 1×10^4 of single histories of individual agents in the (3×3)-grid world as shown in Fig. 9. The data points distinguish various rewarding strategies and values of gradient glow η as explained in Fig. 10. The upper and lower dashed lines reflect a random walk and the shortest path, respectively, and the letters (a)–(f) correspond to the respective plots in Fig. 10. The optimum value of η depends on the details of the rewarding strategy.

change from the nearest neighbourhood of G to cells more distant from G as the number of episodes increases. The agent settles in the optimal policy in the cases in Figs. 10(b), 10(d) and 10(e).

The policy resulting after 10^4 episodes in case Fig. 10(d) is given in Fig. 9, where the numbers in each cell present the probability to move in the respective direction. While the agent finds the optimal policy for all cells forming the shortest path, it remains ignorant for the remaining cells. As the agent finds and consolidates the shortest path, then episode over episode, it soon visits the off-path cells less frequently, so that the transition probabilities from these cells do not accumulate enough iterations and are "frozen" in suboptimal values. This is characteristic of RL and can also be observed in learning to play games such as backgammon [18], where it is sufficient to play well only in typical rather than all possible constellations of the game. Since for large games, the former often form a small subset of the latter, this can be seen as a strategy to combat with large state spaces (such as number of possible game constellations). To find an optimal policy for *all* cells in Fig. 9, we may start each episode from a random cell, analogous to initializing the agent in an overcomplete basis as explained in Fig. 7. The red numbers in parentheses shown in Fig. 9 present a new policy obtained after 10^5 episodes in this way. In contrast to the old policy, it is optimal or nearly optimal for all cells, with the difference between 1 and the sum of these numbers quantifying the deviation from optimality for each cell ($\leqq 0.02$). Since on average, the agent starts from a given cell only in one-seventh of all episodes, the learning is slowed down, analogous to Fig. 7(a).

Figure 11 summarizes the effect of gradient glow illustrated in Fig. 10 for the two rewarding strategies. To limit the numerical effort, we have averaged the episode lengths over the last 500 episodes in Fig. 10 for individual agents as a "rule of thumb" measure of the agent's performance for the strategy chosen. For a deterministic calculation, we must instead average the length of each episode (and for each η) over a sufficiently large ensemble of independent agents for as many episodes as needed to reach convergence. Despite these shortcomings, the results indicate a qualitatively similar behavior as Fig. 4(a) in Ref. [15]. Figures 10 and 11 demonstrate that gradient glow improves the agent performance, irrespective of whether or not it receives information on false intermediate moves by means of negative rewards, although the latter reduce the required length of glow. It is expected that for an ensemble average, an optimal value of η can be found, with which the fastest convergence to the shortest path can be achieved. Figure 10 distinguishes two qualitatively different modes of convergence. If η is larger than optimal, a gradual improvement is observed, as seen by the damping of spikes in Fig. 10(d). If η is smaller than optimal, then an abrupt collapse to the optimal policy without visible evidence in the preceding statistics that would provide an indication is observed, cf. Fig. 10(e). If η is decreased further, this transition is likely to happen later, to the point it will not be observed within a fixed number of episodes. This results in the steep increase in episode length shown in Fig. 11, which would be absent if the ensemble average was used instead. This sudden transition as shown in Fig. 10(e) can also be observed for individual agents in [15] (not shown there), which applies a softmax-policy function along with edge glow. It is surprising that the quadratic measurement-based policy simulated here exhibits the same phenomenon. Note, however, that convergence does not imply optimality. In tabular RL and PS, such an abrupt transition can be observed if the λ parameter and hence the "correlation length" is too large (in RL) or if the η parameter is too small, so that the glow lasts too long (in PS). The policies obtained in this way are typically suboptimal, especially in larger scale tasks such as bigger grid worlds, for which the agent learns "fast but bad" in this case. It is hence expected that a similar behavior can be observed for our method if we increased the size of the grid.

VI. FINITE DIFFERENCE UPDATES

This work's numerical experiments rely on a symbolic expression (A7) for the gradient ∇_t in (5) for simplicity, which is usually not available in practice, also keeping in mind the variety of compositions, Fig. 2, so that the agent's memory $\hat{U}(\boldsymbol{h})$ is generally unknown. As explained in the discussion of Fig. 1, the agent may then apply a measurement-based internal loop by repeatedly preparing its memory in a state that corresponds to the last percept s_t and register whether or how often the last measurement outcome a_t can be recovered. This approach can be done with either infinitesimal or finite changes in the control vector $\boldsymbol{h}$, where we can distinguish between expectation value- and sample-based updates, depending on how many internal cycles are performed between consecutive external cycles. It should be stressed that the external cycles in Fig. 1 represent the agent-environment interaction, resulting in sequences of state-action pairs and corresponding rewards. While in an elementary optimal control problem, a given objective is to be optimized, here the environment poses at each external cycle a separate and generally unpredictable control problem, all of which must be addressed by the agent simultaneously.

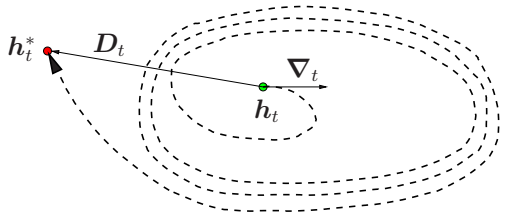

FIG. 12. Following the direction $\nabla p(a_t|s_t)$ of steepest ascent (dashed line) does not necessarily lead to the shortest route $\boldsymbol{D}_t = \boldsymbol{h}_t^* - \boldsymbol{h}_t$ from a given control vector $\boldsymbol{h}_t$ at cycle t to a location $\boldsymbol{h}_t^*$, for which $p(a_t|s_t)$ becomes maximum.

Because of the small learning rate α, the update rule (4) is in all cases local in parameter space, which reflects the assumption that a physical agent cannot completely reconfigure its "hardware" in a single instant. While it is then consistent to apply a gradient $\boldsymbol{D}_t = \nabla_t$ as a local quantity in (4), from a computational perspective, it has a few drawbacks, however. One is that the direction of steepest accent at the current control vector $\boldsymbol{h}_t$ does not need to coincide with the direction $\boldsymbol{D}_t = \boldsymbol{h}_t^* - \boldsymbol{h}_t$ toward the optimum $\boldsymbol{h}_t^*$, as illustrated in Fig. 12.

Another aspect is the vanishing of the gradient. Consider, for example, the initialization of the action system in a mixed state (13) as done in Fig. 4(b). In particular, the graph with $p_{\text{coh}} = 0$ does not display any learning ability. Substituting the corresponding $\hat{\varrho}_{\text{A}} = \hat{I}_{\text{A}}/2$ in (13) and $\hat{U} = \hat{I}$ into (A7), we see that the reason is the vanishing gradient, $\nabla_k = \text{Im}\text{Tr}[\hat{\varrho}_{\text{S}}\hat{\Pi}(a)\hat{H}_k] = 0$. On the other hand, the corresponding setup, Fig. 5(a), reveals that in this case substituting a SWAP gate between S and A for $\hat{U}$ provides an optimal solution (along with an X gate if the meaning of the symbols is reversed) for any $\hat{\varrho}_{\text{A}}$ that is obviously not found in Fig. 4(b). This failure occurs despite the fact that the agents explore, as indicated by the fluctuations in Fig. 4(b). To understand the difference, note that we may generate an ε-greedy policy function by replacing in (6) an (arbitrarily given) state $\hat{\varrho}(s)$ with $\frac{\hat{\varrho}(s)+\varepsilon\hat{I}}{1+\varepsilon d}$, where $0 < \varepsilon \ll 1$ and $d = \text{Tr}\hat{I}$. The term with $\hat{I}$ then gives to (6) a contribution $\sim\text{Tr}_{\text{A}}\hat{\Pi}(a)$, that is independent of s. At the same time, it does not contribute in (A7) to the gradient, $\nabla_k = 0$. If $\boldsymbol{D}_t = \nabla_t = 0$ for all t in (4), the agent's learning comes to rest, however. Finite difference- and sample-based updates here offer a possibility to explore *in parameter space* the neighborhood of the present location $\boldsymbol{h}_t$ (or, colloquially, the "state") of the agent's memory, as a consequence of asymmetries in the control landscape or statistical fluctuations in the samples.

Of particular relevance is a final fixpoint (B11). Intuitively, one would assume that (despite the compactness of the (S)U(n) groups, that is in contrast to the potentially unbounded values of U in RL or h in PS) once an agent has settled in a point (B11), due to the vanishing gradient, it will not be able to react quickly if the environment suddenly changes its allocation of rewards (without confronting the agent with percepts it has not perceived before). However, the learning curves for controllable memories (16 and 32 controls) in Fig. 6(a) demonstrate that relearning after 5×10^3 cycles is not affected. A study of individual agents with 32 controls in Fig. 6(a) reveals that the Euclidean length of the numerical gradient rises from 10^{-14} at cycle 5000 to a value >1 in only 15 cycles. Better understanding of this is left for future study. In what follows, we outline the mentioned alternatives in some more detail.

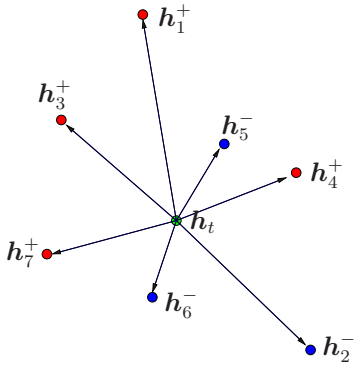

FIG. 13. Internally generated random cloud of sample controls $\boldsymbol{h}_k$ around a given control vector $\boldsymbol{h}_t$ at cycle t for which binary measurements "given s_t, detect a_t, or not" are carried out between external cycles, yielding positive ($\boldsymbol{h}_k^+$) or negative ($\boldsymbol{h}_k^-$) outcomes.

A. Expectation value-based updates

If the time consumed by the internal cycles is uncritical with respect to the external cycles, the agent can obtain estimates of $p(a_t|s_t)$ from a sufficiently large number of internal binary measurements. With these, it can either approximate the components $\nabla_k p(a_t|s_t;h_j) \approx [p(a_t|s_t;h_j+\delta_{jk}\delta h_k) - p(a_t|s_t;h_j)]/\delta h_k$ of the local gradient $\nabla_t = \nabla p(a_t|s_t;\boldsymbol{h}_t)$, which is then substituted as $\boldsymbol{D}_t = \nabla_t$ into (4). Alternatively, it can perform a global search for the location $\boldsymbol{h}_t^*$ of the maximum of $p(a_t|s_t)$. A possible algorithm for the latter is differential evolution, which relies on deterministic values $p(a_t|s_t;\boldsymbol{h})$ rather than noisy samples. Once an estimate for $\boldsymbol{h}_t^*$ has been found, the difference $\boldsymbol{D}_t = \boldsymbol{h}_t^* - \boldsymbol{h}_t$ is used in (4).

B. Sample-based updates

Reliance on expectation values may give away potential speed gains offered by a quantum memory, which poses the question of whether a finite number of sample measurements is sufficient. Since individual updates in (4) are made with a small fraction α of the whole $\boldsymbol{D}_t$, the assumption is that the individual statistical errors in the sampled $\boldsymbol{D}_t$ cancel out in the long run.

As for the expectation value-based updates discussed above, samples can be used to either create *discrete* estimates $\nabla_k p(a_t|s_t;h_j) \approx [s(h_j+\delta_{jk}\delta h_k) - s(h_j)]/2$ for the components k of the local gradient $\nabla_t = \nabla p(a_t|s_t;\boldsymbol{h}_t)$, where $s = s(\boldsymbol{h}_t) = \pm 1$ depending on whether the outcome of the binary measurement $(a_t|s_t;\boldsymbol{h}_t)$ is positive or not. Alternatively, for finite difference updates, one may consider a neural gas [28] inspired approach depicted in Fig. 13. In this approach, the differences

$$\boldsymbol{D}_t^{(n)} = \frac{1}{n}\sum_{k=1}^{n} s_k \boldsymbol{h}_k = \frac{n-1}{n}\boldsymbol{D}_t^{(n-1)} + \frac{1}{n}s_n \boldsymbol{h}_n \qquad (18)$$

between the sampled centers of positive [$s_k = s(\boldsymbol{h}_k) = +1$, i.e., $\boldsymbol{h}_k = \boldsymbol{h}_k^+$] and negative outcomes ($s_k = s(\boldsymbol{h}_k) = -1$, i.e., $\boldsymbol{h}_k = \boldsymbol{h}_k^-$) of the binary measurements $(a_t|s_t;\boldsymbol{h}_k)$ are then applied in (4). Although one could store a current estimate $\boldsymbol{D}_t^{(n)}$ for each observed state-action pair $(a_t|s_t)$ and merely update it according to (18) with each new measurement point $s_{n+1}\boldsymbol{h}_{n+1}$, this would give away the generalization capability described in Sec. V A 3. One would hence need to perform n internal cycles with the POVM-based internal loop between each external cycle. The $\boldsymbol{h}_k$ could be drawn, e.g., from a Gaussian centered around the respective $\boldsymbol{h}_t$. The variance of this Gaussian could be gradually decreased with the number of external cycles to increase the locality (local resolution) of the cloud.

Figure 12 gives the misleading impression that finite difference updates are superior to gradient-based methods. To give an illustrative counterexample, one could think of a two-dimensional $\boldsymbol{h} = (h_x,h_y)$ and a control landscape $p(\boldsymbol{h})$ modeled by the monotonically increasing height $p(l)$ along the length l of a tape of paper bent into a spiral and placed onto the dashed line in Fig. 12, such that one end with $p(0) = 0$ is located at $\boldsymbol{h}_t$ and the other one at $\boldsymbol{h}_t^*$. Here, a gradient-based method would safely follow the long path on the tape's upper edge, whereas a finite difference method would trade a potential speedup with the risk of missing the paper at all trials. Since a comparison of state-of-the-art optimal control methods based on noisy samples for the agent's learning would go beyond the scope of this work, we here restrict ourselves to these sketchy lines of thought, whose numerical study is pending, and leave open the question of what the best method is for a given task.

A characteristic shared by the loops of Fig. 1 and optimal control setups is the need of an experimental "mastermind" who controls the controls. An agent which is supposed to act autonomously would be required to accomplish this by itself, ideally in a "natural" or "organic computing" sense. An elementary example from everyday life are "desire paths" which form or dissipate, depending on their usage and without a designated planner.

VII. SUMMARY AND OUTLOOK

In summary, we have considered a basic PS-inspired quantum agent scheme and analyzed its performance for the invasion game and grid world tasks analogous to Refs. [7,8,15]. The numerical results show that similar behavior can be observed for the quantum agent, as long as the memory is not underactuated. This is not obvious because of the fundamental difference in the number of free parameters. If N_S and N_A denote the number of possible percepts and actions, respectively, then in classical tabular action value RL methods, the estimated values of all percept-action pairs are combined to a $(N_S \times N_A)$ matrix, i.e., we have $(N_S N_A)$ real parameters. If we encoded in our scheme each percept and action category by a separate subsystem, whose dimensionalities correspond to the number of values, the respective category can adopt, then $\hat{U}$ is an at least $U(N = N_S N_A)$ matrix for which we are faced with $(N_S N_A)^2$ real parameters.

Our agent inherited from PS the absence of a value function as expectation of a discounted sum of future rewards, which distinguishes it from standard RL algorithms. Moreover, our examples may have also allowed us to omit the learning rate α as in PS, which here merely served as a constant reward rescaling. In value-function-based RL algorithms with deterministic environments, a constant $\alpha = 1$ is optimal, whereas α is gradually decreased to 0 to ensure convergence in stochastic (but on average time-independent) environments. In contrast, our examples also include deterministic time-dependent environments whose rewarding changes abruptly.

For its decision making, the agent can query two oracles: its internal memory and the external environment. One important purpose of the internal "thinking" is to reduce the number of external queries, since in practice, the (potentially hazardous) environment is the more expensive oracle to query. On the other hand, the time associated with internal queries may be detrimental when the environment changes over time. References [14,29], for example, implement an adaptation of Grover search [30] by applying a Szegedy-type quantum walk [31,32] on a Hilbert space $\mathcal{H}_{\mathrm{SA}} \otimes \mathcal{H}_{\mathrm{SA}}$, where $\mathcal{H}_{\mathrm{SA}}$ is spanned by the clip basis states $|c_j\rangle$. Internal queries have been formalized in [14,29] as mixing of a classical Markov chain (which in turn formalizes an elementary ECM query). The quantum walk implementation of the mixing allows a quadratic reduction in the number of elementary ECM queries required for this mixing. The resulting speedup in internal "thinking steps" thus refers to composite random walks in the ECM (such as reflection [7]) that are absent in basic PS, where each walk between successive environmental interactions reduces to traversing one single ECM edge. In its basic form, our scheme also applies just one single query of the memory Fig. 2 (although Fig. 1 allows for optional internal feedback loops in principle). In addition, our approach follows the tradition of gradient-based methods and must be distinguished from analytically defined quantum algorithms such as Grover search.

While each individual memory query in our scheme resembles circuit-based quantum computing, the quantum walk sped-up Markov chain mixing in Ref. [14,29] appears in a wider sense to be more closely related to quantum annealing, which has recently attracted interest for RL [33,34]. This in turn raises the question of the relation between the nature of typical optimization problems and those of agents coping with environments. In the latter case, for example, the learning of a good policy only for percepts which are "typical" and have thus been encountered sufficiently often in the past, shares features with "soft computing," where it is sufficient to find a good rather than an exact solution, where the latter would here consist in a policy that is optimal for all possible percepts. One may think of, e.g., simplifying a symbolic mathematical expression: While all transformation steps themselves must be exact, there are no strict rules, as far as the best way of its formulation is concerned.

Apart from a potential speedup in decision making, the nonorthogonality of quantum states could form a natural basis for (a) implementing generalization during state initialization, as we have shown in the never-ending-color scenario in Sec. V A 3, and (b) balancing exploration and exploitation during state measurement. An open problem is how this can be scaled up to allow tasks such as dimensionality reduction or feature learning, which are relevant in "big data" machine learning. Finally, an interesting direction for future work is

to incorporate a framework of quantum walks and networks [31,32,35–37] with the goal to find physically feasible alternatives to a computational determination of the gradients.

ACKNOWLEDGMENTS

This work was supported in part by the Austrian Science Fund (FWF) through Grant No. SFB FoQuS F4012, and by the Templeton World Charity Foundation (TWCF) through Grant No. TWCF0078/AB46.

APPENDIX A: UPDATING THE MEMORY $\hat{U}$

In Appendix A, we focus on a model-based (i.e., symbolic) determination of the gradient $\nabla\hat{U}(\boldsymbol{h})$. The exact form of the gradient depends on the parametrization. For example, if $\hat{U}(\boldsymbol{h}) = e^{-i\hat{H}(\boldsymbol{h})}$ is given by some Hermitian $\hat{H}$, then

$$\nabla\hat{U} = \int_0^1 e^{-ix\hat{H}}(-i\nabla\hat{H})e^{-i(1-x)\hat{H}}dx. \tag{A1}$$

For small $\hat{H}$, we can expand the exponentials in $\hat{H}$ to lowest order, and the approximation $\nabla\hat{U} \approx -i\nabla\hat{H}$ holds.

In the case of a continuous time dependence, the vector $\boldsymbol{h}$ can be replaced by a function $h(t)$, with which a unitary propagator from time t_1 to time t_3 is given as a positively time-ordered integral

$$\hat{U}(t_3,t_1) = \mathcal{T}e^{-i\int_{t_1}^{t_3} dt_2 h(t_2)\hat{H}(t_2)} \tag{A2}$$

and

$$\frac{\delta\hat{U}(t,0)}{\delta h}(t_1) = -i\hat{U}(t,t_1)\hat{H}(t_1)\hat{U}(t_1,0). \tag{A3}$$

If $\hat{H}(t) = \sum_k \tilde{h}_k(t)\hat{H}_k$ is expanded in terms of some fixed Hamiltonians $\hat{H}_k$, then with $h_k = h\tilde{h}_k$, (A2) becomes $\hat{U}(t_3,t_1) = \mathcal{T}e^{-i\int_{t_1}^{t_3} dt_2 \sum_k h_k(t_2)\hat{H}_k}$, and (A3) is replaced with $\frac{\delta\hat{U}(t,0)}{\delta h_k}(t_1) = -i\hat{U}(t,t_1)\hat{H}_k\hat{U}(t_1,0)$. Navigation on unitary groups becomes discretized if only a restricted (finite) set of Hamiltonians $\hat{H}_k$ can be implemented at a time rather than an analytically time-dependent Hamiltonian $\hat{H}(t)$, so that only one of the $\tilde{h}_k$ is nonzero for a given t. A known example is the alternating application of two fixed Hamiltonians [26], $\hat{H}_{2k} = \hat{H}^{(2)}$ and $\hat{H}_{2k+1} = \hat{H}^{(1)}$ $(k = 0,1,2,\ldots,\lfloor\frac{n}{2}\rfloor)$, for a set of times to be determined from the target unitary [38]. In this discrete case as defined by a piecewise constant normalized $\hat{H}$ in (A2), the function $h(t)$ can be replaced with a vector $\boldsymbol{h}$, and the functional derivatives with respect to $h(t)$ reduce to gradients with respect to $\boldsymbol{h}$.

1. Adding layer after layer

We can update the present unitary $\hat{U}$ by multiplying it from the left with a new layer $\hat{U}(\delta\boldsymbol{h})$ after each cycle,

$$\hat{U} \leftarrow \hat{U}(\delta\boldsymbol{h})\hat{U}. \tag{A4}$$

If $\hat{U}(\delta\boldsymbol{h}) = \mathrm{e}^{-\mathrm{i}\sum_k \delta h_k\hat{H}_k}$ is a small modification close to the identity, then the mentioned approximation of (A1) gives $(\nabla\hat{U})_k \approx -i\hat{H}_k\hat{U}$. This is of advantage if the agent or its simulation is able to store only the present $\hat{U}$ and not the history of updates (layers). The components of (5) then become

$$\frac{\partial p(a|s)}{\partial h_k} = 2\mathrm{Im}\langle\hat{U}^\dagger\hat{\Pi}(a)\hat{H}_k\hat{U}\rangle. \tag{A5}$$

2. Fixed number of layers

In our numerical examples, we consider a discretized memory model, for which (A2) reduces to a product of n unitaries

$$\hat{U} = \hat{U}_n\ldots\hat{U}_2\hat{U}_1, \quad \hat{U}_k = e^{-ih_k\hat{H}_k}, \tag{A6}$$

which simplifies the determination of the gradient, since $(\nabla\hat{U})_k = -i\hat{U}\hat{H}_k(t_k)$, where $\hat{H}_k(t_k) = (\hat{U}_k\cdots\hat{U}_1)^\dagger\hat{H}_k(\hat{U}_k\cdots\hat{U}_1)$, so that the components of (5) now become

$$\frac{\partial p(a|s)}{\partial h_k} = 2\mathrm{Im}\langle\hat{U}^\dagger\hat{\Pi}(a)\hat{U}\hat{H}_k(t_k)\rangle. \tag{A7}$$

In this work, we use alternating layers defined by two fixed Hamiltonians $\hat{H}^{(1)}$ and $\hat{H}^{(2)}$ as mentioned at the beginning of this section.

APPENDIX B: NUMERICAL AND MEASUREMENT-BASED OBJECTIVES

1. Distance and uniformly averaged fidelity

Consider an n-level system and two unitary operators $\hat{U}$ and $\hat{U}_\mathrm{T}$, where $\hat{U} = \hat{U}(h)$ depends on an (unrestricted) external control field $h(t)$, and $\hat{U}_\mathrm{T}$ is a desired target. Their (squared) Euclidean distance as induced by the Hilbert-Schmidt dot product is given by

$$\begin{aligned} D &\equiv \|\hat{U} - \hat{U}_\mathrm{T}\|^2 = \mathrm{Tr}[(\hat{U} - \hat{U}_\mathrm{T})^\dagger(\hat{U} - \hat{U}_\mathrm{T})] \\ &= 2n - 2\mathrm{ReTr}(\hat{U}_\mathrm{T}^\dagger\hat{U}) \in [0,4n]. \end{aligned} \tag{B1}$$

If $\hat{U}(h)$ is controllable in the sense that at least one $h(t)$ exists such that $\hat{U}(h) = \hat{U}_\mathrm{T}$, then (B1) has the set $\{0,4,\ldots,4n\}$ as possible extremal values (i.e., $\frac{\delta D}{\delta h} = 0$), where the values 0 and $4n$ are attained for $\hat{U} = \pm\hat{U}_\mathrm{T}$, while the remaining extrema are saddle points [39]. A measure insensitive to global phases $\hat{U} = \mathrm{e}^{\mathrm{i}\varphi}\hat{U}_\mathrm{T}$ is the average fidelity defined by

$$F \equiv \overline{|\langle\Psi|\hat{U}_\mathrm{T}^\dagger\hat{U}|\Psi\rangle|^2} = \frac{n + |\mathrm{Tr}(\hat{U}_\mathrm{T}^\dagger\hat{U})|^2}{n(n+1)}, \tag{B2}$$

where the overline denotes uniform average over all $|\Psi\rangle$ [40,41]. Note that $F \in [(n+1)^{-1},1]$ for $n > 1$, and $F = 1$ for $n = 1$. Both (B1) and (B2) are determined by the complex

$$\cos\sphericalangle(\hat{U},\hat{U}_\mathrm{T}) \equiv \frac{\mathrm{Tr}(\hat{U}_\mathrm{T}^\dagger\hat{U})}{\sqrt{\mathrm{Tr}(\hat{U}^\dagger\hat{U})}\sqrt{\mathrm{Tr}(\hat{U}_\mathrm{T}^\dagger\hat{U}_\mathrm{T})}} = \frac{\mathrm{Tr}(\hat{U}_\mathrm{T}^\dagger\hat{U})}{n}, \tag{B3}$$

which is confined to the complex unit circle and whose expectation $\langle|\cos\sphericalangle(\hat{U},\hat{U}_\mathrm{T})|^2\rangle = n^{-1}$ for uniform random $\hat{U}$ drops to zero with growing n.

If in (B2), the averaging is restricted to a d-dimensional subspace P, we must replace (B2) with

$$F_\mathrm{P} \equiv \overline{|\langle\Psi|\hat{M}|\Psi\rangle|^2}^{(\mathrm{P})} = \frac{\mathrm{Tr}(\hat{M}^\dagger\hat{M}) + |\mathrm{Tr}(\hat{M})|^2}{d(d+1)}, \tag{B4}$$

where $\hat{M} = \hat{P}\hat{U}_{\mathrm{T}}^{\dagger}\hat{U}\hat{P}$, with $\hat{P}$ being the projector onto P. Note that $F_{\mathrm{P}} \in [\frac{\max(0,2d-n)}{d(d+1)},1]$ for $n > 1$ [since $\mathrm{Tr}(\hat{M}^{\dagger}\hat{M}) = \mathrm{Tr}(\hat{P}_{\mathrm{T}}\hat{P}_{\mathrm{U}}) \in \max(0,2d-n)$ with $\hat{P}_{\mathrm{T}} = \hat{U}_{\mathrm{T}}\hat{P}\hat{U}_{\mathrm{T}}^{\dagger}$, $\hat{P}_{\mathrm{U}} = \hat{U}\hat{P}\hat{U}^{\dagger}$], and $F_{\mathrm{P}} = 1$ for $n = 1$. While for a one-dimensional $\hat{P} = |\Psi\rangle\langle\Psi|$, (B4) reduces to $F_{\Psi} = |\langle\Psi|\hat{U}_{\mathrm{T}}^{\dagger}\hat{U}|\Psi\rangle|^2$, the other limit $d = n$ recovers (B2).

If in (B4), $\hat{U} = \hat{U}_{\mathrm{SB}}$ couples the quantum system S to a bath B, then we define a projector $\hat{\Pi} = \hat{U}_{\mathrm{T}}\hat{P}|\Psi\rangle\langle\Psi|\hat{P}\hat{U}_{\mathrm{T}}^{\dagger} \otimes \hat{I}_{\mathrm{B}}$ and generalize (B4) to

$$F_{\mathrm{P}} \equiv \overline{\mathrm{Tr}_{\mathrm{SB}}[\hat{U}\hat{P}|\Psi\rangle\hat{\varrho}_{\mathrm{B}}\langle\Psi|\hat{P}\hat{U}^{\dagger}\hat{\Pi}]}^{(\mathrm{P})} \tag{B5}$$

$$= \left\langle \frac{\mathrm{Tr}_{\mathrm{S}}(\hat{M}^{\dagger}\hat{M}) + (\mathrm{Tr}_{\mathrm{S}}\hat{M})^{\dagger}(\mathrm{Tr}_{\mathrm{S}}\hat{M})}{d(d+1)} \right\rangle_{\mathrm{B}}, \tag{B6}$$

where $\langle\cdots\rangle_{\mathrm{B}} \equiv \mathrm{Tr}_{\mathrm{B}}[\hat{\varrho}_{\mathrm{B}}(\cdots)]$ with a fixed bath state $\hat{\varrho}_{\mathrm{B}}$.

Replacing $\hat{U}|\Psi\rangle\langle\Psi|\hat{U}^{\dagger}$ in (B2) with the output $\mathcal{M}(|\Psi\rangle\langle\Psi|)$ of a quantum channel generalizes (B2) to [40]

$$F = \overline{\langle\Psi|\hat{U}_{\mathrm{T}}^{\dagger}\mathcal{M}(|\Psi\rangle\langle\Psi|)\hat{U}_{\mathrm{T}}|\Psi\rangle} = \frac{n + \sum_k |\mathrm{Tr}(\hat{U}_{\mathrm{T}}^{\dagger}\hat{G}_k)|^2}{n(n+1)}, \tag{B7}$$

where $\hat{G}_k$ are the Kraus operators of the decomposition of the channel map $\mathcal{M}(\hat{\varrho}) = \sum_k \hat{G}_k\hat{\varrho}\hat{G}_k^{\dagger}$. Note that a change $\hat{G}'_k = \sum_j V_{kj}\hat{G}_j$ of the Kraus operators as described by a unitary matrix V leaves (B7) invariant.

2. Percept statistics-based fidelity

The uniform average in (B7) can be generalized to an arbitrary distribution $p(|\Psi_k\rangle)$ of possible input states $|\Psi_k\rangle$,

$$\hat{\varrho}_{\mathrm{in}} = \sum_k p(|\Psi_k\rangle)|\Psi_k\rangle\langle\Psi_k|, \tag{B8}$$

that reflects the statistics of their occurrence in different instances of applications of the device (external cycles in a control loop). This generalizes (B7) to

$$F_{\hat{\varrho}_{\mathrm{in}}} \equiv p(\hat{U}_{\mathrm{T}}) = \sum_k p(\hat{U}_{\mathrm{T}}||\Psi_k\rangle)p(|\Psi_k\rangle), \tag{B9}$$

$$p(\hat{U}_{\mathrm{T}}||\Psi_k\rangle) = \langle\Psi_k|\hat{U}_{\mathrm{T}}^{\dagger}\mathcal{M}(|\Psi_k\rangle\langle\Psi_k|)\hat{U}_{\mathrm{T}}|\Psi_k\rangle, \tag{B10}$$

which is just the total probability $p(\hat{U}_{\mathrm{T}})$ of correctly detecting a $\hat{U}_{\mathrm{T}}$-transformed pure (but unknown) input state drawn from a distribution (B8). Once $p(\hat{U}_{\mathrm{T}}) = 1$, the channel's effect is indistinguishable from that of $\hat{U}_{\mathrm{T}}$ for the set of possible inputs $|\Psi_k\rangle$, (i.e., those for which $p(|\Psi_k\rangle) > 0$). The case $p(\hat{U}_{\mathrm{T}}) \lessapprox 1$ is relevant from a numerical and experimental point of view. Rare inputs $|\Psi_k\rangle$, for which $0 < p(|\Psi_k\rangle) \ll 1$, will hardly affect $p(\hat{U}_{\mathrm{T}})$ in a control loop, which relaxes the demands on the channel compared to the uniform average (B7). The channel optimization itself is thus economized in the sense that it is required to perform well only on *typical* rather than *all* inputs.

Here, we consider a navigation of $\hat{U}$ in the above-mentioned discretized case, $\hat{U} = \hat{U}(\boldsymbol{h})$, that starts at the identity $\hat{U}(\boldsymbol{h} = \boldsymbol{0}) = \hat{I}$ and from there undertakes a gradient-based maximization of $p(\hat{U}_{\mathrm{T}})$ as defined in (B9) to a point where

$$\nabla p(\hat{U}_{\mathrm{T}}) = 0. \tag{B11}$$

The control vector $\boldsymbol{h}$ typically represents system variables and not those of the bath. Rather than solving for the parameters [38] for which the scheme [26] yields a desired given unitary, we search parameters, for which the unitary, whose specific form we are not interested in, solves a given task.

3. Optimal memory navigation and constrained optimization

While here we have discussed and compared concrete types of algorithms, a more fundamental question concerns the optimality and derivation of general (e.g., speed) limits of the learning process. Although the physical time is given as the sum of the agent and environment response times over each cycle, one may restrict to counting the number of (a) memory cycles in total, (b) external cycles only (c) episodes or (d) parameter updates $\delta\boldsymbol{h}$ of the memory $\hat{U}(\boldsymbol{h})$, depending on what the most critical criterion is. Not only should the navigation of $\hat{U}$ from the identity to a point (B11) follow an optimal trajectory, but also the navigation of the percept states by a given $\hat{U}$ should be such that the individual physical memory evolution times become minimum. Such demands may conflict with restrictions on the practical implementability and complexity of the memory. Since these questions are beyond the scope of this work, in what follows we restrict ourselves to outline a connection to constrained optimization as a possible formal approach.

Assuming the Schrödinger equation $\frac{d}{dt}\hat{U} = -i\hat{H}\hat{U}$ and a fixed energy-type constraint $\|\frac{d}{dt}\hat{U}\|^2 = \mathrm{Tr}(\hat{H}^2) \stackrel{!}{=} E^2$, the length of a curve in the space of unitaries becomes

$$L = \int_0^T \left\| \frac{d\hat{U}}{dt} \right\| dt \stackrel{!}{=} ET, \tag{B12}$$

where T is the arrival (or protocol) time, $\hat{U}(t = T) = \hat{U}_{\mathrm{T}}$ [42,43]. (A "protocol" refers to a prescription for the time dependence of h, $\hat{H}$, or $\hat{U}$.) In addition to (or instead of) the protocol time T, we may also consider

$$C \equiv \int_0^T \left\| \frac{d\hat{H}}{dt} \right\| dt \tag{B13}$$

as a measure of the complexity of the protocol that integrates the changes that have to be made on $\hat{H}$ via the control fields.

If the optimization problem comprises two objectives such as minimizing a distance D or maximizing a fidelity F with minimum amount of (B12) or (B13), then an *approximate* approach consists in first finding an $h(t)$ that optimizes an objective function J_1 under a fixed constraint J_2. Here, J_1 represents D or F, while J_2 may represent L or C. This can be formulated as an Euler-Lagrange equation

$$\frac{\delta J_1}{\delta h} - \lambda\frac{\delta J_2}{\delta h} = 0, \tag{B14}$$

where the Lagrange multiplier λ must finally be substituted with the given constant such as L or C. This optimization is then repeated with stepwise decreased L or C, until the deterioration of the achievable J_1 exceeds a certain threshold. Equivalently, (B14) may also be thought of optimizing J_2 under the constraint of constant J_1. Equation (B14), which contains

both derivatives in a symmetric way, merely states the linear dependence of the functional derivatives at an extremal point h in the function space $\{h\}$.

In the discrete case, the time integrals in Eqs. (B12) and (B13) reduce to sums over time intervals with constant $\hat{H}$, and in (B13) we assume that each jump of $\hat{H}$ gives a fixed finite contribution. To gain some intuition into the meaning of C, we may think of navigating through a classical rectangular grid. There is a set of shortest paths connecting the diagonal corners, but they are not equivalent with respect to the number of turns the navigator has to make along its way. In the quantum context, the number of switches equals the number of intervals with constant $\hat{H}$, which may be thought of as "gates." In contrast to an analytical design of quantum circuits, the circuit is here generated numerically, however. Since each switch to a different $\hat{H}$ changes the instantaneous eigenbasis, we may thus think rather of "layers," drawing an analogy to classical artificial neural networks [44].

[1] Nature (London) **521**, 435 (2015), special issue on machine intelligence.

[2] Science **349**, 248 (2015), special issue on artificial intelligence.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2000).

[4] P. Wittek, *Quantum Machine Learning* (Elsevier, Amsterdam, 2014).

[5] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemp. Phys. **56**, 172 (2015).

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).

[7] H. J. Briegel and G. D. las Cuevas, Sci. Rep. **2**, 400 (2012).

[8] J. Mautner, A. Makmal, D. Manzano, M. Tiersch, and H. J. Briegel, New Gener. Comput. **33**, 69 (2015).

[9] D. D'Alessandro, *Introduction to Quantum Control and Dynamics* (Chapman & Hall/CRC, Boca Raton, FL, 2008).

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

[11] D. Dong and I. R. Petersen, IET Control Theory Appl. **4**, 2651 (2010).

[12] S. J. Glaser, U. Boscain, T. Calarco, C. P. Koch, W. Köckenberger, R. Kosloff, I. Kuprov, B. Luy, S. Schirmer, T. Schulte-Herbrüggen, D. Sugny, and F. K. Wilhelm, Eur. Phys. J. D **69**, 279 (2015).

[13] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems* (Oxford University Press, Oxford, UK, 2002).

[14] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, Phys. Rev. X **4**, 031002 (2014).

[15] A. A. Melnikov, A. Makmal, and H. J. Briegel, arXiv:1405.5459.

[16] A. A. Melnikov, A. Makmal, V. Dunjko, and H. J. Briegel, Sci. Rep. **7**, 14430 (2017).

[17] Ø. F. Bjerland, Projective simulation compared to reinforcement learning, Master's thesis, University of Bergen, Norway, 2015 (unpublished).

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).

[19] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, Englewood Cliffs, NJ, 2009).

[20] D. Dong, C. Chen, H. Li, and T.-J. Tarn, IEEE Trans. Syst. Man, Cybern. B, Cybern **38**, 1207 (2008).

[21] J. Bang, J. Ryu, S. Yoo, M. Pawlowski, and J. Lee, New J. Phys. **16**, 073017 (2014).

[22] J. Clausen, arXiv:1507.08990.

[23] J. Clausen, G. Bensky, and G. Kurizki, Phys. Rev. Lett. **104**, 040401 (2010).

[24] H. van Seijen, A. R. Mahmood, P. M. Pilarski, M. C. Machado, and R. S. Sutton, J. Mach. Learn. Res. **17**, 5057 (2016).

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature (London) **323**, 533 (1986).

[26] S. Lloyd, Phys. Rev. Lett. **75**, 346 (1995).

[27] M. H. Goerz, K. B. Whaley, and C. P. Koch, EPJ Quantum Technol. **2**, 21 (2015).

[28] T. M. Martinetz and K. J. Schulten, in *Proceedings of the 1991 International Conference on Artificial Neural Networks (Icann-91), Espoo, Finland, 24–28 June, 1991*, edited by K. Mäkisara, O. Simula, J. Kangas, and T. Kohonen (Elsevier, North-Holland, Amsterdam, 1991), pp. 397–402; B. Fritzke, in *Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, Colorado, USA, 1994*, edited by G. Tesauro, D. S. Touretzky, and T. K. Leen (MIT Press, Cambridge, MA, 1995), pp. 625–632.

[29] V. Dunjko, N. Friis, and H. J. Briegel, New J. Phys. **17**, 023006 (2015).

[30] L. K. Grover, in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, SOTC '96, Philadelphia, Pennsylvania, USA, May 22–24, 1996* (ACM, New York, 1996), pp. 212–219.

[31] M. Szegedy, in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, (FOCS 2004), 17–19 October 2004, Rome, Italy* (IEEE, Piscataway, NJ, 2004), pp. 32–41.

[32] F. Magniez, A. Nayak, J. Roland, and M. Santha, SIAM J. Comput. **40**, 142 (2011).

[33] D. Crawford, A. Levit, N. Ghadermarzy, J. S. Oberoi, and P. Ronagh, arXiv:1612.05695.

[34] A. Levit, D. Crawford, N. Ghadermarzy, J. S. Oberoi, E. Zahedinejad, and P. Ronagh, arXiv:1706.00074.

[35] M. Schuld, I. Sinayskiy, and F. Petruccione, Quant. Info. Proc. **13**, 2567 (2014).

[36] M. Schuld, I. Sinayskiy, and F. Petruccione, Phys. Rev. A **89**, 032333 (2014).

[37] J. Biamonte, M. Faccin, and M. D. Domenico, arXiv:1702.08459.

[38] G. Harel and V. M. Akulin, Phys. Rev. Lett. **82**, 1 (1999).

[39] H. Rabitz, M. Hsieh, and C. Rosenthal, Phys. Rev. A **72**, 052337 (2005).

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

[41] C. Dankert, Efficient simulation of random quantum states and operators, Master's thesis, University of Waterloo, Ontario, Canada, 2005 (unpublished).

[42] B. Russell and S. Stepney, Phys. Rev. A **90**, 012303 (2014).

[43] X. Wang, M. Allegra, K. Jacobs, S. Lloyd, C. Lupo, and M. Mohseni, Phys. Rev. Lett. **114**, 170501 (2015).

[44] R. Rojas, *Neural Networks: A Systematic Introduction* (Springer-Verlag, Berlin, 1996).