Quantum Information Processing (2020) 19:8
https://doi.org/10.1007/s11128-019-2470-8

Quantum Lyapunov control with machine learning

S. C. Hou[1] · **X. X. Yi**[1]

Received: 31 January 2019 / Accepted: 9 October 2019 / Published online: 26 November 2019

Abstract
Quantum state engineering is a central task in Lyapunov-based quantum control. Given different initial states, better performance may be achieved if the control parameters, such as the Lyapunov function, are individually optimized for each initial state, however, at the expense of computing resources. To tackle this issue, we propose an initial-state-adaptive Lyapunov control strategy with machine learning. Specifically, artificial neural networks are used to learn the relationship between the optimal control parameters and initial states through supervised learning with samples. Two designs are presented where the feedforward neural network and the general regression neural network are used to select control schemes and design Lyapunov functions, respectively. We demonstrate the performance of the designs with a three-level quantum system for an eigenstate control problem. Since the sample generation and the training of neural networks are carried out in advance, the initial-state-adaptive Lyapunov control can be implemented for new initial states without much increase of computational resources.

Keywords Lyapunov control · Machine learning · Neural network · Quantum state preparation

1 Introduction

Quantum control [1–6] plays a fundamental role in modern quantum technologies such as quantum computation, quantum communication and quantum metrology. A central goal in quantum control is designing time-varying external control fields to effectively engineer quantum states and operators. More than a decade ago, the Lyapunov-based method was developed for the control of quantum systems [7,8]. In quantum Lyapunov

✉ X. X. Yi
yixx@nenu.edu.cn

S. C. Hou
housc495@nenu.edu.cn

[1] Center for Quantum Sciences and School of Physics, Northeast Normal University, Changchun 130024, China

control, the control fields are obtained by simulating the dynamics only once without iterations as those used in quantum optimal control algorithms [5]. The method has the merits of simplicity in generating control fields and flexibility in designing the control field shapes. In recent years, numerous efforts have been devoted to investigate or improve the convergence of Lyapunov control for different quantum control models [9–16]. Meanwhile, Lyapunov control method is successfully employed for diverse quantum information processing tasks [16–31]. For example, it is recently used to realize topological modes [19], quantum synchronization [17] and speed up adiabatic passage [30].

Previous research usually considers an initial-state-independent design of quantum Lyapunov control. When dealing with different initial states, better performance (e.g., control time or fidelity) may be achieved if the control parameters (such as those in the Lyapunov function) are individually optimized for each initial state. However, it is usually hard to find an explicit (analytic) relationship between the optimal control parameters and the initial states since the control fields are generated numerically. On the other hand, numerical optimizing the Lyapunov control parameters typically requires simulating the dynamics many times, which requires more computing resources such as the simulation time. Thus an initial-state-adaptive quantum Lyapunov control without a significant increase in computing resources is desirable.

Machine learning [32,33] is a powerful tool to improve a performance criterion from experience or data, which has been extensively applied in internet technology, artificial intelligence, finance, medical diagnosis and so on. Recently, machine learning technology has been successfully employed to advance quantum physics problems [34–44], such as quantum many-body problems [38,39], quantum state identification [39,40] and quantum control [41–44]. Motivated by its ability and versatility, we intend to use machine learning techniques, specifically, (artificial) neural networks [33] to design an initial-state-adaptive quantum Lyapunov control. The basic idea is as follows. First, numerically generate a certain number of samples that encode different initial states and their corresponding optimal parameters. Then, train a neural network with these samples through supervised learning until its performance is satisfactory. At last, apply the trained neural network to predict control parameters for new initial states. Two designs are proposed to select control schemes and design Lyapunov functions. The initial-state-adaptive designs would be helpful when the number of initial states is large or fast calculation of control fields is needed, such as in quantum feedback control scenarios [3,22–25]. We investigate an initial-state-adaptive eigenstate control problem in a three-level atomic system to demonstrate the usage and performance of the designs. The task is to drive different initial states to an eigenstate of the free Hamiltonian with high fidelity at a particular time. In the first example, the feedforward neural network is used to select the best control Hamiltonian from several candidates. In the second example, the general regression neural network is used to predict the Lyapunov function.

The remainder of the paper is organized as follows. Section 2 reviews the Lyapunov control method for the eigenstate preparation problem. In Sect. 3, we introduce the feedforward neural network (multilayer perceptron) and the general regress neural network which are used as the tools for classification and regression in this paper, respectively. Two initial-state-adaptive designs are proposed in Sect. 4 and then illus-

trated with a three-level quantum system in Sect. 5. Finally, the results are summarized and discussed in Sect. 6.

2 Quantum Lyapunov control

Lyapunov control is a useful technique for quantum control tasks, typically eigenstate control [24,26–31]. The control fields of a quantum Lyapunov control are designed by a feedback law [7,8] where the control fields are functions of the quantum states. Since measurement destroys quantum states, the control law is not directly used in experiments. Instead, the control law is applied in computer simulations to generate fields. Then the control fields can be applied in experiments without measurements.

We introduce the mathematical formula of quantum Lyapunov control with an n-dimensional closed quantum system described by the Schrödinger equation ($\hbar = 1$ is assumed)

$$\frac{\mathrm{d}}{\mathrm{d}t}|\Psi\rangle = -i\left[H_0 + \sum_{k=1}^{m} f_k(t)H_k\right]|\Psi\rangle. \tag{1}$$

Here H_0 is the free (drift) Hamiltonian, H_k is the kth control Hamiltonians and $f_k(t)$ is its corresponding control field which is a time-dependent real function. The aim is to find proper $f_k(t)$ such that desired states are prepared at some time.

In a quantum Lyapunov control problem, a real function V called Lyapunov function (conventionally $V \geq 0$) is assigned and $f_k(t)$ are designed to guarantee $\dot{V} \leq 0$. Through this, the quantum system is driven to the LaSalle invariant set satisfying $\dot{V} = 0$ as $t \to \infty$ [1,10,11]. The choice of Lyapunov function V is not unique. For example, V could be chosen as the distance between the quantum state and the desired state, the expectation value of a Hermitian operator and so on [10]. Here we consider the second form of Lyapunov function, i.e.,

$$V(|\Psi\rangle) = \langle\Psi|P|\Psi\rangle, \tag{2}$$

where P is a Hermitian and positive semi-definite operator, so that $V \geq 0$ and $|\Psi\rangle$ is the argument of V. This form of Lyapunov function is widely studied in literature and covers some other forms of Lyapunov function such as that based on the Hilbert–Schmidt distance [1]. More importantly, there is freedom in designing P enabling us to optimize it for different initial states for the purpose of this paper.

To design the control fields that decrease the Lyapunov function V, it is helpful to calculate its time derivative,

$$\dot{V} = \langle\Psi|i\left[H_0 + \sum_{k=1}^{m} f_k(t)H_k, P\right]|\Psi\rangle \tag{3}$$

$$= \sum_{k=1}^{m} f_k(t)\langle\Psi|i[H_k, P]|\Psi\rangle. \tag{4}$$

Here $[H_0, P] = 0$ is assumed to cancel the drift term. This condition could be realized by constructing the operator P as $P = \sum_{l=1}^{n} p_l|E_l\rangle\langle E_l|$, where $|E_l\rangle$ is the lth

eigenstate of H_0 and p_l are nonnegative real coefficients. For the eigenstate control problem, assume that the goal state is $|E_g\rangle$ with coefficient p_g. By setting p_g to be the smallest one, the goal state is obtained if V decreased to its minimum. Thus the operator P can be expressed as

$$P = p_g|E_g\rangle\langle E_g| + \sum_{l\neq g} p_l|E_l\rangle\langle E_l|, \qquad p_g < p_l. \tag{5}$$

The control fields $f_k(t)$ are conventionally designed as

$$f_k(t) = -K\langle\Psi|i[H_k, P]|\Psi\rangle \tag{6}$$

where K is a real constant associated with the control strength. Other approaches [12–15] to design the control fields have also been proposed to improve the performance of quantum Lyapunov control. From Eqs. (4) and (6), we have

$$\dot{V} = -K^{-1}\sum_{k=1}^{m} f_k^2(t) \leq 0, \tag{7}$$

i.e., the Lyapunov function remains non-increasing with time. By numerically simulating the nonlinear dynamics Eq. (1) with the control law Eq. (6) using, e.g., the Runge–Kutta algorithm, one can obtain the time dependence of the quantum states and the control fields without the need of iterations.

With ideal free Hamiltonian and control Hamiltonians, it is known that if $p_l = p \geq p_g$ for all $l \neq g$, the control law determined by Eqs. (2, 5, 6) can drive any initial state $|\Psi_0\rangle$ (except those satisfies $\langle\Psi_0|E_g\rangle = 0$) asymptotically to the goal state $|E_g\rangle$ as $t \to \infty$ [1,7,8,15]. The convergence for a more general case where p_l are different is investigated in [1,8,10] where the values of p_l influence the convergence and its region of attraction. In this work, the coefficients p_l ($l \neq g$) will be optimized for different initial states and predicted by trained artificial neural networks. Obviously, the performance (e.g., fidelity, control time) of quantum Lyapunov control depends on the control parameters such as Lyapunov function V and control Hamiltonian H_k. Choosing appropriate parameters is therefore of great importance for Lyapunov control problems.

3 Artificial neural networks

In this section, we briefly introduce two neural network models used in this paper, feedforward neural network and general regress neural network. Mathematically, these neural networks could be understood as a function that maps an input real vector X to an output real vector Y.

3.1 Feedforward neural network

Feedforward neural network is the most well-known neural network. A schematic diagram of a feedforward neural network is shown in Fig. 1. A feedforward neural

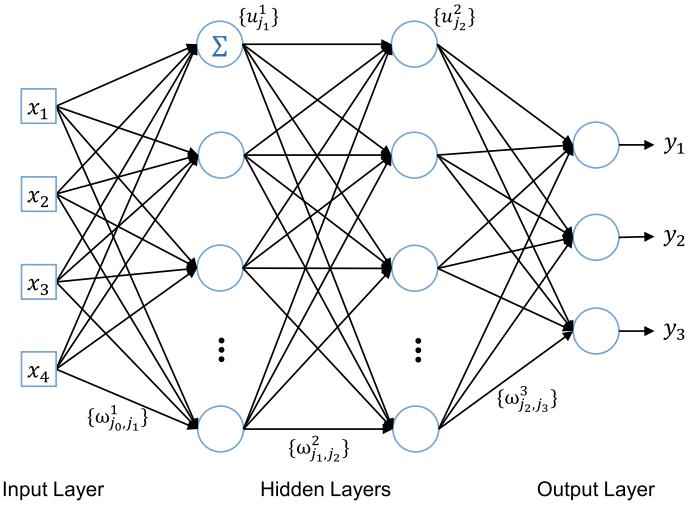

Fig. 1 A schematic diagram of a feedforward neural network 4 input nodes, 3 output neurons and 2 hidden layers. Here $j_l = 1, 2, \ldots, n_l$ where n_l is the node (neuron) number of the lth layer

network consists of a layer of input nodes (by squares in Fig. 1), an output layer of neurons (processing units, by circles in Fig. 1), and possibly a set of hidden layers of neurons. In feedforward neural networks, the signal flows from the input layer to the output layer without feedback loops. A feedforward neural network with one or more hidden layers is called a multilayer perceptron [32,33]. With enough neurons, a multilayer perceptron is able to approximate any continuous nonlinear function and solve many complicated tasks. In a feedforward neural network, the output y of a single neuron is expressed by

$$y = s\left(\sum_{i=1}^{m} x_i \omega_i + b\right), \tag{8}$$

where the x_i is the output of the ith neuron (node) of the last layer, ω_i is the weight of x_i corresponding to the arrows in Fig. 1, and b is a bias (threshold) which is omitted in Fig. 1. $s(...)$ is called an activation function which is usually a nonlinear sigmoid function limiting the strength of the output signal. The logistic function

$$s(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{9}$$

is used as the activation function in this paper that transfers any input signal to the range 0 to 1 .

In a feedforward neural network with m layers of neurons (n_l neurons in the lth layer) and an input layer with n_0 nodes, the input $X = [x_1\ x_2\ \ldots\ x_{n_0}]^T$ is transformed to the output $Y = [y_1\ y_2\ \ldots\ y_{n_m}]^T$ by

$$u_{j_1}^1 = s\left(\sum_{j_0=1}^{n_0} x_{j_0} w_{j_0,j_1}^1 + b_{j_1}^1\right), \tag{10}$$

$$u_{j_2}^2 = s\left(\sum_{j_1=1}^{n_1} u_{j_1}^1 w_{j_1,j_2}^2 + b_{j_2}^2\right), \tag{11}$$

$$\vdots$$

$$y_{j_m} = s\left(\sum_{j_{m-1}=1}^{n_{m-1}} u_{j_{m-1}}^m w_{j_{m-1},j_m}^m + b_{j_m}^m\right) \tag{12}$$

where $j_l = 1, 2, \ldots, n_l$ with $l = 0, 1, \ldots, m$. Here $u_{j_l}^{\alpha}$ is the output of the j_lth neuron of the αth neuron layer. In the above equations, the superscript (1,2,…,m) denotes the index of the neuron layers, and the subscript j_l represents the j_lth neuron or node of the lth layer, as shown in Fig. 1.

For a specific problem, the design of the feedforward neuron network structure is generally empirical. The training of a feedforward neuron network is implemented by adjusting its weights and biases. In supervised learning, the weights and biases are able to be efficiently trained by the back propagation (BP) algorithms [32,33] with training samples including a number of input vectors and their target output vectors.

3.2 General regression neural network

General regression neural network (GRNN) is a type of radial basis function (RBF) network proposed by D. F. Specht in 1991 [45]. It is a powerful tool to estimate continuous variables [46–49] even when the training data is few. A general regression neural network consists of four layers: an input layer, a pattern layer, a summation layer and an output layer as shown in Fig. 2. In contrast to the feedforward neural network, the neuron number in each layer of a GRNN is determined from its training samples. For a training set with N samples $\{X^k, Y^k\}, k = 1, 2, \ldots, N$ where $X^k = [x_1^k\ x_2^k\ \ldots\ x_{n_I}^k]$ is the kth input vector and $Y^k = [y_1^k\ y_2^k\ \ldots\ y_{n_O}^k]$ is its target output, the pattern layer neuron number is N, the number of the input (output) layer node is n_I (n_O), and the number of the summation layer neuron is $n_O + 1$.

For an input X, the output of the pattern layer ($u_k^p, k = 1, 2, \ldots, N$), the summation layer ($u_{j_s}^s$, $j_s = 0, 1, 2, \ldots, n_O$) and the output layer ($y_j$, $j = 1, 2, \ldots, n_O$) is given by

$$u_k^p = \exp\left(-\frac{(X - X^k)^T (X - X^k)}{2\sigma^2}\right), \tag{13}$$

$$u_0^s = \sum_{k=1}^{N} u_k^p, \quad u_{j_s(j_s\neq 0)}^s = \sum_{k=1}^{N} y_{j_s}^k u_k^p, \tag{14}$$

$$y_j = \frac{u_j^s}{u_0^s}, \tag{15}$$

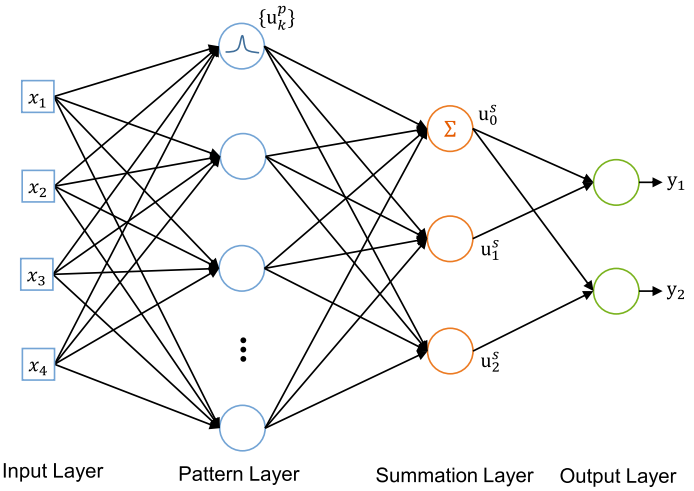

Fig. 2 A schematic diagram of a general regression neural network with four input nodes and two output neurons

respectively. The estimation y_j for an input X can be understood as an average of all y_j^k weighted exponentially according to the Euclidean distance between X and X^k. Here σ is called a smoothing parameter ($\sigma > 0$). When σ is small, the estimation y_i for X is closely related to y_i^k whose inputs X^k are close to X. When σ is large, y_i approaches the mean of all y_j^k. The GRNN is established as soon as the training samples are stored while the smoothing parameter σ is the only adjustable parameter needed to train.

4 Initial-state-adaptive strategy

When dealing with different initial states, better performances may be achieved if the parameters in quantum Lyapunov control, such as those in the Lyapunov function V or the control Hamiltonian H_k, are optimally chosen for each initial state, i.e., initial-state-adaptive. However, finding optimized parameters with standard techniques typically costs more computing resources since Lyapunov control fields are calculated numerically. In this work, the main computing resource we care about is the computational time to generate control fields. In this way, quantum Lyapunov control would lose its simplicity that the control fields are obtained with only one simulation of system dynamics. To tackle this issue, we propose to design an initial-state-adaptive Lyapunov control with neural networks.

The basic strategy comprises the following steps.

1. Generate a certain number of initial states that are randomly distributed in an interested space.
2. Numerically find the optimal control parameters (or the best choice from several candidate schemes) for each initial state. Data from 1. and 2. constitute the training samples (testing samples if needed).

3. Build a neural network whose input is associated with the initial state parameters and output associated with the optimal control parameters (or the best choices of schemes).
4. Train the neural network by supervised learning with the training set until the neural network performance is satisfactory.
5. Apply the trained neural network to predict control parameters (choices of schemes) for new initial states.

As the processes of generating samples and training neural networks are implemented in advance, the computing resources (processing time) of our neural network control would not significantly increase in the application stage. To analyze the efficiency of our strategy, we may consider two cases of applications: the real-time-control case and the many-initial-states case. In the first case, assume that the control fields need to be calculated immediately for initial states that are unknown before the application. Then the training samples cannot be directly used in the application. So the sample generation and training would have been carried out regardless of the computational cost to some extent. In the many-initial-states case, consider that a large number of initial states is involved in an application. Now, our strategy is more efficient than the standard approaches only if the initial states in the application stage are much more than the training (testing) samples.

With the eigenstate control problem described in Sect. 2, we propose two designs where the mentioned neural networks are used to select control schemes or predict Lyapunov functions for different initial states. Two important functions of neural networks, classification and regression, are employed.

4.1 Classification: selecting control schemes

Given a quantum control task with many initial states, we assume that there are several possible Lyapunov control schemes that appear in an application, where the Hamiltonians, Lyapunov functions, or other parameters are different. Meanwhile, one of these schemes will be eventually adopted for each initial state. Our aim is to use neural networks to predict which scheme is the best one for each initial state.

Specifically, assume there are M candidate control Hamiltonians $H_c (c = 1, 2, \ldots, M)$ in a application. These Hamiltonians could represent different physical settings in experiments. For example, they may correspond to control fields acting on different subsystems in a complex system or control fields coupling different levels of a n-level system, etc. One of the control Hamiltonians will be selected in experiment for simplicity or feasibility. Thus the dynamics is described by

$$\frac{\mathrm{d}}{\mathrm{d}t}|\Psi\rangle = -i[H_0 + f(t)H_c]|\Psi\rangle. \tag{16}$$

Other parameters such as the Lyapunov function V, the strength K, and H_0 are fixed. The task is preparing an eigenstate of H_0, say, $|E_g\rangle$, as discussed in Sect. 2. Given an initial state, we will use a feedforward neural network to predict the control Hamiltonian that leads to the highest fidelity defined by $F = |\langle\Psi(T)|E_g\rangle|^2$ at a certain

control time T. The problem is solved by classifying the initial states according to their corresponding control Hamiltonians with the feedforward neural network.

For a n-dimensional system, the initial state $|\Psi_0\rangle$ written in the eigenbasis of H_0 is

$$|\Psi_0\rangle = c_1|E_1\rangle + c_2|E_2\rangle + \cdots c_n|E_n\rangle, \tag{17}$$

where $c_{1,2,\ldots,n}$ are the complex probability amplitudes. Due to the normalization condition $\sum_{l=1}^{n} |c_l|^2 = 1$ and the arbitrariness of a non-physical global phase, the initial state could be described by $2(n-1)$ parameters ($|c_l|$ and ϕ_l, $l = 1, 2, \ldots, n-1$) as $e^{i\phi_g}|\Psi_0\rangle = \sum_{l=1}^{n-1} |c_l|e^{i\phi_l}|E_l\rangle + |c_n||E_n\rangle$. Here the phase of $|E_n\rangle$ is eliminated by ϕ_g and ϕ_l is the phase difference between $|E_l\rangle$ and $|E_n\rangle$.

We define the training set with N_{train} samples as

$$S = \{(X^1, Y^1), (X^2, Y^2), \ldots, (X^{N_{\text{train}}}, Y^{N_{\text{train}}})\}. \tag{18}$$

In the kth sample, X^k is the input vector with $2(n-1)$ elements defined as

$$X^k = \left[\begin{array}{cccccccc} |c_1^k|^2 & |c_2^k|^2 & \ldots & |c_{n-1}^k|^2 & \phi_1^k & \phi_2^k & \ldots & \phi_{(n-1)}^k \end{array} \right]^T, \tag{19}$$

where $|c_l^k|^2$ is the absolute square of c_l^k and $-\pi \leq \phi_l^k \leq \pi$ is specified for consistency. The state vectors $|\Psi_0\rangle$ in the training set are randomly picked with a uniform distribution on the unit sphere in $\mathbb{C}^n$. One simple method [50] to do this is to generate n independent complex random numbers $z_{l=1,2,\ldots,n}$ with a standard normal distribution and let $c_l = z_l / \sqrt{\sum_l |z_l|^2}$ in Eq. (17). For an initial state, its corresponding control Hamiltonian is determined by simulating the dynamics M times with the candidate control Hamiltonians and choosing the one with the highest fidelity. The target output vector Y^k indicating the choices are represented by the standard basis $\mathbf{e}$ with $M+1$ elements. For example, the control Hamiltonian could be mapped to the output vector Y by

$$\begin{aligned} H_1 &\longrightarrow \mathbf{e}_1 = \quad [1\,0\,\ldots\,0]^T, \\ H_2 &\longrightarrow \mathbf{e}_2 = \quad [0\,1\,\ldots\,0]^T, \\ &\vdots \\ H_M &\longrightarrow \mathbf{e}_M = \quad [0\,\ldots\,1\,0]^T, \\ others &\longrightarrow \mathbf{e}_{M+1} = \quad [0\,\ldots\,0\,1]^T. \end{aligned} \tag{20}$$

Here *others* refers to the case of inefficient controls, i.e., all fidelities are lower than a limit. H_j refer to the case that H_j corresponds to the highest fidelity (greater than the limit). On the other hand, a testing set S_T with N_{test} samples could be generated in a similar way as the training set Eq. (18). The testing set (outside the training set) is used to estimate the generalization ability of the neural network and help determine the stopping time of the training process.

Next, a feedforward neural network with $2(n-1)$ input nodes, $M+1$ output neurons, plus some hidden layers is set up. For an input vector X, the output of

the neural network could be represented by the linear combination of all the basis vectors, i.e., $Y' = \sum_{j=1}^{M+1} q_j \mathbf{e}_k$, $(0 < q_j < 1)$. The classification is implemented by selecting the choice with the largest coefficient q_j. Here q_j might be understood as an unnormalized probability that the choice is j. The performance of a neural network could be measured by the mean squared error (MSE)

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(Y'^k - Y^k)^T(Y'^k - Y^k) \tag{21}$$

where Y'^k is the output of the neural network for X^k and Y^k is the kth target output vector. N is the number of the training (or testing) samples.

With the training set Eq. (18) determined by Eqs. (19) and (20), one can use the back propagation (BP) algorithm to train the feedforward neural network efficiently. The iteration number of the BP training process could be determined by checking the mean squared error (or the classification success rate) for the testing set. Before the training (testing, application) process, the input vector $X^k = [x_1^k\ x_2^k\ \ldots\ x_{n_I}^k]$ is normalized to $X'^k = [x_1'^k\ x_2'^k\ \ldots\ x_{n_I}'^k]$ by $x_j'^k = 2(x_j^k - x_j^{\min})/(x_j^{\max} - x_j^{\min}) - 1$ where $x_j^{\max}$ $(x_j^{\min})$ is the maximum (minimum) of the N_{train} input vector elements x_j^k $(k = 1, 2, \ldots, N_{\text{train}})$ from the training set. In this way, all the input elements (input node signals) are scaled to the range $[-1, 1]$ such that the sigmoid functions in the neural network are sensitive to them. Finally, the trained neural network will be used to select control Hamiltonian for new initial states (out of the training set).

In this subsection, there is one control Hamiltonian in each control scheme for simplicity. Note that the classification method also applies to other cases. For example, selecting schemes where there are more control Hamiltonians in each scheme or selecting schemes where the Lyapunov function, control law, free Hamiltonian or other parameters are different. For these cases, our method could be applied in a similar way.

4.2 Regression: designing Lyapunov function

In this section, the GRNN is used to design initial-state-adaptive Lyapunov functions V of the form Eq. (2) where the operator P depends on the initial state, i.e., $V = \langle\Psi|P(|\Psi_0\rangle)|\Psi\rangle$. The free Hamiltonian H_0 and control Hamiltonian(s) H_k are fixed. The task is to prepare an eigenstate of H_0 with a high fidelity defined as $F = |\langle\Psi(T)|E_g\rangle|^2$ at time T.

Notice that the strength coefficient K in Eq. (6) can be absorbed into the operator P, i.e., $V' = \langle\Psi|KP|\Psi\rangle = \langle\Psi|P'|\Psi\rangle$. Therefore, we set $K = 1$ and merely discuss P for simplicity. Assume the goal state is the gth eigensate of H_0 denoted by $|E_g\rangle$. For a n-dimensional system, the operator P is designed as

$$\begin{aligned} P = &\quad p_g|E_g\rangle\langle E_g| + \sum_{l\neq g} p_l|E_l\rangle\langle E_l| \\ = &\sum_{l\neq g} p_l|E_l\rangle\langle E_l| \qquad (p_l > p_g = 0). \end{aligned} \tag{22}$$

We have set the minimum coefficient p_g to 0 without loss of generality, since if $p_g \neq 0$, one can shift it to zero by adding $-p_g \sum_l |E_l\rangle\langle E_l| = -p_g \mathrm{I}$ to P, which does not change the control fields according to Eq. (6). Now the optimal Lyapunov function for an initial state could be obtained by optimizing $p_{l(l\neq g)}$ numerically (the number of p_l is $n-1$), using, for example, gradient-descent methods or derivation-free algorithms. In principle, there is no limitation on the bounds of p_l in optimization. In practice, some constraints of p_l are required to limit the strength of the control fields and facilitate the numerical optimizations, e.g., $0 < p_l \leq p_l^{\max}$.

In the second design, the training set (testing set if needed) with N_{train} (N_{test}) samples can be defined similarly as Eq. (18). The input vector X^k is given by Eq. (19) and the kth output vector Y^k is defined as

$$Y^k = \left[p_1^k \; p_2^k \; \ldots \; p_{n-1}^k\right]^T. \tag{23}$$

The elements of the target vector $p^k_{j=1,2,\ldots,n-1}$ are the optimal values in one-to-one correspondence with p_l in Eq. (22). Given an input vector X^k (an initial state), Y^k is obtained by numerically finding p_j^k that maximizing the fidelity F_k with the constraints of p_j^k, in which the dynamics needs to be simulated many times.

With the training set, a GRNN with $2(n-1)$ input nodes, N_{train} pattern layer neurons and $n-1$ output neurons can be built straightforwardly. The smoothing parameter σ [defined in Eq. (13)] could be determined by checking the GRNN performance for the testing samples without many trials. Different performances for the testing samples might be used such as the MSE or the averaged logarithmic infidelity defined by

$$\epsilon = \frac{1}{N}\sum_{k=1}^{N} \log(1 - F_k). \tag{24}$$

Here F_k is the fidelity of the kth initial state. In this design, we let the input vectors elements for the GRNN normalize to $[-1, 1]$ as done in the first design, such that the GRNN is sensitive to every element of an input vector. The average space between two neighboring normalized input vectors could be roughly estimated by $D = 2/\sqrt[n_I]{N_{\text{train}}}$ where $n_I = 2(n-1)$ is the dimension of the input vector. We suggest finding the smoothing parameter in the range $0 < \sigma < \sigma_{\max}$ where $\sigma_{\max} \sim D$. With an appropriate smoothing parameter, the GRNN will be finally used to predict optimal P of the Lyapunov function for new initial states.

5 Examples

In this section, we demonstrate the usage and performance of our designs with a driven three-level atomic system ($n = 3$). The time-independent free Hamiltonian is described by

$$H_0 = \sum_{n=1}^{3} \omega_n |n\rangle\langle n| + g(|1\rangle\langle 2| + |2\rangle\langle 1|), \tag{25}$$

where ω_n is the frequency of the nth level of the three-level atom and the state $|2\rangle$ and $|1\rangle$ are coupled with a constant strength g. The Hamiltonian could be realized in a three-level atom driven by a classical electromagnetic field with a constant strength in a rotating frame. The task is to prepare an eigenstate of H_0 with high fidelity, e.g., $|E_3\rangle$ where E_3 is the highest eigenenergy. The fidelity is calculated by $F = |\langle\Psi(T)|E_3\rangle|^2$ with T the control time. The dynamics is described by Eq. (16) and the control field is given by Eq. (6).

5.1 Selecting control Hamiltonians

To illustrate the first design, we consider 4 candidate control Hamiltonians ($M = 4$),

$$H_1 = |1\rangle\langle 3| + |3\rangle\langle 1|, \tag{26}$$

$$H_2 = |2\rangle\langle 3| + |3\rangle\langle 2|, \tag{27}$$

$$H_3 = |1\rangle\langle 3| + |3\rangle\langle 1| + |1\rangle\langle 2| + |2\rangle\langle 1|, \tag{28}$$

$$H_4 = |2\rangle\langle 3| + |3\rangle\langle 2| + |1\rangle\langle 2| + |2\rangle\langle 1|. \tag{29}$$

The associated control fields could be realized amplitude-modulated classical electromagnetic fields in a rotating frame. Given an arbitrary initial state $|\Psi_0\rangle$, we will use a feedforward neural network to select the control Hamiltonian that leads to the highest fidelity. In this example, $P = |E_1\rangle\langle E_1| + |E_2\rangle\langle E_2| = \mathrm{I} - |E_3\rangle\langle E_3|$ is used in the Lyapunov function of the form Eq. (2). Thus $V = \langle\Psi|P|\Psi\rangle = 1 - |\langle\Psi|E_3\rangle|^2$ can be understood as either Eq. (22) with $p_1 = p_2 = 1$ (unoptimized) or a distance between the controlled state and the goal state. According to our method, the input vectors in the training (testing) set are given by $X^k = [|c_1^k|^2\ |c_2^k|^2\ \phi_1^k\ \phi_2^k]^T$, $k = 1, 2, \ldots, N_{\text{train}}(N_{\text{test}})$ where $|c_{1,2}^k|^2$ and $\phi_{1,2}^k$ is defined as in Eq. (17) with $n = 3$. We consider five choices ($H_{1,2,3,4}$, and *others*) corresponding to target output vectors $\mathbf{e}_{1,2,\ldots,5}$, respectively. Here *others* refers to all fidelities less than 0.99.

To illustrate the training process, we generated a training set with 10^4 samples (19% H_1 , 15% H_2, 42% H_3, 19% H_4 and 5% *others*) plus a testing set with $N_{\text{test}} = 5 \times 10^3$ samples through simulating the dynamics with different candidate Hamiltonians by the fourth-order Runge–Kutta method. In our simulations, the control parameters are $\omega_2 = 2\omega_1$, $\omega_3 = 5\omega_1$, $g = 0.5\omega_1$, $K = 1$, and the control time is $T = 6\pi/\omega_1$. We set up a feedforward neural network with four input nodes, five output neurons and two hidden layers of 40 neurons, respectively. The feedforward neural network was trained by a back propagation algorithm to minimize the MSE for the training set where a scaled conjugate gradient algorithm was used. The training was implemented by the "trainscg" function of the MATLAB Neural Network Toolbox with default training parameters. The training process is illustrated in Fig. 3 where the MSEs and the classification success rates R for the training set and the testing set are evaluated every 50 iterations and plotted. The MSEs for both set decreased dramatically in hundreds of iterations together with a rapid increase of the classification success rates (exceed

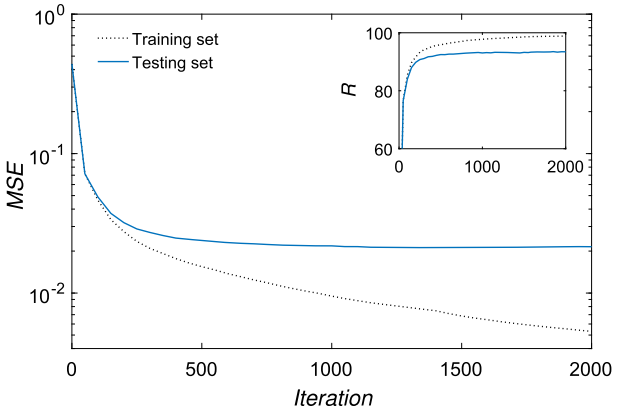

Fig. 3 The training process of the feedforward neural network. The mean squared error (MSE) for the training set (dotted line) and the testing set (solid line) versus the iteration number. Inset: the percentage of the classification success rate (R) for the training set and the testing set

90%). The training set MSE monotonically decrease through the training process due to the gradient algorithm, whereas the testing set MSE might slightly oscillate in the later iterations.

We then conducted five studies with different numbers of training samples (N_{train}) where the testing sample numbers are the same ($N_{\text{test}} = 5 \times 10^3$) for comparison. The control parameters, the neural network structure, and the training algorithms are the same as those in Fig. 3. In these studies, we adopted the iteration numbers corresponding to the minimal testing set MSEs (evaluated every 50 iterations in at most 5×10^3 iterations) to determine the weights and biases of the neural networks. Finally, the trained neural networks were applied to predict control Hamiltonians for $N_{\text{app}} = 10^5$ new random initial states (generated similarly as done for the samples) as applications. The corresponding classification success rates are denoted by R_{app}. The training and application details are shown in Table 1. In these studies, the MSEs for the testing set almost decreased to their minimums with hundreds of iterations. It is seen that the success rate R_{app} was greater than 90% with 5×10^3 training samples. Further studies showed that more training samples, more hidden layer neurons, more training iterations, or less choice number M generally results in higher success rate R_{app}. Next we compared the neural network predictions ($N_{\text{train}} = 5 \times 10^4$) with the result by

Table 1 Feedforward neural network performances for different training sample numbers

N_{train}	N_{test}	MSE	Iteration	N_{app}	R_{app} (%)
5×10^2	5×10^3	0.0619	2×10^2	10^5	79.9
1×10^3	5×10^3	0.0465	2.5×10^2	10^5	85.5
5×10^3	5×10^3	0.0243	8×10^2	10^5	92.1
1×10^4	5×10^3	0.0194	1.1×10^3	10^5	94.0
5×10^4	5×10^3	0.0109	4.55×10^3	10^5	96.6

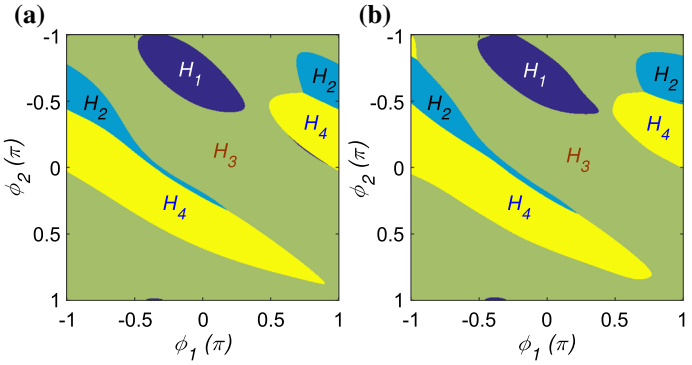

Fig. 4 The dependences of the choice on ϕ_1 and ϕ_2 for the standard approach (**a**) and the neural network control (**b**). $|c_1|^2 = 0.2$ and $|c_2|^2 = 0.3$ are set. The 5th choice (low fidelities) does not exist with these parameters. Each subfigure contains 500×500 pixels corresponding to different ϕ_1 and ϕ_2

the standard approach (simulating the dynamics M times with different candidate Hamiltonians). The dependences of the choice on ϕ_1 and ϕ_2 (with $|c_1|^2 = 0.2$ and $|c_2|^2 = 0.3$) for the two approaches are compared in Fig. 4. The similarity of the two results reveals the learning capability of the feedforward neural network.

In the following, we compare the processing time of the standard approach (denoted by T_S) with that of our neural network control (denoted by T_{NN}) in the application stage (run on the same PC with MATLAB). We generated 10^3 samples with $M = 4$. The average simulation time for one dynamics (with 500 time steps) was about $t_1 = 3.5 \times 10^{-2}$ s. In our studies, the prediction time of the feedforward neural network depends on how the input vectors are sent to the neural network function, in batch or one-by-one. Using the neural network in Fig. 4 as an example, with a batch of 10^4 input vectors, the average processing time for one prediction was about $t_2 = 4 \times 10^{-6}$ s which is negligible compared with t_1. t_2 became even shorter with larger batch size. Using one-by-one inputs, a test with 10^4 input vectors showed that the average prediction time for one initial state was about $t_3 = 8 \times 10^{-3}$ s, which is also evidently smaller than t_1. With the above time scales, in the many-initial-states case with a batch input, the efficiency of our design is reflected by the ratio $T_{NN}/T_S \approx (t_1 + t_2)/(Mt_1) \approx 1/M$ where $M = 4$ is the number of candidates. In the cases with a one-by-one input, the ratio is $T_{NN}/T_S \approx (t_1 + t_3)/(Mt_1) \approx 1.2/M$. Here T_S itself is a small number although $T_S \sim MT_{NN}$. Nevertheless, our design would save more time compared with the standard one with a more complex model. More importantly, in the real-time-control case, e.g., the measurement-based feedback controls [3], the time-delay effects of feedback are often detrimental. Thus the reduction of computational time by our design would be helpful even if T_S is small.

5.2 Designing Lyapunov functions

In this example, we will use GRNN to design initial-state-adaptive Lyapunov functions for an eigenstate control problem. We consider the same physical system in the last example but with $g = 0$ in the free Hamiltonian, i.e.,

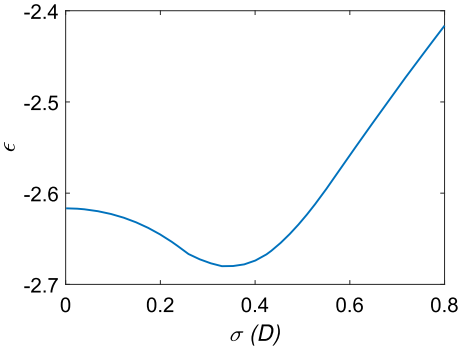

Fig. 5 The dependence of the averaged logarithmic infidelity ϵ on the smoothing parameter σ of the GRNN [defined in Eq. (13)] for $N_{\text{test}} = 2 \times 10^3$ testing initial states

$$H_0 = \sum_{n=1}^{3} \omega_n |n\rangle\langle n|. \tag{30}$$

The control parameters are $\omega_2 = 2\omega_1$, $\omega_3 = 5\omega_1$, $K = 1$ and $T = 0.3\pi/\omega_1$. The task is to prepare the eigenstate of H_0 with the highest energy, i.e., $|E_g\rangle = |E_3\rangle = |3\rangle$ with high fidelity. The dynamics is described by Eq. (1) where two control fields $f_1(t)$ and $f_2(t)$ are used with control Hamiltonians given by Eqs. (26) and (27), respectively. The Lyapunov function is of the form Eq. (2). The operator P is designed by $P = p_1|E_1\rangle\langle E_1| + p_2|E_2\rangle\langle E_2|$ ($p_3 = 0$) according to Eq. (22).

We generated totally 10^5 samples for training where the input vector is $X^k = [|c_1^k|^2 \ |c_2^k|^2 \ \phi_1^k \ \phi_2^k]^T$ and the target output vector is $Y^k = [p_1^k \ p_2^k]^T$. Here p_1^k and p_2^k correspond to p_1 and p_2, respectively, which were found by minimizing the infidelity with the interior-point algorithm using the function "fmincon" in the MATLAB Optimization Toolbox. For each random initial state, five optimizations with random starting points were implemented to avoid local minimums. $p_{1,2}$ were optimized with the constraints $0 \le p_1 \le 20$, $0 \le p_2 \le 20$. It is observed that most optimal values of $p_{1,2}$ for the initial states lie inside the area given by the constraints (rather than near the edges of the area), implying the constrains are plausible. The averaged logarithmic infidelity ϵ [defined by Eq. (24)] for these training samples is -3.23 and the fraction of fidelities that are greater than 0.999 is $R_{F>0.999} = 41.4\%$.

With a training set of $N_{\text{train}} = 5 \times 10^4$ samples, we set up a GRNN with four input nodes, two output neurons, three summation layer neurons and N_{train} pattern layer neurons. The smoothing parameter σ was determined by checking the averaged logarithmic infidelity ϵ for $N_{\text{test}} = 2 \times 10^3$ random testing initial states (out of the training set) with several trials and choosing the smoothing parameter with the minimal ϵ. The dependence of ϵ on the smoothing parameter σ is shown in Fig. 5 with $0.001D \le \sigma \le 0.8D$ where $D = 2/\sqrt[4]{N_{\text{train}}}$ is the scale of the average space (introduced in Sect. 4.2) between two neighboring normalized input vectors in the training set. Finally, $\sigma = 0.33D$ was adopted for the GRNN. As an application, the trained GRNN was used to predict Lyapunov functions for $N_{\text{app}} = 2 \times 10^5$ new random initial states, after which the fidelities was calculated by simulating the dynamics. The averaged logarithmic infidelity and the fraction of $F > 0.999$ are $\epsilon_{\text{app}} = -2.69$ and $R_{F>0.999} = 35.5\%$, respectively. To further demonstrate the performance of the GRNN, we calculated

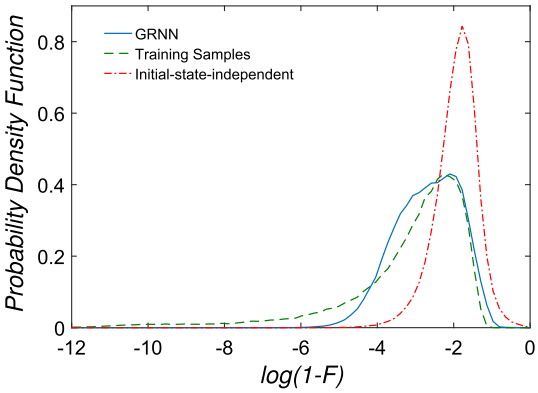

Fig. 6 Infidelity distributions of the final states for different control approaches. The solid line and dashed line represent the distributions for the GRNN control ($N_{\text{train}} = 5 \times 10^4$) and the training samples (by numerical optimizations), respectively. The dot-dashed line represents the result with an optimized initial-state-independent Lyapunov function

Table 2 GRNN performances for different training sample numbers

N_{train}	N_{test}	σ/D	N_{app}	ϵ_{app}	$R_{F>0.999}$ (%)
1×10^3	2×10^3	0.26	2×10^5	−2.29	14.8
5×10^3	2×10^3	0.26	2×10^5	−2.48	23.5
1×10^4	2×10^3	0.31	2×10^5	−2.55	28.1
5×10^4	2×10^3	0.33	2×10^5	−2.69	35.5
1×10^5	2×10^3	0.38	2×10^5	−2.74	37.5

the distribution of the infidelities in the application. This distribution is compared with that of all the 10^5 training samples and that from an initial-state-independent Lyapunov control with $V = \langle\Psi|P_{\text{ind}}|\Psi\rangle$ for $N_{\text{app}} = 2 \times 10^5$ random initial states (see Fig. 6). Here $P_{\text{ind}} = 4.406|E_1\rangle\langle E_1| + 4.117|E_2\rangle\langle E_2|$ was obtained by numerically minimizing ϵ for 2×10^3 testing random initial states implemented also by the interior-point algorithm. As seen in Fig. 6, the infidelity distribution for the GRNN control is similar to that of the training samples, especially when $\log(1 - F) > -2.5$. The fraction of $F > 0.999$ for the GRNN control is near that of the training samples. In contrast, the initial-state-independent control created more low-fidelity final states with $\epsilon_{\text{app}} = -1.93$ and $R_{F>0.999} = 4.1\%$, although P_{ind} had been optimized. We remind that an arbitrary unoptimized initial-state-independent P may lead to a much worse result. For example, $p_1 = p_2 = 10$ lead to $\epsilon_{\text{app}} = -1.56$ and $R_{F>0.999} = 1.2\%$ in a application with $N_{\text{app}} = 2 \times 10^5$ random initial states.

We further tested the performances of several GRNNs based on different numbers of training samples. For these GRNNs, we found their smooth parameters as done in Fig. 5 with $N_{\text{test}} = 2000$ testing initial states. The GRNNs were applied to the same Lyapunov control problem for $N_{\text{app}} = 2 \times 10^5$ new random initial states. The details of the training and applications are shown in Table 2. It is seen that $R_{F>0.999}$ grows (ϵ_{app} deceases) with more training samples. The performance of the GRNN control is evidently better than the initial-state-independent control ($\epsilon_{\text{app}} = -1.93$, $R_{F>0.999} = 4.1\%$) even with relatively less training samples.

In the following, we compare the processing time of the standard approach (denoted by T_S) used in sample generation with the processing time of the GRNN control

(denoted by T_{NN}) in the application stage (run on the same PC with MATLAB). We generated 10^2 samples as before where five starting points were used in the numerical optimization for each initial state. The average optimization time for one initial state was about $t_1 = 9.4\,\mathrm{s}$ where the dynamics was simulated about 360 times. So the average simulation time (with 200 time steps) for one dynamics was roughly $t_1' = t_1/360 = 2.6 \times 10^{-2}\,\mathrm{s}$. The prediction time of the GRNN function depends on the number of training samples stored in it, also on whether the input vectors are sent in batch. Using the GRNN trained by 5×10^4 samples in a test, the average prediction time was about $t_2 = 1.8 \times 10^{-3}\,\mathrm{s}$ with a batch of 10^4 samples. With one-by-one inputs, the average prediction time was about $t_3 = 2.6 \times 10^{-2}\,\mathrm{s}$ with a test of 10^3 initial states. The predicting time of the GRNN with $N_{\mathrm{train}} = 5 \times 10^4$ is similar to the time of one simulation in this model. With the above time scales, in the many-initial-states case with a batch input, the efficiency of our GRNN design is reflected by the ratio $T_{NN}/T_S \approx (t_1' + t_2)/t_1 = 0.30\%$. In the cases with a one-by-one input, the ratio is $T_{NN}/T_S \approx (t_1' + t_3)/t_1 = 0.55\%$. It is seen that the GRNN would require much less processing time to predict an optimal Lyapunov function in the application stage.

6 Summary and discussion

We have proposed two initial-state-adaptive Lyapunov control designs with machine learning where the feedforward neural network and the GRNN are used to select control schemes and predict Lyapunov functions for different initial states. The aim of the designs is to improve the control performance without much increase of computing resources in the application stage. We illustrated our designs with an eigenstate control problem in a three-level atomic system. Our results show that the neural networks can efficiently learn the relationship between the initial states and the optimal control schemes or the optimal Lyapunov functions.

Our designs apply to Lyapunov control problems when many initial states are involved or fast calculation of control fields is needed to improve the control performance for many initial states. In the real-time-control case, the sample generation and neural network training are carried out regardless of the computational cost to some extent. In the many-initial-states case, our designs are more efficient than the standard techniques if the number of initial states in the applications is much higher than the training (testing) sample number. A possible scenario for our designs is the quantum measurement-based feedback control with quantum filtering [3,22–25]. In these models, the states of the quantum system are estimated using the results from weak measurements and the feedback is designed based on the estimations. During the dynamics, a large number of different states might be estimated and fast calculations of control fields are required. Thus our control might be used as a step in the measurement-based feedback control to transfer different states to a final state.

In our examples, the samples were divided into one training set and one testing set where we have assumed that the number of testing samples is large enough to reflect the generalization ability of the neural network. When the number of samples is limited, other methods such as a k-fold cross-validation [32] might be used to fully take advantage of the samples. In Eq. (17), all the possible initial states (described by

$2(n-1)$ parameters) are considered in the sample generation, which is a strict assumption. The sample numbers and training efforts are expected to grow exponentially with the dimension of the system. For specific problems, one might be interested in initial states from a subset (subspace, subsystem) of all the possible states. For example, a coherent state is determined by a complex number α. A thermal state is characterized by its temperature if its Hamiltonian is fixed. A product state $|\Psi_A\rangle \otimes |\Psi_B\rangle$ is characterized by the parameters in each subspace. Thus given a particular problem, one may parameterize the initial states in other ways to reduce the training efforts. Our initial-state-adaptive Lyapunov designs might also be used for other Lyapunov control problems or predict other continues parameters, for example, parameters in other forms of Lyapunov function.

The data and codes that support our findings are available at https://github.com/hscwork/nnlyactrl.

Acknowledgements This work is supported by the National Natural Science Foundation of China under Grant No. 11705026, 11534002, 11775048, 61475033, the China Postdoctoral Science Foundation under Grant No. 2017M611293, and the Fundamental Research Funds for the Central Universities under Grant No. 2412017QD003.

References

1. D'Alessandro, D.: Introduction to Quantum Control and Dynamics. Chapman & Hall, Boca Raton (2007)
2. Wiseman, H.M., Milburn, G.J.: Quantum Measurement and Control. Cambridge University Press, Cambridge (2009)
3. Zhang, J., Liu, Y.-X., Wu, R.-B., Jacobs, K., Nori, F.: Quantum feedback: theory, experiments, and applications. Phys. Rep. **679**, 1–60 (2017)
4. Glaser, S.J., Boscain, U., Calarco, T., Koch, C.P., Köckenberger, W., Kosloff, R., Kuprov, I., Luy, B., Schirmer, S., Schulte-Herbrüggen, T., Sugny, D., Wilhelm, F.K.: Training Schrödinger's cat: quantum optimal control. Eur. Phys. J. D **69**, 279 (2015)
5. Machnes, S., Sander, U., Glaser, S.J., de Fouquières, P., Gruslys, A., Schirmer, S., Schulte-Herbrüggen, T.: Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework. Phys. Rev. A **84**, 022305 (2011)
6. Gough, J.E., Belavkin, V.P.: Quantum control and information processing. Quantum Inf. Process. **12**, 1397–1415 (2013)
7. Vettori, P.: On the convergence of a feedback control strategy for multilevel quantum systems. In: Proceedings of the Mathematical Theory of Networks and Systems Conference (2002)
8. Grivopoulos, S., Bamieh, B.: Lyapunov-based control of quantum systems. In: Proceedings of the 42nd IEEE International Conference on Decision and Control, Maui, Hawaii USA, pp. 434–438 (2003)
9. Mirrahimi, M., Rouchon, P., Turinici, G.: Lyapunov control of bilinear Schrödinger equations. Automatica **41**, 1987–1994 (2005)
10. Kuang, S., Cong, S.: Lyapunov control methods of closed quantum systems. Automatica **44**, 98–108 (2008)
11. Wang, X., Schirmer, S.G.: Analysis of Lyapunov method for control of quantum states. IEEE Trans. Autom. Control **55**(10), 2259–2270 (2010)
12. Hou, S.C., Khan, M.A., Yi, X.X., Dong, D., Petersen, I.R.: Optimal Lyapunov-based quantum control for quantum systems. Phys. Rev. A **86**, 022321 (2012)
13. Wang, L.C., Hou, S.C., Yi, X.X., Dong, D., Petersen, I.R.: Optimal Lyapunov quantum control of two-level systems: convergence and extended techniques. Phys. Lett. A **378**, 1074 (2014)
14. Zhao, S., Lin, H., Xue, Z.: Switching control of closed quantum systems via the Lyapunov method. Automatica **48**(8), 1833–1838 (2012)

15. Kuang, S., Dong, D., Petersen, I.R.: Rapid Lyapunov control of finite-dimensional quantum systems. Automatica **81**, 164–175 (2017)
16. Silveira, H.B., da Silva, P.S.P., Rouchon, P.: Quantum gate generation for systems with drift in U(n) using Lyapunov–LaSalle techniques. Int. J. Control **89**(12), 2466–2481 (2016)
17. Li, W., Li, C., Song, H.: Quantum synchronization in an optomechanical system based on Lyapunov control. Phys. Rev. E **93**, 062221 (2016)
18. Shi, Z.C., Wang, L.C., Yi, X.X.: Preparing entangled states by Lyapunov control. Quantum Inf. Process. **15**, 4939–4953 (2016)
19. Shi, Z.C., Zhao, X.L., Yi, X.X.: Preparation of topological modes by Lyapunov control. Sci. Rep. **5**, 13777 (2015)
20. Hou, S.C., Wang, L.C., Yi, X.X.: Realization of quantum gates by Lyapunov control. Phys. Lett. A **378**(9), 699–704 (2014)
21. Yi, X.X., Huang, X.L., Wu, C., Oh, C.H.: Driving quantum systems into decoherence-free subspaces by Lyapunov control. Phys. Rev. A **80**, 052316 (2009)
22. Amini, H., Somaraju, R.A., Dotsenko, I., Sayrin, C., Mirrahimi, M., Rouchon, P.: Feedback stabilization of discrete-time quantum systems subject to non-demolition measurements with imperfections and delays. Automatica **49**(9), 2683–2692 (2013)
23. Ge, S.S., Vu, T.L., Lee, T.H.: Quantum measurement-based feedback control: a nonsmooth time delay control approach. SIAM J. Control Optim. **50**(2), 845–863 (2012)
24. Sayrin, C., Dotsenko, I., Zhou, X., Peaudecerf, B., Rybarczyk, T., Gleyzes, S., Rouchon, P., Mirrahimi, M., Amini, H., Brune, M., et al.: Real-time quantum feedback prepares and stabilizes photon number states. Nature **477**, 73–77 (2011)
25. Dotsenko, I., Mirrahimi, M., Brune, M., Haroche, S., Raimond, J.M., Rouchon, P.: Quantum feedback by discrete quantum nondemolition measurements: towards on-demand generation of photon-number states. Phys. Rev. A **80**, 013805 (2009)
26. Wang, X., Schirmer, S.G.: Entanglement generation between distant atoms by Lyapunov control. Phys. Rev. A **80**, 042305 (2009)
27. Dong, D., Petersen, I.R.: Sliding mode control of two-level quantum systems. Automatica **48**(5), 725–735 (2012)
28. Shi, Z.C., Zhao, X.L., Yi, X.X.: Robust state transfer with high fidelity in spin-1/2 chains by Lyapunov control. Phys. Rev. A **91**, 032301 (2015)
29. Shi, Z.C., Hou, S.C., Wang, L.C., Yi, X.X.: Preparation of edge states by shaking boundaries. Ann. Phys. **373**, 286–297 (2016)
30. Ran, D., Shi, Z.-C., Song, J., Xia, Y.: Speeding up adiabatic passage by adding Lyapunov control. Phys. Rev. A **96**, 033803 (2017)
31. Li, C., Song, J., Xia, Y., Ding, W.: Driving many distant atoms into high-fidelity steady state entanglement via Lyapunov control. Opt. Express **26**, 951–962 (2018)
32. Alpaydin, E.: Introduction to Machine Learning, 2nd edn. MIT Press, Cambridge (2010)
33. Haykin, S.S.: Neural Networks and Learning Machines, 3rd edn. Pearson, New Jersey (2009)
34. Magesan, E., Gambetta, J.M., Córcoles, A.D., Chow, J.M.: Machine learning for discriminating quantum measurement trajectories and improving readout. Phys. Rev. Lett. **114**, 200501 (2015)
35. Mills, K., Spanner, M., Tamblyn, I.: Deep learning and the Schrödinger equation. Phys. Rev. A **96**, 042113 (2017)
36. Melnikov, A.A., Nautrup, H.P., Krenn, M., Dunjko, V., Tiersch, M., Zeilinger, A., Briegel, H.J.: Active learning machine learns to create new quantum experiments. Proc. Natl. Acad. Sci. **115**(6), 1221–1226 (2018)
37. Torlai, G., Mazzola, G., Carrasquilla, J., Troyer, M., Melko, R., Carleo, G.: Neural-network quantum state tomography. Nat. Phys. **14**, 447–450 (2018)
38. Carleo, G., Troyer, M.: Solving the quantum many-body problem with artificial neural network. Science **355**, 602–906 (2017)
39. Deng, D.-L.: Machine learning detection of bell nonlocality in quantum many-body systems. Phys. Rev. Lett. **120**, 240402 (2018)
40. Gao, J., Qiao, L.-F., Jiao, Z.-Q., Ma, Y.-C., Hu, C.-Q., Ren, R.-J., Yang, A.-L., Tang, H., Yung, M.-H., Jin, X.-M.: Experimental machine learning of quantum states. Phys. Rev. Lett. **120**, 240501 (2018)
41. Zahedinejad, E., Ghosh, J., Sanders, B.C.: Designing high-fidelity single-shot three-qubit gates: a machine-learning approach. Phys. Rev. Appl. **6**, 054005 (2016)

42. Mavadia, S., Frey, V., Sastrawan, J., Dona, S., Biercuk, M.J.: Prediction and real-time compensation of qubit decoherence via machine learning. Nat. Commun. **8**, 14106 (2017)
43. August, M., Ni, X.: Using recurrent neural networks to optimize dynamical decoupling for quantum memory. Phys. Rev. A **95**, 012335 (2017)
44. Yang, X.-C., Yung, M.-H., Wang, X.: Neural-network-designed pulse sequences for robust control of singlet-triplet qubits. Phys. Rev. A **97**, 042324 (2018)
45. Specht, D.F.: A general regression neural network. IEEE Trans. Neural Netw. **2**(6), 568–576 (1991)
46. Leung, M.T., Chen, A.S., Daouk, H.: Forecasting exchange rates using general regression neural networks. Comput. Oper. Res. **27**, 1093–1110 (2000)
47. Li, C., Bovik, A.C., Wu, X.: Blind image quality assessment using a general regression neural network. IEEE Trans. Neural Netw. **22**(5), 793–799 (2011)
48. Liu, J., Bao, W., Shi, L., Zuo, B., Gao, W.: General regression neural network for prediction of sound absorption coefficients of sandwich structure nonwoven absorbers. Appl. Acoust. **76**, 128–137 (2014)
49. Panda, B.N., Bahubalendruni, M.R., Biswal, B.B.: A general regression neural network approach for the evaluation of compressive strength of FDM prototypes. Neural Comput. Appl. **26**, 1129–1136 (2015)
50. Życzkowski, K., Sommers, H.-J.: Induced measures in the space of mixed quantum states. J. Phys. A Math. Gen. **34**, 7111 (2001)