Q-learning optimized diagonal recurrent neural network control strategy for brushless direct current motors

Huangshui Hu¹, Tingting Wang¹, Hongzhi Wang² and Chuhang Wang³

Abstract
In order to improve the working stability of brushless direct current motors (BLDCM), a diagonal recursive neural network (DRNN) control strategy based on Q-learning algorithm is proposed in this paper which is called as Q-DRNN. In Q-DRNN, DRNN iterates over the output variables through a unique recursive loop in the hidden layer, and its key weight is optimized to speed up the iteration. Moreover, an improved Q-learning algorithm is introduced to modify the weight momentum factor of DRNN, which makes DRNN have the ability of learning and online correction so as to make the BLDCM achieve better control effect. In MATLAB/Simulink environment, Q-DRNN is tested and compared with other popular control methods in terms of speed and torque response under different operating conditions, and the results show that Q-DRNN has better adaptive and anti-interference ability as well as stronger robustness.

Keywords
DRNN, Q-learning algorithm, optimal control strategy, key weight, BLDCM

Date received: 17 May 2020; accepted: 20 August 2020

Handling Editor: James Baldwin

Introduction
Due to its simple structure, high efficiency, long service life and low noise, BLDCM has been widely used in national defense, aerospace, robotics, and so on.¹⁻⁵ BLDCM plays an important role in the modern motor control system. Therefore, it has important practical significance and application prospect to study the control strategy of BLDCM with fast response, strong regulation ability and high control accuracy.

PID control is one of the earliest linear control strategies with a long history. It is still the most commonly used control algorithm in industrial control system. In common, P (proportion), I (integral) and D (differential) can be combined into many kinds of controllers. In Jigang et al.,⁶ Wei et al.,⁷ Premkumar and Manikandan,⁸ and Gundogdu and Komurgoz,⁹ PI, PD and PID controllers are respectively used to control the speed of BLDCM. However, the uncertainties, nonlinearity and parameters manual adjustment of typical PI, PD or PID controllers make it difficult to determine the appropriate gain to achieve the optimal performance of the control system.

With the development of computer technology and intelligent control theory, various types of intelligent

¹College of Computer Science and Engineering, Changchun University of Technology, Changchun, China
²College of Computer Science and Engineering, Jilin University of Architecture and Technology, Changchun, China
³College of Computer Science and Technology, Changchun Normal University, Changchun, China

Corresponding author:
Hongzhi Wang, College of Computer Science and Engineering, Jilin University of Architecture and Technology, Changchun 130114, China. Email: 601630419@qq.com

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
algorithms optimized PID controllers have been proposed in the last decades. In Ahmed and Rajoriya, and Afraasiabi and Yazdi, the sliding mode controllers are proposed to realize the speed control of the motor. However, the chattering problem inevitably exists in the sliding mode control, which results in the overall performance degradation of the system. In Ramya et al., and Premkumar and Manikandan, fuzzy logic control algorithms are proposed, but the algorithms rely on expert knowledge rule base. In Demirtas, genetic algorithm is used to optimize the gain of PI controller. However, the initial population of genetic algorithm may not be easy to determine. In Khubalkar et al., and Badar et al., particle swarm optimization (PSO) is adopted, but the presented algorithms have some problems such as slow convergence speed and local optimization. So many methods are proposed to simplify or improve them. The algorithms based on neural network have shown promising results.

The PID gain updating algorithm based on neural network has been successfully applied to the control of servo motor, computerized numerical control machine tool, etc. In Xia et al., a single neuron PI controller is designed for the control system of BLDCM. In Kumar et al., a kind of PID controller based on neural network is proposed, which consists of a hybrid local recurrent neural network including at most three hidden nodes so as to form a structure similar to PID. The controller is easy to realize, but its number of parameters is difficult to be determined. Moreover, the training algorithm based on gradient descent is a time-consuming process. In Premkumar and Manikandan, an Online fuzzy supervisory learning method based on RBFNN (Online-RBFNN) for BLDCM is presented. The fuzzy PID supervisory algorithm is applied to RBFNN. Compared with PID controller, Online-RBFNN has significantly smaller oscillation phenomenon and steady-state error, but its selection of fuzzy rules still needs to be optimized. In Premkumar et al., aiming at the speed control problem of BLDCM, antlion algorithm optimized fuzzy PID supervised online recurrent fuzzy neural network based controller is proposed. The learning parameters of supervised online recurrent fuzzy neural network controller are optimized by using antlion algorithm. In Kang et al., PSO is used to initialize the weights of the adaptive PID neural network controller, and the improved gradient descent algorithm is used to adjust the parameters of the PID neural network. The disadvantage of this method is that PSO takes a long time to initialize the PID neural network. Then, GA-PSO algorithm is used to optimize the online adaptive neuro fuzzy inference system (ANFIS) for the speed control of BLDCM in Premkumar and Manikandan. The hybrid GA-PSO algorithm is used to optimize the learning rate, forgetting factor and the maximum decreasing momentum constant of the online ANFIS controller under different torque conditions of BLDCM, and the effectiveness of the method has been verified by simulation experiments. Moreover, the implementation of these controllers are relatively complicated in practical, so FPGAs (Field Programmable Gate Array) are usually used as high-speed hardware to run the controllers because of their offered parallel processing abilities as well as short execution time compared with traditional microcontrollers and DSPs (Digital Signal Processor).

In recent years, machine learning (ML) has become a popular topic. Reinforcement learning is a branch of ML whose purpose is to find the optimal strategy that must be followed in the transition of some states so as to maximize the total return of the selected operation. Q-learning is one of the most popular and successful reinforcement learning methods. In Sarigul and Avci, a learning system with general recurrent neural network topology based on Nadaraya-Watson kernel adopts Q-learning method to evaluate a fast and effective behavior selection strategy for reinforcement learning problems, and its effectiveness is verified. However, the Nadaraya-Watson kernel regression based recurrent neural network needs to deal with all samples, which deteriorates the overall control effect in case of high-dimensional samples.

Combined with the strong search ability of Q-learning and the advantages of DRNN such as recursive loop structure, dynamic mapping ability and adaptability to time-varying, this paper presents a control strategy Q-DRNN to improve the performance of BLDCM. Q-DRNN optimizes the key weight in normal DRNN, and introduces Q-learning to modify the weight momentum term factor so as to achieve better control effect. Q-DRNN has the ability of learning and online correction, which enhances the anti-interference (Anti-interference ability refers to the ability of the control system to maintain some characteristics under the condition of load and speed mutation.) and robustness of the system. In order to verify the effectiveness of Q-DRNN, its performance is tested and compared with neural network PID (NNPID) control method. Online fuzzy supervisory learning method based on RBFNN (Online-RBFNN), antlion algorithm optimized fuzzy PID supervised online recurrent fuzzy neural network (ALO-RFNN) based control method, and Q-learning optimized regression neural network (QLRNN) control method under different operating conditions.

The rest of this paper is organized as follows: Section 2 introduces the mathematical model of BLDCM, Section 3 describes the proposed Q-DRNN in detail,
Section 4 gives the simulation results, and Section 5 concludes this paper.

The mathematical model of BLDCM

The mathematical model is the basis of the performance analysis and control system design of the BLDCM. The differential equation model of the two-stage three-phase BLDCM is established in this section.

1. Voltage equation

According to the knowledge of motor science, the voltage equation of stator three-phase winding can be expressed as:

\[
\begin{bmatrix}
ua \\
ub \\
uc \\
\end{bmatrix} =
\begin{bmatrix}
R & 0 & 0 \\
0 & R & 0 \\
0 & 0 & R \\
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b \\
i_c \\
\end{bmatrix} +
\begin{bmatrix}
L - M & 0 & 0 \\
0 & L - M & 0 \\
0 & 0 & L - M \\
\end{bmatrix}
\frac{d}{dt}
\begin{bmatrix}
i_a \\
i_b \\
i_c \\
\end{bmatrix} +
\begin{bmatrix}
e_a \\
e_b \\
e_c \\
\end{bmatrix}
\]

where \(u_a, u_b, u_c\) are the phase voltage (V), \(R\) is the stator winding resistance (Ohm), \(i_a, i_b, i_c\) are the phase current (Ampere) of the motor, \(L\) and \(M\) represent the self-inductance of the motor winding and the mutual inductance between the stator winding. \(e_a, e_b, e_c\) represent the reverse electromagnetic force of each phase, in volts.

2. Torque equation

The torque equation is:

\[
T_e = \left(e_{aia} + e_{bhb} + e_{cic}\right) \frac{1}{\Omega}
\]

where \(T_e\) is the electromagnetic torque, \(\Omega\) is the mechanical angular speed of the motor.

3. Equation of motion

\[
T_e = B\Omega + J \frac{d\Omega}{dt} + T_L
\]

where \(T_L\) is load torque, \(J\) is rotor moment of inertia, and \(B\) is viscous friction coefficient.

4. Equation of state

The voltage equation can be rewritten as a state equation:

\[
\frac{d}{dt}
\begin{bmatrix}
i_a \\
i_b \\
i_c \\
\end{bmatrix} =
\begin{bmatrix}
1/L - M & 0 & 0 \\
0 & 1/L - M & 0 \\
0 & 0 & 1/L - M \\
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b \\
i_c \\
\end{bmatrix}
\]

The equivalent circuit diagram of the voltage equation is shown in Figure 1:

Based on the BLDCM model shown in Figures 1 and 2 shows the BLDCM speed control system.

The system consists of three-phase voltage source PWM inverter, three-phase BLDCM, controller, logic switch and motor measurement sensor.

The proposed control strategy

The control strategy proposed in this paper uses Q-learning algorithm to modify the weight momentum term fact of DRNN so as to improve the control effectiveness of BLDCM, which consists of three aspects: the design of DRNN, Q-learning algorithm and Q-DRNN controller. The detailed description of the control strategy is given as follows.

Design of DRNN

Consider the BLDCM as a discrete nonlinear system

\[
y(k + 1) = \phi(y(k), u(k))
\]

where \(u(k)\) and \(y(k)\) are the control input and speed output of the system respectively, \(\phi(\cdot)\) is a nonlinear function, and let, s say that \(\phi(\cdot)\) can take the derivative of the control variable \(u\).

The structure of the direct intelligent control system composed of a diagonal recurrent neural network is shown in Figure 3. Similar to feed forward neural network, the diagonal recurrent neural network is composed of input layer, hidden layer and output layer. The difference is that each neuron in the hidden layer
has its own recursive loop. In Figure 3, the input and output of DRNN controller are \(x_i (i = 1, 2, 3)\) and \(u_k (k)\) respectively.

\[
u_k (k) = O_k (k) = f_2 \left[ \sum_{j=1}^{q} w_{jk} \times O_j (k) - \theta_k \right]
\]

\[
O_j (k) = f_1 (I_j (k))
\]

\[
I_j (k) = \sum_{i=1}^{p} w_{ij} \times x_i + w_{jj} \times O_j (k-1) - \theta_j
\]

where, \(w_{ij}, w_{jj}, w_{jk}, \theta_j, \theta_k\) are the weights between the neurons in the input layer and the hidden layer, the weights of the self-feedback loop in the hidden layer, the weights between the neurons in the hidden layer and the output layer, the offsets of the neurons in the \(j\) hidden layer and the offsets of the neurons in the \(k\) output layer respectively, \(p\) is the number of input neurons and \(q\) is the number of neurons in the hidden layer. \(f_1(x), f_2(x)\) are the activation functions of the hidden layer and the output layer, and the sigmoid function is as follows

\[
f_1 (x) = \tanh (x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]

\[
f_2 (x) = \frac{\tanh (x) + 1}{2} = \frac{e^x}{e^x + e^{-x}}
\]

The system input can be expressed as

\[
u(k) = K \cdot u_k (k)
\]

where \(K\) is the gain coefficient.

**Q-learning algorithm**

Q-learning algorithm is an iterative incremental online learning method. It enables the agent to select the optimal action sequence in the Markov decision-making process through the interaction with the external environment. Figure 4 shows the principle of the Q-learning algorithm.

The value function of Q-learning is defined as follows:

\[
Q(s_k, a_k) = r_k + \gamma \max_a Q(s_{k+1}, a)
\]
where \( \gamma \) is the discount factor, \( s \) is the state, \( a \) is the action, and \( r \) is the reward and punishment signal. The agent receives the input state \( s_k \) in the environment and outputs the corresponding action \( a_k \) through the internal reasoning mechanism. Under the action of \( a_k \), the environment becomes a new state \( s_{k+1} \). At the same time, it generates real-time reward and punishment signal \( r_{k+1} \) for agent. \( r_{k+1} \) is the evaluation of agent action \( a_k \) under the environment state \( s_k \). If the behavior strategy gets a positive return and gets a reward from the environment, the agent’s tendency to choose action will increase, otherwise the tendency will decrease. The iterative formula of \( Q \) value optimization is as follows:

\[
Q^{k+1}(s_k, a_k) = Q^k(s_k, a_k) \\
+ \alpha \left[ r_k + \gamma \max Q^k(s_{k+1}, a) - Q^k(s_k, a_k) \right]
\]

where: \( Q^k \) represents the \( k \)-th iterative value of the optimal value function \( Q^* \), \( \alpha \)(0 < \( \alpha \) < 1) is the learning factor, which controls the update speed of the action. The smaller the \( \alpha \) value is, the larger the search space of the algorithm is, and the better the stability of the algorithm is. Q-learning algorithm always selects the action with the highest \( Q \) value in the current state, which is called greedy strategy \( \pi^* \) given as follows:

\[
\pi^*(s) = \arg \max Q^k(s, a)
\]

The disadvantage of Q-learning is that it is easy to converge to partial optimum, but not to global optimum. This is because the final \( Q \) value is not the optimal solution, so there are errors. In order to eliminate the errors, a tracking algorithm\(^{25}\) based on probability distribution is used to construct a dynamic selection strategy. In this strategy, the probability of each action selected in the initial state is equal, but with the continuous iteration of the action value function, the higher the probability of action selected in the higher \( Q \) value is, and the action with higher probability is taken as the initial action in the next moment. The probability iteration formulas of this strategy are shown as follows:

\[
P_{s}^{k+1}(a) = P_{s}^{k}(a) + \beta(1 - P_{s}^{k}(a))
\]

where the value of \( \beta(0 < \beta < 1) \) represents the speed of action search. It can be seen that the closer the value of \( \beta \) is to 1, the closer the current action strategy is to greedy strategy. \( P_{s}^{k}(a) \) represents the probability of selecting action \( a \) under state \( s \) in the \( k \)-th iteration. If the number of iterations explored and utilized reaches a certain critical value, \( Q^* \) will converge to the optimal value function \( Q^* \).

**Design of Q-DRNN controller**

In order to improve the control performance of BLDCM, the Q-DRNN controller is designed based on the combination of the strong search ability of Q-learning and the advantages of DRNN, such as its own recursive loop structure, dynamic mapping ability and adaptability to time-varying. The control structure is shown in Figure 5.

In this paper, the iteration of Q value table is mainly carried out by the following formula:

\[
Q(s_k, a_k) = \gamma \max Q(s_{k+1}, a) + R(s_{k+1}, s_k)
\]

where \( \gamma \) is the discount factor, \( Q(s_{k+1}, a) \) is the maximum value in \( s_{k+1} \) state, and \( R(s_{k+1}, s_k) \) is the return value of the reward and punishment function.

For the given system, its performance index function is defined as

\[
J = \sum_{k=1}^{m} \sum_{k=1}^{m} [e^T(k+1)Ce(k+1) + u_k^T(k)D_{uk}(k)]
\]

where \( C \) and \( D \) are weight matrix, \( m \) is the upper limit of the number of iterations, \( e(k) = Y_d(k) - y(k) \) is system error, and \( u_k(k) \) is control output of Q-DRNN controller.

The connection weights \( W_{\theta}(k) \) and \( W_{\beta}(k) \) of Q-DRNN are adjusted by the steepest gradient descent method with momentum term:

\[
W_{\theta}(k+1) = W_{\theta}(k) - \eta \left[ 1 - \xi(k) \right] \frac{\partial e}{\partial W_{\theta}(k)} + \xi(k) \frac{\partial e}{\partial W_{\theta}(k)}
\]

\[
W_{\beta}(k+1) = W_{\beta}(k) - \eta \left[ 1 - \xi(k) \right] \frac{\partial e}{\partial W_{\beta}(k)} + \xi(k) \frac{\partial e}{\partial W_{\beta}(k)}
\]

where \( \eta(\eta>0) \) is the learning rate and \( \xi(k)(0 \leq \xi(k) < 1) \) is the momentum factor. The introduction of momentum term into DRNN is essentially equivalent to damping term, which reduces the oscillation trend of learning.
process and improves convergence. The momentum factor is usually set as a constant, which cannot be adjusted adaptively according to the system change. Therefore, in this paper, the momentum factor is set as a variable varying with the number of iterations k to better adjust the weight of DRNN.

In the process of Q-learning, the momentum factor correction term \( \Delta r \) of the weight is taken as the action set, and the input term \( x_i (i = 1, 2, 3) \) of the Q-DRNN controller is taken as the state set. The momentum factor is defined by the method based on natural logarithm decay. The correction formula of momentum factor \( \xi \) is:

\[
\xi(k) = \begin{cases} \xi_0 \Delta r, & 0 \leq \Delta r < 1 \\ \xi_0 \Delta r - \lceil \Delta r \rceil, & \Delta r \geq 1 \end{cases}
\]

where \( \xi_0 (0 < \xi_0 < 1) \) is the initial value of momentum factor. When the error of \( e(k) \) is closer to 0, \( \Delta r \) is closer to 1, and the correction of \( \xi \) is smaller.

Similarly, \( \frac{\partial J_k}{\partial W_{ij}(k)} \) and \( \frac{\partial J_k}{\partial \theta_i(k)} \) can be calculated. \( \frac{\partial J_k}{\partial W_{ij}(k)} \) can be obtained by deriving \( W_{ij}(k) \), \( W_{ji}(k) \) and \( W_{ji}(k) \) from formula (6).

For \( \frac{\partial e(k + 1)}{\partial u_i(k)} \) calculate:

\[
\frac{\partial e(k + 1)}{\partial u_i(k)} = \frac{\partial [Y_d(k) - y(k + 1)]}{\partial u_i(k)}
\]

\[
= -\frac{\partial y(k + 1)}{\partial u_i(k)} = -\frac{\partial \phi(y(k), u_i(k))}{\partial u_i(k)}
\]

(25)

It can be seen from equation (5) that for step \( k \) control \( u_i(k) \), it first acts on state \( y_a(k + 1) \), then controls \( y_a(k + 2) \) in step \( (k + 2) \), so that \( n \) steps are passed to control \( y_1(k + n) \). Therefore, it is necessary to stabilize the output \( y(k) \) of equation (5) at the set point \( Y_d(k) \), that is, \( e(k) = Y_d(k) - y(k) \rightarrow 0 \). In this paper, an approximate method is used to solve \( \frac{\partial e(k + 1)}{\partial u_i(k)} \), the formula is as follows:

\[
\left[ \frac{\partial y(k + 1)}{\partial u_i(k)} \right]_j = \begin{cases} \left( y_j(k + 1) - y_j(k) \right) , & u_j(k) = u_j(k - 1) \\ \left( y_j(k + 1) - y_j(k) \right) , & u_j(k) \neq u_j(k - 1) \end{cases}
\]

(26)

For all Q-learning, there is always a reward and punishment function. Here, the introduced reward and punishment function is related to the input of the system. Since the ideal goal of system error \( e(k) \) is zero, the smaller the error is in the optimization process, which means that the learning direction is the reward direction and can continue to adjust in this direction, the larger the error is, which means that the learning direction is the punishment direction, and should be
adjusted in the opposite direction. The reward function \( R(k) \) can be designed as the square of the error between the actual value of the system error integral and the target value, namely:

\[
R(k) = -\left[ \int_{k_1}^{k_2} |e(k)| dk - 0 \right]^2
\]

where \( k_1, k_2 \) represents the lower limit and the upper limit of the integral.

After determining the reward and punishment function \( R(k) \), input state set and control action set, the online self-learning and dynamic optimization of Q-DRNN controller can be carried out according to the iterative formula of the algorithm. The steps are as follows:

1. Select the structure of Q-DRNN in advance, that is, select the number of input layer nodes and hidden layer nodes, randomly generate the initial values \( W_{ij}(0) \), \( W_{ik}(0) \) and \( W_{kk}(0) \) of the weighting coefficient between each layer, select the learning rate \( \eta \) and the initial value \( \xi_0 \) of momentum factor, initialize all parameters of \((s, a)\), observe the current state \( S(0) \), make \( k = 0 \).
2. After sampling, \( Y_d(k) \) and \( y(k) \) are obtained, and \( e(k) = Y_d(k) - y(k) \) is calculated; action \( a(k) \) is selected from action set by action probability distribution.
3. Normalize \( e(k), e(k) - e(k - 1) \) and \( e(k) - 2e(k - 1) + e(k - 2) \) as the input of Q-DRNN, and observe the next state \( S(k + 1) \).
4. According to formula (6), the output of Q-DRNN output layer is calculated, and the control law calculated at this time is \( u(k) \).
5. A reward signal \( R(k) \) is obtained from equation (27), and the Q value of Q-DRNN in this state is calculated.
6. Calculate the greedy action \( a_g(k) \) according to equation (15).
7. Modify the weights \( W_{ij}(k) \), \( W_{ik}(k) \) and \( W_{jk}(k) \) of Q-DRNN.
8. Update the action probability distribution according to equation (16–18).

Let \( k = k + 1 \), return (2), until \( Q^k \) converges to the optimal value function \( Q^* \).

### Simulation results

In order to further verify the performance of Q-DRNN in the control system of BLDCM, the system model is established by using Matlab/Simulink toolbox, and the controllers under different operating conditions are simulated and compared with NNPID, Online-RBFNN, ALO-RFNN and QLRNN. The specifications of BLDCM are shown in Table 1. The given input speed of BLDCM is 3000 r/min, the number of DRNN nodes is 3-6-1, the number of hidden nodes is determined by a large number of simulation experiments, the initial weight is the random number of \([-0.5,0.5]\) interval, the learning rate \( \eta = 0.05, \xi_0 = 0.01 \). The discount factor \( \gamma = 0.9 \), learning factor \( \alpha = 0.1 \), \( \beta = 0.5 \) for Q-learning.

### Speed and torque response with the absence of load

First, the tests are performed under the condition of absence of load, the sample time of the control system is \( T_s = 0.5 \times 10^{-5} \) s. The speed response comparison curve and torque response comparison curve are shown in Figure 6(a) and (b) respectively.

From Figure 6(a), it can be seen that the peak overshoot is 105.66, 117.70 and 65.31 r/min for the ALO-RFNN controller, the Online-RBFNN controller and the NNPID controller during the transient period, while no obvious overshoot occurs for QLRNN and Q-DRNN. During the steady-state period, the Online-RBFNN control algorithm has the longest setting time 0.10 s and Q-DRNN has the shortest setting time 0.03 s. Moreover, Q-DRNN has the lowest steady-state error 0.11 r/min. From the torque response curve in Figure 6(b), it can be seen that the Q-DRNN controller has the highest amplitude, but it recovers to a stable state at the fastest speed. Although the amplitude of the QLRNN controller, the ALO-RFNN controller and the Online-RBFNN controller is lower than that of the Q-DRNN controller, their chattering phenomenon is obvious, the setting time is longer, and the setting time of the NNPID controller is the longest. Specific performance indexes are shown in Table 2. Based on the comparison and analysis of the above controllers, it can be concluded that the control effect of Q-DRNN controller is better than that of other controllers.

### Speed and torque response with load

Next, the control system of BLDCM is operated with sudden change in load conditions to determine the
advantages of Q-DRNN controller. The input speed of the control system is still 3000 r/min, the sample time of the control system is also $T_s = 0.5 \times 10^{-5}$ s, and a load of 2 Nm is applied to the system at 0.2 s. Figures 7(a) and 8(b) respectively show the speed response curve and torque response curve under the condition of various loads.

As can be seen from Figure 7(a) that when the load is applied at 0.2 s, ALO-RFNN controller, Online-RBFNN controller and NNPID controller have obvious oscillation phenomenon, and only after a period of time can they recover stable. QLRNN controller also has some fluctuations, and finally it recovers stable. The minimum steady-state error, the minimum peak undershoot, and overshoot of Q-DRNN are 0.51 r/min, 0.83 r/min, and 0, respectively. Moreover, the recovery time of Q-DRNN is 0.02 s, which is the shortest. As can be seen from Figure 7(b) that at 0.2 s, the ALO-RFNN controller has the largest amplitude and only after a period of time can it recover to be stable, and the Q-DRNN controller has the smallest amplitude and is closer to 2 Nm. Therefore, Q-DRNN controller is better than other controllers. This also strongly reflects the anti-interference and robustness of the Q-DRNN Controller under the condition of various loads. The performance indexes in this condition are shown in Table 3.

### Speed and torque response with various set speed

Finally, the control performance of Q-DRNN controller is verified under the condition of sudden change of speed, the sample time of the control system is $T_s = 0.5 \times 10^{-6}$ s. When the system runs to 0.2 s, the speed is reduced from 3000 r/min to 2500 r/min. The speed response comparison curve and torque response comparison curve in this state are shown in Figure 8(a) and (b) respectively.

As can be seen from Figure 8(a) that when the speed changes suddenly at 0.2 s, oscillation occurs for ALO-RFNN, Online-RBFNN and NNPID controllers. QLRNN controller has no obvious undershoot, after a period of time, it is stable. Q-DRNN controller has the minimum recovery time 0.03 s and the minimum steady-state error 0.56 r/min. As can be seen from Figure 8(b) that at 0.2 s, the amplitude of Q-DRNN controller is large, but it recovers to a stable state at the fastest speed. Although the amplitude of other

---

**Table 2.** The performance indexes under the condition of no load.

| Controllers | Speed response parameters | Torque response parameters |
|-------------|----------------------------|---------------------------|
|             | Peak overshoot (r/min)     | Maximum amplitude (N·m)   |
|             | Setting time (s)           | Settling time (s)          |
|             | Steady-state error (r/min)|                           |

| Controls  | Peak overshoot | Setting time | Steady-state error | Maximum amplitude | Settling time |
|-----------|----------------|--------------|--------------------|-------------------|---------------|
| Q-DRNN    | -              | 0.03         | 0.11               | 93.51             | 0.03          |
| QLRNN     | -              | 0.06         | 1.73               | 31.12             | 0.06          |
| ALO-RFNN  | 105.66         | 0.09         | 0.15               | 32.33             | 0.04          |
| Online-RBFNN | 117.70    | 0.10         | 0.26               | 29.72             | 0.05          |
| NNPID     | 65.31          | 0.08         | 0.21               | 27.02             | 0.08          |

---

![Figure 6. Speed and torque response with no load: (a) comparison of speed response and (b) comparison of torque response.](image-url)
controllers is lower than that of Q-DRNN controller, the stability time is longer. Combining the above performance indexes, it can be concluded that the control effect of Q-DRNN controller is better than that of other controllers. The performance indexes in this condition are shown in Table 4.

From the above three groups of simulation results under different control conditions, it can be concluded

| Controllers   | Speed response parameters | Torque response parameters |
|---------------|---------------------------|----------------------------|
|               | Peak overshoot (r/min)    | Peak undershoot (r/min)    | Recovery time (s) | Steady-state error (r/min) | Maximum amplitude (N·m) |
| Q-DRNN        | -                         | 0.83                       | 0.02              | 0.51                        | 0.60                     |
| QLRNN         | -                         | 4.43                       | 0.04              | 2.62                        | 0.84                     |
| ALO-RFNN      | 6.68                      | 50.51                      | 0.06              | 0.65                        | 1.74                     |
| Online-RBFNN  | 7.38                      | 53.35                      | 0.07              | 0.69                        | 1.73                     |
| NNPID         | 4.85                      | 54.53                      | 0.08              | 0.73                        | 1.64                     |

Table 3. The performance indexes under the condition of various loads.
that all controllers can correctly track the set speed. However, the proposed Q-DRNN controller is superior to other controllers in terms of steady-state error, setting time and recovery time. Therefore, the Q-DRNN controller is proved to have better adaptive ability, anti-interference ability and strong robustness.

**Conclusion**

A novel control strategy called Q-DRNN by using Q-learning optimized DRNN is proposed in this paper for BLDCM to achieve better performance of speed and torque control. Q-DRNN obtains the abilities of self-learning and online correction by modifying the weight momentum factor and optimizing the key weight of DRNN, and then the control effectiveness is greatly improved. Under no load conditions, Q-DRNN is proved to have the minimum overshoot, the shortest recovery time and the minimum steady-state error of speed response whose values are 0 r/min, 0.03 s, 0.11 r/min respectively, as well as the shortest recovery time of torque response. Under with load conditions, Q-DRNN has the minimum peak undershoot 0.83 r/min, the shortest recovery time 0.02 s, the minimum steady-state error 0.65 r/min and the minimum amplitude 0.60 r/min. Under different set speed conditions, Q-DRNN has the shortest recovery time and the minimum steady-state error compared with other controllers. Therefore, Q-DRNN is significantly superior to other control methods.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work supported by Capital construction funds in the provincial budget of Jilin development and reform commission in 2019 (grant no 2019C054-4); National nature fund project (grant no 61803044); Science and technology development project of Jilin Province (grant no 20190802005ZG); Science and technology plan project of Jilin Province (20200201009JC).

**ORCID iD**

Tingting Wang https://orcid.org/0000-0002-0684-8331

**References**

1. Feng J, Liu K and Wang Q. Scheme based on buck-converter with three-phase H-bridge combinations for high-speed BLDC motors in aerospace applications. *IET Electr Power Appl* 2018; 12(3): 405–414.
2. Meza JL, Santibanez V, Soto R, et al. Fuzzy self-tuning PID semiglobal regulator for robot manipulators. *IEEE Trans Ind Electron* 2012; 59(6): 2709–2717.
3. Yadav AK and Gaur P. Improved self-tuning fuzzy proportional–integral–derivative versus fuzzy-adaptive proportional–integral–derivative for speed control of nonlinear hybrid electric vehicles. *J Comput Nonlinear Dyn* 2016; 11(6): 061013.
4. Yadav AK and Gaur P. An optimized and improved STF-PID speed control of throttle controlled HEV. *Arab J Sci Eng* 2016; 41(9): 3749–3760.
5. Godfrey AJ and Sankaranarayanan V. A new electric braking system with energy regeneration for a BLDC motor driven electric vehicle. *Int J Eng Sci Technol* 2018; 21: 704–713.
6. Jigang H, Hui F and Jie W. A PI controller optimized with modified differential evolution algorithm for speed control of BLDC motor. *Automatika-J Control Meas Electron Comput Commun* 2019; 60(2): 135–148.
7. Wei X, Jiesheng W and Haibo W. PI controller of speed regulation of brushless DC motor based on particle swarm optimization algorithm with improved inertia weights. *Math Probl Eng* 2019; 2019(2): 1–12.
8. Premkumar K and Manikandan BV. Bat algorithm optimized fuzzy PD based speed controller for brushless direct current motor. *Int J Eng Sci Technol* 2015; 19(2): 818–840.
9. Gundogdu T and Komurgoz G. Self-tuning PID control of a brushless DC motor by adaptive interaction. *IEEE Trans Electr Electron Eng* 2014; 9(2014): 384–390.
10. Ahmed H and Rajoriy A. A hybrid of sliding mode control and fuzzy logic control using a fuzzy supervisory

| Controllers | Speed response parameters | Torque response parameters |
|-------------|--------------------------|---------------------------|
|             | Peak undershoot (r/min)  | Recovery time (s)        |
|             | Steady-state error (r/min) | Maximum amplitude (N·m)  | Recovery time (s) |
| Q-DRNN      | -                        | 0.03                      | 0.56               | -11.53               | 0.02 |
| QLRNN       | -                        | 0.07                      | 2.71               | -3.85                | 0.04 |
| ALO-RFNN    | 53.52                    | 0.06                      | 0.65               | -3.60                | 0.03 |
| Online-RFNN | 56.01                    | 0.07                      | 0.72               | -3.38                | 0.04 |
| NNPID       | 31.96                    | 0.08                      | 0.70               | -2.83                | 0.05 |

Table 4. The performance indexes for varying set speed condition.
switched system for DC motor speed control. *Turk J Elec Eng & Comp Sci* 2017; 2017(25): 1993–2004.

11. Afrasiabi N and Yazdi MH. Sliding mode controller for DC motor speed control. *Global J Sci, Eng Technology* 2013; 2013(11): 45–50.

12. Ramya A, Ahamed I and Balaji M. Hybrid self tuned Fuzzy PID controller for speed control of Brushless DC Motor. *Automatica* 2016; 57(3): 672–679.

13. Premkumar K and Manikandan BV. Fuzzy PID supervised online ANFIS based speed controller for brushless DC motor. *Neurocomputing* 2015; 157(2015): 76–90.

14. Demirtas M. Off-line tuning of a PI speed controller for a permanent magnet brushless DC motor using DSP. *Energy Convers Manag* 2011; 52(1): 264–273.

15. Khubalkar S, Junghare A, Aware M, et al. Modeling and control of a permanent-magnet brushless DC motor drive using a fractional order proportional-integral-derivative controller. *Turk J Elec Eng & Comp Sci* 2017; 25(5): 4223–4241.

16. Badar AQH, Umre BS and Junghare AS. Reactive power control using dynamic particle swarm optimization for real power loss minimization. *Int J Elec Power Energy Sys* 2012; 41(1): 133–136.

17. Kento K, Shin W and Shuichi O. Design of neural network PID controller based on E-FRIT. *Electr Eng Jpn* 2018; 205(2): 33–42.

18. Cho CN, Song YH, Lee CH, et al. Neural network based real time PID gain update algorithm for contour error reduction. *Int J Precis Eng Man* 2018; 19(11): 1619–1625.

19. Sheng PH and Thong SH. Dynamic modeling and neural network self-tuning PID control design for a linear motor driving platform. *IEEE Trans Electr Electron Eng* 2010; 5(6): 701–707.

20. Ma L and Li D. Adaptive neural networks control using barrier Lyapunov functions for DC motor system with time-varying state constraints. *Complexity* 2018; 2018(5082401): 1–9.

21. Khoshtarrregi MR, Tappe S and Altintas Y. Integrated five-axis trajectory shaping and contour error compensation for high-speed CNC machine tools. *IEEE/ASME Trans Mechatro* 2014; 19(6): 1859–1871.