© 2019 IJSRSET | Volume 6 | Issue 6 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology
DOI : https://doi.org/10.32628/IJSRSET196644

Data-driven Nonlinear MIMO ADP Method and Its Application in PMSM Control

Bowen Sui*
Lab of Intelligent Control and Computation, Shanghai Maritime University, Shanghai, 201306, China

ABSTRACT

The power system is a nonlinear, time-varying, high-dimensional system. How to carry out effective control to ensure its safer and more stable operation has been the subject of many scholars' research, and with the continuous expansion of the power system scale and randomness. With the access of stronger new energy sources, the challenges facing the security and stability of power systems are becoming more and more severe. The conventional optimal control method has certain limitations. For example, the variational method can only solve the optimal problem that the control quantity is not constrained. The maximal/minimum value principle can only solve the optimal control problem described by ordinary differential equations. Although the plan can solve the more general optimal control problem than that described by the ordinary differential equation, it is a problem of dimensionality hazard because it is a time-backward algorithm. Adaptive dynamic programming is the product of the integration of artificial intelligence and control technology. Its essence is to approximate the solution of Hamilton-Jacobi-Bellman equation by using the approximate structure of the function of the neural network. This method does not depend on the mathematical model of the controlled object, nor does it need to define the performance index accurately, and can learn online. The introduction of this method into the power system can provide a new idea for the non-linear optimal control of the power system. Based on the traditional Adaptive Dynamic Programming (ADP) algorithm, this paper proposes a data-driven nonlinear Multi-Input and Multi-Output (MIMO) adaptive dynamic programming algorithm, and applies this algorithm to Permanent Magnet Synchronous Motor (PMSM) related control. The simulation of single objective control and under-actuated control model proves that the data-driven adaptive dynamic programming method based on least squares strategy iteration has strong robustness.

Keywords : Adaptive Dynamic Programming(ADP), Data-driven, Intelligent power systems, Permanent Magnet Synchronous Motor (PMSM), Nonlinear systems, Multiple Input Multiple Output systems(MIMO).

I. INTRODUCTION

Adaptive dynamic programming (ADP) [1]-[4] is an effective method to solve the problem of dynamic programming. It is a new subject formed by the convergence of the development of artificial intelligence and control.

Adaptive dynamic programming includes techniques such as Adaptive Critic Designs [5]-[6] and Reinforcement Learning [7]. Adaptive dynamic programming was first proposed by Werbos [8]. Its core idea is to use the function approximation structure to approximate the optimal performance index and control law, so that the corresponding dynamic programming calculation results can be obtained. In the selection of the approximation

structure of the function, Werbos uses a neural network to approximate the performance index, and another neural network guides the approximation control law under the result of the previous neural network, and has obtained good results [9]. Later, Bertsekas applied this structure extensively to the optimal control of nonlinear systems [10], so adaptive dynamic programming is also called Neuro Dynamic Programming [11]. Because the neural network can not only adaptively adjust its own weight, adaptively approach the performance index function, but also give the evaluation signal to the approximation of the optimal control, it is also called adaptive dynamic programming as adaptive evaluation design (Adaptive). Critic Designs, ACDs) [5]-[12], or Approximate Dynamic Programming [3]-[13]. In 1974, Werbos elaborated on the idea of neural network back-propagation algorithms and neural network-based intelligent control in his doctoral thesis [14]. In 1977, Werbos proposed the concepts of "Heuristic Dynamic Programming (HDP)" and "Dual Heuristic Programming (DHP)". Based on dynamic programming, two neural networks were proposed. The structure approaches the performance indicators and control laws to solve the problem of "dimensionality disaster" [8]. In 1992, Werbos proposed the "Action Dependent Heuristic Dynamic Programming (ADHDP)" and "Action Dependent Dual Heuristic Programming (ADDHP)" forms, through the development of dynamic programming algorithms. An adaptive dynamic programming algorithm using time-forward dynamic programming is proposed, and a method for solving complex numerical problems is proposed through "adaptive dynamic programming" [9]. Werbos' groundbreaking research work laid the foundation for ADP and made a significant contribution to the development of ADP.

In 1996, Berthsekas and Tsitsiklis [15] of the Massachusetts Institute of Technology published a special book on Neural Dynamic Programming. This book begins with a dynamic programming approach and background, and describes adaptive neural programming using neural networks. And summarizing the training method of this structure, finally found the general convergence theorem of the stochastic approximation method, which can be used as the basis for analyzing various neural dynamic programming algorithms.

In 1997, along with the proposed new neural network method, Prokhorov and Wunsch [16] proposed the design method and design procedure of HDP, DHP, GDHP, ADHDP, ADDHP based on BP neural network, using a neural network (evaluation network) estimation. The value function, another neural network (execution network) estimates the control action under the results of the previous neural network, while adjusting the two neural network weights using the least squares method. In order to prove that the weights of the ADP evaluation network and the execution network can converge during the iterative process, some random number estimations are used to conduct experiments, and the results all converge. Finally, the research examples of implementing the control of the club and the simulation of the landing of the aircraft with ADP are given, and an important contribution is made in the concrete realization of the ADP theory. In 2001, Jennie Si et al.[17]-[19] proposed a self-adaptive dynamic programming method for online learning neural networks that can be used directly without a mathematical model of the nonlinear system of the controlled object. In 2002, Murray et al. [20] proposed an iterative adaptive dynamic programming algorithm for a class of continuous-time nonlinear systems. Murray gave an analysis of the stability of the system stability and performance index function, making the adaptive dynamic programming online continuous. The implementation of time nonlinear systems has a theoretical basis. In 2005, Lee et al. [21] proposed a function approximator with local weighted averaging and ADP algorithm for Q-learned reinforcement learning. This algorithm can not only use the offline data to estimate the Q function, but also calculate the Q function online based on the relevant output data

of the system. Therefore, the algorithm does not need to use the neural network to form a model network to simulate the mathematical model of the controlled object. This is a non-model algorithm. In 2006, Padhi et al. [22] proposed an algorithm for adaptive dynamic programming using a single neural network adaptive evaluation (SNAC) structure. This method removes the execution network and uses a neural network to perform value functions and control actions. It is estimated that it has three potential advantages: a simpler structure, a smaller amount of calculation, and no approximation error. In 2007, Lweis et al. [23] studied the value function iteration method based on adaptive dynamic programming theory to solve the optimal control problem of discrete-time nonlinear systems. It is proved that the performance index function converges to the optimal solution satisfying the discrete-time HJB equation in the discrete case, which lays a theoretical foundation for the realization of adaptive dynamic programming in discrete-time nonlinear systems. In 2009, T. Dierks et al. [24] of the University of Missouri proposed a new ADP method. By using on-line system identification and off-line optimal control training, the application of ADP method does not need to know the dynamic characteristics of the non-linear system. In 2011, T. Dierks et al. [25] proposed an on-line optimal ADP control method for affine nonlinear discrete-time systems with unknown internal dynamics. In 2014, Heydari et al. [26] proposed a global optimal approximation dynamic programming algorithm for non-convex functions. The algorithm is also suitable for solving global optimal problems of nonlinear systems. In October 2015, AlphaGo [27] became the first computer program to defeat the Go pro on the 19th board. Its core algorithm uses approximate dynamic programming [28] combined with reinforcement learning and the deep learning of deep neural networks approximates the chessboard situation and the design of the drop strategy. In May 2017, Chinese chess player Ke Jie, who ranked the world number one in the past three years, defeated AlphaGo in three games. This is also the "historical leap" of artificial intelligence.

II. Adaptive Dynamic Programming Method

2.1 Basic Theory

The basic idea of adaptive dynamic programming uses the function approximation structure to approximate the optimal performance index function and the optimal control strategy in the dynamic programming equation to satisfy the optimality principle. Thereby obtaining the optimal control and optimal performance index function. Werbos has achieved good results in solving the problem of approximate dynamic programming by using the idea of performing network and evaluating network functions to approximate structure and control strategy. Bertsekas then applied this structure to the optimal control of nonlinear systems, which brought a huge breakthrough in solving nonlinear problems. Figure1 shows the schematic diagram of adaptive dynamic programming.

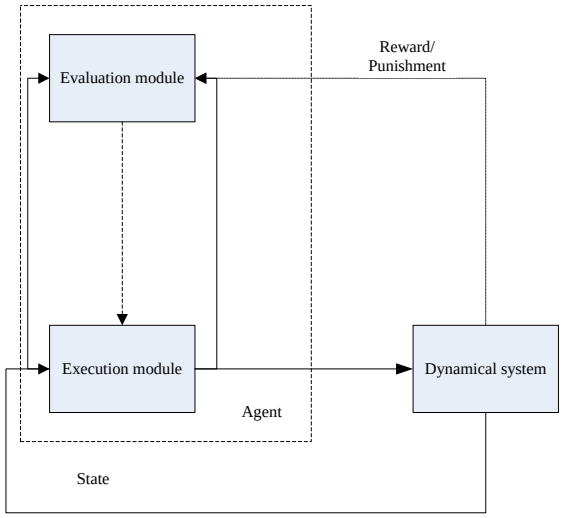


Figure 1. Schematic diagram of adaptive dynamic programming

Adaptive dynamic programming consists of three parts [29]: dynamic system, execution module and evaluation (Critic) module. These three parts can be constructed using neural networks. The input and

output relationships of dynamic systems can be identified and trained through neural networks. The implementation of the network is based on the guidance of the evaluation network to approximate the optimal control strategy. The evaluation network is used to approximate the optimal. Performance indicator function. The combination of the evaluation network and the execution network is equivalent to an agent, and the reward/penalty generated at different stages after the control/execution of the dynamic system (or the controlled object) affects the evaluation function. Next, the neural network is used to approximate the executive function and the evaluation function, but the executive function is based on the estimation of the evaluation function, that is, the evaluation function must be minimized. The parameter updating of the evaluation function is based on Bellman optimality principle, which not only reduces the forward calculation time, but also responds to the dynamic changes of the unknown system on-line, so that it can automatically adjust some parameters in the network structure.

The iterative algorithm of adaptive dynamic programming is as follows:

Step 1 Initialize：

① Setting Performance Indicator Function；

② Set the initial state；

③ Given Computational Accuracy ε.

Step 2 For $k = 0,1,2,\cdots$,the update control law is as follows:

$$u\left[x(k)\right] = \arg\min_{u(k)}\left\{U\left[x(k),u(k)\right] + J\left[x(k+1)\right]\right\} \tag{1}$$

Step 3 The update performance indicator function is as follows：

$$\begin{aligned} J\left[x(k)\right] &= \min_{u(k)}\left\{U\left[x(k),u(k)\right] + J\left[x(k+1)\right]\right\} \\ &= U\left[x(k),u(k)\right] + J\left[f\left(x_k,u\left(x(k)\right)\right)\right] \end{aligned} \tag{2}$$

Step 4 If $\left\|J\left[x(k)\right] - J\left[x(k+1)\right]\right\| < \varepsilon$, stop updating.

It can be seen from the algorithm of adaptive dynamic programming that the weight update of the execution network and the evaluation network must be iterated before it can be obtained. Evaluating the network term approximation performance index function plays an important role in guiding the implementation of network approximation control actions. At the same time, adaptive dynamic programming is not calculated according to the time-backward algorithm like dynamic programming, but is continuously corrected in the positive order according to the random initial value under the Bellman equation. This enables the development of online adaptive dynamic programming algorithms, which are improvements of adaptive dynamic programming over dynamic programming algorithms.

2.2 Heuristic Dynamic Programming (HDP) method

The basic idea of adaptive dynamic programming is to first realize the approximate representation of the performance index function through a function approximation tool, and then use another approximation tool to realize the selection of the optimal control sequence, thus achieving the approximate optimal control of the whole system. In order to be able to solve the dynamic programming problem, different kinds of function approximation structures are proposed to directly or indirectly approximate the optimal performance index function and the optimal control.

Werbos [60] first proposed two approximate dynamic programming structures. One is called Heuristic Dynamic Programming (HDP) and the other is called Dual Heuristic Programming (DHP). The HDP structure based on neural network is shown in Figure 2.The approximate structure of the function uses neural network approximation. Next, Werbos gave two other forms of adaptive dynamic programming "execution dependencies", called Action Dependent Heuristic Dynamic Programming (ADHDP) and execution dependent Dependent Dual (Action

Dependent Dual). Heuristic Programming, ADDHP). These four structures are the basic structures in the adaptive dynamic programming method.

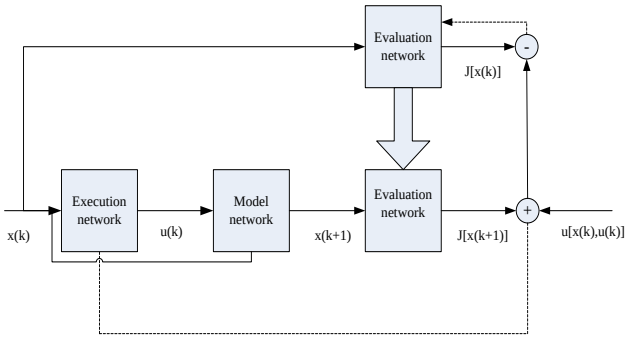


Figure 2. HDP structure diagram

In HDP, the performance indicator function can be written as:

$$J\left(x\left(k\right)\right)=U\left\{x\left(k\right),u\left[x\left(k\right)\right]\right\}+J\left[x\left(k+1\right)\right] \tag{3}$$

$u[x(k)]$ is the feedback control variable; the performance index functions $J[x(k)]$ and $J[x(k+1)]$ are the evaluation of the output of the neural network. If the weight of the evaluation network is ε, the right side of (3) can be

$$d\left[x\left(k\right),\omega\right]=U\left\{x\left(k\right),u\left[x\left(k\right)\right]\right\}+J\left[x\left(k+1\right),\omega\right] \tag{4}$$

The left side of Equation (3) can be written as J[x(k), ω], Therefore, by evaluating the evaluation of the neural network weight ω, the following mean square error function is minimized.

$$\omega^{*}=\arg\min_{\omega}\left\{\left|J\left[x\left(k\right),\omega\right]-d\left[x\left(k\right),\omega\right]\right|^{2}\right\} \tag{5}$$

Thereby obtaining the optimal performance index function. According to the principle of optimality, the optimal control should satisfy the first-order differential necessary condition, that is

$$\begin{aligned}\frac{\partial J^{*}\left[x\left(k\right)\right]}{\partial u\left(k\right)}&=\frac{\partial U\left[x\left(k\right),u\left(k\right)\right]}{\partial u\left(k\right)}+\frac{\partial J^{*}\left[x\left(k+1\right)\right]}{\partial u\left(k\right)}\\&=\frac{\partial U\left[x\left(k\right),u\left(k\right)\right]}{\partial u\left(k\right)}+\frac{\partial J^{*}\left[x\left(k+1\right)\right]}{\partial x\left(k+1\right)}\frac{\partial f\left[x\left(k\right),u\left(k\right)\right]}{\partial u\left(k\right)}\end{aligned} \tag{6}$$

2.3 Adaptive Execution Dependent Heuristic Dynamic Programming (ADHDP) method

The biggest difference between the HDP and ADHDP algorithms is that the evaluation network of ADHDP not only takes the system state as input, but also takes the control variable as input. The output of the evaluation network is often called the Q function, so ADHDP is also often called Q-learning. ADHDP merges the model network and the evaluation network into a new evaluation network, as shown in Figure 3. And Figure.4 shows the structure and schematic of ADHDP.

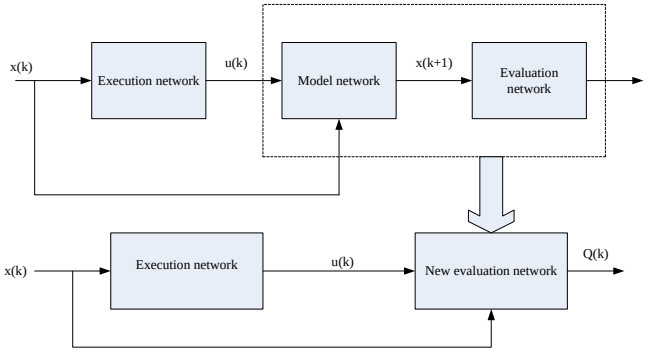


Figure 3. Construction of a new evaluation network in ADHDP

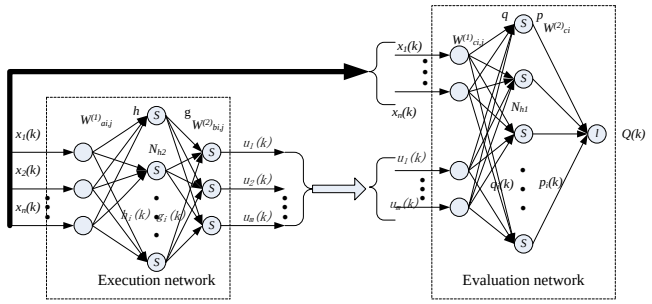


Figure 4. Structure and principle of ADHDP

Evaluate network training by minimizing the following error function:

$$\left\|E_{c}\left(k\right)\right\|=\sum_{k}\frac{1}{2}e_{c}^{2}\left(k\right)=\sum_{k}\frac{1}{2}\left[\gamma Q\left(k\right)-Q\left(k-1\right)+U\left(k\right)\right]^{2} \tag{7}$$

In (7), $Q(k)=Q[x(k),u(k),k,w_c]$, $U(k)$ is the utility function at time k; γ(0＜γ＜1) is the discount factor; w_c is the weight of the evaluation network.

According to (7), for all k, when $E_c(k) = 0$时, then further derivation:

$$\begin{aligned} Q(k) &= U(k+1) + \gamma Q(k+1) \\ &= U(k+1) + \gamma\left[Q(k+1) + \gamma Q(k+2)\right] \\ &= \ldots \\ &= \sum_{i=k+1}^{\infty} \gamma^{i-k-1} U(i) \end{aligned} \tag{8}$$

Suppose the input vector in the evaluation network:

$$\boldsymbol{y} = \left[x_1(k), x_2(k), \ldots, x_n(k), u_1(k), u_2(k), \ldots, u_m(k)\right] \tag{9}$$

The calculation method of the hidden layer input $q_i(k)$:

$$\boldsymbol{q} = w_{c_{i,j}}^{(1)} \times \boldsymbol{y}^T \tag{10}$$

In each operation, it is necessary to minimize $E_c(k)$, and use the gradient descent algorithm to update the weight of the evaluation network:

$$w_c(k+1) = w_c(k) + \Delta w_c(k) \tag{11}$$

$$\Delta w_c(k) = l_c(k)\left[-\frac{\partial E_c(k)}{\partial w_c(k)}\right] = l_c(k)\left[-\frac{\partial E_c(k)}{\partial e_c(k)} \cdot \frac{\partial e_c(k)}{\partial Q(k)} \cdot \frac{\partial Q(k)}{\partial w_c(k)}\right]$$

$$= -l_c(k) \cdot e_c(k) \cdot \gamma \cdot \frac{\partial Q(k)}{\partial w_c(k)} \tag{12}$$

After the completion of the network training, the training of the execution network is performed. The training goal of the execution network:

$$E_a(k) = \frac{1}{2} e_a^2(k) = \frac{1}{2} Q^2(k) = 0 \tag{13}$$

Therefore, after the optimization function $Q(k)$ is obtained, the optimal control $u^*(k)$ is easily obtained.

$$u^*(k) = \arg\min_{u(k)} Q^*\left[x(k), u(k)\right] \tag{14}$$

The calculation method for the input $h_i(k)$ from the input layer to the hidden layer in the execution network:

$$\boldsymbol{h} = w_{a_{i,j}}^{(1)} \times \boldsymbol{x}^T \tag{15}$$

The activation function used by the hidden layer of the execution network is a Sigmodial nonlinear function, and the output of the hidden layer:

$$g_i(k) = \frac{1 - \exp\left(-h_i(k)\right)}{1 + \exp\left(-h_i(k)\right)} \tag{16}$$

The calculation method of the output layer a of the execution network:

$$v = w_{a_{o,i}}^{(2)} \times g^T \tag{17}$$

The activation function used by the output layer of the execution network is a Sigmodial nonlinear function, and the output of the output layer:

$$u_o(k) = \frac{1 - \exp\left(-v_o(k)\right)}{1 + \exp\left(-v_o(k)\right)} \tag{18}$$

Minimize $E_a(k)$ in the operation, and use the gradient descent algorithm to update the weight of the execution network.

$$w_a(k+1) = w_a(k) + \Delta w_a(k)$$

$$\Delta w_a(k) = l_a(k)\left[-\frac{\partial E_a(k)}{\partial w_a(k)}\right] = l_a(k)\left[-\frac{\partial E_a(k)}{\partial e_a(k)} \cdot \frac{\partial e_a(k)}{\partial Q(k)} \cdot \frac{\partial Q(k)}{\partial w_a(k)}\right]$$

$$= -l_a(k) \cdot e_a(k) \cdot \frac{\partial Q(k)}{\partial w_a(k)} \tag{19}$$

III. Implementation of PMSM Control Algorithm Based on Data Driven Nonlinear MIMO ADP

3.1 Mathematical model of PMSM

According to the ideal motor model assumption, the three-phase voltage equation of PMSM in the natural coordinate system is

$$\boldsymbol{u}_{3s} = \boldsymbol{R}\boldsymbol{i}_{3s} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_{3s} \tag{20}$$

The flux linkage equation:

$$\boldsymbol{\psi}_{3s} = \boldsymbol{L}_{3s}\boldsymbol{i}_{3s} + \phi_f \cdot \boldsymbol{F}_{3s}(\theta_e) \tag{21}$$

$\boldsymbol{\psi}_{3s}$ is the flux linkage of the three-phase winding; $\boldsymbol{u}_{3s}$, $\boldsymbol{R}$, $\boldsymbol{i}_{3s}$ are the phase voltage, resistance and current of the three-phase winding; $\boldsymbol{L}_{3s}$ is the inductance of the three-phase winding; ϕ_f is a permanent magnet flux linkage; $\boldsymbol{F}_{3s}(\theta_e)$ is the flux linkage of the three-phase winding. And it satisfies the following conditions

$$\boldsymbol{i}_{3s}=\begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix}, \boldsymbol{R}_{3s}=\begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix}, \boldsymbol{\psi}_{3s}=\begin{bmatrix} \phi_A \\ \phi_B \\ \phi_C \end{bmatrix}, \boldsymbol{u}_{3s}=\begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix}, \boldsymbol{F}_{3s}(\theta_e)=\begin{bmatrix} \sin\theta_e \\ \sin(\theta_e-2\pi/3) \\ \sin(\theta_e+2\pi/3) \end{bmatrix}$$

$$\boldsymbol{L}_{3s}=L_{m3}\begin{bmatrix} 1 & \cos 2\pi/3 & \cos 4\pi/3 \\ \cos 2\pi/3 & 1 & \cos 2\pi/3 \\ \cos 4\pi/3 & \cos 2\pi/3 & 1 \end{bmatrix}+L_{l3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Where， L_{m3} is the stator mutual inductance； L_{l3} is the stator leakage inductance.

The mechanical motion equation of the motor:

$$J\frac{\mathrm{d}\omega_m}{\mathrm{d}t}=T_e-T_L-B\omega_m \tag{22}$$

Where ω_m is the mechanical angular velocity of the motor； J is the moment of inertia； B is damping coefficient； T_e is the electromagnetic torque； T_L is the load torque.

According to the principle of energy conversion, the electromagnetic torque T_e is equal to the partial derivative of the magnetic field energy storage to the mechanical angle θ_m displacement. Therefore,

$$T_e=\frac{1}{2}p_n\frac{\partial}{\partial\theta_m}\left(\boldsymbol{i}_{3s}^T\cdot\boldsymbol{\psi}_{3s}\right) \tag{23}$$

p_n is the pole pair of three-phase PMSM.

3.2 FOC control model of PMSM

Vector control refers to the idea that armature current and excitation current of DC motor are mutually perpendicular, uncoupled and independently controlled. Through d-q conversion in synchronous rotating coordinate system of three-phase PMSM motor, the size and direction of stator current of motor are controlled in synchronous rotating coordinate system. The purpose of decoupling d axis and q axis components is achieved, and the decoupling control of magnetic field and torque is realized. The control performance of AC motor is similar to that of DC motor.

Common methods of FOC are control and maximum torque to current ratio control. This paper uses the control method. As shown in Figure 5, FOC control mainly includes three parts: speed loop control, current loop control and two-level space vector modulation (SVPWM) algorithm.

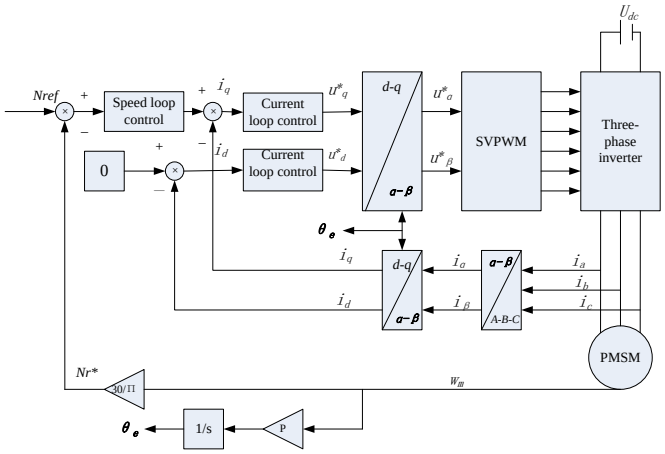


Figure 5. Three-phase PMSM vector control system diagram

3.3 Implementation of PMSM Control Algorithm Based on Data Driven Nonlinear MIMO ADP

In order to verify the data-driven ADP algorithm, and in the FOC control system for PMSM, in order to compare the PMSM control with the ADHDP controller, the compensation controller is first added only for the speed loop PI control. A schematic diagram of a system for adding a compensation controller to a speed loop PI controller is shown in Figure 6.

Define the utility function as

$$U\left(x_k,u_{s,k}\right)=R_N\Delta N_k^2+R_u u_{s,k}^2 \tag{24}$$

where $R_N>0$， $R_u>0$，They are the coefficients that compensate the input and output of the controller.

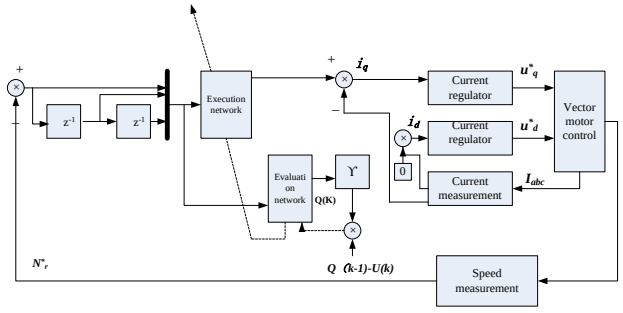


Figure 6. Speed loop ADP compensation controller system schematic

Design approximate cost function parameter is $W^{(i)}=\left[S_{11}^{(i)},S_{12}^{(i)},S_{22}^{(i)}\right]^{\mathrm{T}}$. The basis function is $\phi\left(x_k,u_{s,k}\right)=\left[\Delta N_k^2,2\Delta N_k u_{s,k},u_{s,k}^2\right]^{\mathrm{T}}$. Therefore, the approximate compensation control that can be obtained is

$$u_{s,k}=K\cdot\Delta N_k \tag{25}$$

$$\text{and } K=-\frac{S_{12}^{(i)}}{S_{22}^{(i)}}$$

The method for compensating only the speed loop PI controller is equivalent to a single-input single-output controller, and the PMSM has three PI controllers in the FOC control. If one compensation controller can be used to compensate three PI controllers simultaneously. Control, extended to multiple input multiple output (MIMO) controllers, not only can increase the application range of this method, but also make the control performance improvement more reliable. Figure 7 is a schematic diagram of the data-driven ADP compensation controller algebraic loop, and Figure 8 is a schematic diagram of the data-driven ADP system with multiple inputs and multiple outputs.

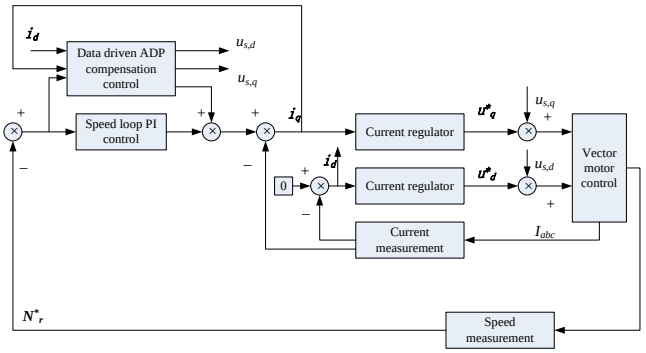


Figure 7. Data Driven ADP Compensation Controller Algebraic Ring Diagram

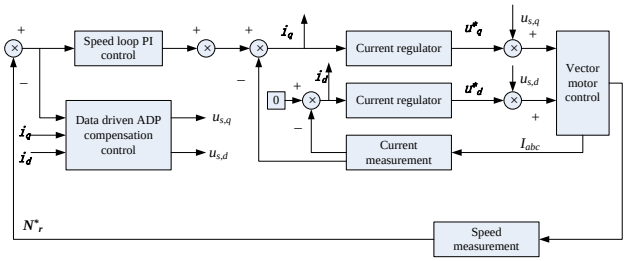


Figure 8. Schematic diagram of data driven ADP system with multi-inputs multi-outputs

For the added part, the algorithm needs to be re-derived. After expanding to a multi-input multi-output controller, the state variable x_k：

$$x_k=\left[\Delta N_r,\Delta I_d,\Delta I_q\right]^{\mathrm{T}} \tag{26}$$

The corresponding compensation control $u_{s,k}$：

$$u_{s,k}=\left[u_d,u_q\right]^{\mathrm{T}} \tag{27}$$

Define the utility function as

$$U\left(x_k,u_{s,k}\right)=R_N\Delta N_r^2+R_d\Delta I_d^2+R_q\Delta I_q^2+R_{u_d}u_d^2+R_{u_q}u_q^2 \tag{28}$$

The cost function is defined as

$$\hat{Q}^{(i)}\left(x_k,u_{s,k}\right)=\left[x_k,u_{s,k}\right]S^{(i)}\left[x_k,u_{s,k}\right]^{\mathrm{T}} \tag{29}$$

Where $S^{(i)}\in ¡^{5\times5}$, $S^{(i)}$ is a symmetric matrix , Therefore, $W^{(i)}$ is defined as the expanded form of the parameter matrix $S^{(i)}$, and only the upper triangular portion is expanded.

$$W^{(i)}=[S_{11}^{(i)},S_{12}^{(i)},S_{13}^{(i)},S_{14}^{(i)},S_{15}^{(i)},S_{22}^{(i)},S_{23}^{(i)},S_{24}^{(i)},S_{25}^{(i)},S_{33}^{(i)},S_{34}^{(i)},S_{35}^{(i)},S_{44}^{(i)},S_{45}^{(i)},S_{55}^{(i)}]^{\mathrm{T}} \tag{30}$$

IV. SIMULATION RESULTS AND ANALYSIS

First analyze the data-driven ADP compensation controller results that only increase the speed loop. The utility function that defines the speed loop data driven ADP:

$$U\left(x_k,u_{s,k}\right)=\Delta N_k^2+0.14\cdot u_{s,k}^2 \tag{31}$$

The simulation condition is set to：Speed $N_{ref}=1000$ rpm, Time is 0.4s, Motor initial load torque is $T_L=0$ N·m, When 0.2s, the load torque is $T_L=10$ N·m. The simulation results are shown in the figure below.

Figure 9 is the speed response curve of the PMSM, the dashed line represents the speed response curve of the ADHDP, and the solid line represents the data-driven speed response curve. The picture shows that the data-driven ADP control method can quickly converge to a preset speed. In addition, after

increasing the load for 0.2 seconds, the data-driven ADP compensation controller has a smaller overshoot than the ADHDP controller and can return to the preset speed more quickly.

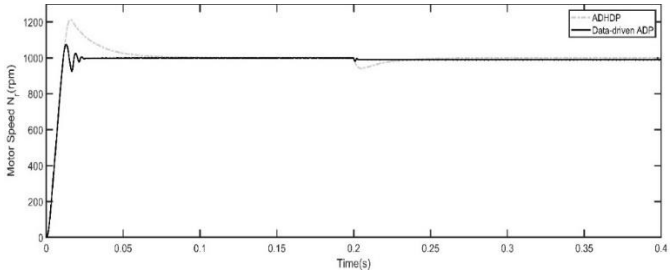
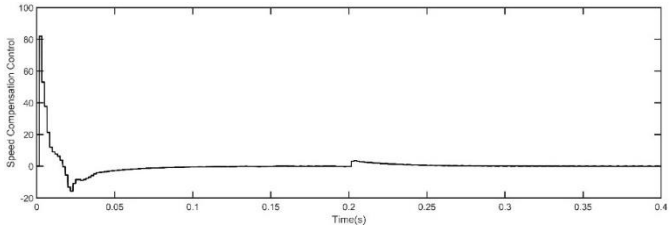
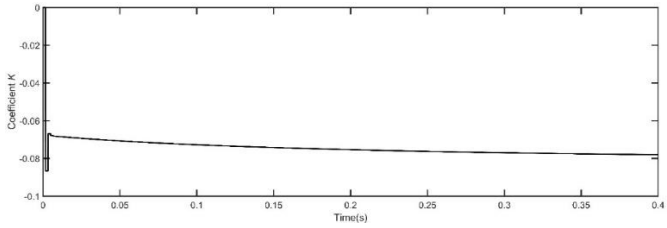


Figure 9. Speed response curve

The speed loop compensation control curve and the compensation controller coefficient K curve are shown in the Figure.10. When the compensator finds a large error at the beginning, it immediately increases the compensation control. This method enables fast calculations and allows the input and coefficient K to converge quickly.



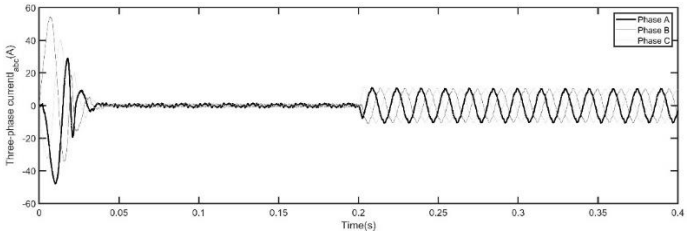
(a). Speed compensation control curve



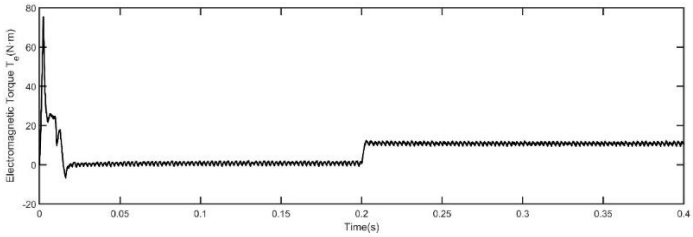
(b). Compensation controller coefficient K

Figure 10. (a) Speed compensation control curve; (b) Compensation controller coefficient K

Figure.11 shows the three-phase current curve and electromagnetic torque curve of the PMSM based on the data driven ADP method. There is no pulse overshoot after increasing the load at 0.2 seconds, so it can be explained that the data-driven ADP controller has strong robustness. At this point, the compensation controller has converged and can make correct compensation control for the system.



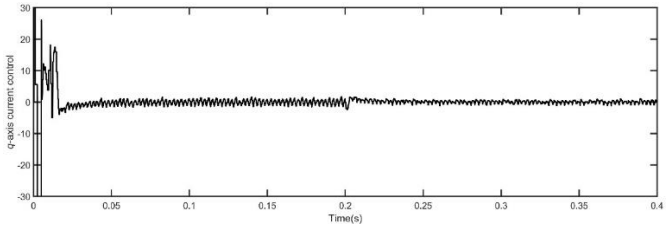
(a) Three-phase current I_{abc} curve



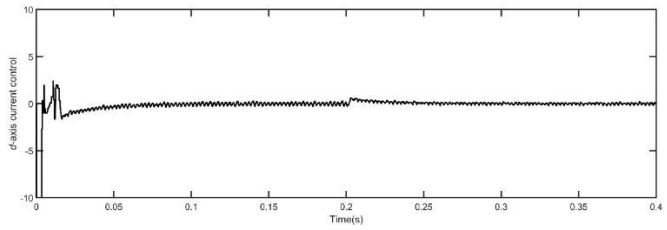
(b) Electromagnetic torque T_e curve

Figure 11. (a) Three-phase current I_{abc} curve; (b) Electromagnetic torque T_e curve

Figure 12 (a) is a d-axis current loop compensation control curve. Since the limit is not set for the compensation controller, the d-axis compensation controller can give a large control output when the PMSM speed is from 0 to 1000 rpm. When the PMSM reaches the set speed, the compensator quickly reduces the compensation control amount and fluctuates around 0. Figure 12 (b) is a q-axis current loop compensation control curve. The compensation control amount is very large in the corresponding initial stage because of the large variation of speed. After reaching the set speed, the compensation control amount decreases rapidly.



（a）q-axis current loop compensation control



(b) d-axis current loop compensation control

Figure 12.（a）q-axis current loop compensation control; (b) d-axis current loop compensation control

After comparing the two compensation applications, the effectiveness of ADP compensation control based on data-driven least squares strategy iteration can be obtained. This kind of controller needs more state information from the system. It can find that the effect of multi-input and multi-output is better than that of single-input and single-output. It also reflects the core idea of data-driven control, and constructs a System-compliant control method in the agent by collecting and learning the relevant data of the system.

V. DISCUSSION AND CONCLUSION

A data-driven ADP control method based on traditional ADP control is proposed, which is especially suitable for control optimization of complex power systems. The feasibility of the method is verified by PMSM modeling and control. By comparing the simulation experiments, it can be concluded that the data-driven nonlinear multi-input and multi-output adaptive dynamic programming algorithm proposed in this paper has obvious advantages. Simulation experiments show that the larger the amount of data, the more effectively the controller can construct the system information and improve the ability of the system to meet the corresponding indicators.

In addition, based on the adaptive dynamic programming of artificial neural network, through the development of reinforcement learning concept, the data-driven adaptive dynamic programming online compensation control method based on least squares strategy iteration is derived. The application of the intelligent compensation control method has important guiding significance for the intelligent transformation of the system. At the same time, it has further research significance for different controlled systems and promoting the development of AI autonomous control.

VI. ACKNOWLEDGMENTS

This work was supported by the NSFC Projects of China under Grant No. 61403250, No. 51509151, the bureau project of China under grant No. 2015HT056, and the Science Commission of Shanghai under grant No. 13510501600.

VII. REFERENCES

[1]. DUNIN-BARKOWSKI W L, WUNSCH D C. Phase-based storage of information in the cerebellum J. Neurocomputing, 1999, s 26–27(26-27): 677-685.

[2]. LIU D, HG Z. A neural dynamic programming approach for learning control of failure avoidance problems M. 2005.

[3]. WERBOS P J. Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design; proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, F, 2007 C.

[4]. WATKINS C J C H. Learning from Delayed Rewards J. Robotics & Autonomous Systems, 1989, 15(4): 233-235.

[5]. PROKHOROV D V, WUNSCH D C. Adaptive critic designs J. IEEE Transactions on Neural Networks, 1997, 8(5): 997.

[6]. LANDELIUS T. Reinforcement Learning and Distributed Local Model Synthesis J. Linkoping University Sweden, 1997,

[7]. SCHERRER B. Asynchronous neurocomputing for optimal control and reinforcement learning with large state spaces J. Neurocomputing, 2005, 63(1-4): 229-251.

[8]. WERBOS P J. Advanced Forecasting Methods for Global Crisis Warning and Models of Intelligence J. General Systems Yearbook, 1977, 22(6): 25-38.

[9]. WERBOS P J. Approximate dynamic programming for real-time control and neural

modeling J. Handbook of Intelligent Control Neural Fuzzy & Adaptive Approaches, 1992,

[10]. BERTSEKAS D P, TSITSIKLIS J N. Neuro-Dynamic Programming M. Athena Scientific, 1996.

[11]. TANG H. Performance Potential-based Neuro-dynamic Programming for SMDPs J. Acta Automatica Sinica, 2005, 31(4): 642-645.

[12]. ZHANG H, LUO Y, LIU D. A New Fuzzy Identification Method Based on Adaptive Critic Designs J. Lecture Notes in Computer Science, 2006, 3971(804-809.

[13]. NASCIMENTO J, POWELL W B. An Optimal Approximate Dynamic Programming Algorithm for Concave, Scalar Storage Problems With Vector-Valued Controls J. IEEE Transactions on Automatic Control, 2013, 58(12): 2995-3010.

[14]. WERBOS P. Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Science J. Phddissertation Harvard University, 1974, 29(18): 65-78.

[15]. BERTSEKAS D P, TSITSIKLIS J N, VOLGENANT A. Neuro-Dynamic Programming J. Encyclopedia of Optimization, 1996, 27(6): 1687-1692.

[16]. PROKHOROV D V, WUNSCH D C. Adaptive critic designs J. IEEE Transactions on Neural Networks, 1997, 8(5): 997-1007.

[17]. SI J, WANG Y T. On-Line Learning Control by Association and Reinforcement; proceedings of the Ieee-Inns-Enns International Joint Conference on Neural Networks, F, 2000 C.

[18]. ENNS R, SI J. Apache Helicopter Stabilization Using Neural Dynamic Programming J. Journal of Guidance Control & Dynamics, 2002, 25(1): 19-25.

[19]. ENNS R, SI J. Helicopter trimming and tracking control using direct neural dynamic programming J. IEEE Transactions on Neural Networks, 2003, 14(4): 929-939.

[20]. MURRAY J J, COX C J, LENDARIS G G, et al. Adaptive dynamic programming J. Systems Man & Cybernetics Part C Applications & Reviews IEEE Transactions on, 2002, 32(2): 140-153.

[21]. LEE J M, LEE J H. Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes M. Pergamon Press, Inc., 2005.

[22]. PADHI R, UNNIKRISHNAN N, WANG X, et al. A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems J. Neural Networks the Official Journal of the International Neural Network Society, 2006, 19(10): 1648.

[23]. ALTAMIMI A, LEWIS F L, ABUKHALAF M. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof J. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society, 2008, 38(4): 943-949.

[24]. DIERKS T, THUMATI B T, JAGANNATHAN S. Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence M. Elsevier Science Ltd., 2009.

[25]. ERKS T, JAGANNATHAN S. Online Optimal Control of Affine Nonlinear Discrete-Time Systems With Unknown Internal Dynamics by Using Time-Based Policy Update J. IEEE Trans Neural Netw Learn Syst, 2012, 23(7): 1118-1129.

[26]. HEYDARI A, BALAKRISHNAN S N. Global optimality of approximate dynamic programming and its use in non-convex function minimization M. Elsevier Science Publishers B. V., 2014.

[27]. SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search J. Nature, 2016, 529(7587): 484.

[28]. GRANTER S R, BECK A H, JR P D. AlphaGo, Deep Learning, and the Future of the Human Microscopist J. Archives of Pathology & Laboratory Medicine, 2017, 141(5): 619.

[29]. WANG F Y, ZHANG H, LIU D. Adaptive Dynamic Programming: An Introduction J. IEEE Computational Intelligence Magazine, 2009, 4(2): 39-47.

Cite this article as :

Bowen Sui, "Data-driven Nonlinear MIMO ADP Method and Its Application in PMSM Control", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 6 Issue 6, pp. 175-186, November-December 2019. Available at doi : https://doi.org/10.32628/IJSRSET196644 Journal URL : http://ijsrset.com/IJSRSET196644