Reinforcement-Learning-Based Parameter Look-Up Table Generating Method for Optimal Torque Control of Induction Motors

Qi Xing, Cao Wenping, Aarniovuori Lassi

DOI: 10.1109/TIE.2022.3189103

Reinforcement-Learning-Based Parameter Look-up Table Generating Method for Optimal Torque Control of Induction Motors

Xing Qi, *Member, IEEE,* Wenping Cao, *Senior Member, IEEE,*
Lassi Aarniovuori, *Senior Member, IEEE*

***Abstract*—In the optimal control of induction motors, it is a challenging task to maintain the optimal torque over the varying operation conditions. This paper proposes a parameter look-up table generating method, that can achieve an optimal torque over a wide range of currents and speeds, even though the commands of current are not set correctly. Based on the motor's testing data, this method uses a reinforcement-learning algorithm to generate parameter look-up tables iteratively. Experimental results show that the proposed method can learn appropriate parameters from the running data to output an optimal torque. The comparative studies show that the proposed method can generate 5%-25% more torque than traditional model-based parameter estimation methods, over a wide range of currents and speeds. Furthermore, the proposed method has a faster convergence feature and a higher identification resolution than many conventional search-based methods.**

***Index Terms*—Induction motor, optimal torque, parameters look-up table, reinforcement learning**

I. Introduction

INDUCTION motors (IMs) are widely used in industrial applications because of their low cost, high reliability and robustness [1]. In recent years, many researchers have developed IMs' optimal control that can fulfill the requirement of torque and efficiency optimization in the applications of automotives [2], electrical propulsion systems [3], and so on. The optimal control of IMs is generally classified into three categories: **1.** search methods, which adjust the optimal working conditions by iterative search [4]; **2.** model-based methods, which set the optimal motor drive command based on the power-loss model [5]; and **3.** hybrid methods, which use both iterative searches and models [6]. The optimal control can be applied to both IMs' scalar control [7] and vector control [8], the former needs to calculate an optimal air-gap flux, and the latter needs to set the correct commands of d-q axis current (denoted by i_{sd} and i_{sq}) under different speeds to satisfy the optimal flux and torque.

In addition to the air-gap flux and d-q axis current commands, there are other motor parameters are also the key factors that affect the motor's performances. For example, if an induction motor runs in an indirect field orient control (IFOC), incorrect values of the rotor resistance (R_r) and the magnetizing inductance (L_m) could lead to an inaccurate rotor field orientation [9]. In the case the torque will be significantly reduced even if i_{sd} and i_{sq} are set correctly. On the contrary, if the values of R_r or L_m can be estimated properly, the motor can be adjusted to an optimal torque condition even if the command of i_{sd} and i_{sq} are set incorrectly [10].

However, considering the effect of magnetic saturation and field weakening, in practice i_{sd} and i_{sq} are usually hard to be set optimally under each working condition. Therefore, it is a challenge to determine the IM's optimal parameter values for producing maximum torque under incorrect i_{sd} and i_{sq} setting. In recent years, many model-based IM's parameter estimation methods have been reported, such as no-load and locked-rotor (NL) [11], model reference adaptive system (MRAS) [12], extended kalman filter (EKF) [13] and full-order observer (FLO) [14]. These methods establish a equivalent circuit model or a differential equation model, and estimate these model's parameters. The model-based methods are easy to implement, and can estimate the parameters online. However, most model-based methods cannot adjust parameters adaptively as the working condition changes. Hence the model-based methods can't be used for generating parameter look-up tables for optimal torque.

The second category of parameter estimation methods are search-based methods, such as genetic algorithm (GA) [15], particle swarm optimization (PSO) [16] and grid optimization (GO) [17]. These methods establish cost functions, and then use some heuristic search techniques to search a minimum cost value. Although search-based methods can be used to generate parameter lookup tables, these methods require a large number of iteration operations and a long test time. Besides, the global convergence of these search-based methods still needs to be further analyzed [18].

To address the above challenges, this paper proposes a reinforcement-learning-based parameter look-up table generating method for optimal torque control. It calculates the optimal values of R_r and L_m, and then generates R_r and L_m look-up tables under different speeds and currents. The

Manuscript received October 06, 2021; revised January 05, 2022 and March 24, 2022; accepted June 18, 2022. This work was supported by the Science and Technology Key Special Project of Anhui Province (No. 2022h11020023 and No. 202103a05020019). (Corresponding authors:Wenping Cao.)

Xing Qi and Wenping Cao are with School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China. (e-mail: 10031@ahu.edu.cn; 19122@ahu.edu.cn)

Lassi Aarniovuori is with the Electrical Drives Laboratory, LUT-University, Lappeenranta 53850, Finland. (e-mail: lassi.aarniovuori@lut.fi)

generated look-up tables can guarantee optimal torque control over a wide range of working conditions, including the case where the commands of i_{sd} and i_{sq} are not correctly set. In the proposed method, the IMs' data are firstly acquired in experimental tests, and the testing data are stored into a data pool. The data pool provides the algorithm learn experience from a large number of data, such that the calculated results are sensitive to the working conditions. Then the data from the data pool are randomly selected and input to a reinforcement-learning algorithm. The algorithm runs iteratively to generate the parameter look-up tables in an end-to-end manner.

II. Parameter analysis for optimal torque control of IMs

According to the IM's control theory, the d-q axis voltage equations are given by:

$$\begin{cases} u_{sd} = R_s i_{sd} - \omega L_{\sigma s} i_{sq} \\ u_{sq} = R_s i_{sq} + \omega L_s i_{sd} + L_{\sigma s}\frac{\mathrm{d}i_{sq}}{\mathrm{d}t} \end{cases} \tag{1}$$

and the torque is denoted by:

$$T_e = \frac{3}{2} n_p \frac{L_m^2}{L_r} i_{sd} i_{sq} \tag{2}$$

where ω is the electrical angular speed, T_e is the output torque, n_p is the number of pole pairs. u_{sd}, u_{sq}, i_{sd} and i_{sq} are the d-q axis voltages and d-q axis currents. R_r and R_s are the stator resistance and rotor resistance, respectively. L_m denotes the magnetizing inductance, and the rotor inductance L_r can be written as $L_r = L_m + L_{\sigma r}$, where $L_{\sigma r}$ is the rotor leakage inductance. Usually $L_{\sigma r} \ll L_m$ and $L_m \approx L_r$.

When the IM runs in IFOC, the exciting current and the flux rotating speed are:

$$\begin{cases} \frac{\mathrm{d}i_{md}}{\mathrm{d}t} = \frac{R_r}{L_m}(-i_{md} + i_{sd}) \\ \omega = n_p\omega_r + \omega_s = n_p\omega_r + \frac{R_r i_{sq}}{L_m i_{md}} \end{cases} \tag{3}$$

where ω_r is the mechanical speed of the IM.

It can be seen in Eqs. 1-3 that the selection of R_r and L_m greatly affects the accuracy of the rotor field orientation. Therefore, in IFOC, R_r and L_m are the key parameters that affect the performance of IM's torque optimal control.

A. Determination of R_r for optimal torque control

Firstly assume that L_m is constant, define R_r as the actual rotor resistance, and $R_r^{'}$ as the estimated rotor resistance, the torque of IM can be rewritten as [19]:

$$T_e = \frac{3}{2} n_p L_m \frac{R_r^{'}}{R_r} i_{sd} i_{sq} \frac{i_{sd}^2 + i_{sq}^2}{i_{sd}^2 + (\frac{R_r^{'}}{R_r})^2 i_{sq}^2} \tag{4}$$

Dividing both the numerator and denominator by i_{sd}^2, Eq. 4 can be rewritten as:

$$T_e = f(\frac{R_r^{'}}{R_r}, \frac{i_{sq}}{i_{sd}}) = \frac{3}{2} n_p L_m (\frac{R_r^{'}}{R_r})(\frac{i_{sq}}{i_{sd}}) \frac{1 + (\frac{i_{sq}}{i_{sd}})^2}{1 + (\frac{R_r^{'}}{R_r})^2 (\frac{i_{sq}}{i_{sd}})^2} \tag{5}$$

Eq. 5 indicates that the torque is a function of $R_r^{'}/R_r$ and i_{sq}/i_{sd}, which is not linear. In order to maintain a maximum output torque, $f(R_r^{'}/R_r, i_{sq}/i_{sd})$ should be set as a convex function. For simplicity, let $R_r^{'}/R_r = \Theta R_r$ and $i_{sq}/i_{sd} = \Theta I$, the *Hessian* matrix of Eq. 5 is:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial \Theta R_r^2} & \frac{\partial^2 f}{\partial \Theta R_r \partial \Theta I} \\ \frac{\partial^2 f}{\partial \Theta I \partial \Theta R_r} & \frac{\partial^2 f}{\partial \Theta I^2} \end{pmatrix} \tag{6}$$

where:

$$\begin{cases} \frac{\partial^2 f}{\partial \Theta R_r^2} = \frac{2\Theta R_r \Theta I^3 (1+\Theta I^2)(\Theta R_r^2 \Theta I^2 - 3)}{(1+\Theta R_r^2 \Theta I^2)^3} \\ \frac{\partial^2 f}{\partial \Theta R_r \Theta I} = -\frac{\Theta R_r^4 \Theta I^6 - \Theta R_r^4 \Theta I^4 + 6\Theta R_r^2 \Theta I^4 + 6\Theta R_r^2 \Theta I^2 - 3\Theta I^2 - 1}{(1+\Theta R_r^2 \Theta I^2)^3} \\ \frac{\partial^2 f}{\partial \Theta I \Theta R_r} = -\frac{\Theta R_r^4 \Theta I^6 - \Theta R_r^4 \Theta I^4 + 6\Theta R_r^2 \Theta I^4 + 6\Theta R_r^2 \Theta I^2 - 3\Theta I^2 - 1}{(1+\Theta R_r^2 \Theta I^2)^3} \\ \frac{\partial^2 f}{\partial \Theta I^2} = \frac{2\Theta R_r \Theta I (\Theta R_r^4 \Theta I^2 - \Theta R_r^2 \Theta I^2 - 3\Theta R_r^4 + 3)}{(1+\Theta R_r^2 \Theta I^2)^3} \end{cases} \tag{7}$$

To keep $f(\Theta R_r, \Theta I)$ as a convex function, the *Hessian* matrix in Eq. 6 needs to be positive semi-definite [20], that is:

$$\frac{\partial^2 f}{\Theta R_r^2}\frac{\partial^2 f}{\Theta I^2} - (\frac{\partial^2 f}{\Theta R_r \Theta I}\frac{\partial^2 f}{\Theta I \Theta R_r}) \geq 0 \tag{8}$$

Substituting Eq. 7 into Eq. 8,

$$\begin{cases} 0 \leq R_r^{'}/R_r \leq 1 \quad \text{if} \quad i_{sq}/i_{sd} \geq 1 \\ R_r^{'}/R_r > 1 \quad \text{if} \quad 0 < i_{sq}/i_{sd} < 1 \end{cases} \tag{9}$$

Eq. 9 indicates that in order to achieve an optimal torque, the estimated $R_r^{'}$ should be lower than R_r for $i_{sq}/i_{sd} \geq 1$, and should be higher than R_r for $0 < i_{sq}/i_{sd} < 1$. For example, Fig. 1 plots the relationship between T_e and i_{sq} at $n = 1000rpm$ and $i_{sd} = 40A$. It can be seen that to locate the torque in the convex area, $R_r^{'} > R_r$ when $i_{sd} > i_{sq}$, and $R_r^{'} < R_r$ when $i_{sd} < i_{sq}$. Noted that if the magnetic saturation effect is considered, the selection rule of $R_r^{'}$ will be more complicated.

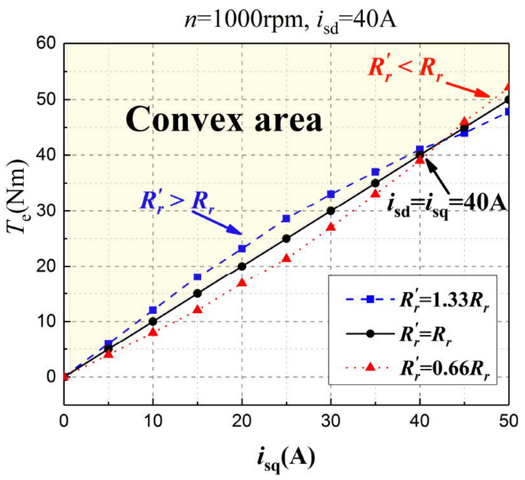


Fig. 1: The relationship between T_e and i_{sq} at $n = 1000rpm$ and $i_{sd} = 40A$.

B. Determination of L_m for optimal torque control

When the motor is working in a high-speed region, the core loss becomes dominant. The torque considering the core loss is given by:

$$T_{eFe} = \frac{3}{2} n_p \frac{L_m^2}{L_r} (i_{mqs}\psi_{dr} - i_{mds}\psi_{qr}) \tag{10}$$

where T_{eFe} is the torque considering the core loss, i_{mds} and i_{mqs} are the d-q axis currents considering the core loss. ψ_{dr} and ψ_{qr} are the d-q axis fluxes, respectively. The ratio of T_e and T_{eFe} is given by [21]:

$$\frac{T_e}{T_{eFe}} = \frac{1 + \omega_s^2 T_r^2}{(\omega_s T_r + \omega_r \Gamma_{rFe})^2 + (1 - \omega_s \omega_r \Gamma_{rFe} T_{\sigma r})^2} \tag{11}$$

where $T_{\sigma r}$ is the leakage inductance time constant, and Γ_{rFe} is the rotor time constant considering the core loss, denoted by:

$$\Gamma_{rFe} = \frac{L_m}{R_{Fe}} = \frac{L_m}{R_r} \frac{R_r}{R_{Fe}} = \frac{L_{mFe}}{R_r} \tag{12}$$

where R_{Fe} is the core loss resistance and L_{mFe} is the equivalent magnetizing inductance considering the core loss.

Since $L_{\sigma r} \ll L_m$, Eq. 11 can be approximated as:

$$\frac{T_e}{T_{eFe}} \approx \frac{1 + (\omega_s^2 T_r^2)}{1 + (\omega_r \frac{L_{mFe}}{R_r})^2} = \frac{R_r^2 + \omega_s^2 L_m^2}{R_r^2 + 2\omega_r L_{mFe} R_r + \omega_r^2 L_{mFe}^2} \tag{13}$$

This suggests that L_{mFe} significantly affects the output torque at high speeds.

According to Eqs. 12 and 13, in the high speed region where the flux is weakened, the value of L_m should be increased to minimize the core loss of the motor.

III. Parameter Look-up Table Generating Based on Deep-Reinforcement-Learning

In this section a reinforcement-learning-based method is introduced to generate R_r and L_m look-up tables under different speeds and currents. The proposed method is based on deep deterministic policy gradient (DDPG) [22], and thus named the deep deterministic policy gradient parameters look-up table generating method, or DDPG-PLTG for short.

A. General Framework of the DDPG-PLTG

The structure of DDPG-PLTG is illustrated in Fig. 2. The test data are firstly acquired and stored into a data pool for reinforcement learning [23], and then the data are selected randomly from data pool as the input, called *mini-batch* [24]. In DDPG-PLTG, the acquired motor's signals are called *observation* (denoted by s), the parameters' values are called *action* (denoted by a), and the output torque is called *reward* (denoted by r). The basic idea for the DDPG-PLTG is to search for the optimal policy (denoted by μ) that can maximize the reward under the current observation, and then select actions from the optimal policy (denoted by $a_t = \mu(s_t)$).

The DDPG-PLTG introduces an Actor-Critic [25] framework to accelerate the convergence, as shown in Fig. 3. Actor-Critic framework combines the feature of policy gradient (Actor) and time differential (Critic). It can output a continuous action with fast convergence. Specifically, the Actor firstly selects a policy with continuous actions using a policy gradient approach, and the Critic calculates the cost value of the Actor's policy using a time differential (TD) approach. Afterwards, the Actor modifies the policy according to the calculated cost values of Critic.

Moreover, DDPG-PLTG introduces a "updater-filter" to improve its robustness [22]. Specifically, it consists of a Q value calculation neural network (Q-net) and a policy gradient calculation neural network (PG-net), as in Fig. 2. The Q-net is used to search the optimal torque, and the PG-net is used to calculate the parameters that correspond to the optimal torque. In order to improve the algorithm's stability, both the Q-net and the PG-net are divided into two sub-networks, called the online update neural network (OU-net) and the filtering neural network (F-net), respectively. The OU-net uses a stochastic gradient descent [26] to update the weight matrix of the neural network, and the F-net uses a first-order filter to smooth the weight matrix in the OU-net, that is:

$$Filter \begin{cases} \boldsymbol{\theta}^{Q'} \leftarrow \tau\boldsymbol{\theta}^{Q} + (1-\tau)\boldsymbol{\theta}^{Q'} \\ \boldsymbol{\theta}^{\mu'} \leftarrow \tau\boldsymbol{\theta}^{\mu} + (1-\tau)\boldsymbol{\theta}^{\mu'} \end{cases} \tag{14}$$

where $\boldsymbol{\theta}^{Q}$ and $\boldsymbol{\theta}^{Q'}$ are the weight matrices of the OU-net and the F-net in the Q-net, and $\boldsymbol{\theta}^{\mu}$ and $\boldsymbol{\theta}^{\mu'}$ are the weight matrices of the OU-net and the F-net in the PG-net, respectively. τ is the filter coefficient. All of the neural networks have the identical 3-level network structure, including 4 input neurons, 30 hidden-level neurons and 1 output neuron.

B. Selection of Observation, Action and Reward

A key task of DDPG-PLTG is to find the best *observation*, *action* and *reward*. In IFOC, i_{sd}, i_{sq}, u_{sd} and u_{sq} can be chosen as the observation, denoted by:

$$s_t = [i_{sdt}, i_{sqt}, u_{sdt}, u_{sqt}] \tag{15}$$

where t denotes the time of calculations.

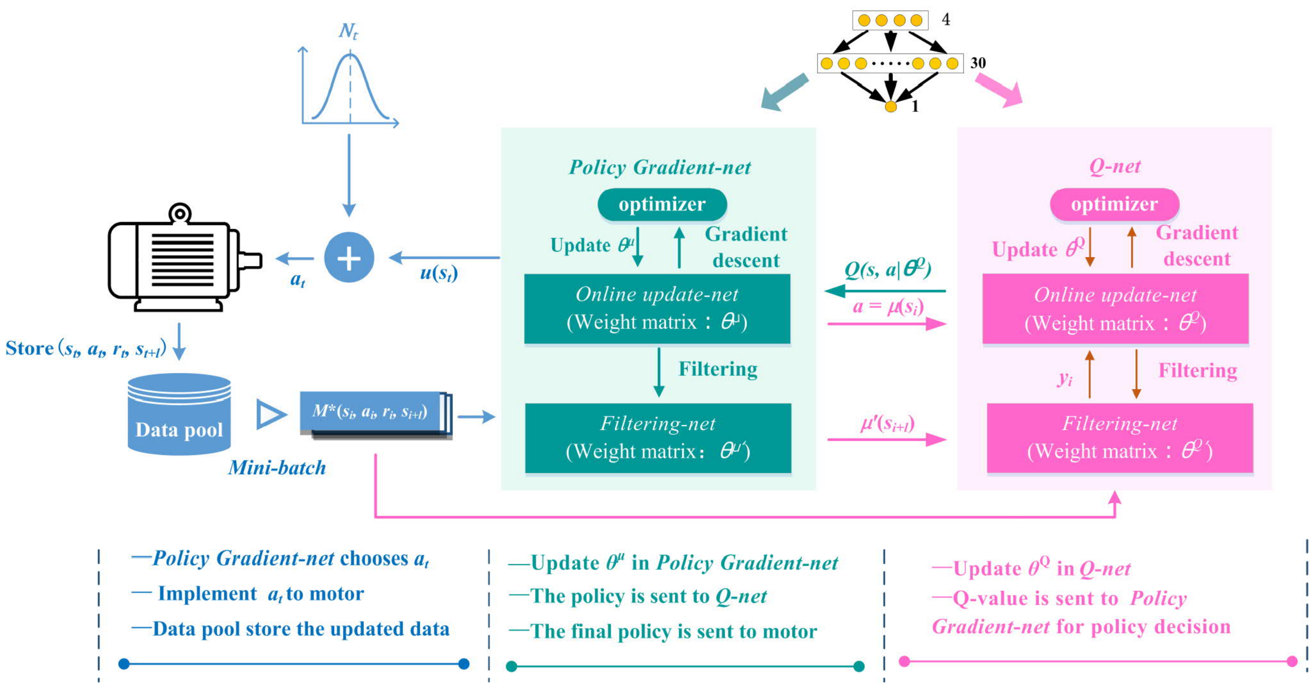
R_r and L_m are the estimated parameters, so the *action* can be set as:

$$a_t = [R_r, L_m] \tag{16}$$

In order to achieve maximum torque and accelerate the convergence, the differential of torque can be used as a reward, denoted by:

$$r_t = T_{et} - \mathbb{E}[T_{e\mathcal{D}}] = T_{et} - \frac{1}{\mathcal{D}} \int_0^{\mathcal{D}} T_e dt \tag{17}$$

where $\mathcal{D}$ denotes the data pool. Finally, the motor runs in the torque loop, and the control block diagram of the DDPG-PLTG is shown in Fig. 4, where the generated R_r and L_m look-up tables are input into the IFOC control strategy.



N_t
4
30
1
Policy Gradient-net
optimizer
Update θ^{μ}
Gradient descent
Online update-net (Weight matrix : θ^{μ})
Filtering
Filtering-net (Weight matrix: $\theta^{\mu'}$)
Q-net
optimizer
Update θ^{Q}
Gradient descent
Online update-net (Weight matrix : θ^{Q})
y_i
Filtering
Filtering-net (Weight matrix : $\theta^{Q'}$)
$Q(s, a|\theta^Q)$
$a = \mu(s_i)$
$\mu'(s_{i+1})$
a_t
$u(s_t)$
Store (s_t, a_t, r_t, s_{t+1})
Data pool
$M^*(s_i, a_i, r_i, s_{i+1})$
Mini-batch
—Policy Gradient-net chooses a_t
— Implement a_t to motor
—Data pool store the updated data
—Update θ^{μ} in Policy Gradient-net
—The policy is sent to Q-net
—The final policy is sent to motor
—Update θ^{Q} in Q-net
—Q-value is sent to Policy Gradient-net for policy decision

Fig. 2: The structure of DDPG-PLTG.

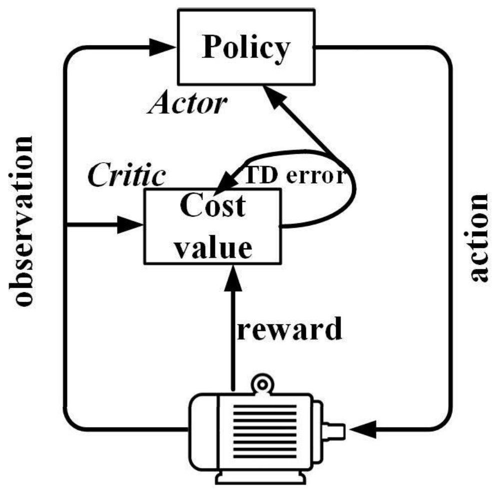


Fig. 3: Actor-Critic framework.

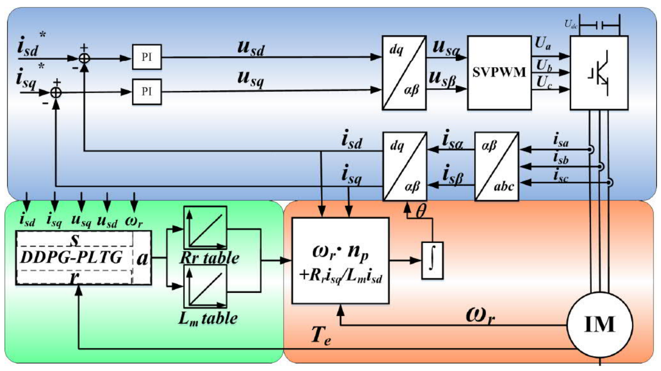


Fig. 4: The control block diagram for the torque loop of the IFOC.

C. Detailed steps of DDPG-PLTG

The detailed steps of DDPG-PLTG is illustrated in Algorithm.1. Firstly, the data pool and the observation are cleared before the calculation, as shown in lines 1 and 2, and all the networks' weight values are set to 0.01. In order to balance the exploration and exploitation [27], the action is superimposed with a Gaussian noise $\mathcal{N}_t$, as shown in line 6, denoted by:

$$\mathcal{N}_t = \epsilon^t \mathcal{N}(v_0, \Sigma_0) \tag{18}$$

where $0 < \epsilon < 1$, and $\mathcal{N}$ is the Gaussian distribution with a mean value of v_0 and a variance value of Σ_0.

Meanwhile, the DDPG-PLTG uses a stochastic gradient descent technology to train the loss function, as shown in lines 13 and 15, where η is the learning rate. The label of OU-net y is acquired by a Bellman function [28] as shown in line 11, where γ is the discounting factor and $0 < \gamma < 1$. Line 12 is the loss function of the OU-net in the PG-net, in which $Q(s_t, a_t|\boldsymbol{\theta}^Q)$ refers to the value function of the Q-net for given $\boldsymbol{\theta}^Q$, s_t and a_t.

The PG-net is trained by a policy gradient method [29], as shown in line 14. The policy gradient of the PG-net can be written as [22]:

$$\begin{aligned}\nabla_{\theta^{\mu}} J(\mu) &= \mathbb{E}_{s\sim\rho}[\nabla_a Q(s, a|\boldsymbol{\theta}^Q)|_{a=\mu(s)} \cdot \nabla_{\boldsymbol{\theta}^{\mu}} \mu(s|\boldsymbol{\theta}^{\mu})] \\ &= \frac{1}{M}\int_0^M [\nabla_a Q(s, a|\boldsymbol{\theta}^Q)|_{s=s_t, a=\mu(s)} \cdot \nabla_{\boldsymbol{\theta}^{\mu}} \mu(s|\boldsymbol{\theta}^{\mu})|_{s=s_t}]dt\end{aligned} \tag{19}$$

where $J(\mu)$ measures the performance of policy μ, $\nabla(\cdot)$ is the policy gradient, $\mathbb{E}_{s\sim\rho}[\cdot]$ is the expectation when s obeys the distribution of ρ, $\mu(s|\boldsymbol{\theta}^{\mu})$ is the policy under the condition of $\boldsymbol{\theta}^{\mu}$ and s, M is the size of mini-batch.

Algorithm 1 The DDPG-PLTG algorithm

```
 1: Init $\mathcal{D}$ and $s$;                                   //Initiate data pool and observation
 2: Init $Q(s,a|\boldsymbol{\theta}^Q)$ and $\mu(s|\boldsymbol{\theta}^\mu)$;   //Initiate the Q-net and the PG-net's weight matrices $\boldsymbol{\theta}^Q$ and $\boldsymbol{\theta}^\mu$
 3: $\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^Q$and $\boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta}^\mu$;   //Assign $\boldsymbol{\theta}^Q$ and $\boldsymbol{\theta}^\mu$ to OU-net's weight matrixs for the Q-net and the PG-net
 4: for $episode = 1, ..., i$ do                                   //Start iteration
 5:     $s_t \leftarrow [i_{sd}, i_{sq}, u_{sd}, u_{sq}]$          //Acquire the running data as the observation
 6:     $a_t = \mu(s_t|\boldsymbol{\theta}^\mu) + \mathcal{N}_t$     // The policy of the PG-net is superimposed with a Gaussian noise as the action
 7:     $[R_r, L_m] \leftarrow a_t$                                // The action is assign to parameters' calibration
 8:     $r_t \leftarrow T_{et} - \frac{1}{\mathcal{D}} \int_0^{\mathcal{D}} T_e dt$   //The torque or efficiency of the motor are assign to the algorithm's reward
 9:     $(s_t, r_t, a_t, s_{t+1}) \rightarrow \mathcal{D}$         //Store the current observation, action, reward and the next observation into data pool
10:     random$[M * (s_t, r_t, a_t, s_{t+1})] \leftarrow \mathcal{D}$   //Mini-batch: select $M$ groups of $\{s_t, r_t, a_t, s_{t+1}\}$ randomly from data pool
11:     $y_t = r_t + \gamma Q^{'}(s_{t+1}, \mu^{'}(s_{t+1|}|\mu^{'})|\boldsymbol{\theta}^{Q^{'}})$   //Calculate the label of the OU-net in the PG-net
12:     $L = \frac{1}{M} \int_0^M (y_t - Q(s_t, a_t|\theta^Q))^2$   //Calculate the loss function of the OU-net in the PG-net
13:     $L_{t+1} = L_t - \eta \nabla L_t$                          //Stochastic gradient descent the loss function
14:     $\nabla_{\theta^\mu} J(\mu) = \mathbb{E}_{s \sim \rho}[\nabla_a Q(s,a|\boldsymbol{\theta}^Q)|_{a=\mu(s)} \cdot \nabla_{\boldsymbol{\theta}^\mu} \mu(s|\boldsymbol{\theta}^\mu)]$   //Calculate the policy gradient of the OU-net in the PG-net
15:     $J_{t+1} = J_t - \eta \nabla J_t$                          //Stochastic gradient descent the policy gradient
16:     $\boldsymbol{\theta}^{Q'} \leftarrow \tau \boldsymbol{\theta}^Q + (1-\tau)\boldsymbol{\theta}^{Q'}$   //Filter the weight matrix of the OU-net in the Q-net
17:     $\boldsymbol{\theta}^{\mu'} \leftarrow \tau \boldsymbol{\theta}^\mu + (1-\tau)\boldsymbol{\theta}^{\mu'}$   //Filter the weight matrix of the OU-net in the PG-net
18: end for                                                        //Iteration finished
19: $[R_r, L_m] = \mu(s_t|\theta^\mu)$                             //Calulate the final action
20: $table \leftarrow [R_r, L_m]$                                  //Store the final action into the look-up tables
```

In each iteration, the output of the PG-net is filtered by the F-net, the final action results are stored as estimated parameters. For each command of speed n and d-q axis current $\{i_{sd}, i_{sq}\}$, recycle the steps in Algorithm.1, so that the R_r and L_m look-up tables can be generated.

D. Global Convergence Analysis of DDPG-PLTG

In order to analyze the global convergence of DDPG-PLTG, the detailed proof is needed in [30]. According to Eq. 1 and Eq. 5, the optimal paprameter calcultion can be equivalent to an optimization issue with constraints, written as:

$$
\begin{aligned}
&max[T_e(R_r, L_m)] = \tfrac{3}{2} n_p \tfrac{L_m^{*2}}{R_r^*} \tfrac{R_r}{L_M} i_{sd} i_{sq} \frac{i_{sd}^2 + i_{sq}^2}{i_{sd}^2 + (\frac{L_m^* R_r}{R_r^* L_M})^2 i_{sq}^2} \\
&s.t \begin{cases} u_{sd} = R_s i_{sd} - \omega L_{\sigma s} i_{sq} \\ u_{sq} = R_s i_{sq} + \omega L_s i_{sd} + L_{\sigma s} \frac{\mathrm{di_{sq}}}{\mathrm{d}t} \\ i_{sd} L_m \leq \psi_{max} \\ (i_{sd} \omega L_m)^2 + R_s^2 (i_{sd}^2 + i_{sq}^2) \leq U_{max}^2 \end{cases}
\end{aligned}
\tag{20}
$$

Substitute to Eqs. 15-17, Eq. 20 can be approximated as the following optimization issue, denote by:

$$
\begin{aligned}
min[-T_e(R_r, L_m)] &= s_t^T \mathbf{M} s_t + a_t^T \mathbf{N} a_t \\
s.t \qquad s_{t+1} &= \mathbf{A} s_t + \mathbf{B} a_t
\end{aligned}
\tag{21}
$$

where $\mathbf{M}, \mathbf{N}, \mathbf{A}, \mathbf{B}$ are the matrices with the variance of R_r and L_m, denoted by $(\mathbf{M}, \mathbf{N}, \mathbf{A}, \mathbf{B}) \sim Matrix(R_r, L_m)$. The superscript T denotes the transpose of a matrix. Noted that the exact expressions of $(\mathbf{M}, \mathbf{N}, \mathbf{A}, \mathbf{B})$ are usually unknown, hence the learning-based method should be used instead of conventional optimization methods, such as Newton or simplex methods.

In DDPG-PLTG, the policy performance J is the gradient descent (see Algorithm.1, line 15), written as $J_{t+1} = J_t - \eta \nabla J_t$, it has been proved that when the learning rate η satisfies the following inequality [30]:

$$
\eta \leq \frac{1}{16} \left\{ \left(\frac{\sigma_{min}(\mathbf{M})}{J} \right)^2 \frac{1}{\|\mathbf{B}\| \, \|\nabla J\| \, (1 + \|\mathbf{A} - \mathbf{B} J\|)} \right\}
\tag{22}
$$

we have:

$$
J_{t+1} - J^* \leq \left(1 - \eta \sigma_{min}(\mathbf{N}) \frac{\left[\sigma_{min}\left(s_t s_t^T\right)\right]^2}{\left\|s_t s_t^T\right\|} \right) (J_t - J^*)
\tag{23}
$$

and:

$$
|J_{t+2} - J_{t+1}| \leq \frac{1}{2} \eta \sigma_{min}(\mathbf{N}) \frac{\left[\sigma_{min}\left(s_t s_t^T\right)\right]^2}{\left\|s_t s_t^T\right\|} \epsilon
\tag{24}
$$

where $\sigma_{min}(\cdot)$ denotes the minimal singular value of a square matrix, and J^* is the optimal policy performance.

Eq. 24 can be rewritten as:

$$
J_{t+2} - J^* \leq \left(1 - \frac{1}{2} \eta \sigma_{min}(\mathbf{N}) \frac{\left[\sigma_{min}\left(s_t s_t^T\right)\right]^2}{\left\|s_t s_t^T\right\|} \right) (J_t - J^*)
\tag{25}
$$

According to Eqs. 22 and .25, as long as η is small enough, it holds that:

$$
J_{t+i} - J^* \leq \varepsilon
\tag{26}
$$

where ε is a very small number. Hence the global convergence is proved.

IV. Experimental Results and Analysis

A. Experimental Setup and Operation Flowchart

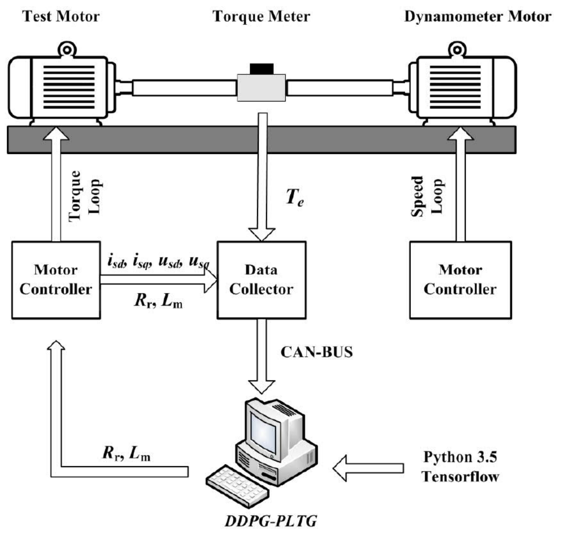
The experimental setup is shown in Fig. 5(a), which consists of a test motor, a dynamometer motor, a torque meter, a data collector and a host computer. In order to implement the DDPG-PLTG in practice, the torque meter is needed for torque data acquisition and the host computer is needed for algorithm running. The test motor runs in a torque loop of IFOC and the dynamometer motor runs in a speed loop. When the test motor is tested, the data of the observer $\{i_{sd}, i_{sq}, u_{sd}, u_{sq}\}$, the action $\{R_r, L_m\}$ and the reward $\{T_e\}$ are acquired by the motor controller and the torque meter. Then the acquired data are sent to the host computer by the data collector. The host computer is used to build the data pool and to run the DDPG-PLTG. In this work, the test motor is an induction motor used for automotive applications, and the parameters of the motor is listed in Table I. The host computer is equipped with: CPU: i7-6700k, GPU: GTX1080, RAM: 8G, ROM: 256G. The DDPG-PLTG algorithm is programmed using Python 3.5 and a deep learning library called *Tensorflow*.

TABLE I: Parameters of the test motor

Parameter	Value
Outer diameter of the stator	120 mm
Stator resistance (at 20°C)	0.0341 Ω
Rotor resistance (at 20°C)	0.0338 Ω
Unsaturated stator self inductance	0.0002 H
Unsaturated rotor self inductance	0.00018 H
Unsaturated mutual inductance	0.0041 H
Maximum speed	6000 r/min
Maximum torque (at 2000r/min)	80 Nm
Maximum power	17 kW
Number of pole pairs	2

An operation flowchart of DDPG-PLTG is shown in Fig. 5(b). Under different speeds, firstly the values of R_r and L_m are randomly selected and input to the IFOC of the test motor, and then the motor's running data $\{i_{sd}, i_{sq}, u_{sd}, u_{sq}, R_r, L_m, T_e\}$ is collected by data collector and then stored into the data pool. The above operations should be repeated under different R_r, L_m values and different i_{sd}, i_{sq} commands to build up the data pool. The data pool is built in the host computer using the SQlite database technology [31] with a "First In First Out (FIFO) " stack data structure, so that the data can be updated continuously. Then the data are selected randomly from the data pool to form a mini-batch, and input into the DDPG-PLTG algorithm. The DDPG-PLTG learns experiences from the data pool. When the DDPG-PLTG's calculation is finished, the new calculated action of R_r, L_m is sent to the motor controller for the next iteration.

The algorithm's hyper-parameters are: the number of iterations: 300; the number of data groups stored in the data pool: 100; the size of the mini-batch: 32; the mean value of Gaussian noise: 0.5; the variance of Gaussian noise: 0.1; the discounting factor: 0.9; the filter coefficient: 0.1; the learning rate: 0.001. At the exploration stage, the search ranges for R_r and L_m are limited to: 0-0.1Ω (for R_r) and 0-0.01H (for L_m), respectively. All of the above parameter settings are empirical data. Noted that since the proposed method is data-driven and in essence does not reply on any prior knowledge of the parameters, some other initial values of hyper-parameters also can make



(a) Experimental setup

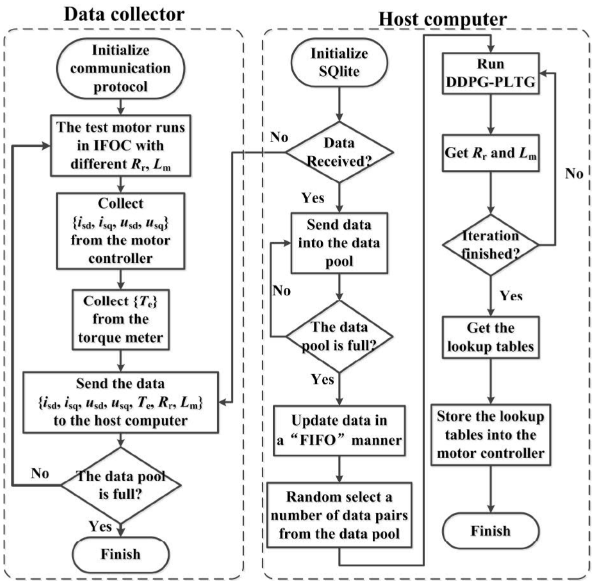

Data collector
Host computer
Initialize communication protocol
The test motor runs in IFOC with different R_r, L_m
Collect $\{i_{sd}, i_{sq}, u_{sd}, u_{sq}\}$ from the motor controller
Collect $\{T_e\}$ from the torque meter
Send the data $\{i_{sd}, i_{sq}, u_{sd}, u_{sq}, T_e, R_r, L_m\}$ to the host computer
The data pool is full?
No
Yes
Finish
Initialize SQlite
Data Received?
No
Yes
Send data into the data pool
The data pool is full?
No
Yes
Update data in a "FIFO" manner
Random select a number of data pairs from the data pool
Run DDPG-PLTG
Get R_r and L_m
Iteration finished?
No
Yes
Get the lookup tables
Store the lookup tables into the motor controller
Finish
(b) Operation flowchart of DDPG-PLTG

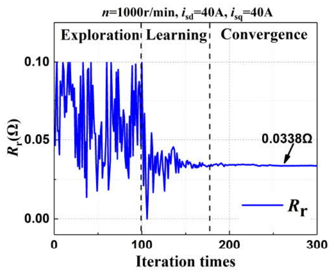
Fig. 5: Experimental setup and operation flowchart.

the algorithm work well. However, providing some reasonable initial values can speed up the search process, which is a balance to strike.

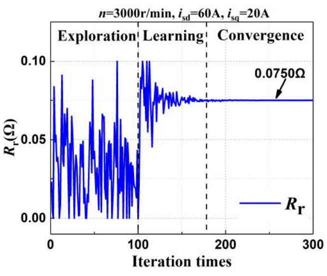
B. Calculations of R_r and L_m

Fig. 6 illustrates the process of R_r and L_m calculations under different i_{sd}, i_{sq} and n. For simplicity, the R_r look-up table is generated in the constant torque region, and the L_m look-up table is generated in the field weakening region. The process of the DDPG-PLTG is divided into three stages: a) **exploration stage**, in this stage the motor runs with random selected $\{R_r, L_m\}$ for 100 times, and then the 100 groups of running data is collected and input to the data pool for learning; b) **learning stage**, in this stage DDPG-PLTG learns the experience from the data pool; c) **convergence stage**, in this stage DDPG-PLTG converges to the optimal parameter value.

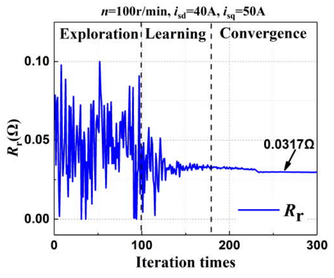
It can be seen in Fig. 6 that all the processes can converge, and can yield appropriate parameter values under different working conditions. Specifically, the DDPG-PLTG overestimates R_r when $i_{sd} \gg i_{sq}$ and under-estimates R_r when $i_{sd} \ll i_{sq}$, and increases L_m in flux weakening region to minimize core loss. Clearly these experimental results agree well with the analytical results, presented in Section II.



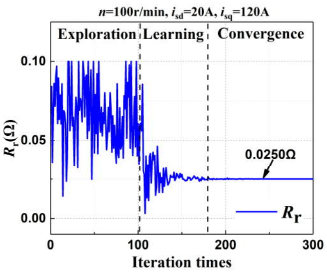
(a) n = 1000r/min, i_{sd} = 40A, i_{sq} = 40A



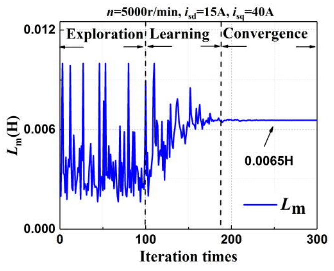
(b) n = 3000r/min, i_{sd} = 60A, i_{sq} = 20A



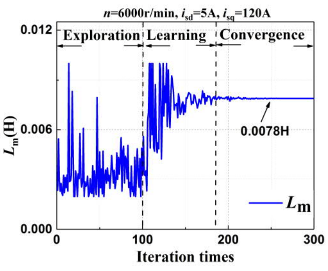
(c) n = 100r/min, i_{sd} = 40A, i_{sq} = 50A



(d) n = 100r/min, i_{sd} = 20A, i_{sq} = 120A



(e) n = 5000r/min, i_{sd} = 15A, i_{sq} = 40A



(f) n = 6000r/min, i_{sd} = 5A, i_{sq} = 120A

Fig. 6: The Calculation process of R_r and L_m for different speeds and d-q axis currents.

C. Comparision with Model-based Parameter Estimation Methods

In order to compare the proposed method with existing ones, the output torque is chosen as an indicator for a given input currents and a given operating speed, which represents a higher output power and motor drive efficiency. This indicator is crucial for many IM drives applications where a strong torque and a low loss are needed, such as electrical vehicles. In this work, two classical model-based parameter estimation methods: NL and MRAS, are used for comparison. NL is a standard test procedure for induction motor parameters estimation, whose detail information can be found in the IEEE Standard 112. MRAS is the most commonly used model-based method to estimate motor parameters online, whose structure can be found in Appendix. Fig. 7(a) and Fig. 7(b) show the comparative results of the rotor resistance and torque, respectively. At the speed 1000r/min, the i_{sd} and i_{sq} are changed from $i_{sd} = 50A, i_{sq} = 10A$, to $i_{sd} = 50A, i_{sq} = 100A$. It can be seen in Fig. 7(b) that the values of R_r calculated by NL and MRAS do not change with the d-q axis currents, while the DDPG-PLTG can solve the optimal parameter values according to different d-q axis currents. Meanwhile, the torque generated under the two conditions by DDPG-PLTG are higher than that of NL and MRAS. That is because the DDPG-PLTG has gained experience from the data pool for optimal torque generating under each operating condition, while NL and MRAS can only estimate parameters based on IM's equivalent circuit model, and cannot adjust parameters adaptively with the working conditions. In Fig. 7(b), a zoom-in plot is added to present a transition from $i_{sq} = 10A$ to $i_{sq} = 100A$. It can be seen that the transition process of the proposed DDPG-PLTG is more smooth than that of NL and MRAS.

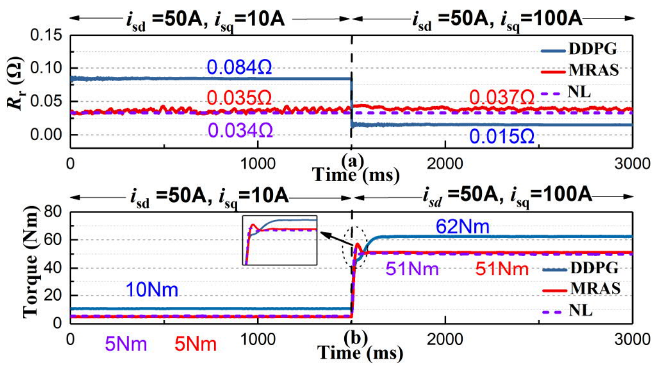


Fig. 7: Comparison of DDPG-PLTG, MRAS and NL under a transition of i_{sq} .

Table II presents a further comparison of NL, MRAS and DDPG-PLTG under different operating conditions. It can be seen that DDPG-PLTG can produce 5%∼25% higher torque than NL and MRAS for the same current and speed. Specifically, in the middle speed region and $i_{sd} \approx i_{sq}$, DDPG-PLTG can generate about 5% more torque than NL and MRAS, while in very low or very high speed region and $i_{sd} \gg i_{sq}$ or $i_{sd} \ll i_{sq}$, DDPG-PLTG can produce 25% more torque than NL and MRAS. As the DDPG-PLTG is data-driven and model-free, it can detect the change of operating conditions and correct the the error of the field orient adaptively, while NL and MRAS are based on motor's model and the model error can not be rectified. For example, MRAS does not work well in low speed regions where the model error may seriously affect the accuracy of the model. Likewise, NL does not work well in high speed regions because it is an offline method based on the motor's equivalent circuit, and can not sense the operating condition changes. In contrast, the DDPG-PLTG overcomes such shortcomings because it is a model-free method. It does not consider any model's information, but instead relies exclusively on the actual testing data, and uses the optimal torque as a reward to yield an accurate field orientation under any currents setting.

TABLE II: Comparison with some model-based methods

Speed	i_{sd}	i_{sq}	T_e(NL)	T_e(MRAS)	T_e(proposed)
100r/min	50A	20A	11Nm	9Nm	**13Nm**
100r/min	40A	50A	19Nm	16Nm	**20Nm**
100r/min	20A	170A	36Nm	32Nm	**44Nm**
1000r/min	50A	20A	11Nm	11Nm	**13Nm**
1000r/min	40A	50A	19Nm	19Nm	**20Nm**
1000r/min	20A	170A	36Nm	36Nm	**45Nm**
6000r/min	20A	20A	3Nm	4Nm	**4Nm**
6000r/min	15A	110A	14m	16Nm	**20Nm**
6000r/min	10A	170A	16Nm	20Nm	**22Nm**

D. Comparison with Search-based Methods

The DDPG-PLTG is compared with some classical search-based parameter estimation methods, including genetic algorithm (GA) [15], particle swarm optimization (PSO) [16], grid optimization (GO) [17] and deep-Q-learning (DQL) [32]. All the algorithms are implemented in the same computer. Table III gives the comparison results of GA, PSO, GO, DQL and DDPG-PLTG in terms of the number of iterations, the parameter searching resolution and the calculation time for each iteration. It can be seen that DDPG-PLTG needs a much lower iteration number than GA and PSO. This is because GA and PSO are realized in the form of "populations×particles", which need more iterations than reinforcement-learning-based methods. Moreover, GA and PSO are heuristic algorithms depending on random search, while DDPG-PLTG is a learning-based algorithm that has a much faster convergence than random search. As to the calculation time, it should be noted that if the actual-test torque is used as a cost function, the motor should keep running for about 2 seconds in each iteration for acquiring the stable torque value. This means that all the methods use a 2s sampling time, and the calculation times of each iteration are different but negligible, compared with the sampling time. Another advantage of DDPG-PLTG is that it has a higher parameter identification resolution than that of GO and DQL because GO and DQL must search in discrete steps, while the DDPG-PLTG has a continuous action and can yield a more precise searching solution. In summary, the DDPG-PLTG has the advantages of both a faster convergence and a higher searching resolution than other search-based methods.

V. Conclusion

Generating an optimal torque in IMs over a wide working conditions is crucial in many IM drives applications. However, an accurate determination of key parameters for IMs to generate optimal torque poses a challenge, especially when the commands of d-q axis currents (i_{sd}, i_{sq}) are not correctly set. To address the above issue, this paper has proposed a reinforcement-learning-based parameter look-up table generating method. It collects the motor's running data, and creates a data pool. Through a reinforcement learning method, the generated parameter look-up tables can produce an optimal torque over a wide range of speeds and currents. In order to implement the proposed method in practice, a motor test bench with a torque meter is needed for torque data acquisition, and a host computer is also needed to build the data pool and to run the proposed algorithm. Unlike model-based methods, the proposed method is data-driven: it relies exclusively on motor's actual testing data, so that it is not disturbed by model errors. Meanwhile, it uses a data pool to gain experience and to learn optimal data, so that the appropriate parameter values can be found as per different speeds and current commands. Unlike search-based methods, the proposed method is based on *learning* rather than *searching*, and thus has a fast convergence feature and a high identification resolution. The global convergence is proved and the experimental results show that the proposed method is effective. The comparative studies show that the proposed method can produce 5%∼25% more torque than conventional parameter estimation methods when i_{sd} and i_{sq} are not correctly set. The proposed method can be applied to the applications where a wide range of optimal torque are needed, such as electrical vehicles and electrical propulsion systems.

TABLE III: Comparison with conventional search-based methods

Method	Number of iterations	Resolution	Calculation time
GA [15]	2000	0.1%	32ms
PSO [16]	1500	0.1%	27ms
GO [17]	500	2%	4.8ms
DQL [32]	300	2%	7.2ms
DDPG-PLTG	**300**	**0.1**%	6.2ms

Appendix: An illustration of Model Reference Adjustable System (MRAS)

The structure of MRAS is shown as Fig. 8. It consists of a *reference model* and an *adjustable model*. Taking the estimation of R_r for example, the reference model of the motor is:

$$\begin{pmatrix} \frac{\mathrm{d}\psi_\alpha}{\mathrm{d}t} \\ \frac{\mathrm{d}\psi_\beta}{\mathrm{d}t} \end{pmatrix} = -\frac{L_r}{L_m} \begin{pmatrix} R_s + \dot{L}_{\sigma s} & 0 \\ 0 & R_s + \dot{L}_{\sigma s} \end{pmatrix} \begin{pmatrix} i_{s\alpha} \\ i_{s\beta} \end{pmatrix} + \frac{L_r}{L_m} \begin{pmatrix} u_{s\alpha} \\ u_{s\beta} \end{pmatrix} \tag{27}$$

where ψ_α and ψ_β are the reference flux in the $\alpha - \beta$ axis, and the adjustable model of the motor is:

$$\begin{pmatrix} \frac{\mathrm{d}\hat{\psi}_\alpha}{\mathrm{d}t} \\ \frac{\mathrm{d}\hat{\psi}_\beta}{\mathrm{d}t} \end{pmatrix} = \begin{pmatrix} -\frac{R_r}{L_m} & -\omega_r \\ \omega_r & -\frac{R_r}{L_m} \end{pmatrix} \begin{pmatrix} \hat{\psi}_\alpha \\ \hat{\psi}_\beta \end{pmatrix} + \frac{L_r}{L_m} \begin{pmatrix} i_{s\alpha} \\ i_{s\beta} \end{pmatrix} \tag{28}$$

where $\hat{\psi}_\alpha$ and $\hat{\psi}_\beta$ are the adjustable flux in the $\alpha - \beta$ axis, respectively.

The goal of MRAS is to enable the adjustable model to accurately track the reference model. According to Popov hyperstability theory or Lyapunov stability theory, the adjusted law of R_r can be designed by:

$$\hat{R}_r = K_p e + K_i \int e \tag{29}$$

where $e = \psi_\alpha \hat{\psi}_\beta - \psi_\beta \hat{\psi}_\alpha$ is the flux error between the reference model and the adjustable model. K_p and K_i are the adjustable coefficients.

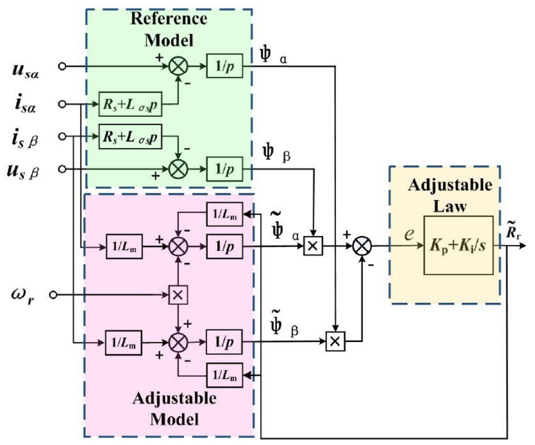


Fig. 8: The structure diagram of MRAS

References

[1] J. Talla, V. Q. Leu, V. Å mÃdl, and Z. Peroutka, "Adaptive speed control of induction motor drive with inaccurate model," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8532–8542, 2018.

[2] Y. Zhang, Y. Zhang, Z. Ai, Y. L. Murphey, and J. Zhang, "Energy optimal control of motor drive system for extending ranges of electric vehicles," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 2, pp. 1728–1738, 2021.

[3] M. Sztykiel, R. Pena-Alzola, C. Jones, P. Norman, S. Galloway, G. Burt, and S. Mukherjee, "Adaptive v/f-based control for induction machines in distributed electric propulsion systems," in *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles International Transportation Electrification Conference (ESARS-ITEC)*, 2018, pp. 1–7.

[4] I. Kioskeridis and N. Margaris, "Loss minimization in scalar-controlled induction motor drives with search controllers," *IEEE Transactions on Power Electronics*, vol. 11, no. 2, pp. 213–220, 1996.

[5] R. Tarvirdilu-Asl, S. Nalakath, Z. Xia, Y. Sun, J. Wiseman, and A. Emadi, "Improved online optimization-based optimal tracking control method for induction motor drives," *IEEE Transactions on Power Electronics*, vol. 35, no. 10, pp. 10 654–10 672, 2020.

[6] A. M. Bazzi and P. T. Krein, "Review of methods for real-time loss minimization in induction machines," *IEEE Transactions on Industry Applications*, vol. 46, no. 6, pp. 2319–2328, 2010.

[7] C. A. Costa, M. A. A. Costa, M. d. A. Turqueti, A. J. Rossa, A. Nied, and F. G. Nogueira, "Enhanced braking control for the induction machine using scalar control," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 11, pp. 9133–9142, 2020.

[8] S. A. Odhano, R. Bojoi, A. Boglietti, S. G. Rosu, and G. Griva, "Maximum efficiency per torque direct flux vector control of induction motor drives," *IEEE Transactions on Industry Applications*, vol. 51, no. 6, pp. 4415–4424, 2015.

[9] S. Yang, D. Ding, X. Li, Z. Xie, X. Zhang, and L. Chang, "A novel online parameter estimation method for indirect field oriented induction motor drives," *IEEE Transactions on Energy Conversion*, vol. 32, no. 4, pp. 1562–1573, 2017.

[10] C. Mastorocostas, I. Kioskeridis, and N. Margaris, "Thermal and slip effects on rotor time constant in vector controlled induction motor drives," *IEEE Transactions on Power Electronics*, vol. 21, no. 2, pp. 495–504, 2006.

[11] "Ieee standard test procedure for polyphase induction motors and generators," *IEEE Std 112-2017 (Revision of IEEE Std 112-2004)*, pp. 1–115, 2018.

[12] P. Cao, X. Zhang, and S. Yang, "A unified-model-based analysis of mras for online rotor time constant estimation in an induction motor drive," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4361–4371, 2017.

[13] E. Zerdali, "A comparative study on adaptive ekf observers for state and parameter estimation of induction motor," *IEEE Transactions on Energy Conversion*, vol. 35, no. 3, pp. 1443–1452, 2020.

[14] H. Kubota, K. Matsuse, and T. Nakano, "Dsp-based speed adaptive flux observer of induction motor," *IEEE Transactions on Industry Applications*, vol. 29, no. 2, pp. 344–348, 1993.

[15] M. Waheeda Beevi, A. Sukesh Kumar, and H. S. Sibin, "Loss minimization of vector controlled induction motor drive using genetic algorithm," in *2012 International Conference on Green Technologies (ICGT)*, 2012, pp. 251–257.

[16] Z. Liu, H. Wei, Q. Zhong, K. Liu, X. Xiao, and L. Wu, "Parameter estimation for vsi-fed pmsm based on a dynamic pso with learning strategies," *IEEE Transactions on Power Electronics*, vol. 32, no. 4, pp. 3154–3165, 2017.

[17] F. Duan, R. Zivanovic, S. Al-Sarawi, and D. Mba, "Induction motor parameter estimation using sparse grid optimization algorithm," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 4, pp. 1453–1461, 2016.

[18] Q. Yuan and G. Yin, "Analyzing convergence and rates of convergence of particle swarm optimization algorithms using stochastic approximation methods," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1760–1773, 2015.

[19] N. P. Quang, J.-A. Dittrich *et al.*, *Vector control of three-phase AC machines*. Springer, 2008, vol. 2.

[20] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] E. Levi, "Impact of iron loss on behavior of vector controlled induction machines," *IEEE Transactions on Industry Applications*, vol. 31, no. 6, pp. 1287–1296, 1995.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." in *ICLR*, 2016.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[25] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[26] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[27] S. Ishii, W. Yoshida, and J. Yoshimoto, "Control of exploitation-exploration meta-parameter in reinforcement learning." *Neural Networks*, vol. 15, no. 4-6, pp. 665–687, 2002.

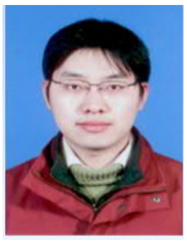
[28] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 1999.

[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," pp. 387–395, 2014.

[30] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. StockholmsmÃ¤ssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1467–1476.

[31] M. Owens and G. Allen, *The Definitive Guide to SQLite*. Apress, 2006.

[32] Q. Xing, "Rotor resistance and excitation inductance estimation of an induction motor using deep-q-learning algorithm," *Engineering Applications of Artificial Intelligence*, vol. 72, no. JUN., pp. 67–79, 2018.



Xing Qi (Member, IEEE) received the B.S. degree and M.S. degree in mechanical and electrical engineering from the Hefei University of Technology in 2007 and 2010. He received Ph.D degree from Anhui University in 2019. His research focuses on the fields of Engineering Application of Artificial Intelligence.

Wenping Cao (Senior Member, IEEE) received the B.Eng. in electrical engineering from Beijing Jiaotong University, Beijing, China, in 1991, and Ph.D. degree in electrical machines and drives from the University of Nottingham, Nottingham, in 2004. Prof. Cao is Chair Professor of Electrical Engineering with Anhui University, Hefei City, China. He received a "Royal Society Wolfson Research Merit Award" in 2016, the "Dragon's Den Competition Award" from Queen's University Belfast in 2014, the "Innovator of the Year Award" from Newcastle University, UK, in 2013. He is IET Fellow, Royal Society Wolfson Fellow and Marie Curie Fellow. He has been Associate Editor for IEEE Transactions on Power Electronics, IEEE Transactions on Industry Application, IET Power Electronics, and editor for several international journals. His research interests include fault analysis and condition monitoring of electrical machines and power electronics.

Lassi Aarniovuori (Senior Member, IEEE) received M.Sc. degree in electrical engineering and the D.Sc. degree in electric motors and drives from the Lappeenranta University of Technology (LUT), Lappeenranta, Finland, in 2005 and 2010, respectively. During 2017-2019 he was a Marie Curie Fellow with the School of Engineering and Applied Science, Aston University, Birmingham, U.K. He is currently an Associate Professor in Electric Transportation with the LUT School of Energy Systems.

His current research interests include all aspcets related to electric transportation systems, especially wide band-gap power switches, high-speed machine technology, modulation methods, simulation of electric drives, efficiency measurements, and calorimetric measurement systems.