Neural-Network Vector Controller for Permanent-Magnet Synchronous Motor Drives: Simulated and Hardware-Validated Results

Shuhui Li, *Senior Member, IEEE*, Hoyun Won, *Student Member, IEEE*, Xingang Fu, *Member, IEEE*, Michael Fairbank, Donald C. Wunsch, *Fellow, IEEE*, and Eduardo Alonso

***Abstract*—This paper focuses on current control in a permanent-magnet synchronous motor (PMSM). This paper has two main objectives: the first objective is to develop a neural-network (NN) vector controller to overcome the decoupling inaccuracy problem associated with the conventional proportional–integral-based vector-control methods. The NN is developed using the full dynamic equation of a PMSM, and trained to implement optimal control based on approximate dynamic programming. The second objective is to evaluate the robust and adaptive performance of the NN controller against that of the conventional standard vector controller under motor parameter variation and dynamic control conditions by: 1) simulating the behavior of a PMSM typically used in realistic electric vehicle applications and 2) building an experimental system for hardware validation as well as combined hardware and simulation evaluation. The results demonstrate that the NN controller outperforms conventional vector controllers in both simulation and hardware implementation.**

***Index Terms*—Approximate dynamic programming (ADP), neural network (NN), permanent-magnet synchronous motor (PMSM), vector control, voltage source inverter (VSI).**

I. Introduction

THE PERFORMANCE of a permanent-magnet synchronous motor (PMSM) depends not only on its hardware design but also on how it is controlled. Motor current control plays a particularly critical role [1]. Since there is a direct relation between motor current and torque, current control is equivalent to torque control [2]. To achieve fast and accurate current or torque tracking, several improved control techniques have been developed, including predictive current control [1], [2]; direct torque control [3], [4]; proportional–integral (PI) plus proportional–resonant (PR) control [5]; and mixed H2/H∞ control [6].

Predictive current control [1], [2] uses a current-prediction equation to estimate the motor current at the next sampling interval and a control equation to determine the next control action. It shows fast current-tracking response but becomes unstable when the actual motor's parameters differ from the programmed parameters used in the predictive controller [2].

Direct torque control provides a simple implementation for instant motor torque control [4]. However, it suffers from drawbacks, such as variable switching frequency, large torque ripple, and high sampling rates for digital implementation [4].

PI–PR control is similar to the conventional standard field-oriented vector control, except that it combines PI with several PR control paths to enhance tracking of the current which may contain a lot of ac disturbance components [5]. The PI–PR approach requires properly tuning parameters of different resonant terms, and its performance can be affected when motor parameters change, or when disturbance harmonics are different from those used to tune the resonant terms.

The mixed H2/H∞ control [7] requires a reasonably accurate system model [8]. Also, it does not handle nonlinear constraints very well [8]. In [9], it was found that applying a mixed H2/H∞ controller in experimental conditions is much more challenging than in simulated environments.

As a result of these weaknesses, the conventional standard field-oriented vector control is still the dominant motor-current control strategy for PMSMs in today's motor drive industry [10], [11]. But, recent studies show this conventional control strategy is inherently limited [12].

Neural networks (NNs) have been applied in PMSM control since 1990s. But, to the best of our knowledge, NNs have not been used for current control of a PMSM based on a voltage source inverter (VSI). In [13], a feedforward NN identifier is utilized to replace the traditional speed-loop controller to generate a reference current. The reference current is then compared with the actual current to drive a current

Manuscript received November 25, 2018; accepted January 20, 2019. This work was supported in part by the U.S. National Science Foundation under Grant IIP 1650564, in part by the Mary K. Finley Endowment, in part by the Missouri S&T Intelligent Systems Center, and in part by the Army Research Laboratory under Cooperative Agreement W911NF-18-2-0260. This paper was recommended by Associate Editor H. Zhang. *(Corresponding author: Shuhui Li.)*

S. Li and H. Won are with the Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL 35487 USA (e-mail: sli@eng.ua.edu; hwon@crimson.ua.edu).

X. Fu is with the Department of Electrical Engineering and Computer Science, Texas A&M University—Kingsville, Kingsville, TX 78363 USA (e-mail: xingang.fu@tamuk.edu).

M. Fairbank is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: m.fairbank@essex.ac.uk).

D. C. Wunsch is with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: dwunsch@mst.edu).

E. Alonso is with the Department of Computer Science, City, University of London, London EC1V 0HB, U.K. (e-mail: e.alonso@city.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2897653

source inverter (CSI) through a hysteresis switching scheme. The NNs presented in [14] and [15] have a similar function to the NN used in [13]. The difference is that a typical current-loop controller is introduced after the NN identifier, and a VSI replaces a CSI as the PMSM inverter. In [16], a feedforward NN is used as an identifier for a PMSM for the purpose of offsetting the impact of uncertainties.

This paper develops a novel control strategy: NN-based current vector-controller for a PMSM trained using an approximate dynamic programming (ADP) method. In recent years, significant research has been conducted in optimal control of nonlinear systems based on ADP [17]–[22], none of which, however, focuses on vector control of a PMSM, although many recent studies have pivoted around developing ADP techniques for optimal energy management in a time scale from several minutes to several hours. These include ADP-based energy storage management with solar renewable [23], ADP-based battery management for residential energy systems [24], and ADP-based home energy management [25]. But, the focus of this paper is on real-time control of PMSMs for a time scale of milliseconds and below. In [26], a preliminary NN vector control for PMSMs was developed.

This paper has extended far beyond [26]. The special contributions of this paper include the following.

1) An ADP-based NN controller.
2) Training of the NN as a recurrent network.
3) Detailed stability evaluation under a wide range of diverse conditions and parameter uncertainties.
4) Implementation and hardware experiment testing of the NN controller.

Several important features of the proposed NN control method include the following.

1) The NN is trained as a recurrent network, enabling it to exhibit fixed-weight adaptive behavior [27], and predictive control ability, like conventional current-predictive controllers.
2) The NN is trained to optimize an ADP-based cost function, making the NN controller an approximate optimal controller, like an H2/H∞ controller.
3) The NN controller takes the error integral information as the input, which guarantees that no steady-state error exists for the reference tracking.
4) The NN can, in theory, emulate PI–PR control features, due to the universal function-approximation capability of NNs.

Thus, the NN has the potential to integrate optimal, predictive, PI, and PR control characteristics together. This paper demonstrates the NN controller's improved performance, under both simulation and hardware conditions as compared to the conventional control methods.

It is worth emphasizing that the NN controller is trained entirely offline under a wide range of simulated circumstances, which allows the NN controller to adapt and respond to changing motor parameters in real time [27]. This leads to three further key benefits of the NN control method.

1) The controller shows sufficient adaptability not to need retuning every time the motor's parameters change slightly.
2) The computational cost at runtime is extremely low and easy to implement in low-cost hardware.
3) As training is completed offline, it is not possible for the weights of the NN to destabilize at runtime.

The trained NN controller's stability for controlling the PMSM at runtime is validated against a test set and a hardware experiment. The test set is intended to cover a sufficiently wide range of circumstances to empirically provide evidence for the stability of the controller. A formal proof of runtime stability is not provided. Such proof would involve the development of an analytical framework that is beyond the scope of this paper.

The rest of this paper is structured as follows. Section II covers the basic equations of the PMSM and conventional field-oriented control. Section III elaborates on the NN control method. Section IV shows how to train an NN based on ADP to implement vector control for a PMSM. Section V shows how to integrate NN control in a nested-loop PMSM control configuration. Sections VI and VII compare the performance of conventional and NN vector-control schemes through simulation and hardware experiments. Finally, this paper concludes with a summary of the main points.

II. Conventional Vector Control

A. PMSM Model

The conventional field-oriented vector control is based on the Park transformation. Using the motor sign convention, this yields the stator voltage equation [28] as

$$\begin{pmatrix} v_{sd} \\ v_{sq} \end{pmatrix} = \begin{pmatrix} R_s + L_d \cdot d/dt & -\omega_e L_q \\ \omega_e L_d & R_s + L_q \cdot d/dt \end{pmatrix} \begin{pmatrix} i_{sd} \\ i_{sq} \end{pmatrix} + \begin{pmatrix} 0 \\ \omega_e \psi_f \end{pmatrix} \tag{1}$$

where R_s is the resistance of the stator winding; ω_e is the motor electrical rotational speed; v_{sd}, v_{sq}, i_{sd}, and i_{sq}, are the d and q components of instant stator voltage and current; L_d and L_q are the stator and rotor d- and q-axis inductances; and ψ_f is the flux linkage produced by the permanent magnet (PM).

The torque balance equation of a PM motor [28] is

$$\tau_{em} = J_{eq} d\omega_m/dt + B_a \omega_m + T_L \tag{2}$$

where J_{eq} is the inertia of the motor; ω_m is the motor rotational speed; B_a is the friction coefficient; T_L is the load torque; and τ_{em} is the electromagnetic drive torque. Depending on the type of a PMSM, a surface PM (SPM) or interior PM (IPM) motor, τ_{em} can be expressed as follows:

$$\tau_{em} = P\left(\psi_f i_{sq}\right) \quad \text{SPM motor} \tag{3a}$$

$$\tau_{em} = P\left(\psi_f i_{sq} + \left(L_d - L_q\right) i_{sd} i_{sq}\right) \quad \text{IPM motor} \tag{3b}$$

in which P represents the number of motor pole pairs. Lastly, the relation between ω_m and ω_e is given by

$$\omega_e = \omega_m \cdot P. \tag{4}$$

B. Conventional PMSM Vector Control

The conventional standard vector control technique usually has two distinctive nested-loop PI controllers: 1) the

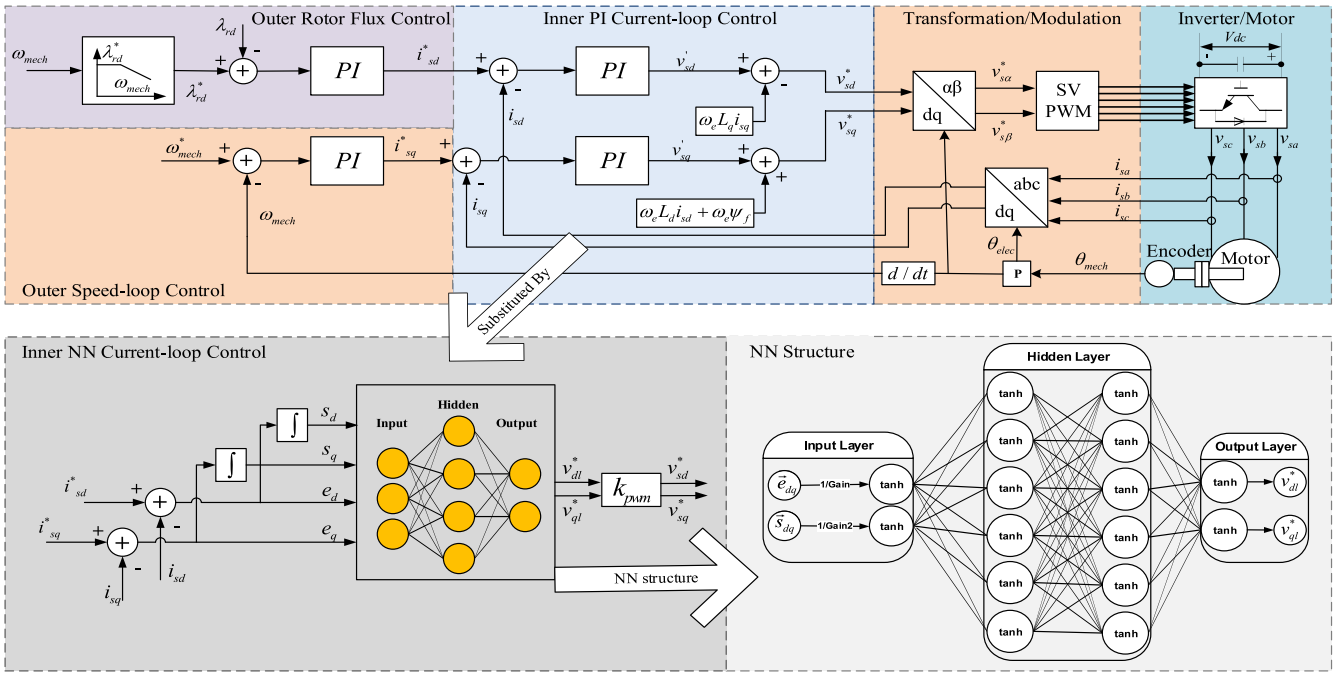


Fig. 1. PMSM conventional (top) and NN (highlighted in gray color) vector control.

outer speed and rotor flux linkage PI controllers and 2) the inner current PI controllers as shown in Fig. 1(top), where λ_{rd} and λ^*_{rd} represent the actual and reference d-axis rotor flux linkages, respectively. The current-loop control strategy is developed by rewriting (1) as (5a), where the terms in the large parentheses are used as the dynamic equations, and the other terms are compensation terms. These compensation terms are ignored when obtaining the system transfer functions [28], [29]. Thus, the consequent transfer functions, $1/(R_S+s\cdot L_d)$ and $1/(R_S+s\cdot L_q)$, are used to tune a conventional d- or q-axis PI controller. The omission of the compensation terms in deriving the transfer functions generates a decoupling inaccuracy. After the d- and q-axis PI controllers are tuned, the compensation terms are added back to the output of the PI controllers, to form the final current-loop control configuration [Fig. 1(top)]

$$v_{sd} = \underbrace{(R_s i_{sd} + L_d di_{sd}/dt)}_{v'_{sd}} - \underbrace{\omega_e L_q i_{sq}}_{\text{Comp.Term}} \tag{5a}$$

$$v_{sq} = \underbrace{\left(R_s i_{sq} + L_q di_{sq}/dt\right)}_{v'_{sq}} + \underbrace{\omega_e L_d i_{sd} + \omega_e \psi_f}_{\text{Comp.Term}}. \tag{5b}$$

The design of the speed-loop controller is based on the transfer function obtained from (2) and (3a), which is $\psi_f.P/(B_a + s\cdot J_{eq})$. Details about how to tune both current- and speed-loop controllers are presented in Section V.

III. NN Vector Control

To address the decoupling inaccuracy associated with the conventional standard vector control, a novel NN controller is proposed to replace the PI-based current controller. The NN controller, known here as the action network, is implemented as shown in Fig. 1(lower right). The outer speed loop remains unchanged. The design stages of the NN controller are analogous to the design stages of a conventional controller. First, a dynamic model of the plant is needed. Second, the NN structure needs to be specified which is analogous to specifying a conventional controller structure. Third, the NN needs to be trained, which is analogous to tuning a conventional controller.

A. State-Space Model of PMSM Current Loop

The NN current vector controller is developed using a state-space model of the PMSM, by rearranging (5a) into the standard state-space form, as shown by

$$\frac{d}{dt}\begin{pmatrix} i_{sd} \\ i_{sq} \end{pmatrix} = -\begin{pmatrix} R_s/L_d & -\omega_e L_q/L_d \\ \omega_e L_d/L_q & R_s/L_q \end{pmatrix}\begin{pmatrix} i_{sd} \\ i_{sq} \end{pmatrix} + \begin{pmatrix} v_{sd}/L_d \\ v_{sq}/L_q - \omega_e \psi_f/L_q \end{pmatrix} \tag{6}$$

where the system states are i_{sd} and i_{sq}. The PM flux ψ_f is assumed to be constant, and the converter output voltages v_{sd} and v_{sq} are proportional to the control voltage generated by the action network [30].

Since the NN controller is a digital controller, a discrete equivalent of the continuous state-space model is required. This is obtained by a zero- or first-order hold discrete equivalent mechanism. This transformation yields

$$\begin{pmatrix} i_{sd}(kT_s + T_s) \\ i_{sq}(kT_s + T_s) \end{pmatrix} = \mathbf{A}\begin{pmatrix} i_{sd}(kT_s) \\ i_{sq}(kT_s) \end{pmatrix} + \mathbf{B}\begin{pmatrix} v_{sd}(kT_s) - 0 \\ v_{sq}(kT_s) - \omega_e \psi_f \end{pmatrix} \tag{7}$$

in which T_s is the sampling period, $\mathbf{A}$ is the system matrix, and $\mathbf{B}$ is the input matrix. Since T_s is present on both sides, (7) can be simplified as

$$\vec{i}_{sdq}(k+1) = \mathbf{A}\cdot\vec{i}_{sdq}(k) + \mathbf{B}\cdot\left(\vec{v}_{sdq}(k) - \vec{v}_{rdq}\right) \tag{8}$$

where k is an integer time step, $\vec{i}_{sdq} = (i_{sd}, i_{sq})^T$, $\vec{v}_{sdq} = (v_{sd}, v_{sq})^T$ is the control action, and $\vec{v}_{rdq} = (0, \omega_e \psi_f)^T$ represents the induced voltage of the PM.

B. NN Structure

The NN has a feedforward network structure, consisting of four different layers, namely an input layer, two hidden layers, and an output layer [Fig. 1(lower right)]. The input layer contains four inputs. Two of these inputs comprise the vector $\vec{e}_{dq}(k)$, the error term, and the other two comprise $\vec{s}_{dq}(k)$, the integral of the error term. These two terms are defined by

$$\vec{e}_{dq}(k) = \vec{i}_{sdq}(k) - \vec{i}^*_{sdq}(k), \vec{s}_{dq}(k) = \vec{s}_{dq}(k-1) + \vec{e}_{dq}(k) \cdot T_s \tag{9}$$

where $\vec{i}^*_{sdq}(k)$ is the reference dq current and $\vec{s}_{dq}(k)$ is the discrete integral of the error term obtained by the forward rectangle rule. A two-hidden-layer NN was selected because it generally yields a stronger approximation ability [31] than a one-hidden-layer NN. The number of nodes in each hidden layer was selected via the trial-and-error method. We also investigated NNs with more hidden layers and more nodes in each hidden layer—but no major improvement was found.

As shown by (9), the NN has the same input signals, error terms, and integrals of error terms, and same output signals as those used in a conventional PI controller. Hence, the NN-based controller can be considered as a "super-PI" controller for a more stable and reliable PMSM operation, which can be conveniently applied to an existing PMSM digital control system. The four inputs to the NN are first divided by their appropriate gains and, then, processed through a hyperbolic tangent function, as shown in Fig. 1. The input layer then feeds into the hidden layers, each of which contains six nodes. Each node uses a hyperbolic-tangent activation function. Finally, the output layer gives $\vec{v}^*_{sdq}(k)$, the output of the NN. This output is multiplied by a gain, k_{PWM}, which represents the pulse-width modulation (PWM) [28], [30], to obtain the final control action applied to the PMSM, $\vec{v}_{sdq}$, given by

$$\vec{v}_{sdq}(k) = k_{\mathrm{PWM}} \cdot A\big(\vec{e}_{dq}(k), \vec{s}_{dq}(k), \vec{w}\big) \tag{10}$$

where $\vec{w}$ is the network's overall weight vector, and $A(\vec{e}_{dq}(k), \vec{s}_{dq}(k), \vec{w})$ denotes the whole action network. The division of the inputs by *Gain* and *Gain*2 in the NN input layer is to avoid the input saturation [32].

IV. Training the NN to Control the PMSM

A. ADP-Based Control Formulation for PMSM

ADP is a very useful tool for solving and approximating an optimal control of a dynamic system [33]. A typical ADP-based control problem consists of a cost function and a mechanism that can minimize the cost function to achieve the ADP-based control. The cost function is used to measure the performance of the ADP-based control in tracking a target trajectory and is typically defined as [33]

$$C(\vec{x}(j), j) = \sum_{k=j}^{N} \gamma^{k-j} \cdot U(\vec{x}(k), \vec{u}(k), k) \tag{11}$$

where N is the trajectory length; γ is the discount factor ($0 \leq \gamma \leq 1$); $\vec{x}(k)$ signifies the states of a dynamic system; $\vec{u}(k)$ denotes the control action applied to the system; and $U(\bullet)$ is the utility function. The cost function $J(\bullet)$, dependent upon the initial time j and the initial state $\vec{x}(j)$, is referred to as the cost-to-go of state $\vec{x}(j)$ in an ADP problem.

For vector control of a PMSM, the system state is $\vec{i}_{sdq}(k)$ and the control action is $\vec{v}_{sdq}(k)$ according to Section III. The goal of the current control for a PMSM is to track any specified target trajectory $\vec{i}^*_{sdq}(k)$ as close as possible. Thus, the utility function in (11) for the PMSM vector control problem is defined as

$$U\big(\vec{i}_{sdq}, \vec{v}_{sdq}, k\big) = \sqrt{\big(i_{sd}(k) - i^*_{sd}(k)\big)^2 + \Big(i_{sq}(k) - i^*_{sq}(k)\Big)^2}. \tag{12}$$

In this paper, we choose $\gamma = 1$, which makes the cost-to-go function of the ADP-based PMSM control problem as

$$C\big(\vec{i}_{sdq}, j\big) = \sum_{k=j}^{N} \sqrt{\big[i_{sd}(k) - i^*_{sd}(k)\big]^2 + \Big[i_{sq}(k) - i^*_{sq}(k)\Big]^2}. \tag{13}$$

The objective of training the NN controller for a PMSM is to have the NN output control actions $\vec{v}_{sdq}(k)$, $k = j, j+1, \ldots, N$ so that the cost-to-go function $C(\bullet)$ of (13) is minimized. It is worth pointing out that although the control action $\vec{v}_{sdq}(k)$ is not involved in (13), it affects (13) indirectly through (8). This impact is considered in the training of the NN as shown in the following section.

B. NN Training Mechanism

The NN is trained to approximate optimal control by using gradient descent to adjust the weights of the NN until its outputs minimize (13). As shown in Fig. 1, the NN receives the dq current feedback signal from the PMSM. Thus, the output control action of the NN at time step k changes the output current of the PMSM at time step $k+1$ via (7) or (8), the output motor current then changes NN inputs at time step $k+1$ via (9) and, then, the NN output control action at time step $k+1$ is modified via (10). This recursive process continues, making the combined system of the PMSM + NN similar to a recurrent NN. This combined "recurrent network" is shown in Fig. 2, unrolled in time, illustrating how the PMSM and the NN interact with each other. The recurrence in this architecture needs to be fully considered when computing the gradient of the ADP cost function (11) so as to enable learning by the gradient descent [34]. Doing so allows the trained NN to gain strong multistep-ahead predictive control ability that is much more powerful than the conventional predictive controllers.

Learning was accelerated using the Levenberg–Marquardt (LM) optimization method [35]. The LM algorithm has been widely used to train feedforward networks, and provides a nice compromise between the speed of Newton's method and the guaranteed convergence of the steepest descent. For a moderate number of network weights, LM appears to be one of the fastest NN training algorithms. However, since the NN and PMSM are treated as a recurrent

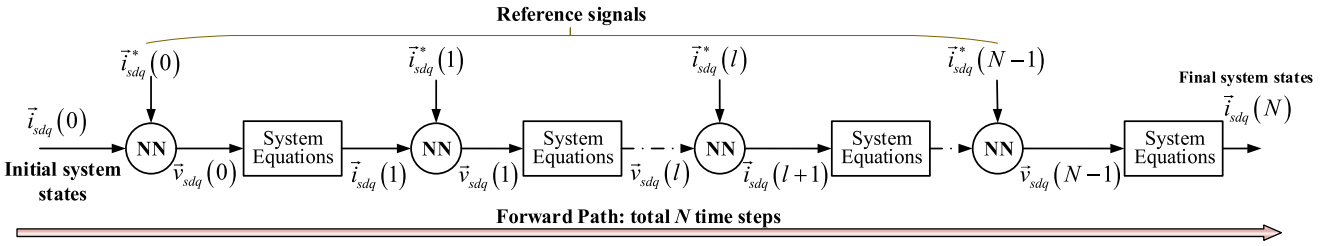


Fig. 2. Feedback loop between the NN and the PMSM [via the system equations (8), and via the neural inputs (9)]. The combined PMSM+NN system is treated as a recurrent NN, shown unrolled in time here.

network, it is necessary to modify the LM algorithm slightly using the method detailed by Fu *et al.* [32] and summarized below.

First, the gradient of (13) is computed with respect to the weight vector $\partial C/\partial \vec{w}$. In the matrix form this is

$$\frac{\partial C}{\partial \vec{w}} = \frac{\partial \sum_{k=1}^{N}(V(k))^2}{\partial \vec{w}} = \sum_{k=1}^{N} 2V(k)\frac{\partial V(k)}{\partial \vec{w}} = 2J_p(\vec{w})^T V \tag{14}$$

where $V(k) = \sqrt{U(\vec{i}_{sdq}, \vec{v}_{sdq}, k)}$, V is a vector containing $V(1)$ to $V(N)$, and $J_p(\vec{w})$ is a Jacobian matrix defined for a recurrent NN by

$$J_p(\vec{w}) = \begin{bmatrix} \frac{\partial V(1)}{\partial w_1} & \cdots & \frac{\partial V(1)}{\partial w_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial V(N)}{\partial w_1} & \cdots & \frac{\partial V(N)}{\partial w_M} \end{bmatrix}, \quad V = \begin{bmatrix} V(1) \\ \vdots \\ V(N) \end{bmatrix}. \tag{15}$$

Then, using these definitions, the weight update is applied as

$$\Delta \vec{w} = -\left[J_p(\vec{w})^T J_p(\vec{w}) + \mu \mathbf{I}\right]^{-1} J_p(\vec{w})^T V \tag{16a}$$

$$\vec{w}_{\text{update}} = \vec{w} + \Delta \vec{w} \tag{16b}$$

where $\mathbf{I}$ is the identity matrix and μ is a scalar that is dynamically adjusted by the LM algorithm [35].

As (14)–(16a) show, the Jacobian matrix, $J_p(\vec{w})$, is the kernel for training used by the LM method. To efficiently compute the Jacobian matrix, we used a forward accumulation through time (FATT) algorithm, described in detail by Fu *et al.* [32]. FATT is analogous to the better-known backpropagation through time algorithm [36]; but with the differences that FATT accumulates its result via a forward-mode automatic differentiation [37], and also that FATT delivers a whole Jacobian matrix as opposed to a single gradient vector. Please note that the use of FATT, in this case, is chosen merely because it is slightly more computationally efficient than computing the Jacobian matrix by a backward accumulation, but gives exactly the same result subject to floating-point rounding errors. It should be emphasized that to correctly compute $J_p(\vec{w})$ by FATT, it is necessary to differentiate through the known motor model equations (8), and feed these derivatives into the next time step's neural inputs, via (9) and (10), and the chain-rule, and ultimately feed this chain of derivatives into the accumulating cost function (13), at each subsequent time step. Fuller details are given by Fu *et al.* [32].

TABLE I
PMSM DATA USED IN SIMULATION/EXPERIMENTAL STUDY

	Parameter	Simulation	Hardware	Units
Motor	Rated Power	50	0.24	kW
	Nominal Speed	1200	2800	RPM
	Nominal Torque	250	1.5	N·m
	Maximum Speed	6000	3800	RPM
	Permanent magnet flux	0.1758	0.01544	Wb
	Inductance in *q-axis*, L_q	1.598	0.255	mH
	Inductance in *d-axis*, L_d	1.598	0.255	mH
	Stator copper resistance, R_s	0.0065	0.22	Ω
	Inertia	0.089	0.0004	kg·m^2
	Friction coefficient	0.1	0.001	N·m·s/rad
	Pole pairs	4	4	
Inverter	Inverter rating	60	0.4	kVA
	dc voltage	500	42	V
	Switching frequency	6	10	kHz

V. TRAINING/TUNING PMSM CURRENT- AND SPEED-LOOP CONTROLLERS FOR SIMULATION AND HARDWARE CASES

The PMSM nested-loop control has been considered in two SPM motor cases: one for simulation and one for hardware experiment. The simulation case uses parameters of a PMSM that are typical for an electric vehicle application [38]. The hardware experiment is based on a laboratory PMSM [39], which has a smaller power rating and is mainly used for the purpose of experimental validation. Table I shows the PMSM parameters used in each case.

A. Tuning Speed-Loop Controller

The PI parameters of the speed-loop controller are tuned using the PI-derivative (PID) tuner function within the PID controller block in MATLAB. Fig. 3 shows the closed-loop Simulink model used to tune the speed-loop PI parameters. The transfer function in Fig. 3 is $\psi_f \cdot P/(B_a + s \cdot J_{eq})$ [derived from (2) and (3a) in Section II-A], where B_a was set to zero. The phase margin was 60°, while the controller bandwidth in terms of angular frequency was chosen as 200 rad/s. Then, the PI gains were adjusted until a better speed tracking performance was achieved. Note that both the NN and conventional controllers use the same speed-loop PI gains.

B. Tuning Conventional Current-Loop Controller

The PI gains of the conventional current-loop controller were also tuned using the PID tuner function. The transfer

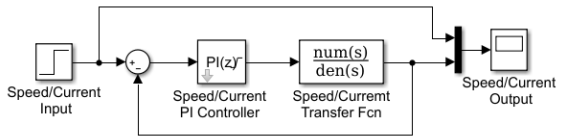


Fig. 3. Using Simulink to tune the speed- and current-loop controllers.

function in Fig. 3 is $1/(R_S + s \cdot L_d)$ or $1/(R_S + s \cdot L_q)$ (see Section II-B). The current controller bandwidth in terms of angular frequency was 2000 rad/s while the phase margin was kept the same as that of the speed-loop controller. Since the current controller is in the inner loop, the bandwidth has to be larger to track the current well. Similarly, the PI gains were then adjusted until a better tracking performance was obtained.

C. Training Neural Network Controllers

For both simulation and hardware experimental cases, an NN was trained using the method of Section IV and the motor parameters of Table I. The training procedure is as follows.

1) Randomly generate changing sample reference dq current trajectories.
2) Randomly generate a sample initial state i_{sdq} (1).
3) Unroll the PMSM current trajectory from the initial state.
4) Train the NN as detailed in Section IV.
5) Repeat the process for all of the reference dq current trajectories and sample initial states until reaching a stop criterion.

Each initial state i_{sdq} (1) was generated randomly within acceptable d- and q-axis current ranges that are within the rated current limit in terms of the current amplitude and also cannot cause the controller to operate beyond the PWM saturation limit, i.e.,

$$\sqrt{I_{sd}^2 + I_{sq}^2} \le I_{sdq\_\text{rated}}, \sqrt{V_{sd}^2 + V_{sq}^2} \le V_{sdq\_\max} \qquad (17)$$

where I_{sdq_rated} and $V_{sdq\_\max}$ denote the rated motor dq current amplitude and maximum dq voltage amplitude that can be applied to the motor due to the motor inverter PWM saturation constraint, respectively. Each trajectory was unrolled for a duration of 1 s, with a sampling time of $T_s = 0.1$ ms, and the reference dq current was changed randomly every 0.1 s also within acceptable d- and q-axis current ranges. All network weights were initially randomized using a uniform distribution within ± 0.1, and ten randomized reference current trajectories were created during each training epoch. Fig. 4 shows a successful training convergence. Note that the NN is trained offline, and no training occurs in the real-time control stage.

VI. Performance and Stability Evaluation of Conventional and NN Vector Controllers Using SimPowerSystems

A simulation model of the PM motor drive was developed using MATLAB SimPowerSystems based on the parameters of the simulated PMSM in Table I. Fig. 5 shows the simulation

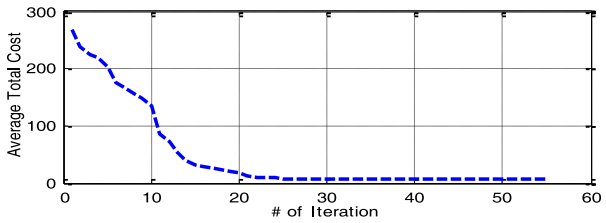


Fig. 4. Average cost per trajectory time step for training NN.

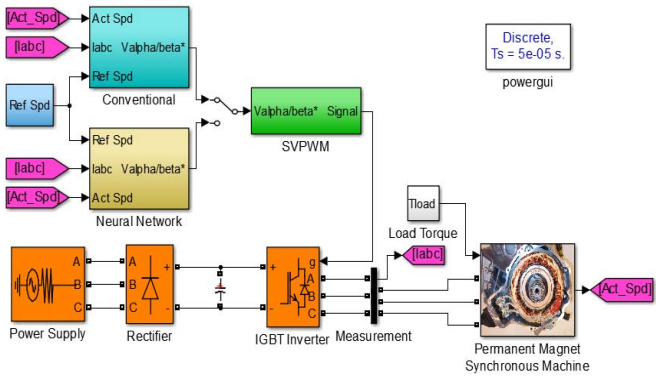


Fig. 5. PMSM motor control configuration in SimPowerSystems.

model containing conventional and NN controllers as shown in Fig. 1. The controller sampling rate is 0.1 ms. Details about how to build a simulation model using SimPowerSystems can be found in [40]. In the MATLAB environment, the computing time for each control action of the NN takes about 20 μs. This execution time would reduce further when the NN is implemented on a DSP chip.

The stability of the NN was evaluated against a testing dataset, as is a common practice in the NN field [41]. The training and testing datasets represent different sets of trajectories. The testing dataset covers an extremely wide range of circumstances to validate the NN controller over various key criteria. These include speed control; current control; robustness of speed and current control; robustness against load disturbance and fluctuations in flux-linkage; improved tolerance to sampling time variations, etc., as shown in Sections VI-A–VI-G.

A. Speed Control Evaluation

Fig. 6 compares the motor speed control using conventional and NN control techniques, in which the friction factor and the load torque are zero. The motor starts with a reference rotational speed increasing linearly from 0 rad/s to 60 rad/s, stays at 60 rad/s for about 0.75 s and, then, reduces to 40 rad/s. At $t = 2$ s, the reference speed increases to 80 rad/s and, then, remains at 80 rad/s. The reference d-axis current is 0 A. Both traditional and NN controllers can track the reference speed properly. But, for each reference speed increase or decrease, the traditional controller shows more overshoot and oscillations, particularly in motor current [Fig. 6(b) and (c)], implying that there are more torque oscillations when using the conventional controller. Note: the reference q-axis

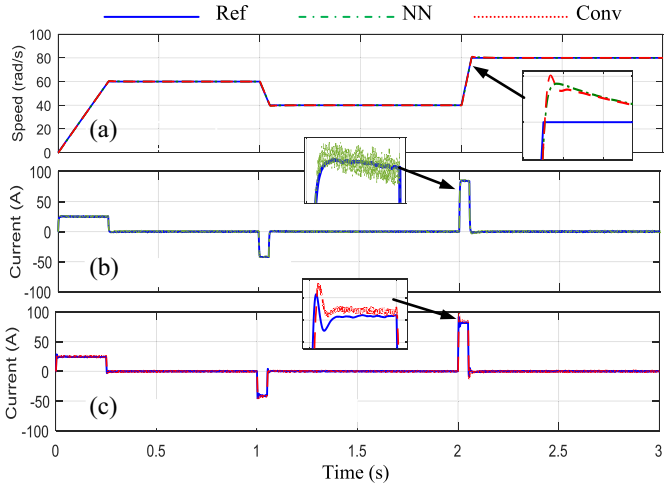


Fig. 6. Speed tracking control—NN versus conventional. (a) Motor rotational speeds. Reference/actual q-axis currents using (b) NN control and (c) conventional control.

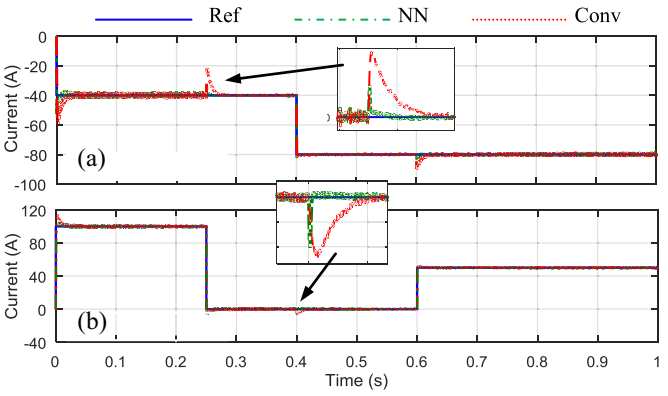


Fig. 7. Current tracking control—NN versus conventional. (a) d-axis current. (b) q-axis current.

current is generated by the speed-loop controller as shown in Fig. 1.

B. Current Control Evaluation

In the motor drive industry, evaluation of the current-loop controllers is normally conducted under the condition that the speed of the test motor is kept constant and the motor is evaluated while tracking reference d- and q-axis currents. This can be achieved, as in Fig. 5, by changing the load torque block to speed block and setting the PMSM to operate according to a specified constant speed value. Fig. 7 compares current-control performance using conventional and NN controllers. The initial d-axis reference current is -40 A and changes to -80 A at 0.4 s. The initial q-axis reference current is 100 A and changes to 0 A at 0.25 s and then to 50 A at 0.6 s. As shown in Fig. 7, the NN controller is more stable and reliable, and responds faster than the conventional controller.

C. Robustness of Speed-Loop Controller

In practical applications, both motor inertia and friction factor may change depending on the load of the EV and road

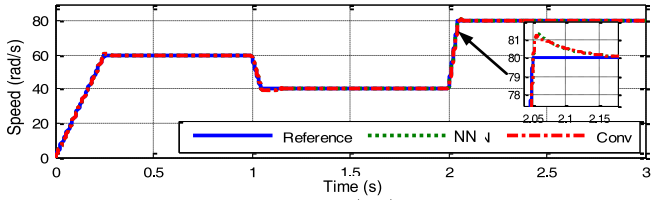


Fig. 8. NN versus conventional. Reference and actual motor rotational speeds for a higher motor inertia and friction factor.

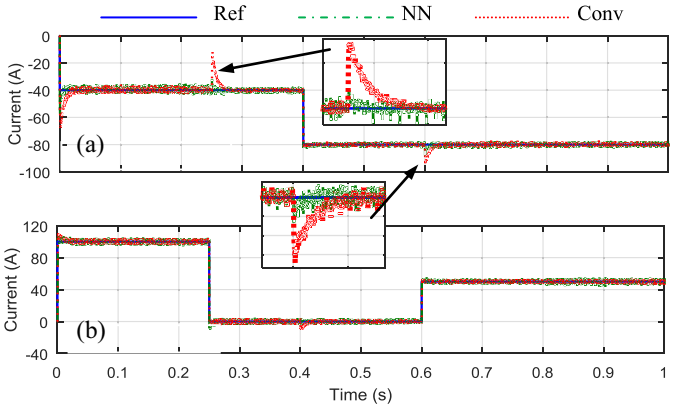


Fig. 9. NN versus conventional for 40% decrease of motor resistance/inductance. (a) d-axis current. (b) q-axis current.

conditions. This will affect the performance of the motor speed-loop controller. Fig. 8 compares the conventional and NN vector-control methods when the friction factor changes from 0 N·m·s in Fig. 6 to 0.2 N·m·s and the inertia is tripled, while the other conditions are the same as those used in Fig. 6, except that a load torque of 20 N·m is included. In general, the speed tracking control is not affected much and shows a similar performance using both NN and conventional vector controllers. Compared to Fig. 6, a little more time is needed for the transition from one reference speed to another, due to a larger inertia and friction factor.

D. Robustness of Current-Loop Controller

In real applications, the motor resistance and inductance may deviate from their nominal values by a significant amount. This affects the PM motor current-loop controllers. Fig. 9 demonstrates what happens when both motor resistance and inductance are reduced by 40%, with all other conditions being the same as those used in Fig. 7. The results show the NN controller is better able to track the reference and actual d- and q-axis motor currents under variable parameter conditions, and is more stable and reliable than the conventional controller.

E. Impact of Rotor Magnet Flux Linkage

In a PM motor, the rotor-magnet flux linkage may change due to an increase or decrease in the motor temperature. This would affect the performance of the motor. A test was carried out to evaluate the conventional and NN vector-control methods when the rotor magnet flux linkage is lower or higher than the nominal value listed in Table I. Fig. 10 compares the

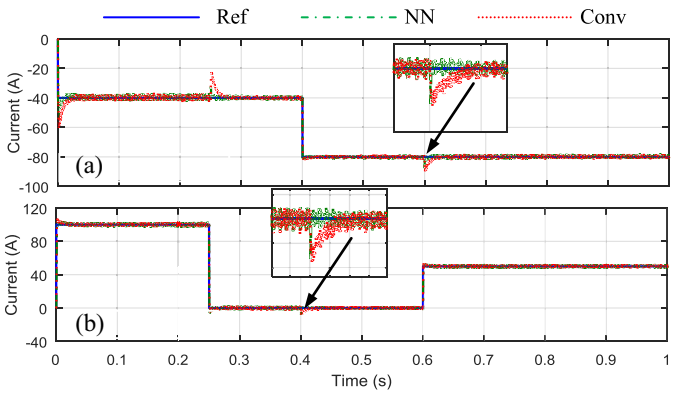


Fig. 10. NN versus conventional for 40% increase of rotor flux linkage. (a) d-axis current. (b) q-axis current.

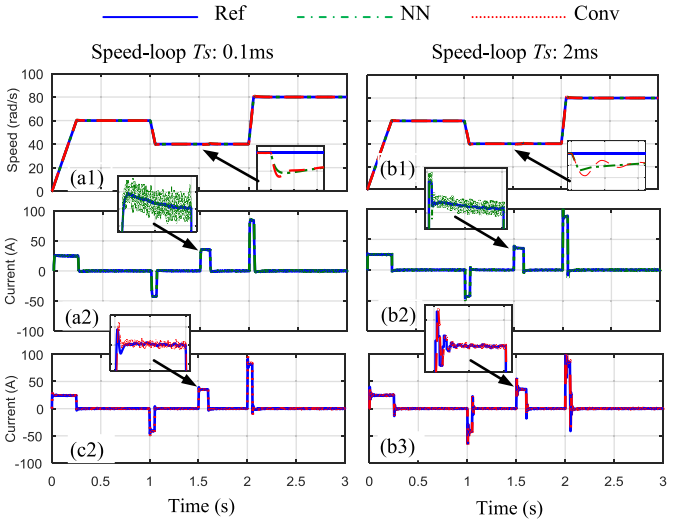


Fig. 11. NN versus conventional—load disturbance and speed-loop sampling time impacts. (a1) and (b1) Reference/actual motor rotational speeds. Reference/actual q-axis currents using (a2) and (b2) NN control and (a3) and (b3) conventional PI control.

current tracking using the conventional and NN control methods, when the rotor magnet flux linkage increases by 40%, while the other conditions remain the same as those used in Fig. 7. The study shows a better dynamic response of d- and q-axis current tracking using the NN controller.

F. Impact of Load Disturbance and Sampling Rate

Load disturbance and sampling rate impact to motor speed control were also investigated. Fig. 11(a) shows the PM motor performance, using conventional and NN controllers, under an impulse load disturbance while the other conditions are the same as those used in Fig. 6. The impulse disturbance of an additional 30 N·m appears at 1.5 s and lasts for 0.1 s. Fig. 11(b) shows the motor performance when a large sampling interval of 2 ms is applied to the speed-loop controller and to read motor position/speed data while the other conditions are the same as those used in Fig. 11(a). The sampling time for the current-loop controller is still 0.1 ms as the motor current changes much faster than the motor speed. The results

TABLE II
TRACKING ERRORS RELATED TO FIGS. 6–11: MAXIMUM, AVERAGE, AND STANDARD DEVIATION VALUES

	Figure #	Maximum		Average		Std	
		NN	Conv.	NN	Conv.	NN	Conv.
Speed tracking (rad/s)	Fig. 6(a)	1.3031	1.5132	0.0594	0.0586	0.1620	0.1585
	Fig. 8	3.5608	3.5610	0.1854	0.1851	0.4905	0.4901
	Fig. 11(a1)	1.3030	1.5133	0.0763	0.0755	0.1708	0.1676
	Fig. 11(b1)	1.7839	2.1820	0.0797	0.0790	0.1836	0.1862
Current tracking (A)	Fig. 7(a)	24.53	29.90	0.54	0.79	0.57	1.41
	Fig. 7(b)	99.25	99.58	0.79	0.86	1.65	1.71
	Fig. 9(a)	17.24	24.81	0.93	1.05	0.78	1.88
	Fig. 9(b)	99.66	100.13	1.70	1.65	1.73	1.71
	Fig. 10(a)	22.36	28.83	0.64	0.84	0.59	1.40
	Fig. 10(b)	99.77	99.86	1.19	1.18	1.68	1.73

show that the NN controller is less impacted by load disturbance and more reliable for a large sampling interval applied to the speed control loop than the conventional one.

A summary in terms of maximum, average, and standard deviation of the absolute tracking errors associated with Figs. 6–11 for the NN and conventional controllers is presented in Table II.

G. Sampling Time Impact at High Speed

The rotating speed of a PM motor is directly related to the electrical frequency of the stator voltage and current. As the rotating speed increases, the electrical frequency increases too. This requires the sampling rate to increase as the maximum demanded motor rotating speed increases. The study shows that for sampling times of 100, 80, 40, and 20 μs, respectively, the NN controller can provide stable control in terms of rotational speed up to 10 000, 11 000, 13 500, and 16 500 rpm, while the traditional controller crashes at 7500, 9000, 10 000, and 13 000 rpm, respectively. Fig. 12 shows the performance using conventional and NN controllers (including flux weakening control) for sampling times of 100 and 20 μs, respectively. The study shows that the NN controller is more stable in supporting the high-speed operation of a PM motor.

The stability of the PMSM control depends strongly on the *abc* to *dq* transformation. Especially, when the rotational speeds increase, the rotor electrical angular position would change quickly. Thus, to capture the three-phase current and the electrical angular position information correctly, a small sampling time would be needed. Otherwise, the calculated *dq* current can be distorted. Similar to Fig. 11, using the same sampling rate, the NN controller is more robust than the conventional controller at a high motor operating speed, as illustrated by Fig. 12. A tentative explanation for this is that the NN is a comparatively flexible function approximator, compared to a PI or alternative controllers, and is specifically optimized to have a fast response time and low overshoot. On the other hand, the conventional PI-based controller only has two parameters to tune and, therefore, cannot compete in fast response time and low overshoot. As a result, the NN

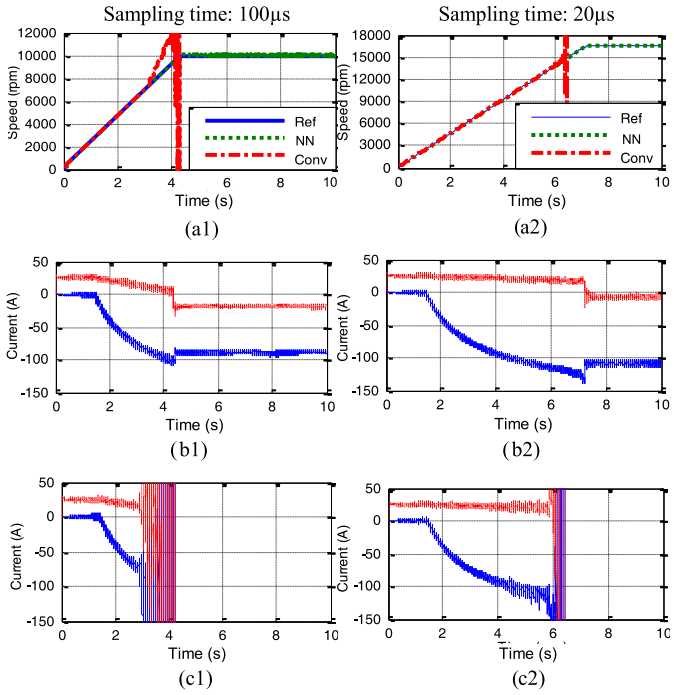


Fig. 12. NN versus conventional. Sampling impact for PM motor control toward high rotational speed. The left and right column of graphs represent a sampling time of 100 and 20 μs, respectively. (a1) and (a2) Ref/actual motor rotational speed. (b1) and (b2) NN: d- and q-axis currents. (c1) and (c2) Conv: d- and q-axis currents.

controller has a stronger ability to compensate for the dq transformation distortion caused by a low sampling rate than the conventional controller.

VII. Hardware Experiment

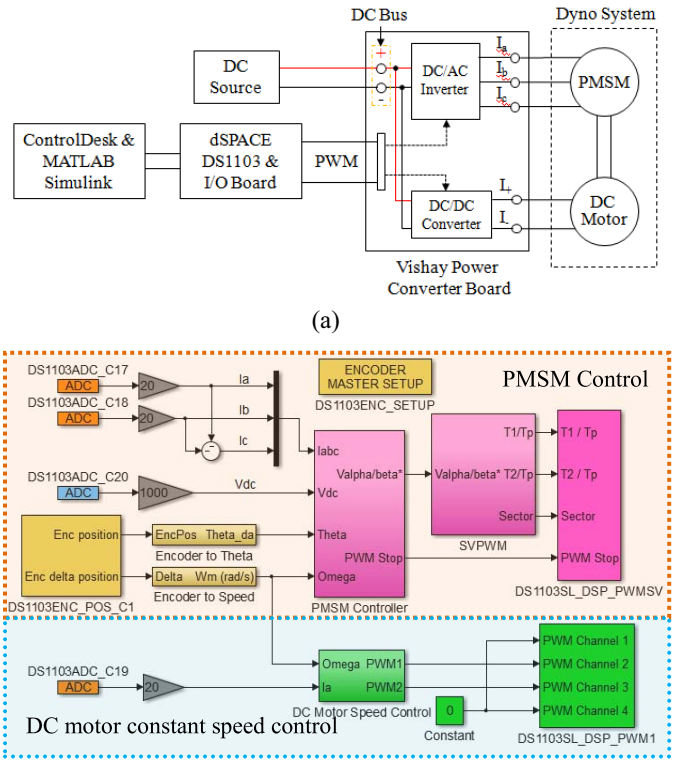
A. Experimental Setup

To further verify the feasibility and stability of the NN vector controller, a DSP-based digital control system was implemented (Fig. 13). The experimental setup [Fig. 13(a)] consists of three major parts.

1) A motor drive system containing an SPM motor from Motorsolver coupled to a dc motor [39].
2) A power converter board from Vishay HiRel systems which has two independent three-leg converters.
3) A dSPACE DS1103 controller board to collect various input signals, e.g., current, voltage, and motor speed, and to generate PWM output for controlling the SPM and dc motors.

One converter was formed as a dc/ac converter to control the SPM motor, while the other one was formed as an H-bridge dc/dc converter to control the dc motor.

The control algorithms for both SPM and dc motors were built in Simulink [Fig. 13(b)]. They were then compiled and loaded as the assembly code to the DSP chip within the DS1103 controller board. In Fig. 13(b), the measurements of the PM motor's speed and rotor position are obtained by the DS1103ENC_POS module, and the voltage and current measurements are obtained by the DS1103ADC module. The speed measurements are passed to both dc and PM



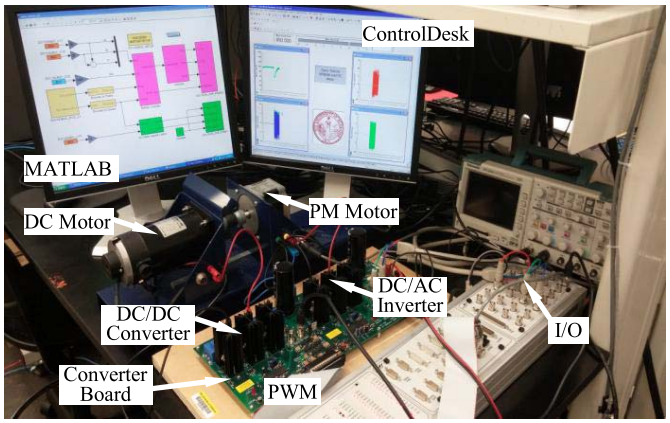


Fig. 13. Hardware laboratory testing and control systems. (a) Circuit connection. (b) dSPACE-based real-time controller. (c) Experimental setup.

motor controllers; the three-phase PM motor current measurements are passed to the PM motor controller; and the dc motor current measurement is passed to the dc motor controller.

The PMSM controller block implements either the conventional or NN vector control, according to Fig. 1, and outputs the α and β reference voltages to the space-vector modulation block, which generates T1, T2, and sector information needed by the DS1103SL_DSP_PWMSV block to generate the driving pulses. The driving pulses are applied to the three-phase dc/ac converter to control the PMSM. The dc motor controller block generates two complementary duty ratio signals that are passed to DS1103SL_DSP_PWM block to produce the

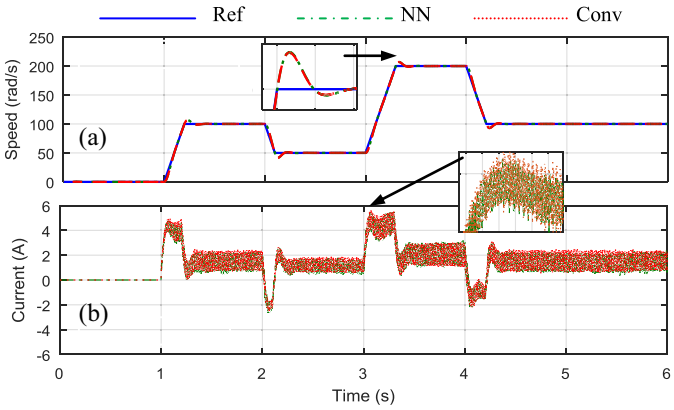


Fig. 14. NN versus conventional: simulation of laboratory PM motor. (a) Reference and actual motor rotational speeds. (b) q-axis currents.

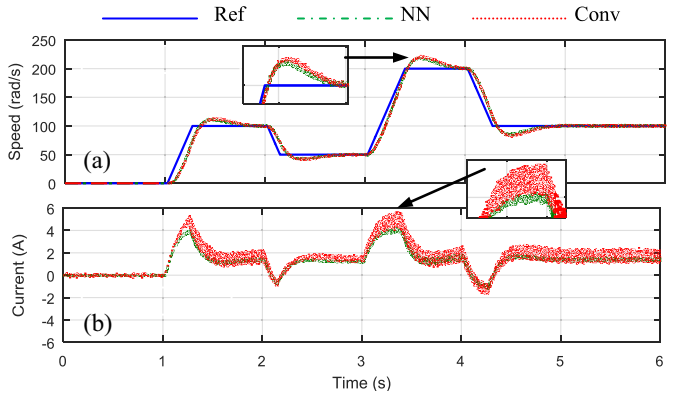


Fig. 15. NN versus conventional: hardware experiment of laboratory PM motor. (a) Reference and actual motor rotational speeds. (b) q-axis currents.

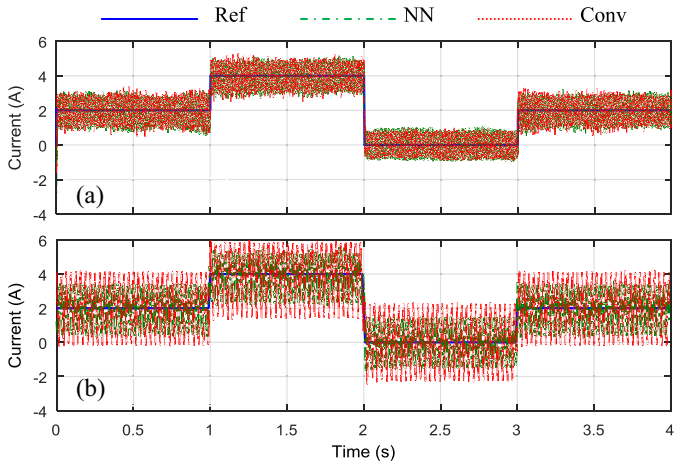


Fig. 16. NN versus conventional. (a) Simulated and (b) experiment q-axis current of laboratory PM motor.

TABLE III
TRACKING ERRORS RELATED TO FIGS. 14–16: MAXIMUM, AVERAGE, AND STANDARD DEVIATION VALUES

Figure #	Maximum		Average		Std	
	NN	Conv.	NN	Conv.	NN	Conv.
Speed tracking Fig. 14(a) (rad/s)	8.3859	8.5428	0.6709	0.6673	1.6791	1.6703
Speed tracking Fig. 15(a) (rad/s)	32.7481	36.0681	5.2326	5.8308	7.1686	8.2316
Current tracking Fig. 16(a) (A)	4.73	3.55	0.44	0.39	0.27	0.26
Current tracking Fig. 16(b) (A)	1.90	3.48	0.82	0.94	0.41	0.64

driving pulses applied to the dc/dc converter to control the dc motor.

Detailed information on how to build a hardware experiment using MATLAB Simulink and dSPACE can be found in [42]. Information about online processing capabilities of dSPACE DS1103 and its real-time coding specifications can be found in [42] and [43].

The simulation cases shown by Figs. 6–12 in Section V are divided into two categories: one corresponds to speed and current tracking control (Figs. 6, 8, 11, and 12), while the other corresponds to current tracking control under constant motor speed (Figs. 7, 9, and 10). Then, hardware experiments were performed based on the laboratory setup for each of these cases, with Section VII-B showing the results associated with the speed and current tracking control and Section VII-C presenting the results associated with current tracking under constant motor speed. The parameters of the laboratory motor are shown in Table I and the controller sampling rate is 0.1 ms. The minimum rotational speed that can be measured using the speed sensor was found to be 1.31 rad/s, or 12.5 rpm in the experimental setup. In the experimental arrangement, the SPM motor parameters could be different from the nominal values shown in Table I [44], [45] and there could be unexpected disturbances and noises.

B. PM Motor Speed and Current Control

In this test, both speed and current controls are applied to the PM motor while the dc motor is idle. The speed- and current-loop controllers of the PM motor were redesigned based on the parameters shown in Table I, and were tested first in the simulation, and then on the hardware motor. The test sequence was scheduled as follows, with $t = 0$ s as the starting point for data recording: at $t = 1$ s, the input speed increases from 0 to 100 rad/s, and then stays at 100 rad/s; at $t = 2$ s, the speed decreases to 50 rad/s and retains this value till $t = 3$ s; and at $t = 3$ s, the speed increases to 200 rad/s and decreases again to 100 rad/s at $t = 4$ s.

Fig. 14 presents the simulation results. Due to the low ratings of the laboratory motor, the relevant oscillation of the simulated stator current looks worse than that of the 50-kW PM motor in tracking the current variation in a much larger range in Section VI. The result of the hardware experiment (Fig. 15) is a little bit different from the simulation result. A potential reason is that the actual motor inertia and friction coefficient are more complicated and different from those used in the simulation. Due to uncertain motor parameters, noises and disturbances, more oscillations of the PM motor were found in the hardware experiment results (Fig. 15). Under these challenging laboratory condition, the NN controller performed better than the conventional controller (Fig. 15).

C. PM Motor Current Control Under Constant Speed

In this case, only the current control is applied to the PM motor while the dc motor is controlled to ensure the speed of the whole system constant. Fig. 16(a) shows the simulation result of the q-axis current for the laboratory PMSM operating at 100 rad/s and zero d-axis reference current. Again, the much smaller current tracking range makes the relevant oscillation of the motor current apparently worse than that of the 50-kW PM motor shown in Section VI. Fig. 16(b) presents the hardware experiment results of the q-axis current under the same condition. The NN controller clearly displays less oscillation than the conventional controller for the laboratory PM motor, showing a strong adaptive control ability of the NN controller under uncertain, noisy, and disturbing laboratory conditions. The success of the hardware experiments indicates that it is possible to implement the NN controller in a real-life PMSM.

A summary in terms of maximum, average, and standard deviation of the absolute tracking errors associated with the experiment results shown by Figs. 14 and 16 for the NN and conventional controllers is presented in Table III.

VIII. Conclusion

PMSMs are widely used in electric drive applications particularly in electric drive vehicles. This paper presents an NN-based vector-control method to overcome the limitations of conventional vector-control approaches. It describes how to achieve approximately optimal vector control using an NN, which is trained to minimize an ADP-based cost function. Compared to the conventional vector control, the NN vector controller produces the fastest response speed, lowest overshoot, and, in general, the best performance. In addition, since an NN is trained under variable system parameters, the NN-based vector controller shows the enhanced performance when the sampling time changes and system parameters are difficult to identify, especially in hardware experiment conditions. The hardware experiment confirmed that the NN-based controller is able to track reference commands while maintaining a high power quality, making it possible to implement the NN vector controller in a real PMSM environment. In hardware experimental conditions, a conventional controller usually needs to be retuned whenever the motor parameters change. In contrast, the NN-based controller retains good performance under a variety of runtime PM motor parameters, despite the NN being trained using the nominal motor parameters of Table I.

Acknowledgment

The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] C.-K. Lin, T.-H. Liu, J.-T. Yu, L.-C. Fu, and C.-F. Hsiao, "Model-free predictive current control for interior permanent-magnet synchronous motor drives based on current difference detection technique," *IEEE Trans. Ind. Electron.*, vol. 61, no. 2, pp. 667–681, Feb. 2014.

[2] T. Türker, U. Buyukkeles, and A. F. Bakan, "A robust predictive current controller for PMSM drives," *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3906–3914, Jun. 2016.

[3] Y. Ren, Z. Q. Zhu, and J. Liu, "Direct torque control of permanent-magnet synchronous machine drives with a simple duty ratio regulator," *IEEE Trans. Ind. Electron.*, vol. 61, no. 10, pp. 5249–5258, Oct. 2014.

[4] Y. Cho, K.-B. Lee, J.-H. Song, and Y. I. Lee, "Torque-ripple minimization and fast dynamic scheme for torque predictive control of permanent-magnet synchronous motors," *IEEE Trans. Power. Electron.*, vol. 30, no. 4, pp. 2182–2190, Apr. 2015.

[5] C. Xia, B. Ji, and Y. Yan, "Smooth speed control for low-speed high-torque permanent-magnet synchronous motor using proportional–integral–resonant controller," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2123–2134, Apr. 2015.

[6] M.-W. Qian, "Mixed H_2/H_∞ control of permanent magnet synchronous motor based on particle swarm optimization algorithm," *J. Comput. Appl.*, vol. 8, p. 76, 2012.

[7] S. Das and I. Pan, "On the mixed H_2/H_∞ loop-shaping tradeoffs in fractional-order control of the AVR system," *IEEE Trans. Ind. Inf.*, vol. 10, no. 4, pp. 1982–1991, Nov. 2014.

[8] A.A. Stoorvogel, *The H_∞ Control Problem: A State Space Approach*, Univ. Michigan, Ann Arbor, MI, USA, Feb. 2000. [Online]. Available: http://wwwhome.math.utwente.nl/∼stoorvogelaa/book2.pdf

[9] Z. Li *et al.*, "Mixed H_2/H_∞ optimal control for three-phase grid connected converter," *Int. J. Electron.*, vol. 104, no. 6, pp. 775–791, 2017.

[10] B. Akin and M. Bhardwaj, *Sensorless Field Oriented Control of 3-Phase Permanent Magnet Synchronous Motors*, Texas Instrum. Inc., Dallas, TX, USA, 2013. [Online]. Available: http://www.ti.com/lit/an/sprabq3/sprabq3.pdf

[11] R. Filka and M. Stulrajter, *3-Phase PMSM Motor Control Kit With the MPC5604P*, Freescale Semicond. Inc., Austin, TX, USA, 2015. [Online]. Available: https://cache.freescale.com/files/product/doc/AN1931.pdf

[12] S. Li, T. A. Haskew, and L. Xu, "Conventional and novel control designs for direct driven PMSG wind turbines," *Elect. Power Syst. Res.*, vol. 80, no. 3, pp. 328–338, Mar. 2010.

[13] M. A. Rahman and M. A. Hoque, "On-line adaptive artificial neural network based vector control of permanent magnet synchronous motors," *IEEE Trans. Energy Convers.*, vol. 13, no. 4, pp. 311–318, Dec. 1998.

[14] Y. Yi, D. M. Vilathgamuwa, and M. A. Rahman, "Implementation of an artificial-neural-network-based real-time adaptive controller for an interior permanent-magnet motor drive," *IEEE Trans. Ind. Appl.*, vol. 39, no. 1, pp. 96–104, Jan./Feb. 2003.

[15] T. Pajchrowski and K. Zawirski, "Application of artificial neural network to robust speed control of servodrive," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 200–207, Feb. 2007.

[16] C. Xia, C. Guo, and T. Shi, "A neural-network-identifier and fuzzy-controller-based algorithm for dynamic decoupling control of permanent-magnet spherical motor," *IEEE Trans. Ind. Electron.*, vol. 57, no. 8, pp. 2868–2878, Aug. 2010.

[17] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, 4th ed. Belmont, MA, USA: Athena Science, 2012.

[18] F. L. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Hoboken, NJ, USA: Wiley, 2012, pp. 474–493.

[19] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[20] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch, "Adaptive critic designs: A case study for neurocontrol," *Neural Netw.*, vol. 8, no. 9, pp. 1367–1372, Dec. 1995.

[21] Y. Yang, D. C. Wunsch, and Y. Ying, "Hamiltonian-driven adaptive dynamic programming for continuous nonlinear dynamical systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1929–1940, Aug. 2017.

[22] G. Xiao, H. Zhang, Y. Luo, and Q. Qu, "General value iteration based reinforcement learning for solving optimal tracking control problem of continuous-time affine nonlinear systems," *Neurocomputing*, vol. 245, pp. 114–123, Jul. 2017.

[23] Q. Wei, G. Shi, R. Song, and Y. Liu, "Adaptive dynamic programming-based optimal control scheme for energy storage systems with solar renewable energy," *IEEE Trans. Ind. Electron.*, vol. 64, no. 7, pp. 5468–5478, Jul. 2017.

[24] Q. Wei, D. Liu, and G. Shi, "A novel dual iterative *Q*-learning method for optimal battery management in smart residential environments," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2509–2518, Apr. 2015.

[25] Q. Wei, D. Liu, G. Shi, and Y. Liu, "Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming," *IEEE Trans. Ind. Electron.*, vol. 62, no. 7, pp. 4203–4214, Jul. 2015.

[26] S. Li, M. Fairbank, X. Fu, D. C. Wunsch, and E. Alonso, "Vector control of permanent magnet synchronous motor using adaptive recurrent neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Dallas, TX, USA, Aug. 2013, pp. 1–8.

[27] D. V. Prokhorov, L. A. Feldkamp, and I. Y. Tyukin, "Adaptive behavior with fixed weights in RNN: An overview," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2002, pp. 2018–2023.

[28] N. Mohan, *Advanced Electric Drives—Analysis, Modeling and Control Using Simulink*. Minneapolis, MN, USA: Minnesota Power Electron. Res. Educ., 2001.

[29] S. Li, T. A. Haskew, E. Muljadi, and C. Serrentino, "Characteristic study of vector-controlled direct-driven permanent magnet synchronous generator in wind power generation," *Elect. Power Compon. Syst.*, vol. 37, no. 10, pp. 1162–1179, Oct. 2009.

[30] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters, Applications, and Design*, 3rd ed. New York, NY, USA: Wiley, Oct. 2002.

[31] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA, USA: PWS, 2002.

[32] X. Fu, S. Li, M. Fairbank, D. C. Wunsch, and E. Alonso, "Training recurrent neural networks with the Levenberg–Marquardt algorithm for optimal control of a grid-connected converter," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 1900–1912, Sep. 2014.

[33] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[34] M. Fairbank, S. Li, X. Fu, E. Alonso, and D. C. Wunsch, "An adaptive recurrent neural-network controller using a stabilization matrix and predictive inputs to solve a tracking problem under disturbances," *Neural Netw.*, vol. 49, pp. 74–86, Jan. 2014.

[35] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[36] P. Werbos, "Backwards differentiation in AD and neural nets: Past links and new opportunities," in *Automatic Differentiation: Applications, Theory, and Implementations*. Heidelberg, Germany: Springer, 2006, pp. 15–34.

[37] L. B. Roll, *Automatic Differentiation Techniques and Applications* (Lecture Notes in Computer Science), G. Goos and J. Hartmanis, Eds. Heidelberg, Germany: Springer, 1981.

[38] O. Tremblay and L. A. Dessaint, *Hybrid Electric Vehicle Power Train Using Battery Model*, MathWorks, Inc., Natick, MA, USA, 2010.

[39] MotorSolver, *Permanent Magnet Brushless Motor (Fitted With 1000 Line Encoder) for DYNO-KITS Used for Teaching Labs*, Univ. Minnesota, Minneapolis, MN, USA. [Online]. Available: http://people.ece.umn.edu/groups/power/mat_lab.html

[40] The MathWorks. *Model and Simulate Electrical Power Systems*. Accessed: Jan. 2019. [Online]. Available: https://www.mathworks.com/products/simpower.html

[41] S. Haykin, *Neural Networks—A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[42] A. Thomas, "dSPACE DS1103 control workstation tutorial and DC motor speed control," ECE Dept., Bradley Univ., Peoria, IL, USA, Rep., May 2009. [Online]. Available: http://ee.bradley.edu/projects/proj2009/dscntrl/Tutorial.pdf

[43] *DS1103 PPC Controller Board*. Accessed: Jan. 2019. [Online]. Available: http://www.ceanet.com.au/Portals/0/documents/products/dSPACE/dspace_2008_ds1103_en_pi777.pdf

[44] S. J. Underwood and I. Husain, "Online parameter estimation and adaptive control of permanent-magnet synchronous machines," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2435–2443, Jul. 2010.

[45] D. Yang, H. Mok, J. Lee, and S. Han, "Adaptive torque estimation for an IPMSM with cross-coupling and parameter variations," *Energies*, vol. 10, no. 2, p. 167, 2017.

Shuhui Li (S'99–M'99–SM'08) received the B.S. and M.S. degrees from Southwest Jiaotong University, Chengdu, China, in 1983 and 1988, respectively, and the Ph.D. degree from Texas Tech University, Lubbock, TX, USA, in 1999, all in electrical engineering.

He was with the School of Electrical Engineering, Southwest Jiaotong University, from 1988 to 1995, where his research interests included electrified railways, power electronics, power systems, and power system harmonics. From 1995 to 1999, he was engaged in research on wind power, artificial neural networks, and applications of massive parallel processing. He joined Texas A&M University, Kingsville, TX, USA, as an Assistant Professor, in 1999, and an Associate Professor in 2003. He joined the University of Alabama, Tuscaloosa, AL, USA, as an Associate Professor in 2006. His current research interests include renewable energy systems, power electronics, power systems, electric machines and drives, and applications of artificial neural networks in energy systems.

Hoyun Won (S'12) received the B.S. and M.S. degrees in electrical engineering from the University of Alabama, Tuscaloosa, AL, USA, in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the University of Alabama.

His current research interests include permanent magnet machine design and drives, micromagnetic simulation, and antenna.

Xingang Fu (S'14–M'15) received the B.S. and M.S. degrees in applied mathematics from the Ocean University of China, Qingdao, China, in 2005 and 2009, respectively, and the Ph.D. degree in electrical engineering from the University of Alabama, Tuscaloosa, AL, USA, in 2015.

He is currently a Visiting Assistant Professor with the Department of Electrical Engineering and Computer Science, Texas A&M University, Kingsville, TX, USA. His current research interests include neural network control, deep learning, cloud computing, power systems, power electronics, renewable energy, and electric drives.

Michael Fairbank received the B.Sc. degree in mathematical physics from Nottingham University, Nottingham, U.K., in 1994, and the M.Sc. degree in knowledge based systems from Edinburgh University, Edinburgh, U.K., in 1995, and the Ph.D. degree from the City University of London, London, U.K., in 2014.

He researches adaptive dynamic programming and neural networks with the University of Essex, Colchester, U.K., where he leads the Games Intelligence Research Group.

Donald C. Wunsch (F'07) received the B.S. degrees in applied mathematics from the University of New Mexico, Albuquerque, NM, USA, and Jesuit Core Honors Program, Seattle University, Seattle, WA, USA, the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, WA, USA, and the Executive MBA degree from Washington University in St. Louis, St. Louis, MO, USA.

He is the Mary K. Finley Missouri Distinguished Professor with the Missouri University of Science and Technology (Missouri S&T), Rolla, MO, USA, where he is the Director of the Applied Computational Intelligence Laboratory, a multidisciplinary research group. He was with Texas Tech University, Lubbock, TX, USA; Boeing, Seattle, WA, USA; Rockwell International, Albuquerque, NM, USA; and International Laser Systems, Albuquerque. He has produced 21 Ph.D. recipients in computer engineering, electrical engineering, systems engineering, and computer science. His current research interests include clustering/unsupervised learning, biclustering, adaptive resonance and adaptive dynamic programming architectures, hardware and applications, neurofuzzy regression, autonomous agents, games, and bioinformatics.

Dr. Wunsch was a recipient of the NSF CAREER Award, the 2015 INNS Gabor Award, and the 2019 Ada Lovelace Service Award. He is the previous International Neural Networks Society President, and an INNS Fellow. He served as the IJCNN General Chair, and on several boards, including the St. Patrick's School Board, the IEEE Neural Networks Council, INNS, and the University of Missouri Bioinformatics Consortium, the Chair of the Missouri S&T Information Technology and Computing Committee as well as the Student Design and Experiential Learning Center Board.

Eduardo Alonso received the M.Sc. degree in artificial intelligence from the AI Marvin Minsky Laboratory, City, University of London, London, U.K.

He is the Head of the Department of Computer Science, City, University of London, where he also acts as the Director of the Data Science Institute and co-directs the Centre for Computational and Animal Learning Research with Dr. E. Mondragon. He is an expert on artificial intelligence. He has published over 100 papers in artificial intelligence journals and contributed to "The Cambridge Handbook of Artificial Intelligence." He has edited special issues for the journals *Autonomous Agents and Multi-Agent Systems* and *Learning and Behavior*, and the book entitled *Computational Neuroscience for Advancing Artificial Intelligence: Models, Methods and Applications*. His current research interests include machine learning methods for systems and control, and models for computational neuroscience.

Mr. Alonso has acted as an OC and a PC of the International Joint Conference on Artificial Intelligence and the International Conference on Autonomous Agents and Multiagent Systems. He has served as the Vice-Chair of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour. He is a member of the U.K. Engineering and Physical Sciences Research Council Peer Review College.