American Journal of Intelligent Systems 2017, 7(1): 1-18
DOI: 10.5923/j.ajis.20170701.01

Neural Network-Based Adaptive Speed Controller Design for Electromechanical Systems (Part 2: Dynamic Modeling Using MLMA & Closed-Loop Simulations)

Vincent A. Akpan[1,*], Michael T. Babalola[2], Reginald A. O. Osakwe[3]

[1]Department of Physics Electronics, The Federal University of Technology, Akure, Nigeria
[2]Department of Physics Electronics, Afe Babalola University, Ado-Ekiti, Nigeria
[3]Department of Physics, The Federal University of Petroleum Resources, Effurun, Nigeria

Abstract The dynamic modeling of an electromechanical motor system (EMS) for different input voltages based on different weights and the corresponding output revolutions per minute using neural networks (NN) is presented in this paper with a view to quantify the effects of voltages based on different weights on the system output. The input–output data i.e. the electrical input voltage and the revolution per minute (rpm) of a PORCH PPWPM432 permanent magnet direct current (PMDC) motor as the output which is obtained from the EMS have been used for the development of a dynamic model of the EMS. This paper presents the formulation and application of an online modified Levenberg-Marquardt algorithm (MLMA) for the nonlinear model identification of the EMS. The performance of the proposed MLMA algorithm is compared with the so-called error back-propagation with momentum (EBPM) algorithm which is the modified version of the standard back-propagation algorithm for training NNs. The MLMA and the EBPM algorithms are validated by one-step and five-step ahead prediction methods. The performances of the two algorithms are assessed by using the Akaike's method to estimate the final prediction error (AFPE) of the regularized criterion. The validation results show the superior performance of the proposed MLMA algorithm in terms of much smaller prediction errors when compared to the EBPM algorithm. Furthermore, the simulation results shows that the proposed techniques and algorithms can be adapted and deployed for modeling the dynamics of the EMS and the prediction of future behaviour of the EMS in real life scenarios. In addition, the dynamic modeling of the EMS in closed-loop with a discrete-time fixed parameter proportional-integral-derivative (PID) controller has been conducted using both networks trained with EBPM and the MLMA algorithms. The simulation results demonstrate the efficiency and reliability of the proposed dynamic modeling using MLMA and closed-loop PID control scheme. However, despite the little poor performance of the PID controller, the accuracy of the NN model trained with the MLMA when used in a dynamic operating environment has been confirmed.

Keywords Artificial neural network (ANN), Dynamic modeling, Electromechanical motor systems (EMS), Error back-propagation with momentum (EBPM), Modified Levenberg-Marquardt algorithm (MLMA), Neural network nonlinear autoregressive moving average with exogenous inputs (NNARMAX), Nonlinear model identification, Proportional-integral-derivative (PID) control

1. Introduction

Adaptive control has been extensively investigated and developed in both theory and application during the past few decades and it is still a very active research field [1-15]. In the earlier stage, most studies in adaptive control concentrated on linear systems [15]. A remarkable development in adaptive control theory is the resolution of the so- called ideal problem, which is the proof that several adaptive control systems are globally stable under certain ideal conditions. Then the robustness issues of adaptive control with respect to non ideal conditions such as external disturbances and unmodeled dynamics were addressed which resulted in many different robust adaptive control algorithms [1, 2].

Adaptive control algorithms can be applied for the control of any nonlinear dynamic system. But a model of the system is required [3, 15] A recent approach to modeling nonlinear dynamical systems is the use of artificial neural networks (ANN) or simply neural networks (NN). The application of neural networks for model identification and adaptive control of dynamic systems have been studied extensively [3-14]. As demonstrated in [3, 7, 8, 10, 11, 15], neural

* Corresponding author:
vaakpan@futa.edu.ng (Vincent A. Akpan)
Published online at http://journal.sapub.org/ajis

networks can approximate any nonlinear function to an arbitrary high degree of accuracy. The adjustment of the NN parameters results in different shaped nonlinearities achieved through a gradient descent approach on an error function that measures the difference between the output of the NN and the output of the true system for given input data, output data or input-output data pairs (training data).

The Adaptive speed control of electromechanical systems has been widely studied and performed using various methods and components in the design and experiments [16-18]. However, the speed control could be achieved with either adaptive or conventional non-adaptive control methods. In adaptive control, there exists a feedback control with the ability of adjusting its speed in a changing environment so as to satisfy or maintain a set or desired speed. The actual speed is kept by speed controller to follow reference speed command. Adaptive control algorithms can be classified as either direct or indirect, depending on whether they employ an explicit parameter estimation algorithm within the overall adaptive scheme. By updating the required modeling information, perhaps through closed-loop identification, a direct adaptive control algorithm can be converted to an indirect adaptive control algorithm, which may yield greater versatility in practice. While model reference adaptive controllers and self tuning regulators were introduced as different approaches, the only real difference between them is that model reference schemes are direct adaptive control schemes whereas self tuning regulators are indirect. The self tuning regulator first identifies the system parameters recursively, and then uses these estimates to update the controller parameters through some fixed transformation. The model reference adaptive schemes update the controller parameters directly (no explicit estimate or identification of the system parameters are made).

Many intelligent control techniques [19], such as artificial neural network and adaptive fuzzy logic control (AFLC) methods, have been developed and applied to control the speed of permanent magnet direct current (d.c.) motor, in order to obtain high operating performance [20]. Moreover, the development of AFLCs can be used to cope with some important complex control problems such as stabilization and tracking system output signals, the presence of nonlinearity and disturbance. Adaptive control schemes are generally used to control systems which include unknown and time-varying parameters [21].

The ultimate aim of this research is to develop an adaptive speed controller that will maintain a desired reference trajectory of 60 rpm despite disturbances and its effects on the electromechanical system. At the center of the electromechanical system is a permanent magnet d.c. (PMDC) motor. Generally, d.c. motors are one of the most widely prime movers in industries today. They play important roles in energy conversion processes where they convert electrical energy into mechanical energy. In mechanical systems, speed varies with the number of tasks. Thus, speed control is necessary to do mechanical work in a proper way. It makes the motor to operate easily [22]. The speed of d.c. motor is directly proportional to the supply voltage. The d.c. motor is the obvious proving ground for advanced control algorithms in electric drives due to the stable and straight forward characteristics associated with it. It is also ideally suited for trajectory control applications. From a control system point of view, the d.c. motor can be considered as single input single output (SISO) plant, thereby eliminating the complications associated with a multiple-input multiple-output (MIMO) systems [23, 24]. To control any system, the basic understanding of the system is required. We need to understand the input and output behaviour of such system which will allow for the control of the system. All the inputs of real systems are always actuated by control signals from the controller while the system outputs on the other hand are measured using a sensors. The speed control and optimization of electromechanical systems has become imperative due to its applications in real life scenarios, any improvement made in this regard will be a novel contribution.

Some of the more commonly occurring electromechanical systems are presented using linear transfer functions within each and every block defining the systems. In real designs, nonlinear elements frequently occur. However, such nonlinear components cannot be approximated by linear differential equations with constant coefficients (e.g. the Laplace solution techniques) [25]. Therefore, this work focuses on the formulation of neural network-based modeling algorithm that will capture the nonlinear dynamics of the EMS where such model can be used for the development of adaptive control algorithm for the adaptive speed control of EMSs.

The paper is organised as follows. The description of the EMS is presented in Section 2. An overview of the EMS design and construction as well as information on the technique of data acquisition from the EMS are also presented in this section. Section 3 presents the formulation of the neural network-based modified Levenberg-Marquardt algorithm (MLMA) for NNARMAX model identification. Three validation algorithms are also presented in this section. The dynamic modeling and closed-loop simulations with a PID controller together with the simulation results are given in Section 4. A brief conclusion and possible directions on future work is given in Section 5.

2. Description of the EMS

This section presents an overview of the design and development of the EMS. The complete design and fabrication procedure for the system can be found in [26]. The measurement procedure with the description of the EMS input-output data that describes the system behaviour, the considerations of the electromechanical system and the effects of the process variables on the system are also detailed in [26].

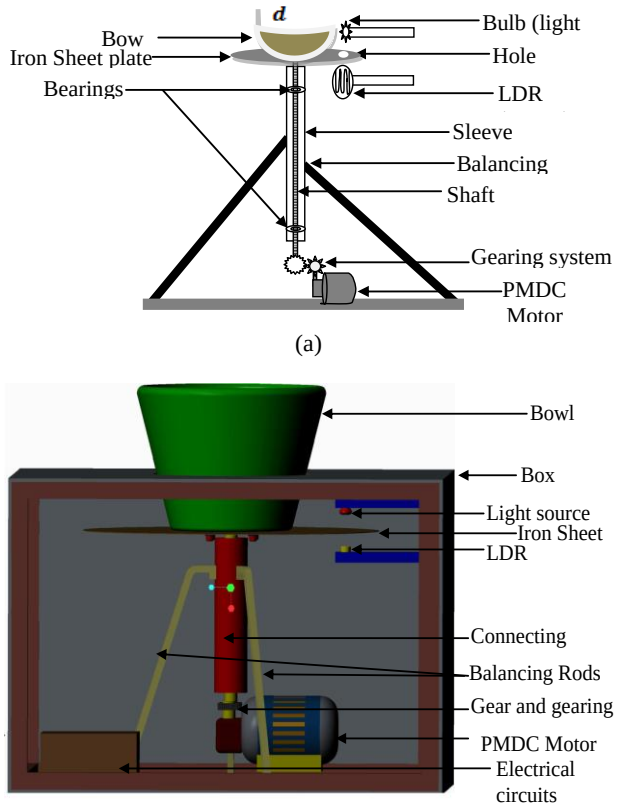


Figure 1. The designed electromechanical motor system: (a) Schematic drawing of the electromechanical system and (b) 3-D drawing of the electromechanical speed control system

2.1. Design of the EMS

The actuator used in this project is a PORCHE PPWPM432 windshield wiper motor which has a worm gear and simple ring gear that gives the device its incredible torque. This type of motor is called a “gearhead” or "gear motor" and has the advantage of having lots of torque. The system has been designed to accommodate different standard weights which are to be loaded through the bowl fitted to the iron sheet plate with bolts and nuts to hold it firmly as it rotates. The sheet (46 cm diameter, 0.1 cm thickness) is welded to the motor system with a shaft connected to the motor through the gearing system which transfers the motion of the motor to the bowl. Bearings were fitted to allow for free movements or rotation of the system and balancing rods were clamped to hold the system from falling and to maintain balance while rotating. The light dependent resistor (LDR) sensor is installed and aligned with the light source to receive the incident light through the hole of 2.5 cm bored on the sheet. The following is a list and specification of materials used for the design of the proposed electromechanical motor system, namely: 1). a PORCHE PPWPM432 windshield wiper motor with 19.6 cm diameter; 2). a container bowl with 21 cm height, 18.8 cm bottom diameter and 30 cm top diameter; 3). an iron sheet plate of 46 cm diameter, 0.1 cm thickness with a 2.5 cm hole for the sensor; 4). two gears 45 teeth and 35 teeth with 4.5 cm diameter and 3.5 cm diameter respectively; and 5). an Iron shaft of 19.4 cm length and 5.1 cm diameter (please see [26] for more detail).

The sensor and bulb are installed at the bottom and top respectively with 4.5 cm equidistant from the iron sheet plate. A well labelled schematic diagram of the proposed electromechanical motor system is shown in Fig. 1(a) and a well labelled 3-D diagram of the proposed electromechanical motor system is also shown in Fig. 1(b) for a 3D view of the system.

Table 1. The dynamics parameters of the EMS based on experimental measurements

S/N	Parameters	Symbols	Units	Minimum Value	Maximum Value
1.	Input voltage to the digital potentiometer	V_i	V	0.25	0.38
2.	Output voltage of the potentiometer	V_o	V	0.38	1.75
3.	Voltage input to the motor	V_k	V	3.86	9.52
4.	Applied weights	W_x	kg	0.5	35
5.	Speed of the motor	S_i	rpm	15	64

Table 2. Manipulated variables (MV) and the controlled variables (CV) with the EMS constraints

Motor System Input/Output Parameters	Measurable parameters of the motor system	Nominal Values	Input Constraints	
			Minimun Value	Maximum Value
Manipuated Variables (Inputs)	Input voltage (V)	0.25	0	5
	Running voltage (V)	3.86	3.86	8.67
Controlled Variables (Outputs)	Speed (rpm) with the prescribed input voltage	60	15	64
	Speed (rpm) with the prescribed running voltage	60	15	64

2.2. Experimental Input-Output Data Collection from the Designed EMS

The dynamic parameters of the electromechanical motor system are listed in Table 1 with their respective minimum and maximum values. The data for the following dynamic parameters have been collected for this work:

1). The input voltage to the digital potentiometer; V_i;
2). The output voltage of the potentiometer in response to changes in the input; V_j;
3). The actual voltage input to the motor; V_k;
4). The input standards weights, W_x; and
5). The corresponding measured speed of the motor (in revolution per minute) in response to the changes in the inputs (voltage and applied weights). S_i.

2.3. Description of the EMS Input-Output Data

The picture of the completely designed and constructed EMS is shown in Fig. 2 [26]. Weights were combined to give a range from 0.5 kg to 35 kg limited by the diameter of the bowl on the iron plate. The input voltages and the corresponding speed of the EMS as measured by the counter circuit are summarized in Table 2. The input voltages were first fixed for no load condition and the speed in rpm was recorded. Then the experiment was repeated for each of the applied weights keeping the voltages fixed and the respective speed (in rpm) as displayed by the digital counter. The minimum recorded rpm with no load is 18 rpm while with 35 kg weight the digital counter recorded 15 rpm with the minimum input voltage of 3.86 V. The maximum speed recorded is 64 rpm which occur at the highest supplied voltage of 9.5 V under no load condition but 51 rpm with 35 kg weights applied [26].

Figure 2. The picture of the completely designed and constructed electromechanical motor system

The speed of the system in revolution per minute which is the output of the system is affected by two major parameters; the input voltage and the applied weights. The higher the input voltage, the faster the speed of the motor moved and the higher the weight applied the slower the speed of the motor.

3. Formulation of the NN-Based MLMA and the Model Validation Algorithms for NNARMAX Model Identification

3.1. Formulation of the Neural Network Model Identification Problem

The method of representing dynamical systems by vector difference or differential equations is well established in systems and control theories [3, 15, 27, 28]. Assuming that a p-input q-output discrete-time nonlinear multivariable system at time k with disturbance $\tilde{d}(k)$ can be represented by the following Nonlinear AutoRegressive Moving Average with eXogenous inputs (NARMAX) model:

$$\left.\begin{aligned} Y(k) = J\big[&Y(k-1),\ldots,Y(k-n_a),\\ &U(k-d),\ldots,U(k-d-n_b),\\ &\varepsilon(k-1),\ldots,\varepsilon(k-n_c)\big]+\tilde{d}(k) \end{aligned}\right\} \quad (1)$$

where $J(\bullet,\bullet)$ is a nonlinear function of its arguments, and $[Y(k-1),\ldots,Y(k-n_a)]$ are the past output vector, $[U(k-d),\ldots,U(k-d-n_b)]$ are the past input vector, $\varepsilon(k-1),\ldots,\varepsilon(k-n_c)$ are the past noise vector, $Y(k)$ is the current output, n_a, n_b and n_c are the number of past values of the system outputs, system inputs and noise inputs respectively that defines the order of the system, and d is the time delay. The predictor form of (1) based on the information up to time $k-1$ can be expressed in the following compact form as [15]:

$$\hat{Y}(k\,|\,k-1,\theta(k)) = J\Big[\varphi(k,\theta(k)),\theta^T(k)\Big] \quad (2)$$

where $\varphi(k,\theta(k)) = \big[Y(k-1),\ldots,Y(k-n_a),\ U(k-d),\ldots,$ $U(k-d-n_b),\ \varepsilon(k-1,\theta(k)),\ldots,\varepsilon(k-n_c,\theta(k))\big]^T$ is the regression (state) vector, $\theta(k)$ is an unknown parameter vector which must be selected such that $\hat{Y}(k\,|\,\theta(k)) \approx Y(k)$, $\varepsilon(k,\theta(k))$ is the error between (1) and (2) defined as

$$\varepsilon(k,\theta(k)) = Y(k) - \hat{Y}(k\,|\,\theta(k)) \quad (3)$$

where $k-1$ in $\hat{Y}(k\,|\,k-1,\theta(k))$ of (2) is henceforth omitted for notational convenience. Not that $\varepsilon(k,\theta(k))$ is the same order and dimension as $\hat{Y}(k\,|\,\theta(k))$.

Now, let Θ be a set of parameter vectors which contain a set of vectors such that:

$$\Theta:\ \theta(k) \in \mathbb{Q}_\theta \subset \Re^v \rightarrow \hat{\theta}(k) \quad (4)$$

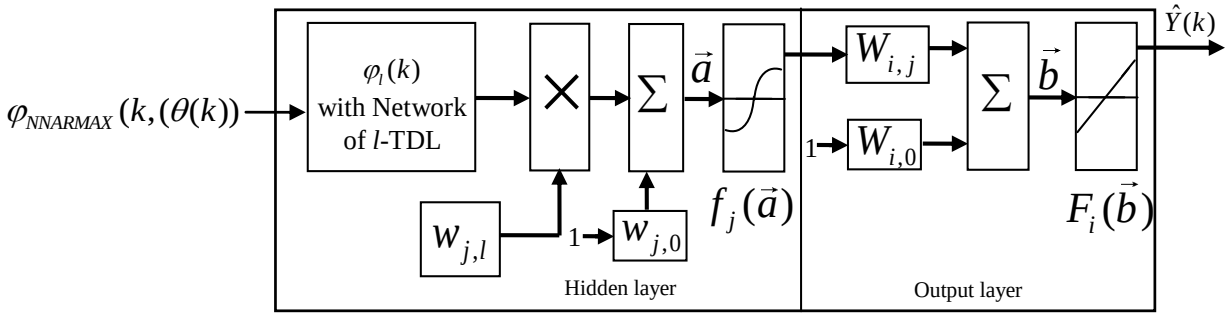


Figure 3. Architecture of the dynamic feedforward NN (DFNN) model

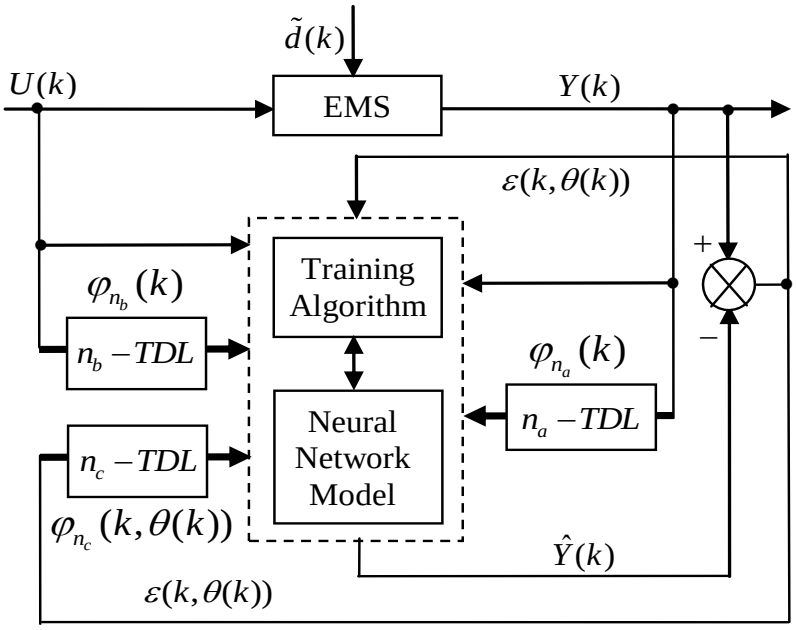


Figure 4. NNARMAX model identification based on the teacher-forcing method

where $\mathbb{Q}_\theta$ is some subset of $\Re^\nu$ where the search for $\hat{\theta}(k)$ is carried out; ν is the dimension of $\theta(k)$; $\hat{\theta}(k)$ is the desired vector which minimizes the error in (3) and is contained in the set of vectors $\Theta = \{\theta_1(k),\ldots,\theta_\tau(k)\}$; $\theta_1(k),\ldots,\theta_\tau(k)$ are distinct values of $\theta(k)$; and $\tau = 1,2,\ldots,\max_{iter}$ is the number of iterations required to determine the $\hat{\theta}(k)$ from the vectors in Θ.

Let a set of N input-output data pair obtained from prior system operation over *NT* period of time be defined:

$$Z^N = \{U(1),Y(1),\ldots,U(N),Y(N)\},\quad N = 1,2,\ldots \tag{5}$$

where T is the sampling time of the system outputs. Then, the minimization of (3) can be stated as follows:

$$\hat{\theta}(k) = \arg\min_{\theta(k)} J(Z^N,\varphi(k,\theta(k)),\theta(k)) \tag{6}$$

where $J(Z^N,\varphi(k,\theta(k)),\theta(k))$ is formulated as a total square error (TSE) type cost function which can be stated as:

$$J(Z^N,\varphi(k,\theta(k)),\theta(k)) = \frac{1}{2N}\sum_{l=1}^{N}[\varepsilon(l,\theta(k))]^2 \tag{7}$$

The inclusion of $\theta(k)$ as an argument in $\varphi(k,\theta(k))$ is to account for the desired model $\hat{\theta}(k)$ dependency on $\tilde{d}(k)$. Thus, given as initial random value of $\theta(k)$, n_a, n_b and (5), the system identification problem reduces to the minimization of (6) to obtain $\hat{\theta}(k)$. For notational convenience, $J(\theta(k))$ shall henceforth be used instead of $J(Z^N,\varphi(k,\theta(k)),\theta(k))$.

3.2. Neural Network Identification Scheme

The minimization of (6) is approached here by considering $\hat{\theta}(k)$ as the desired model of network and having the DFNN architecture shown in Fig. 3. The proposed NN model identification scheme based on the teacher-forcing method is illustrated in Fig. 4. Note that the "Neural Network Model" shown in Fig. 4 is actually the DFNN shown in Fig. 3 via

tapped delay lines (TDL). The inputs to NN of Fig. 4 are $\varphi_{n_b}(k)=[U(k-d),\ldots,U(k-d-n_b)]$, $\varphi_{n_a}(k)=[Y(k-1),\ldots,Y(k-n_a)]^T$ and $\varphi_{n_c}(k,\theta(k))=[\varepsilon(k-1,\theta(k)),\ldots,\varepsilon(k-n_c,\theta(k))]^T$ which are concatenated into $\varphi_{NNARMAX}(k,\theta(k))$ or simply $\varphi(k,\theta(k))$ as shown in Fig. 3. The output of the NN model of Fig. 4 in terms of the network parameters of Fig. 3 is given as:

$$\left.\begin{aligned}\hat{Y}(k\,|\,\hat{\theta}(k)) &= F_i\left(\sum_{j=1}^{n_h} W_{i,j}\, f_j(\vec{a}) + W_{i,0}\right)\\ \vec{a} &= \sum_{l=1}^{n_\varphi} w_{j,l}\,\varphi(k,\theta(k)) + w_{j,0}\end{aligned}\right\} \tag{8}$$

where n_h and n_φ are the number of hidden neurons and number of regressors respectively; i is the number of outputs, $w_{j,l}$ and $W_{i,j}$ are the hidden and output weights respectively; $w_{j,0}$ and $W_{i,0}$ are the hidden and output biases; $F_i(\vec{b})$ is a linear activation function for the output layer and $f_j(\vec{a})$ is an hyperbolic tangent activation function for the hidden layer defined here as:

$$f_j(\vec{a}) = 1 - \frac{2}{e^{2\cdot\vec{a}}+1} \tag{9}$$

Bias is a weight acting on the input and clamped to 1. Here, $\hat{\theta}(k)$ is a collection of all network weights and biases in (8) in terms of the matrices $\mathbf{w}=\{w_{j,l}\ \ w_{j,0}\}$ and $\mathbf{W}=\{W_{i,j}\ \ W_{i,0}\}$. Equation (8) is here referred to as NN NARMAX (NNARMAX) model predictor for simplicity.

Note that $\tilde{d}(k)$ in (1) is unknown but is estimated here as a covariance noise matrix, $\Gamma[\theta(k)]=\mathbf{E}[\tilde{d}^T(k)\tilde{d}(k)]$. Using $\Gamma[\theta(k)]$, Equation (7) can be rewritten as [3], [15], [27]:

$$J(\theta(k)) = \frac{1}{2N}\left(\sum_{l=1}^{N}\varepsilon[l,\theta(k)]^T\,\Gamma^{-1}[\theta(k)]\varepsilon[l,\theta(k)] + \theta^T(k)D\theta(k)\right) \tag{10}$$

where the second term in (10) is the regularization (weight decay) term which has been introduced to reduce modeling errors, improve the robustness and performance of the proposed training algorithms [3, 15, 27]. The term $D=\alpha_d I=[\alpha_h\ \ \alpha_o]I$ is a penalty norm and also removes ill-conditioning, where I is an identity matrix, α_h and α_o are the weight decay parameters for the input-to-hidden and hidden-to-output layers respectively. Note that both $\hat{\Gamma}^{(j)}[\theta(k)]$ and D are adjusted simultaneously during network training with $\theta(k)$ and are used to update $\hat{\theta}(k)$ iteratively. The algorithm for estimating the covariance noise matrix and updating $\hat{\theta}(k)$ is summarized in Table 3. Note that this algorithm is implemented at each sampling instant until $\hat{\Gamma}^{(j)}[\theta(k)]$ has reduced significantly as in step *7)*.

Table 3. Iterative algorithm for estimating the covariance noise matrix

1) Given initial network weights $\theta(k)=\theta^{(0)}(k)$ and $j=j_{\max}$.

2) For $k=1$ to Number of Samples (*N*), Do,

3) Initialize $\Gamma^{(0)}[\theta(k)]=I$, Do,

4) Set $j=1$

5) Train the network for τ iterations with a training algorithm using $\Gamma^{(j-1)}[\theta(k)]$ to obtain $\hat{\theta}(k)\leftarrow\theta_\tau(k)\triangleq\theta^{(j)}(k)$.

6) Estimate the covariance matrix for the noise using

$$\hat{\Gamma}^{(j)}[\theta(k)] = \frac{1}{2N}\sum_{l=1}^{N}\varepsilon^T[\theta^{(j)}(k)]\varepsilon[\theta^{(j)}(k)]$$

7) If $\hat{\Gamma}^{(j)}[\theta(k)]<esp$, where esp is a convergence criteria.

Set $j\leftarrow j+1$ and Go To Step 4).

Else, set $\hat{\theta}(k)=\theta^{(j)}(k)$ and End *Set j*.

8) End *For k.*

3.3. Formulation of the NN-Based MLMA

Unlike the standard back-propagation (BP) algorithm which is a steepest descent algorithm, the MLMA algorithm proposed here is based on the Gauss-Newton method with the typical updating rule given from [3, 15, 27] as:

$$\hat{\theta}(k) = \theta_\tau(k) + \Delta\theta_\tau(k) \tag{11}$$

$$\text{where}\quad \Delta\theta_\tau(k) = -R[\theta_\tau(k)]^{-1}G[\theta_\tau(k)] \tag{12}$$

$\theta_\tau(k)$ denotes the value of $\theta(k)$ at the current iterate τ, $\Delta\theta_\tau(k)$ is the search direction, $G[\theta_\tau(k)]$ and $R[\theta_\tau(k)]$ are the Jacobian (or gradient matrix) and the Gauss-Newton Hessian matrices evaluated at $\theta(k)=\theta_\tau(k)$.

As mentioned earlier, due to the model $\theta(k)$ dependency on the regression vector $\varphi(k,\theta(k))$, the NNARMAX model predictor depends on a *posteriori* error estimate using the feedback as shown in Fig. 4. Suppose that the derivative of the network outputs with respect to $\theta(k)$ evaluated at $\theta(k)=\theta_\tau(k)$ is given as [15]:

$$\psi[k,\theta_\tau(k)] = \frac{d\hat{Y}(k\,|\,\theta(k))}{d\theta(k)} \tag{13}$$

The derivative of (13) is carried out in a BP fashion for the input-to-hidden layer and for the hidden-to-output layer respectively for the two-layer DFNN of Fig. 3. Thus, the derivative of the NNARMAX model predictor can be

expressed as [15]:

$$\left.\begin{aligned}\psi[k,\theta_\tau(k)] = &\frac{\partial \hat{Y}(k\,|\,\theta(k))}{\partial\theta(k)} \\ &-\frac{\partial \hat{Y}(k\,|\,\theta(k))}{\partial\varepsilon(k-1,\theta(k))}\frac{\partial \hat{Y}(k-1\,|\,\theta(k))}{\partial\theta(k)} \\ &-\cdots-\frac{\partial \hat{Y}(k\,|\,\theta(k))}{\partial\varepsilon(k-n_c,\theta(k))}\frac{\partial \hat{Y}(k-n_c\,|\,\theta(k))}{\partial\theta(k)}\end{aligned}\right\} \quad (14)$$

Thus, Equation (14) can be expressed equivalently as

$$\left.\begin{aligned}\psi[l,\theta(k)] = &\frac{d\hat{Y}(k\,|\,\theta(k))}{d\theta(k)} - C_1(k)\psi[k-1,\theta(k)] - \\ &\cdots - C_n(k)\psi[k-n,\theta(k)]\end{aligned}\right\} \quad (15)$$

By letting $C(k,z^{-1}) = I + C_1(k)z^{-1} + \cdots + C_n(k)z^{-n_c}$, then (15) can be reduced to the following form [15]

$$\psi[k,\theta(k)] = \frac{1}{C(k,z^{-1})}\frac{d\hat{Y}(k\,|\,\theta(k))}{d\theta(k)} \quad (16)$$

As it can be seen from (16), the gradient is calculated by filtering the partial derivatives with the time-varying filter $1/C(k,z^{-1})$ which depends on the prediction errors based on the predicted outputs. Equation (16) is the only component that actually impedes the implementation of the NN training algorithms depending on its computation.

Table 4. An algorithm for placing the roots of the time-varying filter of the NNARMAX model predictor within the unit circle for stability

1) Given network weights $\theta(k) = \theta^{(0)}(k)$, time-varying filter $C(k,z^{-1}) = C^{(0)}(k,z^{-1})$ and regression vector $\varphi(k,\theta(k))$

2) Compute the roots of $C(k,z^{-1})$ as $C_{Roots}(k,z^{-1})$ and length of $C_{Roots}(k,z^{-1})$ as l_{CRoots}.

3) Compute the absolute value of

$$C_{Roots}(k,z^{-1}) = abs(C_{Roots}(k,z^{-1}))$$

4) For $i=1$ to l_{CRoots},

if $abs(C^{(i)}_{Roots}(k,z^{-1})) > 1$

$$C^{(i)}_{Roots}(k,z^{-1}) = \frac{1}{C^{(i)}_{Roots}(k,z^{-1})}$$

End if, End for

5) Compute the $C(k,z^{-1})$ using the real root from Step *4)*.

Due to the feedback signals, the NNARMAX model predictor may be unstable if the system to be identified is not stable since the roots of (16) may, in general, not lie within the unit circle. The approach proposed here to iteratively ensure that the predictor becomes stable is summarized in the algorithm of Table 4. Thus, this algorithm ensures that roots of $C(k,z^{-1})$ lies within the unit circle before the weights are updated by the training algorithm proposed in the next sub-section.

3.3.1. The Proposed Modified Levenberg-Marquardt Algorithm (MLMA)

The Levenberg-Marquardt [28-30] modification to (12) is the inclusion of a non-negative parameter λ_τ to the diagonal of $R[\theta_\tau(k)]$ with a new iterative updating rule as follows:

$$\hat{\theta}(k) = \theta_\tau(k) + \Delta\theta_\tau(k) \quad (17)$$

$$\Delta\theta_\tau(k) = -\left[R[\theta_\tau(k)] + \lambda_\tau I\right]^{-1} G[\theta_\tau(k)] \quad (18)$$

where I is a diagonal matrix, $G[\theta_\tau(k)]$ and $R[\theta_\tau(k)]$ are:

$$G[\theta_\tau(k)] = -\frac{1}{N}\sum_{l=1}^{N}\left(\begin{matrix}\psi[l,\theta_\tau(k)]\cdot\Gamma^{-1}[\theta(k)]\cdot\varepsilon[l,\theta_\tau(k)] \\ + D\theta_\tau(k)\end{matrix}\right) \quad (19)$$

$$R[\theta_\tau(k)] = \frac{1}{N}\sum_{l=1}^{N}\psi^T[l,\theta_\tau(k)]\cdot\Gamma^{-1}[\theta(k)]\cdot\psi[l,\theta_\tau(k)] + D \quad (20)$$

and $\psi[l,\theta_\tau(k)]$ the derivative of the network outputs with respect to $\theta(k)$ evaluated at $\theta(k) = \theta_\tau(k)$ and is computed according to (16).

The parameter λ_τ characterizes a hybrid of searching directions and has several effects [7, 29-32]: *1)* for large values of λ_τ (18) becomes steepest descent algorithm (with step $1/\lambda_\tau$) which requires a descend search method; and *2)* for small values of λ_τ (18) reduces to the Gauss-Newton method and $\left[R[\theta_\tau(k)] + \lambda_\tau I\right]^{-1}$ may become non-positive definite matrix.

Despite the fact that (10) is a weighted criterion, the convergence of the Levenberg-Marquardt algorithm (LMA) may be slow since $\theta_\tau(k)$ contains many parameters of different magnitudes, especially if these magnitudes are large as in most cases [8, 10, 28]. This is the major reason for not using the LMA in online training of the NNs.

This problem can be alleviated by adding a scaling matrix $S_\tau = sI$ (where s is the scaling parameter and I is an identity matrix) which is adjusted simultaneously with $\theta_\tau(k)$ and instead of checking $\left[R[\theta_\tau(k)] + \lambda_\tau I\right]^{-1}$ in (18) for positive definiteness, the check is expressed as

$$V[\theta_\tau(k)] = \left[R[\theta_\tau(k)] + (S_\tau)^T\lambda_\tau(S_\tau)\right]^{-1} \quad (21)$$

This will ensure that (21) is always positive definite with fast convergence if a suitable value for λ_τ is chosen.

Different from other methods [28, 32-34] the method proposed here uses the Cholesky factorization algorithm and then iteratively selects λ_τ to guarantee positive

definiteness of (21) for online application. First, (21) is computed and the check is performed. If (21) is positive definite, the algorithm is terminated, otherwise λ is increased iteratively until this is achieved. The method is summarized in Table 5. The key parameter in the algorithm is η. Next, the Cholesky factor $L_{b,a}(\theta(k))$ given by (T.2) in Table 5 is reused to compute the search direction from (18) in two-stage forward and backward substitution given respectively as:

Table 5. Iterative Algorithm for Selecting λ_τ

Initialize $km \in [0.5, \eta] = [0.5, 1, 2, 4, 6, 8]$ of length kl.

Let $[sm,\ sn] = size(V_\tau[\theta(k)])$. Set $L_{a,a}(k) = -1$.

Evaluate (29).

for $i = 1$ to sm

while $iter < kl$

for $kn = 1$ to kl

for $a = 1$ to sn

$$L_{a,a}(\theta(k)) = \sqrt{V_{a,a}[\theta(k)] - \sum_{j=1}^{a-1} L_{a,j}^2(\theta(k))} \quad \text{(T.1)}$$

end for a.

if $L_{a,a}(k) < 0$, $\lambda = \lambda * km(1, kn)$ and recomputed (29)

Set $a = a + 1$, recomputed (T.1)

else, for $b = a + 1$ to sn

$$L_{b,a}(\theta(k)) = \frac{\sqrt{V_{b,a}[\theta(k)] - \sum_{j=1}^{a-1} L_{a,j}(\theta(k)) L_{a,j}(\theta(k))}}{L_{a,a}(\theta(k))} \quad \text{(T.2)}$$

end for b, end if $L_{a,a}(k)$, end for *kn*.

$iter = iter + 1$

if $iter > kl$ and $L_{a,a}(k) < 0$, break, end.

Set $\lambda_\tau \leftarrow \lambda$ and recomputed (29) using λ_τ.

end while $iter$,

end for sn.

$$L_{b,a}[(\theta_\tau(k)]\Delta\theta_\tau(k) = G[\theta_\tau(k)] \quad (22)$$

$$\Delta\theta_\tau(k) = \left(L_{b,a}[(\theta_\tau(k)]^T\right)^{-1} G[\theta_\tau(k)] \quad (23)$$

The convergence of the LMA using (17) to (23) may again be slowed if the initial guess $\theta^{(0)}(k)$ is too far from the optimum value $\hat{\theta}(k)$. Thus, the LMA is sometimes combined with the trust region method [35] so that the search for $\hat{\theta}(k)$ is constrained around a trusted region δ_τ. The problem can be defined as [3, 15]:

$$\theta_\tau(k) = \arg\min_{\theta(k)} \tilde{J}_\tau(\theta(k)) \quad (24)$$

$$\text{Subject to } \left|S_\tau(\theta(k) - \theta_\tau(k))\right| \le \delta_\tau \quad (25)$$

where $\tilde{J}_\tau(\theta(k))$ is the second-order Gauss-Newton approximate of (10) which can be expressed as:

$$\tilde{J}_\tau(\theta(k)) = \frac{1}{2N}\sum_{l=1}^{N}\left(\varepsilon[l, \theta_\tau(k)] - [\theta(k) - \theta_\tau(k)]^T \psi[l, \theta_\tau(k)]\right)^2 \quad (26)$$

which is expected to be valid only in a neighborhood around the current iterate $\theta_\tau(k)$. Thus, with this combined method and using the result from (23), Equation (17) can be rewritten as

Table 6. The modified Levenberg-Marquardt algorithm (MLMA) incorporating trust-region algorithm for updating $\hat{\theta}(k)$ *

1) Specify τ, $\tau_{\max}$, D, $\lambda_{\max} \in [1,\ 10^3]$, $s \in [0.1,\ 10^{-2}]$, *m* and *n* for $\varphi(k, \theta(k))$, $\lambda_\tau \in [0.1,\ 10^{-3}], \delta_\tau \in [0.1,\ 10^{-4}]$

2) Initialize the weights $\theta(k) = \theta^0(k)$ and the time-varying filter $C(k, z^{-1}) = C^0(k, z^{-1})$ with appropriate dimensions.

3) While $\tau = 1$, Do.

4) Evaluate $J(\theta(k))$ using (10) for the *a priori* estimate.

5) Ensure that the roots of $C(k, z^{-1})$ in (16) are within unit circle using the algorithm of Table 4 using $\varphi(k, \theta(k))$.

6) Compute $G[\theta_\tau(k)]$ using (19) and $R[\Delta\theta_\tau(k)]$ using (28).

7) Evaluate $V_\tau[\theta(k)]$ in (21) using the algorithm of Table 5 and determine the searching direction $\Delta(\theta_\tau(k))$ using (23).

8) Evaluate $J(\theta(k))$ using (10) for the *posteriori* estimate.

9) Evaluate (26) and (24) subject to (25).

10) Evaluate the ratio α_τ in (28).

11) Update λ_τ according to the following conditions on α_τ:

If $\alpha_\tau > 0.75$, then $\lambda_\tau \leftarrow 0.5 * \lambda_\tau$ and Go To 12).

If $\alpha_\tau < 0.25$, then $\lambda_\tau \leftarrow 2 * \lambda_\tau$ and Go To 12).

12) If $\left|S_\tau(\theta(k) - \theta_\tau(k))\right| \le \delta_\tau$, $\lambda_\tau < \lambda_{\max}$ and $ared > 0$.

Accept $\Delta(\theta_\tau(k))$ in (23), Set $\theta_\tau(k) \leftarrow \theta_\tau(k) + \Delta(\theta_\tau(k))$ and Go To 13).

Else $\tau \leftarrow \tau + 1, \lambda_\tau \leftarrow \lambda_{\tau+1}, \theta_\tau(k) \leftarrow \theta_{\tau+1}(k)$ and Go To 3).

13) Accept $\hat{\theta}(k) \leftarrow \theta_\tau(k)$ in (27).

*This algorithm is implemented in step 5) in algorithm of Tabe1.

$$\hat{\theta}(k) = \theta_\tau(k) + \Delta\theta_\tau(k) \quad (27)$$

The choice of selecting and/or adjusting δ_τ and λ_τ has led to the coding of several algorithms [7, 28-34]. In stead of adjusting δ_τ directly, this paper develops on the indirect approach proposed in [35] but reuses λ_τ computed in Table 5 to update the weighted criterion (10). Here, λ_τ is adjusted according to the ratio α_τ between the actual reduction ($ared$) of (10) and theoretical predicted decrease ($pdec$) of (10) using (26). The ratio can be defined as:

$$\alpha_\tau = \frac{ared}{pdec} = \frac{J(\theta_\tau(k)) - J(\theta_\tau(k) + \Delta\theta_\tau(k))}{J(\theta_\tau(k)) - \tilde{J}_\tau(\theta(k) + \Delta\hat{\theta}_\tau(k))} \quad (28)$$

where $J(\theta_\tau(k)) = J(Z^N, \varphi(k), \theta(k))$ in (10) for convenience and $\tilde{J}_\tau(\theta_\tau(k) + \Delta\theta_\tau(k))$ is the Gauss-Newton estimate of (10) using (26).

The complete modified Levenberg-Marquardt algorithm (MLMA) for updating $\hat{\theta}(k)$ is summarized in Table 6. Note that after $\hat{\theta}(k)$ is obtained using the algorithm of Table 6, the algorithm of Table 3 is implemented until the conditions set out in Step *7)* of the algorithm are satisfied.

3.4. Proposed Validation Methods for the Trained NNARMAX Model

Network validations are performed to assess to what extend the trained network has approximated and capture the operation of the underlying dynamics of a system and as measure of how well the model being investigated will perform when deployed for the actual process [3, 15, 27].

The first test involves the comparison of the predicted outputs with the true training data and the evaluation of their corresponding errors using (3).

The second validation test is the Akaike's final prediction error (AFPE) estimate [3, 15, 17, 28, 34] based on the weight decay parameter D in (10). A smaller value of the AFPE estimate indicates that the identified model approximately captures all the dynamics of the underlying system and can be presented with new data from the real process. Evaluating the $\varepsilon(k, \hat{\theta}(k))$ portion of (3) using the trained network with $\theta(k) = \hat{\theta}(k)$ and taking the expectation $\mathbf{E}\{J(Z^N, \hat{\theta}(k))\}$ with respect to $\varphi(k, \hat{\theta}(k))$ and $\tilde{d}(k)$ leads to the following AFPE estimate [3], [15], [27]:

$$\hat{F}(Z^N\hat{\theta}(k)) \approx \frac{N + p_a}{N - p_b} J(Z^N\hat{\theta}(k)) + \gamma \quad (29)$$

where $p_a = \mathbf{tr}\left\{V(\hat{\theta}(k))\left[V(\hat{\theta}(k)) + D\right]^{-1} V(\hat{\theta}(k))\left[V(\hat{\theta}(k)) + D\right]^{-1}\right\}$ and $\mathbf{tr}\{\cdot\}$ is the trace of its arguments and it is computed as the sum of the diagonal elements of its arguments, $p_b = tr\{V(\hat{\theta}^*)[V(\hat{\theta}^*) + (1/N)D]^{-1}\}$ and $\boldsymbol{\gamma}$ is a positive quantity that improves the accuracy of the estimate and can be computed according to the following expression:

$$\gamma = \frac{\hat{\theta}(k)^T D}{N^2}\left(R[\hat{\theta}(k)] + \frac{D}{N}\right)^{-1} R[\hat{\theta}(k)]\left(R[\hat{\theta}(k)] + \frac{D}{N}\right)^{-1} D\hat{\theta}(k)$$

The third method is the *K*-step ahead predictions [10] where the outputs of the trained network are compared to the unscaled output training data. The K-step ahead predictor follows directly from (8) and for $\varphi(k, \hat{\theta}(k)) = \hat{\varphi}(k + K, \hat{\theta}(k))$ and $\theta(k) = \hat{\theta}(k)$, takes the following form:

$$\hat{Y}((k + K) \mid k, \hat{\theta}(k)) = \hat{J}(Z^N, \hat{\varphi}(k + K), \hat{\theta}(k)) \quad (30)$$

where

$$\hat{\varphi}(k + K, \hat{\theta}(k)) = [U((k + K - 1) \mid \hat{\theta}(k)), \ldots,$$
$$U((k + K - n_b) \mid \hat{\theta}(k)),$$
$$\hat{Y}((k + K - 1) \mid \hat{\theta}(k)), \ldots, \hat{Y}((k + K + 1 - \min(k, n_a)) \mid \hat{\theta}(k)),$$
$$Y((k + K - 1) \mid \hat{\theta}(\mathrm{k})), \ldots, Y((k + K - \max(n_a - k, 0) \mid \hat{\theta}(k))]^T$$

The mean value of the K-step ahead prediction error (MVPE) between the predicted output and the actual training data set is computed as follows:

$$MVPE = mean\left(\sum_{k=m+K}^{N} \frac{Y(k) - \hat{Y}((k + K) \mid k, \hat{\theta}(k))}{Y(k)}\right) \times 100\% \quad (31)$$

where $Y(k)$ corresponds to the unscaled output training data and $\hat{Y}((k + K) \mid k, \hat{\theta}(k))$ the *K*-step ahead predictor output.

4. Dynamic Modeling and Adaptive Closed-Loop Simulations of the EMS Using Discrete-Time PID Controller

4.1. Selection of the Manipulated Inputs and Controlled Outputs for the Dynamic EMS Modeling Problem

The manipulated variable (MV) is the variable chosen to affect control over an output variable. As the output is being controlled it is normally referred to as the controlled variable (CV). The objective of a control system is to keep the CV at their desired values (or setpoints). This is achieved by manipulating the MV using a control algorithm [36]. The manipulated variables with the nominal values and constraints as well as the controlled variable with the nominal values and the constraints are as shown in Table 2. The manipulated variable is the input voltage to the digital potentiometer and the running voltage of the motor with nominal values of 0.25 V and 3.86 V respectively (see Tables 1 and 2). The controlled variable is the desired output rpm of the motor with nominal value of 60 rpm.

Disturbances are variables that fluctuate and cause the

process output to move from the desired operating value (setpoint). A disturbance could be a change in flow or temperature of the surroundings or pressure etc. Disturbance variables can normally be further classified in terms of measured or unmeasured signals. The different random weights applied to the EMS serves as the disturbance introduced to the system for the study and it ranges from 0.5 kg to 35 kg. More weights could be accommodated for further studies but the diameter of the bowl would need to be increased and the complete design of the EMS would require total adjustments.

4.2. Formulation of the EMS NNARMAX Model Identification and Prediction Problem

4.2.1. Statement of the EMS Neural Network-Based Model Identification Problem

The development of accurate system models from first-principles or analytically for dynamical systems could be difficult and/or frustrating if not practically impossible [3, 15]. The modeling task becomes even more challenging for systems with relatively short sampling time such as the EMS considered in this study with an approximate sampling time of 16 milliseconds. However, the emergence of NNs simplified the process of capturing relatively accurate dynamic discrete-time models of dynamical systems based on the availability of either only input, only output or input-output data [1-15]. This sub-section develops on Sections 2 and 3.

Thus, from the discussions so far, the only measured input that influence the behaviour of the EMS is the input voltage (V_i) given by:

$$U(k) = \left[V_{i\_in}(k) \right]^T \tag{32}$$

Furthermore, based on the discussions thus far, the output parameter that can be used to determine the behaviour of the EMS is the speed (S_i in rpm) given by:

$$Y(k) = \left[S_{i\_\text{out}}(k) \right]^T \tag{33}$$

Although, the EMS is formulated as a SISO problem, the NN architecture is also a simplified SISO system. The series-parallel NN model identification scheme used here is shown in Fig. 4 and is based on the NNARMAX model predictor discussed in Section 3. The input vector to the neural network (NN) consists of the regression vectors which are concatenated into $\varphi_{NNARMAX}(k,\theta(k))$ for the NNARMAX model predictor discussed in Section 3 and defined here as follows:

$$\varphi_{n_a}(k) = \left[S_{i\_\text{out}}(k-n_a) \right]^T \tag{34}$$

$$\varphi_{n_b}(k) = \left[V_{i\_in}(k-n_a) \right]^T \tag{35}$$

$$\varphi_{n_c}(k) = \left[\varepsilon_{Si\_\text{out}}(k-n_c,\theta(k)) \right]^T \tag{36}$$

$$\varphi_{NNARMAX}(k,\theta(k)) = \left[\varphi_{n_a}(k) \;\; \varphi_{n_b}(k) \;\; \varphi_{n_c}(k,\theta(k)) \right] \tag{37}$$

Again, the output of the NN for the EMS is the predicted value of the speed (S_{i_out} in rpm) at each sampling instant given by:

$$\hat{Y}(k) = \left[\hat{y}_{Si\_\text{out}}(k) \right]^T \tag{38}$$

4.2.2. Experiment with EMS for Neural Network Training Data Acquisition

Based on previous discussions, the PMDC–based EMS can be considered as a SISO system, thereby eliminating the complications associated with a MIMO systems [41]. The input to the electromechanical system is an electrical voltage sent to the PMDC motor while the output is the rpm of the PMDC motor measured by an opto-sensor. The input-output measurements (data) obtained from the electromechanical system was used for the development of dynamic models which was used for the adaptive controllers design, since the dynamic model of the system will ensure stability in situations where the system operates outside the normal operating conditions.

For the purpose of neural network modeling of the system, a total of 476 data samples were obtained from the experiments performed on the designed and constructed EMS developed in [26]. Of the 476 experimental data acquired from the EMS, 381 data (representing 80%) is for the NN training while the remaining 95 data (representing 20%) have been reserved for the trained NN model validation. The entire simulation of the EMS is achieved using MATLAB and Simulink® software from The MathWorks [42].

4.2.3. Formulation of the Error Back-Propagation with Momentum (EBPM) Algorithm

In order to investigate the performance of the proposed MLMA, the so-called error back-propagation with momentum (EBPM) algorithm is used for this purpose. The EBPM algorithm is a variation of the standard back-propagation algorithm originally proposed by [37] which has been modified in [15] for use in this paper. The EBPM algorithm is summarized from [15] as follows:

1). The weight of a connection is adjusted by an amount proportional to the product of an error signal δ, on the unit k receiving the input and the output of the unit j sending the signal along the connection as follows:

$$\Delta w_{j,l} = \gamma \delta_j^p \hat{Y}_j^p(k) \tag{39}$$

2). If the unit is an output unit, the error signal is given by:

$$\delta_o^p = \left[Y_o^p(k) - \hat{Y}_o^p(k) \right] f_j'(\vec{a}_o^p) \tag{40}$$

For the logistic sigmoidal activation function $f_j(\bullet)$ defined in (9), the output $\hat{Y}^p(k)$ can be expressed as:

$$\hat{Y}^p(k) = f(\vec{a}^p) = \frac{1}{1+e^{-\vec{a}^p}} \tag{41}$$

so that the derivative of (41) can be expressed as:

$$\left.\begin{aligned} f'(\vec{a}^p) &= \frac{\partial}{\partial \vec{a}^p}\left(\frac{1}{1+e^{-\vec{a}^p}}\right) = \frac{1}{\left(1+e^{-\vec{a}^p}\right)^2}\left(-e^{-\vec{a}^p}\right) \\ &= \hat{Y}^p(k)\left[1-\hat{Y}^p(k)\right] \end{aligned}\right\} \tag{42}$$

and such that the error signal for an output unit can be expressed as:

$$\delta_o^p = \left[Y_o^p(k) - \hat{Y}_o^p(k)\right]\hat{Y}_o^p(k)\left[1-\hat{Y}_o^p(k)\right] \tag{43}$$

3). The error signal for a hidden unit is determined recursively in terms of error signals of the units to which it is directly connected and the weights of those connections. Thus, for the sigmoid activation function, we have:

$$\left.\begin{aligned} \delta_h^p &= f'(\vec{a}_h^p)\sum_{o=1}^{N_o}\delta_o^p w_{ho} \\ &= \hat{Y}_h^p\left[1-\hat{Y}_h^p(k)\right]\sum_{o=1}^{N_o}\delta_o^p w_{ho} \end{aligned}\right\} \tag{44}$$

4). From (39), the learning procedure requires that the change in weight be proportional to $\partial E^p / \partial w$ expressed as:

$$\Delta w_{j,l} = \gamma \frac{\partial E^p(k)}{\partial w_{j,l}} \tag{45}$$

The true gradient descent method requires that infinitesimal steps are taken. For practical purpose, the learning rate γ in (39) and (45) is chosen as large as possible without leading to oscillation. To avoid oscillation at large γ, the change in weight is made to be dependent on past weight change by adding a momentum term as follows:

$$\Delta w_{j,l}(k+1) = \gamma \delta_j^p \hat{Y}_j^p(k) + \alpha \Delta w_{j,l}(k) \tag{46}$$

where j indexes the presentation number and α is a constant which determines the effects of the previous weight change. When no momentum term is used, it can take a long time before the minimum is reached with a low learning rate, whereas for high learning rates the minimum is never reached because of the oscillations. When a momentum term is added, the minimum is reached faster [38-40].

4.2.4. Scaling the Training Data and Rescaling the Trained Network that Models the EMS

Due to the fact the input and outputs of a process may, in general, have different physical units and magnitudes; the scaling of all signals to the same variance is necessary to prevent signals of largest magnitudes from dominating the identified model. Moreover, scaling improves the numerical robustness of the training algorithm, leads to faster convergence and gives better models. The training data are scaled to unit variance using their mean values and standard deviations according to the following equations [3, 15, 27]:

$$\left.\begin{aligned} U^{(s)}(k) &= \frac{U(k)-\bar{U}(k)}{\sigma_{U(k)}} \\ Y^{(S)}(k) &= \frac{Y(k)-\bar{Y}(k)}{\sigma_{Y(k)}} \end{aligned}\right\} \tag{47}$$

where $\bar{U}(k)$, $\bar{Y}(k)$ and $\sigma_{U(k)}$, $\sigma_{Y(k)}$ are the mean and standard deviation of the input and output training data pair; and $U^{(S)}(k)$ and $Y^{(S)}(k)$ are the scaled inputs and outputs respectively. Also, after the network training, the joint weights are rescaled according to the expression

$$\hat{Y}(k,\hat{\theta}(k)) = \hat{Y}(k,\hat{\theta}(k))\sigma_{Y(k)} + \bar{Y}(k) \tag{48}$$

so that the trained network can work with other unscaled validation data and test data not used for training. However, for notational convenience, $U(k) = U^{(S)}(k)$ and $Y(k) = Y^{(S)}(k)$ shall be used in the discussion of results.

4.2.5. Training the Neural Network that Models the EMS

The NN input vector to the neural network (NN) is the NNARMAX model regression vector $\varphi_{NNARMAX}(k,\theta(k))$ defined by (37). The input $\varphi_{n_c}(k,\theta(k))$, that is the initial error estimates $\varepsilon(k,\theta(k))$ given by (3), is not known in advance and it is initialized to small positive random matrix of dimension n_c by n_c. The outputs of the NN are the predicted values of $\hat{Y}(k)$ given by (8).

For assessing the convergence performance, the network was trained for τ = 20 epochs (number of iterations) with the following selected parameters: $p=1$, $q=1$, $n_a=2$, $n_b=2$, $n_c=2$, $n_\varphi=6$ (NNARMAX), $n_h=5$, $n_o=1$, $\alpha_h=1e-6$ and $\alpha_o=1e-5$. The details of these parameters are discussed in Section 3; where p and q are the number of inputs and outputs of the system, n_a, n_b and n_c are the orders of the regressors in terms of the past values, n_φ is the total number of regressors (that is, the total number of inputs to the neural network), n_h and n_o are the number of hidden and output layers neurons, and α_h and α_o are the hidden and output layers weight decay terms. The two design parameters $\lambda_\tau = 10^{-3}$ and $s = 0.05$ were selected to initialize the MLMA algorithm. The maximum number of times the algorithm of Table 3 is implemented is 6 in all the simulations; that is $j_{\max} = 6$.

For the EBPM, the two design parameters are selected as $\gamma = 1$ and $\alpha = 0.5$.

The 381 training data is first scaled using equation (47) and the network is trained for τ = 20 epochs using the proposed MLMA and the EBPM algorithm proposed in Sections 3.3 and 4.2.3 respectively.

4.3. Validation of the Trained NNARMAX Model for the Modeling and Prediction of the EMS Dynamics

According to the discussion on network validation in Section 3.4, a trained network can be used to model a process once it is validated and accepted, that is, the network demonstrates its ability to predict correctly both the data that were used for its training and other data that were not used during training. The networks trained with the EBPM and the proposed MLMA algorithms needs to be validated with proposed three different methods by the use of scaled and unscaled 381 training data as well as with the 95 data reserved for the validation of the trained network for the EMS.

The three different validation techniques used to evaluate the performances of the two trained networks are: *1).* one-step ahead prediction of the scaled training data; *2).* one-step ahead prediction of the unscaled validation data; *3).* *K*-step ahead prediction of the unscaled training data; and *4).* Akaike's final prediction error (AFPE) estimate.

4.3.1. Network Training of the EMS Using EBPM and the Proposed MLMA Algorithms

The two training algorithms used here are the EBPM and the proposed MLMA algorithms discussed in Sections 4.2.3 and 3.3 respectively. The training data is first scaled according to equation (47) and the network is trained using the two algorithms.

After network training, the trained network is again rescaled according to (48) so that the resulting network can work with unscaled EMS real-time data. The performances of the EBPM and the MLMA algorithms are shown in Fig. 5 through Fig. 7 while the Table 7 presents the summary of the training and validation results for quick comparison.

The computation time for training the networks using each of the algorithms are shown in the first row of Table 7.

Although, the convergence curves of the EBPM and the MLMA algorithms for 20 epoch are not shown but the MPI for both algorithms are given in the third row of Table 7. As one can observe from Table 7, the MLMA has smaller MPI when compared to that of EBPM which is an indication of good convergence property of the MLMA at the expense of higher computation time when compared the small computation time used by the EBPM for 20 epochs as evident in the first row of Table 7.

The total square error (TSE) discussed in sub-section 3.1, for the network trained with the EBPM and the MLMA algorithms are given in the second row of Table 7. Again, the MLMA algorithm also has smaller TSE and minimum performance indices when compared to that of the EBPM algorithm. These small values of the TSE and the MPI indicate that MLMA performs better than the EBPM for the same number of iterations (epoch). These small errors suggest that the MLMA model approximates the EMS better due to the smaller errors when compared to those of the EBPM.

These small errors suggest that the network trained with the proposed MLMA algorithm approximates the dynamics of the EMS with better accuracy compared to that obtained by the network trained with the EBPM algorithm.

4.3.2. One-Step Ahead Predictions Simulation for the EMS

In the one-step ahead prediction method given by (8), the scaled training data are compared with the one-step ahead output predictions of the trained network and an assessment of their corresponding errors is made. The comparison of the one-step ahead predictions of the scaled training data (target output, blue —) against the trained network output predictions (red --*) by the networks trained for 20 epochs using the EBPM and the MLMA algorithms are shown in Fig. 5.

Table 7. Summary of the training results for the designed electromechanical motor system based on EBPM and MLMA

S/N	Performance Parameters	Training Algorithms	
		EBPM	MLMA
1.	Computation time for model identification (sec)	1.0614e+00	3.9228e+00
2.	Total square error (TSE)	1.6482e+01	2.9554e-03
3.	Minimum performance index (MPI)	2.6193e-01	5.8183e-05
4.	Mean value of one-step ahead prediction error of the scaled training data	5.3969e+01	5.4315e-04
5.	Mean value of one-step ahead prediction error of the unscaled validation (test) data	6.1027e-02	1.9426e-03
6.	Mean value of 5-step ahead prediction error of the unscaled training data	5.3905e+01	3.4321e-02
7.	Akaike's final prediction error (AFPE) estimate	3.9218e+01	9.2187e-03

The mean value of the one-step ahead prediction errors for the prediction of the scaled training data by the network trained using the EBPM and the MLMA algorithms are given in the fourth row of Table 7. It can be seen in Fig. 5, the network predictions of the training data based on the network trained using the MLMA algorithm closely match the original training data used whereas there are much prediction mismatch obtained with the network trained using the EBPM algorithm. Also, the smaller one-step ahead prediction error obtained using the network trained by the MLMA when compared to that by EBPM algorithm are also evident in the fourth row of Table 7. This error is an indication that the trained networks using the MLMA algorithm captures and approximates the nonlinear dynamics of the EMS accurately. This is further justified by the small mean value of the TSE obtained using MLMA algorithms given in the second row of Table 7.

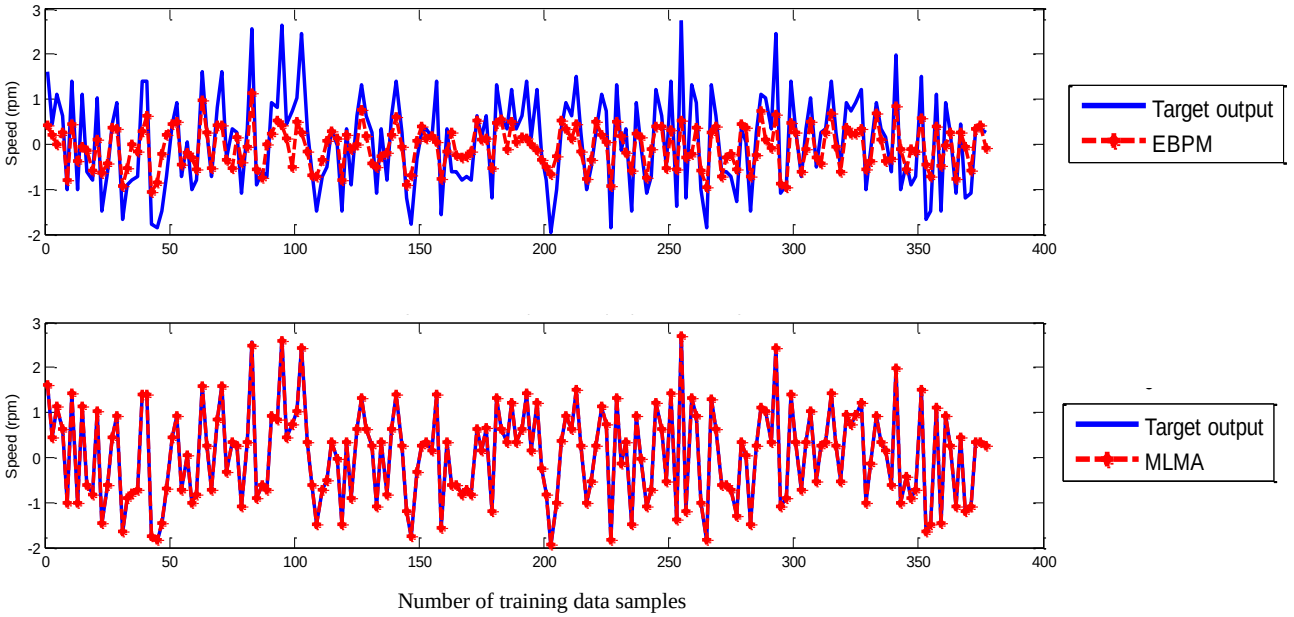


Figure 5. One-step ahead output prediction of scaled training data

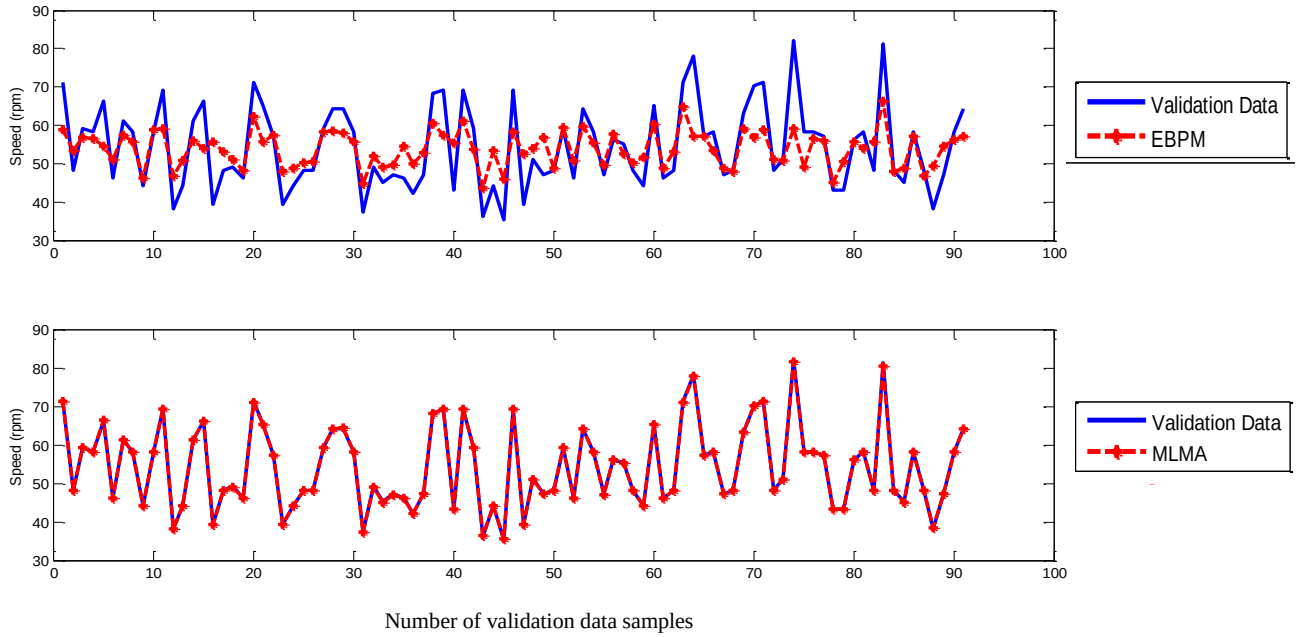


Figure 6. One-step ahead output prediction of unscaled validation (test) data

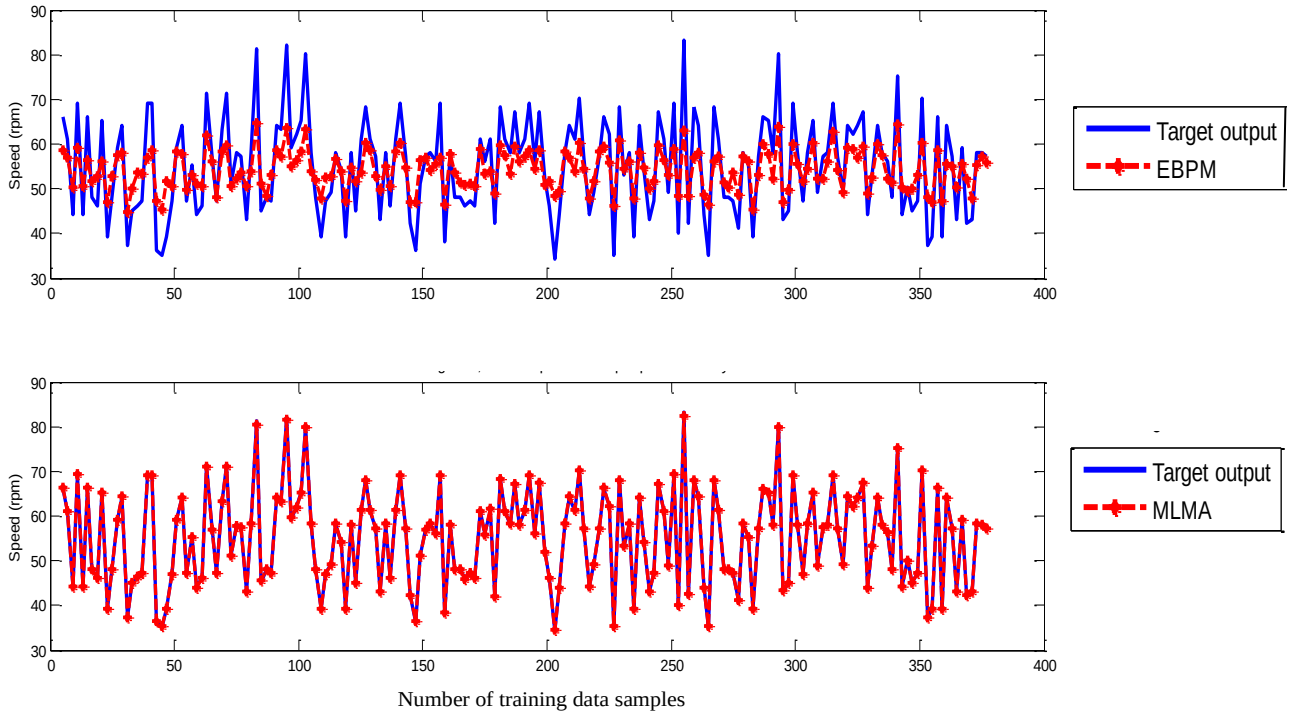


Figure 7. Five-step ahead output prediction of unscaled training data

Furthermore, the suitability of the EBPM and proposed MLMA algorithms for NN model identification for use in the EMS is investigated by validating the trained network with 95 unscaled test data. The comparison of the trained network predictions (red --*) of the test data with the actual test data (test data, blue —) for 20 epoch are shown in Fig. 6 for the EBPM and the MLMA algorithms. It is evident that the unscaled test data predictions by network trained using the MLMA algorithm match the true test data to a high accuracy when compared to that obtained by the network trained using EBPM. The superior performance of the proposed MLMA algorithm over the EBPM algorithm proves the effectiveness of the proposed MLMA approach.

The one-step ahead prediction accuracies of the unscaled test data by the networks trained using the EBPM and the MLMA algorithms is evaluated by the computed mean prediction errors shown in the fifth row of Table 7. It can be seen that the one-step ahead test data prediction errors by the network trained using MLMA algorithm are much smaller than those obtained from the network trained using the EBPM algorithm.

This one-step ahead unscaled validation data prediction results given by Fig. 6 as well as the mean value of the one-step ahead prediction error of the validation data shown in the fifth row of Table 7 justify that the network trained using the MLMA algorithm mimic the dynamics of the electromechanical system and that the resulting network can be used to model the actual EMS in an industrial environments and/or in real life scenarios.

4.3.3. K–Step Ahead Prediction Simulations for the EMS

The results of the *K*-step ahead output predictions (red --*) using the *K*-step ahead prediction validation method for 5-step ahead output predictions (K = 5) compared with the unscaled training data (target output) are shown in Fig. 7 for the network trained using the EBPM and MLMA algorithms. The value K = 5 is chosen since it is a typical value used in most model predictive control (MPC) applications. The comparison of the 5-step ahead output predictions performance by the network trained using EBPM and the MLMA algorithms shows the superior performance of the MLMA algorithm over the EBPM algorithms for use in distant or multi-step ahead predictions.

The computation of the mean value of the K-step ahead prediction error (MVPE) using equation (31) gives $5.3905\times10^{+01}$ and 3.4321×10^{-02} respectively by the network trained using the EBPM and MLMA algorithms as shown in the sixth row of Table 7. The relatively smaller MVPE obtained by the network trained with the MLMA algorithm is indications that the trained network approximates the dynamics of the EMS to a high degree of accuracy.

4.3.4. Akaike's Final Prediction Error (AFPE) Estimates for the EMS

The implementation of AFPE algorithm discussed in chapter four and defined by equation (29) for the regularized criterion for the network trained with the EBPM and the MLMA algorithms with multiple weight decay gives the respective AFPE estimates of the two algorithms as shown in the seventh row of Table 7.

These small values of the AFPE estimate indicate that the trained networks capture the underlying dynamics of the EMS and that the network is not over-trained [3, 15, 27]. This implies that optimal network parameters have been

selected including the weight decay parameters. Again, the results of the AFPE estimates obtained with the networks trained using the MLMA algorithm are by far smaller when compared to that obtained using EBPM algorithm.

4.4. Dynamic Model Validation and Closed-Loop Simulations of the EMS Using PID Controller

Besides the training of the NN model with static data taken from plant tests, it would be of interest to validate the prediction accuracy of a trained network under the same dynamic conditions in which the system is operating in the presence of a disturbance $\tilde{d}(k)$.

Disturbances are variables that fluctuate and cause the process outputs to move from the desired operating values (set-points or desired trajectories). The prescribed desired speed trajectory specified for the EMS is 60 rpm which must be maintained irrespective of the applied weight(s). A disturbance could be a change in flow or temperature of the surroundings or pressure etc. Disturbance variables can normally be further classified in terms of measured or unmeasured signals. The different weights (in kg) applied in this research serves as the disturbances introduced randomly to the EMS and it ranges from 0.5 kg to 35 kg.

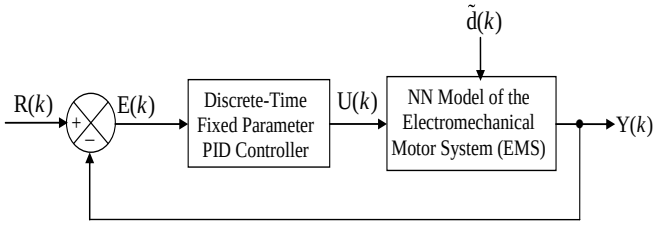


Figure 8. The Discrete-time PID control scheme

In the simplest case, the EMS affected by the above disturbance is controlled by a discrete-time fixed parameter PID controller used in a closed-loop configuration as illustrated in Fig. 8. This operation is imitated by placing the network trained by each one of the two algorithms in a control loop as it happens in real plants. The mathematical relationships implemented for the PID controller that computes the EMS control inputs $U(k) = [\mathrm{V}_{i\_}inD(k)]$ is:

$$\left.\begin{aligned} U(k) = K_P E(k) + K_I \frac{T}{2}\sum_{k=1}^{N}\left[E(k-1)+E(k)\right] \\ + K_D \frac{\left[E(k)-E(k-1)\right]}{T} \end{aligned}\right\} \quad (49)$$

where K_P, K_I and K_D are the proportional, integral and derivative gains respectively, T is the sampling time and $E(k) = R(k) - \hat{Y}(k)$ is the error between the desired reference $R(k)$ and predicted output $\hat{Y}(k) = [\mathrm{S}_{i\_out}]$ and N is the number of samples. The minimum and maximum constraints imposed on the PID controller to penalize changes on the EMS control inputs $U(k)$ and outputs $Y(k)$ are given as:

$$\left.\begin{aligned} U_{\min} \le U(k) \le U_{\max} \\ Y_{\min} \le Y(k) \le Y_{\max} \end{aligned}\right\} \quad (50)$$

A major problem with PID controllers is the "wind up" of the integrator resulting in the saturation of the integral term for control signal of large magnitude. However, rich literatures exist on anti-wind up techniques which addresses this problem [43, 44]. According to this technique, the integrator is switched off when the actuator output exceeds a predefined limit subject to input constraints imposed on the control inputs defined in (50) for the EMS.

The dynamic modeling and the closed-loop control with the PID controller shown in Fig 8 is simulated in MATLAB for 300 simulation samples with the disturbances discussed above. The PID controller parameters in (49) were selected to be $K_P = 25$, $K_I = 32$ and $K_D = 87$ for S_{i_out} (in rpm). The constraints imposed on the EMS defined in (50) are summarized in Table 8 together with the initial control inputs and outputs.

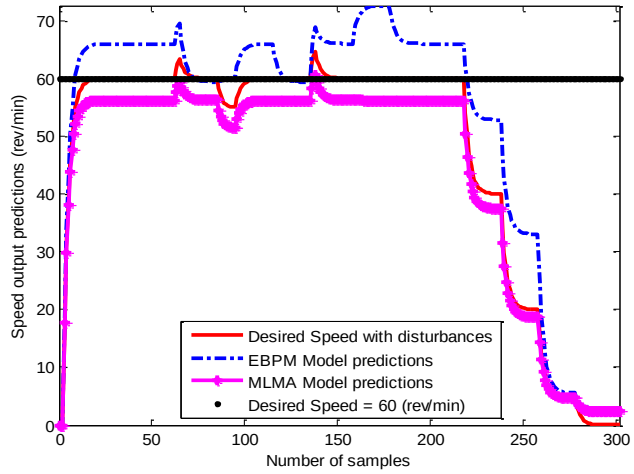
Table 8. Input and Output Constraints for the PID Control of the EMS

EMS Control Parameters	Constraints	
	EBPM	MLMA
Initial control input, $U(k)$	-20	-10
Initial control output, $Y(k)$	0	0
Minimum control input, $U_{\min}(k)$	0	0
Maximum control input, $U_{\max}(k)$	8.67	8.67
Minimum predicted output, $Y_{\min}(k)$	0	0
Maximum predicted output, $Y_{\max}(k)$	60	60
Desired reference signal, $R(k)$	60	60

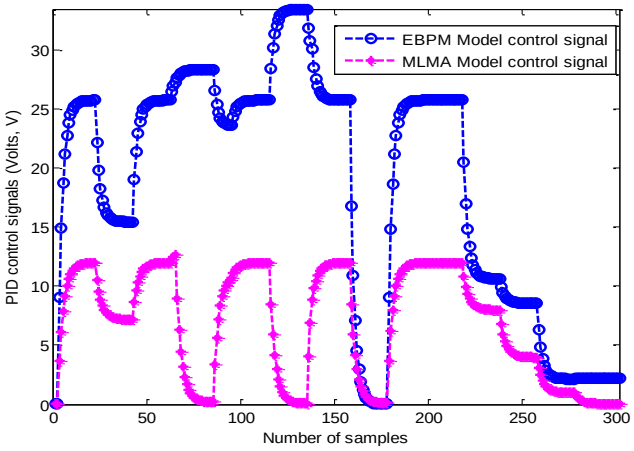
The results for the S_{i_out} output predictions (in rpm) is shown in Fig. 9(a) while the manipulated inputs, the V_{i_in} (in Volt, V) is shown in Fig. 9(b) using the models trained with EBPM and MLMA algorithms for 20 epochs. It can be observed that the model based on EBPM exhibits oscillatory behaviour in S_{i_out} predictions as in Fig. 9(a). This behaviour is not unusual because of the strong nonlinearity associated with the EMS especially at the initial application of the start-up voltage to initiate rotation.

Comparing the EMS discrete-time PID control performance of Fig. 9 based on the models obtained using EBPM and proposed MLMA algorithms, it is evident that the control result based on the model trained with the proposed MLMA gives good control performances even with the fixed parameter PID controller under disturbances.

With the dynamic feedforward neural network (DFNN) based on the teacher-forcing method and the MLMA training algorithm proposed in this work, changes on the process dynamics seem to be captured adequately.



(a)



(b)

Figure 9. Closed-loop PID control performance of the EMS using NN model trained with EBPM and MLMA algorithms: (a) output speed predictions and (b) control signals

Furthermore, this study has shown that the control performance based on the NN model trained using the MLMA in tracking the desired trajectory with small overshoot outperforms that based on the EBPM method with large overshoot as evident especially in Fig. 9(a) with reduced control effort as can be seen in Fig. 9(b).

5. Conclusions and Future Directions

This paper presents a novel technique for the dynamic modeling of an electromechanical motor system (EMS) and the closed-loop prediction of EMS behaviour in the presence of disturbances using an advanced online nonlinear model identification algorithm called the modified Levenberg-Marquardt algorithm (MLMA) based on artificial neural networks for the nonlinear model identification of an EMS. The paper also presents the complete formulation of the proposed MLMA.

In order to investigate the performance of the proposed MLMA algorithm, the error back-propagation with momentum (EBPM) algorithm is implemented and its performance compared with proposed MLMA. The simulation results from the application of these algorithms to the dynamic modeling of the EMS as well as the validation results show that the neural network-based MLMA outperforms the EBPM algorithm with much smaller predictions error and good tracking abilities with high degree of accuracy.

The simulation results from the dynamic modeling in closed-loop with a discrete-time fixed parameter PID control shows that the proposed MLMA model identification algorithm can be used for the EMS in real life scenarios and/or in industrial environments.

Although, it is evident the performance of the PID controller is not satisfactory due to poor tracking of the desired trajectory with to oscillations below the desired trajectory. Thus, the next aspect of the work could be on the development of efficient adaptive control algorithms to replace the fixed-parameter PID controller for the EMS so as to obtain an adaptive electromechanical speed control system.

REFERENCES

[1] M. Lazar and O. A. Pastravanu, "Neural predictive controller for non-linear systems", Technical University "Gh. Asachi" of Iasi Department of Automatic Control and Industrial Blvd. Mangeron 53A, 6600 Iasi, Romania Informatics, vol. 60, no (3-5), pp. 315 – 324, 2000.

[2] G. Feng and R. Lozano, *"Adaptive control systems",* Newness, Oxford, 1999.

[3] V. A. Akpan and G. D. Hassapis, "Nonlinear model identification and adaptive model predictive control using neural networks", *ISA Transactions,* vol. 50, pp. 177 – 194, 2011.

[4] Y. Jin and C. Su, "Adaptive model predictive control using diagonal recurrent neural network", *Fourth Int'l. Conf. on Natural Computation, Jinan,* pp. 276 – 280, 2008.

[5] F. S. Mjalli, "Adaptive and predictive control of liquid-liquid extractors using neural-based instantaneous linearization technique". *Chem. Eng. Technol.*, vol. 29, no. 5, pp. 539 – 549, 2006.

[6] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Networks,* vol. 1, no. 1, 4 – 27, 1990.

[7] M. Nørgaard, O. Ravn, N. K.Poulsen and L. K. Hansen, *"Neural networks for modelling and control of dynamic systems: A practitioner's handbook",* London: Springer-Verlag, 2000.

[8] O. M. Omidvar and D. L. Elliott, *"Neural systems for control",* Academic Press, San Diego, 1997.

[9] K. Salahshoor, E. Safari. and M. F. Samadi, M. F., "Adaptive model predictive control of a hybrid motorboat using self-organizing GAP-RBF neural network and GA algorithm",

2nd IEEE Int'l Conf. on Adv. Computer Control, Shenyang, China, vol. 27, no. 29, pp. 588 – 592, 2010.

[10] J. Sarangapani, *"Neural network control of discrete-time systems",* Boca Raton: CRC Press Abingdon, 2006.

[11] J. T. Spooner, M. Maggiore, R. Ordóñez. and K. M. Passino, *"Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques",* Wiley & Sons Inc., USA, 2002.

[12] C. Su and Y. Wu, "Adaptive neural network predictive control based on PSO algorithm" Chinese Control and Decision Conference, Guilin, China, vol. 17, no. 19, pp. 5829 – 5833, 2009.

[13] G. I. Suárez, O. A. Ortiz, P. M. Aballay and N. H. Aros, "Adaptive neural model predictive control for the grape juice concentration process", *2010 IEEE Int'l Conf. on Industrial Tech., Vi a del*, vol. 14, no. 17, pp. 57 – 63, 2010.

[14] D. W. Yu and D. L. Yu, D. L. "Multi-rate model predictive control of a chemical reactor based on three neural models" *Biochemical Engineering Journal*, vol. 37, pp. 86 – 97, 2007.

[15] V. A. Akpan *"Development of new model adaptive predictive control algorithms and their implementation on real-time embedded systems",* Ph.D. Dissertation, Department of Electrical and Computer Engineering, School of Engineering, Aristotle University of Thessaloniki, Greece, 517 pages, July, 2011. [Online] Available: http://invenio.lib.auth.gr/record/127274/files/GRI-2011-7292.pdf.

[16] J. R. Cowan and W. N. Myers, "Design of high power electromechanical actuator for thrust vector control", *AIAA 27th Joint Propulsion Conference (AIAA'91), Sacramento, CA, U.S.A*, pp. 1849, 1991.

[17] R. A. Weir and J. R. Cowan, "Development and test of electromechanical actuators for thrust vector control", *29th Joint Propulsion Conference/Monterey, CA, U.S.A*, pp. 349 – 366, 1993.

[18] G. Zhong. and G. L. Jiang, "Design of the closed loop speed control system for DC motor", *CCSE computer and information science*, vol. 2, no. 1, pp. 95 – 103, 2009.

[19] S. Weerasooriya and M. A. El-Sharkawi, "Identification and control of a DC motor using back-propagation neural networks", *IEEE Trans. Energy Conversion*, vol. 6, pp. 663 – 669, 1991.

[20] M. Ristanovic, Z. Cojbasic and D. Lazic, "Intelligent control of DC motor driven electromechanical fin actuator", *Control Eng Pract.*, vol. 20, no. 6, pp. 610 – 617, 2012.

[21] J. Mendes, J. R. Arau, P. Sousa, S. F. Apo and L. Alves, "An architecture for adaptive fuzzy control in industrial environments", Computer in Industry, vol. 62, pp. 364 – 373, 2011.

[22] D. Anurag, "Speed control of DC shunt motor with field and armature rheostat control simultaneously" *Advances in Electronic and Electric Engineering*, vol. 3, no. 1, pp. 77 – 80, 2013.

[23] B. A. A. Omar, A. Y. M. Haikal and F. F. G. Areed, "Design adaptive neuro-fuzzy speed controller for an electro-mechanical system", *Ain Shams Engineering Journal*, vol. 2, pp. 99 – 107, 2011.

[24] R. G. Madhusudhana and R. B. V. Sanker, "A neural network based speed control for DC motor", *Int'l J. Recent Trends Eng.*, vol. 2, no. 6, pp. 121 – 124, 2009.

[25] E. F. Fuchs and M. A. S. Masoum, *"Power conversion of renewable Energy systems",* Springer; XIII, Hardcover, 2011.

[26] M. T. Babalola, V. A. Akpan and C. O. Ajayi, "Neural Network-Based Adaptive Speed Controller Design for Electromechanical Systems (Part 1: System Design & Instrumentation)', *American Journal of Intelligent Systems*, vol. 6, no. 3, pp. 1 – 11, 2016. (Accepted).

[27] V. A. Akpan and G. D. Hassapis, "Training dynamic feedforward neural networks for online nonlinear model identification and control applications", *International Reviews of Automatic Control: Theory & Applications*, vol. 4, no. 3, pp. 335 – 350, 2011.

[28] L. Ljung, *"System identification: theory for the user",* 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1999.

[29] S. Haykin, *"Neural networks: A comprehensive foundation",* 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1999.

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

[31] R. Chiong, "Intelligent systems for automated learning and adaptation: emerging trends and applications", *International Journal Information and decision Science*, vol. 2, no. 4, pp. 427 – 430, 2010.

[32] J. Wu, "Multilayer potts perceptrons with Levenberg-Marquardt learning", *IEEE Trans. on Neural Netw*, vol. 19, no. 12, pp. 2032 – 2043, 2008.

[33] D. Mirikitani and N. Nikolaev, Recursive Bayesian Levenberg-Marquardt training of recurrent neural networks. *In. Proc. of Int'l Joint Conf. on Neural Netwk.,* Florida, USA, 12 – 17 August, 2007, pp. 282 – 287.

[34] J. Sjöberg and L. Ljung, "Overtraining, regularization, and searching for minimum in neural networks", *Int'l J. of Control.*, vol. 62: 1391 – 1408, 1995.

[35] R. Fletcher, *"Practical Methods of Optimization",* 2nd ed., Wiley & Sons, 1987.

[36] M. J. Willis, (1999) *"Some conventional process control schemes",* Department of Chemical and Process Engineering University of Newcastle upon Tyne, vol. 138, no. 3, pp. 256 – 266, 1999.

[37] P. J. Werbos, "Backpropagation through time: What it does and how to do it". In *Proc. IEEE*, vol. 78, no. 10, pp. 1550 – 1560, 1990.

[38] V. V. Phansalkar and P. S. Sastry, "Analysis of the Back-Propagation Algorithm with Momentum". *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 505 – 506, 1994.

[39] X. G. Wang, Z. Tang, H. Tamura, M., Ishii and W. D. Sun, "An improved backpropagation algorithm to avoid the local minima problem". *Neurocomputing*, vol. 56, pp. 455 – 460, 2004.

[40] Yu, X., Loh, N. K. and Miller, W. C. (1993). "A new

acceleration technique for the back propagation algorithm". *In Proc. Int'l Conf. on Neu. Netw.,* San Francisco, 28th March -1st April, pp. 1157 – 1161.

[41] R. Grepl, "Modelling and control of electromechanical servo system with high nonlinearity", *Mechatronic System Simulation, Modelling and Control,* vol. 26, no. 2, pp. 307 – 319, 2010.

[42] The MathWorks Inc., MATLAB & Simulink R2009b, *Natick,* USA. www.mathworks.com.

[43] A. Visioli, *Practical PID Control* (Springer-Verlag Ltd., 2006).

[44] P. Hippe, *Windup in Control,* (Springer-Verlag Ltd., 2006).