Prediction of Joint Moments Using a Neural Network Model of Muscle Activations From EMG Signals

Lin Wang and Thomas S. Buchanan, *Member, IEEE*

***Abstract*—Because the relationship between electromyographic (EMG) signals and muscle activations remains unpredictable, a new way to determine muscle activations from EMG signals by using a neural network is proposed and realized. Using a neural network to predict the muscle activations from EMG signals avoids establishing a complex mathematical model to express the muscle activation dynamics. The feed-forward neural network model of muscle activations applied here is composed of four layers and uses an adjusted back-propagation training algorithm. In this study, the basic back-propagation algorithm was not applicable, because muscle activation could not be measured, and hence the error between predicted activation and the real activation was not available. Thus, an *adjusted* back-propagation algorithm was developed. Joint torque at the elbow was calculated from the EMG signals of ten flexor and extensor muscles, using the neural network result of estimated activation of the muscles. Once muscle activations were obtained, Hill-type models were used to estimate muscle force. A musculoskeletal geometry model was then used to obtain moment arms, from which joint moments were determined and compared with measured values. The results show that this neural network model can be used to represent the relationship between EMG signals and joint moments well.**

***Index Terms*—Artificial neural network, back-propagation, muscle models.**

I. Introduction

OVER the years, the neurophysiology and biomechanics of muscle systems have been investigated quite extensively in order to characterize the relations between muscle activity (EMG) and various dynamic and/or kinematic aspects of the movement behavior. The EMG signal is a direct reflection of muscle activity. Raw EMG activity increases both as the firing rates of individual motor units rise and as previously inactive units become recruited. Nevertheless, the relationship between EMG and muscle activation remains unclear. A new method to predict muscle activation from EMG signals is proposed in this paper. This method involves the novel combination of artificial neural networks with a Hill-type muscle–tendon model that will transform the muscle activations to muscle forces.

The major advantages of neural networks are that they exhibit adaptation and learning, they are fault tolerant, and they avoid establishing a complex mathematical model. Another important advantage is that the neural network could lump muscles together and reduce the number of variables. It has been argued that for multijoint movements, each muscle is not always treated by the nervous system as a separate variable [1]. That is, muscles could be coupled together such that one muscle's activation is relevant to another's.

As Fig. 1 shows, the transformation from EMG to muscle force can be divided into activation dynamics and contraction dynamics [2]. The excitation of muscle tissue (neural excitation) acts through activation dynamics to generate an internal muscle tissue state, which is called muscle activation. Through muscle contraction dynamics, muscle activation energizes the cross bridges and muscle force is developed. Both the force–length and force–velocity relationships are dependent on muscle activation [2]. To estimate accurately the muscle forces, the first step is to estimate muscle activation correctly. Once muscle forces are found, they can be multiplied by the respective muscle moment arms and summed to yield the joint moment.

The existing applications of artificial neural networks in biomechanics have dealt primarily with the estimation of joint angles and moments in gait stimulation. Sepulveda *et al.* made use of artificial neural network with the back-propagation algorithm to map two different transformations: 1) EMG $\rightarrow$ joint angles and 2) EMG $\rightarrow$ joint moments. Both networks were successfully trained to map the input vector onto the output vector [3]. This type of model does not account for known relationships involving muscle moment arms, contraction velocity, joint angles, etc. Luh *et al.* constructed a three-layer feed-forward network with an adaptive learning rate to determine the relations between the EMG activity and isokinetic elbow joint torque [4]. The input signals of the neural network included not only novel EMG but also joint position and joint angular velocity. The output was the prediction of isokinetic joint torque. Based on Fig. 1, their work used a neural network to replace blocks 1, 2, 3.

Savelberg *et al.* did similar work [5], [6]. They used an artificial neural network to predict dynamic tendon forces from EMG signals in an animal model. Liu *et al.* also used an artificial neural network approach to predict dynamic muscle force prediction from EMG [6]. The neural networks created in these studies were used to nonlinearly map the relationship between EMG activity and muscle–tendon force. Their neural network replaced muscle model blocks #1 and #2.

Despite the success of such models, there are several known factors about the generation of muscle forces and joint moments that are generally neglected in this approach. For example, it is known that muscle forces change with length and velocity and that muscle moment arms change with joint angles. Since the neural network approach is essentially a black box, one could

Manuscript received June 15, 2001; revised December 7, 2001. This work was supported in part by the National Institutes of Health under AR40408 and AR46386.

The authors are with the Center for Biomedical Engineering Research, University of Delaware, Newark, DE 19716 USA (e-mail: buchanan@udel.edu).

Publisher Item Identifier S 1534-4320(02)05845-X.

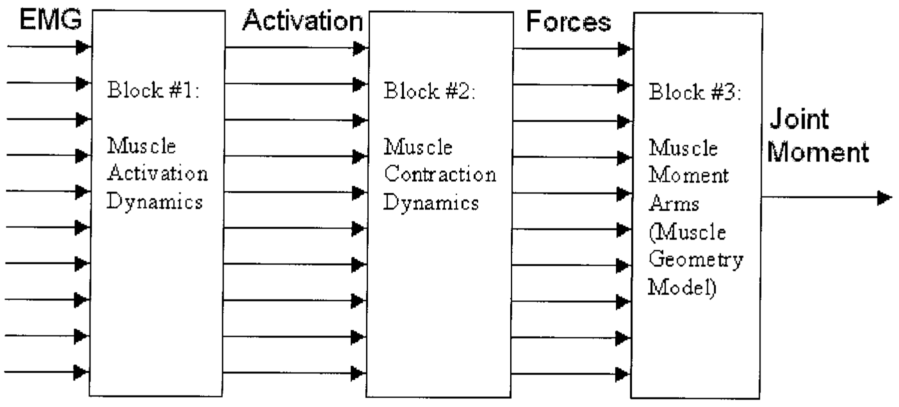
EMG
Activation
Forces
Block #1: Muscle Activation Dynamics
Block #2: Muscle Contraction Dynamics
Block #3: Muscle Moment Arms (Muscle Geometry Model)
Joint Moment

Fig. 1. A block diagram of the transformation from EMG to joint movement. Muscle activation dynamics transforms EMG to muscle activation. Muscle contraction dynamics transforms activation to muscle forces by using Hill-type muscle models that account for the length–tension and force–velocity relationships, etc. Finally, the muscle geometry model accounts for muscle moment arms and transforms muscle forces to muscle joint moment contributions. These are summed to obtain joint moment.

argue that such things are absorbed in the neural network model. However, since such relationships are established, we thought it might be best to take advantage of them by using the neural network to only model that which is not well established: the muscle activation dynamics (block #1). This offers the advantage of allowing us to use established Hill-type models when computing muscle forces that account for physiologically established relationships such as the force–length and force–velocity curves. This way, we can also use established musculoskeletal models that characterize changes in muscle lengths and moment arms as functions of joint angles. In this way, we are modeling only the muscle activation dynamics as a black box, which is reasonable since these relationships are not well established.

The goal of this paper is to establish a neural network model for muscle activation dynamics. An adjusted back-propagation algorithm was created in order to solve the problem due to the unavailability of accurate muscle activations. The joint moment was then calculated using the muscle activations, which the neural network model predicted.

II. Methods

A. Neural Network Model

The basic structure of the neural network proposed in this paper (Fig. 2) consists of four arrays of several neurons and interconnections between all elements from consecutive rows. The first layer, also called the input layer, has ten neurons. The second and third layers are hidden layers, which have 15 neurons each. The fourth layer, also called the output layer, has ten neurons. Inputs to the network are the normalized EMG magnitudes of the ten muscles under study (biceps long head, biceps short head, triceps long head, triceps lateral head, triceps medial head, brachialis, brachioradialis, pronator teres, extensor carpi radialis longus, and anconeus). Outputs are the corresponding ten muscles' activations. A node's input is determined by multiplying each input signal by the corresponding connection weight. The net input is transformed to an output signal by a sigmoid activation function. The sigmoid function is used because of its nonlinearity. By using this sigmoid function,

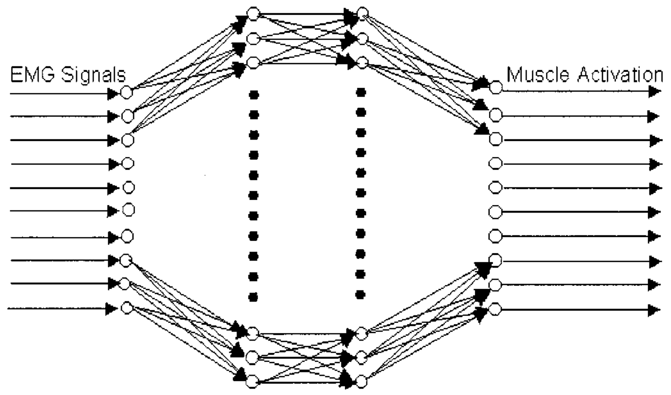

Fig. 2. The four-layer neural network model representing the EMG-to-activation relationship.

the output layer unit is constrained to generate signals between zero and one.

The artificial neural network has a specific procedure to "catch" the information contained in the given data. This procedure is called the "learning" process, and the given data provided for the artificial neural network to begin learning are called the "training set." The error back-propagation learning algorithm is one of the most popular training algorithms. It is composed of two stages: a feed-forward step, where the neural nodes' output is specified; and a learning stage, where the connection weights and bias terms are updated. The two steps are repeated until the difference between the network predicted output signal (predicted muscle activation) and desired output signal (measured muscle activation) is below a specified tolerance value.

B. Neural Network Algorithm

The objective of this research is to predict muscle activation from muscle EMG signals. To do this, we need a set of training data, which includes the ten muscles' measured EMG signals and the corresponding muscles' activations. Unfortunately, muscle activation is not a measurable quantity. However, we can measure elbow joint moment M. Hence, we will use this in our adjusted algorithm.

Error in the predicted joint moment can be described as follows:

$$\Delta M = \frac{\partial M}{\partial a_1}\Delta a_1 + \frac{\partial M}{\partial a_2}\Delta a_2 + \cdots + \frac{\partial M}{\partial a_{10}}\Delta a_{10} = \sum_{i=1}^{10}\frac{\partial M}{\partial a_i}\Delta a_i \tag{1}$$

where a_i stands for the ith muscle activation and M is the measured elbow joint moment. ΔM is the difference between predicted joint moment with measured joint moment.

For a Hill-type muscle model, joint moment is a function of muscle activation, muscle length, and contraction velocity. This can be expressed by (2)

$$M = M(a_1, a_2, \ldots, a_{10}, l_1, l_2, \ldots, l_{10}, v_1, v_2, \ldots, v_{10}). \tag{2}$$

The partial derivative of elbow joint moment M referent to muscle activation $\partial M/\partial a_i$ in (1) can be solved by (3)

$$\begin{aligned}\delta M =& M(a_1, a_2, \ldots, a_{10}, l_1, l_2, \ldots, l_{10}, v_1, v_2, \ldots, v_{10}) \\ &- M(a_1, a_2, \ldots (a_i + \delta a_i) \ldots, a_{10}, l_1, l_2, \ldots, l_{10}, v_1, v_2, \ldots, v_{10}) \\ \frac{\partial M}{\partial a_i} \approx& \frac{\delta M}{\delta a_i}\end{aligned} \tag{3}$$

where δa_i is a small change at a_i and δM is the corresponding change in joint moment while other parameters kept unchanged except adding δa_i to a_i. Although ΔM and $\partial M/\partial a_i$ are available, we cannot solve $\Delta a_1, \Delta a_2, \ldots \Delta a_{10}$ from (1).

To solve this problem, we need to see the details in a basic back-propagation algorithm [7]. In mathematical terms, we may describe a neuron k by writing the following pair of equations:

$$u_k = \sum_{j=1}^{p} w_{kj}x_j \tag{4}$$

$$y_k = \psi(u_k - \theta_k) \tag{5}$$

where $x_1, x_2, \ldots, x_p$ are the input signals; $\omega_{k1}, \omega_{k2}, \ldots, \omega_{kp}$ are the synaptic weights of neuron k; u_k is the linear combiner output; θ_k is the threshold; $\psi(\bullet)$ is the sigmoid activation function; and y_k is the output signal of the neuron. The threshold θ_k is an external parameter of artificial neuron k. We may formulate the combination of (4) and (5) as follows:

$$v_k = \sum_{j=0}^{p}\omega_{kj}x_j \quad x_0 = -1 \quad \omega_{k0} = \theta_k \tag{6}$$

$$y_k = \psi(v_k). \tag{7}$$

Each hidden or output neuron of a multilayer neural network is designed to perform two computations. The first is the computation of the function signal appearing at the output of a neuron, which is expressed as a continuous nonlinear function of the input signals and synaptic weights associated with that neuron. The second computation is an instantaneous estimate of the gradient vector (i.e., the gradient of the error surface with respect to the weights connected to the inputs of a neuron), which is needed for the backward pass through the network.

The error signal at the output of neuron j at iteration n (i.e., presentation of nth training pattern) is defined by

$$e_j(n) = d_j(n) - y_j(n) \tag{8}$$

where neuron j is an output node and $e_j(n)$ refers to the error signal at the output of neuron j; $d_j(n)$ refers to the desired response for neuron j; and $y_j(n)$ refers to the function signal appearing at the output of neuron j.

We define the instantaneous value of the squared error for neuron j as 1/2 $e_j^2(n)$. Correspondingly, the instantaneous value $\xi(n)$ of the sum of squared errors is obtained by summing 1/2 $e_j^2(n)$ over all neurons in the output layer

$$\xi(n) = \tfrac{1}{2}\sum_j e_j^2(n). \tag{9}$$

The average squared error is obtained by

$$\xi_{av}(n) = \frac{1}{N}\sum_j \xi(n) \tag{10}$$

where N is the total number of patterns contained in the training set. The objective of the learning process is to adjust the weights of the network so as to minimize $\xi_{av}(n)$.

Case I—Neuron j Is in the Output Layer: The correction $\Delta\omega_{ji}(n)$ applied to $\omega_{ji}(n)$ is defined by

$$\Delta\omega_{ji}(n) = -\eta\frac{\partial\xi(n)}{\partial\omega_{ji}} = \eta\delta(n)_j y_i(n) \tag{11}$$

$$\delta_j(n) = -\frac{\partial\xi(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\psi_j'(v_j(n)) \tag{12}$$

where η is a constant that determines the rate of learning; it is called the *learning-rate parameter* of the back-propagation algorithm. The use of a minus sign accounts for gradient descent in weight space. The net internal activity level $v_j(n)$ produced at the input of the nonlinearity associated with neuron j is

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n)y_i(n). \tag{13}$$

Case II—Neuron j Is Located in a Hidden Layer: When neuron j is located in a hidden layer of the network, there is no specified desired response for that neuron. Accordingly, the error signal for a hidden neuron would have to be determined recursively in terms of the error signals of all the neurons to which that hidden neuron is directly connected

$$\Delta\omega_{ji}(n) = -\eta\frac{\partial\xi(n)}{\partial\omega_{ji}} = \eta\delta_j(n)y_i(n) \tag{14}$$

$$\begin{aligned}\delta_j(n) &= -\frac{\partial\xi(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= \psi_j'(v_j(n))\sum_k \delta_k(n)\omega_{kj}(n).\end{aligned} \tag{15}$$

We may now summarize the relations that we have derived for the back-propagation algorithm. First the correction $\Delta\omega_{ji}(n)$

applied to the synaptic weights connecting neuron i to j is defined by the delta rule

$$\Delta\omega_{ji}(n) = -\eta \frac{\xi(n)}{\omega_{ji}} = \eta\delta_j(n)y_i(n) \tag{16}$$

where η is the learning-rate parameter, $\delta_j(n)$ is the local gradient, and $y_i(n)$ is the input signal of neuron j. Second, the local gradient $\delta_j(n)$ depends on whether neuron j is an output node or a hidden node.

1) If neuron j is an output node, $\delta_j(n)$ equals the product of the derivative $\psi_j'(v_j(n))$ and the error signal $e_j(n)$, both of which are associated with neuron j.
2) If neuron j is a hidden node, $\delta_j(n)$ equals the product of the associated derivative $\psi_j'(v_j(n))$ and the weighted sum of the δs computed for the neurons in the next hidden or output layer that are connected to neuron j.

From the introduction of the back-propagation algorithm, we notice that there is no specified desired response for hidden neurons, which is very similar to the problem we meet with the neural network model for EMG to muscle activation. To be more specific, what we need to train our neural network is the correction $\Delta\omega_{ji}(n)$. So we can treat the output layer of our neural network model as a hidden layer

$$\Delta\omega_{ji}(n) = -\eta \frac{\partial\xi(n)}{\partial\omega_{ji}} = \eta\delta_j(n)y_i(n) \tag{17}$$

$$\delta_j(n) = -\frac{\partial\xi(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial\xi(n)}{\partial y_j(n)}\psi_j'(v_j(n)). \tag{18}$$

What we need to know is $\partial\xi(n)/\partial y_j(n)$.

For our EMG-to-muscle-activation neural network model, the instantaneous sum of squared error is

$$\xi(n) = \tfrac{1}{2}(M_{\text{measured}} - M_{\text{output}})^2; \quad y_j(n) = a_j(n) \tag{19}$$

where M_{measured} is measured elbow joint moment, M_{output} is the calculated joint moment using the muscle activation output of neural network model, $a_j(n)$ is the jth muscle activation, i.e., the output of the neural network model. Thus we can obtain

$$\frac{\partial\xi(n)}{\partial y_j(n)} = -\frac{\partial M_{\text{output}}}{\partial a_j} \tag{20}$$

$$\delta_j(n) = \frac{\partial M_{\text{output}}}{\partial a_j}\psi_j'(v_j(n)) \tag{21}$$

$$\Delta\omega_{ji} = \eta\delta_j(n)y_i(n) = \eta\frac{\partial M_{\text{output}}}{\partial a_j}\psi_j'(v_j(n))\,y_i(n). \tag{22}$$

For the previous layers, we continue to use the back-propagation algorithm to adjust the synaptic weights.

C. Supplemental Algorithms

There is an eletromechanical delay between EMG signals and muscle activation. To treat this factor, we add a time-delay parameter Δt in our model. During the training process, the training set data were composed of the EMG signals at time t and the joint moment at $t + \Delta t$. During the predicting process, we used EMG signals at time $t - \Delta t$ to predict the muscle activation at the current time.

Because the muscle itself functions as a low-pass filter, we added a low-pass filter (3 Hz, second-order Butterworth) between the neural network model and Hill-type muscle model. There is no such low-pass filter used in the training process.

The muscle contraction dynamics employed here have been previously described [8], [9] and were based on the models described by Zajac [2]. The musculoskeletal geometry model that was used to compute the moment arms and muscle lengths has been described by Murray *et al.* [10], [11].

III. Experiments

The subject was seated with his shoulder at 90° abduction, 0° extension, and no external rotation, and the elbow at 90° flexion with neutral pronation–supination. The shoulder was secured with straps, and the flexion axis of the elbow was in line with the vertical axis of the rotation table. The subject's torso was stabilized using automobile seat belts that passed over the shoulders, crossed at the chest, and were secured at the back. This fixed the subject in the heavy, rigid chair in order to minimize shoulder movement during the study. A Fiberglass cast approximately 5 cm wide was made on the subject's distal forearm just proximal to the ulnar styloid. The cast was bolted to a six-degree-of-freedom load cell with a resolution of 0.56 N (Assurance Technologies, Inc., Model 150-600).

EMG data were collected from ten muscles simultaneously during isometric flexion–extension time-varying loads. Intramuscular electrodes were placed into all muscles under study. These electrodes were used to avoid signal contamination from nearby muscles and were made from 75-mm-diameter Teflon-coated stainless-steel wire. The Teflon coating was removed from 0.5 cm of either end of the wire, and the wire was placed using a 27-gauge hypodermic needle. Two electrodes were placed in each muscle, approximately 2 cm apart, in locations as described by Perotto [12], and tests were performed to verify proper placement. All electrodes were connected to a custom-made preamplifier: a two-pole, high-pass filter with 30-Hz cutoff and a gain of 1000. The signal was then sent to a filter/amplifier, an eight-pole low-pass Butterworth filter with 300-Hz cutoff, and a variable gain between 2.5 and 40. Elbow torque was recorded from the load cell rigidly. Myoelectric and load cell signals were sampled at 1000 Hz via analog-to-digital converter and stored in the memory of an Intel-based PC. EMGs and joint moments were recorded during maximal voluntary contractions, and these maximal values were used to normalize the EMG data.

Three experimental protocols were employed. In Experiment 1, the subject was asked to perform one cycle of 75% maximum isometric extension and flexion. In Experiment 2, the subject was asked to perform a ramp up to 50% maximum isometric flexion followed by a ramp down to relaxation. In Experiment 3, the subject was asked to perform three cycles of 100% maximum isometric extension and flexion. The EMG measured in the 100% maximum isometric contraction was used as reference of normalization. EMG data were processed (filtered, full-wave rectified, and normalized) and passed to neural network model.

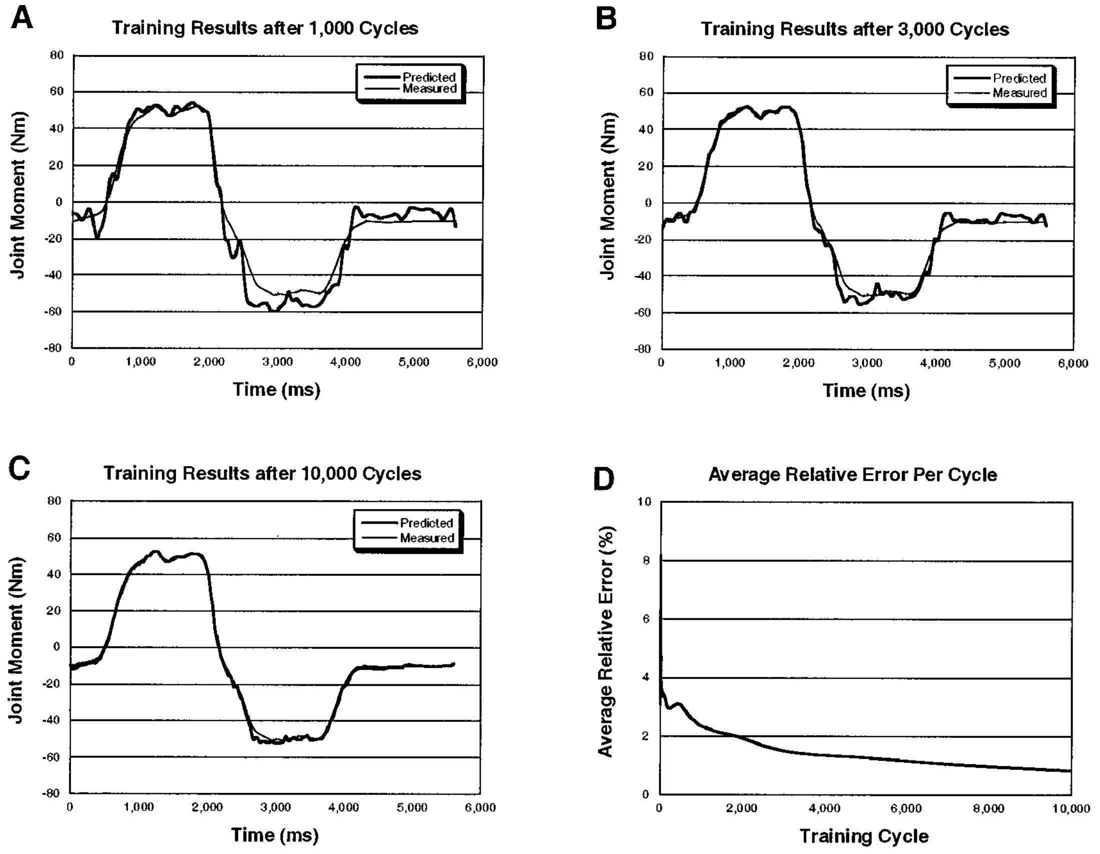

Fig. 3. Training results from the neural network model. Predicted elbow flexion moment is compared with measured joint moment after (A) 1000 cycles, (B) 3000 cycles, and (C) 10 000 cycles. Panel (D) shows how the error diminished with the number of cycles.

The outputs of the neural network model, the estimated muscle activation, were used as input to a Hill-type muscle model to calculate the elbow joint torque. We then compared the calculated value with the measured torque, used the torque error to calculate the error of estimated activation, and updated the connection weights and bias terms (as described in Section II-B) until the torque error was below a specified tolerance value or until the preset number of training cycles has been reached.

The data from Experiment 1 were used for training the neural network, and the data from Experiments 2 and 3 were used to test the neural network prediction performance.

IV. Results

The data from a trial of Experiment 1 were used to train the neural network model. All synaptic weights were randomized at the beginning of the training process. An adjusted back-propagation algorithm was applied.

After an appropriate number of training cycles, the model accurately estimated the joint moment of the training set (Fig. 3). After the neural network had been trained for 1000 cycles, the average relative error was 3% [Fig. 3(A)]. The training of 1000 cycles took approximately 30 min running on a PC with an Intel PII 400-MHz CPU. After 3000 training cycles, the average relative error was 1.5% [Fig. 3(B)]. The time cost was about 1.5 h. After 10 000 training cycles, there was no discernible difference between predicted and measured values [Fig. 3(C)]. The average relative error was 0.8%. The time required for the neural network to be trained was around 5 h. It can be seen that the average relative error got smaller as the number of training cycles increased [Fig. 3(D)]. The error decreased very quickly at first with the increase in the number of the training cycles, and then the subsequent addition of training cycles yielded diminished benefits.

All the information about the relationship between EMG signals and muscle activation was saved in the adjusted synaptic weights. Keeping these weights unchanged and being fed with new EMG signals, the neural network model could predict the corresponding muscle activation.

In the first prediction test, the same Experiment 1 task was chosen (Fig. 4). A new trial from Experiment 1 from the same subject was used as input, and the neural network model produced the predicted muscle activation. With these muscle activations and Hill-type muscle models, the predicted elbow joint moment was calculated. The predicting period takes no more than a minute since all this required was a single forward pass through the (already trained) neural network. The predicted average relative error was 8.3%. The covariance was 7.4%. This error is substantially lower than that obtained when this muscle activation step is ignored and rectified and filtered EMG is used as input to the model of muscle contraction dynamics (relative error: 15.3%). The results show the neural network model works well for a different trial of the same experiment as the training data.

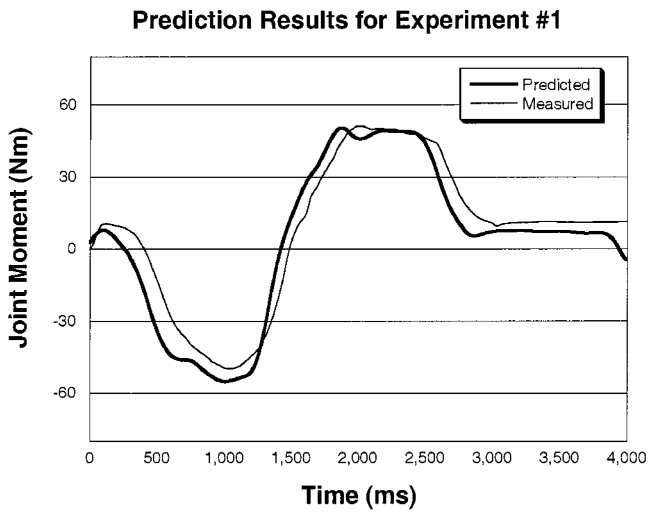

Fig. 4. Comparison of the predicted elbow flexion moments with the measured joint moments. This figure shows the predictions of joint moments based on use of the neural network estimations of muscle activations from Experiment 1. The model was trained on a data set obtained from a single trial during Experiment 1: a similar cyclic movement of about 0.3 Hz (but a different trial than used for prediction).

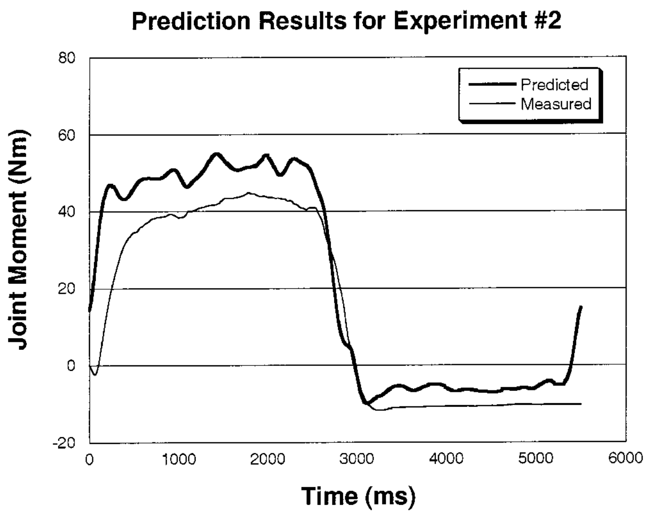

Fig. 5. Comparison of the predicted elbow flexion moments with the measured joint moments. This figure shows the predictions of joint moments based on use of the neural network estimations of muscle activations from Experiment 2. The model was trained on a data set obtained from a single trial during Experiment 1.

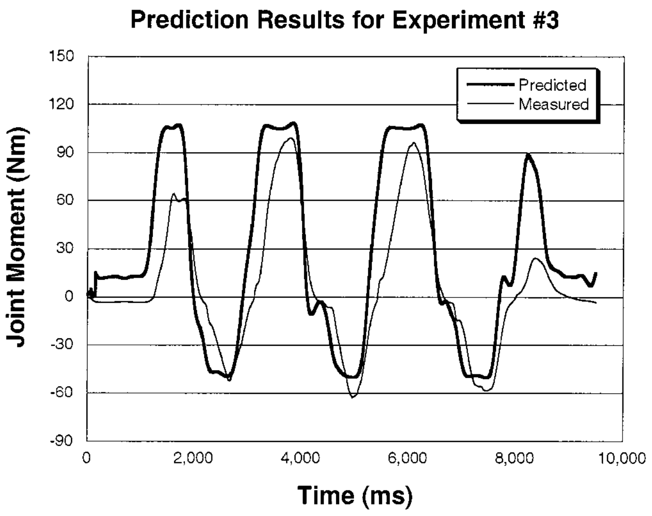

Fig. 6. Comparison of the predicted elbow flexion moments with the measured joint moments. This figure shows the predictions of joint moments based on use of the neural network estimations of muscle activations from Experiment 3. The model was trained on a data set obtained from a single trial during Experiment 1.

In Prediction Test 2, we tried to apply the trained neural network to predict muscle activation in Experiment 2 (Fig. 5). The subject was asked to perform a 50% maximum isometric flexion in this experiment. The average relative error was 4.9%, and the covariance was 10.3% The prediction result for Experiment 2 matches the experimental data nearly as well as in Experiment 1, except there are larger errors at the beginning and the end of the trial.

In Prediction Test 3, we tried to apply the trained neural network to predict muscle activation in Experiment 3 (Fig. 6). Here the subject was asked to perform three cycles of 100% maximum isometric flexion and extension. The average relative error was 34.2%, and the covariance was 32.3% Although the error is larger than that for different trials of Experiment 1 and Experiment 2, the neural network model predicted the joint moment rather closely. The larger error is not unexpected because Experiment 3 is substantially different from Experiment 1, which is used to train the neural network.

The relationship between the EMG signals and muscle activations was examined for the biceps and triceps (Fig. 7). The muscle activation is the predicted result of the neural network model. Other muscles have similar results. The activations predicted from the neural network model have a positive relationship with the EMG signals measured in the experiment. The results match our knowledge that the EMG signal is correlated with muscle activity, i.e., the larger the EMG signal, the more activated the muscle is. The results also match our knowledge that the activation has a nonlinear relationship with EMG signals. It is difficult to express the relation accurately in a simple equation.

Although these results show great success in predicting overall joint moment, they also reveal some inappropriate predictions of specific muscle activations profiles. Specifically, the predicted muscle activations all have a baseline offset. This is about 0.4 for the biceps and 0.3 for the triceps. This is an artifact of the neural network because when the EMG is near zero, the activations should be near zero as well.

V. Discussion

The term *activation dynamics* describes the nonlinear relationship between EMG activity and muscle activation. The neural network model used a particular training algorithm to optimize the connection weights between neurons to map this relationship. The nonlinearity of the relationship was represented by all the connection weights and neuron activation functions. From the training results and prediction results, we can see that the model works well in predicting joint moment from EMG signals.

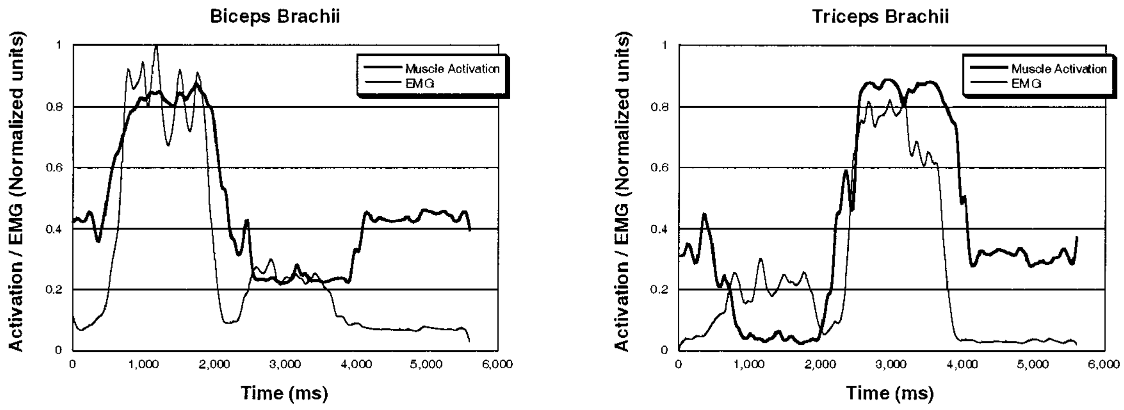

Fig. 7. Comparison of the EMG and predicted activations for (A) the biceps brachii long head and (B) the triceps brachii medial head. The EMG signal is filtered and normalized for comparison.

There have been some very successful studies on the application of neural networks to muscle models, but they typically use the neural network to model the whole process from EMG to muscle force [5], [6], EMG to joint moment [13], or EMG to joint angles [3]. The neural network models are essentially black-box models. Although we can get good predictions of muscle forces or joint moments, we cannot use neural network models to learn about intermediate steps such as how much each muscle has been activated or how much force the muscles have produced when estimating joint moments from EMGs. By including the Hill-type muscle models in the contraction dynamics, the neural network does not have to account for established length–tension or force–velocity affects. On the other hand, we know that the relationship of EMG to muscle force is often nonlinear in somewhat unpredictable ways [14], and this would not be accounted for by simply using normalized EMG as the input to the Hill models. Since this relationship is not well established, a black-box approach for this part of the model is reasonable.

Although it is reasonable to model just the muscle activation dynamics as a neural network, it is also more difficult. To use neural networks to map the relation between EMG signals and muscle activation, we have to train the neural network by giving it the information of both EMGs and muscle activation. EMG signals can be measured in our experiment, but the problem is that we do not have accurate muscle activation values to help to train the model. In this research, an adjusted back-propagation algorithm has been proposed and implemented. In this algorithm, we used the joint moment error to calculate the derivative of joint moment to muscle activation function and used this to change the interconnection weights in the neural network.

A guideline for the minimum amount of data required for training is $10(M + N)$, where M equals the number of inputs and N equals the number of outputs [7]. In this research, $M = N = 10$. Another rule of thumb is that the number of training cases should be ten times the number of model weights. In our neural network model, there are four layers with 10, 15, 15, and 10 neurons, respectively. Thus, we have 525 ($10 \times 15 + 15 \times 15 + 15 \times 10$) neurons. The training experiment was approximately 10 s; the sampling rate was 1000 Hz, yielding 10 000 samples. This is 20 times the number of weights. Thus the training data amount in this research satisfies the above requirements.

The critical issue in developing a neural network is generalization: how well will the network make predictions for cases that are not in the training set? Neural networks, like other flexible nonlinear estimation methods, can suffer from either underfitting or overfitting. A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A network that is too complex may fit the noise, not just the signal, leading to overfitting. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data with many of the common types of neural networks. Overfitting can also produce wild predictions in multilayer perceptrons, even with noise-free data.

The best way to avoid overfitting is to use a large amount of training data. If there are at least 30 times as many training cases as there are weights in the network, overfitting is less unlikely, although some slight overfitting may occur no matter how large the training set is. For noise-free data, five times as many training cases as weights may be sufficient [7].

As calculated before, in this paper the training data amount was around 20 times that of the number of interconnection weights. This should be enough to avoid significant overfitting, but it is not enough to say that there is no overfitting in this system, which is another explanation for the larger error in the predictions associated with Experiment 3.

The model was developed using data for each subject independently; therefore, a neural network constructed from one subject should not be used for others. For similar reasons, a neural network constructed from one group of muscles and joints should not be used for others. In fact, a neural network constructed from a group of muscles could not be reused the next day for the same muscles on the same subject without recalculating the model, because the EMG signal could change with renewed electrode placement. That means that no general artificial neural network can be obtained. However, we can use the current connection weights as initial values to train new neural networks in order to expedite the training process.

From the prediction results, we can see that the prediction error associated with Experiment 3 was larger than the error for Experiments 1 and 2. The reason for this lies in the difference between Experiment 3 and the training experiment. The neural network does not have information about the difference in the protocols. Another reason is due to the overfitting of the neural network model. Thus the approach is best used when the data from which the predictions are to be made are fairly similar to that on which the model was trained. This limitation can be diminished by combining several different trials of different experiments when constructing the training set. This will give the neural network more information about different situations, making it more robust.

Acknowledgment

The authors wish to thank Dr. K. Manal for his assistance with the muscle contraction model and X. Shen for his assistance with the musculoskeletal geometry model and other software issues.

References

[1] N. Bernstein, *The Co-Ordination and Regulation of Movements*. New York: Pergamon, 1967.
[2] F. E. Zajac, "Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control," *Crit. Rev. Biomed. Eng.*, vol. 17, pp. 359–411, 1989.
[3] F. Sepulveda, D. M. Wells, and C. L. Vaughan, "A neural network representation of electromyography and joint dynamics in human gait," *J. Biomech.*, vol. 26, pp. 101–109, 1993.
[4] J. J. Luh, G. C. Chang, C. K. Cheng, J. S. Lai, and T. S. Kuo, "Isokinetic elbow joint torques estimation from surface EMG and joint kinematic data: Using an artificial neural network model," *J. Electromyogr. Kinesiol.*, vol. 9, pp. 173–183, 1999.
[5] H. H. Savelberg and W. Herzog, "Prediction of dynamic tendon forces from electromyographic signals: An artificial neural network approach," *J. Neurosci. Meth.*, vol. 78, pp. 65–74, 1997.
[6] M. M. Liu, W. Herzog, and H. H. Savelberg, "Dynamic muscle force predictions from EMG: An artificial neural network approach," *J. Electromyogr. Kinesiol.*, vol. 9, pp. 391–400, 1999.
[7] S. Haykin, *Neural Networks, A Comprehensive Foundation*. New York: Macmillan, 1994.
[8] D. G. Lloyd and T. S. Buchanan, "A model of load sharing between muscles and soft tissues at the human knee during static tasks," *J. Biomech. Eng.*, vol. 118, pp. 367–376, 1996.
[9] K. Manal, R. V. Gonzalez, D. G. Lloyd, and T. S. Buchanan, "A real-time EMG driven virtual arm," *Comp. Biol. Med.*, vol. 32, pp. 25–36, 2002.
[10] W. M. Murray, S. L. Delp, and T. S. Buchanan, "Variation of muscle moment arms with elbow and forearm position," *J. Biomech.*, vol. 28, pp. 513–525, 1995.
[11] W. M. Murray, T. S. Buchanan, and S. L. Delp, "The isometric functional capacity of muscles that cross the elbow," *J. Biomech.*, vol. 33, pp. 943–952, 2000.
[12] A. O. Perotto, *Anatomical Guide for the Electromyographer*, 3rd ed. Springfield, IL: Charles C. Thomas, 1994.
[13] A. T. Au and R. F. Kirsch, "EMG-based prediction of shoulder and elbow kinematics in able-bodied and spinal cord injured individuals," *IEEE Trans. Rehab. Eng.*, vol. 8, pp. 471–480, 2000.
[14] J. J. Woods and B. Bigland-Ritchie, "Linear and nonlinear surface EMG/force relationships in human muscles. An anatomical/functional argument for the existence of both," *Amer. J. Phys. Med.*, vol. 62, pp. 287–299, 1983.

Lin Wang was born in Harbin, China. She received the B.E. degree in automation and the M.S. degree in automatic control from Beijing Polytechnic University, Beijing, China, in 1995 and 1998, respectively. She received the M.S. degree in mechanical engineering from the University of Delaware, Newark, in 2001.

Her technical interests include muscle activation estimates, muscle modeling, neural networks, and artificial intelligence.

Thomas S. Buchanan (M'87) studied at the University of California at San Diego, Northwestern University, and the Massachusetts Institute of Technology.

He is a Professor of mechanical engineering at the University of Delaware, where he is also Director of the Center for Biomedical Engineering Research and Academic Director of the Biomechanics and Movement Science Program. His research interests are in neural control of human movement and biomechanical models of the musculoskeletal system.