
Neural Network for prediction of the muscle forces in the Human Hand and Forearm system

Hadi El Daou*, **Bilal El Hajjar***, **Rafic Younès***, **Fethi Ben Oueddou****

* *Université Libanaise, Faculté de Génie,
Equipe de recherche 3M, Beyrouth, Liban;*
hadieldaou@hotmail.com
bilal1487@hotmail.com
ryounes@ul.edu.lb

** *Université de Versailles Saint Quentin,
Laboratoire LIRIS, Versailles, France;*
ouezdou@liris.uvsq.fr

Abstract — Our goal is to predict the muscle forces in the Human Hand and Forearm system with Neural network during a given movement. In this paper we're going to study 3 movements of the system HHF, the vector forces used to train the networks is calculated using non linear optimization methods, our first step was to do the batch training using the first movement, the second step is to adapt network with the second movement and test it with the third movement. Two neural networks are proposed: The first one is to approximate the vector force in the muscles; the second one is to classify the forces in two categories: zero forces or no zero forces.

Résumé — L'objectif principal de ce papier est de pouvoir estimer, par les réseaux de neurones, les forces musculaires lors d'une activité libre quelconque du système humain bras/main. Nous avons étudié 3 mouvements pour ce système. Le vecteur force nécessaire pour l'entraînement du réseau est issu des méthodes d'optimisation non linéaire. Le premier mouvement est utilisé pour un apprentissage par block du réseau neuronal, le deuxième mouvement sert pour effectuer un apprentissage adaptatif et le troisième valide le réseau obtenu. Nous avons pu proposer deux réseaux de neurones: Un premier pour estimer les forces dans les muscles et un second permettant de les classer en deux catégories : forces nulles ou non nulles.

1. Introduction

Our goal is to predict the muscle forces in the Human Hand and Forearm system with Neural network during a given movement. Many studies were done in this subject, for instance, the one presented by E.Y. Chao [Chao, 1978] predicts the forces in the normal and abnormal hand during isometric hand function, mainly during the pinch position. Another, presented by V. M. Zatsiorsky [Zatsiorsky & al, 2002], studies the force and torques production in static multifinger prehension. E.Y. Chao [Chao, 1976] developed a three-dimensional force analysis of finger joints in selected isometric hand functions for the pinch task. J. H. Challis [Challis, 1997] examines four functions objectives by static optimization procedures to estimate muscles forces. Nevertheless, all these studies were done in static mode where no movement of the joints is produced. In this paper the dynamic movement of the HHF system is taken into account. The computation of the forces is made, on line, during the movement of the hand.

The system dynamic model, as well as the different parts that form the dynamic equation will be presented. After that the forward model and the inverse model are formulated. The different steps to construct the neural networks are presented. We use a neural network with 150 neurons in one hidden layer to approximate the force vector. Therefore, we use the neural network to know the cases where the forces are active. The results obtained are validated by comparing three elementary movements of the hand. Finally, further developments of the model are discussed.

2. Dynamic equation

The general form of the dynamic equation of the HHF system can be stated as follows [3]:

$$[A(q)] * \left[\frac{d^2 q}{dt^2} \right] + [B(q)] * \left[\frac{dq}{dt} \right]' * \left[\frac{dq}{dt} \right] + [C(q)] = [T_{it}] * [F_i] + [T_c] * [F_c] + [T_i] * [F_i]$$

On the left side of this equation, A is the generalized mass matrix, B represents the Coriolis and centrifugal terms and C concerns the gravity effect. These three matrices depend on the angular position (q) of the joints. The vector (q) represents the generalized coordinates. Hence, this vector determines the configuration of the human hand in three-dimensional space. $\left[\frac{d^2 q}{dt^2} \right]$ and $\left[\frac{dq}{dt} \right]$ indicate the joint acceleration, and velocity.

The right side of this equation represents the interaction torques applied on the system which could be divided into two kinds: those resulting from the muscles of the hand $[F_i]$ (the internal forces) and those produced by the interaction forces $[F_c]$ with the grasped objects. This paper focuses on the calculation of the internal forces produced by the muscles of the hand. These forces are related to the joints torques by the following relation: $[T_{it}] \cdot [F_i] = [\tau]$.

Where $[F_i]$ is the vector of the tendons forces (38 columns), $[\tau]$ is the vector of the joints torques (24 columns) and $[T_{li}]$ is the matrix (20 x 38) which elements are proportional to a distance.

3. Optimization

However the choice of the optimization method is very important. Since the calculation must be made in real-time, the method chosen must converge quickly and steadily toward the optimum solution. The technique chosen is the ‘‘Penalized Lagrangian Multiplier method subject to equality and inequality constraints’’ [Chalfoun & al, 2003]. This technique can be stated as follows [Minoux, 1983]:

$$\left\{ \begin{array}{l} \text{minimize:} \\ L(x, \lambda) = f(x) + \sum_{i=1}^m \lambda_i h_i(x) + k \sum_{i=1}^p P_i[g_i(x)] \\ \text{according to the variables } x \text{ and } \lambda \end{array} \right.$$

Where: $x \equiv F$ the muscles forces, $f(x)$ is the objective function, λ is the Lagrangian multipliers, m is the number of equality equations, p is the number of inequality equations, $h(x)$ is the equality constraints $h(x) = [T_{li}] \cdot [F] - [\tau]$, $g(x)$ is the inequality constraints $0 \leq F \leq F_{\max}$, k is a scalar that increases with each iteration, $k = 3^l$, l number of iteration, $P_i(x) = \text{Max}[0, g_i(x)]^2$ is the penalty term, if $g_i(x) > 0$ then $P_i(x) \neq 0$

This method converts a constrained optimization problem to an unconstrained one. It is chosen for its robustness and its fast convergence. Several cost functions can be chosen to find the solution. However, the cost function must be as simple as possible for quick convergence purpose. In this paper the cost function chosen will include only the muscle forces.

The objective function to minimize is the global effort of the forces [Chalfoun & al, 2003]:

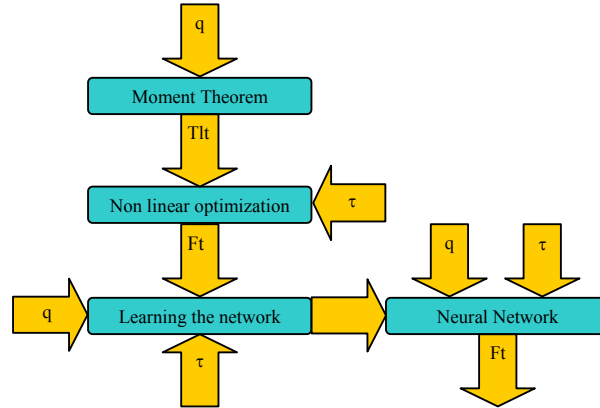
$$f = \sum_{i=1}^n \left(\frac{F_i}{(F_{\max})_i} \right)^2$$

Where $n = 38$ is the number of the muscles in the HHF.

The optimization method is complicated and non realistic, for that and to do analogy with brain behavior we proposed to use neural network. In this paper we

are going to construct two neural networks: The first one is to approximate the vector force in the muscles. The second one is to classify the forces in two categories: zero forces and no zero forces.

The following chart shows the different steps to construct the neural networks.



The brain must command the muscle forces knowing the angular positions and the interaction torques. So both networks have 40 entries (20 entries come from the vector τ and 20 come from vector q , one hidden layer, and 38 outputs which come from vector F_i (figure 1).

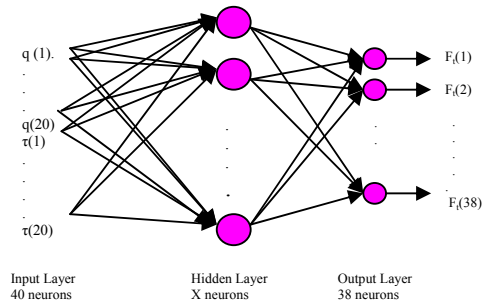


Figure 1: architecture of the neural networks

We have $[q]$, $[\tau]$, and $[F_i]$ of three movements.

The first movement is used for the batch training of the neural network, the second is used to adapt the neural network, and the third is used to simulate the neural network. The three movements are given in the figure 2, 3, and 4.

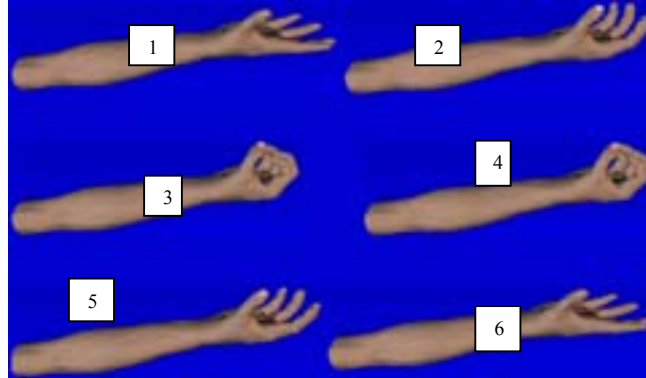


Figure 2: Sequence of movement 1.

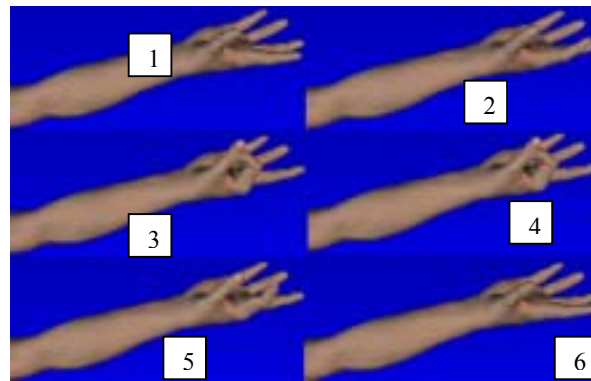


Figure 3: Sequence of movement 2.

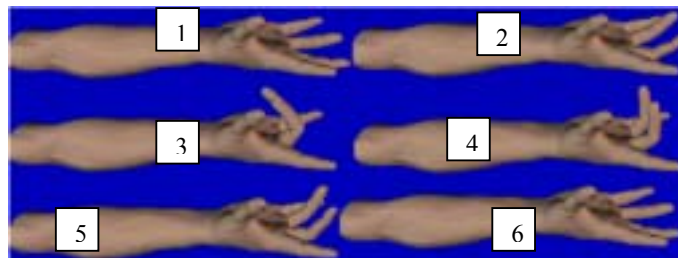


Figure 4: Sequence of movement 3.

Each movement is characterized by 203 vectors $[\mathbf{q}]$, 203 vectors $[\boldsymbol{\tau}]$, and 203 vectors $[\mathbf{F}_i]$.

4. Approximation of the Vector F_t

We use a neural network with one hidden layer; **Figure 1** shows the form of our neural network. We have 40 input neurons corresponding to 20 angular positions q , and the 20 interaction torques applied to the 20 joints. We have x neurons in the hidden layer, and 38 neurons in the output layer corresponding to the 38 muscle forces in the HHF system.

4.1 The transfer function

a) The transfer function of the input layer is "tansig" (**figure 5**), because a multilayer perceptron trained with the back-propagation algorithm may, in general learn faster (in terms of the number of training iterations required) when the sigmoid activation function built into the neuron model of the network is asymmetrical than when it is nonsymmetrical [Dreyfus & al, 2002].

b) The transfer function of the hidden layer is "purelin" (**figure 5**), because we need real outputs.



Figure 5: Transfer function.

One of the problems that occur during neural network training is called over fitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.

This method is used to improve generalization. This involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. The next subsection explains how the performance function can be modified.

The typical performance function that is used for training feed forward neural networks is the mean sum of squares of the network errors.

$$F = mse = \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2$$

It is possible to improve generalization if we modify the performance function by adding a term that consists of the mean of the sum of squares of the network weights and biases [Dreyfus & al, 2002] : $msereg = \gamma.mse + (1 - \gamma).msw$

where γ is the performance ratio, and $msw = \frac{1}{n} \sum_{j=1}^n w_j^2$

Using this performance function will cause the network to have smaller weights and biases, and this will force the network response to be smoother and less likely to overfit. The problem with regularization is that it is difficult to determine the optimum value for the performance ratio parameter. If we make this parameter too large, we may get over fitting. If the ratio is too small, the network will not adequately fit the training data.

4.2 Back propagation algorithm

We use the conjugate gradient algorithm “*scg*” for the batch training, for two reasons: The large number of weights in our problem and this algorithm is fast on function approximation problems.

The “*scg*” seem to perform well over a wide variety of problems, particularly for networks with a large number of weights. The “*scg*” algorithm is almost as fast as the LM (Levenberg-Marquardt) algorithm on function approximation problems (faster for large networks) and is almost as fast as resilient back propagation algorithm “*rp*” on pattern recognition problems. The conjugate gradient algorithms have relatively modest memory requirements; this algorithm is supported in matlab in the toolbox “*ntool*” by the function “*trainscg*”.

We haven’t any rule for choosing the number of neurons in the hidden layer, we will begin with a number of neurons equal the number of neurons in the input layer and then we test several numbers of neurons to choose the optimal number of neurons using the following criterion: The number of neurons in the hidden layer yielding to the best Percentage of error <1%, 5% and 10%. The results are given in the figures 6, 7, and 8.

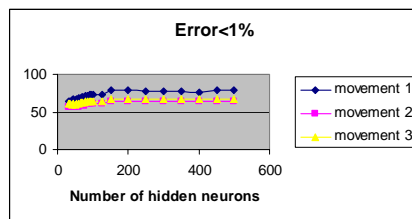


Figure 7: Percent of muscle forces of error<1%

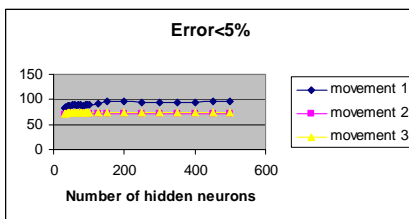


Figure 8: Percent of muscle forces of error<5%

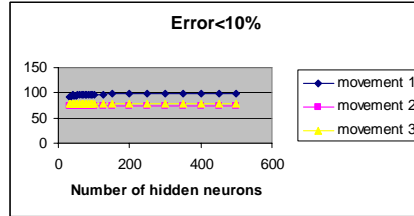
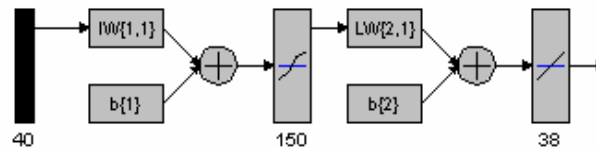


Figure 9: Percent of muscle forces of error<10%

Using the table and the figures we choose 150 neurons because beyond this number the results are very close. The neural network becomes:



Now we will adapt the neural network with the second movement using 1000 passes.

The results become:

Movement 1: Error <1%: 62, Error <5%: 69, Error <10%: 71
 Movement 2: Error <1%: 80, Error <5%: 86, Error <10%: 89.
 Movement 3: Error <1%: 66, Error <5%: 72, Error <10%: 75.

5. Classification as Zeros and No Zeros Forces

In this problem, we will use the neural network to know the cases where the constraint " $T>0$ " is active. If the constraint " $T>0$ " is active, the solution is " $T=0$ ", else the muscle forces will be calculated using the Lagrange method with the equality constraint. In this case the outputs of the neural network must be -1 if the constraint is active and 1 in the other case. Thus we will replace the values of the muscle forces $<1e^{-4}$ by -1 and the other forces by 1.

We have 3 layers:

- a) an input layer having 40 neurons corresponding to the 20 angular positions \mathbf{q} , and the 20 interaction torques τ applied to the 20 joints.
- b) One hidden layer having x neurons.

c) An output layer having 38 neurons corresponding to the 38 muscle forces in the HHF system.

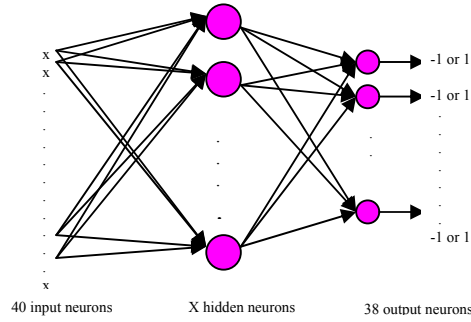


Figure 10: the classification network

The activation function of the hidden layer and the output layer is “*tansig*”. We have used the resilient backpropagation algorithm supported in matlab by the function “*trainrp*”, this algorithm is the fastest on pattern recognition problems, and the memory requirements for this algorithm are relatively small. In this problem we have used the function “*msereg*” described in paragraph 3 as a performance function. The choice of the number of neurons in the hidden layer is done experimentally. The following table contains the different number of neurons used and the number of data wrongly classified.

Nombre of neurons	Movement 1	Movement 2	movement 3
80	113	453	620
90	115	451	618
100	111	451	620
110	109	451	622
120	109	451	626
130	107	451	626
140	103	451	624
150	103	449	626
160	105	449	628
170	101	449	628
180	101	449	626

Table 1: number of neurons used and the number of data wrongly classified

The results described in this figures are extremely near in our problem. We are going to choose 100 neurons in the hidden layer, the general shape of the network become:



Now we are going to adapt the network to movement 2, using 1000 passes after training the network we obtained the following results:

- Movement 1: 280/7714 ~ 3.63 %.
- Movement 2: 130 /7714 ~ 1.69 %.
- Movement 3: 638 /7714 ~ 8.27 %.

We see that the errors due to the movements 1 and 3 rise ,and the error in movement 2 shrinks, this results come from the fact that the performance function “*msereg*” has a big capability of generalization facing the adapt function.

6. Lagrangian Solution under Inequality Constraint

We are going to use neural network to predict the zero forces and the no zero forces.

Knowing the zero forces, the problem become minimizes $T = \sum_{i=1}^x T_i^2$ subjected to: $[T_i] \times [F_i] = [\tau]$. x : number of no zero forces, $[T_i]$ is a $20 \times x$ matrix, F_i is a vector $x \times 1$, $[\tau]$ is a 20×1 vector.

A classical method to dealing with this problem is the method of Lagrange multipliers this procedure begins by formulating the Lagrange function [Minoux, 1983]:

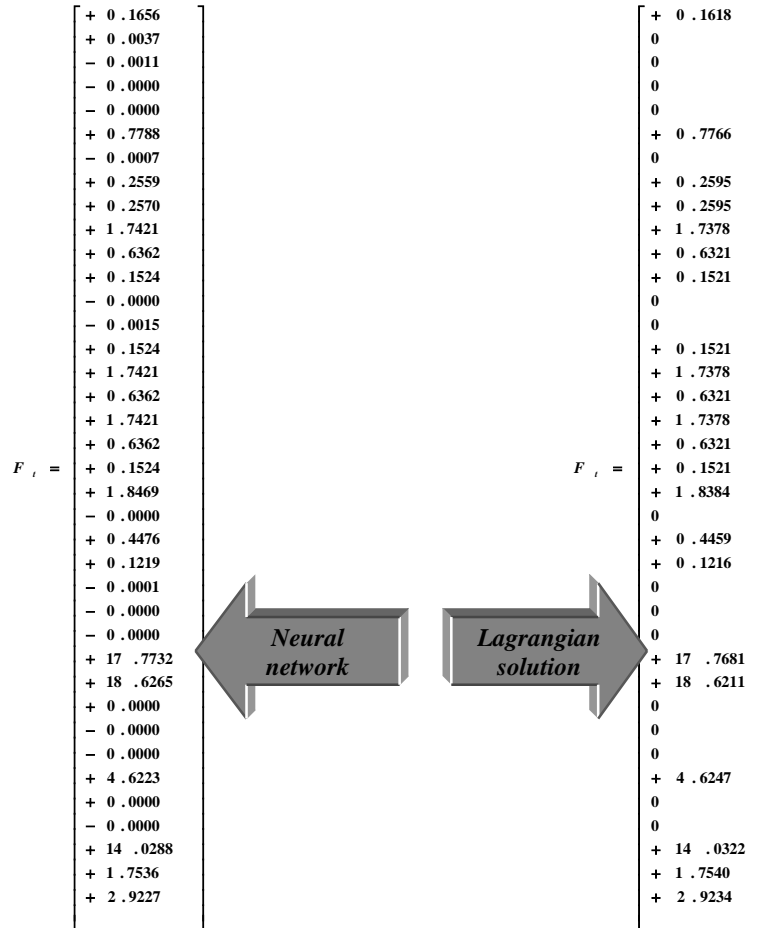
$$L = \sum_{i=1}^x T^2(i) + \sum_{i=1}^{20} \lambda_i (\sum_{j=1}^x T_u(i, j) * T(j) - \tau(i))$$

$T_u(i, j)$ is the element (i,j) of the matrix T_u , $\tau(i)$ is the element I of the vector τ . Where the new variables λ are called Lagrange multipliers

$$\frac{\partial L}{\partial T_j} = 2T_j + \sum_{i=1}^{20} \lambda_i * T_u(i, j) = 0 \quad j = 1, \dots, x$$

$$\frac{\partial L}{\partial \lambda_k} = \sum_{j=1}^x T_{it}(k, j) * T_j - \tau(k) = 0 \quad k = 1, \dots, 20$$

Using the neural network we can predict the zero forces and using Lagrange multipliers we can obtain the nonzero forces. Let's take as an example the first elements of T_{it} , F_t and τ of the first movement.



7. Conclusion

In this paper we have proved that we are capable to construct a neural network that can simulate the vector of the tendons forces F_t , the advantage of the neural

network technique is the possibility of generalization the neural network can memorize the training examples and use it to generalize for new situations.

In other hand we constructed a neural network witch helped us to simplify an optimization problem.

For the first network we used the algorithm SCG (scaled conjugate gradient) to do the batch training of the network, for adaptive training we used the algorithm back propagation with momentum and the function *tansig* for the entries layer and the function $f(x)=x$ in the hidden layer. The determination of the number of neurons in the hidden layer is done experimentally; to avoid over fitting we choused 150 neurons.

For the second network we used the algorithm RP (resilient back propagation) to do the batch training of the network, for adaptive training we used the algorithm back propagation with momentum and the function *tansig* for the entries layer and the hidden layer. The determination of the number of neurons in the hidden layer is done experimentally to avoid over fitting we choused 100 neurons.

Both networks gave very good results for the training and test sets, the feature development is to adapt both networks with new movements until we construct networks with a very good capability of generalization.

REFERENCES

- E. Y. Chao, "Determination of internal forces in human hand". Journal of the engineering mechanics division, 1978. Vol. 104, No. EM1.
- M. V. Zatsiorsky & al, "Force and torque production in static multiphinger prehension: biomechanics and control. I. Biomechanics". Biol. Cybern, 2002.
- J. Chalfoun, M. Renault, R. Younes, F.B. Ouezdou, "Muscle Forces Prediction of the Human Hand and Forearm System in Highly Realistic Simulation". IROS 2004, Japan.
- E.Y. Chao, *Three-dimensional force analysis of finger joints in selected isometric hand functions*. J. Biomechanics, 1976. 9: p. 387-396.
- J. H. Challis, "Producing physiologically realistic individual muscle force estimations by imposing constraints when using optimization techniques". Med. Eng. Phys. Vol. 19. No 3. pp 253-261, 1997
- M. Minoux, "Programmation mathématique". tome 1 & 2, Editions Dunod, 1983
- G. Dreyfus & al, "Réseaux de neurones, Méthodologie et applications". Editions Eyrolles, 2002.