Designing a Muscle like System based on PID Controller and Tuned by Neural Network

Hayssam Serhan, Chaiban G. Nasr, Patrick Henaff

Designing a Muscle like System Based on PID Controller and Tuned by Neural Network

Hayssam J. Serhan, Chaïban G. Nasr (SM IEEE) and Patrick Henaff

Abstract—**This paper presents a study on a Muscle like System based on a PID Controller tuned by a Neural Network. The approach is based on a non linear Muscle Model using system identification based on a NNARX (Neural Network AutoRegressive eXogenous) [3] structure. This model is used in a special configuration of an MLP in order to let the output of the closed loop formed by the motor and controller to follow that of this non linear Muscle Model. Two structures are compared and the robustness of the approach is tested with different models of DC motors.**

I. Introduction

Creating an artificial system that can behave as a real Muscle is an important task that has many applications in two domains:

- The first one is in the field of artificial prostheses for handicap applications.
- The second is in the field of bio inspired robots in which we try to let a robot leg for example to behave as a human or an animal one.

Our aim is to control an existing walking robot that walk and behave like a human or an animal. To do that, one way consists to control implemented DC-Motors and their PID-Controllers to obtain with legs same equivalent behaviors as real muscles. Thus, the muscle-like system is based on a PID controller tuned by an external system that has learned a model of the muscle. Some researchers in the field of muscle modeling created different models like Hill [22] and others used linear or non linear system identification. In the modeling of a muscle, like model based on non linear system identification, many studies have been done. Some are based on an RBF network [1]. This structure is found to be suitable only for muscle with a majority of fast motor units. Big number of RBF neurons function was noted. Others are based on Local Model Network [2] which is proved to be suitable for a wider range of muscles than the previous one, but with a more complicated system design. In our approach, the identification was done based on an NNARX structure. Good performances of ARX method in non linear system identification and the properties of learning of Neural Networks have contributed to a simplified network structure with better identification results. Then we have implemented the learned neural network that can change on-line the parameters of the PID-Controller.

This paper is organized as follows: In section II, a non linear identification of a muscle model based on NNARX structure is presented. In section III, a special Neural Network architecture is discussed and implemented for controlling the parameters K_c, T_i *and* T_d of the PID Controller. Here two configurations are being discussed. In section IV, comparison results for the two configurations, presented in the previous section, are discussed. The paper concludes with a perspective for future works.

II. Muscle Model

To represent the muscle, we propose to use a system identification based on NNARX structure [3]. NNARX stands for Neural Network ARX in which the internal engine is a Neural Network. NNARX model are similar to their ARX linear counterpart. This predictor is always stable (even if the system is unstable) because there is a pure algebraic relationship between prediction, past measurements and inputs. This is a very important feature of the NNARX structure in the nonlinear case since the stability issue is more complex than for linear systems. In [3], Norgaard shows that this is a very good method for deterministic or noiseless system. The principle of NNARX method consists to use past measurements and past inputs as inputs to the NNARX network, while considering the internal architecture as a feed forward MLP network.
The regression vector is represented as follows:

$$\varphi(t) = \left[y(t-1) \ldots y(t-n_a)\, u(t-n_k) \ldots u(t-n_b-n_k+1)\right]^T \quad (1)$$

y(t) is the output of the Muscle Model, u(t) is the neuromuscular command (input), n_a indicates the number of past measurements used (here n_a = 3), n_k is the time delay (here n_k=1), n_b is the number of past inputs used (here n_b=3). The predictor vector is represented as follows:

$$\hat{y}(t|\theta) = \hat{y}(t|t-1,\theta) = g(\varphi(t),\theta) \quad (2)$$

θ is the vector of adjustable parameters.

The training algorithm used is the Levenberg-Marquardt method [4], this algorithm is the most widely used optimization algorithm. It outperforms simple gradient descent and other conjugate gradient methods. The problem

This work was supported in part by the Lebanese University - Faculty of Engineering I and LIRIS Laboratory - Versailles - Saint Quentin University.

Hayssam J. Serhan is with the Lebanese University - Faculty of Engineering I - Lebanon (phone: 961-3-317992; Fax: 961-5-452656; email: hserhan@hotmail.com)

Chaïban G. Nasr is with the Lebanese University - Faculty of Engineering I - Lebanon (phone: 961-3-369245; Fax: 961-6-385087; email: chnasr@ieee.org)

Patrick Henaff is with LIRIS Laboratory - Versailles - Saint Quentin University – France (phone: 01.39.25.49.91 ; email: patrick.henaff@liris.uvsq.fr).

for which the LM algorithm provides a solution is called "Nonlinear Least Squares Minimization". The Levenberg-Marquardt algorithm dynamically mixes Gauss-Newton and Gradient-Descent iterations and thus the update rule is the following: $x_{i+1} = x_i - (H + \lambda diag[H])^{-1} \nabla f(x_i)$

H Being the Hessian matrix and $\nabla f(x_i)$ is the gradient of the function to be minimized both evaluated at x_i.

The system identification used in this paper is based on the toolbox created by Magnus Norgaard, which is a nonlinear system identification toolbox developed under Matlab.

Figure 1 presents the NNARX structure for our problem of muscle identification. In this scheme, y(t-nk) are data issued from experimentation described by Gollee and Donaldson et al. in [1] and [2]. More details of our approach used in the creation of this muscle model and its comparison with other models will be presented in another next paper.

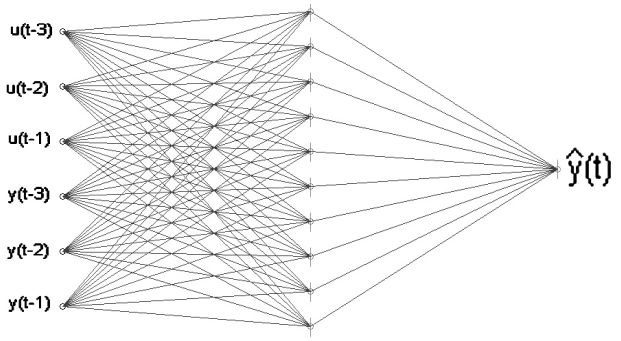

Fig. 1. NNARX Muscle Model

The results obtained with this method are presented in figure 2. We can see that the error is of 10^{-4} order and thus the superposition of the real Muscle output and the NNARX model is acceptable. A good amelioration compared with the identification based on RBF network developed by Donaldson et al. [1] and with the LMN Muscle Model created by H. Gollee et al. [2] was obtained.

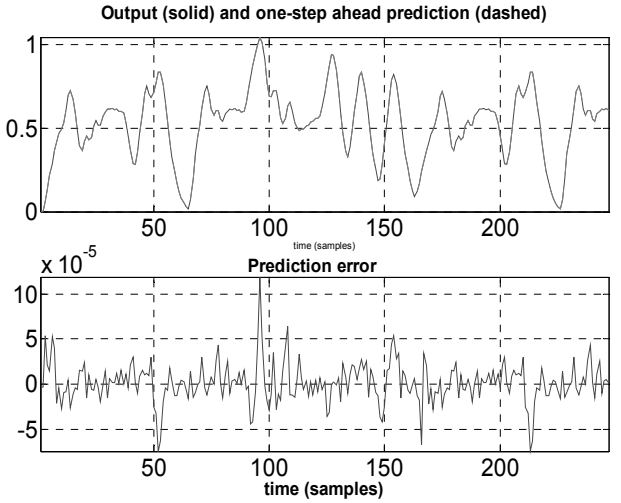

Fig. 2. Results of superposition between the real signal and the NNARX Model (the superposition is perfect so we cannot distinguish between dashed and solid lines).

III. Physical Emulation Of A Muscle

The idea discussed in this section, is to keep the output of the DC Motor, combined with the PID Controller in closed loop, behave like a real muscle by simple variation of the parameters K_c, T_i and T_d of the PID Controller. So a comparison with the Muscle Model Output is necessary. In order to tune the parameters of the PID Controller to let the output y(t) of the DC Motor follows that of the Muscle Model $y_d(t)$, we use an Artificial Neural Network (MLP) in two different structures:

1-In the first one (see Fig. 3), the Muscle Model is connected in parallel with the system [DC-Motor and PID controller]. In this structure, I(t) is the ANN Input vector defined by :

$$I(t) = [r(t), r(t-1), r(t-2), y_d(t), y_d(t-1), y_d(t-2), e(t), e(t-1), e(t-2), \rho(t)] \quad (3)$$

y(t) is the torque output of the DC motor, $y_d(t)$ is the output of the Muscle Model, r(t) is the neuromuscular command .

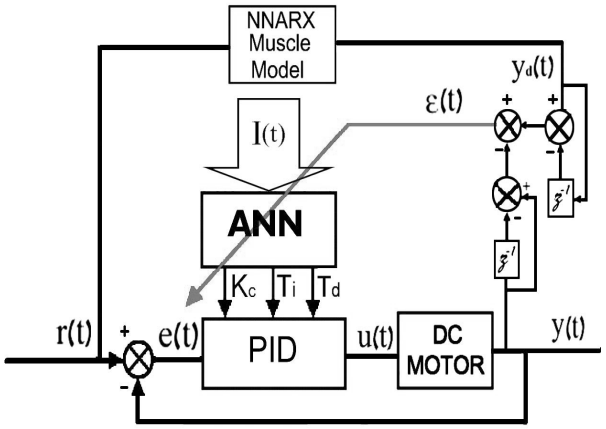

Fig. 3. Block diagram of the parallel configuration.

As mentioned in [5] by S. Takagi et al., $\rho(t)$ is a signal which gives the neural network information about the direction when the command signal $r(t)$ rises up or falls down. These authors have demonstrated that it will be useful to use this signal $\rho(t)$ as input to the neural network. They propose the next expression for $\rho(t)$:

$$\rho(t) = \begin{cases} \eta & \Delta r(t) = r(t) - r(t - T_s) > 0 \\ \rho(t-1) & \Delta r(t) = r(t) - r(t - T_s) = 0 \\ -\eta & \Delta r(t) = r(t) - r(t - T_s) < 0 \end{cases} \quad (4)$$

2-In the second structure (see Fig.4), the output of the Muscle Model is connected like a reference to the system (DC – Motor and PID Controller).

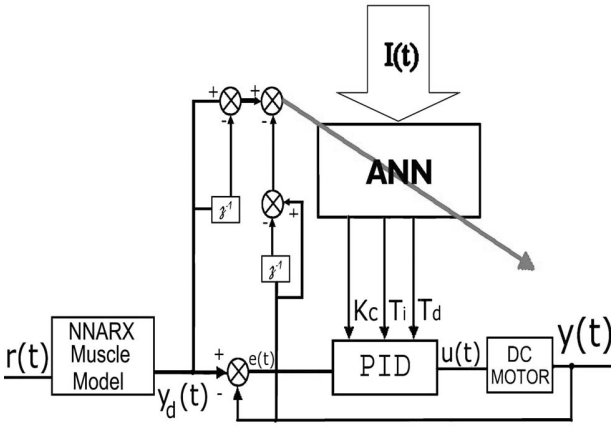

Fig. 4. Block diagram of the serial configuration

In this structure, I(t) is ANN vector Input defined by :

$$I(t) = \left[e(t), \int e(t), \frac{de(t)}{dt}, \rho(t) \right] \quad (5)$$

The Architecture of the ANN used is the two structures is represented as follows in figure 5:

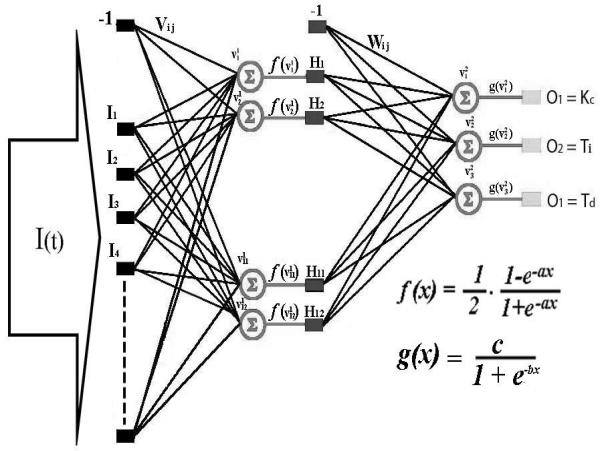

Fig. 5. Artificial Neural network tuner

The parameters of this ANN tuner have been determined after few experiments. In this way, the parameters of the two nonlinear functions corresponding to the internal neurons layer depend of the structure:

- For the first one, we have a=b=1 and c=0.3.
- For the second one, we have a=b=1 and c=4.

The number of neurons in the hidden layer depends also of the structure.

- For the first one, we have NH=60.
- For the second one, we have NH=12.

The number and types of input variables depends also of the structure.

- For the first one, input I(t) is given by equation (3)
- For the second one, input I(t) is given by (5)

Which gave the same result as for:

$$I(t) = [e(t), e(t-1), e(t-2), \rho(t)]$$

The value of the sampling time period T_s used for calculating the outputs and for updating the weights of the network is set at 10 ms for the first structure and 100 ms for the second structure.

In order to update the weights of the network, we have to design a special back-propagation algorithm. In fact, the usual algorithms for back-propagation are based on direct calculations in which we compare each output with its desired value and we calculate the corresponding error etc... Here the training of the network must be implemented in an indirect manner because we don't know the desired value of the parameters Kc, T_i and, T_d. However, the output of the DC-Motor y(t) must follows that of the Muscle Model $y_d(t)$. Then, it is necessary to develop an indirect training algorithm that is presented in next section.

III.1. Indirect Learning with Back–Propagation Algorithm

The basic Back–Propagation Algorithm is based on the minimization of the classical output quadratic error:

$$\varepsilon^2(t) = (y_d(t) - y(t))^2 \quad (6)$$

But, referencing [5], S. Takagi et al. propose to use the following expression for training:

$$\varepsilon^2(t) = (\Delta y_d(t) - \Delta y(t))^2$$

This is equivalent to the expression:

$$\varepsilon^2(t) = [(y_d(t) - y_d(t-Ts)) - (y(t) - y(t-Ts))]^2$$

This will result in a quicker learning rate in the field of fast variations of the input signal and slower learning rate in the field of slow variations. Therefore, this form is very convenient for our study because the muscle present many fast variation zones.

Referencing to figure *5*, we consider the following parameters:

- V_{ij} and W_{jk} are the weights of the two successive Neural Network layers.
- H_j , is the output of the hidden layer :

$$H_j = f\left(\sum_{i=0}^{p} V_{ij}.I_i \right) \quad (7)$$

- O_k is the output of the output layer :

$$O_k = g\left(\sum_{j=0}^{q} W_{jk}.H_j \right) \quad (8)$$

III.1.1. Calculation for Output Layer

We have to recalculate the entire back-propagation algorithm for both the output and the hidden layer of the neural network. The idea is to update the value of the neural network weights in order to minimize the global error:

$$\xi(t) = \frac{1}{2}\varepsilon^2(t) \quad (9)$$

This minimization leads to have the partial derivative $\frac{\partial \xi(t)}{\partial W_{jk}}$ equal to zero, so this partial derivative is decomposed in the following form:

$$\frac{\partial \xi(t)}{\partial W_{jk}} = \frac{\partial \xi}{\partial \varepsilon}.\frac{\partial \varepsilon}{\partial \Delta y}.\frac{\partial \Delta y}{\partial \Delta u}.\frac{\partial \Delta u}{\partial O_k}.\frac{\partial O_k}{\partial W_{jk}} \quad (10)$$

Some terms are easily calculated:

$$\frac{\partial \xi}{\partial \varepsilon} = \varepsilon(t) \quad \text{and} \quad \frac{\partial \varepsilon}{\partial \Delta y} = -1 \quad (11\text{-}12)$$

Due to the PID control, we know that:

$$\Delta u(t) = K_c\left(\Delta + \frac{T_s}{T_i} + \Delta^2 \frac{T_d}{T_s}\right)e(t) \quad (13)$$

where: $\Delta = 1 - z^{-1}$, then :

$$\frac{\partial \Delta u}{\partial O_k} = \begin{cases} \left(\Delta + \frac{T_s}{T_i} + \frac{T_d}{T_s}\Delta^2\right)e(t) \rightarrow O_k = K_c \ \ for\ k=1 \\ -\frac{K_c T_s}{T_i^2}e(t) \rightarrow O_k = T_i \ \ for\ k=2 \\ \frac{K_c}{T_s}\Delta^2 e(t) \rightarrow O_k = T_d \ \ for\ k=3 \end{cases} \quad (14)$$

The partial derivative of the output of the neural network in correspondence to the weights is simply the derivative of the non-linear function of the output layer, so:

$$\frac{\partial O_k}{\partial W_{jk}} = \frac{b}{c}O_k(c - O_k)Hj \quad (15)$$

The general transfer function of the motor is as follows:

$$H(p) = \frac{Y(p)}{U(p)} = \frac{A}{(1 + \tau p)(1 + \tau' p)} \quad (16)$$

Then we can write in the discrete time:

$$Au(t) = y(t) + \frac{\tau + \tau'}{Ts}\left(y(t) - y(t - Ts)\right) + \frac{\tau \tau'}{T^2 s}\left[y(t) - 2y(t - Ts) + y(t - 2Ts)\right]$$

and,

$$Au(t - Ts) = y(t - Ts) + \frac{\tau + \tau'}{Ts}\left(y(t - Ts) - y(t - 2Ts)\right) + \frac{\tau \tau'}{T^2 s}\left[y(t - Ts) - 2y(t - 2Ts) + y(t - 3Ts)\right]$$

So, we have:

$$A[u(t) - u(t - T_s)] = \Delta y(t) + \frac{\tau + \tau'}{Ts}\left(\Delta y(t) - \Delta y(t - Ts)\right) + \frac{\tau \tau'}{T^2 s}\left[\Delta y(t) - 2\Delta y(t - Ts) + \Delta y(t - 2Ts)\right]$$

Finally, we obtain:

$$\Rightarrow \frac{\partial \Delta y(t)}{\partial \Delta u(t)} = \frac{A}{1 + 2\left(\frac{\tau + \tau'}{Ts}\right)\Delta y(t) + \frac{3\tau \tau'}{T^2 s}\Delta^2 y(t)} \quad (17)$$

This expression will contribute to the following updating rule for the weights of the output layer W_{jk} as:

$$\Delta W_{jk}(t) = -\alpha \frac{\partial \xi(t)}{\partial W_{jk}} + \mu \Delta W_{jk}(t-1) \quad (18)$$

III.1.2. Calculation for the Hidden Layer

The same calculations and derivations could be done for updating the weights of the hidden layer:

$$\frac{\partial \xi(t)}{\partial V_{ij}} = \sum_{k=1}^{3} \frac{\partial \xi}{\partial \varepsilon}.\frac{\partial \varepsilon}{\partial \Delta u}.\frac{\partial \Delta u}{\partial O_k}.\frac{\partial O_k}{\partial H_j}.\frac{\partial H_j}{\partial V_{ij}} \quad (19)$$

We have: $\frac{\partial \xi}{\partial \varepsilon} = \varepsilon(t)$ and $\frac{\partial \varepsilon}{\partial \Delta u} = -1$ (20-21)

And $\frac{\partial \Delta u}{\partial O_k}$ is given by equation (14)

Also we have: $\frac{\partial O_k}{\partial H_j} = \frac{b}{c}O_k(c - O_k)W_{jk}$ (22)

and $\frac{\partial H_j}{\partial V_{ij}} = a\left(\frac{1}{2} + H_j\right)\left(\frac{1}{2} - H_j\right)$ (23)

Finally, we can compute the expression of the updating rule for the hidden layer V_{ii} as:

$$\Delta V_{ij}(t) = -\alpha \frac{\partial \xi(t)}{\partial V_{ij}} + \mu \Delta V_{ij}(t-1) \quad (24)$$

IV. Implementation And Results

We have implemented the two models under Simulink in order to test the performance of each of the two structures: the parallel one (Fig 3) and serial one (Fig 4). Each performance is based on the calculation of the mean square error given by:

$$E = \frac{1}{N}\sum_{n=0}^{N}(y(n) - y_d(n))^2 \quad (25)$$

IV.1. Tuning the PID with a Parallel structure

In the parallel structure, we have connected the Muscle Model in parallel with the physical system (DC Motor – PID Controller) as we can see in the Simulink scheme (Fig. 6).

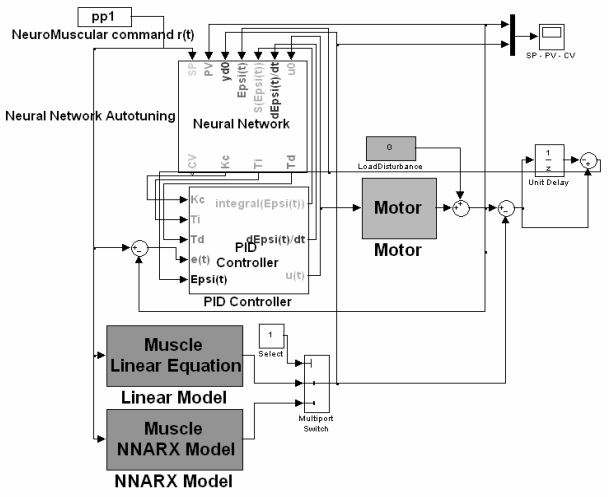

Fig. 6. Simulink Block scheme for the parallel structure

The PID controller is designed to have its parameters K_c, T_i *and* T_d as inputs, also having the integral of the error e(t) and the derivative of the error as outputs.
The Neural Network is designed as an S-Function. The Motor block is a transfer function of the designated DC Motor, and the input command r(t) is an especially designed signal under Matlab in order to emulate as much as possible the neuromuscular command.

In order to tune the parameters and to choose the appropriate inputs to the neural network and the necessary number of neurons in the hidden layer, we have made many combinations with these parameters. Four significant results are presented in next sub-sections.

IV.1.1 First Result

In this first simulation we set the parameters as follows:

- Ts = 20ms for the sampling period.
- 30 neurons in the hidden layer.
- r(t), y(t), $y_d(t)$, ρ(t) as inputs.

Results are presented on Fig 7.

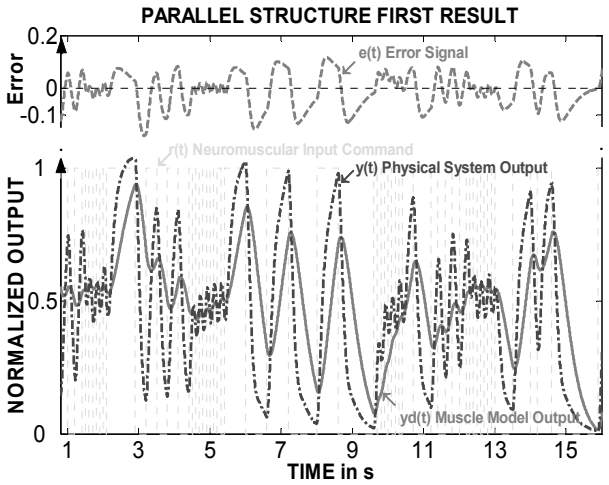

Fig. 7. First results obtained (Muscle Model output in Magenta $y_d(t)$ and physical system output in Cyan $y(t)$, in blue is r(t))

It is clear that we have obtained some kind of superposition but it is not so perfect. The error E is remarkable in the two zones of fast and slow variations of the Muscle Model output. Calculation of E gives:

- E = 20% in the field of Slow variation.
- E =29% in the field of Fast variation.

IV.1.2 Effect of size of hidden layer and of sampling period time

From the first result, we have increased the number of neurons in the hidden layer up to 60 and decreased the sampling period down to 10ms. Some amelioration in the superposition was noted. But in another way, more increasing in the number of neurons in the hidden layer for more than 60 and/or decreasing the sampling period time for less than 10ms haven't contributed to any enhancements in the results.

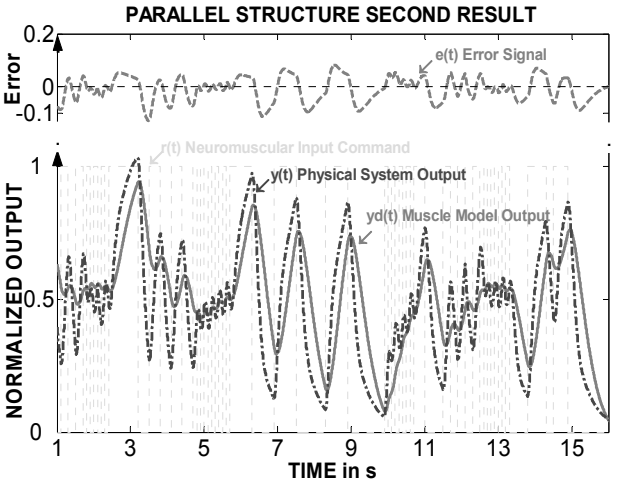

Fig. 8. Second result obtained.

The error in the two zones of fast and slow variations of the Muscle Model output is:

- E =10% in the field of Slow variation.
- E =33% in the field of Fast variation.

IV.1.3 Effect of adding delayed inputs

In this simulation, we change the nature or the inputs of the neural network in order to give more information about past inputs, past outputs, errors and past errors. So, to consider only the first order of past input, past output and past error, the new input vector is:

$$I(t) = [r(t), r(t-1), y_d(t), y_d(t-1), e(t), e(t-1), \rho(t)]$$

And we set:

- Ts = 10ms for the sampling period.
- 60 neurons in the hidden layer.

The results obtained are presented below (Fig.9):

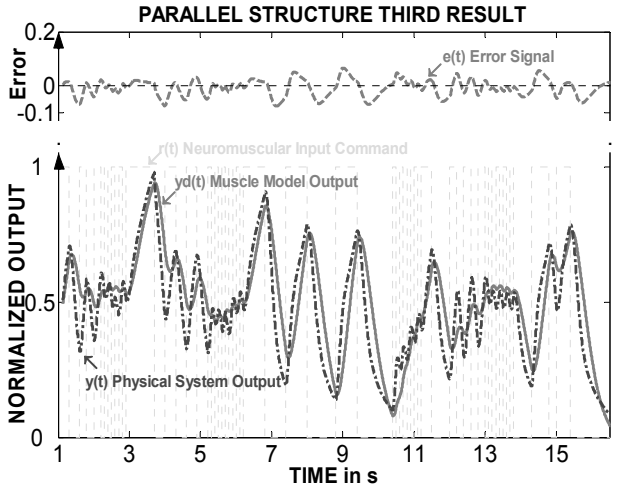

Fig. 9. Third result obtained

Ameliorations are clear in the two zones of fast and slow variations of the Muscle Model output. Calculation of error gives:

- E = 8% in the field of Slow variation.
- E = 14% in the field of Fast variation.

IV.1.4 Increasing the size of hidden layer with delayed inputs

In the fourth experiment, we increase the neurons in the hidden layer up to 80.The input vector I(t) is given by (3). Results are presented below:

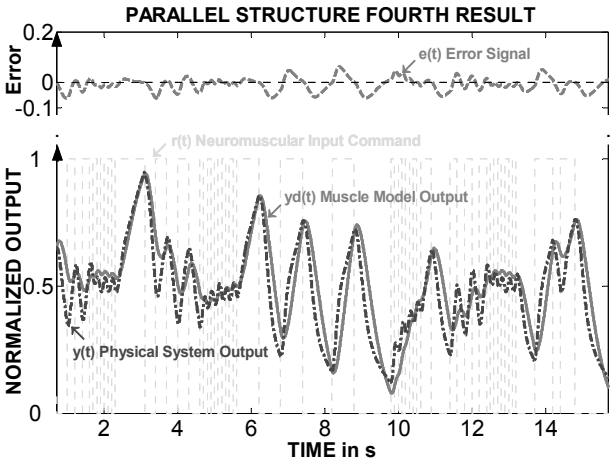

Fig. 10. Fourth result obtained(Ts=10ms)

Ameliorations are clear in the two zones of fast and slow variations of the Muscle Model output. The error is:

- E =6% in the field of Slow variation.
- E =10% in the field of Fast variation.

It's clear that this combination of parameters gives the best superposition between the Muscle Model output and the physical system output (DC motor – PID controller). This is normal because finally, the input gives more information on the dynamic of the muscle, the sample frequency is higher, and the number of hidden neurons allows the network to better interpolate. For this set of parameters, we plot on Fig 11 the mean square error during the learning phase. The curve shows clearly the convergence of the network.

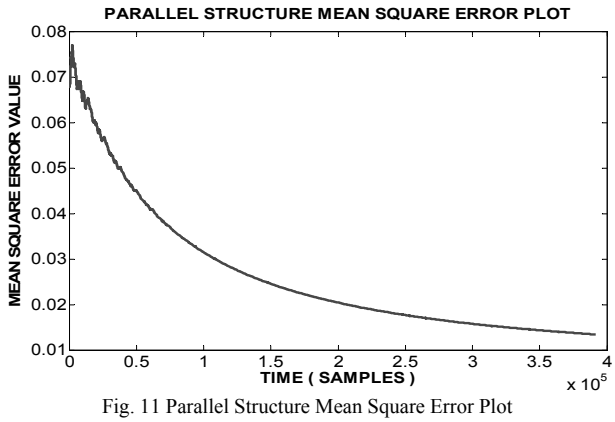

Fig. 11 Parallel Structure Mean Square Error Plot

IV.2 Tuning the PID with a Serial structure

The second structure (Fig 11) consists in connecting in serial the output of the Muscle Model as a reference to the closed loop of the physical system (DC Motor – PID controller).

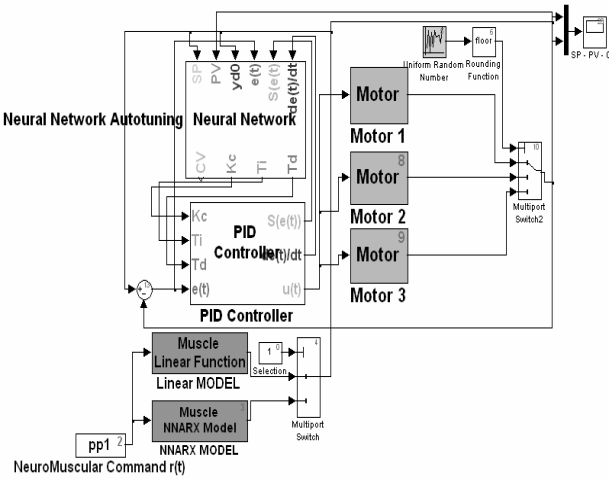

Fig. 12. Simulink block diagram of the second configuration

In this scheme, the blocks are the same as the first one of the Fig 6. The only modification made here is that we have implemented three transfer functions for the DC Motor, In order to evaluate the robustness of our controller… So, we have implemented those transfer functions corresponding to different operating conditions in order to see how the system will behave when the DC Motor or its parameters change. The nominal plant is given below (taken from Ching-Hung Lee et al. [6]):

$$\frac{15}{0.005s^2 + 0.1s + 1.305}$$

Plant with loading variant:

$$\frac{18}{0.0032s^2 + 0.072s + 1.28}$$

The third function is an identification of the DC-Motor:

$$\frac{18}{0.0031s^2 + 0.077s + 1.25}$$

To do that, we have implemented a random function which selects one of the three transfer function randomly and continuously during each simulation and a test of robustness is being given in *IV.2.3*. Like for the parallel structure, we test the serial one with different parameters. Significant results are presented next.

IV.2.1 First Result

An example of result of superposition between the output of the Muscle Model and the physical system was with the following values:

- Ts = 100ms
- 10 neurons in the hidden layer
- $r(t), y(t), y_d(t), \rho(t)$ as inputs

The results obtained are the following:

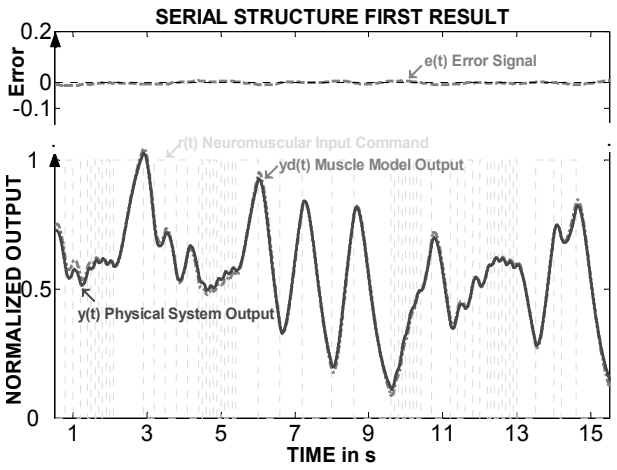

Fig. 13. First result obtained

If we compare to the parallel structure, ameliorations are clear, particularly in the two zones of fast and slow variations of the Muscle Model output. More precisely, we obtain:

- E =3% in the field of Slow variation.
- E =1% in the field of Fast variation.

Others simulations show that more increases to the number of neurons in the hidden layer and/or decreases of the sampling period have not contributed to any enhancements.

IV.2.2 Effect of delayed errors as inputs

The more important signal working in this configuration is the error, the integral of the error and the derivative of the error. Hence, why not use them as inputs to the neural network and this configuration had given better results than those mentioned above with 1% error on all zones of operations. This simulation uses the following variables:

- Ts = 100ms
- 12 neurons in the hidden layer
- $e(t), e(t-1), e(t-2), \rho(t)$ as inputs

The graph is represented in Fig. 14:

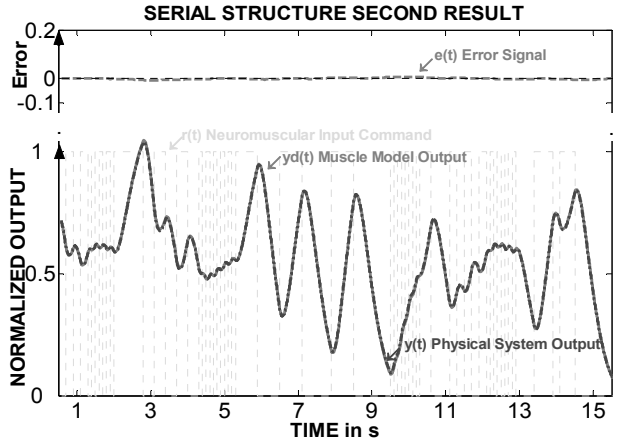

Fig. 14. Second result obtained

Ameliorations are clear in the two zones of fast and slow variations of the Muscle Model output: calculation of the Means square error gives:

- 1% in the field of Slow variation.
- 1% in the field of Fast variation.

However, the mean square error plot during the learning is given below (Fig 15).

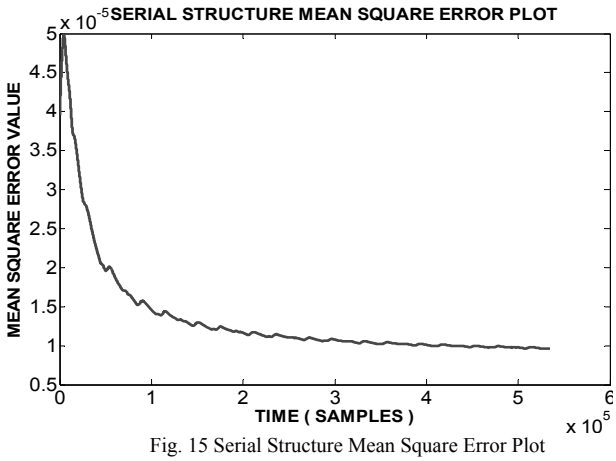

Fig. 15 Serial Structure Mean Square Error Plot

IV.2.3 Test of Robustness

The final simulation consists of the behavior of the system when the transfer function of the DC Motor changes (test of robustness). Fig.16 presents an enlarged scale of the graph:

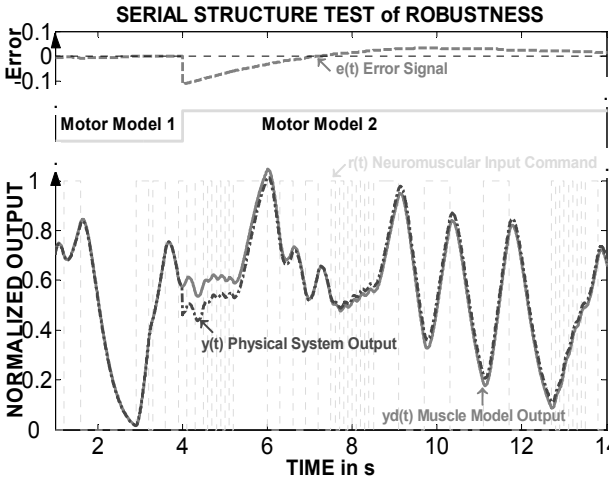

Fig. 16. Test of robustness

We can see clearly that an error will be presented when the transfer function changes but for less than 2 seconds the neural network reprogram the PID controller with the necessary parameters and the superposition remain very good.

I.V. 3 Discussion

In the first tuning structure, we put the Muscle Model in parallel with the closed loop of the DC Motor and the PID controller. This approach is interesting because after training

the Neural Network we can eliminate the Muscle Model for real time implementation. However, this structure is very complex (have 10 inputs and 80 neurons in the hidden layer), the training time is very long (about 10 hours), the sampling period is low (10ms) and the overall error obtained at the final stage is about 8% which is disadvantageous.

In the second structure, the Muscle Model is connected as a reference for the closed loop formed by the PID controller and the DC Motor. It is more advantageous. Because of its simple structure (4 inputs and 12 neurons in the hidden layer), the training time is about 1 minute and the sampling period is not necessary high (100ms). It has contributed in a much less calculation time with output error about 1%. The system has also passed the test of robustness. In the latter test we have changed online the transfer function of the DC Motor. It was demonstrated that the Neural Network has responded and programmed the PID controller for perfect results in less than a second.

The second structure presented good performance, simple structure and acceptable calculation time that facilitate the implementation in real time. All those results obtained had encouraged us to continue development in implementing this muscle model already created into a robot. This robot must have DC-Motors acting as actuators and PID controller. We think that it is possible for us to let the robot walk in a way similar to that of a human or animal one by adopting the second structure as control for the motors of the robot.

V. Conclusion and Future Work

In conclusion, we can see well that we were able to create a physical emulation of a real muscle based on a DC Motor, PID controller and tuned by a Neural Network. So this study could be incorporated into an existing robot that is using a DC Motor as joint actuators and a PID as controller in order to let the robot walk like a human.

We are now working on implementing this emulation, and emulation of others kind of muscles, into two robots. The first one is Rabbit simulator and the second one is ROBIAN, an anthropomorphic biped robot that is being developed at LIRIS Laboratory.

References

[1] N.de N.Donaldson, H.Gollee, K.J.Hunt, J.C.Jarvis, and M.K.N.Kwende, “A radial basis function model of muscle stimulated with irregular inter-pulse intervals,” *Med. Eng. Phys.*, vol. 17, no. 6, pp. 431–441, 1995.

[2] H.Gollee, D.Murray-Smith, and J.C.Jarvis, “A Nonlinear Approach to Modeling of Electrically Stimulated Skeletal Muscle”, *IEEE Trans.Biomed. Eng.„ VOL. 48, NO. 4, APRIL 2001*

[3] M.Noorgard, “Neural Networks for Modelling and Control of Dynamics Systems”, Springer 2000, ISBN : 1-85233-227-1

[4] J.Bobet, R.B.Stein, and M.N.Oguztoreli, “A linear time-varying model of force generation in skeletal muscle,” *IEEE Trans.Biomed.Eng.*, vol.40, pp. 1000–1006, Oct. 1993

[5] S.Takagi, T.Oki, T.Yamamoto and M.Kaneda, “A Skill-Based PID Controller Using Artificial Neural Networks”, Department of Communication Engineering, Okayama Prefectural University, 111 Kuboki, Soja, Okayama, 719-11 Japan p. 4454-4459, 1997

[6] C.-H.Lee,a, C.-C.Tengb, “Calculation of PID controller parameters by using a fuzzy neural network”, *ISA Transactions 42* p.p 391–400 - 2003.

[7] L.A.Bernotas, P.E.Crago, and H.J.Chizeck, “A discrete-time model of electrically stimulated muscle,” *IEEE Trans. Biomed. Eng.*, vol. 33, pp. 829–838, Sept. 1986

[8] S.J.Dorgan and M.J.O’Malley, “Nonlinear mathematical model of electrically stimulated skeletal muscle,” *IEEE Trans. Rehab. Eng.*, vol.5, pp. 179–194, Feb. 1997.

[9] W.K.Durfee and K.I.Palmer, “Estimation of force-activation, force-length and force-velocity properties in isolated, electrically stimulated muscle,” *IEEE Trans.Biomed. Eng.*, vol. 41, pp. 205–216, March 1994

[10] H.Gollee, K.J.Hunt, N.de N.Donaldson, and J.C.Jarvis, “Modeling of electrically stimulated muscle,” in *Multiple Model Approaches to Modeling and Control* R. Murray-Smith and T. A. Johansen, Eds. New York: Taylor & Francis, 1997, ch. 3.

[11] K.J.Hunt, M. Munih, N.D.Donaldson, and F.M.D.Barr, “Investigation of the Hammerstein hypothesis in the modeling of electrically stimulated muscle,” *IEEE Trans.Biomed. Eng.*,vol. 45,no.8, pp. 998–1009, 1998.

[12] T.A.Johansen and R.Murray-Smith, “The operating regime approach to nonlinear modeling and control,” in *Multiple Model Approaches to Modeling and Control*. New York: Taylor & Francis, 1997, ch. 1.

[13] M.M.N.Kwende, J.C.Jarvis, and S.Salmons, “The input–output relations of skeletal muscle,” in *Proc. R. Soc. Lond. Biol.*, vol. 261, 1995, pp. 193–201.

[14] G.I.Zahalak, “An overview of muscle modeling,” in *Neural Prostheses.Replacing Motor Function after Disease or Disability*. Oxford, U.K.:Oxford Univ. Press, 1992, pp. 17–57.

[15] H.HATZE, “The Complete Optimization of a Human Motion”, *National Research Institute for Mathematical Sciences, CSIR, P. 0. Box 395, Pretoria 0001, South Africa* Communicated by R. Bellman 1976

[16] T.Matsukuma, A.Fujiwara, M.Namba and Y.Ishida, “Non-Linear PID Controller Using Neural Networks”, *IEEE 1997*

[17] Jing-Chung Shen, “New tuning method for PID controller”, *ISA Transactions 41* - p.p. 473–484 - 2002.

[18] Yu Yongquan; Huang Ying, Zeng Bi, “A PID neural network controller”, Institute of Computer Science and Intelligent Engineering, Guangdong University of Technology, *IEEE 2003*

[19] L.A.Aguirre, “PID tuning based on model matching”, *Electronics Letters,* Vol.28 No. 25, 3rd December 1992.

[20] C.Lazar, S.Carari, D.Vrabie and M.Kloetzer, “Neuro-predictive control based self-tuning of PID controllers”, *ESANN'2004 proceedings - European Symposium on Artificial Neural Networks, Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 391-396*

[21] T.A.Johansen and B.A.Foss, “Constructing NARMAX models using ARMAX models,” *Int. J. Control*, vol. 58, pp. 1125–1153, 1993.

[22] HILL,A.V. (1938). The heat of shortening and the dynamic constants of muscle Proceedings of the Royal Society, B126, 136-195.