Subject:IES 2012: Paper Notification
From: Admin IES 2012 (ies@eepis-its.edu)
To: felixpasila@yahoo.com;
Date: Friday, August 3, 2012 10:22 AM

Dear Felix Pasila,

We are pleased to announce that your paper is ACCEPTED in IES 2012. The evaluation of your paper and all comments from reviewers of your paper are enclosed to this message. We hope that you can take them into account as much as possible in preparing the camera ready of your paper in order to accomplish a high quality paper.

Title : An Efficient Procedure For Activating Bi-State Actuator Arrays Using Neuro-Fuzzy Network
Author : Felix Pasila,,,,,
Status : ACCEPTED

And the aspects bellow is the scale of assigning score :
6 : Strong Accept (Excelent quality, cutting-edge ideas)
5 : Accept (Good quality, sound and interesting results)
4 : Weak Accept (Vote acceptance, but won't argue for it)
3 : Weak Reject (Don't like it, but won't argue against it)
2 : Reject (I will argue to reject this paper)
1 : Strong Reject (Unsuitable for publication in this forum)

==
Reviewer 1
Relevance Aspect : 5
Originality Aspect : 5
Significance Aspect : 5
Technical Aspect : 4
Clarity Aspect : 4
Overall Aspect : 5
Positive Aspect : hasil eksperimen menunjukkan efektifitas metode yang digunakan
Negative Aspect : aktuator hanya digunakan dalam 2 dimensi
Additional Comment :

==
Reviewer 2
Relevance Aspect : 6
Originality Aspect : 5
Significance Aspect : 5
Technical Aspect : 6
Clarity Aspect : 6
Overall Aspect : 5
Positive Aspect :

Negative Aspect :
Additional Comment :

We would like to inform you that the deadline of Final Manuscript Submission (camera ready) is on August 16, 2012.

We also would like to inform you later about the registration process. We are looking forward to seeing you at the conference in IES 2012, Surabaya,Indonesia.

Sincerely Yours,
Amang Sudarsono
IES 2012 General Chair

Subject:Re: Internet Banking Mandiri - Fund Transfer: IES 2012 for Felix Pasila
From: Zaqiatud Darojah (zaqiah@eepis-its.edu)
To: felixpasila@yahoo.com;
Date: Thursday, October 11, 2012 10:36 AM

Yth. Bapak Felix Pasila

Salam hormat,

Terimkasih atas konfirmasi dan partispasinya pada Seminar IES 2012 Bapak Felix. Kami tunggu kehadiran Bapak pada pelaksanaan IES tanggal 24 Oktober 2012 di Politeknik Elektronika Negeri Surabaya. Atas perhatian dan kerjasama Bapak, kami mengucapkan terimakasih.

Hormat kami
Panitia IES 2012

EEPIS - Bridge To The Future
This email was sent using EEPIS Webmail
https://mail.eepis-its.edu

An Efficient Procedure For Activating Bi-State Actuator Arrays Using Neuro-Fuzzy Network

Felix Pasila
Electrical Engineering Department
Petra Christian University
Surabaya, Indonesia
felix@petra.ac.id

Abstract— **A novel approximation procedure based on hybrid neuro-fuzzy (NF), called Takagi-Sugeno multi-real-input multi-binary-output (TS-MIMO) neuro-fuzzy network, is proposed to control the bi-state actuator arrays. This efficient procedure is used in static force mechanism of 2D-rigid body manipulator. The proposed NF model employed off-line neuro-mechanism using Levenberg-Marquardt algorithm (LMA) and Takagi-Sugeno model as inference system in fuzzy systems. Additional hill climbing method as optimal searched procedure improved the minimum error in the learning computation. Simulation results are provided not only to demonstrate the efficiency of the NF model in activating the bi-state in arrays, but also to explain how to find the minimum number of actuators that should be put in the rigid body system. These first results will lead the NF mechanism as a universal force control for bi-state actuator arrays.**

Keywords: NF type TS-MIMO procedure, general approximator, bi-state actuator arrays, one-DOF manipulator, force control fashion

I. Introduction

Binary manipulators have been used in several applications and played significant roles, especially in robotics and biomechanics applications [1,2,3,13]. These bi-state manipulators are particular kind of discrete device, in which the states flip between two possible values, such as activated-not activated, open-closed or contracted-relaxed (e.g. pneumatics, dielectric elastomer actuators, shape memory alloy). Because of the states condition, binary manipulators have several potential benefits compared to continuous manipulator systems, for example: they are relatively inexpensive and lightweight; minimal support devices because they can be operated without extensive feedback control. In contrast with the advantages, they have problem in actuating the binary states, especially when the manipulators have a massive number of actuators.

Concerning the control strategy, some previous works has been done related to discrete-actuated manipulators control. For instance, from the mid 1990s, Chirikjian [1,4] proposed a variety of efficient algorithms for trajectory tracking approximation of binary manipulators. In these algorithms, the manipulators are actuated by binary actuators that placed in a serial-parallel configuration with position control approach. They also introduced the new concept of discretely actuated manipulators called an extended Ebert-Uphoff algorithm [2,3], which considering the full position-orientation of inverse kinematics problem rather than the pure position problem. This algorithm gives many influences to researchers, such as constructing hyper-redundant manipulators based on mathematical model and controlling the desired position-orientation of end-effector.

In contrast with position control fashion, not many researchers try to control binary manipulators in force control fashion. This is because the complexities of force control model increase exponentially proportional to the number of actuator arrays. For instance, Yang et.al [13] proposed some online mechanisms based on Brute Force procedure for force control mechanism. This approach requires too many calculations for real-time manipulator control.

Regarding the force problem in binary manipulator systems, we propose a new procedure of hybrid neuro-fuzzy networks (NF) to approximate bi-state (1 and 0 values) of the actuator arrays. This configuration is an off-line model and proposes very fast learning mechanism (compared to Brute Force procedure) for dealing with the huge input datasets which enter to the network. Moreover, an NF is already well-known as one of universal approximator's tools in computational intelligence field (CI) that combines the learning ability from neural network and rule-based interpretation from fuzzy systems. Because of this reason, an NF is widely exploited to discover optimal parameters from the learning mechanism efficiently. In addition, an NF is also known as an adaptive network, because this network consists of nodes and directional relations in which the nodes are connected. Most of the nodes are adaptive, which denote their outputs depend on the parameters concerning to these nodes, and the learning rule indicates how these parameters should be updated to minimize the error performance. According to Palit et.al [5-9], the NF mechanism has been proved as an efficient approximator in engineering applications, such as time series forecasting, pattern recognition, and for modeling and control. In addition, they clarified that the fuzzy if-then rules on NF tries to fit the non-linear relation between input-output pairs that represented in large array datasets (as data training) and in the meantime learning mechanism updates the parameters until minimum error is fulfilled. For validation purposes, they use a large number of new datasets as data testing to verify the training network performance.

Furthermore, we explore a simple 2D-rigid body system as manipulator. This manipulator is actuated by binary force actuators in parallel configuration and has one DOF which rotates respect to the reference frame. For control scheme, we recommend an NF with Levenberg-Marquardt algorithm (LMA) and hill climbing method (HC) as learning and optimization procedures respectively. Moreover, both inputs and outputs of the NF are generated from forward static force problem which mean: given the rotate angle of the rigid body (fixed) and all combination of the states of actuators (F_{bin}) that defines which of the actuators are active, we calculate the resultant vector of forces (R) and moments (M) from equilibrium equations in the array fashion. Using the reverse way, we exploit the result of R and M from forward static force equations as inputs of the NF whereas the goal of reverse problem is F_{bin}, denotes as binary output prediction on the networks.

As results, this proposed method has two main objectives: First, designing the NF mechanism with off-line learning and flexible input output arrays. This purpose is important for designing the general control mechanism with flexible number of inputs and outputs. Second, proposing additional search algorithm method to discover the optimized parameters of the network. For determining these two goals, we validate the control mechanism with standard error performance and several number of data tests.

For more details, the explanation of the rigid body system with pure rotation and its mathematic equations are described in Section II, whereas TS type MIMO neuro-fuzzy network, including its training algorithm and simple optimization parameters are explained in Section III. Thereafter, the simulation experiments and results are shown in Section IV. Finally, brief concluding remarks are presented in Section V.

II. Equilibrium Equations Of Rigid Body Systems

For designing a force control of rigid body, we must recognize the term of equilibrium condition. The rigid body is said to be in equilibrium if the sum of forces and moments acting on it, about any arbitrary point O are equivalent to zero. For instance, if rigid body manipulator is in plane, we need to derive at least two equilibrium equations in order to find three variables (2 forces and 1 moment, all in vector arrays). Let us consider a rigid body manipulators with parallel configuration in Fig.1. A set of binary force actuators are acting at upper position $\overline{A_\iota}$ and the lower position $\overline{B_\iota}$, whereas $l = 1,2,\dots,n$, n is the number of actuators. Both positions here are respect to reference $O(0,0)$. If array of forces actuated the binary actuators, then the manipulator will rotate with angle α. The new position of $\overline{A_\iota}$will become $\overline{A_{l\alpha}}$ while the position of $\overline{B_\iota}$ is always fix after rotation. Hereafter, the individual force $\overline{F_\iota}$ which is produced by actuators, has vector direction of $\overline{A_{l\alpha}}$ and $\overline{B_\iota}$. Additionally, we define $\sum \overline{F_\iota}$ or $\overline{F}_{sum}$ as the resultant force of n actuators, corresponding to $\overline{F}_{sum} = \overline{F_1} + \overline{F_2} + \cdots + \overline{F_l}$, and $\sum \overline{F_\iota} \times \overline{A_{l\alpha}}$ as equivalent to the summing of each moments $\sum \overline{M_\iota} = \overline{M_1} + \overline{M_2} + \cdots + \overline{M_l}$. The term $\overline{M_l}$ is an individual moment, derived from cross product between force and related upper position, with respect to the reference O (0,0). Henceforward, (2.1) and (2.2) show $\bar{R}$ and $\bar{M}$ as the total force and moment should be given to keep the rigid body in the equilibrium position:

$$\bar{R} = -\sum \overline{F_\iota} \tag{2.1}$$

$$\bar{M} = -\sum \overline{F_\iota} \times \overline{A_{l\alpha}} \tag{2.2}$$

More details, these steps below explain how to calculate all forces and moments of a rigid body system based on (2.1) and (2.2) as shown below:

$$\overline{A_{l\propto}} = \overline{R_\alpha}.\overline{A_\iota},\ \overline{R_\alpha} = \begin{bmatrix} c\propto & -s\propto \\ s\propto & c\propto \end{bmatrix} \tag{2.3}$$

$$\overline{U_\propto} = \frac{\overline{A_{l\propto}} - \overline{B_\iota}}{\|\overline{A_{l\propto}} - \overline{B_\iota}\|} \tag{2.4}$$

In general, if the system rotate with angle α, then the new upper position $\overline{A_{l\propto}}$ is determined easily by multiplying rotation matrix $\overline{R_\alpha}$ with $\overline{A_\iota}$. After Eq. 2.3, we calculate the unit vector of forces $\overline{U_\propto}$ in Eq. 2.) as well. Hereafter, the individual force $\overline{F_l}$ is also established by multiplying the each combination of the state vector $\overline{F_{b\iota n}}$ (in total = 2^n combinations for every pose) with the constant force amplitude F_{amp} and unit vector $\overline{U_\propto}$ correspondingly (2.5) as follows

$$\overline{F_l} = F_{amp}.\overline{F_{b\iota n}}.\overline{U_\propto} \tag{2.5}$$

Hence, the force reaction $\bar{R}$ with two components in X and Y axis are recognized by taking the opposite value of $\overline{F_l}$ summation (2.6). Finally, the moments $\bar{M}$ is calculated by summing all individual moments(in opposite direction) that produced by cross product between forces and upper position after rotation (2.7).

$$\bar{R} = \begin{Bmatrix} R_x \\ R_y \end{Bmatrix} = -\sum \overline{F_l} \tag{2.6}$$

$$\bar{M} = -\sum(\overline{F_l} \times \overline{A_{l\propto}}) \tag{2.7}$$

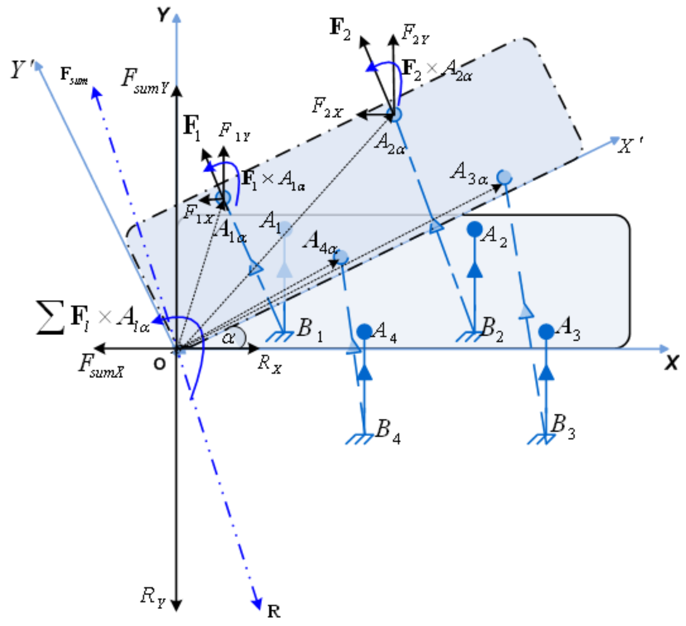

Figure 1. Four-bit actuator arrays in planar (pure rotation)

In order to find the minimum actuators that should be put on rigid body system, like describe in Fig.1, we propose equivalent systems of 4, 8, 12 actuators (also known as 4-bit, 8-bit and 12-bit actuators) in which they minimize error testing prediction. This means, we do not need to put more than 20 actuators to control the states of actuators from any dataset inputs, if 8-bit or 12-bit can fulfill the error requirement, such as the average error prediction not more than 10% (this approach reduces time computing and cost). For this reason, we need to build input-output arrays of several equivalent force systems and by applying NF to these arrays we can compare the error performances.

Now we describe briefly about equivalent force systems. Two systems of forces are said equivalent if the sum of forces and moments acting on them, about any arbitrary point are equal (equivalent, because they would have the same effect on a rigid body, according to Newton's law).

In addition, for calculating the performance error of these three equivalent systems, we used data test from equivalent force system of 15-bit. For more details, we exploit equations from (2.8a) until (2.10c) to show how to make equivalent force system from 4-bit, as based actuators, and the 8-bit, 12-bit and 15-bit are the equivalent force systems. In practice, we define a point O_1, where the resultant moment in this point is equal to zero. In this case, we find out the position of actuators and determined the point O_1 that gives the total moment ≈ 0, which is done by experiments (this experiment is done by choosing the centre of mass as a point O_1 from n-bit system). As results, forces and moments of these four equivalent systems can be seen in Table 1.

$$\overline{M_{O_1}^{n}} = \overline{F^{n}} \times (\bar{A_\iota} - O_1) = 0, \qquad (2.8a)$$

where n = no. of based actuator (n=4)

$$\overline{M_{O_1}^{4}} = \overline{F^{4}} \times (\bar{A_\iota} - O_1) = 0 \qquad (2.8b)$$

$$0 = \overline{F^4} \times \bar{A_\iota} + \overline{F^4} \times (O - O_1), \qquad (2.8c)$$

$$0 = \overline{M_O^4} + \overline{F^4} \times (O - O_1) \qquad (2.8d)$$

$$\overline{M_O^4} = -\overline{F^4} \times (O - O_1) \qquad (2.8e)$$

By control the force amplitude so the equivalent force becomes:

$$\overline{F^4} = \overline{F^8} = \overline{F^{12}} = \overline{F^{15}} \qquad (2.9)$$

Then we can state equivalent moments as:

$$\overline{M_O^8} = -\overline{F^8} \times (O - O_1) \qquad (2.10a)$$

$$\overline{M_O^{12}} = -\overline{F^{12}} \times (O - O_1) \qquad (2.10b)$$

$$\overline{M_O^{15}} = -\overline{F^{15}} \times (O - O_1) \qquad (2.10c)$$

The results of equivalent force systems of 4-bit, 8-bit. 12-bit and 15-bit that calculated on equations (2.8 – 2.10) must be shown the similarity. The similarity of the forces and moment can be seen on Table I.

TABLE I. TABLE COMPARISON OF EQUIVALENT OF FORCE SYSTEMS WITH $\alpha = 20, F_{amp} = \frac{10x4}{no\_of\ actuators}$:

N-bit Actuators	**Range Force R_x**	**Range Force R_y**	**Range Moment M**
4	0 to 15.515	0 to -36.589	0 to 401.96
8	0 to 15.544	0 to -36.648	0 to 399.79
12	0 to 15.513	0 to -36.642	0 to 400.17
15	0 to 15.538	0 to -36.677	0 to 402.82

III. NEURO-FUZZY PROCEDURES

In the field of computational intelligence, neuro-fuzzy refers to combinations of artificial neural networks and fuzzy logic. This idea was proposed first by J. S. R. Jang [11]. Recently, this hybrid approach have received many attention from researchers whose dealing with non-linear control applications [8-9]. Moreover, neuro-fuzzy model is a hybrid intelligent system which combines the human-like reasoning style of fuzzy systems with the learning ability of neural networks. The main advantages of neuro-fuzzy system are: it interprets IF-THEN rules from input-output relations and it is an efficient universal-approximator. These are the main motivation for us to develop a control scheme of binary manipulator. In case of rigid body manipulator, the relation between inputs (forces and moments) and outputs (state of the actuators) in neuro-fuzzy model is explicitly shown in Fig. 2. Three normalized inputs (minimum value = 0 and maximum value = 1) and n binary outputs denote as input-output arrangement. Next, more explanation about the procedure of the fuzzy structure will be explained in Section 3.1 along with the learning procedure which is explored in Section 3.2. Additionally, the strategy to convert the real value of outputs fuzzy into binary value by using average threshold strategy can be seen in Section 3.3.

A. *Fuzzy logic system: Takagi-Sugeno type MIMO*

Neuro-fuzzy model as shown in Fig. 2 below is based on Gaussian membership functions (GMFs). It uses Takagi-Sugeno (TS) model (note: compare to Mamdani model and Pedrycs model, both linguistic fuzzy models that are focused on interpretability, a TS model is a precise fuzzy model that focused on accuracy and it has strongly connection between input and output). Moreover, a TS model is easy to interpret, and its outputs obtained from the networks are crisp outputs, means directly compatible to the actuator's states.

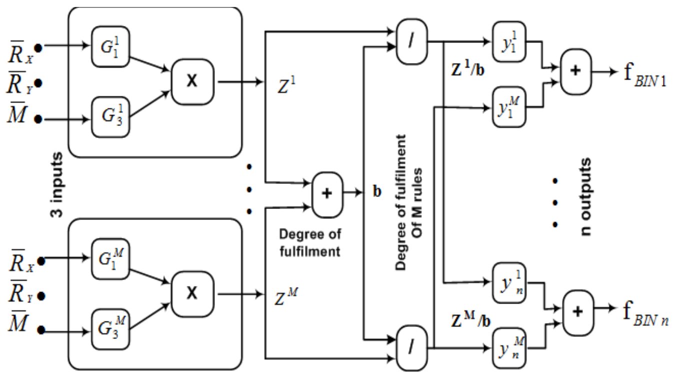

Figure 2. Takagi-Sugeno-type MIMO (with input real and output binary) feedforward Neuro-Fuzzy network, no. input = 3, no. output = n-bit actuators, no. membership function = M, training method: LMA

As shown in Fig.2, the Gaussian nodes G_1^1 to G_3^M calculate the degree of membership of the numerical input values in the antecedent (IF part) fuzzy sets. The product nodes (×) represent the antecedent conjunction operator and the output of this node is the corresponding degree of fulfillment Z^l, with l = 1, 2,…, M, representing the number of membership rules.

After this, the division symbol (/), together with summation (+), join to make the normalized degree of fulfillment (Z^l/b) of the corresponding rule. Then, the multiplication of Z^l/b with the corresponding TS rule consequent y_j^l (THEN part) is used as input to the last summation part (+) at the crisp output value f_{BIN}, which is directly compatible with the bi-state of the actuator arrays.

Furthermore, fuzzy model type TS-MIMO, with Gaussian membership functions (GMFs), product inference rule, and a weighted average defuzzifier can be defined as three layers feedforward network, as explained in (3.1a-3.1c) below: (see [5] for details).

$$z^l = \prod_{i=1}^{3} G_i^l(x_i)\,;\; G_i^l(x_i) = exp\left(-\left(\frac{x_i-c_i^l}{\sigma_i^l}\right)^2\right) \quad (3.1a)$$

$$y_j^l = W_{0j}^l + W_{1j}^l x_1 + W_{2j}^l x_2 + W_{3j}^l x_3 \quad (3.1b)$$

$$f_{BINj} = \sum_{l=1}^{M} y_j^l \cdot h^l \quad (3.1c)$$

where $h = z^l/b$, and $b = \sum_{l=1}^{M} z^l$

and f_{BINj} is the output binary state of the *j* actuators(output prediction).

The corresponding l^{th} rule from the above fuzzy logic system (FLS) can be written as

$$R^l : IF\ x_1 \text{ is } G_1^l \text{ AND ... AND } x_3 \text{ is } G_3^l\ THEN$$
$$y_j^l = W_{0j}^l + W_{1j}^l x_1 + ... + W_{3j}^l x_3. \quad (3.2)$$

where, x_i with *i* = 1, 2, 3; are the *3* inputs, f_j with *j* = 1, 2, …, *n*; are its *n* outputs, and G_i^l with *i* = 1, 2, 3 and *l* = 1, 2, …, *n* are the Gaussian membership functions of form (3.1) with the corresponding mean and variance parameters c_i^l and σ_i^l respectively and with y_j^l as the output consequent of the l^{th} rule. It must be remembered that the Gaussian membership functions G_i^l actually represent linguistic terms such as low, medium, high, very high, etc. The rules as written in (3.2) are known as Takagi-Sugeno rules.

Because of the neural network procedure (neuro) is implemented to the Takagi-Sugeno model, this figure represents a Takagi-Sugeno-type of MIMO (multi real-input multi binary-output) neuro-fuzzy network, where instead of the connection weights and the biases in neural network, we have here the mean c_i^l and also the variance σ_i^l parameters of Gaussian membership functions, along with the rules consequent W_{oj}^l, W_{ij}^l parameters, as the equivalent adjustable parameters of the network. If all these parameters of NF network are properly selected by training mechanism, then the FLS can correctly approximate any nonlinear systems based on given data of inputs (forces, moments)-outputs(state of the actuators) pairs.

B. Levenberg Marquardt Algorithm (LMA) procedures

More detail of LMA equation, will be explained below. If a function $V(w)$ is to be minimized with respect to the parameter vector *w* (these parameters are to be updated in training algorithm) using Newton's method [12], the updated parameter vector w and Δw are defined as:

$$\Delta w = -\left[\nabla^2 V(w)\right]^{-1} \cdot \nabla V(w) \quad (3.3a)$$

$$w(k+1) = w(k) + \Delta w \quad (3.3b)$$

In equation (3.3a), $\nabla^2 V(w)$ is defined as the Hessian matrix and $\nabla V(w)$ is the gradient of $V(w)$. If the function $V(w)$ is taken to be a SSE function as follows:

$$V(w) = 0.5 \cdot \sum_{r=1}^{N} e_r^2(w), \quad (3.4)$$

then the gradient of $V(w)$ and the Hessian $\nabla^2 V(w)$ are generally defined as:

$$\nabla V(w) = J^T(w) \cdot e(w) \quad (3.5a)$$

$$\nabla^2 V(w) = J^T(w) \cdot J(w) + \sum_{r=1}^{N} e_r(w) \cdot \nabla^2 e_r(w) \quad (3.5b)$$

where, the Jacobian matrix $J(w)$ is as follows:

$$J(w) = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_{N_p}} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_{N_p}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_{N_p}} \end{bmatrix} \quad (3.5c)$$

From (3.5c), the dimension of the Jacobian matrix is $N \times N_p$, where N is the number of training samples and N_p is the number of adjustable parameters in the network. For the Gauss-Newton method, the second term in (3.5b) is assumed to be zero. Therefore, the updated equations according to (3.3a) will be:

$$\Delta w = -\left[J^T(w) \cdot J(w)\right]^{-1} \cdot J^T(w) \cdot e(w) \quad 3.6a)$$

Now let us see the Levenberg-Marquardt's modifications of the Gauss-Newton method:

$$\Delta w = -\left[J^T(w) \cdot J(w) + \mu \cdot I\right]^{-1} \cdot J^T(w) \cdot e(w) \quad (3.6b)$$

where, *I* is the $N \times N_p$ identity matrix, and the parameter μ is multiplied or divided by some gain/factor whenever the iteration step increases or decreases the value of $V(w)$.

Here, the updated version of (3.3a), can be seen on (3.6c) as follows:

$$w(k+1) = w(k) - \left[J^T(w) \cdot J(w) + \mu \cdot I\right]^{-1} \cdot J^T(w) \cdot e(w) \quad (3.6c)$$

It is important to know that for large μ, the algorithm becomes the steepest descent algorithm with step size $1/\mu$, and for small μ, it becomes the Gauss-Newton method.

Furthermore, Xiaosong et.al [10] also proposed to add modified error index (MEI) term in order to improve training convergence. The corresponding gradient with MEI can now be defined by using a Jacobian matrix as:

$$\nabla SSE_{new}(w) = J^T(w) \cdot \left[e(w) + \gamma \cdot \left(e(w) - e_{avg}\right)\right] \quad (3.7)$$

where $e(w)$ is the column vector of errors, e_{avg} is the average training error of each column, while γ is a constant factor, $\gamma << 1$ has to be chosen appropriately.

Now, the computation of Jacobian matrix can be performed as follows. The gradient $\nabla V(W_{0j}^{l})$ can be written as:

$$\nabla V(W_{0j}^{l}) \equiv (\partial S / \partial W_{0j}^{l}) = \{z^{l}/b\} \cdot (f_j - d_j) \quad (3.8)$$

where, f_j and d_j are respectively the actual output and desired output of the Takagi-Sugeno type MIMO neuro-fuzzy network. Now, by comparing (3.8) to (3.5a), where the gradient $\nabla V(w)$ is expressed as the transpose of the Jacobian matrix multiplied with the network's error vector, i.e.

$$\nabla V(w) = J^{T}(w) \cdot e(w) \quad (3.9)$$

the corresponding Jacobian matrix for the parameter W_{0j}^{l} of the NF network can be written as:

$$J(W_{0j}^{l}) \equiv [J^{T}(W_{0j}^{l})]^{T} = [z^{l}/b]^{T} \quad (3.10)$$

where prediction error of NF network is written as:

$$e_j \equiv (f_j - d_j) \quad (3.11)$$

If the normalized prediction error on NF network is considered, then instead of equations (3.10), the corresponding Jacobian will be as follows:

$$J(W_{0j}^{l}) \equiv [J^{T}(W_{0j}^{l})]^{T} = [z^{l}]^{T} \quad (3.12)$$

This is because the normalized prediction error of the MIMO-NF network is

$$e_j(normalized) \equiv (f_j - d_j)/b \quad (3.13)$$

Similarly, Jacobian matrix itself for the parameter w_{ij}^{l} of the NF network can be written as:

$$J(W_{ij}^{l}) \equiv [J^{T}(W_{ij}^{l})]^{T} = [(z^{l}/b) \cdot x_i]^{T} \quad (3.14)$$

Also, by considering normalized prediction error from (3.11), equations (3.14) then become:

$$J(W_{ij}^{l}) \equiv [J^{T}(W_{ij}^{l})]^{T} = [z^{l} \cdot x_i]^{T} \quad (3.15)$$

Now, the Jacobian matrix computation of the remaining parameters c_i^{l} and σ_i^{l} are performed by defining the terms D_{eqv} and e_{eqv} as

$$D \equiv D_{eqv} \cdot e_{eqv} = (D_1 \cdot e_1 + D_2 \cdot e_2 + \cdots + D_m \cdot e_m) \quad (3.16)$$

Where, $D_j = (y_j^{l} - f_j)$ and $e_j = (f_j - d_j)$ with j = 1, 2, .., m and the term e_{eqv} is such that it contributes the same amount of sum squared error that can be obtained jointly by all the errors e_j from the MIMO network. Therefore,

$$e_{eqv}^{p} = \sqrt{(e_1^{p^2} + e_2^{p^2} + \cdots + e_m^{p^2})} \quad (3.17)$$

Where, p = 1, 2, …, n; corresponding to n as number of training samples/data.

Now, by considering normalized equivalent error in (3.13), taking into account the equation (3.9), the transposed Jacobian matrix for the parameters c_i^{l} and σ_i^{l} can be computed as:

$$J(c_i^{l}) = [J^{T}(c_i^{l})]^{T} = \left[2 \cdot D_{eqv} \cdot z^{l} \cdot (x_i - c_i^{l}) / (\sigma_i^{l})^{2}\right]^{T} \quad (3.18)$$

$$J(\sigma_i^{l}) = [J^{T}(\sigma_i^{l})]^{T} = \left[2 \cdot D_{eqv} \cdot z^{l} \cdot (x_i - c_i^{l})^{2} / (\sigma_i^{l})^{3}\right]^{T} \quad (3.19)$$

The above procedure describes actually layer by layer computation of Jacobian matrices for various parameters of neuro-fuzzy network (see [8] for details). Again, after finish all the Jacobian computation, back to the equation (3.6c) for updating the parameters. This updating procedure stops after achieving the maximum iteration or the minimum error, with setting parameters: maximum iterations = 500, momentum constant = 0.005, oscillation control *WF* = 0.5%, constant factor γ = 0.0007.

In addition to LMA procedure, we proposed a simple optimization procedure in order to find the tuning parameters on neuro-fuzzy systems, known as randomized Hill Climbing procedure (HC). This procedure is a local search algorithm and tries to find the best local minimum from huge stochastic procedure, by permitting the best training parameters that minimize the error function (RMSE) and neglecting the others. The results of using HC can be shown latter in Table III.

The following steps of HC are carried out:

a. Run the loop: No. membership function form 2-15 (M)
b. Run the loop: No. experiment of each initial parameters = 100
c. Run the loop: No. of iteration, each experiment has iteration from 50 to 500
d. Calculate training output and find the suitable prediction model $\widehat{\boldsymbol{F_{bin}}}$
e. Save the parameters after step ***d*** completed. If the next iteration produce better result, replace the old parameters, otherwise the new parameters are neglected.
f. Determine the forces and moments prediction and calculate the prediction performance error
g. Early stopping criteria[6]: Terminate the program if criteria are satisfy in step ***f***
h. Repeat again step ***a*** if needed and save the best parameters (time computing in 2 weeks).

IV. EXPERIMENT PROCEDURES AND RESULTS

By using neuro-fuzzy network that shown in Fig.2, we propose two experiments for modeling and testing the rigid body manipulator as follows:

1. Input forces and moments from fix angle $\alpha = 20$
   There are 3 equivalent actuator model (4-bit, 8-bit and 12-bit actuators) and testing data taken from equivalent 15 actuators. The results of this experiment are shown in Table II and Fig.3 .
2. Input forces and moments from variety angle α (10, 20, 30, 40, 50, 60, 70).
   In this experiment, a rigid body system with 12 actuators is used to generate input-output arrays for training data. Testing data is taken from 12 actuators systems with different angle α (17, 25, 35, 45). Detail explanation and results are clarified in Table III and Fig.4 to Fig 6.

TABLE II. 1ST EXPERIMENT WITH 4-BIT, 8-BIT AND 12-BIT PERFORMANCES (AFTER 2 WEEKS LEARNING)

Descriptions	4-bit Actuators	8-bit Actuators	12-bit Actuators
RMSE RX	1.6862	1.2726	0.8440
RMSE RY	4.2326	2.5840	2.0005
RMSE MR	34.9179	27.3226	20.0735
% Error RX	17.20	13.99	9.19
% Error RY	24.30	13.17	9.84
% Error MR	15.51	12.56	8.78

Procedure for 1st experiment according to Fig.2 can be explained with several steps:

a. Derive the equilibrium equations (2.1 to 2.7) using 4-bit, 8-bit and 12-bit actuator system with alpha=20 deg and put the result in input output arrays $XIO = [\overline{\boldsymbol{R_x}} \;\; \overline{\boldsymbol{R_y}} \;\; \overline{\boldsymbol{M}} \;\; \overline{\boldsymbol{F_{b\imath n}}}]$

b. Calculate training output of the state actuators $\widehat{\boldsymbol{F_{b\imath n}}}$

c. Generate testing data from equivalent system 15-bit with 50 data set.

d. Determine binary prediction $\widehat{F_{b\imath n}}^*$

e. Calculate again forces and moments model using $\widehat{F_{b\imath n}}^*$ (for validation)

f. Determine performance error

g. Repeat step a to f using search method (Hill Climbing) for finding the best parameter that give minimum error in two weeks.

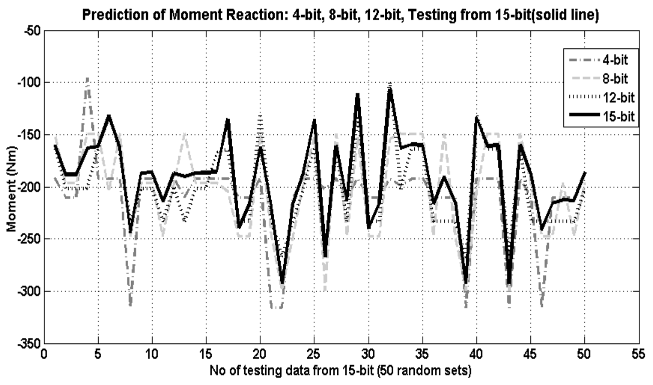

Figure 3. Moment model and testing with single $\alpha = 20$ deg and actuators = 4-bit, 8-bit, 12-bit with data testing 15-bit

Moreover, Fig.3 demonstrates the performance of the neuro-fuzzy model using 3 inputs with orientation angle 20 (deg). The dashed lines are the validation of forces and moments of the model when 50 sets of testing data from equivalent system 15 actuators (black-solid lines) are entered the trained model of 4-bit, 8-bit and 12-bit. Furthermore, Table II shows the performance results after 2 weeks learning time. We can see that 12-bit model can bring the performance RMSE down to 10%, as error minimum required.

TABLE III. 2ND EXPERIMENTS, 12-BIT PERFORMANCES

No. Actuators = 12, Optimized Parameters Alpha learning = 10,20,30,40,50,60,70, with 100 data testing from alpha 17, 25, 35, 45		
Description	***Learning after 1 weeks***	***Learning after 2 weeks***
RMSE RX	1.7455	1.3499
RMSE RY	2.6064	2.2929
RMSE MR	22.8522	17.0078
% Error RX	13.96	11.69
% Error RY	12.47	10.77
% Error MR	12.74	9.84

Procedure for 2nd experiment according to Fig.2 is similar with the first experiment but we used several learning orientation angle. Here, 12-bit actuators are chosen as the optimized number of actuator arrays. The result of 1 week and 2 weeks computing time can be seen on Table III as well as some figures on the validation using 100 data test from the 12-bit with angle 17, 25, 35, 45 (25 data from each angle).

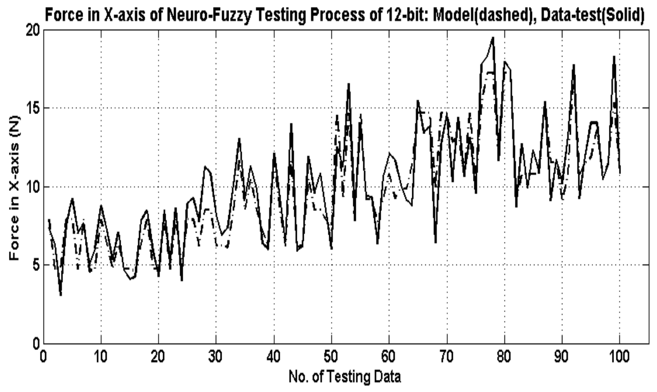

Figure 4. Force X-axis validation of the neuro-fuzzy 12-bit model, using data testing 12-bit from angle α (17, 25, 35, 45) after 2 weeks learning

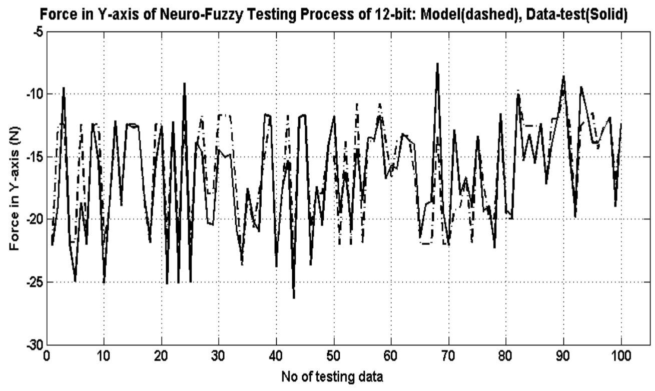

Figure 5. Force Y-axis validation of the neuro-fuzzy 12-bit model, using data testing 12-bit from angle α (17, 25, 35, 45) after 2 weeks learning

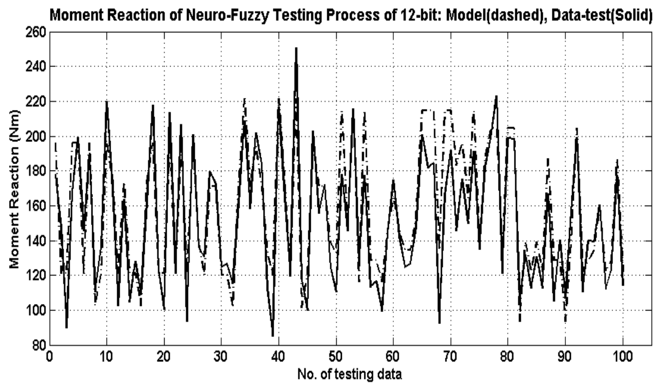

Figure 6. Moments validation of the neuro-fuzzy 12-bit model, using data testing 12-bit from angle α (17, 25, 35, 45) after 2 weeks learning

Table III shows the performance result of the input model of 12-bit with alpha from 10, 20, 30, 40, 50, 60 and 70, with its testing data from 12-bit with alpha 17, 25, 35, and 45. In addition, Figs. 4-6 illustrate the performance of the neuro-fuzzy model using 3 inputs and 12-bit outputs of forces and moments, to keep the rigid body in equilibrium position, when 100 sets of testing data(random testing) are entered the NF 12-bit model. The results of the model validation feature error average validation RMSE around 13% after 1 week computation and RMSE around 10% after 2 weeks computation. These results give the improve of using randomized Hill Climbing in the learning computation.

V. Conclusion

In this paper, a neuro-fuzzy network type TS MIMO have been presented for prediction the states of the actuators with the input networks are forces and moments of rigid body manipulator (parallel configuration, in 2D) that already known before. It has been demonstrated that neuro-fuzzy network and trained with Levenberg-Marquardt algorithm, is very efficient to find performance error around 10% in off-line learning and with around hundred testing data. The neuro-fuzzy model also proposed the number of actuators = 12 as the minimum actuators that should be put in the rigid body system. From our first results, it can be inferred that this model has flexibility to test the trained model with any orientation inputs.

For the next issue, we would like to use the efficient NF to more complex geometry of binary manipulators such as robot manipulator with bi-state actuator arrays for rehabilitation purpose.

Instead of neuro-fuzzy, other approximator method, such as recurrent neural network (RNN) can be used in these problems. Theoretically, the biologically inspired solution, such as RNN method can bring the performance error less than neuro-fuzzy, although it uses a huge computation time.

References

[1] G. S. Chirikjian, A binary paradigm for robotic manipulators, in: Proc. IEEE Int. Conf. On *Robotics and Automation*, San Diego, CA, pp. 3063–3069 (1994).

[2] I. Ebert-Uphoff and G. S. Chirikjian, Efficient workspace generation for binary manipulators with many actuators, *J. Robotic Syst.* 12 (6), 383–400 (1995).

[3] I. Ebert-Uphoff, On the development of discretely-actuated hybrid-serial–parallel manipulators, PhD Dissertation, Johns Hopkins University (1997).

[4] J. Suthakorn and G. S. Chirikjian, “A new inverse kinematics algorithm for binary manipulators with many actuators,” Adv. Robot., vol. 15, no. 2, pp. 225–244, 2001.

[5] A.K. Palit, R Babuška, “Efficient training algorithm for Takagi-Sugeno type Neuro-Fuzzy network,” Proc. of FUZZ-IEEE, Melbourne, Australia, vol. 3: 1538-1543, (2001)

[6] A.K. Palit, D. Popovic, “Nonlinear combination of forecasts using ANN, FL and NF approaches,” FUZZ-IEEE, 2:566-571, (2002).

[7] A. K. Palit, G. Doeding, W. Anheier, and D. Popovic, “Backpropagation based training algorithm for Takagi-Sugeno-type MIMO neuro-fuzzy network to forecast electrical load time series,” Proc. Of Fuzz-IEEE, Honolulu, Hawai, vol. 1:86-91, (2002)

[8] A. K. Palit, D. Popovic, “Computational Intelligence in Time Series Forecasting, Theory and Engineering Applications”, Springer, (2005).

[9] F. Pasila, A.K. Palit, G. Thiele, “Neuro-Fuzzy Approaches for Electrical Load forecasting using additional Moving Average Window Data Filter on Takagi-Sugeno Type MISO Network” Journal of Advanced Computational Intelligence & Intelligent Informatics, JACIII 12(4): 361-369, Fuji Technology Press Ltd, Japan, (2008)

[10] D. Xiaosong, D. Popovic, G. Schulz-Ekloff, Oscillation resisting in the learning of Backpropagation neural networks, Proc. of 3rd IFAC/IFIP, Ostend, Belgium, (1995)

[11] J.S.R. Jang, “ANFIS: Adaptive network Based Fuzzy Inference System,” IEEE Trans. On SMC., 23(3):665-685, (1993)

[12] M.T. Hagan, “Training feedforward networks with the Marquardt algorithm”, IEEE Transaction on Neural Networks, Vol. 5, No. 6, (1994)

[13] P.H. Yang, “Design and Control of Bundles of Binary Actuators for Manipulator Actuation,” PhD Dissertation, The Ohio State University, (2001)