2

A Framework for Bridging the Gap Between Symbolic and Non-Symbolic AI

Gehan Abouelseoud[1] and Amin Shoukry[2]
[1]*Alexandria University*
[2]*Computer Science and Eng. Dept, Egypt-Japan University of Science and Technology (EJUST), Alexandria,*
Egypt

1. Introduction

FRS (Fuzzy Rule Systems) and ANNs (Artificial Neural Networks) have gained much popularity due to their capabilities in modeling human knowledge and in learning from data, respectively. Fuzzy systems have the advantage of allowing users to incorporate available experts' knowledge directly in the fuzzy model [2]. Thus, decisions made by fuzzy systems are transparent to the user (i.e. the reasons behind the decisions made are clearly understood by tracing the decision and finding out which rules fired and contributed to it). However, there are many parameters whose values are arbitrary. These values have to be "guessed" by a fuzzy system designer, yet they largely influence the system behavior. Thus, a fuzzy system is as good as its programmer.

On the other hand, ANNs have the advantage of being universal functions approximators, requiring only sample data points and no expert knowledge [3]. Despite their advantages, they are essentially black box models. This means that the reasons behind their decisions are concealed in the knowledge acquired in the trained weights. However, these usually have no clear logical interpretation. Thus, their reliability is questionable.

To combine the advantages of both systems while overcoming their disadvantages, two approaches have been proposed in literature. The first is rule extraction from weights of trained ANNs [4]-[7]. However, the proposed approaches often yield some "un-plausible" rules, thus rule pruning and retraining is often required. For examples, some rules may be impossible i.e. their firing depends on conditions that can never occur in reality (*impossible antecedents*). For example, a rule dictating that a certain action is to be taken in case time is negative. The second approach is ANFIS [8], which attempts to cast the fuzzy system as an ANN with five layers. Although, only two of these layers are adaptable, this model is still more complicated to build and train than a conventional feed-forward ANN for two main reasons. First, the user’s expertise is required to choose appropriate consequent (output) membership functions. Second, the desired output needs to be known a priori. This may not be possible for several applications including design problems, inverse problems and high dimensional problems. For example, in a robot path tracking problem, the ANN is required to predict the correct control input. In such application, the desired performance is known but no real-solid rules exist, especially, if the robot is required to be self-adaptive. Similarly, consider a car shape optimization problem. The ANN is required to estimate the shape parameters required to achieve certain air resistance during car motion. Constraints on the shape parameters exist; however, no clear rule database exists relating shape parameters to the desired performance.

The present work proposes a new approach that combines the advantages of fuzzy systems and ANNs through a simple modification of ANN's activation calculations. The proposed approach yields weights that are readily interpretable as logical consistent fuzzy rules because it includes the "semantic" of both input and output variables in the learning/optimization process.

The rest of the chapter is organized as follows. Section II describes the proposed framework. Section III demonstrates its effectiveness through a case study. Section IV shows how it to can be generalized to solve optimization problems. An illustrative example is given for this purpose. Finally, Section V concludes the chapter with a summary of the advantages of the proposed approach.

2. The proposed approach

Fig.1 shows a typical feed-forward ANN with a single hidden layer of sigmoid neurons. Conventionally, the output of such an ANN is given by:

$$o_k = \sum_{j=1}^{N_h} w_{jk} sig\left(\sum_{i=1}^{N_i} w_{hij} x_i^p + b_j \right) \tag{1}$$

where $w_{hij}, b_j, w_{jk}, x_i^p, N_h, N_i, o_k$ are the weight of the connection between the ith input to the jth hidden neuron, the bias of the jth hidden neuron, the weight of the connection between the jth hidden neuron and the kth output neuron, the number of hidden neurons, the number of inputs (elements in each input pattern), the kth output, respectively. The number of output neurons is N_o.

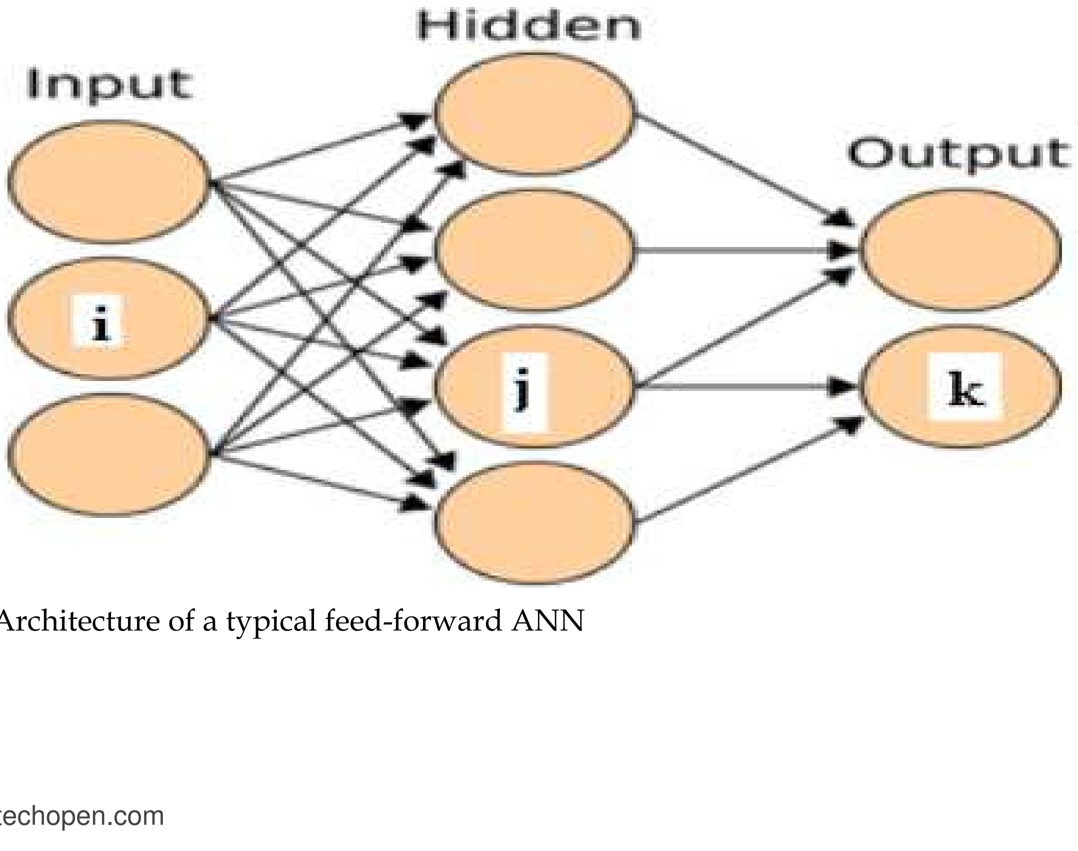

Fig. 1. Architecture of a typical feed-forward ANN

It has been proved in [1] that the sigmoid response to a sum of inputs is equivalent to combining the sigmoid response to each input using the fuzzy logic operator "ior" (interactive or). The truth table of the "ior" operator is shown in table 1. The truth table can be readily generalized to an arbitrary number of inputs. Eq.(1) can, thus, be interpreted as the output of a fuzzy inference system, where the weight w_{jk} is the action recommended by fuzzy rule *j*. However, this w_{jk} does not contribute directly to the ANN output. Instead, its contribution to the output is weighted by the sigmoid term $sig\left(\sum_{i=1}^{N_i} w_{hij} x_i^p + b_j\right)$.

The sigmoid term corresponds to the degree of firing of the rule, which judges to what extent rule 'j' should participate in the ANN final decision. Moreover, the inference is based on 'ior' rather than the product/'and' fuzzy operator used in ANFIS. It is clear from the 'ior' truth table that the 'ior' operator decides that a rule fully participate in the ANN final decision if all its inputs satisfy their corresponding constraints or if some of them does, while the others are neutral. On the other hand, it decides that the rule should not participate if one or some of the inputs do not satisfy their constraints, while the others are neutral. In the case that some of the inputs completely satisfy the constraints; while others completely violate them, the rule becomes neutral participating by half-weighted recommended action in the final ANN output. The mathematical expression for "ior" is as follows [1]:

$$ior(a_1, a_2, \ldots\ldots\ldots, a_n) = \frac{a_1 . a_2 \ldots\ldots\ldots\ldots a_n}{(1-a_1).(1-a_2)\ldots\ldots\ldots(1-a_n) + a_1 . a_2 \ldots\ldots\ldots a_n} \tag{2}$$

In linguistic terms, an antecedent formed by "ior-ing" several conditions, is equivalent to replacing the conventional phrase: "*if* A & B & --- then" with "So long as none of the conditions A, B, … are violated --- then". Throughout the paper we will use the Mnemonic "SLANCV" as a shortcut for this phrase. Thus we can say that Eq. 1 can be restated as a set of rules taking the following format:

$$SLANCV \quad x_i^p > -\left(b_j / N_i\right) / w_{hij} \quad then \quad o_{jk} = w_{jk}$$
$$i = 1,2,\ldots,N_i$$

Despite the successful deployment of the "ior" based rule extraction in several applications ([1], [6] and [7]), it has several disadvantages. For example, the weights and biases of a hidden neuron have no direct clear logical interpretation. This makes the incorporation of available knowledge difficult. Such knowledge is of great use in accelerating the ANN training procedure. Besides, leaving weights and biases values unconstrained often lead to some un-plausible rules (rules with impossible antecedent) that need pruning. Therefore, to overcome these disadvantages, our approach is to modify Eq.(1) as follows:

$$o_k = \sum_{j=1}^{N_h} w_{jk}^c sig\left(\sum_{i=1}^{N_i} w_{hij}^c (x_i^p - b_{ij}^c)\right) \tag{3}$$

where, w_{hij}^{c} are the weights joining input *i* to hidden neuron *j*, and w_{jk}^{c} are the weights joining hidden neuron *j* to output neuron *k*.

The superscript '*c*' denotes that these weights are constrained. In general, a constrained variable Par^{c} that directly appears in Eq. (3) is related to its corresponding free optimization variable *Par* by the following transformation:

$$Par^{c} = \left(Parmx - Parmn\right) sig\left(\frac{Par}{\max\left(Parmx,\left(Parmn\right)\right)}\right) + Parmn \tag{4}$$

where, $Parmx, Parmn$ are the maximum and minimum values of the parameter, respectively.

Comparing Eqs. (1) and (3), it is clear that our approach introduces two simple, yet effective, modifications:

- First, in Eq. (3), w_{hij} is taken as a common factor to the bracket containing the input and bias. Second, there is a bias corresponding to each input (b_{ij}). Using these two modifications, Eq.(3) has a simple direct fuzzy interpretation.

$$SLANCV \quad x_i^p > b_{ij}^c \left(if w^c{}_{hij} > 0\right), x_i^p < b_{ij}^c \left(if w^c{}_{hij} < 0\right) \quad then$$
$$o_{jk} = w^c{}_{jk}, where\ i = 1,2,\ldots,N_i$$

First Input	Second Input	IOR Output
0	0	0
0	0.5	0
1	0.5	1
0.5	0.5	0.5
1	0	0.5
0	1	0.5
1	1	1
0.5	1	1
0.5	0	0

Table 1. Truth Table of the IOR- Operator.

This direct interpretation makes it easy for the designer to incorporate available knowledge through appropriate weight initialization; as will be made clear in the adopted case study.

- The weights and biases included in Eq. (3) are constrained according to limits defined by the system designer. This ensures that the deduced rules are logically sound and consistent.

Furthermore, often, the nature of a problem poses constraints on the ANN output. Two possible approaches; are possible; to satisfy this requirement:

- Modifying Eq.(2) by replacing the sigmoid with a normalized sigmoid.

$$o_k = \sum_{j=1}^{N_h} w_{jk}^c \frac{sig\left(\sum_{i=1}^{N_i} w_{hij}^c (x_i^p - b_{ij}^c)\right)}{\sum_{j=1}^{N_h} sig\left(\sum_{i=1}^{N_i} w_{hij}^c (x_i^p - b_{ij}^c)\right)} \tag{5}$$

- Adding a penalty term to the objective function used in the ANN training so as to impose a maximum limit to its output.

In this research, we adopted the first approach.

To apply the proposed approach to a particular design problem, there are essentially three phases

1. Initialization and knowledge incorporation: In this phase, the designer defines the number of rules (hidden neurons) and chooses suitable weights and biases constraints.
2. Training phase.
3. Rule Analysis and Post-Rule-Analysis Processing: The weights are interpreted as fuzzy rules. A suitable method is used to analyse the rules and improve the system performance based on the insight gained from this rule analysis.

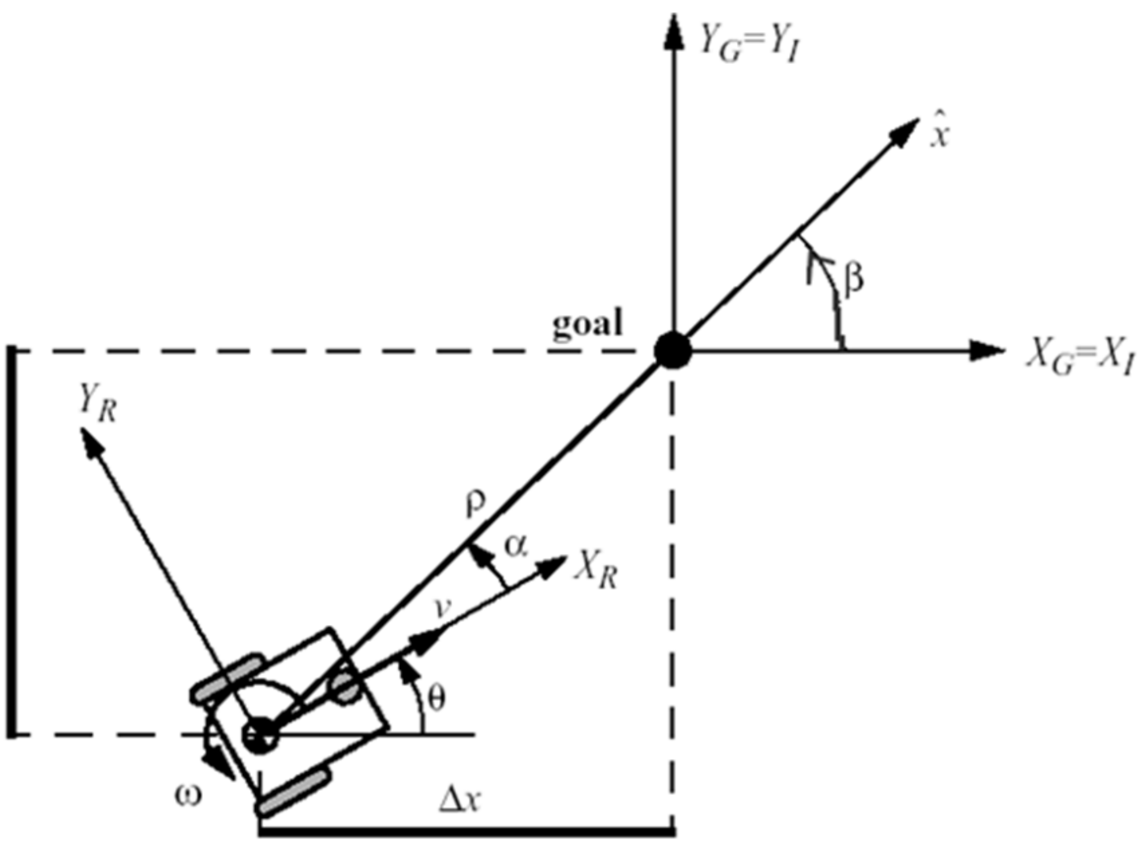

Fig. 2. Parameters used to describe the path tracking problem.

3. Case study: Robot path tracking

In this example, an ANN is adapted to assist a differential-wheel robot in tracking an arbitrary path. Fig. 2 illustrates the geometry of the problem. The robot kinematic model, the need for a closed loop solution, the choice of a suitable representation for the ANN's inputs and outputs as well as the initialization, training and interpretation of the weights of the obtained ANN, are discussed below.

3.1 The robot kinematic model

The kinematic model of the robot is described by the following equations:

$$\begin{aligned} \dot{X}_R &= v \quad \cos\theta \\ \dot{Y}_R &= v \quad \sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{6}$$

Where

$$\dot{X}_R, \dot{Y}_R, \theta, \omega$$

represent the horizontal, vertical components of the robot linear velocity v, its orientation angle and its angular velocity, respectively. Once a suitable control algorithm determines v and ω, it is straightforward to determine the corresponding v_r , v_l values by solving the following 2 simultaneous equations:

$$v = \frac{v_r + v_l}{2} \quad \omega = \frac{v_r - v_l}{l} \tag{7}$$

where l is the distance between the right and left wheels to yield:

$$v_r = \frac{2\text{v} + \omega l}{2}, v_l = \frac{2\text{v} - \omega l}{2} \tag{8}$$

It is clear from (7) that an average positive (negative) velocity indicates forward (reverse) motion along the robot current axis orientation. Similarly, it is clear from (8), that a positive (negative) difference between v_r and v_l indicates a rotate-left or counter-clockwise (rotate-right or clockwise) action. In case both wheels speeds are equal in magnitude and opposite in sign, the robot rotates in-place.

3.2 Open vs closed loop solutions of the path tracking problem

Consider a path with known parametric representation $\left(x(t), y(t)\right)$. In this case, the robot reference linear and angular velocities (denoted v_{ref} and ω_{ref}) can be calculated based on the known desired performance. For the robot to follow the desired path closely, its velocity should be tangent to the path. Thus the reference velocities can be computed using:

$$v_{ref} = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}$$
$$w_{ref} = \frac{d}{dt}\sin^{-1}\left(\frac{\dot{y}(t)}{v_{ref}}\right) \tag{9}$$

Applying the control inputs (v_{ref} and ω_{ref}) (or equivalently (v_r and v_l)) to the robot would enable it; in the absence of noise and other types of inaccuracies; to follow the required path. This open loop design will be called the "*direct forcing case*".

However, to assist the ANN in learning the concept of a path (not an instance of a path) as well as making it robust against disturbances, we need to find a closed-loop control law. This is not straightforward because the kinematics model is nonlinear. In what follows, our objective is to show how the proposed ANN-based framework can provide reliable closed-loop control; of the form shown in Fig. (3); compared to the direct forcing case. In Fig. (3), the role of the Robot kinematics simulator is to predict the robot location at the current time step given its current control inputs.

For the purpose of illustration, we will restrict our case study to the following family of paths:

$$x(t) = bt; 0 < b < 1 \quad \text{and} \quad y(t) = ct^3; -1 < c < 1$$

where 't' is the time vector= [0 (start time):0.05(time step):1 (final time)].

The ANN is trained on randomly chosen 11 members of this family and tested on different 11 members of the same family. To demonstrate the robustness of the proposed approach an additive disturbance (of uniform distribution) is added to both v and ω. The value of the disturbance can reach up to 200% of v value and 100% of ω value.

3.3 Choice of the ANN's inputs and outputs

Several possible input-output choices exist. The first has been reported in [11]. The time is the input and the speeds are the output. An alternative choice is to consider the coordinates (x, y) of each point on the path; as inputs; and the corresponding actions (speeds); as output. The third choice is to input the path as a whole as a single input vector and the corresponding sequence of actions (speeds) as a single output vector. All these choices share two fundamental disadvantages. First, it is impossible to interpret the trained weights as fuzzy rules. Furthermore, the ANN does not learn the "concept" of path tracking in general. Instead, it learns to track a single path only. In addition, the first and second choices do not explicitly capture the relation between consecutive path points. To overcome these limitations, we investigated different combinations of ANN inputs and outputs.

Only the two input-output combinations, that produced the best results, are discussed:

i. Case "A":

The inputs to the ANN are chosen to be:

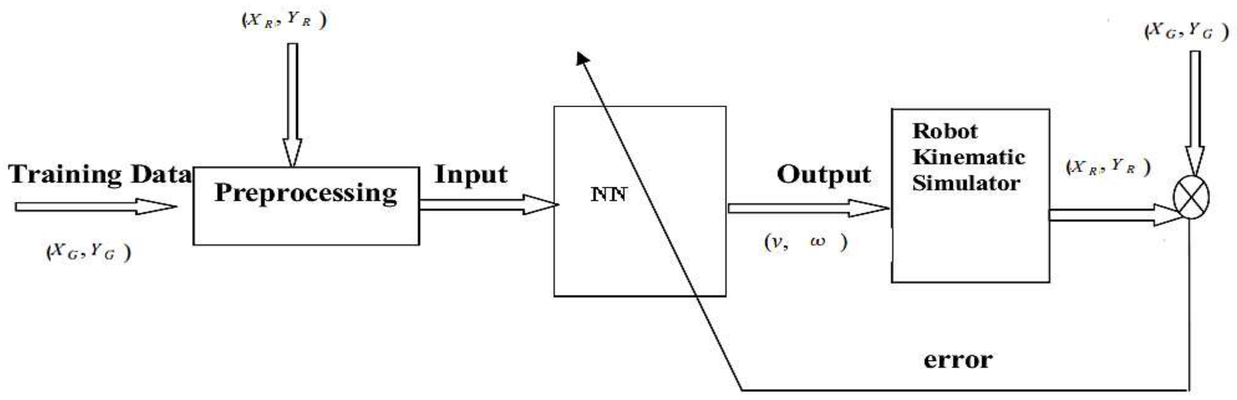

Fig. 3. A typical closed loop Block diagram for the flow of information in the Robot Path Tracking Problem

- the distance ' ρ ' to the goal point, $\rho = \sqrt{(X_G - X_R)^2 + (Y_G - Y_R)^2}$
- the angle ' α ' that the robot needs to rotate to orient itself in the direction of the goal.

Refering to Fig. (2), α is calculated using the following equations:

$$\beta = \tan^{-1}\left(\frac{Y_G - Y_R}{X_G - X_R}\right), \alpha = -\theta + \beta$$

- the deviations between the actual and reference robot linear and angular velocities $(v_{ref} - v_{rob})$ and $(\omega_{ref} - \omega_{rob})$

The outputs are the corrections (increase/ decrease) (v_{inc}, ω_{inc}) needed for (v, ω) to keep following the required path. Fig. 4 shows the block diagram of this system.

ii. Case "B"

The inputs and outputs are as in case A, with the following two additional inputs:

- the previous control inputs (v, ω).

Fig. 5 shows the block diagram of this system.

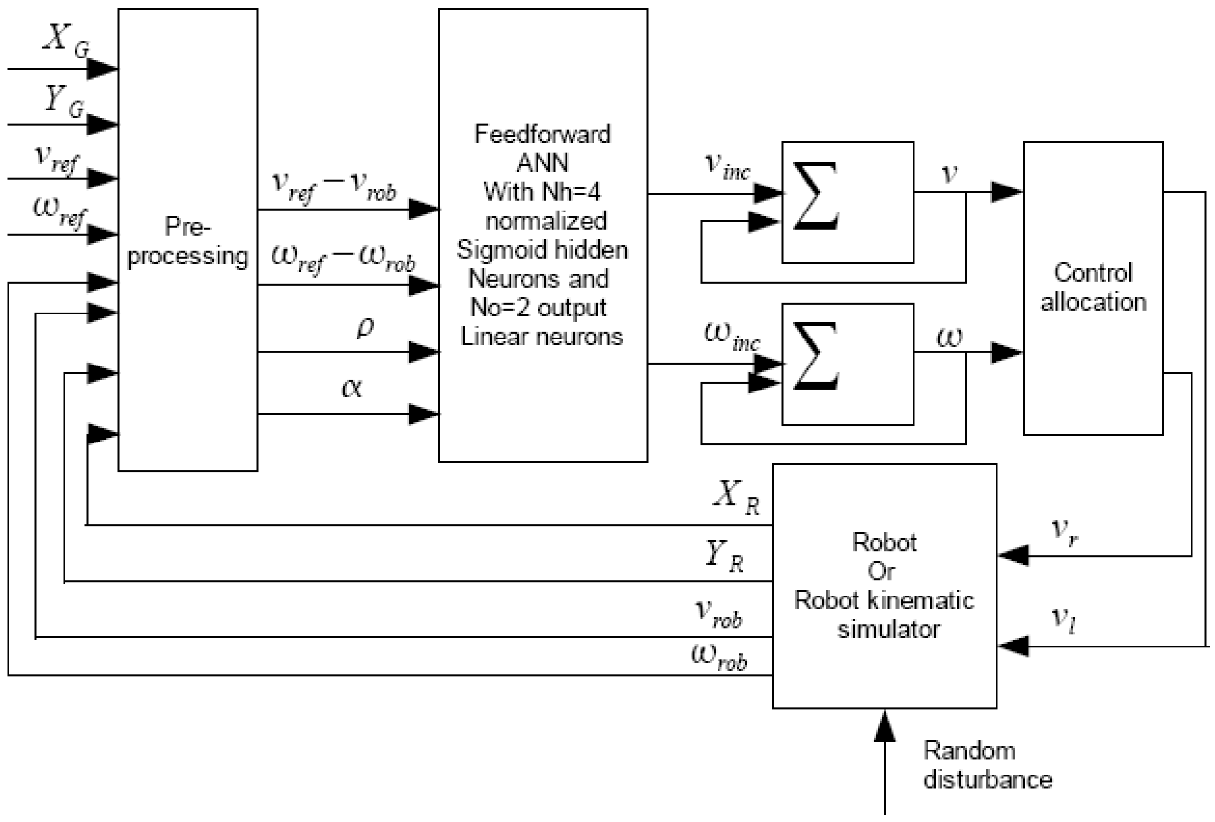

Fig. 4. Block diagram of the system in case study case A

3.4 Initialization and knowledge incorporation phase

Knowledge of the range of each input and output variables helps in the proper initialization of both the weights and biases. Recall that N_i is the number of inputs, N_o is the number of outputs, N_h is the number of hidden neurons (which corresponds to the number of rules used by the ANN in decision making). Biases are stored in a matrix B^c of dimensions N_i x N_h : Thus, each column of B^c corresponds to a certain rule. Each entry (row number, column number), in B^c, contains the threshold to which the corresponding input is compared in a particular rule (corresponding to that column). The weights connecting the inputs to the hidden neurons are stored in a matrix Wh^c of dimensions N_i x N_h . As before, each column corresponds to a certain rule. This time, however, the sign of the number in each entry (row number , column number), in Wh^c, controls the comparison operator with the corresponding threshold stored in B^c (negative corresponds to ”less than”, while positive corresponds to ”greater than”). Thus, each column from B^c together with the corresponding column from Wh^c determine the antecedent of a 'SLANCV' rule. The weights connecting the hidden neurons to the output neurons are stored in a matrix Wo^c of dimensions N_h x N_o . The numbers in each row of Wo^c indicate the outputs (consequents) suggested by a certain rule. Such an interpretation of the weights/bias matrices as rules antecedents/ consequents helps in selecting suitable initial values for the ANN's parameters as well as understanding its decisions after training. For example, at initialization, the values in each row of B^c should be constrained to lie within the range of possible values for this input. For instance, if the third input is ρ , then the entries in the third row of B^c should all be positive. Similarly, if the first output is the velocity correction v_{inc} then the first column of Wo^c can take both negative /positive small values to allow the ANN to increase or decrease the robot speed while avoiding instability.

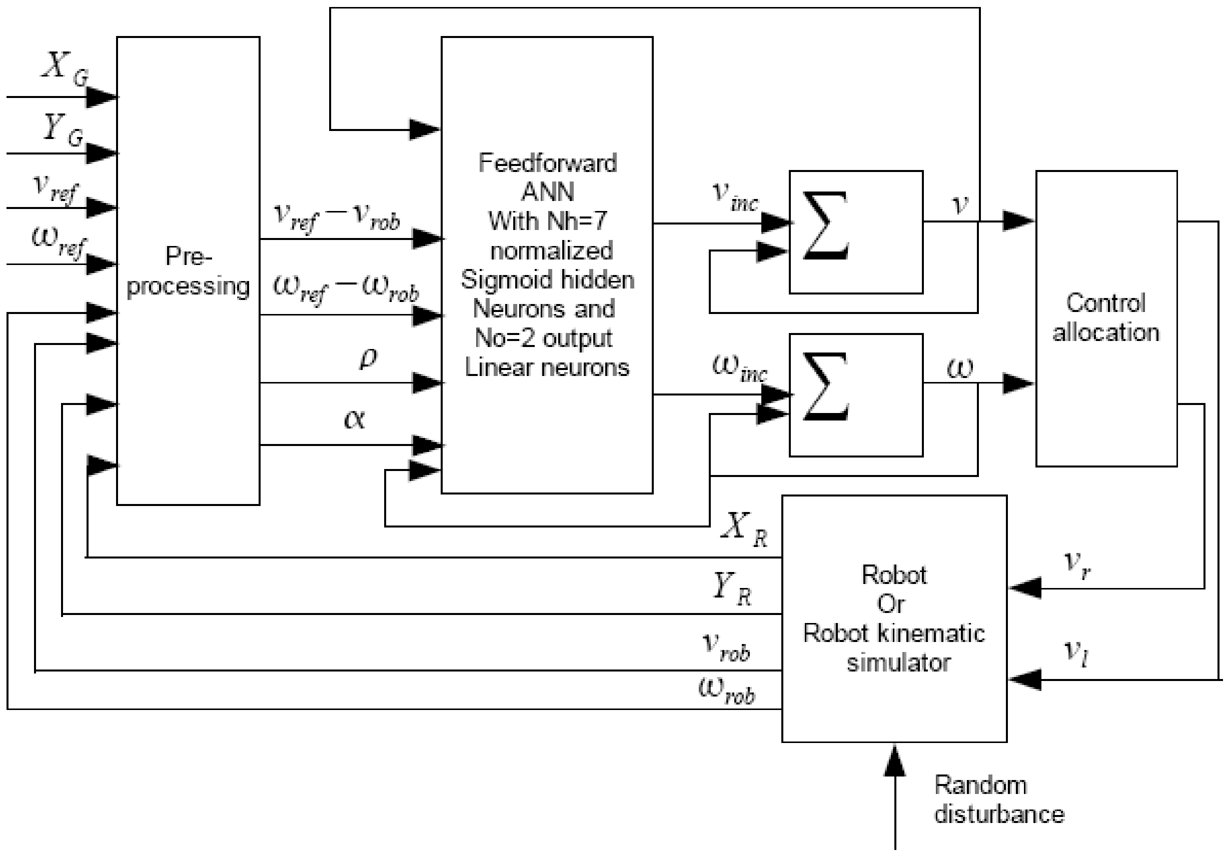

Fig. 5. Block diagram of the system in case study case B

3.5 Training phase

To train the ANN, the following objective function is minimized:

$$objg = \frac{1}{N_p} \sum_{p=1}^{N_p} \sum_{i=1}^{K_i} \rho_i^p \tag{10}$$

where ' ρ_i^p ' is the distance between the desired location on the pth path and the corresponding robot's actual location at the ith time instant, K_i is the number of points along the path to be tracked and N_p is the number of paths used in training. Since the objective function is not a closed form function (in the ANN weights and biases) and can only be computed through a simulator, a numerical optimization algorithm is used [10], [11], where gradients are computed using a numerical version of the Broyden–Fletcher–Goldfarb–Shanno (*BFGS*) optimization algorithm. The gradients are calculated numerically using the following formula:

$$\frac{\partial Objg}{\partial Par_i} = \frac{Objg(Par_i + \varepsilon) - Objg(Par_i - \varepsilon)}{2\varepsilon} \tag{11}$$

where ε is a very small number, typically $\varepsilon = 10^{-11}$. Par_i is the ith element of the vector Par

For case 'A', the number of adjustable parameters is 2* (4 (inputs) * 4 (hidden neurons/ rules)) + 4 (hidden neurons)* 2 (outputs), which equals 40 parameters. For case B, the number of adjustable parameters is 2* (6 (inputs) * 7 (hidden neurons/ rules)) + 7 (hidden neurons)* 2 (outputs), which equals 98 parameters . Figs. (6 and 7) show the results for cases 'A' and 'B', respectively.

It is clear that case 'B' is more robust to disturbance. However, both cases clearly outperform the open loop (direct forcing) case.

The fact that case 'B' produced better results is to be expected. Certainly, feedback provides memory to the system and helps in accumulating knowledge, which is an essential aspect of learning. We tried two types of feedback; direct feedback in which the ANN learns the link between its own outputs (v_{inc}, ω_{inc}) and the errors in performance and indirect feedback in which the ANN learns the link between a certain action (v, ω) and the errors in performance. The second memory type produced better results. Appendix A compares these results obtained using our approach with those obtained using conventional ANN [1,6,7] and ANFIS [8].

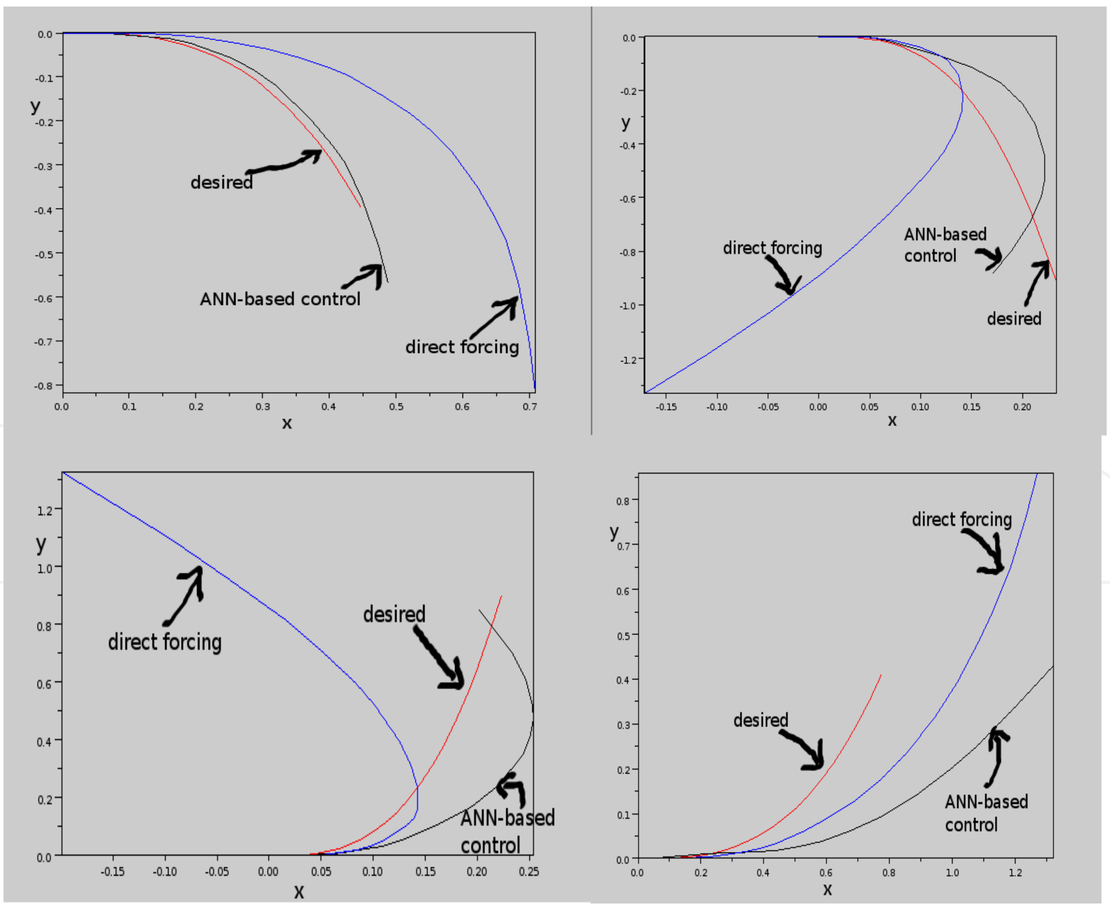

Fig. 6. Results for case A. It is clear that closed loop ANN-based control is more robust to disturbances than open-loop direct forcing

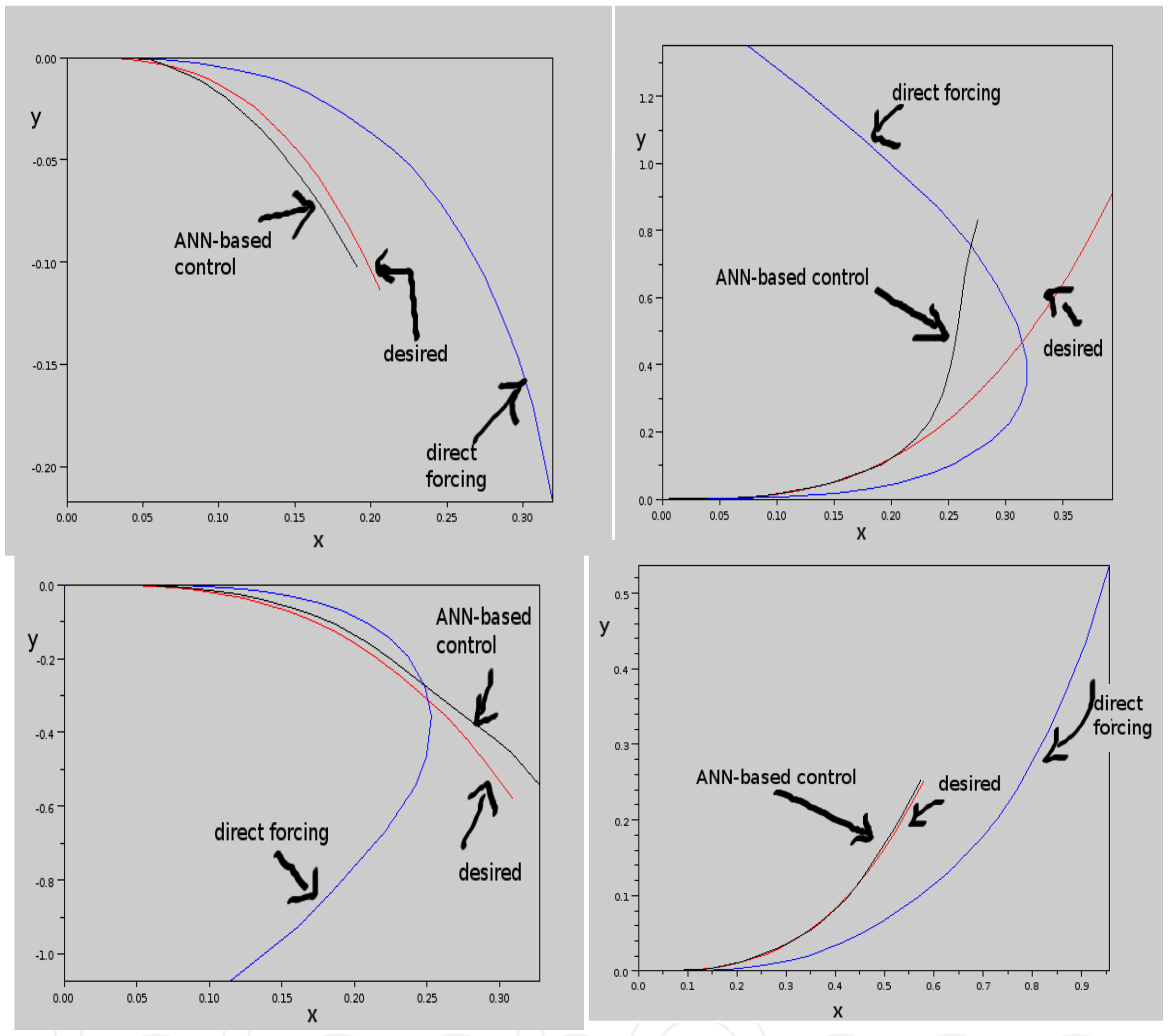

Fig. 7. Results for case B. It is clear that closed loop ANN-based control is more robust to disturbances than open-loop direct forcing. It is evident that adding to the ANN inputs the control actions at the previous step improved the results compared to those of case A.

3.6 Rule extraction, analysis and post-processing

Following the guidelines given in section 3.4, rule extraction from the trained weights and biases becomes straightforward.

Fig. (8) illustrates how the rules, for case 'A', have been extracted. The discovered rule-base is summarized as follows :

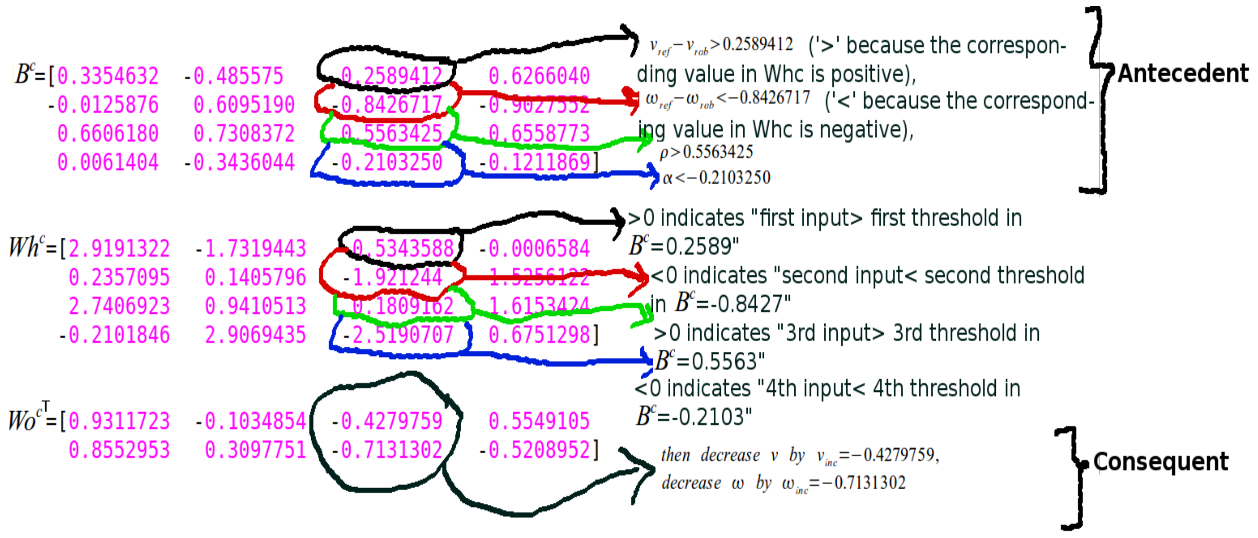
B^c=[0.3354632 -0.485575 0.2589412 0.6266040
-0.0125876 0.6095190 -0.8426717 -0.9027552
0.6606180 0.7308372 0.5563425 0.6558773
0.0061404 -0.3436044 -0.2103250 -0.1211869]
Wh^c=[2.9191322 -1.7319443 0.5343588 -0.0006584
0.2357095 0.1405796 -1.921244 1.5256122
2.7406923 0.9410513 0.1809162 1.6153424
-0.2101846 2.9069435 -2.5190707 0.6751298]
Wo^{cT}=[0.9311723 -0.1034854 -0.4279759 0.5549105
0.8552953 0.3097751 -0.7131302 -0.5208952]
$v_{ref}-v_{rob}>0.2589412$ ('>' because the corresponding value in Whc is positive),
$\omega_{ref}-\omega_{rob}<-0.8426717$ ('<' because the corresponding value in Whc is negative),
$\rho>0.5563425$
$\alpha<-0.2103250$
Antecedent
>0 indicates "first input> first threshold in B^c=0.2589"
<0 indicates "second input< second threshold in B^c=-0.8427"
>0 indicates "3rd input> 3rd threshold in B^c=0.5563"
<0 indicates "4th input< 4th threshold in B^c=-0.2103"
then decrease v by $v_{inc}=-0.4279759$,
decrease ω by $\omega_{inc}=-0.7131302$
Consequent

Fig. 8. An illustration of the rules extraction from weights and biases matrices.

$$SLANCV\quad v_{ref}-v_{rob}>0.3354632,\quad \omega_{ref}-\omega_{rob}>-0.0125876,\quad \rho>0.6606180,\quad a<0.0061404$$
$$then\quad increase\quad v\quad by\quad v_{inc}=0.9311723,\quad increase\quad \omega\quad by\quad \omega_{inc}=0.8552953$$
$$SLANCV\quad v_{ref}-v_{rob}<-0.485575,\quad \omega_{ref}-\omega_{rob}>0.6095190,\quad \rho>0.7308372,\quad a>-0.3436044$$
$$then\quad decrease\quad v\quad by\quad v_{inc}=-0.1034854\quad increase\quad \omega\quad by\quad \omega_{inc}=0.3097751$$
$$SLANCV\quad v_{ref}-v_{rob}>0.2589412,\quad \omega_{ref}-\omega_{rob}<-0.8426717,\quad \rho>0.5563425,\quad a<-0.2103250$$
$$then\quad decrease\quad v\quad by\quad v_{inc}=-0.4279759,\quad decrease\quad \omega\quad by\quad \omega_{inc}=-0.7131302$$
$$SLANCV\quad v_{ref}-v_{rob}<0.6266040,\quad \omega_{ref}-\omega_{rob}<-0.9027552,\quad \rho>0.6558773,\quad a>-0.1211869$$
$$then\quad increase\quad v\quad by\quad v_{inc}=0.5549105,\quad decrease\quad \omega\quad by\quad \omega_{inc}=-0.5208952$$

Similarly, the rule-base for case 'B' is summarized as follows:

$$SLANCV\quad v>1.1800783,\quad \omega<0.5797996,\quad v_{ref}-v_{rob}>0.3283071,\quad \omega_{ref}-\omega_{rob}>-0.1895662,$$
$$\rho>1.2352848,\quad a>-0.3088343\quad then$$
$$increase\quad v\quad by\quad v_{inc}=0.6156011\quad increase\quad \omega\quad by\quad \omega_{inc}=0.8038080$$
$$SLANCV\quad v>1.258573,\quad \omega>0.817993,\quad v_{ref}-v_{rob}<-0.4339714,\quad \omega_{ref}-\omega_{rob}>0.0552364,$$
$$\rho>1.246724,\quad a>0.2514984\quad then$$
$$decrease\quad v\quad by\quad v_{inc}=-0.6117282,\quad increase\quad \omega\quad by\quad \omega_{inc}=0.4437878$$
$$SLANCV\quad v<0.2235206,\quad \omega>0.7800814,\quad v_{ref}-v_{rob}<-0.7777185,\quad \omega_{ref}-\omega_{rob}<-0.7120344,$$
$$\rho>1.5151515,\quad a<-0.0003327\quad then$$
$$decrease\quad v\quad by\quad v_{inc}=-0.4927100,\quad decrease\quad \omega\quad by\quad \omega_{inc}=-0.0680181$$
$$SLANCV\quad v>1.4049258,\quad \omega<-0.3766352,\quad v_{ref}-v_{rob}<0.0756778,\quad \omega_{ref}-\omega_{rob}<-0.7105109,$$
$$\rho<0.1169906,\quad a<-0.2358675\quad then$$
$$decrease\quad v\quad by\quad v_{inc}=-0.2173556,\quad decrease\quad \omega\quad by\quad \omega_{inc}=-0.3592529$$
$$SLANCV\quad v<0.9930632,\quad \omega>0.8202830,\quad v_{ref}-v_{rob}>-0.1180958,\quad \omega_{ref}-\omega_{rob}<-0.6506539,$$
$$\rho>0.1156700,\quad a<0.2116636\quad then$$
$$increase\quad v\quad by\quad v_{inc}=0.3642978\quad decrease\quad \omega\quad by\quad \omega_{inc}=-0.8061992$$

$$SLANCV \quad v < 0.7807096, \quad > 0.4849204, \quad v_{ref} - v_{rob} > 0.4796375, \quad \omega_{ref} - \omega_{rob} > 0.5922044,$$
$$\rho > 1.2921157, \quad a > 0.2758564 \quad then$$
$$increase \quad v \quad by \quad v_{inc} = 0.3424317, \quad increase \quad \omega \quad by \quad \omega_{inc} = 0.8207999$$
$$SLANCV \quad v < 0.5131913, \quad \omega > -0.2240559, \quad v_{ref} - v_{rob} < -0.4493099, \quad \omega_{ref} - \omega_{rob} > 0.7691425,$$
$$\rho < 1.0904348, \quad a > 0.2174491 \quad then$$
$$decrease \quad v \quad by \quad v_{inc} = -0.5255254, \quad increase \quad \omega \quad by \quad \omega_{inc} = 0.8943484$$

In order to improve system performance and remove any inconsistencies, the rules above must be analysed. The following rule analysis procedure is limited to case 'B'. This procedure is assisted by a plot of the DOF (Degree-Of-Firing) of each rule (output of sigmoid) versus time. Such plots help in visualizing which rules the ANN is applying at each time instant and judging the decision/ performance made at this particular time. In particular, it is highly useful to identify dominant rules (rules having relatively high outputs) at the time a wrong decision is made. Correction is then possible by retraining the ANN keeping all rules fixed except the faulty dominant one. For example, as shown in Fig. 9, at the time the deviation from the desired path becomes maximum, rule 5 is dominant followed by rule 4. Accordingly, three different strategies, have been attempted, to retrain the ANN, to improve its performance. In all three strategies, all rules have been kept fixed except:

- for the first strategy: rule 5 (5th column of Wh^c ,5th column of B^c and the 5th row of Wo^c),
- for the second strategy: both the degrees of firing of rules 4 and 5 (4th and 5th column of Wh^c), and their then part (4th and 5th row of Wo^c),
- for the third strategy: the degree of firing of rule 5 and its then part (5th column of Wh^c and the 5th row of Wo^c), with the addition of a new rule (rule 8) whose SLANCV part is the same as that of rule 5 (8th column of B^c is same as its 5th column) but whose DOF (8th column of Wh^c) and consequent part (or then part) (8th row of Wo^c) are to be determined by the training algorithm. The idea is to insert a new rule that co-fires with the malfunctioning rule to provide some correcting action. For our case study, this last strategy gave us the best results. The results after retraining are shown in Fig. 10.

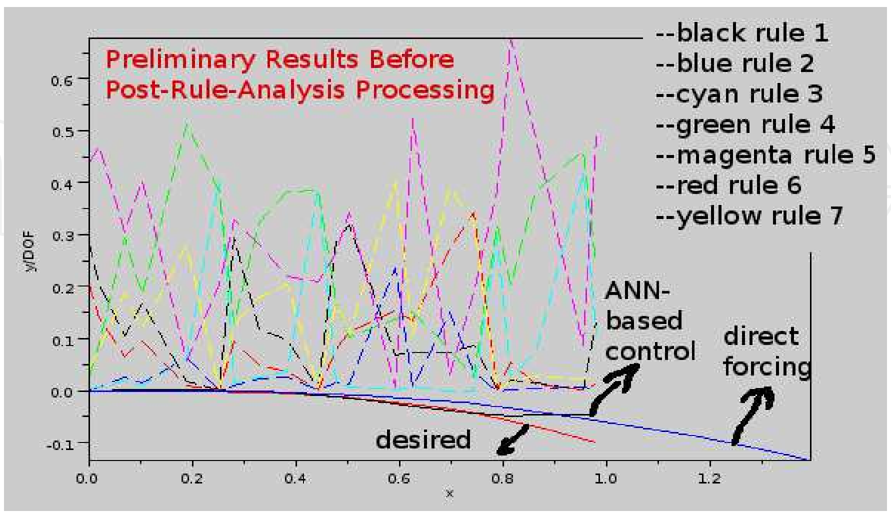

Fig. 9. Preliminary results of case B before post-rule analysis processing. The dashed curves represent the degree of firing of each rule with time. It is clear that rule 5 is the dominant rule at the time the deviation from the desired path became maximum.

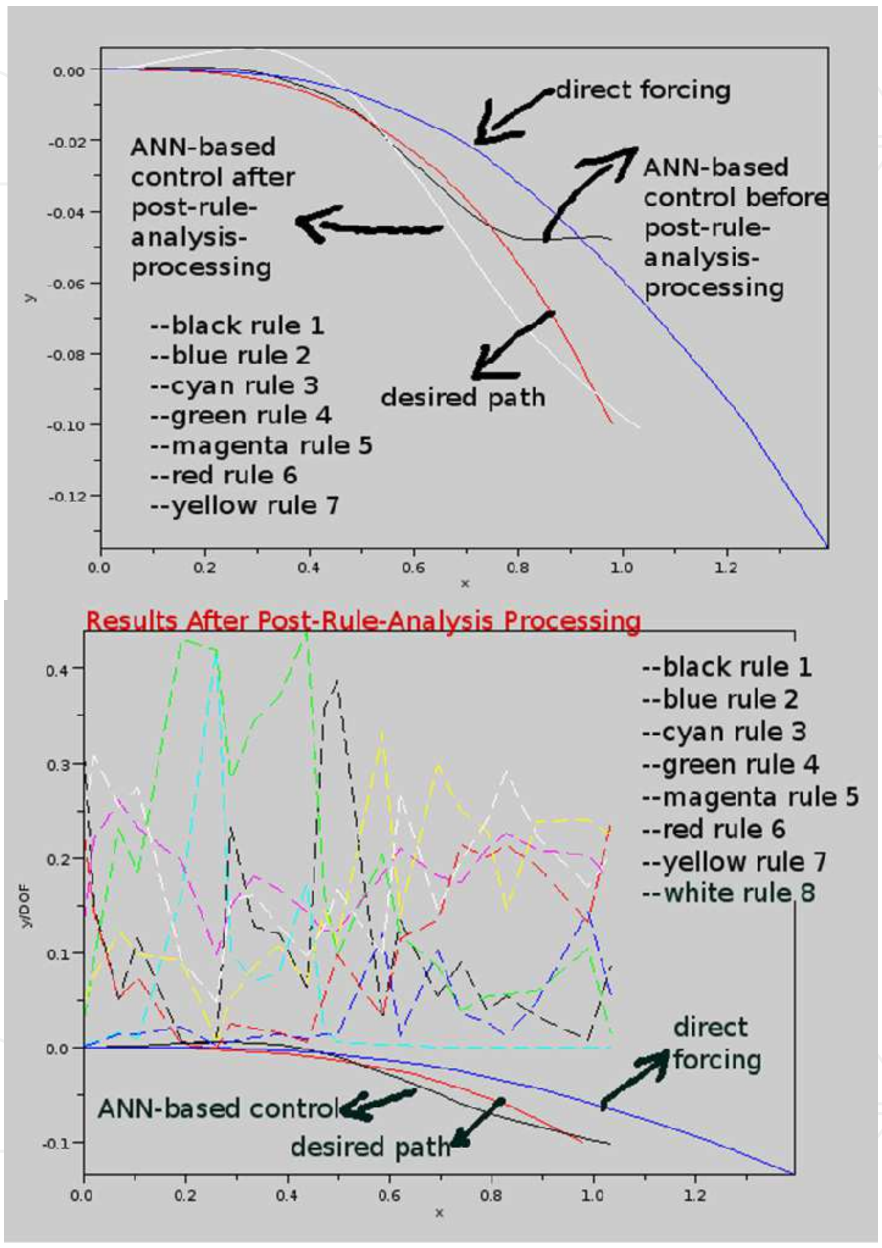

Fig. 10. Results of case B after retraining with only the parameters corresponding to rules 5 degree of firing and its "then" part allowed to vary in addition to adding a new rule 8 with the same SLANCV part as rule 5 to co-fire with it and to provide corrective action. It is clear that this retraining improved the results.

3.7 Discussion

Although the rule analysis method presented, helped in improving the results, it still needs improvement. In fact, rule analysis is a rather complex task. Understanding the logic behind the rules (why each rule recommends a particular consequent given a certain antecedent) is not simple for the following reasons:

1. Rules operate in parallel and collectively. Each rule recommends a certain consequent depending on the context of the neighbouring rules. Collectively, the overall decision helps achieving the desired objective.
2. According to the training algorithm, rules recommend certain consequents in order to minimize the desired objective function which is a function of the robot kinematics.
3. Rules are derived through batch (off-line) training. Therefore, the overall objective function is minimized over time and not at a particular instant (rules are derived based on a global point of view). Therefore, a rule may not sound reasonable to employ at a certain moment. This what makes it necessary to train an ANN over a certain family of curves. Different families of curves are expected to require different global rules.
4. Rule analysis is a trial and error process. Its complexity is proportional to the dimensionality (number of independent variables).

4. Directions for future research: Solving general optimization problems

The proposed approach can be applied to general optimization problems and not just path-tracking. From an abstract point of view, any optimization problem can be mapped to a goal-tracking problem in which:

- The goal point is the desired performance or desired objective function value.
- The input is the absolute difference between the objective function value at the current solution and the desired objective function value.
- The output is the correction, to the current solution, recommended by the ANN. This is fed-back as an input to the ANN at the next iteration.
- Time evolution corresponds to iterations.

Hence, over the different iterations the ANN is expected to suggest a sequence of corrections that helps in approaching the required objective function value.

This approach to optimization is expected to be less prune to local minima trapping, provides better insight into the nature of the investigated problem and can easily deal with multi-objectives/ errors optimization. For example, in the robot path tracking problem, $\alpha, \rho, v_{ref} - v_{rob}, \omega_{ref} - \omega_{rob}$ were error measures that, ideally are required to be all zeros. In this case, the robot perfectly tracks the desired path and remains tangent to it at all times. This property (being tangent at all times) is desirable, from a practical point of view, because even if the robot path is close to the desired path but with too many frequent changes of orientation, wear out will occur to the robot parts and it will be questionable whether the robot can, practically, makes these moves).

4.1 Illustrative example: Minimizing the camel benchmark function

The camel objective function is defined as follows

$$f_{Sixh}\left(x_1,x_2\right)=\left(4-2.1x_1^{\ 2}+\frac{x_1^{\ 4}}{3}\right)x_1^{\ 2}+x_1x_2+\left(-4+4x_2^2\right)x_2^2;-3\le x_1\le 3,-2\le x_2\le 2.$$

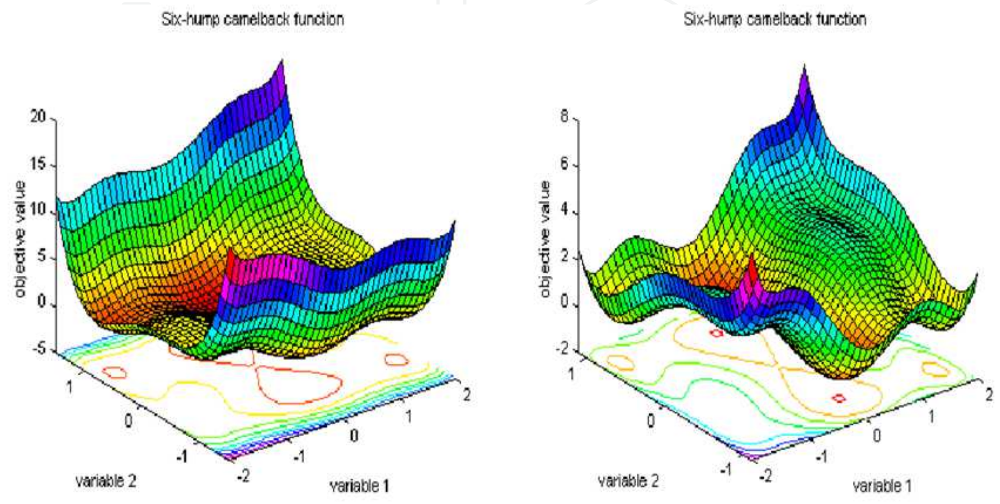

Fig. 11. Camel function objective function.

The global minimum is $f\left(x_1,x_2\right)=-1.0316;\left(x_1,x_2\right)=\left(-0.0898,0.7126\right),\left(0.0898,-0.7126\right)$. Fig. 11 shows a plot of the camel objective function.

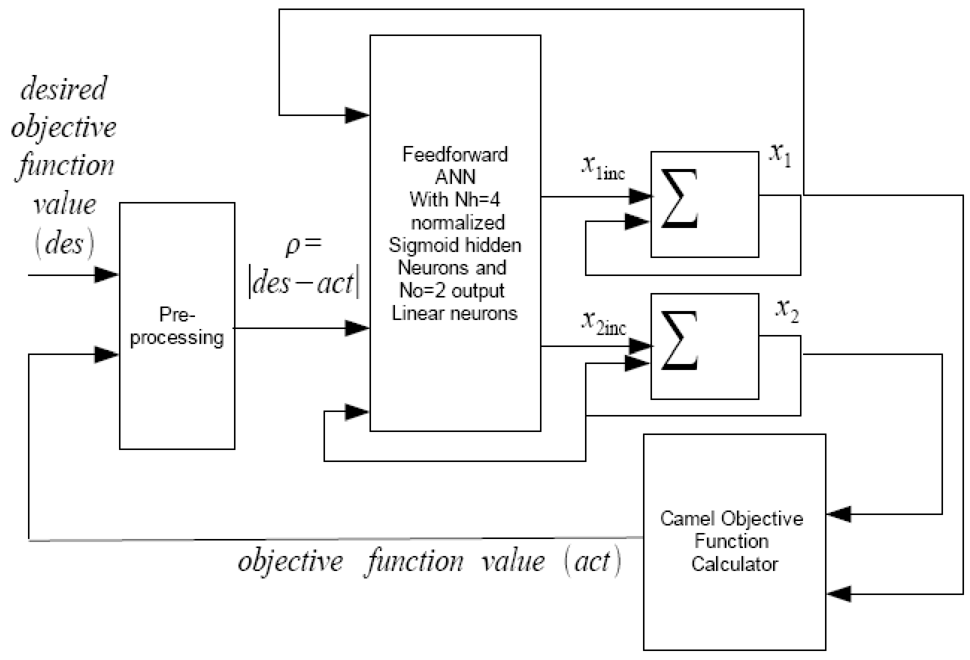
desired objective function value (des)
Pre-processing
ρ= |des−act|
Feedforward ANN With Nh=4 normalized Sigmoid hidden Neurons and No=2 output Linear neurons
x1inc
x2inc
Σ
Σ
x1
x2
Camel Objective Function Calculator
objective function value (act)

Fig. 12. Block diagram of the system used to find the global minimum of the camel objective Function.

To apply the proposed ANN-based approach to find the global minimum of this function, the closed loop system in Fig. 12 has been adopted. The inputs to the ANN are the values of x_1 and x_2 at the previous iteration, the absolute difference between the desired value of the objective function at the current iteration and its current actual value. The outputs of the ANN are the recommended corrections to x_1 and x_2 (increase or decrease). The ANN weights and biases were estimated numerically as before such that the ANN minimize the camel objective function. The desired value for the objective function at any iteration is chosen to be slightly less than its current value (For example, for decrratio=0.85, the desired objective function at a certain iteration =decrratio* the actual objective function value corresponding to the current solution). This idea is borrowed from [13], where it has been recommended to be adopted with any optimization method. The ANN has been trained, as before, using a numerical BFGS algorithm, where the objective function to be minimized is defined as :

$$Objg = \rho_F \tag{12}$$

where, ρ_F is the absolute difference between the desired objective function value and its actual value at the last iteration. The rules extracted from the trained ANN are as follows:

$$\begin{gathered}
SLANCV \quad x_1 > 0.6755704, \quad x_2 > 1.5209156, \quad \rho > 0.0161157, \\
decrease \quad x_1 \quad by \quad x_{1\text{inc}} = -0.0668506 \quad decrease \quad x_2 \quad by \quad x_{2\text{inc}} = -0.0203858 \\
SLANCV \quad x_1 > 1.4023561, \quad x_2 > 0.4688876, \quad \rho > 0.1058053, \\
increase \quad x_1 \quad by \quad x_{1\text{inc}} = 0.0617143 \quad decrease \quad x_2 \quad by \quad x_{2\text{inc}} = -0.0195689 \\
SLANCV \quad x_1 > 0.6147782, \quad x_2 < 0.1270187, \quad \rho < 1.504679, \\
increase \quad x_1 \quad by \quad x_{1\text{inc}} = 0.0301060 \quad decrease \quad x_2 \quad by \quad x_{2\text{inc}} = -0.0128413 \\
SLANCV \quad x_1 > 0.0459426, \quad x_2 > 0.3802873, \quad \rho < 0.5840457, \\
decrease \quad x_1 \quad by \quad x_{1\text{inc}} = -0.0279751 \quad decrease \quad x_2 \quad by \quad x_{2\text{inc}} = -0.0338863
\end{gathered}$$

Fig. 13 shows the results (the objective function value versus iteration number). Clearly, the ANN-based optimization technique found the global optimal (The initial solution was $x_1 = -2.5, \quad x_2 = 2.5$). When using the BFGS technique to minimize the camel objective function, with the same learning rate, the algorithm completely diverged. For a lower learning rate, direct BFGS reached the global optimal. However, ANN-based optimization offers greater promise for higher dimensions/ multi-objective/ error problems and provides insight into the solution through the analysis of the derived rules. A key reason for the robustness of the ANN-based optimization over direct optimization is that it explores the objective function and derives problem-dependent heuristics (as opposed to meta-heuristics).

An important direction for future investigation is the use of concepts borrowed from "robust optimization" to enhance the quality of the rules extracted from the ANN. Robust optimization concepts can be applied at two different levels:

- At the action level, we can include noise during ANN training. The error in path-tracking can be defined as a function of the difference between the average expected location and the desired location as well as the variance (or standard deviation of this error). This will make the ANN develop a decision strategy that is more prudent and is less likely to cause divergence from the desired path, in case of disturbance.

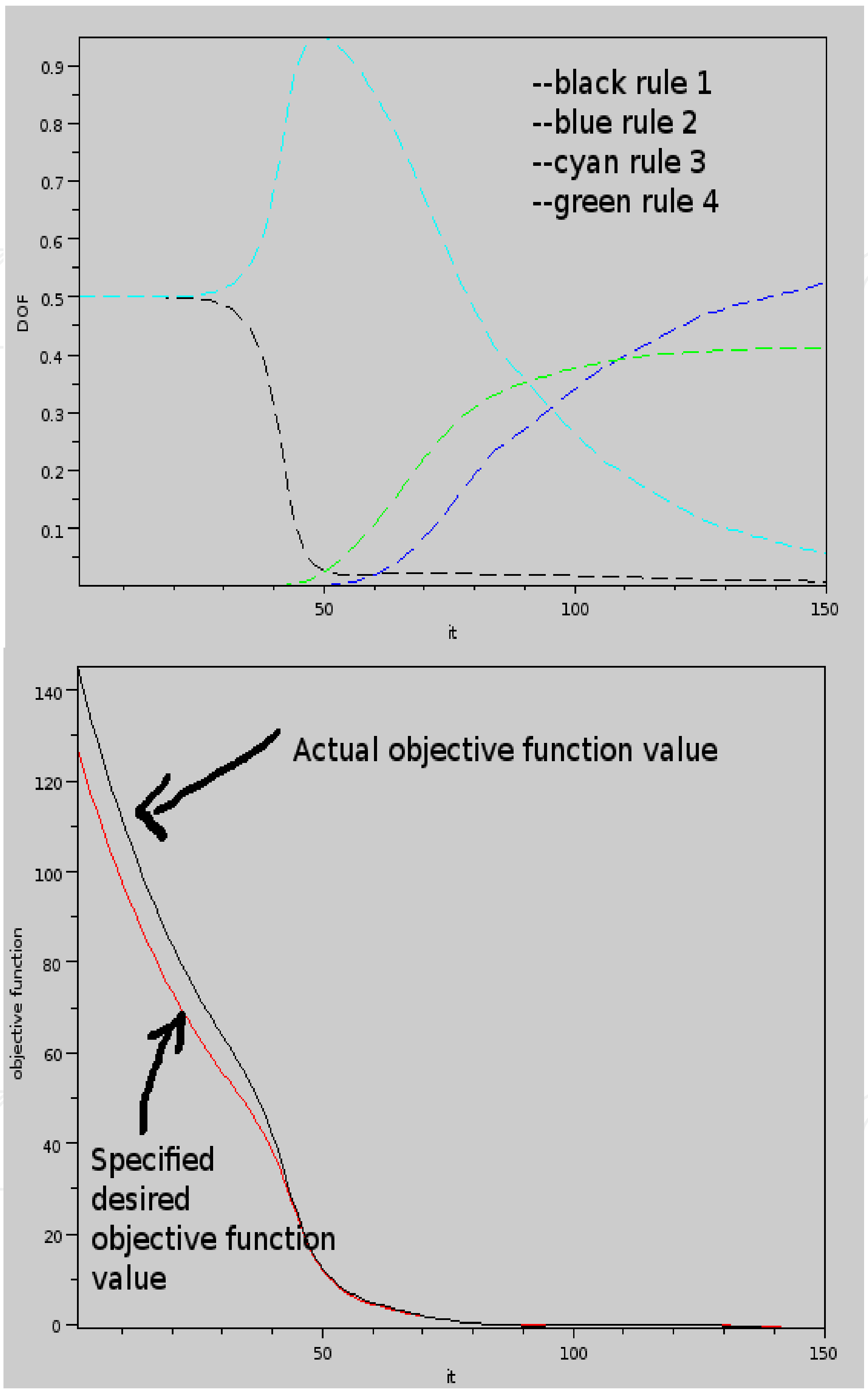

Fig. 13. The results of using the proposed approach to find the global minimum of the camel objective function. The top figure shows the degree of firing of the different rules versus iteration number. The bottom figure depicts the objective function value versus iteration number. The approach successfully converged to the global minimum.

- At the weights/biases level, robust optimization can be used to develop robust rules. Rules are robust when they are reachable from different weight initializations (i.e. they are not sensitive to a particular initialization) and lead to acceptable performance when subject to small perturbations.

5. Conclusion

We believe that merging symbolic AI (logic) with non-symbolic AI (ANNs) through our new proposed framework can achieve the following advantages:

1. The resulting learning system is transparent to the user and its reliability can be easily assessed. A suitable strategy has been outlined for improving its reliability based on the analysis of the extracted rules.
2. The system is robust to noisy inputs and disturbances.
3. The logic-based approach to optimization can be less prune to local minima trapping.
4. The approach is applicable to a broad class of engineering problems where the corresponding correct output to a certain example input is not necessarily available (but a means of assessing the fitness of the output is available through simulation or practical measurements).

We do not claim that the proposed approach can outperform existing approaches in all problems, however, we can certainly claim that we offered researchers, a framework truly worthy of investigation for complex optimization, control and design problems. The best approach will always remain problem-dependent which is the charm and challenge of engineering optimization.

6. Acknowledgement

After thanking Allah almighty for giving the authors the stimulus required to complete this work, the first author would like to thank her students for their valuable inquiries during a summer course on ANNs that she gave. Their eagerness to understand, queries and continuous criticisms helped her a lot to better formulate her thoughts and served as a valuable encouragement, especially during the early part of this work. It is true that, at many occasions, we learn from our students even more than we teach them! The second author acknowledges the support of EJUST.

7. Appendix A

In this appendix, we apply the conventional ANN formulation ([1] ,[6] ,[7]) and ANFIS [8] to case 'B' of the adopted case study. It is noteworthy that strict application of the conventional ANN or ANFIS to this case study is not possible because the desired system output is unknown (it is not possible to evaluate the objective function except by using the robot simulator). Therefore, the numerical BFGS has been used, as before, in the training phase with the same objective function defined in Eq. (9). As said earlier, with the conventional ANN formulation the output of the ANN described in Eq. (1) can be interpreted as fuzzy rules of the form:

$$SLANCV \quad x_i^p > -\left(b_j / N_i\right) / w_{hij} \quad then \quad o_{jk} = w_{jk}$$
$$i = 1,2,\ldots,N_i$$

The fact that the antecedent of a rule depends on both the bias of the corresponding hidden neuron and the weights from inputs-to this hidden neuron makes it difficult the use of known inputs constraints in weights/biases initialization. Thus, we are forced to use small random weights and biases initially, train the ANN, extract the rules and then re-train in case some of the rules yield un-plausible antecedents. For our case study, 3 out of the 7 rules had un-plausible antecedents. For example, one of the rules stated:

$$\begin{gathered} SLANCV \quad v < -1.1959689, \quad w > 0.1675315, \quad v_{ref} - v_{rob} > 0.8878251, \quad \omega_{ref} - \omega_{rob} > 0.1353403, \\ \rho < -2.7918388, \quad a < -0.2060677, \\ decrease \quad v \quad by \quad v_{inc} = -6.6637447, \quad decrease \quad \omega \quad by \quad \omega_{inc} = -5.8271221 \end{gathered}$$

Clearly comparing ρ to a negative threshold is not logical.

Fig. 14 illustrates a typical ANFIS architecture for the case of a 2 inputs $(x_{1,}x_2)$ single output, 2 rules example. A_{ij} is the membership function of the ith input in the jth rule. The DOF of a rule is computed using the 'Product' operator, i.e. it is the product of the output of the membership functions of a certain rule (Layer 2). NORM units (Layer 3) divides the individual DOF of each rule by the sum of DOF of all rules to produce a normalized degree of firing $\overline{w}_j$. Layer 4 computes the consequent of each rule j for each output k, f_{jk} as a function of the inputs $f_{jk} = \sum_{i=1}^{Ni} p_{ijk} + r_{jk}$. The overall system output is computed as a weighted sum of the different rules consequents ($f_k = \sum_{j=1}^{N_h} \overline{w}_j f_{jk}$, N_h is the number of rules which equals 2 in Fig. 14). Similarly, to be able to compare our approach to ANFIS [8], we use the same block diagram given in Fig. 5 but replacing the typical feed-forward ANN with an ANFIS. The ANFIS formulation does not impose restrictions on membership function's choices. Therefore, sigmoid membership functions have been chosen, for the purpose of comparison with our approach. In this case, the membership function of the jth rule takes the form:

$$A_{ij} = \mathrm{sig}\left(a_{ij}\left(x_i - c_{ij}\right)\right)$$

Our approach can be viewed as a modified ANFIS system with the '*Product*' operator replaced by the 'ior' operator and with $p_{ijk} = 0$. As indicated by the results (Fig. 15), these modifications enhance the performance considerably. ANFIS training involves the estimation of the parameters p_{ijk}, r_{jk} for each rule contributing to the output as well as the membership functions parameters a_{ij}, c_{ij} of each rule. The extracted rules after training are as follows:

If $v < 0.7060525$ and $\omega < -0.2171979$ and $(v_{ref} - v_{rob}) > -0.4880470$ and $(\omega_{ref} - \omega_{rob}) < -0.3037414$ and $\rho < 0.5722985$ and $\alpha < -0.2705797$ then $v_{inc} = -0.4977126 * v + -0.2252019 * \omega + -0.2252019 * (v_{ref} - v_{rob}) + 0.6536675 * (\omega_{ref} - \omega_{rob}) + -0.9451361 * \rho + 0.4929671 * \alpha + 0.5642881 + 0.2698283,$

$\omega_{inc} = -0.2601446 * v + -0.0526525 * \omega + -0.0526525 * (v_{ref} - v_{rob}) + -0.0256069 * (\omega_{ref} - \omega_{rob}) + 0.2929497 * \rho + -0.6628894 * \alpha + 0.6927095 + 0.5608534$

If $v > 0.9536605$ and $\omega < -0.3640576$ and $(v_{ref} - v_{rob}) > 0.3635925$ and $(\omega_{ref} - \omega_{rob}) > -0.5739730$ and $\rho < 1.0224728$ and $\alpha > 0.2518629$ then $v_{inc} = -0.2252019 * v + 0.5305053 * \omega + 0.5305053 * (v_{ref} - v_{rob}) + 0.5254107 * (\omega_{ref} - \omega_{rob}) + 0.6691905 * \rho + 0.1321803 * \alpha + -0.4524829 + 0.5491040,$

$\omega_{inc} = -0.0526525 * v + -0.1341857 * \omega + -0.1341857 * (v_{ref} - v_{rob}) + -0.4909975 * (\omega_{ref} - \omega_{rob}) + -0.2141134 * \rho + -0.2020394 * \alpha + -0.1888051 + 0.0712943$

If $v < 0.7970844$ and $\omega > 0.2311402$ and $(v_{ref} - v_{rob}) > -0.2606914$ and $(\omega_{ref} - \omega_{rob}) > 0.3577566$ and $\rho > 0.5432450$ and $\alpha > 0.3045580$ then $v_{inc} = -0.4283727 * v + 0.5254107 * \omega + 0.5254107 * (v_{ref} - v_{rob}) + -0.1505240 * (\omega_{ref} - \omega_{rob}) + 0.8359596 * \rho + -0.0326892 * \alpha + 0.3081844 + -0.0003588,$

$\omega_{inc} = 0.3454005 * v + -0.4909975 * \omega + -0.4909975 * (v_{ref} - v_{rob}) + -0.2778212 * (\omega_{ref} - \omega_{rob}) + 0.6994976 * \rho + 0.4886705 * \alpha + 0.2289957 + 0.5789568$

If $v < 1.0159674$ and $\omega > 0.1276233$ and $(v_{ref} - v_{rob}) < -0.2720805$ and $(\omega_{ref} - \omega_{rob}) > -0.0879307$ and $\rho > 0.7092691$ and $\alpha > -0.1685077$ then $v_{inc} = -0.9451361 * v + 0.4569717 * \omega + 0.4569717 * (v_{ref} - v_{rob}) + 0.8359596 * (\omega_{ref} - \omega_{rob}) + 0.6869079 * \rho + 0.7200467 * \alpha + 0.7449960 + 0.8263730,$

$\omega_{inc} = 0.2929497 * v + 0.2791020 * \omega + 0.2791020 * (v_{ref} - v_{rob}) + 0.6994976 * (\omega_{ref} - \omega_{rob}) + 0.1821506 * \rho + 0.6257491 * \alpha + 0.4543635 + 0.2199392$

$$\begin{aligned}
&\textit{If}\quad v>1.205344 \text{ and } \omega>-0.0968983 \text{ and } (v_{ref}-v_{rob})>0.0373195 \\
&\text{and } (\omega_{ref}-\omega_{rob})<0.1962357 \text{ and } \rho>1.2538755 \text{ and } \alpha<0.1218786 \\
&\text{then}\quad v_{inc}= 0.5642881*v + -0.2429051*\omega + -0.2429051*(v_{ref}-v_{rob}) + \\
&\qquad 0.3081844*(\omega_{ref}-\omega_{rob}) + -0.6377921*\rho + \\
&\qquad 0.2372806*\alpha + 0.3039157 + -0.7144687, \\
&\quad \omega_{inc}= 0.6927095*v + 0.7480350*\omega + 0.7480350*(v_{ref}-v_{rob}) + \\
&\qquad 0.2289957*(\omega_{ref}-\omega_{rob}) + 0.5583952*\rho + \\
&\qquad -0.4286680*\alpha + 0.2080972 + -0.4508324
\end{aligned}$$

$$\begin{aligned}
&\textit{If}\quad v>1.205344 \text{ and } \omega>-0.0968983 \text{ and } (v_{ref}-v_{rob})>0.0373195 \\
&\text{and } (\omega_{ref}-\omega_{rob})<0.1962357 \text{ and } \rho>1.2538755 \text{ and } \alpha<0.1218786 \\
&\text{then}\quad v_{inc}= 0.5642881*v + -0.2429051*\omega + -0.2429051*(v_{ref}-v_{rob}) + \\
&\qquad 0.3081844*(\omega_{ref}-\omega_{rob}) + -0.6377921*\rho + \\
&\qquad 0.2372806*\alpha + 0.3039157 + -0.7144687, \\
&\quad \omega_{inc}= 0.6927095*v + 0.7480350*\omega + 0.7480350*(v_{ref}-v_{rob}) + \\
&\qquad 0.2289957*(\omega_{ref}-\omega_{rob}) + 0.5583952*\rho + \\
&\qquad -0.4286680*\alpha + 0.2080972 + -0.4508324
\end{aligned}$$

$$\begin{aligned}
&\textit{If}\quad v>0.6196452 \text{ and } \omega<0.2323706 \text{ and } (v_{ref}-v_{rob})>0.5647629 \\
&\text{and } (\omega_{ref}-\omega_{rob})>-0.0327666 \text{ and } \rho>0.4904380 \text{ and } \alpha>-0.1630150 \\
&\text{then}\quad v_{inc}= -0.0378747*v + -0.3878541*\omega + -0.3878541*(v_{ref}-v_{rob}) + \\
&\qquad 0.1955760*(\omega_{ref}-\omega_{rob}) + 0.0637325*\rho + \\
&\qquad -0.1970121*\alpha + -0.2060373 + 0.4825770, \\
&\quad \omega_{inc}= 0.1229261*v + 0.1073914*\omega + 0.1073914*(v_{ref}-v_{rob}) + \\
&\qquad -0.2504428*(\omega_{ref}-\omega_{rob}) + 0.6417333*\rho + \\
&\qquad 0.2868955*\alpha + 0.1431067 + 0.1357565
\end{aligned}$$

It is clear from Fig. 15, that both the conventional ANN and ANFIS produce inferior results to those obtained using the proposed approach (refer to Fig. 7). Thus, the proposed modifications to conventional ANNs succeeded in producing an improved ANFIS system capable of outperforming both conventional ANNs and ANFIS for problems where the desired ANN output is not known a priori (like in the path tracking case study).

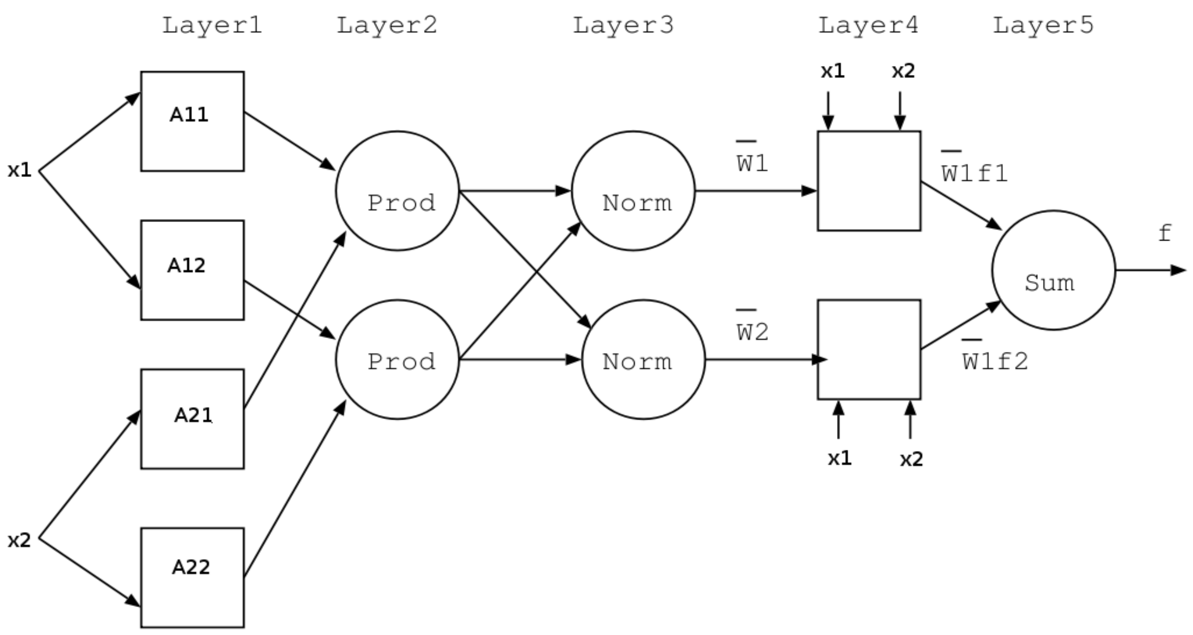

Fig. 14. Architecture of a typical ANFIS system for a 2 inputs single output example

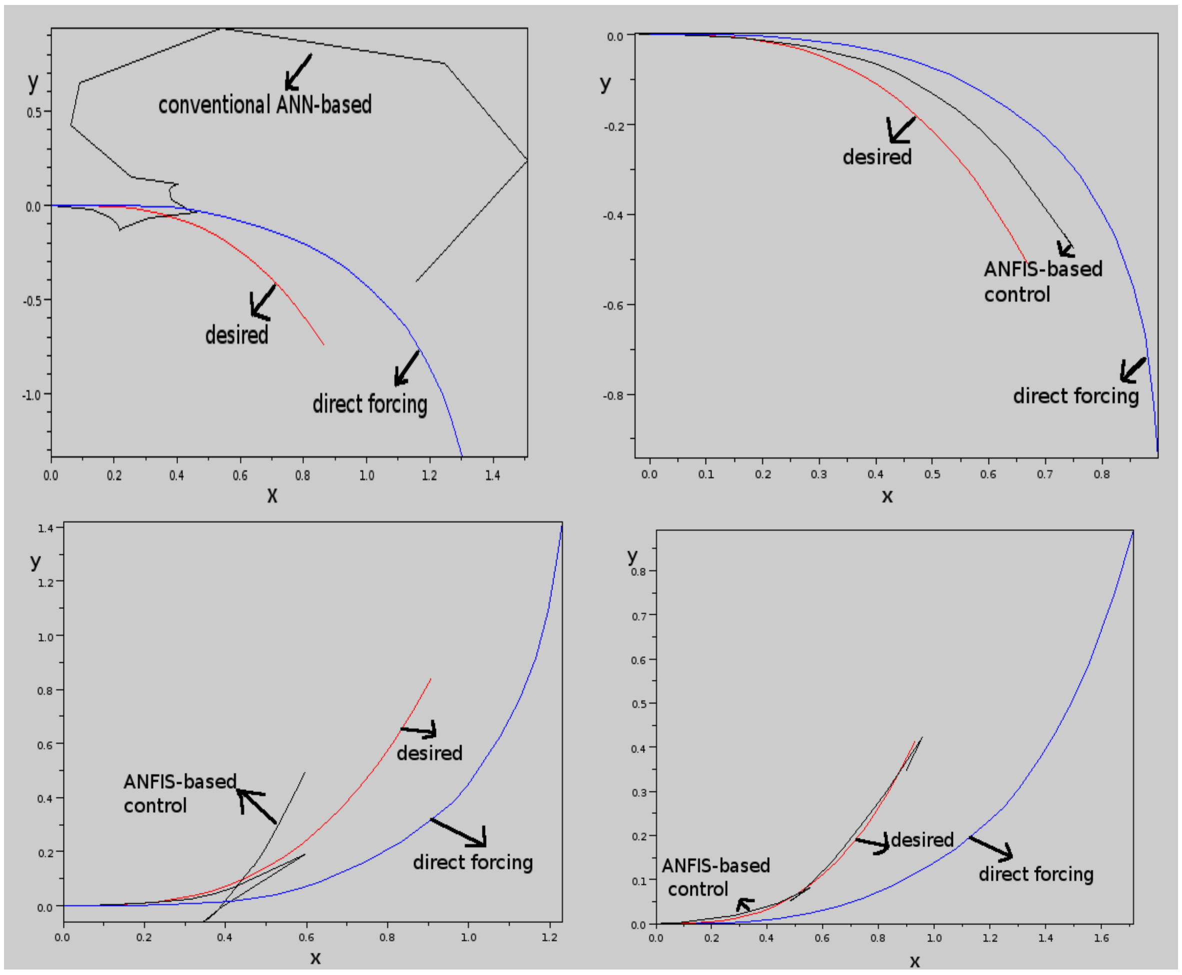

Fig. 15. Shows the results for using conventional ANNs and ANFIS instead of the proposed formulation for case B. By comparing these results with those in Fig. 7 , the superiority of the proposed approach-thanks to Allah- is clear.

8. References

[1] G J. Benitez, J. Castro, I. Requena, "Are Artificial ANNs Black Boxes?", IEEE Trans. on ANNs, September, 1997.

[2] J. Espinosa, J. Vandewalle, V. Wertz, *Fuzzy Logic, Identification and Predictive Control (Advances in Industrial Control)*, Springer-Verlag, London, 2005.

[3] Simon Haykin, ANNs and Learning Machines, Prentice Hall, November 2008.

[4] R. Setiono, "Extracting M-of-N Rules from Trained ANNs", *IEEE Trans. on ANNs*, Vol. 11, No.2, January 2000. pp. 510-519.

[5] S. Mitra, S. Pal, "Fuzzy Multi-Layer Perceptron, Inferencing and Rule Generation" , *IEEE Trans. on ANNs*, Vol. 6, No.1, January 1995. pp. 51-63.

[6] H. Senousy, M. Abou-El Makarem, "New Reliable Neural-Based Automatic Diesel Fault Diagnosis Systems", International Conference on Mechanical Engineering and Production MDP9, Cairo, Egypt, January 2008.

[7] I. Hamid, H. Senousy, M. Abou-Elmakarem, "An Improved Fuzzy Logic Controller For Ship Steering Based on Ior Operator and Neural Rule Extraction", ICCES08,

Faculty of Engineering - Ain Shams University, Computer Engineering & Systems Department Cairo, EGYPT, November 25-27, 2008.

[8] J.S. R. Jang "ANFIS: Adaptive Network based Fuzzy Inference Systems", *IEEE Transactions on Systems, Man, and. Cybernetics*, vol. 23, no. 3, (1993) 665–685.

[9] http://sourceforge.indices-masivos.com/projects/forallahfacon/

[10] S. Rao, Optimization, *Theory & Applications 2ed*, July 1984, John Wiley & Sons (Asia).

[11] R. Gonzalez, *ANNs for Variational Problems in Engineering*, PhD Thesis, Department of Computer Languages and Systems, Technical University of Catalonia,21 September 2008.

[12] G. Dudek, Michael Jenkin, *Computational Principles of Mobile Robotics*, Cambridge University Press; April 15, 2000.

[13] A. El-Bastawesy , A. El-sayed, M. Abdel-Salam, B. Salah, I. Adel, M. Alaa El-laffy, M. Tariq, B. Magdi, O. Fathi, A New Intelligent Strategy for Optimal Design of High Dimensional Systems, 2011 IEEE GCC Conference & Exhibition, Dubai, United Arab Emirates, 2011