Fuzzy Controller for Aircraft Auto-landing

Sudesh Kumar Kashyap and Girija Gopalarathnam

Abstract— **In the paper presented at UKIERI 2009 Workshop, an Extended Minimum Resource Allocation Neural Network (EMRANN) based controller was designed for auto-landing of high performance fighter aircraft subjected to severe wind and ineffective control surfaces due to stuck actuators. The performance of EMRANN was compared with a classical Baseline Trajectory Following Controller (BTFC). Under certain failure conditions, it was found that the BTFC alone could not meet the required performance for landing whereas EMRANN augmentation considerably improved the ability of BTFC to handle large faults while meeting the desired flight path and stringent touchdown conditions for auto-landing. This paper presents the design and implementation of a fuzzy controller named SAFIS (sequential adaptive fuzzy inference system). SAFIS is functionally equivalent to EMRANN. The performance of SAFIS is found to be identical or slightly better than EMRANN and also it requires less number of rules as compared to number of neurons for EMRANN.**

***Index Terms*— BTFC, Radial Basis Function, EMRANN, Fuzzy Logic, SAFIS**

I. INTRODUCTION

In the previous paper [1], an artificial intelligence technique based on Neural Network (NN) was used as a controller for auto-landing of a high performance aircraft with ineffective control surfaces due to stuck actuators and under severe winds. The NN used was Radial Basis Function Neural Network (RBFNN) having good local interpolation and global generalization abilities [2,3,4] as compared to Multilayer feed-forward neural networks (MFNN). An upgraded version of MRANN (variant of RBFNN) named Extended Minimum Resource Allocation Neural Network (EMRANN [5]) was used in conjunction with the classical controller (named Baseline Trajectory Following Controller or BTFC) for auto-landing.

Figure 1 shows the top view of architecture for aircraft Auto-landing using BTFC and EMRANN controller [1]. Based on aircraft current position in inertial frame, the tracking command generator generates the reference commands such as desired total velocity, altitude, heading angle and tracking error to steer the aircraft to a desired flight path for auto-landing. BTFC in the inner loop is designed using conventional method (root locus) under the no-fault condition but subjected to severe wind. BTFC is not only used to stabilize the overall system but also provides the error signals to train the EMRANN online. EMRANN controller aids the BTFC under failure conditions by learning the aircraft inverse by observing total signal to the actuators and comparing it to its output. By doing so, EMRANN is basically trained in such a way that BTFC output (i.e. error signals to EMRANN) tends to zero.

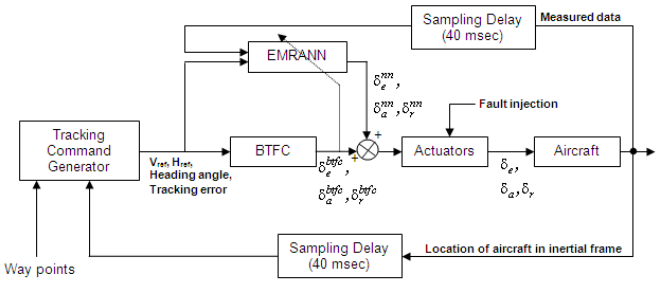

Fig.1: BTFC/EMRANN for Auto-Landing [1]

This paper provides details of implementation and validation of Fuzzy logic based controller in MATLAB® for auto-landing of a high performance aircraft with ineffective control surfaces due to stuck actuators and under severe winds. Section II of the paper covers the design of Fuzzy controller. Section III provides the results of auto-landing using Fuzzy controller. Conclusion with future work is drawn in section IV of the paper.

II. FUZZY CONTROLLER - SAFIS

Zadeh introduced the term *fuzzy logic* first time in his seminal work titled "Fuzzy sets" [6-7]. Before that Plato indicated a third possibility beyond True and False (of binary logic). Lukasiewicz further clarified this and proposed that third value could be translated as "possible" with numeric value between True and False and also there could be infinite-valued logic possible to represent the uncertainty. Zadeh's Fuzzy Logic (FL) facilitates to model the conditions, which are inherently imprecisely defined. Fuzzy techniques in the form of approximate reasoning provide decision support and expert system with powerful reasoning capabilities. In the past few decades, FL techniques have been used in varieties of applications such as i) image-analysis - detection of edges, feature extraction, classification, and clustering, ii) parameter estimation of unknown dynamic systems – aircraft, iii) home appliances – washing machine, air conditioning systems, and iv) decision fusion – situation and threat assessment. Fuzzy logic has inherent abilities to mimic the human mind so that it can be deployed for reasoning that are approximate rather than exact.

Development of FL system for each of such applications requires the following:

- Selection of Fuzzy sets and their membership functions for Fuzzification process
- Creation of rule base, with a help of domain expert, for inputs-output mapping
- Selection of Fuzzy operators required in fuzzy implication and aggregation process
- Selection of Fuzzy implication method and aggregation method
- Selection of defuzzification method

A Sequential Adaptive Fuzzy Inference System or SAFIS [8] is basically functionally equivalent to EMRANN or any RBFNN. The following is the relation between SAFIS and EMRANN:

- The number of basis function units is equal to the number of fuzzy if-then rules
- The consequent part of each fuzzy rule is a constant
- The membership functions of the premise variable within each fuzzy rule are Gaussian functions with the same variance
- The T-norm operator used to compute the firing strength of each rule is of multiplication type
- Both RBF network and the fuzzy inference system under consideration use the same method (i.e., normalized or non-normalized calculation) to derive the overall outputs

From the literature survey [8] it is found that SAFIS can perform better than RBF based Neural Network with lesser number of rules to meet the problem objective. The SAFIS algorithm requires only current data for the learning hence storage wise it is very efficient algorithm. The mechanism of SAFIS algorithm is similar to RBFNN except the growing and pruning criteria of used. RBFNN makes use of the significance of a neuron to add and remove the hidden neurons. In SAFIS growing and pruning criteria are based on the influence of a fuzzy rule. Figure 2 shows the structure of SAFIS.

SAFIS has a total 5 layers to map any kind of dynamical system. Layers 1 to 5 represent various rules as follows:

The i[th] rule for Multi-input and Multi-output (MIMO) system can be represented as

IF (x_1 is A_{1i}) AND (x_2 is A_{2i}) ,..., (x_n is A_{ni}) THEN (y_1 is w_{i1}) ,..., (y_m is w_{im}) (1)

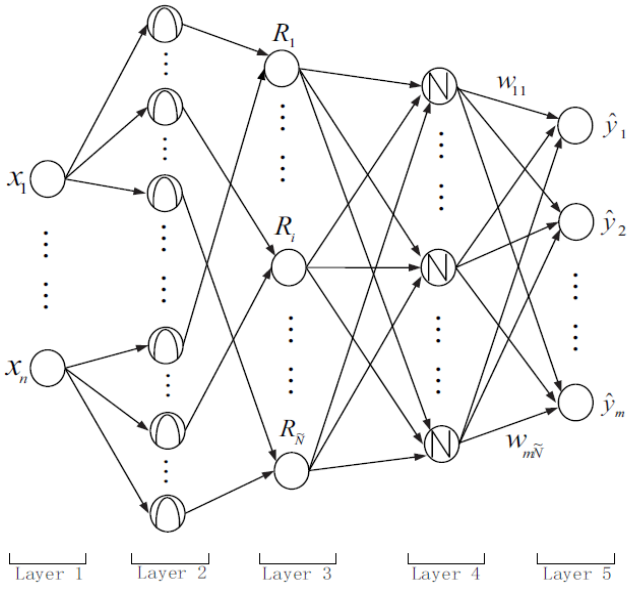

Fig 2: Structure of SAFIS [8]

where, x is input vector of size 'n', $A_{(.)i}$ is the Gaussian function of i[th] rule for each input, y is the output vector of size 'm'. The input and output vector are represented by $x=[x_1, x_2, ..., x_n]$ and $y=[y_1, y_2, ..., y_m]$ respectively. The functional detail of each layer is given as follows:

Layer 1: It consists of input vector x of size 'n'. The input vector can have also elements of output vector of previous time instant.

Layer 2: This layer consists of series of Gaussian functions with each input having total number of such functions equals to number of Fuzzy rules represented by $\tilde{N}$. Hence total number of Gaussian functions in a SAFIS would be n times $\tilde{N}$. The Gaussian function for i[th] input and j[th] rule is given by

$$A_{ij} = e^{\frac{-(x_i - c_{ij})^2}{a_j^2}} \quad (2)$$

with i=1,2,...,n and j=1,2,.., $\tilde{N}$

where c_{ij} is the centre of function and a_j is its 1-sigma or width. Figure 3 shows the typical Gaussian function.

Layer 3: Each node of this layer represents a fuzzy rule. At this layer the consequent part of each fuzzy rule (eq.1) is processed using T-norm operator. The output (i.e. firing strength) of this layer using T-norm operator such as Algebraic product is given by

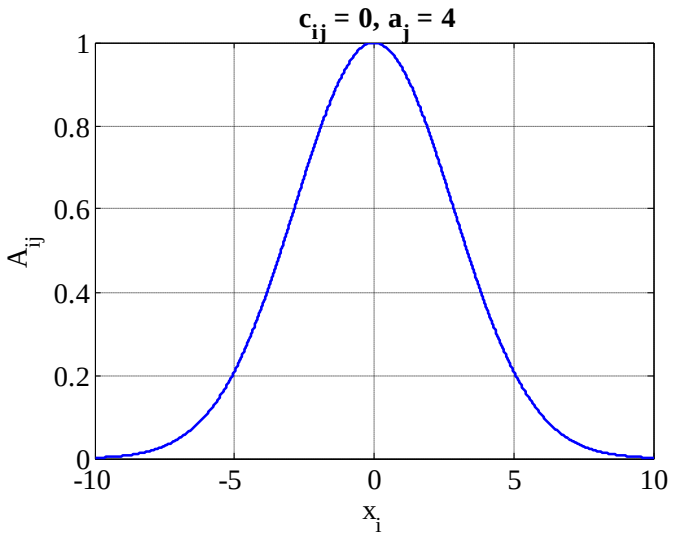

Fig. 3: Gaussian function

$$R_j = \prod_{i=1}^{n} A_{ij}(x_i) = e^{-\sum_{i=1}^{n} \frac{x_i - c_{ij}}{a_j^2}^2}, \quad j=1,..., \tilde{N} \tag{3}$$

Layer 4: At this layer output of each node from previous layer is normalized and is given by

$$G_j = \frac{R_j}{\sum_{j=1}^{\tilde{N}} R_j}, \quad j=1,..., \tilde{N} \tag{4}$$

Layer 5: An output layer with each node in it corresponds to an output variable computed as follows:

$$y_k = \sum_{j=1}^{\tilde{N}} w_{kj} G_j, \quad k=1,...,m \tag{5}$$

Algorithm

The purpose of SAFIS is to model a dynamical system (linear or non-linear) in real-time. This can be achieved by continuous learning using input-output data of a given system. The learning is the processing of growing and pruning the fuzzy rules using the concept called "influence of rule".

The complete algorithm is summarized as follows:

- Initialize SAFIS as follows
  a. Total number of rules , $\tilde{N}$ =1
  b. Supposing the dimension of input vector is 'n' then total number of Gaussian function is also 'n'. The width of each function is same and given by, a_j = $\kappa \|x\|$, where x is the input vector, j= $\tilde{N}$ =1 and κ is overlap factor. The centre of each function is given by, c_{ij} = x_i with i=1,..,n.

- Estimate the system output using following formula (see equation 5)

$$\hat{y}_k = \sum_{j=1}^{\tilde{N}} w_{kj} G_j, \quad k=1,...,m \tag{6}$$

- Compute estimation error

$$e = y - \hat{y} \tag{7}$$

where, y is measured output vector of the system.

- Check whether any new fuzzy rule is required to be added, i.e. influence of rule

  a. New rule is added only if it satisfy following two criteria

  ➔ Distance criteria: If the norm of vector $x - c_{nr}$ is greater than distance threshold ε , i.e.

$$\|x - c_{nr}\| > \varepsilon \tag{8}$$

  Where, distance threshold is given by

  $\varepsilon = \max(\varepsilon_{\max} . \gamma^t, \varepsilon_{\min})$, 't' is current time

  ➔ if influence of new rule computed below is greater than growing threshold e_g

$$E_{\inf}(\tilde{N}+1) = |e| \frac{1.8\kappa \|x - c_{nr}\|^n}{\sum_{j=1}^{\tilde{N}+1} (1.8 a_j)^n} > e_g \tag{9}$$

  Where, width of new rule $a_{\tilde{N}+1}$ = $\kappa \|x - c_{nr}\|$ and c_{nr} is a centre vector of nearest rule (from all rules $\tilde{N}$) to current input vector x in Euclidian sense

  b. If above two criteria are satisfied then following changes needed in SAFIS architecture

$$\left.\begin{aligned} \tilde{N} &= \tilde{N} + 1 \\ c &= [c \; x'] \\ a &= \left[a \; \kappa \|x - c_{nr}\|\right] \\ w &= \; w \; e' \end{aligned}\right\} \tag{10}$$

- If above two criteria are not satisfied then new rule will not be created and parameters (say θ) of nearest rule are updated using following Extended Kalman Filter (EKF) equations:

  Parameters given by $\theta = \left[w_{nr} \; c_{nr} \; a_{nr}\right]$

$$\left.\begin{aligned} \boldsymbol{B} &= \begin{bmatrix} w_d & \mathbf{c}_d & \mathbf{a}_d \end{bmatrix}' \\ \boldsymbol{K} &= \boldsymbol{PB}\ \ \boldsymbol{r} + \boldsymbol{B}'\boldsymbol{PB}\ \ ^{-1} \\ \theta &= \ \ \theta' + \boldsymbol{K}\boldsymbol{e}\ \ ' \\ \boldsymbol{P} &= \ \ \boldsymbol{I} - \boldsymbol{K}\boldsymbol{B}'\ \ \boldsymbol{P} + \boldsymbol{q} \end{aligned}\right\} \tag{11}$$

where, q (m+n+1 by m+n+1) and r (m by m) are process noise covariance and measurement noise covariance matrices respectively, P (m+n+1 by m+n+1) is covariance matrix, K is filter gain and B is filter gradient computed as follows:

$$\boldsymbol{w}_d = \dot{\boldsymbol{w}}_{nr} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{w}_{nr}} = \frac{\boldsymbol{R}_{nr}}{\sum_{j=1}^{\tilde{N}} \boldsymbol{R}_j} \boldsymbol{I}(\boldsymbol{m},\boldsymbol{m}) \tag{12}$$

where, I is an identity matrix of size m by m and $\boldsymbol{R}_{nr}$ is the firing strength of nearest rule (see eq. 3).

$$\boldsymbol{c}_d = \dot{\boldsymbol{c}}_{nr} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{c}_{nr}} = \frac{\boldsymbol{w}_{nr} - \hat{\boldsymbol{y}}}{\sum_{j=1}^{\tilde{N}} \boldsymbol{R}_j} 2\boldsymbol{R}_{nr} \frac{\boldsymbol{x} - \boldsymbol{c}_{nr}}{\boldsymbol{a}_{nr}^2} \tag{13}$$

$$\boldsymbol{a}_d = \dot{\boldsymbol{a}}_{nr} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_{nr}} = \frac{\boldsymbol{w}_{nr} - \hat{\boldsymbol{y}}}{\sum_{j=1}^{\tilde{N}} \boldsymbol{R}_j} 2\boldsymbol{R}_{nr} \frac{\left\| \boldsymbol{x} - \boldsymbol{c}_{nr} \right\|^2}{\boldsymbol{a}_{nr}^3} \tag{14}$$

Dimension of filter gradient $\boldsymbol{B}$ (see eq. 11) would be m+n+1 by m

- The reason of updating the parameters of nearest rule is because gradient vector for all rules except nearest rule will approach to zero more quickly. The existing rules (other than nearest) will maintain their influence because their parameters remain unchanged after learning the new observation. The newly added rule is also influencing and therefore it is not necessary to check for pruning after a new rule is added. It seems then the nearest rule needs to be checked for pruning based on satisfying following criteria

$$\boldsymbol{E}_{\inf}(\boldsymbol{nr}) = \left| \boldsymbol{w}_{nr} \right| \frac{\boldsymbol{1.8a}_{nr}{}^{\boldsymbol{n}}}{\sum_{i=1}^{\tilde{N}} (\boldsymbol{1.8a}_i)^n} < \boldsymbol{e}_p \tag{15}$$

Once a nearest rule is pruned then following other changes in SAFIS architecture are required:

$$\left.\begin{aligned} \tilde{\boldsymbol{N}} &= \tilde{\boldsymbol{N}} - \boldsymbol{1} \\ \boldsymbol{w} &= \boldsymbol{w}_j \\ \boldsymbol{c} &= \boldsymbol{c}_j \\ \boldsymbol{a} &= \boldsymbol{a}_j \end{aligned}\right\}, \boldsymbol{j} = \boldsymbol{1},..,\boldsymbol{nr}-\boldsymbol{1},\boldsymbol{nr}+\boldsymbol{1},...\tilde{\boldsymbol{N}} \tag{16}$$

Selection Criteria of SAFIS Parameters

The performance of SAFIS depends upon how well its parameters are selected for a given application. The various parameters of SAFIS are as follows:

$$\varepsilon_{\min}, \varepsilon_{\max}, \gamma, \kappa, \boldsymbol{e}_g, \boldsymbol{e}_p, \boldsymbol{q}, \boldsymbol{r} \text{ and } \boldsymbol{p}$$

where, $\varepsilon_{\min}, \varepsilon_{\max}, \gamma, \kappa$, $\boldsymbol{e}_g$ growing threshold for new rule, $\boldsymbol{e}_p$ is pruning threshold, and $\boldsymbol{q}$, $\boldsymbol{r}$, $\boldsymbol{p}$ are tuning parameters of EKF. The general rules (based on experience from previous experiments) to tentatively assign value to these constant parameters are as follows:

- $\varepsilon_{\max}$ is set around upper bound of input signals and $\varepsilon_{\min}$ is set to around 10% of $\varepsilon_{\max}$
- Decay constant γ should be within [0.9, 0.999] range
- Growing threshold $\boldsymbol{e}_g$ should be selected based on system performance. Small value of $\boldsymbol{e}_g$ results in better system performance with a cost of complex system structure
- Pruning threshold $\boldsymbol{e}_p$ can be set to around 10% of $\boldsymbol{e}_g$
- Overlap factor κ determines the width of the newly added rule and is can be within [1.0 2.0] range
- In general, q/r is not kept very low/high to avoid filter lag and filter covariance p is kept high to allow the filter to learn faster.

Comparison with EMRANN

Before designing SAFIS controller for auto-landing, it is tested with the following non-linear system [8] and compared with EMRANN.

$$y(t) = \frac{y(t-1)y(t-2)\ \ y(t-1) - 0.5}{1 + y^2(t-1) + y^2(t-2)} + u(t-1)$$

with initial condition y(1)=y(2)=0 and system input uniformly selected in the range [-1.5 1.5]. Total of 5000 data points were simulated. The data [u y] is used to train the SAFIS and EMRANN. SAFIS parameters used are as follows:

γ = 0.997, $\varepsilon_{\max}$ = 1.0, $\varepsilon_{\min}$ = 0.1* $\varepsilon_{\max}$, κ =1, e_g = 0.05, e_p = 0.005, m = 1, n =1, p = eye(m+n+1), q = eye(m+n+1), r = 10

Figure 4 shows the time history of rules and neurons. It can be seen that number of rules required are less as compared to number of neurons required to achieve comparable estimated system response by both the technique (see figure 5).

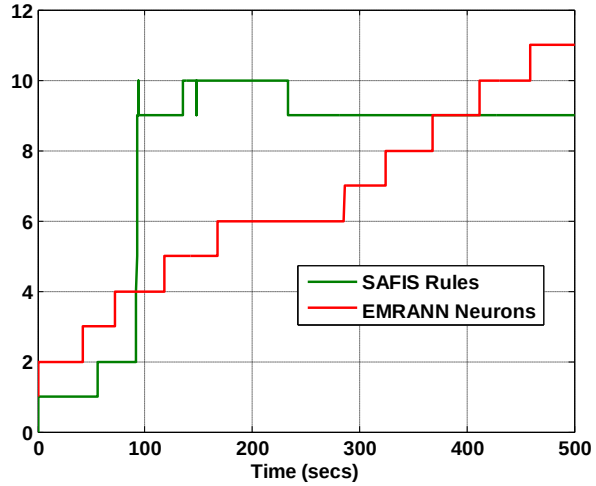

Fig. 4: Rules and Neurons

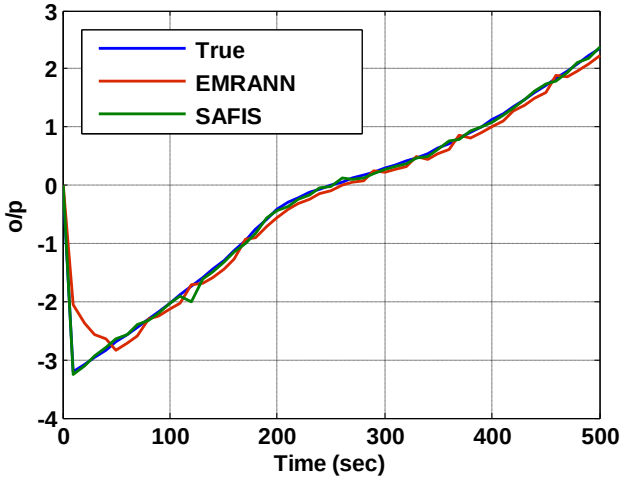

Fig. 5: System o/p response

III. AUTO-LANDING USING SAFIS CONTROLLER

In present paper, EMRANN for longitudinal and lateral-directional control is replaced by SAFIS controller. Figure 6 shows SAFIS implementation for longitudinal and lateral-directional control. The scaling mechanism 'f(u)' of input signals is same as used in EMRANN controller. The output of SAFIS is a desired control surface deflection. The gain 'k' represents maximum deflection possible. Figure 7 shows the proposed architecture of SAFIS controller for auto-landing.

Genetic algorithm

In order to select the SAFIS parameters ($\varepsilon_{min}, \varepsilon_{max}, \gamma, \kappa, \boldsymbol{e}_g, \boldsymbol{e}_p, \boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{p}$) genetic algorithm is used. Table 1 shows the lower and upper bounds of parameters used. The search is allowed within these bounds only. Total population of parameters considered is 20. For every population e $N = 11$ scenarios (see Table 2) of control surface failure are considered. The cost function (f) for j[th] population is obtained by computing the norm of following distance vector obtained from those 11 scenarios:

$$f_j = \frac{\sqrt{\sum_{i=1}^{N} x_i^2 + y_i^2}}{N}, \text{ with j=1,…,20}$$

where, variables x, y are the aircraft location at the end of simulation for each scenario.

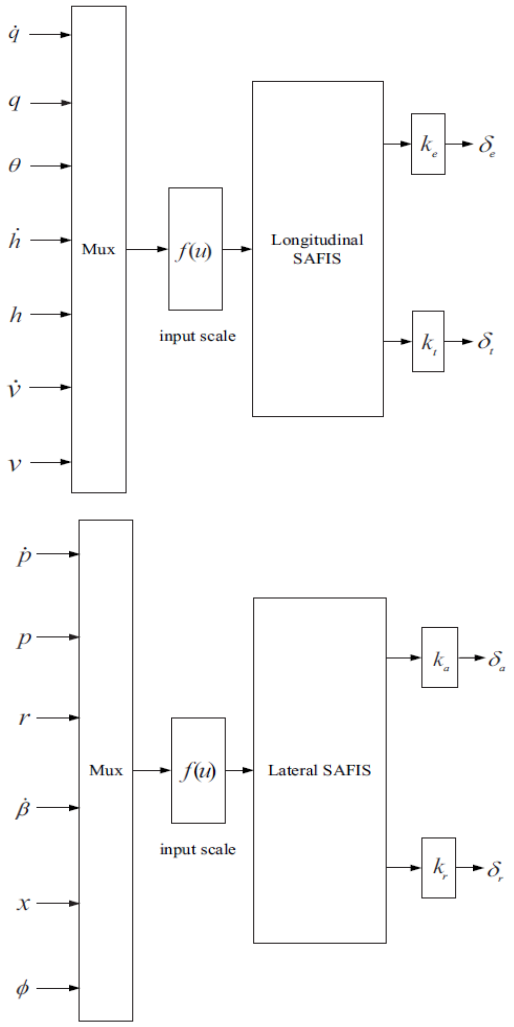

Fig. 6: SAFIS for longitudinal and lateral-directional control [8]

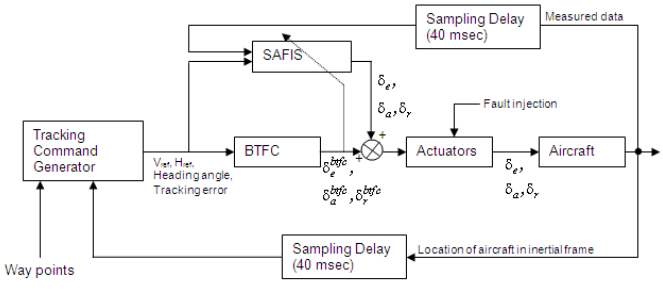

Fig. 7: SAFIS augmented with BTFC for auto-landing

Table 1: Bounds of Parameters [8]

Parameters	ϵ_{max}	ϵ_{min}	γ	κ	e_g	e_p	p_0	q	r
Lower Bound	0.1	0.01	0.9	1.0	0.0001	0.00001	0.5	0.0001	0.5
Upper Bound	1.0	0.1	0.999	2.0	0.001	0.0001	1000	1.0	1000

Table 2: Control Surface Failure scenarios [5]

Cases	Left elevator	Left aileron
1	0	0
2	0	16
3	0	-20
4	6	-6
5	-6	6
6	-10	18
7	20	-20
8	12	-12
9	-12	6
10	-12	16
11	20	-8

The best possible value of SAFIS parameters that assures a successful auto-landing under control surface failure of above cases are as follows:

$\gamma = 0.9456,\ \varepsilon_{max} = 0.5081, \varepsilon_{min} = 0.0161, \kappa = 1.5103, e_g = 0.0007,\ e_p = 0.000012,\ p = 551.18,$

$q = 0.6530,\ r = 795.07$

Figure 8 shows the cost function for 20 iteration of Genetic algorithm. It can be seen from the plot that after every iteration cost function reduces till the minimum value is reached. The final SAFIS parameter is chosen corresponding to minimum cost function.

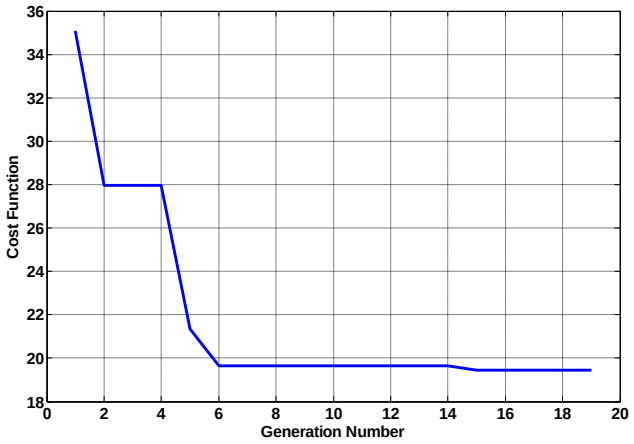

Fig. 8: Genetic Algorithm – cost function after 20 iterations

Results and Discussion

Left elevator failure

In the simulation left elevator is stuck to -10 degree at 10 second onwards. Figure 9 shows the comparison of aircraft response for BTFC alone, BTFC augmented with EMRANN and BTFC augmented with SAFIS. It can be seen from the plot that BTFC alone is not able to handle the left elevator failure and aircraft responses diverges significantly from desired values. In case of BTFC augmented with EMRANN or SAFIS, auto-landing is successful. Although the performance of EMRANN and SAFIS are almost identical but the response generated through SAFIS is much smoother as compared to those from EMRANN. This observation can be seen more clearly in figure 10 where comparisons are made in terms of elevator and aileron control surfaces deflections. Also it can be seen from this figure that number of fuzzy rules required are much less as compared to number of hidden layer neurons to achieve identical performance. Therefore, SAFIS's architecture is less complex as compared to EMRANN.

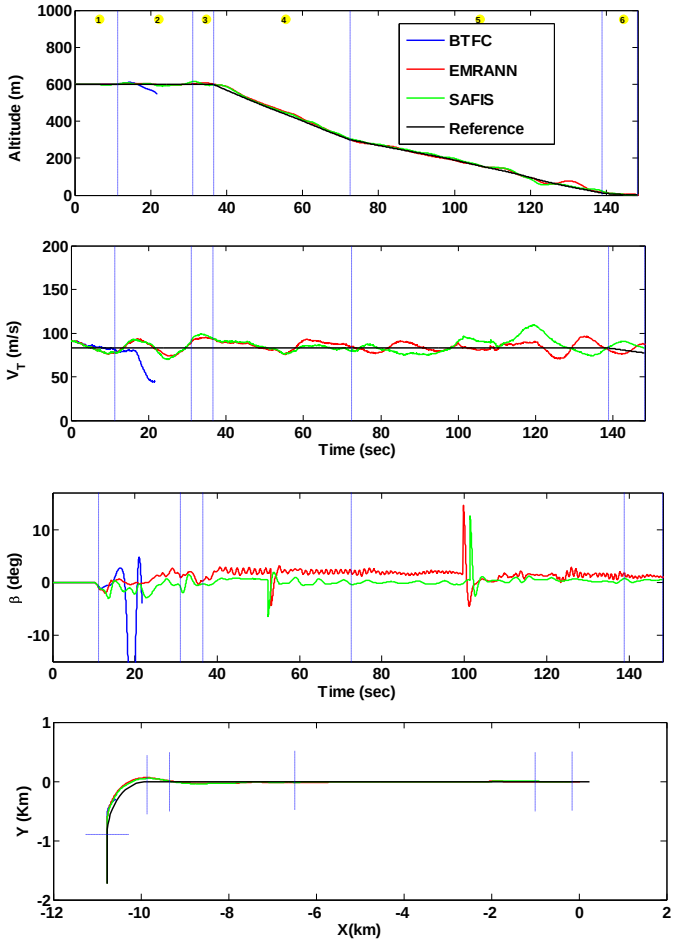

Fig. 9: Comparison of Aircraft altitude, airspeed, side slip angle and X-Y position under the left elevator stuck at -8 degree at 10th second

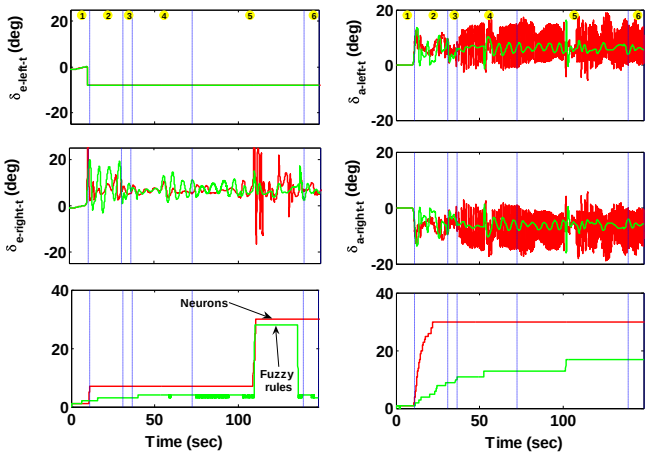

Fig. 10: Left/Right elevator/aileron and neuron/rules response under the left elevator stuck at -8 degree at 10th second

Left elevator and Right Aileron failure

In the simulation left elevator is stuck to -10 degree and right aileron stuck to 8 degree at 10 second onwards. Figure 11 shows the comparison of aircraft response for BTFC alone, BTFC augmented with EMRANN and BTFC augmented with SAFIS. It can be seen from the plot that BTFC alone is not able to handle the simultaneous failure of left elevator and right aileron and aircraft responses diverges significantly from desired values. In case of BTFC augmented with EMRANN or SAFIS, auto-landing is successful. Although the performance of EMRANN and SAFIS are almost identical but the response generated through SAFIS is much smoother as compared to those from EMRANN. This observation can be seen more clearly in figure 12 where comparisons are made in terms of elevator and aileron control surfaces deflections. Also it can be seen that in this failure case also the number of fuzzy rules required are much less as compared to number of hidden layer neurons to achieve identical performance. Therefore, SAFIS's architecture is again less complex as compared to EMRANN.

The PC based flight simulator software named Flightgear [9] is integrated with SAFIS controller based auto-landing system for the visualization of aircraft auto-landing (see fig. 13).

IV. Concluding Remarks

Sequential Adaptive Fuzzy Inference System or SAFIS is augmented with BTFC for aircraft auto-landing in case of severe wind and control surfaces failures. SAFIS is functionally equivalent to EMRANN. The performance of SAFIS is found to be identical or slightly better than EMRANN and also it requires less number of rules as compared to number of neurons for EMRANN. Therefore, SAFIS's architecture is less complex as compared to EMRANN.

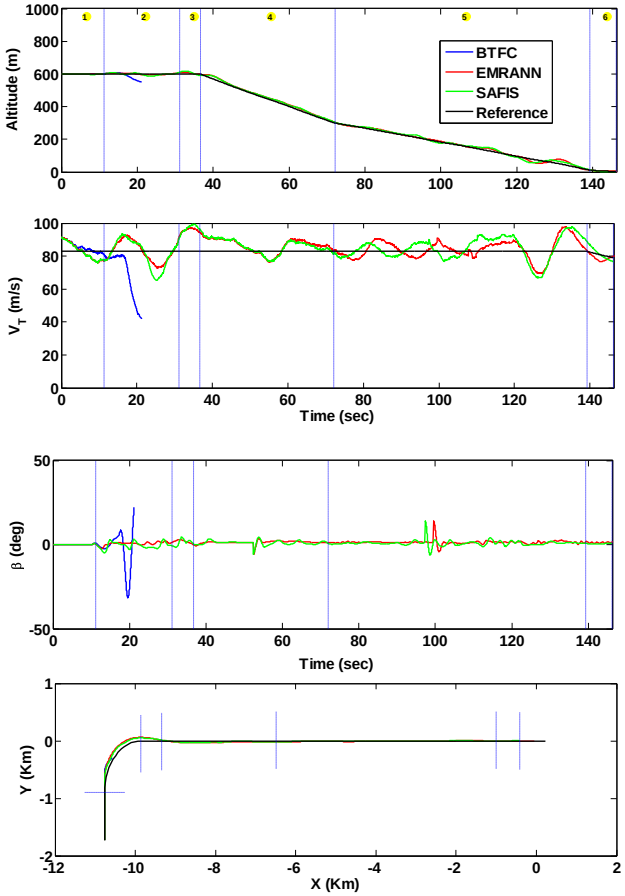

Fig. 11: Comparison of Aircraft altitude, airspeed, side slip angle and X-Y position under the left elevator stuck at -8 degree and right aileron stuck at 8 degree at 10 second

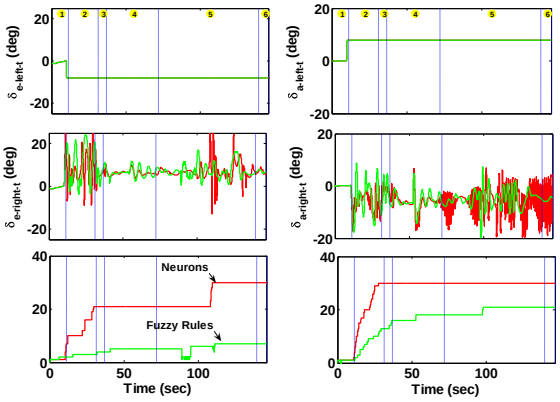

Fig. 12: Left/Right elevator/aileron and neuron/rules response under the left elevator stuck at -8 degree and right aileron stuck at 8 degree at 10 second

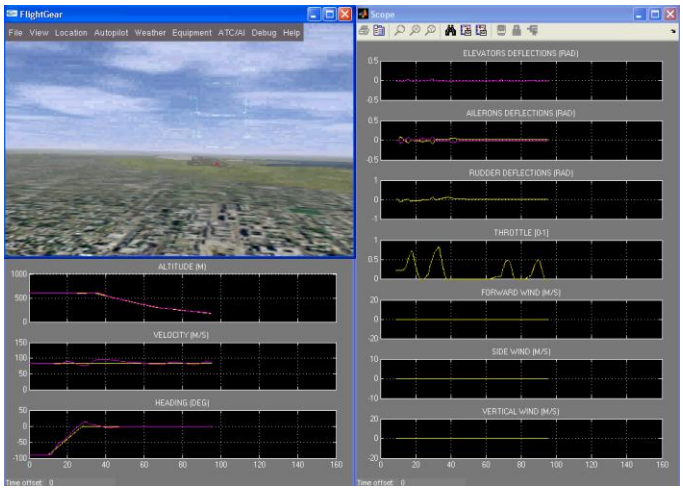

Fig. 13: Snapshot of Integrated Flightgear's Visuals

ACKNOWLEDGMENT

Authors acknowledge Prof. N. Sundarajan, School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore for the mathematical model and Dr. Abhay Pashilkar, Senior Scientist, FMCD, NAL, Bangalore for providing SIMULINK model of the aircraft, BTFC and EMRANN and technical support

REFERENCES

[1] Sudesh K. Kashyap and G Girija, Aircraft Auto-Landing Using Enhanced Fault-Tolerant Neural Networks, NAL/BC, UK-TR-0901, October, 2009.

[2] N. Sundararajan, P. Saratchandram and Lu Ying Wei, "Radial Basis Function Neural networks with Sequential Learning: MRAN and its Applications", Progress in Neural Processing, Vol. 11, ISBN 981-02-3771-5, 1999.

[3] Y. Lu, N. Sundararajan and P. Saratchandran, "A Sequential Learning Scheme for Function Approximation using Minimal Radial Basis Function Neural Networks", Neural Computation, Vol. 9, pp. 461-478, 1997.

[4] Y. Li, N. Sundararajan and P. Saratchandran, "Analysis of Minimum Radial Basis Function Network Algorithm for Real-time Identification of Non-Linear Dynamic Systems", IEE Proceeding on Control Theory Applications, Vol. 147, No. 4, pp. 478-484, 2000.

[5] A. A. Pashilkar, "Enhanced Fault-Tolerant Neural Controller for Aircraft Auto-landing", NAL PDFC 0505, April 2005.

[6] S. K. Kashyap and J.R. Raol, Unification and Interpretation of Fuzzy Set Operations, PDFC 0502, 15th March, 2005.

[7] S.K. Kashyap and Raol J.R, "Unification and Interpretation of Fuzzy Set operations", CCECE/CCGEI, IEEE Electrical & computer section, Ottawa, Canada, pp 355-358, May 2006.

[8] Rong Haijun, Efficient Sequential Fuzzy-Neural Algorithms for Aircraft Fault-Tolerant Control, PhD thesis, School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore, 2007.

[9] www.flightgear.org

Dr. Sudesh K. Kashyap received his PhD degree titled "Decision fusion using fuzzy logic" from University of Mysore and M.E. degree in Electrical engineering with specialization in Automatic Control and Robotics from M.S. University of Baroda. Currently, he is working at National Aerospace Laboratories, Bangalore as scientist since 2002. His areas of interest are Kalman filtering, sensor fusion, Fuzzy logic, Bayesian theory and neural network.

Dr. (Mrs.) Girija G, received her Ph.D. from Bangalore University in 1996. She is presently the Deputy Head of the Flight Mechanics and Control Division at National Aerospace Laboratories (NAL), Bangalore. Her areas of research are: modeling, parameter estimation of aerospace vehicles, multi sensor data fusion and Enhanced and Synthetic Vision systems