Predicting with Confidence - An Improved Dynamic Cell Structure

Yan Liu[1], Bojan Cukic[1], Michael Jiang[2], and Zhiwei Xu[2]

[1] Lane Department of Computer Science and Electrical Engineering,
West Virginia University, Morgantown, WV 26506, USA
{yanliu, cukic}@csee.wvu.edu
[2] Motorola Labs, Motorola Inc., Schaumburg, IL 60196, USA
{Michael.Jiang, Zhiwei.Xu}@motorola.com

Abstract. As a special type of Self-Organizing Maps, the Dynamic Cell Structures (DCS) network has topology-preserving adaptive learning capabilities that can, in theory, respond and learn to abstract from a much wider variety of complex data manifolds. However, the highly complex learning algorithm and nonlinearity behind the dynamic learning pattern pose serious challenge to validating the prediction performance of DCS and impede its spread in control applications, safety-critical systems in particular.

In this paper, we improve the performance of DCS networks by providing confidence measures on DCS predictions. We present the validity index, an estimated confidence interval associated with each DCS output, as a reliability-like measure of the network's prediction performance. Our experiments using artificial data and a case study on a flight control application demonstrate an effective validation scheme of DCS networks to achieve better prediction performance with quantified confidence measures.

1 Introduction

Often viewed as black box tools, neural network models have a proven track of record of successful applications in various fields. In safety-critical systems such as flight control, neural networks are adopted as a popular soft-computing paradigm to carry out the adaptive learning. The appeal of including neural networks in these systems is in their ability to cope with a changing environment. Unfortunately, the validation of neural networks is particularly challenging due to their complexity and nonlinearity and thus reliable prediction performance of such models is hard to assure. The uncertainties (low confidence levels) existed in the neural network predictions need to be well analyzed and measured during system operation. In essence, a reliable neural network model should provide not only predictions, but also confidence measures of its predictions.

The Dynamic Cell Structures (DCS) network is derived as a dynamically growing structure in order to achieve better adaptability. DCS is proven to have topology-preserving adaptive learning capabilities that can respond and learn to abstract from a much wider variety of complex data manifolds [1,2]. The structural flexibility of DCS

L. Wang, K. Chen, and Y.S. Ong (Eds.): ICNC 2005, LNCS 3610, pp. 750–759, 2005.

network has gained it a good reputation of adapting faster and better to a new region. A typical application of DCS is the NASA Intelligent Flight Control System (IFCS). DCS is employed in IFCS as online adaptive learner and provides derivative corrections as control adjustments during system operation. Within this application, it has been proven to outperform Radial Basis Function (RBF) and Multi-Layer Perceptron network models [3]. As a crucial component of a safety critical system, DCS network is expected to give robust and reliable prediction performance in operational domains.

Our research focuses on validating and improving the prediction performance of DCS network by investigating the confidence for DCS outputs. We present the Validity Index, as a measure of accuracy imposed on each DCS prediction. Each validity index reflects the confidence level on that particular output. The proposed method is inspired by J. Leonard's paper on the validation of Radial Basis Function (RBF) neural networks [4]. Leonard developed a reliability-like measure called validity index which statistically evaluates each network output. Different from the pre-defined static RBF network structure, the DCS progressively adjusts (grows/prunes) its structure including locations of neurons and connections between them to adapt to the current learning data. Thus, unbiased estimation of confidence interval is impossible to obtain through S-fold cross-validation due to constraints of time and space. Yet, DCS emphasizes topological representation of the data, while RBF does not. By the end of DCS learning, the data domain is divided into Voronoi regions. Every region has a neuron as its centroid. The "locality" of DCS learning is such that the output is determined by only two particular neurons, the best matching unit and the second best matching unit. Intuitively, if the Voronoi region of a neuron does not contain sufficient data, it is expected that the accuracy in that region will be poor. Based on the "local error" computed for each neuron, our approach provides an estimated confidence interval, called the Validity Index for DCS outputs.

The paper is organized as follows. The architecture of DCS network and its learning algorithm are described in Section 2. The concept of validity index and its statistical computation are presented in detail in Section 3. In Section 4, we further illustrate the validity index in DCS networks by presenting the experimental results using an artificial data set. Section 5 describes a case study on a real-world control application, the IFCS, and presents experimental results on the validity index in DCS using flight simulation data. In the end, conclusions are discussed in Section 6.

2 The Dynamic Cell Structures

The Dynamic Cell Structure (DCS) network can be seen as a special case of Self-Organizing Map (SOM) structures. The SOM is introduced by Kohonen [5] and further improved to offer topology-preserving adaptive learning capabilities that can, in theory, respond and learn to abstract from a much wider variety of complex data-manifolds. The DCS network adopts the self-organizing structure and dynamically evolves with respect to the learning data. It approximates the function that maps the input space. At last, the input space is divided into different regions, referred to as the Voronoi regions [1,2,6]. Each Voronoi region is represented by its centroid, a neuron associated with its reference vector known as the "best matching unit (BMU)". Further, a "second best

matching unit (SBU)" is defined as the neuron whose reference vector is the second closest to a particular input. Euclidean distance metric is adopted for finding both units. The set of neurons connected to the BMU are considered its neighbors and denoted by NBR.

The training algorithm of the DCS network combines the competitive Hebbian learning rule and the Kohonen learning rule. The competitive Hebbian learning rule is used to adjust the connection strength between two neurons. It induces a Delaunay Triangulation into the network by preserving the neighborhood structure of the feature manifold. Denoted by $C_{ij}(t)$, the connection between neuron i and neuron j at time t is updated as follows:

$$C_{ij}(t+1) = \begin{cases} 1 & (i = BMU) \wedge (j = SBU) \\ 0 & (i = BMU) \wedge (C_{ij} < \theta) \\ & \wedge (j \in NBR \setminus \{SBU\}) \\ \alpha C_{ij}(t) & (i = BMU) \wedge (C_{ij} \geq \theta) \\ & \wedge (j \in NBR \setminus \{SBU\}) \\ C_{ij}(t) & (i, j \neq BMU) \end{cases}$$

where α is a predefined forgetting constant and θ is a threshold preset for dropping connections.

The Kohonen learning rule is used to adjust the weight representations of the neurons which are activated based on the best-matching methods during the learning. Over every training cycle, let $\Delta \boldsymbol{w_i} = \boldsymbol{w_i}(t+1) - \boldsymbol{w_i}(t)$ represent the adjustment of the reference vector needed for neuron i, the Kohonen learning rule followed in DCS computes $\Delta \boldsymbol{w_i}$ as follows.

$$\Delta \boldsymbol{w}_i = \begin{cases} \varepsilon_{BMU}(\boldsymbol{m} - \boldsymbol{w_i}(t)) & (i = BMU) \\ \varepsilon_{NBR}(\boldsymbol{m} - \boldsymbol{w_i}(t)) & (i \in NBR) \\ 0 & (i \neq BMU) \wedge (i \notin NBR) \end{cases}$$

where $\boldsymbol{m}$ is the desired output, and $0 < \varepsilon_{BMU}, \varepsilon_{NBR} < 1$ are predefined constants known as the learning rates that define the momentum of the update process. For every particular input, the DCS learning algorithm applies the competitive Hebbian rule before any other adjustment to ensure that the SBU is a member of NBR for further structural updates.

The DCS learning algorithm is diplayed in Figure 1. According to the algorithm, N is the number of training examples. Resource values are computed at each epoch as local error measurements associated with each neuron. They are used to determine the sum of squared error of the whole network. Starting initially from two connected neurons randomly selected from the training set, the DCS learning continues adjusting its topologically representative structure until the stopping criterion is met. The adaptation of lateral connections and weights of neurons are updated by the aforementioned Hebbian learning rule and Kohonen learning rule, respectively. The resource values of the neurons are updated using the quantization vector. In the final step of an iteration, the local error is reduced by inserting new neuron(s) in certain area(s) of the input space where the errors are large. The whole neural network is constructed in a dynamic way such that in the end of each learning epoch, the insertion or pruning of a neuron can be triggered if necessary.

```
Initialization;
Repeat until stopping criterion is satisfied
{
   Repeat N times
        {
        Determine the BMU and SBU;
        Update lateral connections;
        Adjust the weights;
        Update resource values;
        }
   If needed, a new neuron is inserted;
   Decrement Resource Values;
}
```

Fig. 1. A brief description of the DCS learning algorithm

It should be noted that while the DCS network is used for prediction, the computation of output is different from that during training. When DCS is in recall, the output is computed based on two neurons for a particular input. One is the BMU of the input; the other is the closest neighbor of the BMU other than the SBU of the input. In the absence of neighboring neurons of the BMU, the output value is calculated using the BMU only.

3 The Validity Index in DCS Networks

As a V&V method, validity check is usually performed through the aide of software tools or manually to to verify the correctness of system functionality and the conformance of system performance to pre-determined standards. The validity index proposed by J. Leonard [4] is a reliability-like measure provided for further validity checking. Validity index is a confidence interval associated with each output predicted by the neural network. Since a poorly fitted region will result in lower accuracy, it should be reflected by poor validity index and later captured through validity checking.

Given a testing input, the validity index in DCS networks is defined as an estimated confidence interval with respect to the DCS output. It can be used to model the accuracy of the DCS network fitting. Based on the primary rules of DCS learning and certain properties of final network structure, we employ the same statistical definition as for confidence intervals and variances for a random variable to calculate the validity index in DCS. The computation of a validity index for a given input x consists of two steps: 1) compute the local error associated with each neuron, and 2) estimate the standard error of the DCS output for x using information obtained from step 1). The detail description of these two steps are as follows.

1. The final form of DCS network structure is represented by neurons as centroids of Voronoi regions. Since the selection of the best matching unit must be unique, only those data points whose BMU are the same will be contained in the same region. Therefore, all Voronoi regions are non-overlapping and cover the entire learned domain. The data points inside each region significantly affect the local fitting accuracy. The local estimate of variance of the network residual in a particular region can be calculated over these data points contained in the region and then be associated with its representative neuron. More specifically, the local estimate of variance s_i^2 associated with neuron i can be computed as:

$$s_i^2 = \frac{1}{(n_i - 1)} \sum_{k=1}^{n_i} E_k,$$

where n_i is the number of data points covered by neuron i and E_k is the residual returned from the DCS recall function for data point k.
In Section 3, we show that the adjustment by competitive Hebbian learning rule concerns connections only between the BMU and its neighbors. The further update of weight values by Kohonen learning rule is performed only on the BMU and its neighbors as well. Consequently, training data points covered by the neighboring neurons of neuron i make proportional contributions to the local error of neuron i. Considering such contributions, we modify the computation of the local estimate of variance, now denoted by $s_i^{'2}$, as follows.

$$s_i^{'2} = \frac{s_i^2 + \sum_{j \in NBR} C_{ij} s_j^2}{1 + \sum_{j \in NBR} C_{ij}}.$$

As a result, the influence of all related data points is taken into account accordingly based on connections, referred to as C_{ij}, between the BMU and its neighbors. It should be noted that since the DCS networks are often adopted for online learning, no cross-validation is allowed. Hence, the residual calculated for each data point is in fact a biased estimate of the expected value of residual due to the fact that each data point itself contributed to its own prediction. Nonetheless, under the assumption that there is no severe multi-collinearity and relatively few outliers exist in the data, the probability that the deviation from the expected value will be significant is very low and thus can be ignored.
2. Recall that the output produced by DCS is determined by the BMU and its closest neighbor (CNB) of the given input. Thus, the local errors associated with these two neurons are the source of inaccuracies of fitting. We use the standard error, a statistic that is often used to place a confidence interval for an estimated statistical value. Provided with the local estimate of variance for every neuron from step 1), we now define the 95% confidence limit for the local prediction error estimate with respect to neuron i as:

$$CL_i = t_{.95} \sqrt{1 + \frac{1}{n_i}} s_i',$$

where $t_{.95}$ is the critical value of the Student's t-distribution with $n_i - 1$ degrees of freedom. The 95% confidence interval for the network output y given a testing input is thus given by:

$$(y - \frac{(CL_i + CL_j)}{2}, y + \frac{(CL_i + CL_j)}{2}),$$

where $i = BMU$ and $j = CNB$ with respect to the input x.

Now we slightly modify the DCS training algorithm in order to calculate the validity index. Note that because all needed information is already saved at the final step of each training cycle, without any additional cost required, we simply calculate $s_i'^2$ for each neuron after the learning stops. When the DCS is in recall for prediction, the validity index is computed based on the local errors and then associated with every DCS output. In order to complete the validity check, further examination needs to be done by software tools or system operators. In the case of a control application, a domain specific threshold can be pre-defined to help verify that the accuracy indicated by the validity index is acceptable.

4 An Example with Artificial Data

In order to demonstrate the validity index in DCS network model as an improvement of the network prediction, we present an example using an artificial data set. The DCS is trained on a single-input, single-output function as seen in [4]:

$$f(x) = 0.2\sin(1.5\pi x + 0.5\pi) + 2.0 + \varepsilon,$$

where ε is a Gaussian noise.

We sample x's from the interval $[-1, 1]$ randomly. Therefore, at least initially, there exist regions where the learning data points are not as dense as in the others. We then obtain two different DCS network models by varying the stopping criterion. Figure 2 illustrates the validity index for these two DCS models, one with 13 neurons and the other with 27 neurons, shown as plot (a) and plot (b), respectively. By comparing the prediction performance of these two models using the validity index, which is shown as

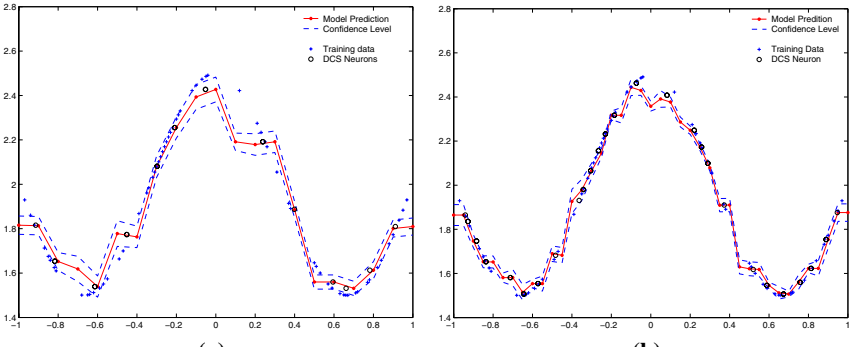

Fig. 2. Examples of validity index for a DCS model. (a): The model with 13 neurons. (b): The model with 27 neurons.

confidence band in both figures, we can conclude that the DCS network model shown in Figure 2 (b) has better prediction performance. Furthermore, we can observe that regions with sparse learning data have low confidence levels.

5 A Case Study

We investigate the prediction performance of DCS networks for the Intelligent Flight Control System (IFCS). The IFCS is an example of adaptive flight control application for NASA F-15 aircraft. As the post-adaptation validation approach, the validity index is a major component of our validation framework for IFCS [7].

5.1 The Intelligent Flight Control System

The Intelligent Flight Control System (IFCS) was developed by NASA with the primary goal to "*flight evaluate control concepts that incorporate emerging soft computing algorithms to provide an extremely robust aircraft capable of handling multiple accident and/or an off-nominal flight scenario*" [8,9].

The diagram in Figure 3 (a) shows the architectural overview of NASA's first generation IFCS implementation using Online Learning Neural Network (OLNN). Figure 3 (b) shows the user interface of an experimental IFCS simulator [10]. The control concept can be briefly described as follows. Notable discrepancies from the outputs of the Baseline Neural Network and the Real-time Parameter Identification (PID), either due to a change in the aircraft dynamics (loss of control surface, aileron, stabilator) or due to sensor noise/failure, are accounted by the Online Learning Neural Network.

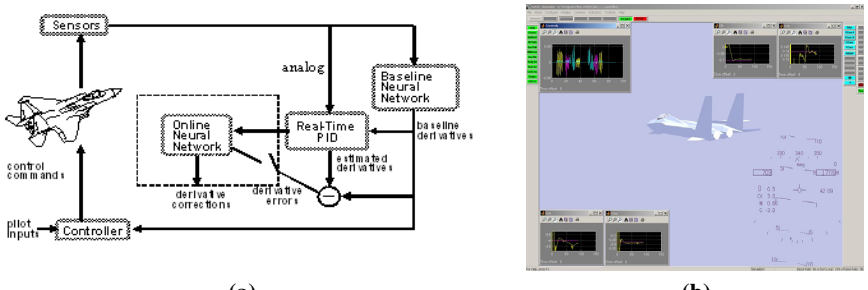

Fig. 3. (a): The Intelligent Flight Control System and (b): NASA-WVU F-15 Simulator

The primary goal of OLNN is to accomplish in-flight accommodation of discrepancies. The critical role played by the OLNN is to fine-tune the control parameters and provide a smooth and reliable control adjustments to system operation. When OLNN performs adaptation, its behavior has a direct consequence on the performance of the flight control system. In such a safety-critical application, it is necessary to understand and assure the prediction performance of the OLNN.

Our previous research provides a validation framework for validating the OLNN learning. It consists of a novelty detection tool to detect novel (abnormal) conditions entering the OLNN, and online stability monitoring techniques to investigate the NN's stability behavior during adaptation [7,11,12]. Although learning can be closely monitored and analyzed, when the system is in operation, it is probable that the predictions of the OLNN will become unreliable and erroneous due to extrapolation. Therefore, providing a reliability-like measurement with respect to each particular output can further enforce safety of the system in operation. In IFCS, the neural network that implements the OLNN component is the Dynamic Cell Structure (DCS).

5.2 Experimental Results

With the aide of the high-fidelity flight control simulator, we are able to test our approach for adaptive flight control through experimentation in simulated environments. The online neural networks in IFCS learn on the environmental changes and accommodate failures. They generate derivative corrections as compensation to the PTNN output (see Figure 3). We use validity index to evaluate the accommodation performance and validate the predictions of the DCS network. In our experiment, we simulate the online learning of a DCS network on a failure mode condition and compute the validity index in real-time.

We simulate the online learning of the DCS network under two different failure mode conditions. One is the stuck-at-surface type of failure. The simulated failed flight condition in this case is the aircraft¡s stuck left stabilator, which is simulated to stuck at an angle of +3 degree. The other is the loss-of-surface type of failure. This simulated failure has 50% of surface loss at the left stabilator. In each run, the DCS network updates its learning data buffer at every second and learns on the up-to-date data set of size 200 at a frequency of 20Hz. We first start the DCS network under nominal flight conditions with 200 data points. After that, every second, we first set the DCS network in recall mode and calculate the derivative corrections for the freshly generated 20 data points, as well as their validity index. Then we set the DCS network back to the learning mode and update the data buffer. While updating the data buffer, we discard the first incoming 20 data points and add the freshly generated 20 data points to maintain the buffer size, i.e., 200. The DCS network continues learning and repeats the recall-learn procedure.

Figure 4 and Figure 5 show the experimental results of the simulations on these two failures, respectively. Plot (a)'s show the final form of the DCS network structure at the end of the simulation. As a three-dimensional demonstration, the x-axis and y-axis represent two independent variables, α and β, respectively. The z-axis represents one derivative correction, $\Delta Cz\alpha$. The 200 data points in the data buffer at the end of the simulation are shown as crosses in the 3-D space. The network structure is represented by circles (as neurons) connected by lines as a topological mapping to the learning data. Plot (b)'s present the validity index, shown as error bars. The x-axis here represents the time frames. In both simulations, the failure occurs at the 100^{th} data frame. We compute the validity index for the data points that are generated five seconds before and five seconds after the failure occurs. In total, Plot (b) illustrates the validity index for 200 data points.

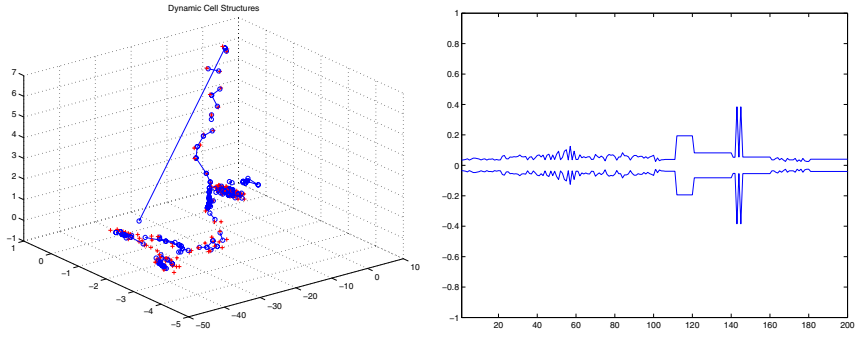

Fig. 4. A stuck-at-surface failure simulation in real-time (20Hz). (a): The final form of DCS network structures. (b): Validity Index shown as error bars for each DCS output.

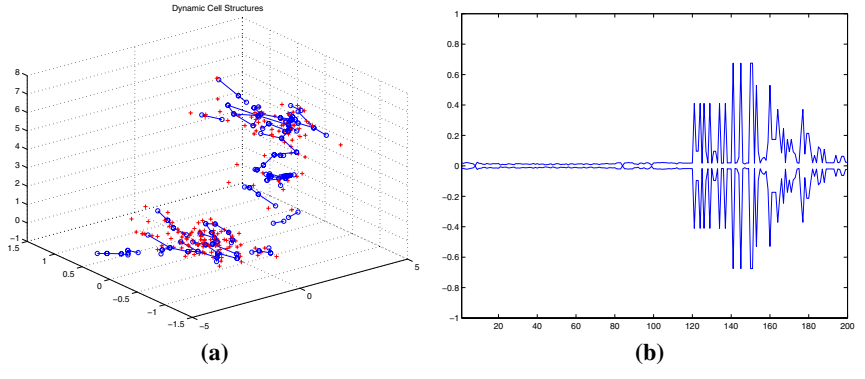

Fig. 5. Testing on loss-of-surface failure simulation data in real-time. (a): The final form of DCS network structures. (b): Validity Index shown as error bars for each DCS output.

A common trend revealed in both figures by the validity index is the increasingly larger error bars after the failure occurs. Then, the error bars start shrinking while the DCS network starts adapting to the new domain and accommodating the failure. After the failure occurs, the change (increase/decrease) of the validity index varies. This depends on the characteristics of the failure as well as the accommodation performance of the DCS network. Nevertheless, the validity index explicitly indicates how well and how fast the DCS network accommodates the failures.

6 Conclusions

Known for its structural flexibility, DCS networks are adopted in safety-critical systems for online learning in order to quickly adapt to a changing environment and provide reliable outputs when needed. However, DCS network predictions cannot be constantly trusted because locally poor fitting will unavoidably occur due to extrapolation. We pro-

pose the validity index in DCS for validating its prediction performance as an improvement to DCS network models. The implementation of validity index is straightforward and does not require any additional learning. Experimental results were obtained by running tests on failure flight data collected from the IFCS simulator. The computed validity index effectively indicates poor fitting within regions characterized by sparse data. It demonstrates that the validity index is a feasible improvement to DCS and can be applied to validate the DCS performance by predicting with confidence.

References

1. J. Bruske, and G. Sommer. Dynamic cell structure learns perfectly topology preserving map, *Neural Computation*, Vol. 7, No. 4, 1995, 845-865.
2. T. Martinetz, and K. Schulten. Topology representing networks, *Neural Networks*, Vol. 7, No. 3, 1994, 507-522.
3. Institute of Software Research. Dynamic cell structure neural network report for the intelligent flight control system, Technical report, Document ID: IFC-DCSR-D002-UNCLASS-010401, January 2001.
4. J.A. Leonard, M.A. Kramer, and L.H. Ungar. Using radial basis functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks*, 3(4):624-627, July 1992.
5. Teuvo Kohonen. The self-organizing map, *Proc. of the IEEE*, Vol. 78, No. 9, September, 1990, 1464-1480.
6. B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, *Neural Networks*, Vol. 7, No. 9, May 1993, 1441-1460.
7. S. Yerramalla, Y. Liu, E. Fuller, B. Cukic and S. Gururajan. An Approach to V&V of Embedded Adaptive Systems, in *Lecture Notes in Computer Science (LNCS) Proceeding of Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems*, Springer-Verlag, 2004.
8. C.C. Jorgensen. Feedback linearized aircraft control using dynamic cell structures, *World Automation Congress* (ISSCI), Alaska, 1991, 050.1-050.6.
9. The Boeing Company, Intelligent flight control: advanced concept program, Technical report, 1999.
10. M. Napolitano, G. Molinaro, M. Innocenti, and D. Martinelli. A complete hardware package for a fault tolerant flight control system using online learning neural networks. *IEEE Control Systems Technology*, January, 1998.
11. Y. Liu, S. Yerramalla, E. Fuller, B. Cukic and S. Gururajan. Adaptive Control Software: Can We Guarantee Safety? *Proc. of the* 28^{th} *International Computer Software and Applications Conference, workshop on Software Cybernetics*, Hong Kong, September 2004.
12. S. Yerramalla, E. Fuller, B. Cukic. Lyapunov Analysis of Neural Network Stability in an Adaptive Flight Control System, *6th Symposium on Self Stabilizing Systems (SSS-03)*, San Francisco, CA, June 2003.