Borrower: WVU
Received :2/7/2007

Lending String: UPM,UPM,FQG,GSU,*EYW

Patron: Cukic, Bojan

Journal Title: Trends in neural computation /

Volume: **Issue:**
Month/Year: **Pages:** 367-89

Article Author:

Article Title: Performance Analysis of Dynamic Cell Structures

Imprint: Berlin ; Springer, 2007

ILL Number: 27355030

Call #: QA 76.87 .T73 2007

Location: sel

ARIEL
Charge
Maxcost: $25IFM

Shipping Address:
Health Sciences Library
Robert C Byrd Health Sciences Center
P.O. Box 9802
West Virginia University

Fax: 304-293-5995
Ariel: Ariel 157.182.94.203
Odyssey: 157.182.232.29

Go WILD!

Wayne Interlibrary Loan and Document Delivery (WILD)

Chapter 15

PERFORMANCE ANALYSIS OF DYNAMIC CELL STRUCTURES

Yan Liu
Motorola Labs, Motorola Inc., Schaumburg, IL 60196, USA
yanliu@motorola.com

Bojan Cukic
Lane Dept. of Computer Science & Electrical Engineering
West Virginia University, Morgantown, WV 26505, USA
cukic@csee.wvu.edu

Johann Schumann
RIACS / NASA Ames, Moffett Field, CA 94035, USA
schumann@email.arc.nasa.gov

Michael Jiang
Motorola Labs, Motorola Inc., Schaumburg, IL 60196, USA
Michael.Jiang@motorola.com

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

In this paper, we analyze the performance of DCS network by providing sensitivity analysis on its structure and confidence measures on its predictions. We evaluate how the quality of each parameter of the network (e.g., weight) influences the output of the network by defining a metric for parameter sensitivity for DCS network. We present the validity index (VI), an estimated confidence associated with each DCS

Y. Liu et al.: *Performance Analysis of Dynamic Cell Structures*, Studies in Computational Intelligence (SCI) **35**, 367–390 (2007)
www.springerlink.com

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 that our analysis effectively measures the network performance and provides validation inferences in a real-time manner.

Keywords: Dynamic Cell Structures, Validity index, sensitivity analysis, performance estimation, confidence measures, neural networks.

1. Introduction

Often viewed as black box tools, neural network models have a proven track record of successful applications in various fields. In safety-critical systems such as flight control, neural networks are adopted as a major soft-computing paradigm to support on-line adaptation and damage-adaptive control. 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 performance prediction of such models is hard to assure. The uncertainties (low confidence) existing 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 a confidence measure of its predictions.

The Dynamic Cell Structure (DCS) network [1] is designed 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 wide variety of complex data manifolds [2, 3]. The structural flexibility of DCS network has gained it a good reputation of adapting faster and better to a new region than most SOMs [2, 3]. A typical application of DCS is the NASA Intelligent Flight Control System (IFCS)[4]. DCS is employed in IFCS as online adaptive learner and provides derivative corrections as control adjustments during system operation. In this application, it outperforms Radial Basis Function (RBF) and Multi-Layer Perceptron (MLP) network models [5]. As a crucial component of a safety critical system, the DCS network is expected to give quality performance in the entire operational domain.

Relying upon learning/training/approximation, a neural network model raises issues in its quality (e.g., [6]). Two aspects are of importance here: if the model has been trained with a set D of input values X, the model should produce the correct (or almost correct) values for these data. In learning theory, this is called *recall*. On the other hand,

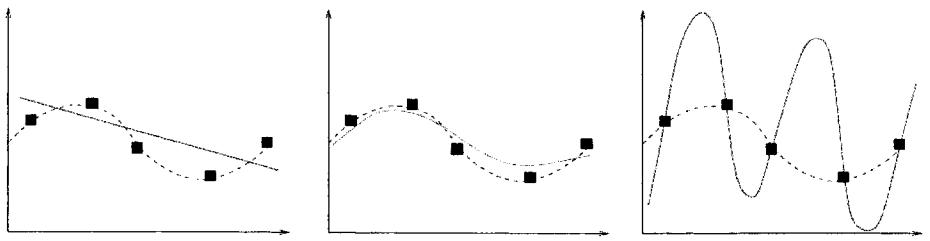

Figure 15.1. Various levels of recall and generalization of the approximation of a sin-curve (dashed line), given 5 points. **A** (linear) approximation: bad recall, reasonable generalization (oversimplification). **B** reasonable recall and generalization. **C** perfect recall, but very bad generalization

the model should also provide reasonable results on inputs, which are not in D. This capability is called *generalization.* Figure 15.1 shows the problem with recall and generalization for a simple sine curve: there exist approximations with very good recall but bad generalization, and vice versa. Most of the neural network based schemes view the problem as deriving model parameter adaptive laws, having chosen a structure for the neural network. However, choosing structure details such as number of basis functions (or hidden units in a single hidden layer) in the model must be done *a priori.* This can often lead to an over-determined or under-determined network structure which in turn leads to an approximation model that is not optimal, i.e., with bad recall and with bad generalization. Methods and guidelines for model selection have been researched and can be found in the neural network literature [7–10]. However, as a dynamically evolving structure, the DCS network is initialized with two connected neurons and then adjusts its own structure to adapt to a better representation of the data. Thus, a DCS network does not require any structure details to be pre-determined. However, an analysis can be done later on its structural representation (e.g., weights of neurons) for a sensitivity estimation with respect to input perturbations, and a confidence measure of network output can be used to estimate the network's generalization ability.

Our sensitivity analysis focuses on how the quality of each parameter of the network influences the output of the network. We define a sensitivity metric for DCS networks, i.e., the partial derivative of the outputs with respect to the inputs. The sensitivity metric can be used to evaluate the quality and the robustness of the model. We propose the Validity Index (VI), as a measure of confidence imposed on each

DCS prediction. Each validity index reflects the confidence level of a particular output.

The paper is organized as follows. Section 2 summarizes related work in validation and verification of neural networks. The architecture of a DCS network and its learning algorithm are described in Section 3. Sensitivity analysis is described in Section 4. The concept of validity index and its statistical computation are explained in detail in Section 5. Section 6 further explains the sensitivity metric and validity index for DCS by experimenting with an artificial data set. Section 7 describes a case study on a real-world control application, the IFCS, and presents experimental results on the validity index and sensitivity analysis of DCS using flight simulation data. Section 8 summarizes the proposed methods and discusses future work.

2. Related Work

Traditional literature describes adaptive computational paradigms, neural networks in particular, with respect to their use, as function approximators or data classification tools. Validation on these systems is usually based on a train-test-re-train empirical procedure. Some bibliographic references also propose methods as part of the training algorithm of neural networks for validation [4, 11]. The ability of interpolating and/or extrapolating between known function values is measured by certain parameters through testing. This evaluation paradigm can be reasonably effective only for pre-trained adaptive systems, which do not require online learning and adaptation and remain unchanged in use. In [12], Fu interprets the verification of a neural network to refer to its correctness and interprets the validation to refer to its accuracy and efficiency. He establishes correctness by analyzing the process of designing the neural network, rather than the functional properties of the final product. Gerald Peterson presents another similar approach in [13] by discussing the software development process of a neural network. He describes the opportunities for verification and validation of neural networks in terms of the activities in their development life cycle, as shown in Figure 15.2

As we can see from Figure 15.2, there is a focus on V&V of neural networks based on the training data. Verification of the training data includes the analysis of appropriateness and comprehensiveness. However, in online learning mode, this technique may not be appropriate due to its real-time training. The data is collected in such a way that the training is completed under intensive computational requirements. Novelty detection is considered an important approach for validating

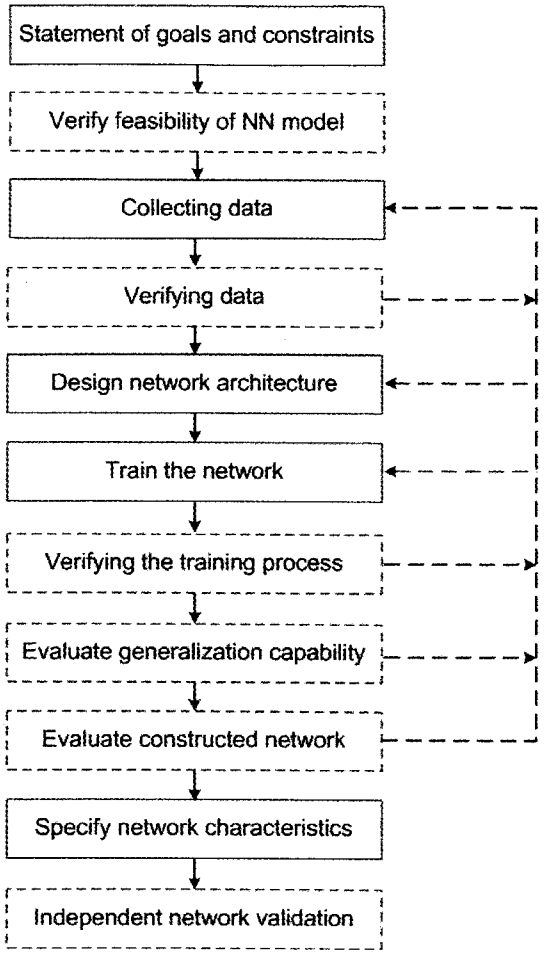

Figure 15.2. The development cycle of a neural network [13].

neural network models [14, 15]. Our parallel research adopts novelty detection techniques for validating a neural network based online adaptive controller [16].

Verification of the training process typically examines the convergence properties of the learning algorithm, which is usually pre-defined by some criteria of error measure. In [17], K.J. Hunt et.al. investigate all different methods for error estimation techniques and make detailed comparisons among them. Nonetheless, effective evaluation methods of interpolation and extrapolation capabilities of the network and domain specific verification activities are still based on empirical testing [18]. Literature addressing this problem analytically is very scarce. In the field of

function approximation theory, MLP networks have been proven to be universal approximators as they are able to achieve any given accuracy provided a sufficient number of hidden neurons [19]. The mathematical analysis and proof can be seen as another effort for validating the learning process as it can provide a theoretical proof for the capabilities of function approximation. The weakness of such an analytical proof is that the number of required hidden neurons is extremely high. Also, for an online adaptive learning systems, where the system function evolves this approach remains impractical.

Most recently proposed techniques on V&V of neural networks are based on empirical evaluation through simulation and/or experimental testing. There are also other approaches to V&V of dynamic neural networks. In an attempt to solve the dilemma of plasticity and stability for neural networks, Grossberg [20, 21] derives a new paradigm, referred to as the Adaptive Resonance Theory (ART-1/2/3). Within such a network, there are two components charging seen and unseen data, respectively. As interesting as is, it provides better understanding for our problem other than applicable tools for validation and verification.

In a survey of methods for validating on-line learning neural networks, O. Raz [22] classifies this approach into on-line monitoring and novelty detection and attributes to it a significant potential for the future use. The other promising research direction, according to Raz, is periodic rule extraction from an on-line neural network (e.g., [23, 24]) and partial (incremental) re-verification of these rules using symbolic model checking [25]. Practical hurdles associated with this approach include determining the frequency of rule extraction and impracticality of near real-time model checking of complex systems [26].

[27] have developed a tool to dynamically estimate the performance of an on-line trained neural network using a Bayesian approach. Dynamical monitoring of the network's current performance is an import step toward V&V of neuro-adaptive systems [28, 29].

The proposed validity index for DCS networks is inspired by J. Leonard's paper on the validation of Radial Basis Function (RBF) neural networks [30]. Leonard developed a reliability-like measure called validity index which statistically evaluates each network output. The validity index in a RBF neural network is a confidence interval associated with each network prediction for a given input. Different from the predefined 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, the

DCS network emphasizes topological representation of the data, while the RBF network does not. By the end of DCS learning, the data domain is divided into Voronoi regions [2]. 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 closest neighbor to the 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 gives the validity index another computational definition that's derived specifically for DCS network.

3. The Dynamic Cell Structure

The Dynamic Cell Structure (DCS) [1, 31] network can be seen as a special case of Self-Organizing Map (SOM) structures. The SOM is introduced by Kohonen [32] and further improved to offer topology-preserving adaptive learning capabilities. 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 [2, 3, 31]. 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. An Euclidean distance metric is adopted for finding both units. The set of neurons connected to the BMU are called its neighbors and denoted by NBR.

The training algorithm of the DCS network combines the competitive Hebbian learning rule with 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:

$$C_{ij}(t+1) = \begin{cases} 1 & (i = \text{BMU}) \wedge (j = \text{SBU}) \\ 0 & (i = \text{BMU}) \wedge (C_{ij} < \theta) \\ & \wedge (j \in \text{NBR} \setminus \{\text{SBU}\}) \\ \alpha C_{ij}(t) & (i = \text{BMU}) \wedge (C_{ij} \geq \theta) \\ & \wedge (j \in \text{NBR} \setminus \{\text{SBU}\}) \\ C_{ij}(t) & \text{otherwise} \end{cases} \tag{15.1}$$

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 $\Delta\vec{w}_i = \vec{w}_i(t+1) - \vec{w}_i(t)$ represent the adjustment of the reference vector needed for neuron i, the Kohonen learning rule followed in DCS computes $\Delta\vec{w}_i$ as follows.

$$\Delta\vec{w}_i = \begin{cases} \varepsilon_{\mathrm{BMU}}(\vec{m} - \vec{w}_i(t)) & (i = \mathrm{BMU}) \\ \varepsilon_{\mathrm{NBR}}(\vec{m} - \vec{w}_i(t)) & (i \in \mathrm{NBR}) \\ 0 & \text{otherwise} \end{cases} \tag{15.2}$$

where $\vec{m}$ is the desired output, and $0 < \varepsilon_{\mathrm{BMU}}, \varepsilon_{\mathrm{NBR}} < 1$ 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 displayed in Figure 15.3. According to the algorithm, N is the number of training examples. Resource values are computed at each epoch as local error measurements associated

```
Initialization;

Repeat until stopping criterion is satisfied;
{
    Repeat N times

    {
        Determine the BMU and SBU;

        Update lateral connections (eq 15.1);

        Adjust the weights (eq 15.2);

        Update resource values;

    }

    If needed, a new neuron is inserted/deleted;

    Decrement resource values;
}
```

Figure 15.3. A brief description of the DCS learning algorithm.

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.

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.

4. Sensitivity Analysis

An important analysis method for any function approximation, e.g., a DCS network, is sensitivity analysis. By calculating the effect of small perturbations of the input on the output, the smoothness and robustness of the function approximator can be assessed. In a sensitivity analysis, the partial derivative of the outputs $\vec{o}$ with respect to the inputs $\vec{x}$, namely $\frac{\partial \vec{o}}{\partial \vec{x}}$ is calculated. Unnecessary high sensitivity of the neural network can lead to problems, in particular, in neuro-adaptive controllers, as effects of perturbations can be amplified by feedback, ultimately leading to oscillation and instability.

While sensitivity analysis offers valuable information on the quality of the model as a black box, it does not provide any information on the impact of variation of the internal network parameters on the output. This measure of parameter sensitivity gives an estimate of the white-box model quality. The information that is contained in the DCS network is stored as a set of parameters, connections C_{ij} and weight vectors $\vec{w}_i$. In order to assess the current quality of the network output, it is important to know, how robust the model is with respect to perturbations of the parameters, i.e., how does the output change if a network weight is changed by, say 1%. If such a small change in a parameter already leads

to a large change in the output, then the selection and tuning of this parameter is highly critical, and should be monitored closely. In the realm of DCS, a overly sensitive neuron could mean that the mechanism for the dynamic introduction of new neurons during training is not set up appropriately. On the other hand, highly insensitive neurons could be safely deleted from the network. Here again, an optimally tuned DCS learning algorithm should have taken care of that situation.

In this paper, we focus on *parameter sensitivity*. We calculate $\frac{\partial \vec{o}}{\partial \vec{w}_i}$ for the neuron reference vectors $\vec{w}_i$, as they play a major role in the network recall mode. Thus, parameter sensitivity can be easily approximated for the DCS network as

$$\frac{\partial \vec{o}}{\partial \vec{w}_i} \approx (R(\vec{x}, \{\vec{w}_1, \ldots, \vec{w}_n\}) - R(\vec{x}, \{\vec{w}_1, \ldots, \vec{w}_i + \Delta, \ldots, \vec{w}_n\}))/\Delta$$

where $R(\cdot)$ is the recall function of the DCS network and Δ is a perturbation.

More information can be obtained if we consider each parameter of the neural network not as a scalar value, but as a probability distribution. Then, we can formulate the sensitivity problem in a statistical way. The probability of the output of the neural network is $p(\vec{o}|\mathcal{P}, \vec{x})$ given parameters $\mathcal{P}$ and inputs $\vec{x}$. If we assume a Gaussian probability distribution, we can define our parameter confidence as the variance $\sigma^2_{\mathcal{P}}$. In contrast to calculating the confidence value of the network output, we do not marginalize over the weights or parameters, but over the inputs.

5. The Validity Index in DCS networks

As a V&V method, a validity check is usually performed through the aide of software tools or manually to verify the correctness of system functionality and the conformance of system performance to predetermined standards. The validity index proposed by J. Leonard [30] is a reliability-like measure for validity checking. The Validity Index (VI) 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 a poor validity index and later be 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 the 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 detailed description of these two steps is as follows:

1. Computation of local error. 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, which have the same BMU 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 showed that the adjustment by competitive Hebbian learning rule concerns connections only between the BMU and its neighbors. The further update of weight values by the Kohonen learning rule is performed only on the BMU and its neighbors. 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.

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

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 the residual due to the fact that each data point itself contributes 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. Estimation of standard error. 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:

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

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. The new algorithm is shown in Figure 15.4. Note that because all needed information is already saved at the final step of each training cycle, we simply calculate $s_i'^2$ for each neuron after the learning stops without any additional cost. When the DCS is in recall mode for prediction, the validity index is computed based on the local errors and then associated with every DCS output. In order to

```
/*DCS Learning (see Figure 15.3*/
...
/* Calculate the validity index */

For every neuron i in the network
{
    For every data point k whose BMU is i

    { Compute E(k) ; }

    Compute the local error s_i^2 using E(k);
}
```

Figure 15.4. The DCS learning algorithm with validity index.

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 predefined to help verify that the accuracy indicated by the validity index is acceptable.

6. An Example with Artificial Data

In order to demonstrate the sensitivity metric and 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 [30]:

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

where $\varepsilon \sim \mathcal{N}(0, \eta)$ is a Gaussian noise and η=0.1 in the following examples.

We sample x's from the interval $[-1, 1]$ randomly. At the beginning of the simulation, the network is initialized with two neurons whose reference vectors represent two randomly selected training data points. The network continues learning and adjusts its own structure to adapt to the data. 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 15.5 shows two sensitivity snapshots at different times of the simulation where the network has been trained with the data. Each neuron is associated with a 2-dimensional sensitivity ellipse. Figure 15.5 (a) shows the situation when the network stops training it has 13 neurons. Figure 15.5 (b) shows the situation when the network stops training it has 27 neurons. In plot (a), more than 50% of the neurons exhibit relatively large sensitivity, while in plot (b) a smaller portion of neurons ($\approx$ 30%) has large sensitivity values.

Meanwhile, at the end of the network training we calculate the validity index values. Figure 15.6 illustrates the validity index for these two DCS models, one with 13 neurons and the other with 27 neurons, shown as Figure 15.6(a) and Figure 15.6(b), respectively. By comparing the prediction performance of these two models using the validity index, which is shown as confidence band in both figures, we can conclude that the DCS network model shown in Figure 15.6 (b) has better prediction performance. Furthermore, we can observe that regions with sparse learning data have low confidence measures.

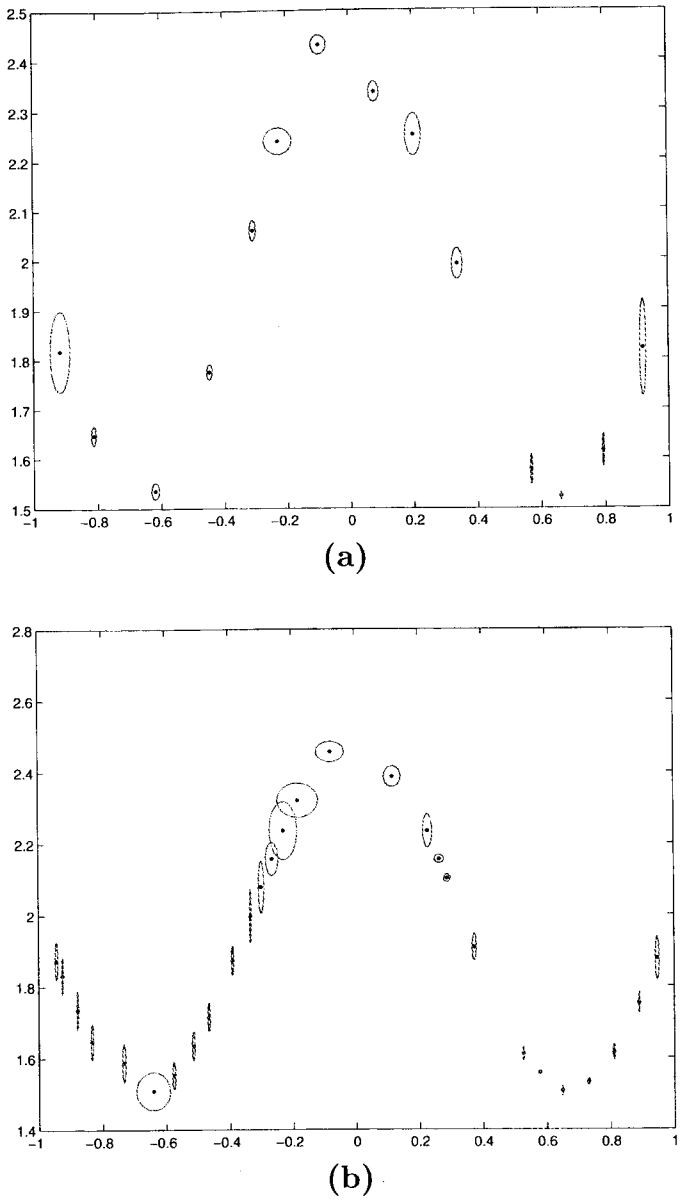

Figure 15.5. Examples of sensitivity metric for a DCS network. (a): The network with 13 neurons. (b): The network with 27 neurons.

7. A Case Study

We conduct the performance analysis of DCS networks for the Intelligent Flight Control System (IFCS). The IFCS is an example of adaptive

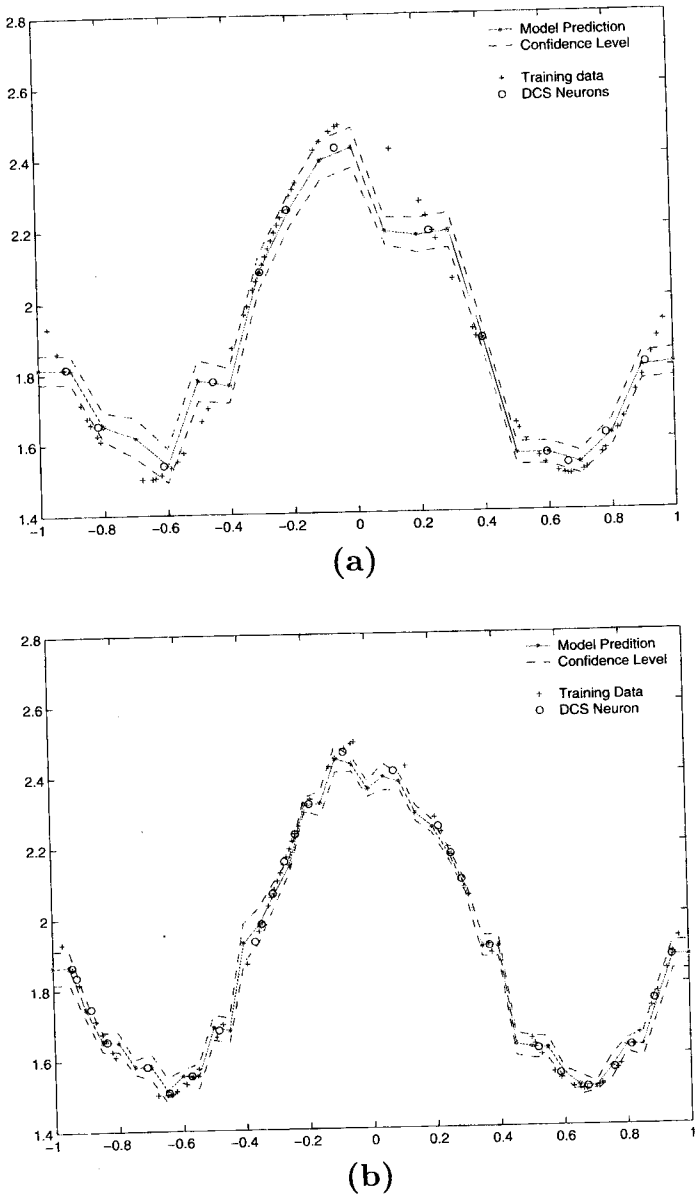

Figure 15.6. Examples of validity index for a DCS network. (a): The model with 13 neurons. (b): The network with 27 neurons.

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 [33].

The Intelligent Flight Control System

The Intelligent Flight Control System 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 scenarios*" [34, 35].

The diagram in Figure 15.7 shows the architectural overview of NASA's first generation IFCS implementation of the online adaptive controller. In this architecture, the proper controller for the aircraft is augmented by two neural networks and a parameter-identification component in the feedback loop. A pre-trained neural network (PTNN), called the Baseline Neural Network stores the data (derivatives) for the nominal mode. A change in the aircraft dynamics due to loss of a control surface (like aileron or stabilator) or due to excessive sensor noise or a sensor failure lead to discrepancies from the outputs of the Baseline Neural Network and the Real-time Parameter Identification (PID) component. In order to obtain a good aircraft dynamics even in the face of failure, notable discrepancies are accounted for by the OLNN (on-line learning neural network). In this architecture, the OLNN is a DCS network. All experiments with this architecture have been carried out with the NASA-WVU F-15 Simulator [36].

The primary goal of the OLNN is to accomplish in-flight accommodation of these 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 the 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.

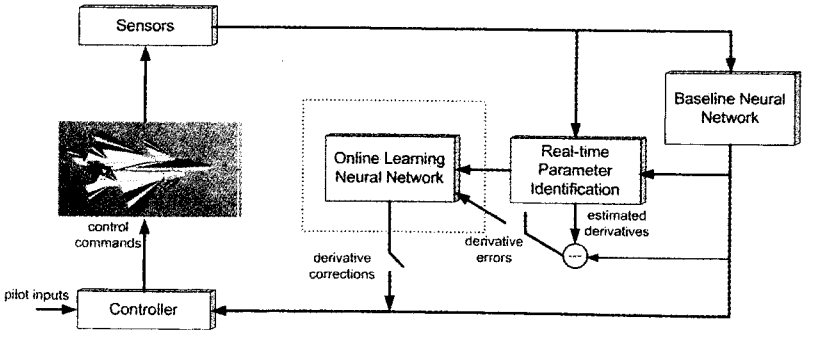

Figure 15.7. Principled Architecture of the Intelligent Flight Control System

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 [33, 37, 38]. 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.

The Sensitivity Metric for DCS Network

Within the IFCS, the DCS network is employed for online adaptation/learning. The DCS parameters (connection strength C_{ij} and reference vectors $\vec{w}_i$) are updated during system operation. It should be noted that the connection strength C_{ij} does not contribute to the network predictions while it is in recall mode. This implies that the sensitivity of the connection strength is merely a structure related parameter that influences the reference vectors instead of the network output. We therefore only measure the sensitivity of the reference vector of the DCS network. Using the simulation data obtained from the IFCS simulator, we calculate the parameter sensitivity s and its confidence σ^2 after each learning epoch during a flight scenario. The sensitivity analysis is conducted on a N-dimension space, where N is the number of dimensions of the input space.

Figure 15.8 shows two sensitivity snapshots at different times of the simulation where the network has been trained with 2-dimensional data. Each neuron is associated with a 2-dimensional sensitivity ellipse. At the beginning of the simulation, the network is initialized with two neurons whose reference vectors represent two randomly selected training data

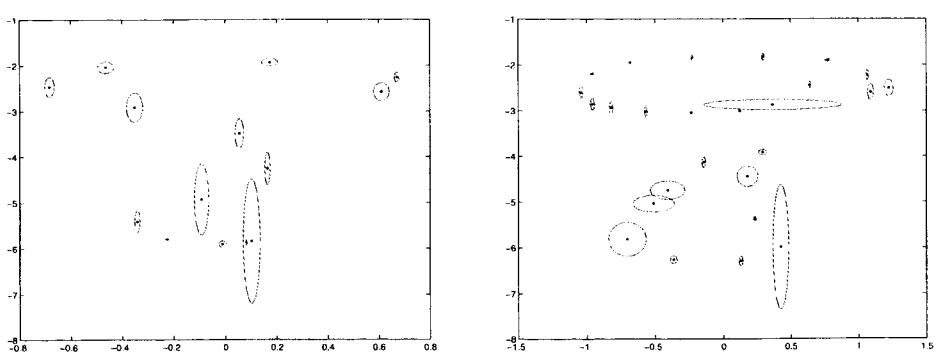

Figure 15.8. Sensitivity analysis for DCS networks

points. The network continues learning and adjusts its own structure to adapt to the data. Figure 15.8 (left) shows the situation at $t = 5.0s$. Figure 15.8 (right) shows the situation at $t = 10.0s$. At $t = 5.0s$, most neurons exhibit relatively large sensitivity, while only a few ($\approx 30\%$) neurons have small sensitivity values. However, at $t = 10.0s$, when the network has well adapted to the data, Figure 15.8 (right) clearly indicates that now most ($\approx 80\%$) neurons have small sensitivity values.

Online Testing of Validity Index

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 output of the PTNN and PID (see Figure 15.7). 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 the DCS network under two different failure mode conditions and calculate the validity index in simulated real-time. The first failure is the stuck-at-surface type of failure, where the aircraft's left stabilator is simulated to be stuck at an angle of +3 degree. The other is the loss-of-surface type of failure, where a 50% loss of the surface at the left stabilator is simulated. Both failures cause the aircraft to start a roll and yaw movement instead of flying a straight line.

In our experiment, simulation runs of 10 seconds were executed; 5 seconds before the failure and 5 seconds after the failure mode was activated. The basic data update rate is 20Hz, this means that each experiment produces 200 data points. Online learning of the DCS within this simulation is accomplished by using a moving window, a buffer, which holds 200 data points, i.e., the data for 10 seconds. This data window is moved every second to incorporate the most recent 20 data points. In each experiment, we first start the DCS network under nominal flight conditions with 200 data points. After that, every second, we first set the DCS network for prediction (it is referred to as the recall mode within IFCS [34, 35]) 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. The DCS network continues learning and repeats the recall-learn procedure.

Figure 15.9 and Figure 15.10 show the experimental results of the simulations on these two failures, respectively. The plots labeled (a)

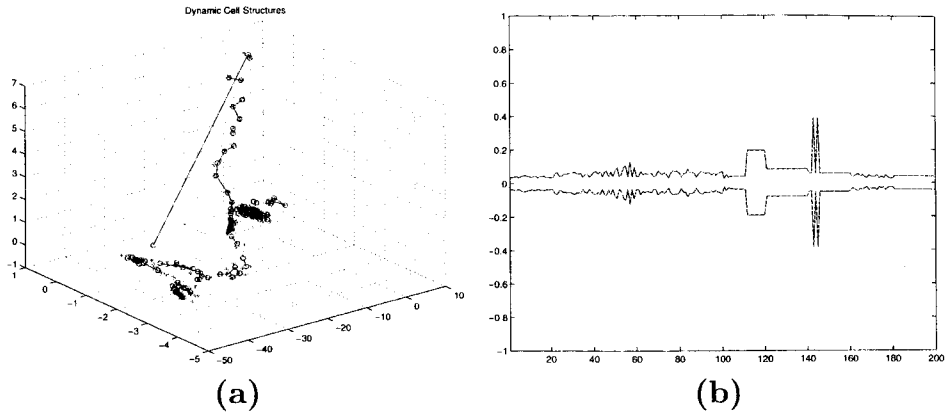

Figure 15.9. A stuck-at-surface failure simulation in real-time (20Hz). (a): The final form of DCS network structures. (b): Validity Index shown as error bars for each DCS output.

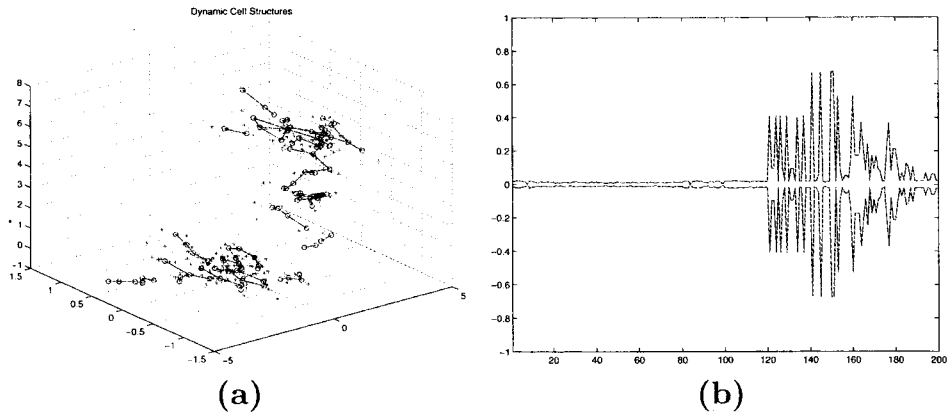

Figure 15.10. Testing on loss-of-surface failure simulation data in real-time. (a): The final form of DCS network structures. (b): Validity Index shown as error bars for each DCS output.

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 selected 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. The plots labeled (b) present the validity index, shown as error bars. The x-axis here represents the time frames in units of 1/20s. In both simulations, the failure occurs at the 100^{th} data frame ($t = 10s$).

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.

8. 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 or a catastrophic failure and to provide reliable outputs when needed. However, DCS network predictions cannot be constantly trusted because locally poor fitting will unavoidably occur due to extrapolation. We propose two approaches to analyze the online prediction performance of DCS network models. The parameter sensitivity is a mathematically simple metric that can be obtained in any phase of network learning. The implementation of validity index is straightforward and does not require any additional learning. Both methods are primarily developed to provide dynamic data on the performance of the DCS network. Experimental results demonstrate that our analysis is capable of calculating a performance index for the DCS neural network during online operation.

Our experimental results further suggest that our analysis provides the basis of validity check for an effective validation of the IFCS as a typical example of a neural network-based online adaptive system. However, in neuro-adaptive control applications, the actual performance of the entire system (in our case study, the aircraft) also depends on a multitude of other parameters (e.g., robustness of controller, performance metric, type of failure). Our future research aims to relate our performance analysis with other aspects of the system quality. With the real-time availability of other quality estimates, our analysis can be used to provide assistance/support to decision making during system operation.

References

[1] Ahrns, I., Bruske, J., Sommer, G.: On-line learning with dynamic cell structure. In: *Proc. of International Conference on Artificial Neural Networks.* Vol. 2. (1995) 141–146

[2] Bruske, J., Sommer,G. : Dynamic cell structure learns perfectly topology preserving map. *Neural Computation.* Vol. 7 (4) (1995) 845–865.

[3] Martinetz, T., Schulten, K.: Topology representing networks, *Neural Networks.* Vol. 7 (3) (1994) 507–522

[4] Boyd, M.A., Schumann, J., Brat, G., Giannakopoulou, D., Cukic, B., Mili, A.: Validation and verification process guide for software and neural nets. Technical report, NASA Ames Research Center. (2001)

[5] 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. (2001)

[6] Reed, R., Marks, R.: Neural Smithing. MIT Press. (1999) 346

[7] Liu, Y.: Neural network model selection using asymptotic jackknife estimator and cross validation. *Advances in Neural Information Processing Systems.* Vol. 5 (1993) 599–606

[8] Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press. (1995)

[9] Lendasse, A., Simon, G., Wertz, V., Verleysen, M.: Fast bootstrap methodology for model selection. *Neurocomputing.* Vol. 64, (2005) 161–181

[10] Kingston, G.B., Maier, H.R., Lambert, M.F.: A Bayesian approach to artificial neural network model selection. In: *Proc. of International Congress on Modelling and Simulation 2005.* (2005) 1853–1859

[11] Tibshirani, R.: Bias, variance and Prediction error for classification rule. Technical Report, Statistics Department, University of Toronto. (1996)

[12] Fu, L.: Neural Networks in Computer Intelligence. (1994)

[13] Peterson, G. E.: A foundation for neural network verification and validation. In: *SPIE Science of Artificial Neural Networks II,* 1966:196–207 (1993)

[14] Bishop, C.M.: Novelty detection and neural network validation. In: *IEE Proceedings: Vision, Image and Signal Processing.* Vol. 141 (4) (1994) 217–222

[15] Roberts, S.J. : Extreme value statistics for novelty detection in biomedical signal processing. In: *IEE Proceedings Science, Technology & Measurement.* Vol. 147 (6) (2000) 363–367

[16] Liu, Y., Cukic, B., Fuller, E., Yerramalla, S., Gururajan, S. : Novelty detection for a neural network based online adaptive control

system. In: *Proc. of the* 29^{th} *International Computer Software and Applications Conference.* (2005)

[17] Hunt, K.J., Sbabaro, D., Zbikowski, R., Gawthrop, P.J.: Neural networks for control systems- a survey. *Automatica.* Vol. 28 (6) (1996) 1707–1712

[18] Lawrence, A., Tsoi, A.C., Back, A.D. : Function approximation with neural networks and local methods: bias, variance and smoothness. In: *Proceedings of Australian Conference on Neural Networks.* (1996) 16–21

[19] Hornik, K.M., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Networks* Vol. 2 (1989) 359–366.

[20] S. Grossberg. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121-134, 1976. Reprinted in Anderson and Rosenfeld, 1988.

[21] Grossberg, S. : Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science.* Vol. 11 (1) (1987) 23–63

[22] Raz, O.: Validation of online artificial neural networks - an informal classification of related approaches. Technical report, NASA Ames Research Center. (2000)

[23] Thrun, S. Extracting rules from artificial neural networks with distributed representations. In: *Advances in Neural Information Processing Systems.* Vol. 7. (1995) 505–512

[24] Taha, I.A., Ghosh, J. : Symbolic interpretation of artificial neural networks. In: *IEEE Transactions on Knowledge and Data Engineering.* Vol. 11. (3) (1999) 448–463

[25] Wen, W., Callahan, J.: Neuralware engineering: develop verifiable ann-based systems. In: *IJSIS.* (1996)

[26] Mili, A., Cukic, B., Liu, Y., Ben Ayed, B.: Towards the verification and validation of online learning adaptive systems. In: *Computational Methods in Software Engineering.* (2003)

[27] Schumann, J., Gupta, P.: Monitoring the performance of a neuro-adaptive controller. In: *MAXENT 2004*, American Institute of Physics. (2004) 289–296

[28] Schumann, J., Gupta, P.,Jacklin, S.: Toward verification and validation of adaptive aircraft controllers. In: *Proc. IEEE Aerospace Conference.* IEEE Aerospace. (2005)

[29] Gupta, P., Loparo, K., Schumann, J., Soares, F.: Verification and validation methodology of real-time adaptive neural networks for

aerospace applications. In: *International Conference on Computational Intelligence for Modeling, Control, and Automation.* (2004)

[30] Leonard, J.A., Kramer, M.A., Ungar, L.H.: Using radial basis functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks*, Vol. 3 (4) (1992) 624–627

[31] Fritzke, B. : Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks.* Vol. 7 (9) (1993) 1441–1460

[32] Kohonen, T. : The self-organizing map. In: *Proc. of the IEEE.* Vol. 78 (9) (1990) 1464–1480

[33] Yerramalla, S., Liu, Y., Fuller, E., Cukic, B., Gururajan, S.: 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.* (2004)

[34] Jorgensen, C.C. : Feedback linearized aircraft control using dynamic cell structures. In: *Proceedings of World Automation Congress.* (1991)

[35] The Boeing Company: Intelligent flight control: advanced concept program. Technical report. (1999)

[36] Napolitano, M., Molinaro, G., Innocenti, M., Martinelli, D.: A complete hardware package for a fault tolerant flight control system using online learning neural networks. *IEEE Control Systems Technology.* (1998)

[37] Liu, Y., Yerramalla, S., Fuller, E., Cukic, B., Gururajan, S. : Adaptive control software: Can we guarantee safety? In: *Proc. of the* 28^{th} *International Computer Software and Applications Conference, workshop on Software Cybernetics.* (2004)

[38] Yerramalla, S., Fuller, E., Cukic, B. : Lyapunov analysis of neural network stability in an adaptive flight control system. In: *Proceedings of 6th Symposium on Self Stabilizing Systems (SSS-03).* (2003)