Tools and Methods for the Verification and Validation of Adaptive Aircraft Control Systems

Johann Schumann[†] and Yan Liu[‡]
[†]RIACS / NASA Ames, Moffett Field, CA 94035
schumann@email.arc.nasa.gov
[‡]Motorola Labs, Schaumburg, IL 60193,
yanliu@motorola.com

Abstract— The appeal of adaptive control to the aerospace domain should be attributed to the neural network models adopted in online adaptive systems for their ability to cope with the demands of a changing environment. However, continual changes induce uncertainty that limits the applicability of conventional validation techniques to assure the reliable performance of such systems. In this paper, we present several advanced methods proposed for verification and validation (V&V) of adaptive control systems, including Lyapunov analysis, statistical inference, and comparison to the well-known Kalman filters. We also discuss two monitoring tools for two types of neural networks employed in the NASA F-15 flight control system as adaptive learners: the confidence tool for the outputs of a Sigma-Pi network, and the validity index for the output of a Dynamic Cell Structure (DCS) network.

Table of Contents

1. Introduction

Adaptive control systems in aerospace applications have numerous advantages. Due to their capability to adapt their internal behavior according to the current aircraft dynamics, they can automatically fine-tune system identification and accommodate for slow degradation and catastrophic failures (e.g., a damaged wing or a stuck rudder) alike. A variety of approaches for adaptive controls, based upon self-learning computational models such as neural networks and fuzzy logic, have been developed (e.g., [19], [21]). Some are in actual use (e.g., in chemical industry) or have been tested (e.g., the NASA Intelligent Flight Control System (IFCS)). However, the acceptance of adaptive controllers in aircraft and other safety-critical domains is significantly challenged by the fact that methods and tools for analysis and verification of such systems are still in their infancy and no widely accepted V&V approach has been developed. Furthermore, the validation of the neural network models is particularly challenging due to their complexity and nonlinearity. Reliability of learning, performance of convergence and prediction is hard to guarantee. The analysis of traditional controllers, which have been augmented by adaptive components require technically deep nonlinear analysis methods.

In this paper, the major characteristics of adaptive control systems and the impact on V&V of such systems are discussed with a specific focus on mathematical analysis, comparison to the well-known Kalman filters, and run-time monitoring. An overview of two monitoring tools for two types of neural networks employed in an adaptive flight controller is presented: the confidence tool for the outputs of a Sigma-Pi network, and the validity index for the output of a Dynamic Cell Structure (DCS) network. Both tools provide statistical inference of the neural network predictions and can give an estimate of the current performance of the network. It should be noted that our tools only provide a performance measure for the network behavior, but not automatically for the entire controller.

2. Neural Network based Flight Control

The approaches introduced in this paper are experimented with the NASA F-15 Intelligent Flight Control System (IFCS) project. The goal of IFCS project is to develop and test-fly a neuro-adaptive intelligent flight control system for a manned F-15 aircraft. Two principal architectures have been developed: the Gen-I architecture uses a DCS neural network as its online adaptive component, the Gen-II architecture a Sigma Pi network.

Figure 1 shows the basic architecture of the Gen-I and Gen-II controllers: pilot stick commands θ_{cmd} are mixed with the current sensor readings θ (e.g., airspeed, angle of attack, altitude) to form the desired behavior of the aircraft. From these data, the PD controller calculates the necessary movements of the control surfaces (e.g., rudder, ailerons) and commands the actuators. The controller incorporates a model of the aircraft dynamics. If

1-4244-0525-4/07/$20.00/©2007 IEEE

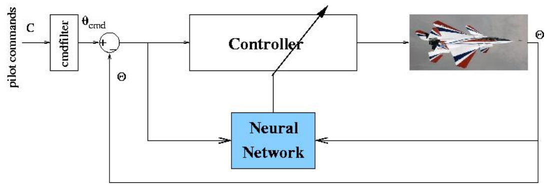

Figure 1. IFCS Generic Adaptive Control Architecture

the aerodynamics of the aircraft changes radically (e.g., due to a damaged wing or a stuck rudder), there is a deviation between desired and actual state. The neural network is trained during operation to minimize this deviation. Whereas in the Gen-I architecture, the appropriate control derivatives are modified with a neural network, Gen-II uses a dynamic inverse controller with control augmentation, i.e., the neural network produces a control correction signal U_{ad}. For details on the control architecture see [21], [5], [22].

The Neural Networks

Dynamic Cell Structure (DCS) Network—The DCS network is derived as a dynamically growing structure in order to achieve better adaptability. DCS can be seen as a special case of Self-Organizing Map (SOM) structures as introduced by Kohonen [12] and further improved to offer topology-preserving adaptive learning capabilities that can respond and learn to abstract from a much wider variety of complex data manifolds [18], [4]. In the IFCS Gen-I controller, the DCS provides derivative corrections during system operation.

The training algorithm of the DCS network combines the competitive Hebbian learning rule and the Kohonen learning rule. The Hebbian learning rule is used to adjust the connection strength C_{ij} between two neurons. The Kohonen learning rule is used to adjust the weight representations of the neurons ($\vec{w}_i$), which are activated based on the best-matching methods during the learning. If needed, new neurons are inserted. After learning, when DCS is used for prediction (recall mode), it will recall parameter values at any chosen dimension. It should be noted that the computation of an output is different from that during training. When DCS is in recall mode, the output is computed based on two neurons for a particular input. One is the best matching unit (bmu) of the input; the other is the closest neighbor (when existing) of the bmu other than the second best matching unit of the input. Since our performance estimation does not depend on the specific learning algorithm, it will not be discussed in this paper. For details on DCS and the learning algorithm see [18], [4], [6], [15].

Sigma Pi Neural Network—The IFCS Gen-II controller uses a Sigma-Pi ($\Sigma\Pi$) neural network [20], where the inputs are $\mathbf{x}$ subjected to arbitrary basis functions (e.g., square, scaling, logistic function). The output of the network o is a weighted sum (Σ) of the Cartesian product of the basis function values (Figure 2):

$$o = \sum_i w_i b_i \quad \text{where} \quad b_i = \prod_j \beta(\mathbf{x}_j)$$

with weights w_i and basis functions $\beta(\mathbf{x}_j)$. Online adaptation (learning) is taking place while the adaptive controller is operating. Figure 3 shows the development of the 60 weights over time. A simulated failure occurs at $t = 1.5s$.

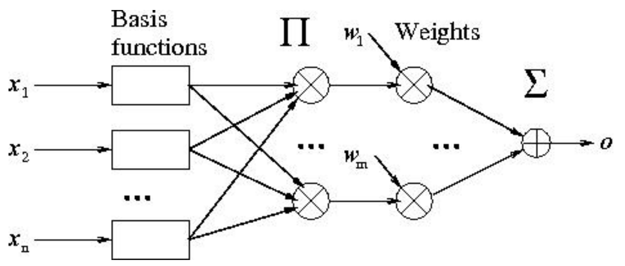

Figure 2. Architecture of $\Sigma\Pi$ network.

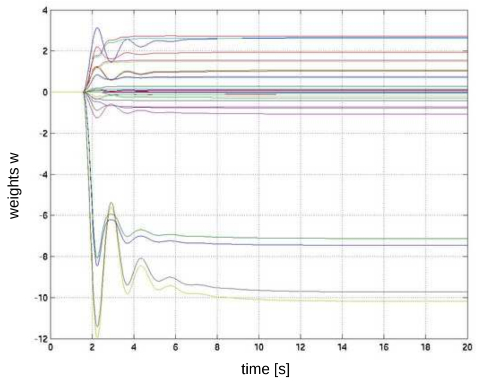

Figure 3. Development of the NN weights over time during adaptation. The failure occurred at $t = 1.5s$.

3. Issues in V&V and Certification of Adaptive Systems

Clearly, an adaptive aircraft controller is a highly safety-critical component of aviation software. Therefore, it has to undergo a rigorous verification, validation, and certification process before such a controller can be deployed. For civil aviation, the standard DO-178B prescribes the process for certification; other institutions, like the NASA, have their own set of standards. In all cases, the certification process has to make sure that the piece of software (as a part of the overall system) performs safely and does not produce any risks. Extended testing and detailed documentation of the entire software development process are key ingredients of each certification process, making certification a costly and highly time-consuming process. Also, certification authorities are very reluctant to certify novel components, architectures, and software algorithms. In particular, for advanced adaptive control algorithms, no standardized way of performing performance analysis and V&V exists. In the following, major characteristics of neuro-adaptive control algorithms and their implication on analysis and verification are discussed.

In essence, the adaptive component (neural network) can be seen as a multivariate function (or look-up table) with non-constant table entries. The learning algorithm itself is a variant of a multivariate (quadratic) optimization procedure. The goal of the adaptation is to minimize the error $\mathbf{e}$ between the actual and desired behavior of the aircraft. In the case of the IFCS, the error (for each axis) is defined as the actual error and the error of the derivatives.

$$\mathbf{e} = \begin{pmatrix} \dot{\theta} - \dot{\theta}_{cmd} \\ \theta - \theta_{cmd} \end{pmatrix}$$

The learning algorithm tries to adapt the weights $\mathbf{W}$ (i.e., w_i for $\Sigma\Pi$, C_{ij} and $\vec{w}_i$ for DCS) the network in such a way that the error $\mathbf{e}$ becomes minimal. Such an algorithm, embedded in a (traditional) PD or PID controller poses some important issues with respect to V&V, which will be discussed in the following.

• One of the most important performance criteria of a controller is it's stability and robustness (i.e., stability in the presence of perturbation or damage). For the practical analysis of an aircraft controller, a large body of linear analysis methods and tools exist (e.g., [24]). However, modeling of damage as well as the adaptive component results in a *nonlinear* system, making the use of linear analysis methods in general impossible. A well-known non-linear analysis technique, Lyapunov stability, will be discussed below.
• Multivariate optimization algorithms have two unpleasant characteristics: they are not guaranteed to reach the global minimum (it can only be proven that they reach a local minimum), and it may take an arbitrary amount of time for the algorithm to converge to that optimum. Both are highly undesirable in a safety-critical real-time system. Next section presents tools which can dynamically analyze the quality of the current solution at any time during the learning process. Such tools can help to provide important performance estimates even in the absence of hard limits.
• Often, adaptive components like neural networks are considered to be *non-deterministic* systems. Except for cases, where initial weights are set to random values (which is not the case in our neural networks), adaptive controllers are fully deterministic. However, the current system status always depends on the entire history, not just the previous state of the aircraft (as holds for Markov processes). Thus, different techniques for analysis and verification are needed.
• Technology available for analysis and monitoring only deals with the adaptive neural network, but not necessarily with the entire system. For the assessment of the performance of an aircraft, its handling qualities (e.g., measured using the Cooper-Harper rating [10], [25]) is highly important. Software certification has to answer the question of how the dynamic adaptation such as failure accommodation influences the aircraft handling qualities. Guarantees must be provided such that a certain level of handling quality is always available.

4. Analysis for V&V

As a prominent example of nonlinear analysis techniques useful for adaptive control, Lyapunov analysis (control theory) provides results on the stability of the controller and Extended Kalman Filters (EKF, a statistical filter often used for GN&C applications) for the analysis of the neural network. Both analysis methods provide actual NN learning algorithms.

Lyapunov Analysis

Stability of a control system is of particular importance for the safe operation of an aircraft. In a nutshell, stability means that every bounded input produces a bounded output. For linear control systems, various methods for stability analysis are available, e.g., the Routh Hurwitz criterion, the root-locus method, or Nyquist's method. Damage-adaptive control systems, however, are nonlinear, so these analysis techniques cannot be used. One of the most popular methods for stability analysis of nonlinear controllers is the Lyapunov stability analysis. Here, an energy-like function over time L (Lyapunov candidate) must be found, which exhibits specific properties with regard to L and $\dot{L}$. If such a Lyapunov function can be constructed it can be shown that the system is stable when the time reaches infinity (asymptotic stability).

For V&V purposes, this method has a number of advantages: if the Lyapunov function is defined with respect to the adaptive parameters (in our case, the neural network weights $\mathbf{W}$), then a learning rule can be extracted easily. More specifically, for the IFCS, a typical Lyapunov candidate function has the form ([21])

$$L(e, \tilde{\mathbf{W}}) = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2\gamma}\tilde{\mathbf{W}}^T\tilde{\mathbf{W}}$$

with a parameter (learning rate)$\gamma > 0$ and a matrix $\mathbf{P}$. For the system to be asymptotically stable, $L > 0$ and $\dot{L} < 0$ must hold. After some calculations (see [21] for details), an equation for the updates of the weights $\dot{\mathbf{W}}$ can be obtained: $\dot{\mathbf{W}} = -\gamma s \beta(\mathbf{x})$, where β are the network basis functions,

$$s = \frac{1}{2K_P}\tilde{\theta} + \frac{1 + K_P}{2K_P K_D}\dot{\tilde{\theta}}$$

where K_P and K_D is the proportional and derivative gain, respectively. It is easy to see that this update equation can be seen as a discrete gradient descent learning method of the form $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta\nabla$ with gradient ∇. Results of this analysis exhibit some restrictions:

- only asymptotic stability is guaranteed, i.e., this analysis does not cover issues of *convergence speed*,
- the detailed analysis in [21] shows that stability is only guaranteed within certain error bounds, and
- this analysis method does not cover performance-oriented aspects (like gain and phase margins, or system-wide properties like aircraft handling quality).

Thus, an analytical method like Lyapunov's stability analysis is an important V&V step, but additional techniques need to be applied during V&V of adaptive controllers.

Extended Kalman Filters—Whereas the application of neural networks and their learning algorithms in safety critical applications is relatively new, another, strongly related technology has been around for a long time: Kalman filters. A Kalman Filter is a recursive linear least squares optimization algorithm (for linear systems). Given a model of the process dynamics and (noisy) measurements, a Kalman Filter (KF) can calculate the statically best possible estimation of the state. Traditionally, KFs are used for navigation, where a number of measurements (from different sensors like GPS, compass, odometer) are combined to a position fix. Developed in the early 1960's, KFs are nowadays part of every aircraft navigation system, spacecraft GN&C, and every GPS receiver. Numerous extensions have been made, and there exists a solid body of engineering knowledge on how to design, V&V and certify a Kalman Filter. For an overview of Kalman Filters see e.g., [3].

The intimate relationship between Kalman Filters and Neural Network training algorithms comes with a Bayesian view of the matter [2]. Following discussion will show that a powerful neural network learning algorithm can be expressed in terms of the well-established Kalman Filter technology. Although this approach is not new and has been used for various applications (e.g., [26], [13], [11]) its application to adaptive flight control provides two major benefits:

- the KF learning algorithm automatically provides a dynamic quality-of-learning measure to indicate, how the learning is progressing. These quality metrics are discussed in conjunction with the monitoring tools in following sections.
- theory and engineering knowledge and experience on KFs are available, so all available techniques for analysis, V&V, and certification can be used "as is"; no new algorithms and paradigms have to be introduced.

On the down-side, however, the KF-based learning algorithm has somewhat higher computational requirements than a simple learning algorithm like gradient descent.

One of the major characteristics of the NN architectures used for adaptive control their nonlinear behavior, caused, e.g., in the $\Sigma\Pi$ network by nonlinear basis functions and the Cartesian product. Therefore, the standard KF, which is for linear systems, cannot be used. Extended Kalman Filters (EKF), however, can be used to estimate nonlinear processes. Here, essentially, a piecewise linearization around the current state estimate is used (for details see [3]).

For the purpose of this study, EKF is not adopted to estimate the state of the aircraft or the output of the NN. Instead, it is used to estimate all the adjustable parameters $\mathbf{W}$ of the neural network. The network weights $\mathbf{W}$ are thus defined as the state vector (usually called $\mathbf{x}$). This is a major difference to traditional state estimation problems, where the actual physical state (e.g., angle of attack, velocity) are estimated. Our learning task is now to *estimate* a set of weights $\hat{\mathbf{W}}$ in such a way that it minimizes the network output $h(\mathbf{W}, \mathbf{x})$ and the required output $\mathbf{z}$. Thus, process and measurement model (in the discrete form) is given by

$$\begin{array}{lcl} \mathbf{W}_{t+1} & = & \mathbf{W}_t + \eta_t \\ \mathbf{z}_t & = & h(\mathbf{W}_t, \mathbf{x}_t) + \nu_t \end{array}$$

where $h(\mathbf{W}_t, \mathbf{x}_t)$ is the output of the network at time t and η^t and ν^t are (Gaussian distributed) process and observation noise vectors, respectively. The Extended Kalman Filter algorithm then is defined in the usual way ([3], [11]) with recursive temporal and measurement update equations

$$\begin{array}{lcl} \mathbf{P}_t^- & = & \mathbf{P}_{t-1}^+ + \mathbf{Q}^t \\ \mathbf{K}_t & = & \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{R}_t + (\mathbf{H}_t)\mathbf{P}_t^- \mathbf{H}_t^T)^{-1} \\ \mathbf{P}_t^+ & = & \mathbf{P}_t^- (I - \mathbf{K}_t \mathbf{H}_t^T) \\ \hat{\mathbf{W}}_t & = & \hat{\mathbf{W}}_{t-1} + \mathbf{K}_t(\mathbf{z}_t - h(\mathbf{W}_{t-1}, \mathbf{x}_t)) \end{array}$$

where $\mathbf{H}$ is the Jacobian of the output with respect to the weights $(\frac{\partial \mathbf{o}_i}{\partial \mathbf{w}_j})$, and $\mathbf{Q}$ and $\mathbf{R}$ the observation and process covariance matrix, respectively. The matrix $\mathbf{K}_t$ is called the Kalman gain, indicating how much the new training data influence the weights $\hat{\mathbf{W}}_t$. In the case of the IFCS $\Sigma\Pi$ architecture, $\mathbf{z}_t$ is not directly available. Hence, the filter is formulated using the control error $\mathbf{e}$. Figure 4 shows the development of the Kalman filter gains K over time for a simulation scenario similar to that of Figure 3.

During each iteration, a new estimate $\hat{\mathbf{W}}$ is estimated, which minimizes the error $\mathbf{e}$. In general, this learning algorithm can converge much faster than a standard gradient descent algorithm. The diagonal elements of the covariance matrix $\sigma = \mathrm{diag}(\mathbf{P})$ provide a quality metric for each weight $\mathbf{w}_i$ in the form of an error bar. A small value of σ_i means that the neural network is confident in weight $\mathbf{w}_i$, large values indicate that the problem at hand could not be learned yet, due to insufficient training or

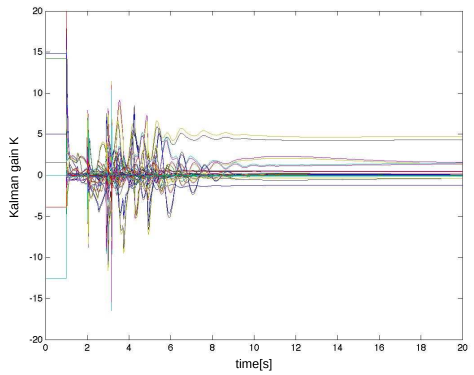

Figure 4. Development of the Kalman Filter gains K over time during adaptation. The failure occurred at $t = 1.5s$.

inability to learn. As will be discussed later, this confidence measure or validity index can play an important role for V&V of the neural network. The well-known and understood problem of numerical instability of the Kalman filter, caused by calculation of the matrix inverse can be overcome, e.g., by using UD-factorization [26], or the Bierman update [8].

5. Advanced Testing and Monitoring Tools

Parameter Sensitivity Analysis

For the analysis of any controller's behavior it is important to estimate its sensitivity with respect to input perturbations. A badly designed controller might amplify the perturbations, which could lead to oscillations and instability. The higher the *robustness* of the controller, the less influence arises from input perturbations. It is obvious that such a metric (i.e., $\frac{\partial \mathbf{o}}{\partial \mathbf{x}}$ for outputs $\mathbf{o}$ and inputs $\mathbf{x}$) is also applicable to an adaptive control system. For an adaptive component, like a neural network, the estimation of the sensitivity is a "black box" method, i.e., no knowledge about the internal structure or parameters is necessary.

During training of the network, the network parameters are adjusted to minimize the training error. Depending on the architecture of the adaptive controller, the network can be pre-trained, i.e., the parameters are determined during the design phase ("system identification"), or the parameters are changing while the system is in operation ("online adaptation"). The *parameter sensitivity* for a neural network model can be computed by $\frac{\partial \mathbf{o}}{\partial p}$ for each of the adjustable parameters $p \in \mathcal{P}$. For a neural network, $\mathcal{P}$ is comprised of the network weights w_i, for the DCS network, it is the reference vectors of the neurons $\vec{w}_i$.

More information can be obtained if each parameter of the neural network is considered not as a scalar value, but as a probability distribution. Then, the sensitivity problem can be formulated statistically. The probability of the output of the neural network is $p(\mathbf{o}|\mathcal{P}, \mathbf{x})$ given parameters $\mathcal{P}$ and inputs $\mathbf{x}$. Assuming a Gaussian probability distribution, the parameter confidence can be obtained as the variance $\sigma^2_{\mathcal{P}}$. In contrast to calculating the network output confidence value, the parameter sensitivity does not marginalize over the weights, but over the inputs.

A Sensitivity Metric for DCS Networks— Within the IFCS Gen-I, the DCS networks are employed for online adaptation/learning. Their parameters (connection strength C_{ij} and reference vectors $\vec{w}_i$) are updated during system operation. Since the parameters C_{ij} do not contribute to the network output during recall mode, we therefore only measure the sensitivity of the reference vector of the DCS network. Using the simulation data obtained from the IFCS Gen-I simulator, the parameter sensitivity s and its confidence σ^2 after each learning epoch during a flight scenario can be calculated. The sensitivity analysis has been conducted on a N-dimension space, where N is the number of dimensions of the input space.

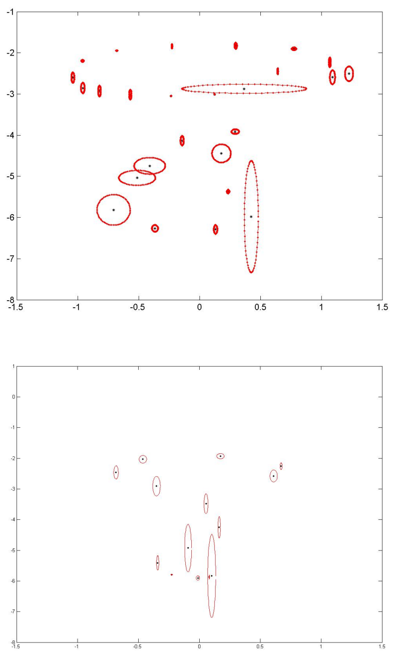

Figure 5. Sensitivity analysis for DCS networks

Figure 5 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 points. The network continues learning and adjusts its own structure to adapt to the data. Figure 5 shows the situation at $t = 5.0s$ (top) and $t = 10.0s$ (bottom). At $t = 5.0s$, most neurons exhibit relatively large sensitivity, while only a few

(31%) neurons have small sensitivity values. However, at $t = 10.0s$, when the network has well adapted to the data, Figure 5 (bottom) clearly indicates that now most (78%) neurons have small sensitivity values.

A Sensitivity Metric for Sigma-Pi Networks—For Sigma-Pi network of the IFCS Gen-II controller, the parameter sensitivity s and its confidence σ^2 for the network parameters w_i at each point in time during a flight scenario are computed. Figure 6 shows two sensitivity snapshots at various stages of the scenario. At the beginning of the scenario, all parameters of the network are set to zero, giving (trivially) in the same sensitivity. At $t = 1.5$, a failure is induced into the system. In order to compensate for the failure, the network weights adapt. Figure 6(top) shows the situation at $t = 5.0s$. A considerable amount of adaptation and weight changes has taken place already. However, the confidence for each of the 60 neurons is still relatively small, as indicated by the large error bars. After approximately 20 seconds, the neural network is fully trained. Figure 6(bottom) now shows quite different values for the sensitivity. Whereas the sensitivity for most of the neurons is really small now, a few (here 7) neurons exhibit high sensitivity. Although their σ^2 is somewhat larger than that for the other neurons, a clear distinction between the different groups can be made.

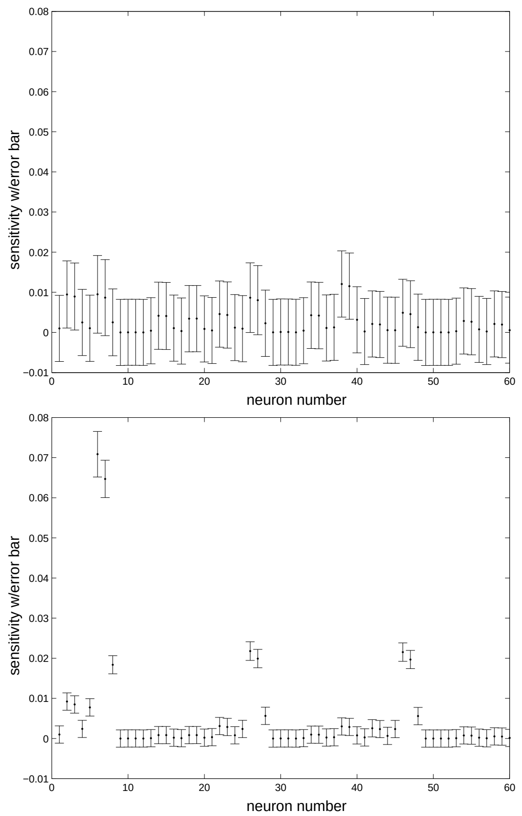

Figure 6. Parameter sensitivity and confidence at $t = 5s$ (top) and $t = 20s$ (bottom).

Network Confidence

Validity Index—Following the definition of Validity Index (VI) in RBF networks by Leonard et.al.[14], the validity index in DCS networks is defined as an estimated confidence measure of a DCS output, given the current input. The VI can be used to measure the accuracy of the DCS network fitting and thus provide inferences for future validation activities. Based on the primary rules of DCS learning and properties of the network structure, the validity index in DCS can be computed using the confidence intervals and variances. The computation of a validity index for a given input consists of two steps: (1) compute the local error associated with each neuron, and (2) estimate the standard error of the DCS output for the given input using information from step (1). Details can be found in [16], [15].

For the calculation of the validity index, the DCS training algorithm needs to be slightly modified, because all necessary information is present at the final step of each training cycle. In recall mode, the validity index is computed based on the local errors and then associated with every DCS output. The online learning of the DCS network is simulated under a failure mode condition. Running at 20 Hz, the DCS network updates its learning data buffer (of size 200) at every second and learns on the up-to-date data set of size 200. The DCS network was first started under nominal flight conditions with 200 data points. After that, every second, the DCS network is set to recall mode and calculates the derivative corrections for the freshly generated 20 data points, as well as their validity index. Then the DCS network is set back to the learning mode and updates the data buffer to contain the new data points.

Figure 7 shows the experimental results of our simulation on the failure mode condition. The top plot shows the final form of the DCS network structure at the end of the simulation. 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 bottom plot presents the validity index, shown as error bars. The x-axis here represents the time frames. The failure occurs at $t = 5.0s$. The validity index is computed for the data points that are generated five seconds before and five seconds after the failure occurs.

A trend revealed by the validity index in our simulations is the increasingly larger error bars after the failure occurs. At $t = 6.0s$, the network has learned these 20 failure data points generated from $\Delta t = 5.0 \sim 6.0s$. The network performance became less stable. After that, the error bars start shrinking while the DCS network adapts to the new domain and accommodates the failure. After the failure occurs, the change (increase/decrease) of the

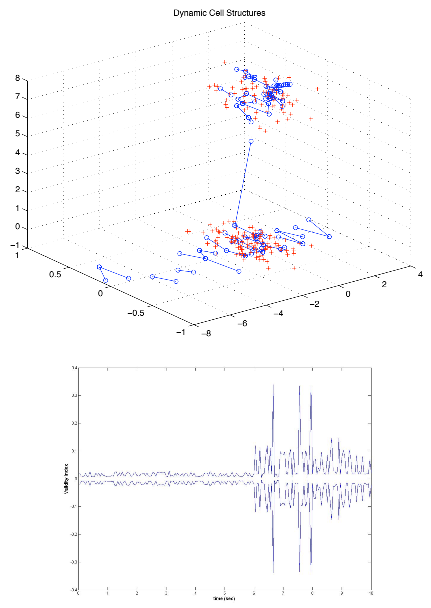

Figure 7. Online operation of DCS VI on failure mode simulation data.

validity index varies depending on the characteristics of the failure as well as the accommodation performance of the DCS network. In this sense, the validity index provides inferences for indicating how well and how fast the DCS network accommodates the failure.

Confidence Tool— For the Gen-II architecture, the ***Confidence Tool*** (CT) [9] produces a quality measure of the neural network output. Our performance measure is the probability density $p(o|\mathbf{x}, D)$ of the network output o given inputs $\mathbf{x}$, when the network has been trained with training data D. Assuming a Gaussian distribution,the standard deviation σ^2 is used as a performance measure. A small σ^2 (a narrow bell-shaped curve) means that, with a high probability, the actual value is close to the returned value. This indicates a good performance of the network. A large σ^2 corresponds to a shallow and wide curve. Here, a large deviation is probable, indicating poor performance.

The confidence tool uses an algorithm, following the derivation in [2] and has been implemented for Sigma-Pi and multi-layer perceptron (MLP) networks in Matlab and in C. Test flights with the Gen-II Sigma-Pi adaptive controller and the Confidence Tool have been successfully carried out in early 2006.

Figure 8 shows the results of a (Simulink) simulation experiment. In the top panel, σ^2 is shown over time. At time $t = 1.0s$, the pilot issues a doublet command (fast stick movement from neutral into positive, then negative and back to neutral position; Fig. 8*(lower panel)*). Shortly afterwards ($t = 1.5s$), one control surface of the aircraft (stabilizer) gets stuck at a fixed angle ("the failure"). Because the system dynamics and the model behavior do not match any more, the neural network has to produce an augmentation control signal to compensate for this deviation. The σ^2 of the network output increases substantially, indicating a large uncertainty in the network output. Due to the online training of the network, this uncertainty decreases very quickly.

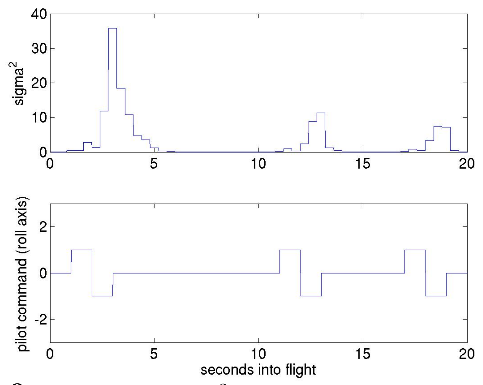

Figure 8. Confidence value σ^2 over time *(top)* and pilot commands for roll axis (*bottom*). A failure has occurred at $t = 1.5s$.

A second and third pilot command (identical to the first one) is executed at $t = 11s$, and $t = 17s$, respectively. During that time, the network's confidence is still reduced, but much less than before. This is a clear indication that the network has successfully adapted to handle this failure situation.

6. CONCLUSIONS

Adaptive control systems can increase safety and performance of an aircraft as it can adapt to accommodate slow degradation and catastrophic failures (e.g., a stuck control surface). Neural networks with suitable machine learning algorithms are often used as the core components of an adaptive controller. Since such systems are highly safety-critical, rigorous methods for V&V and certification are needed. However, the nonlinearity of an adaptive controller and the iterative nature of the learning algorithm makes traditional rigorous (linear) analysis techniques difficult and useless.

In this paper, we have discussed major issues that arise during the analysis and V&V of an adaptive controller. We have presented several analysis techniques and tools that dynamically monitor the behavior and performance of the network. These tools are not only useful during V&V, but also have been incorporated into the actual flight software to monitor the online network's behavior in real time. When put in the right perspective with respect to traditional control and state estimation algorithms, our monitoring tools can effectively analyze neural network based adaptive control systems and provide help for system certification consideration.

REFERENCES

[1] Ahrns, I., Bruske, J., Sommer, G.: On-line Learning with Dynamic Cell Structures. In: Proc. Artificial Neural Networks, Vol. 2. EC2, France (1995) 141-146

[2] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK (1995)

[3] Brown, R., Hwang, P.: *Introduction to Random Signals and Applied Kalman Filtering*. Wiley (1997)

[4] Bruske, J., Sommer, G.: Dynamic Cell Structures. In: NIPS, Vol. 7. (1995) 497-504

[5] Calise A., Rysdyk. R.: Nonlinear Adaptive Flight Control Using Neural Networks. IEEE Control Systems Magazine 21(6) (1998) 14-26

[6] Fritzke, B.: Growing Cell Structures - a Self-organizing Network for Unsupervised and Supervised Learning. Neural Networks 7(9) (1993) 1441-1460

[7] Fuller, E., Yerramalla, S., Cukic, B.,Gururajan, S.: An Approach to Predicting Non-deterministic Neural Network Behavior. In: Proc. IJCNN (2005)

[8] Grewal, M. and Andrews, A.: *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley (2001)

[9] Gupta, P., Schumann, J.: A Tool for Verification and Validation of Neural Network Based Adaptive Controllers for High Assurance Systems. In: Proc. High Assurance Software Engineering, IEEE Press (2004)

[10] Harper, R., Cooper, G.: Handling qualities and pilot evaluation. *J. Guidance, Control, and Dynamics*, 9 (1986)515–529

[11] Haykin, S. (ed): *Kalman Filtering and Neural Networks*. Wiley (2001)

[12] Kohonen, T.: Self-Organizing Maps. Springer (1997)

[13] Lary, D., Mussa, H.: Using an extended Kalman filter learning algorithm for feed-forward neural networks to describe tracer correlations. *Atmos. Chem. Phys. Discuss.*, 4:3653–3667, 2004.

[14] 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 3(4) (1992) 624-627

[15] Liu, Y.: Validating A Neural Network-based Online Adaptive System. PhD thesis, West Virginia University, Morgantown (2005)

[16] Liu, Y., Cukic, B., Jiang, M., Xu, Z.: Predicting with Confidence - An Improved Dynamic Cell Structure. In: Wang., L, Advances in Neural Computation, Vol. 1, Springer (2005) 750-759

[17] Mackall, D., Nelson, S., Schumann, J.: Verification and Validation of Neural Networks of Aerospace Applications. Technical Report CR-211409, NASA (2002)

[18] Martinez, T., Schulten, K.: Topology Representing Networks. Neural Networks 7(3) (1994) 507-522

[19] Norgaard, M., Ravn O., Poulsen, N., Hansen, L.K.: Neural Networks for Modeling and Control of Dynamic Systems. Springer (2002)

[20] Rumelhart, McClelland, and the PDP Research Group: Parallel Distributed Processing. MIT Press (1986)

[21] Rysdyk R.,Calise, A.: Fault Tolerant Flight Control via Adaptive Neural Network Augmentation. AIAA-98-4483 (1998) 1722-1728

[22] Schumann, J., Gupta, P.: Monitoring the Performance of A Neuro-adaptive Controller. In: Proc. MAXENT, AIP (2004) 289-296

[23] Schumann, J., Gupta, P., Jacklin. S.: Toward Verification and Validation of Adaptive Aircraft Controllers. In: Proc. IEEE Aerospace Conference. IEEE Press (2005)

[24] Tischler, M., Lee, J., Colbourne, J.: Optimization and comparison of alternative flight control system design methos using a common set of handling-qualities criteria. In *AIAA Conference* (2001)

[25] Tseng, C., Gupta, P., Schumann, J.: Performance analysis using a fuzzy rule base representation of the cooper-harper rating. In *Aerospace Conf.* IEEE (2006)

[26] Zhang, Y. and Li, X.R.: A fast u-d factorization-based learning algorithm with applications to nonlinear system modeling and identification. *IEEE Transactions on Neural Networks*, 10(4) (1999)930–938

Dr. Johann Schumann *(PhD 1991, Dr. habil 2000, Munich, Germany) is a Senior Scientist with RIACS and working in the Robust Software Engineering Group at NASA Ames. He is engaged in research on verification and validation of autonomy software and adaptive controllers, and on automatic generation of reliable code for data analysis and state estimation. Dr. Schumann is author of a book on theorem proving in software engineering and has published more than 70 articles on automated deduction, automatic program generation, and neural network oriented topics.*

Yan Liu *received the BS degree in computer science from Wuhan University, China, and the MS and PhD degrees in computer science from West Virginia University. She is currently a research scientist at Motorola Labs, Motorola Inc. Her research interests are in the areas of software V&V, machine learning and statistical learning.*