2008 Guidance, Navigation, and Control Conference, Honolulu, Hawaii, Aug 18-21, 2008

Closing the Certification Gaps in Adaptive Flight Control Software

Stephen A. Jacklin, NASA Ames Research Center

Abstract

Over the last five decades, extensive research has been performed to design and develop adaptive control systems for aerospace systems and other applications where the capability to change controller behavior at different operating conditions is highly desirable. Although adaptive flight control has been partially implemented through the use of gain-scheduled control, truly adaptive control systems using learning algorithms and on-line system identification methods have not seen commercial deployment. The reason is that the certification process for adaptive flight control software for use in national air space has not yet been decided. The purpose of this paper is to examine the gaps between the state-of-the-art methodologies used to certify conventional (i.e., non-adaptive) flight control system software and what will likely to be needed to satisfy FAA airworthiness requirements. These gaps include the lack of a certification plan or process guide, the need to develop verification and validation tools and methodologies to analyze adaptive controller stability and convergence, as well as the development of metrics to evaluate adaptive controller performance at off-nominal flight conditions. This paper presents the major certification gap areas, a description of the current state of the verification methodologies, and what further research efforts will likely be needed to close the gaps remaining in current certification practices. It is envisioned that closing the gap will require certain advances in simulation methods, comprehensive methods to determine learning algorithm stability and convergence rates, the development of performance metrics for adaptive controllers, the application of formal software assurance methods, the application of on-line software monitoring tools for adaptive controller health assessment, and the development of a certification case for adaptive system safety of flight.

1.0 Introduction.

Over the last five decades, extensive research has been performed to design and develop adaptive control systems for aerospace systems and other applications where the capability to change controller behavior at different operating conditions is highly desirable. An adaptive controller changes its behavior by allowing the controller forward and/or feedback gains to be adjusted once the controller has been deployed.[1-5] Because designing such a control system introduces many complexities, it is generally held to be good practice to use a non-adaptive or "classical" controller design if one can be found that delivers acceptable performance. This is because although proven techniques to evaluate the dynamic response and controller stability exist for non-adaptive controllers (e.g., root-locus, Bode plots, Nichols charts, etc.[6, 7]), techniques for adaptive systems are only yet in their infancy.

The most prevalent adaptive flight control system technology in use today are those that involve the use of a technique called gain scheduling.[6] To implement this scheme, a classical non-adaptive controller is designed first, and then a number of controller gain sets are determined and stored for a finite number of operating conditions and aircraft configurations. The flight control computer is programmed to select the correct gains based on the current flight condition (airspeed, altitude, etc.) and vehicle configuration. At each specific (constant) flight condition, the controller is a classical, non-adaptive controller. Therefore, the individual controller gain sets can be verified and validated by simulation and flight testing at the specific flight condition for which they were chosen. Within each flight regime, the controller performance is tuned using many well-established techniques for non-adaptive flight control systems. Gain-scheduling thereby offers a means of partial adaptive control capability, while avoiding many of the problems associated with fully adaptive flight control systems. By using a fine enough grid of flight conditions and vehicle configurations, virtually any pre-defined set of flight conditions can be handled using the gain scheduling method. The success of the gain scheduling part depends in large part on the degree to which the system can be rightly characterized to operate in sufficiently few discrete regimes.

The focus of this paper does not concern the certification of gain-scheduled flight controllers, but rather what is required to certify adaptive controllers that use system identification or some form of on-line learning to identify optimal controller gain settings, system transfer matrices, and/or control derivative matrices in real-time. Adaptive flight control systems are currently being developed to help pilots recover from aerodynamic upset conditions[8,9], to regain vehicle handling qualities and stability in the event of aircraft damage or control surface failure[10], to automatically fly vehicles autonomously in both air and space environments[11-15], to maintain vehicle performance during changing operating environments through use of neural networks[16-18], and to guide munitions to their targets [19]. These types of adaptive flight control systems have high degrees of non-linearity and non-determinism. Gain-scheduled control cannot be effectively used for these applications because the specification of all upset flight environments, the degree of control surface failure, or extent of aircraft or engine damage requires infinite sets of conditions. Instead, for these applications, it is more efficient to assume the structure of the controller and use learning algorithms or on-line system identification methods to obtain the locally valid operating parameters. Figure 1 taken from [Ref. 20] provides a notional diagram of an adaptive control system to illustrate two possible ways to implement adaptive control. In one case, the learning algorithm is used to compute augmentation flight control inputs; in the other a system identification method is used to identify controller parameters, gains, or transfer matrices.

Although the potential benefits of adaptive flight control systems are substantial, no adaptive flight control systems have been certified by the Federal Aviation Authority (FAA) for use in the National Air Space (NAS). The reason is that the means whereby adaptive flight control software can be routinely verified, validated, and certified for use in national air space has not yet been decided. As will be shown in the next section, the FAA has endorsed the use of RTCA DO-178 to provide FAA certification standards for all flight software. Although software techniques exist to verify and validate conventional flight control software, the means to provide sufficient assurance of adaptive flight control software functionality, reliability, safety, and the absence of unintended functionality remains elusive. Since by its very design the adaptive controller can make rapid and automatic adjustments to enable self-healing in the event of vehicle damage, it also carries the potential to make a healthy aircraft un-flyable or a safety hazard to other vehicles in the event of software malfunction. In a military application over restricted air space, such failures may be tolerated if not too high, but for a commercial application in civilian air space, such failures are unacceptable. Adaptive control systems with learning software will never become part of the future unless it can be proven that this software is highly safe and reliable.

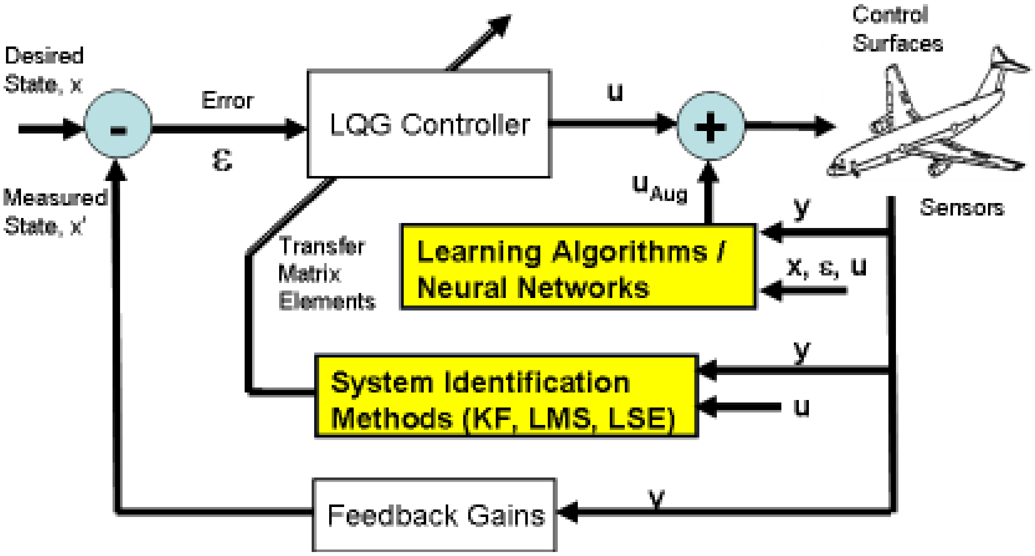

Fig. 1 A few ways to make flight controllers adaptive.

The objective of this paper is to examine the challenge areas that need to be addressed in order to enable the certification of adaptive flight control software for use in civilian air space. The purpose of this paper is to examine the gaps between the state-of-the-art methodologies used to certify conventional (i.e., non-adaptive) control system software and what is likely to be needed to satisfy FAA airworthiness requirements. These gaps include the lack of a certification plan or process guide, the need to develop verification and validation tools and methodologies to analyze adaptive controller stability and convergence, as well as the development of metrics to evaluate adaptive controller performance at off-nominal flight conditions.

2.0 What are the Certification Gaps ?

In the United States, the authority responsible for certifying flight control software is the Federal Aviation Administration (FAA). The FAA has stated in Advisory Circular 20-115B that all flight critical software must be developed according the guidance provided in RTCA DO-178B [21] or show compliance to airworthiness standards using some alternate means of compliance. Since alternate means of compliance are undefined, the statements in DO-178B are generally viewed as certification requirements, rather than mere guidelines. The objective of DO-178B is to help the aviation community develop flight software that can perform its intended functions while not negatively impacting other systems or the safety of aircraft operations. The document is maintained by the RTCA (Radio Technical Commission for Aeronautics) which is a private association of over 250 aeronautical organizations (established 1935).

DO-178B is not a process guide to software certification, but rather a description of what high-quality software development processes should be put in-place in order to create airborne software that has been properly verified and validated. It levies no special requirements for adaptive flight control software; it is meant to apply to all airborne software. If it can be adequately demonstrated that these processes have been correctly and appropriately implemented, then the software is in principle certifiable. This section highlights the specific guidelines recommended by RTCA DO-178B that, in the author's view, are potentially difficult for adaptive flight software control to fully satisfy and represent the gaps requiring the infusion of new verification and validation techniques and methods.

Table 1 provides a list or summary of the basic guidelines offered by DO-178B. This list has been generated by the author and is not intended to serve as a comprehensive index to the standard since it combines the intent of similar topics under general headings. However, it is useful for discussion purposes. The first column provides a list of the guideline categories or processes called for in DO-178B. The second column indicates which of these represent potential problem areas for adaptive flight control software. So, for example, the first guideline, “provide an overview of the system and target application software”, is marked “No” in the second column because this task is not more difficult to perform for adaptive flight control software than for conventional flight control software. As can be seen from the table, the majority of the guidelines do not present special problems for adaptive software; they are equally difficult for non-adaptive flight software. The satisfaction of these common guidelines will not be discussed in the sequel.

The rows marked with the "YES" designation in column 2 of Table 1 are the gap areas impeding the certification of adaptive systems. These areas are held to be in adaptive flight software requirements definition, performance specification, and definition of verification plans and test cases. Two additional related areas held to be problematic are software lifecycle data collection and development of a Plan for Software Aspects of Certification (PSAC). These are problem areas partly because of the first four and also partly because the design and test procedures for adaptive systems are not well-established.

Some airborne software developers argue that there is no guideline in RTCA DO-178B that cannot be satisfied using the same software assurance methods as are presently used for non-adaptive software [22]. In truth, it is possible to create documents stating adaptive controller

Table 1: List of DO-178B Guidelines for Software Certification

DO-178B Guideline Topic	More Difficult for Adaptive Systems ?
Provide an overview of system and target application of software	No
Provide an overview of what the software does	No
Identify the software lifecycle	No
Define the software performance requirements	Yes
Provide a Software System Safety Assessment (SSA) Report that lists all software failure modes and conditions and categorizes the failures according to their severity	No (although increased complexity)
Provide a Software Development Plan	No
Provide a Software Verification Plan	Yes
Provide a Software Configuration Management Plan	No
Provide a Software Quality Assurance Plan	No
Define Software Requirement Standards	No
Define Software Design Standards	No
Define Software Code Standards	No
Define Software Requirements and all Derived Requirements	Yes
Software Design and Traceability Document	No
Provide Tool Qualification Data (e.g., autocoders, compilers)	No
Provide Source Code	No
Provide Executable Object Code	No
Provide Software Verification Test Cases and Procedures	Yes
Provide Software Verification Results	No
Provide Problem Reports	No
Provide Software Configuration Management Records	No
Provide Software Qualify Assurance Records	No
Provide Plan for Software Aspects of Certification (PSAC)	Yes
Provide Software Accomplishment Summary	No
Provide Software Life Cycle Data	Yes

requirements, performance requirements, and V&V methods. The problem lies in that DO-178B does not provide metrics to assess the adequacy of these plans. This function is relegated to the DER (Designated Engineering Representative) working out the certification process with the FAA. It is the DER and the FAA who decide if the software documentation and procedures provided are sufficiently detailed and accurate. Since no adaptive flight control systems have been certified for use in the NAS, it is very difficult to judge the adequacy of these methods without references to previous work done by industry and government to show that these methods are sufficient to prove adaptive control system air worthiness.

The remaining sections of this paper discuss major gap areas in regards to the certification of adaptive flight control systems. Each of these sections describes a major gap area, presents a description of the present day state of the art in these areas, and cites what further research efforts will likely be needed to close the gaps. It must be mentioned that although these sections highlight many significant areas, it is not known to what degree this list is complete since no adaptive systems have been certified for routine civilian use in the national airspace. Nevertheless, the sections below are held to provide a very good starting point toward identifying what needs to be done to create a valid adaptive system certification process.

3.0 Gap in Defining Adaptive Controller Requirements

A critical gap which needs to be closed to facilitate certification is to develop procedures and methodologies to completely and correctly specify the design requirements of adaptive flight controllers. Not only are classical controller performance requirements specified by well-known metrics (e.g., gain margin and phase margin), the controllers are also designed to address usually very well defined requirements. In contrast, both the requirements and the controller performance metrics are difficult to clearly define for adaptive systems.

Most software life cycles (development through deployment) begin with an analysis to carefully define the software requirements as shown in Fig. 2. The left side shows the steps used to transform requirements into software code. The right side shows the steps of software integration and testing to make sure the code ultimately satisfies the software requirements. The process of testing the performance of the final code against the defined software requirements is called software validation. This is the meaning of the second "V" in the often used acronym "V&V" for verification and validation. (The word validation is also often used in connection with model validation, but this is a very different process consisting of comparing the predicted output of a dynamic model against measured data.) Software verification is the analysis and testing processes used to ensure the software code does what it was designed to do.

The software requirements define as precisely as possible what the software is supposed to do. DO-178B recommends that requirements be written in a manner that allows them to be tested and may include such things as performance, precision, accuracy, and timing constraints. The requirements are frequently decomposed into derived requirements to address such considerations as computer speed, memory size, interfaces, and frequency of inputs and outputs.

The most common reason software fails validation testing is incomplete or poorly defined requirements. Software developers may follow a verification process that proves that the software does exactly what it was designed to do algorithmically, but then discover that it does not meet the real software requirements because of improper specification. Such failures are very expensive to correct because when an error is found in late stage validation, the entire software design, verification, integration, and validation process shown in Fig. 2 must be performed all over again. Whereas errors in requirements specification for classical controllers usually results because some important test condition or environment was not stated, requirement specification errors for adaptive controllers can result because they are only notionally known, and not easily specifiable using metrics for classical controllers.

Current State of the Art: The availability of modeling and simulation programs such as Matlab/Simulink [23] have encouraged the simulation of controller performance (both classical and adaptive) prior to coding actual source code for the target flight control computer. Model-based design methods tests candidate software designs for performance and conformance to requirements by using simulation to model of the input, plant (aircraft), controller, and disturbances (Fig. 3). The utility of this approach is that the model-based design and simulation programs allow the plant and controller mathematics to be represented by block diagrams rather than writing actual code for the target host computer. In Fig. 3, the software requirements suggest a software architecture that is modeled in the simulation environment. When executed, the behavior of the controller can be observed and, in theory, the performance of the controller can be tested to see if it meets the software requirements. If it does not, the software architecture can be quickly modified in the simulation environment and the tests repeated. Once the software meets the requirements, the process illustrated in Fig 2 may be followed as usual, but in this case the validation tests are much more likely to be satisfied.

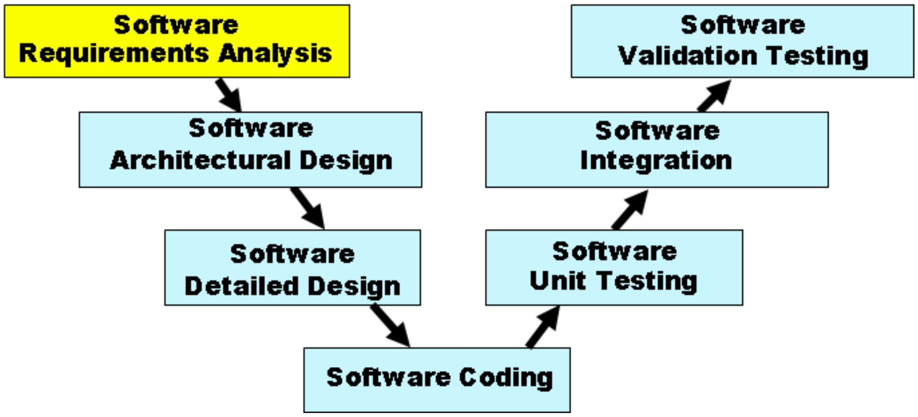
Software Requirements Analysis
Software Architectural Design
Software Detailed Design
Software Coding
Software Unit Testing
Software Integration
Software Validation Testing

Fig. 2 Software life cycle or development process.

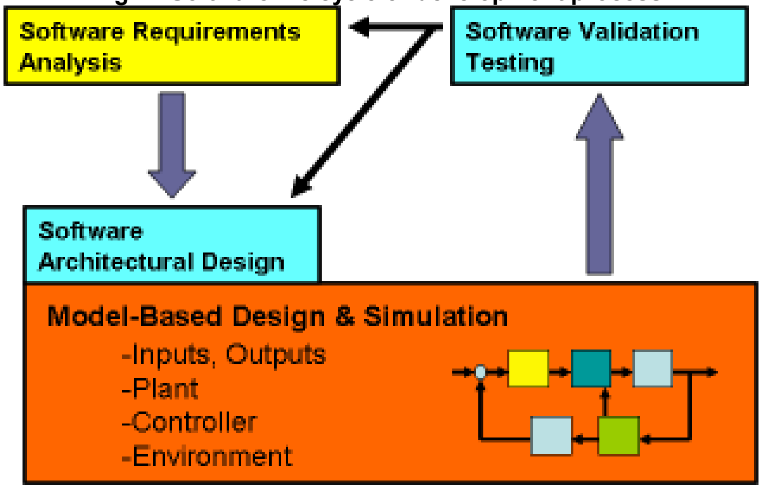
Software Requirements Analysis
Software Validation Testing
Software Architectural Design
Model-Based Design & Simulation
-Inputs, Outputs
-Plant
-Controller
-Environment

Figure 3. Model-based design methods tests candidate software designs for conformance to requirements prior to producing code for the target computer.

The present state of the art doesn't stop with desktop simulation, but includes increasing level of simulation complexity. Desktop simulation may be followed by sub-scale testing on small models of the aircraft or spacecraft. More commonly, the simulation complexity is increased through the incorporation of non-linear aerodynamic and structural dynamic models. These simulations are usually run on a dedicated workstation computer platform. The next step in the simulation hierarchy includes using the actual target flight control computer in the simulation, as well as other hardware placed in the control loop such as cockpit controllers and actual sensor input. After this may follow motion-based simulation where by the simulation is flown by a test pilot in a simulated cockpit environment that receives both visual and motion feedback. The last step is testing on the actual target flight vehicle. DO-178B presently allows certification credit to be obtained for both high-fidelity simulation testing as well as actual flight testing.

The present state of the art is to analyze adaptive system learning convergence and stability using simulation environments thought to provide enough fidelity to model significant nonlinear aerodynamics, dynamics, and other factors. To be certain, mathematical analysis of stability has been done (as explained in the next section). However, although there are analytical equations to calculate optimal gain selection for rapid, stable learning, it has been found that high adaptation gains will often lead to high frequency oscillation in the tracking errors, especially in poor signal to noise environments.[4, 24] It is usually only in high-fidelity simulation that the compromise between rapid learning and oscillatory tracking can be found. Simulation provides a fairly rapid way to

- Evaluate and compare different learning algorithms,
- Tune control gains and learning weights,
- Determine how much learning is actually being accomplished at each step of the simulation,
- Evaluate of the effect of process and measurement noises on learning convergence,
- Determine learning stability boundaries,
- Test algorithm execution speed on actual target flight computer,
- Conduct piloted evaluation of the learning system in a flight simulator, or,
- Simulate ad-hoc techniques of improving the learning process, such as adding persistent excitation to improve identification and convergence, or such as stopping the learning process after error is less than a specified error.

Simulation is frequently used to assess the effect of process noise and measurement noise on controller performance. It has been shown, for example, that adaptive control system learning or system identification convergence requires persistent excitation.[2] This problem generally occurs when the controller computes the correct optimal control before the system identification or learning method completely converges. If the controller is able to find a control vector that effectively nulls the error between the desired state and measured state, then updates to the weight values based on that error signal will also tend to zero and the system does not learn until the control error becomes higher. Further, as the optimal control vector is changed only slightly near the optimal control point, a system identification algorithm will compute the transfer matrix relating the very small changes in control to the measurement noises.[25] A persistent excitation signal added to the control signal yields better learning, but at the expense of poorer steady-state controller performance. An alternative is to disable the learning process when the control error becomes low. References 24 and 25 provide detailed insight to this problem.

The effect of noise on learning and identification performance as well as ad hoc approaches such as disabling learning during period of low control error are difficult to analytically verify and validate, but relatively easy to evaluate in simulation. In addition, is has been shown that the acceptable gains for stable learning law must be found by trial and error.[24] Even though conventional control system robustness measures such as gain margin and phase margin cannot be applied to an adaptive system, such quantities might be calculated (in flight) during periods of steady-state control behavior, but can always be computed in simulation, one point at a time. It is highly likely, therefore, that a simulation will form of the certification process for adaptive systems.

Further Research Needed*:* Certification of adaptive control systems will be significantly aided by the development of more precise ways to specify requirements (metrics), by the development of simulation benchmarks for adaptive systems, and by the development of automated analysis tools to support verification and validation using model-based design simulation.

Since DO-178B allows certification credit for the use of high-fidelity simulations and test beds, it is highly likely that simulation will be a key part of any adaptive control certification process. This is because testing the controller in the target operating environment for a representative test case may not always be possible or prudent. For example, a controller for space craft orbiting an asteroid or a controller to help an aircraft pilot recover from severe wing damage. Simulation is clearly a preferable and safer alternative, but the simulation fidelity must be sufficiently high so that important nonlinear effects are not missed.

A critical aspect of obtaining certification credit for simulation is proving that the simulation fidelity is acceptably high. If simulation is viewed as a tool, then this is very similar to the DO-178B requirement to use certified tools in the software development program. The present lack of common simulation models also inhibits comparison of adaptive controllers, in addition to impeding certification.

A important consideration that will influence the development of a common simulation model is that adaptive control systems are comprised of hybrid systems. Hybrid systems attempt to model the full control system so that all components can be tested at once rather than in isolation. The definition of a hybrid system can take different meanings. Reference [24] defines a hybrid adaptive controller as a combination of direct and indirect adaptive control. Hybrid systems may also refer to control systems comprised of finite state executive controllers coupled to a continuous domain adaptive controller.[20, 26] A related characterization of a hybrid system is one in which a finite state controller using a continuous learning algorithm is coupled to a discrete model of the (normally) continuous environment.[27]

The latter characterization might seem odd, but this type of hybrid system is being studied as a means of leveraging the power of model checking software such as SPIN, JPF2, and NuSMV.[28-31] Model checking is a technique by which a finite state system model can be exhaustively explored to make sure the system never reaches an unacceptable state. The method relies on being able to express adaptive controller safety properties as temporal logic.[30] More importantly, however, an approximation function is required to convert the continuous variables into discrete values.

Lastly it is mentioned that a major gap area for adaptive controller certification are ways to define the meaning of acceptable controller performance. In the next section, some proposed metrics to assess control system robustness and other quantities as mathematical quantities will be discussed. Here it is noted that a clear certification path must begin with a clear definition of how a successful adaptive controller operates. Is it a requirement that the adaptive flight control software be always active, or does it become active only when enabled by the pilot? Pilot activation might allow certification at a lower safety category by making it a pilot aid or tool. For example, Cirrus has introduced an autopilot with a wings-level button. This autopilot, the Garmin G1000 autopilot that can recover the aircraft from an attitude of 75 degrees of roll and 50 degrees of pitch.[32] This system, however, is certified as part of the autopilot (a pilot tool) and not as a high-bandwidth, automatic adaptive controller. The purpose of this autopilot tool is to help pilots recover aircraft attitude during upset or times of reduced spatial awareness in order to give them time to assess the situation. Nevertheless, this is a step forward toward acquiring data that may be used as part of a case for certification.

Any plan for certification hoping to gain FAA approval will likely need to have firm answers to these questions. Improper specification of the requirements hampers the development of suitable verification and validation test cases needed to show certification compliance to the requirements.

4.0 Gaps in Software Verification and Validation Methods

The software verification plan provides a description of each activity in the software verification process. Generally, software verification is comprised of software review, software analysis, simulation, and testing. These activities may include the use of software programming checklists and formal software analysis and testing methods. Software analysis methods can include formal methods, static analysis, code reviews, traceability analyses, and coverage analyses[20]. The software verification plan establishes the rational for the development of software test cases and methods.

A critical gap in the validation and verification plans for adaptive control systems is the lack of procedure that can reliably verify that the learning algorithm or system identification method learns correctly and converges to the correct solution in an acceptable time. This verification will be the key step in any verification and validation plan for an adaptive system, since other than learning and system identification, there is no difference between adaptive and non-adaptive controllers. Consider the neural network neuron shown in Fig. 4. A learning algorithm of the type

$$\Delta w_i = f(e)$$

$$w_i(k+1) = w_i(k) + \Delta w_i$$

is typically used to update the weight values based on some error metric, *e*. The function *f* depends on the learning algorithm used (steepest descent, Gauss-Newton, Levenberg-Marquardt , etc.). It can be seen in Fig. 4 that if six weighted inputs are summed together to form one output, the values of the weights offering a correct solution is not necessarily unique, nor guaranteed to exist. System identification methods for transfer matrix determination are alike in respect to this problem; the neuron shown in Fig. 4 is mathematically equivalent to a row-column matrix multiplication operation. There are many excellent texts (e.g., [33]) that have analyzed the necessary conditions for a unique solution. Since there are six inputs and one output, the minimum requirement is that the outputs be known for at least 6 linearly independent input vectors. Then six equations in six unknowns exist, and values of the weights can be determined. However, if the second input is the square of the first input, and the third input is the product of the fourth input and the first input, and so on, the inputs are not independent and convergence to a solution may not be possible. Neural networks often employ such input combinations in an attempt to offer superior curve fits to input data. However, sensor input measurements that are related to each other electronically or mechanically (e.g., two accelerometers on a wing located some distance apart but measuring the same acceleration) are other reasons for dependencies to exist between different inputs.

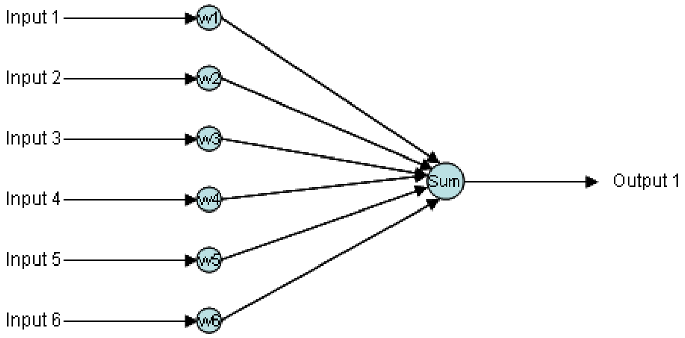

Fig. 4. Neural network neuron.

Even if great care is taken to ensure a linearly independent set of input measurements, another problem which makes learning and identification difficult is the presence of process and measurement noise.[33] As an adaptive controller achieves a steady-state solution to the learning problem, the input vectors from one cycle to the next may become very small as the optimal control is reached. If process noise adds uncertainty to the value of the weight vector from one cycle to the next and/or measurement noise is present in the measured output, the linear independence of the input vector stream can be lost.

These factors make finding a certification test case to prove learning convergence to the correct weight vector under all conditions very difficult to find. A certification test seeking to assess “how far away” from the true values the weights is difficult to specify in a general way.

If convergence to one of the valid or unique weight sets is achieved, a related learning problem occurs in the presence of measurement noise. In this case, as the system is operated in a neighborhood of the optimal control solution, the learning algorithm or system identification method will many times destabilize by computing weight values or transfer matrix elements to relate small changes in the control vector to measurement noise. This produces divergent learning behavior.

Current State of the Art: At the present time, no adaptive systems have been approved by the FAA for use in commercial airspace apart from those based on gain-scheduled control. Therefore there is no example of an approved verification and validation plan for an adaptive flight controller. Even so, most parts of such a plan already exist because the best practices used to certify non-adaptive controllers can still be used, at least in part. These best practices include the use of software programming checklists, code reviews, traceability analyses, static analyses, and coverage analyses.

The verification and validation plans for an adaptive flight controller, however, will require an extension to evaluate the performance and all failure modes of the learning algorithms or system identification methods. Therefore, the focus of the verification and validation plan for adaptive controllers comes down to verifying that the learning algorithm or system identification method converges to the correct system parameter values in a stable manner and at an acceptably fast rate. This will be the crux of an acceptable certification plan. Presently, two paths are being explored as a means to evaluate stability and convergence: 1) high-fidelity simulation, and 2) mathematical analysis.

With regard to simulation, the current general practice is to use variations of the Monte Carlo analysis method.[34] In this method, the range of values each parameter may take is determined. A finite number of parameter test values is then selected. If nothing is known about the parameter's expected value, then a uniform spacing throughout the parameter range is a logical choice. Alternatively, if the expected value of a parameter is known, then the test values can be more closely spaced near the expected value. Once all parameter test values have been selected, the matrix of simulation runs is comprised of every parameter varied in combination with all other parameter values. So, if there are three parameters that can take 5 unique values each, the number of simulation runs is 3^5 or 243. Even for a non-adaptive controller, Monte Carlo simulation can be very time consuming considering the number of possible changes in the parameters just describing the operating condition (airspeed, altitude, weight, etc.). When number of parameters in the state vector is included, the number of simulation cases required can easily render the task of full Monte Carlo simulation intractable, except for very sparse parameter variations that leave large portions on the state space unexplored. From a certification standpoint, that would not be acceptable.

Mathematical proofs of adaptive controller stability generally seeks to show that the vehicle state returns to a neighborhood about the undisturbed state for every defined disturbance. The most commonly used proof is based on Lyapunov's second method.[4, 35-37] For linear time invariant systems of the form,

$$\dot{x} = Ax$$

where *x* is the state vector and *A* is the state transition matrix, the Lyapunov method states that the system is stable (will return to the origin) if a Lyapunov function *V(x)* can be found that is always positive and that has a time derivative that is always negative or equal to zero,

$$V(x) > 0$$

$$\dot{V}(x) \leq 0$$

The Lyapunov function usually chosen is

$$V(x) = x^T Px$$

and the system is said to be stable if, and only if, given any symmetric positive-definite matrix *Q*, there exists a symmetric positive-definite matrix P, which is the unique solution of the set of

$$A^T P + PA = -Q$$

equations.[4] Although finding a Lyapunov function has in the past been somewhat of a cumbersome trial and error process, recent advances in semidefinite programming and semialgebraic geometry have afforded more of an algorithmic procedure to find a valid Lyapunov function through the use of the sum of squares (SOS) method.[38]

From a certification perspective, a weakness of the Lyapunov approach to prove stability is that the proof requires a polynomial representation of the plant (*A* matrix) for all flight conditions of interest. If the values of this representation changes, perhaps due to aircraft damage, then nothing can be said about controller stability. More importantly, the Lyapunov analysis only guarantees the ultimate stability of the learning algorithm; the proof does not guarantee how fast the system returns to the origin. In adaptive controller parlance, this means that Lyapunov proofs cannot guarantee the rate of learning convergence. This is an important point for system performance, because if learning happens too slowly, an adaptive controller may be rendered ineffective for the control task at hand.

Further Research Needed*:* Although Lyapunov stability proofs have been thoroughly investigated and analyzed for nearly five decades, one gap that still remains in regards to certification is that these proofs are not easily understood by most engineers. Consequently, one problem facing the certification of adaptive control systems is that there is a large gap between the mathematicians understanding of stability and credence an designated engineering representative (DER) or and FAA official will be willing to give it in the plan for certification. A mathematical proof that is understood only by experts in the control field may fail to inspire certification authorities that all due diligence has been done. Moreover, since these proofs depend on knowledge of the "*A*" and "*B*" matrices of the control system representation, $\dot{x} = Ax + Bu$, and these may actually change with time (e.g., with aircraft damage), such that Lyapunov proofs may well lack sufficient conditions necessary for convergence. A similar criticism also holds for the method of attempting to prove regions of controller stability through the use of barrier certificates. [39, 40] Like Lyapunov, this approach seeks to prove that the state trajectories starting from a given set of initial conditions never reach an unsafe region. The barrier certificate represents a guaranteed upper bound on the probability that the system trajectories do not reach the unsafe set. However, application of the method also requires the plant be expressed in polynomial form as well as the boundaries of the unsafe states.

One possible approach to this problem is to develop software algorithms to monitor the control system health by having a means to monitor the necessary conditions for successful adaptive controller operation. Such health monitoring could also conceivably assess when the flight conditions or aircraft state was beyond the authority of the controller.

Another gap of Lyapunov stability analysis that will need to be addressed is the development of metrics to quantify the robustness of the control system. Metrics that can assess how far away the system is from instability are needed that are analogous to the gain margin and phase margin metrics used to assess non-adaptive, linear controllers. Because adaptive controllers may change their gains on-the-fly, these terms have no meaning because the system is never constant.

One method being studied by NASA under the Aviation Safety Program to help close the certification gap is to look at ways to extend the traditional Monte Carlo analysis method to assess the robustness of adaptive control systems. At the Langley Research Center, an analysis tool called RASCLE (for Robustness Analysis for Control Law Evaluation) has been developed to help explore combinations of learning system parameters and operating conditions [41]. The RASCLE simulation tool is used to interface with existing nonlinear simulations and incorporates search algorithms to uncover regions of instability with as few runs as possible. RASCLE uses a gradient algorithm to identify the direction in the uncertainty space along which the stability of the system is most rapidly decreasing. RASCLE provides an intelligent simulation-based search capability that can be used in Monte Carlo simulation evaluations [42]. At the Ames Research Center, another approach to extend Monte Carlo analysis has been studied for analysis of large complex aerospace systems, such as adaptive systems have a highly coupled nonlinear nature. In this approach, an algorithm has been developed to only limit the number of combinatorial cases required of Monte Carlo analysis, but also to explore interactions in the parameter space in a systematic fashion. The data generated is automatically analyzed through a combination of unsupervised learning using a Bayesian multivariate clustering technique (AutoBayes) and supervised learning of critical parameter ranges using the machine-learning tool TAR3, a treatment learner.[43] Covariance analysis with scatter plots and likelihood contours are used to visualize correlations between simulation parameters and simulation results.

Although the mathematical soundness of Lyapunov stability analysis theory is not in question, it is highly possible that this method of analysis will never be able to provide the type of software assurance necessary for certification. To close this gap, therefore, research to discover other avenues to Lyapunov analysis should be explored. As was mentioned at the outset, the most likely other alternative is to use high-fidelity simulations incorporating nonlinear effects, hardware-in-the-loop, and motion-based simulations. Although it is conceivable that the creation of better learning algorithms through analytical development could potentially improve the learning rate of convergence, it has been shown that fast convergence in the presence of measurement noise and un-modeled dynamics actually produces instability.[24] Hence the challenge problem for certification is not so much the search for better learning algorithms, but more towards finding ways to find ways to demonstrate adaptive controller performance is simulation, but at the same time taking steps to avoid the trial and error adaptation gain selection process or system identification tuning process. Trial and error is not a good solution because not only does it hamper a certification test process by not having predefined test values, but it is also difficult to later repeat the same testing on actual flight vehicles that may match or may exceed the fidelity of the simulation.

Nearly all software verification and validation plans for airborne software are ultimately aimed at proving that the software code fulfils its intended function, but this is very difficult to do with learning or system identification software proposed for adaptive control systems. The problem is that the weights in a neural network or number of parameters in a transfer matrix may not have a solution, or if it does, the solution may not be unique as mentioned above. Consequently, it is felt that the correct values of the network weights or transfer matrix parameters cannot be know, making software verification an NP hard problem in this case.[44] The usual practice is to couple the learning algorithm or system identification method to the adaptive controller and evaluate the behavior as a unit. However, this ignores the obvious merit of being able to prove proper learning behavior as part of a certification process. One possibility may be to separately test learning algorithms and system identification methods in simulation using contrived aircraft models that have a unique, known solution. The effects of measurement noise, process noise, persistent excitation, and ad hoc means to stop learning for low control errors could then be easily evaluated. One such attempt at doing such experimentation for helicopter adaptive vibration and noise algorithms is presented in Reference 24. Such simulation cannot verify the learning behavior of over-parameterized systems, yet can provide proof that the basic learning algorithm was verified to work at least under ideal or known test conditions.

One avenue that offers a possibility to help close this certification gap is to develop, in combination with simulation, probabilistic uncertainty models for adaptive controllers in order to quantify their robustness. Reference [45] discusses the development of probabilistic uncertainty models to assess the effect of parameter variations on controller stability. By making various cross-plots using the controller parameters, the plot space can be divided into regions called the Failure Domain and the Admissible Domain. By varying the parameters, probabilistic uncertainty methods can define a set of plants and associate a weight (or probability) for each. This then facilitates a search for a robust controller by being able to quantify how far away the system is from instability or some other problem by through parameter variation (homothetic deformations). A gap for certification purposes is that although probabilistic analysis helps prove a design using the most robust controller design, safety of flight demands stability at the corner cases too. It must be remembered that the certification world often wants to see failure rates less than 10**-9 for single, non-redundant systems, or the use of a collection of cascaded or redundant systems that achieve that as a joint probability.

Failing to prove that an adaptive flight controller has sufficient reliability at all operating conditions and for all failure scenarios is a certification gap that can be closed in only one of two ways. The first way is to tighten the performance requirements placed on the adaptive controller. As the performance specifications in this regard becomes more narrow and specific, the adaptive control system is required to do less, but the V&V plan is able to become more specific and well-defined. This is not always a bad thing, since it is good to be able to know and to design to specific requirements. However, a second way is to field the adaptive controller with a set of on-line tools that can monitor the state of the controller to continuously assess its performance.

One example of an on-line software assurance tool is the Confidence Tool developed under the NASA Aviation Safety Program.[46]. This tool provides a useful metric to assess neural network weight convergence using a Bayesian approach. The Confidence tool is a dynamic monitor, which checks the output values of the neural network and determines if the output of the neural network is reliable by calculating a confidence measure. This metric is based upon a statistical model of the learning system originally developed for pre-trained neural networks, but recently extended for use with on-line learning neural networks. The Confidence tool uses a Bayesian approach to dynamically calculate a confidence measure.

Another example of an on-line tool to evaluate control system robustness is a tool developed at the NASA Dryden Flight Research center to provide a method for in-flight stability estimation of the X-38 crew return vehicle.[47] This method introduced a small-amplitude to an elevator and rudder tailored-force excitation that was targeted to a specific frequency range. The frequency response at these frequencies was then used to calculate the stability margins of the flight control system using a modification of the method in Reference [48]. A recursive Fourier transformation was used to make the method compatible with real-time calculation. The stability calculated by the on-line method compared well to the X-38 nonlinear simulation. The utility of having a metric of stability that can be computed in flight was a great increase in test efficiency for the X-38 flight test. For the certification of adaptive control systems, such methods might be extended to evaluate adaptive control system stability. In this way, the gap created by the fielded software not being quite the same as the fielded software might be further closed through the use of tools for on-line controller performance monitoring.

As a last thought on tool development for adaptive controller verification, it is mentioned that some researchers are seeking to adaptive formal methods developed primarily for finite state systems to the continuous domain of control. Methods such as compositional verification have been used to break apart large complex finite state systems into smaller parts that can be analyzed using the powerful and exhaustive model checking methods. For example, Reference [49] describes an application of the NASA Ames Java PathFinder model checker to the control the guidance of a robotic vehicle. Using compositional verification to verify that the interface logic between components will function properly when integrated together to make the full system requires only a domain change and is very feasible. The difficult part is finding a way to approximate the continuous state vector and measurement domains by a set of discrete values. The idea is similar to rounding a decimal number to the nearest integer, only in this case, the truncation must be considerably coarser. With this type of hybrid model approximation, the state and measurement values take on finite values. This allows for the recognition of previous "states" in the model checking sense of the word, and hence an exploration of the continuous model checking space becomes possible. Of course, this search is exhaustive only to the extent the approximation function is valid. If the approximation function is too coarse, important states will likely be missed.

5.0 Gaps in the Adaptive Controller Software Development Process

The software development process described by DO-178B for airborne software is one in which the verification and validation plans are developed before any code is written. This includes both the overall plan as well as the specific plans down to the level of unit testing. A difficulty with even non-adaptive controller development is that the usual path is an engineer does an analysis of a proposed controller design and then moves immediately into desktop simulation. Subsequently, as problems are encountered, various fixes and modifications are tested until something appears to work. At that time, a documented design may be produced, but documentation of the all the development problems and description of everything tried that did not work is typically lost. For adaptive controllers having greater complexity, the problem is significantly worse. Skipping the time it takes to document failed approaches and tuning values is no doubt a major time saving step, but the problem is that it also prevents collection of valuable lifecycle data needed for certification.

Current State of the Art: It is not possible to mention the many on-going efforts by industry and government projects to develop adaptive flight control systems.[10, 13, 15, 16, 35, 50-53] Although most of the industry development programs are proprietary, the Air Force VVIACS (Verification and Validation of Intelligent and Adaptive Control Systems)[54] and NASA IRAC (Intelligent Resilient Adaptive Control)[55] efforts represent multi-year programs with industry partners have been initiated to define methodologies and test procedures for adaptive flight control systems. The continuing IRAC Project is sponsored by the NASA Office of Aviation Safety. The goal of the IRAC Program is to conduct research to advance the state of aircraft flight control to provide onboard control resilience for ensuring safe flight in the presence of unforeseen, adverse conditions. The objective is to advance the state-of-the-art of adaptive controls as a design option to provide enhanced stability and maneuverability margins for safe landing. It is anticipated that the outcome of the IRAC project research will be a set of validated, multidisciplinary integrated aircraft control design tools and techniques for enabling safe flight in the presence of adverse conditions such as structural damage, control surface failures, or aerodynamic upsets. With regard to the certification of adaptive flight control systems, it is hoped that the analysis, simulation, sub-scale and full-scale flight tests of this research program will help form the basis for a valid Plan for Software Aspects of Certification (PSAC) for adaptive flight control systems as part of a certification plan.

Further Research Needed*:* A difficulty is that performing verification and validation to enable research to progress in a development environment is not necessarily the same as the software assurance testing required by the Federal Aviation Authority (FAA) to certify the software for operation. Although in the research and development environment every effort is made to ensure that the adaptive software functions as required, the operating conditions, test hardware configurations, and types of adaptation tasks are highly restricted in order to allow a focused program to proceed along a well-defined path. This approach offers the ability to conduct a proof of concept demonstration in a relatively short amount of time, but unfortunately leaves the development of a certifiable control system to future developers.

A very practical aspect of a certifiable adaptive flight control systems is that DO-178B advises that safety-critical software should provide a measure of software redundancy and fault tolerance. The preferable level of redundancy is two systems doing the same thing, but using different calculation methods to arrive at the same answer. This is referred to in DO-178B as redundancy achieved by using dissimilar implementations. A problem with using the technique of dissimilar implementations for adaptive flight systems is that the dissimilar implementations could take different control trajectories to achieve the same end state and yet not be comparable along the way. Designing in the required level of fault tolerance for adaptive flight control systems is a major certification gap. Another gap is the usage of partitioned real-time operating systems (RTOS) that are equipped with vehicle health management tools to ensure any failures in the controller remain isolated, while allowing another partition to perform health management and failure detection.

Verification and validation plans for certifiable software would need to provide a test matrix together with an explanation why each test point has been chosen and how together all of the test points will provide adequate test coverage. RTCA DO-178B recommends that the report should include a description of the conditions under which each test is to be performed and state the pass/fail criteria. Step by step instructions for performing each test are to be provided along with instructions with how to evaluate the test results. DO-178B stresses that it is important that these procedures and criteria be developed prior to the actual testing. In fact, DO-178B states that the verification and validation tests should be defined prior to writing any code. This is of course not possible for a research program. For a commercial certification effort of an adaptive control system, the experience to know the best test practices will hopefully come from the IRAC program and other similar efforts.

Once a sufficient set of best practices for the verification and validation of adaptive flight control systems becomes available, it may be possible to augment the traditional DO-178B PSAC with a Safety Case argument. Safety cases have been created for certification of nuclear industry in Europe and off-shore oil refineries in Australia.[56-57] A safety case is a document that identifies all hazards and risks, describes how the risks are controlled, and describes the safety management plan to ensure the controls and guidelines are effectively and consistently applied. The safety case represents a collection of processes to ensure all identified risks are mitigated. Obviously, the development of stability analysis methods for adaptive controllers, metrics for adaptive controller performance (or learning), hybrid high-fidelity simulation methods, the usage of formal methods, and other technologies would conceptually become part of the safety case. In essence, the safety case argues for software certification on the basis that every best practice to ensure safety has been followed. Whether or not this is the same thing as proving the system is safe is a valid gap for certification using the safety case approach. In fairness, however, any certification procedure fulfilling the spirit of the DO-178B guidelines might also end up not being safe.

Summary

This paper has provided an examination of the gaps between current state-of-the-art methodologies used to certify airborne software and what is likely to be needed to satisfy FAA airworthiness requirements for the certification of adaptive flight control systems. These controllers use system identification or some form of on-line learning algorithm to identify optimal controller gain settings, system transfer matrices, and/or control derivative matrices in real-time. These gaps include the lack of a certification plan or process guide, the need to develop verification and validation tools and methodologies to analyze adaptive controller stability and convergence, as well as the development of metrics to evaluate adaptive controller performance at off-nominal flight conditions. This paper has provided the major certification gap areas and has presented for each a description of the present day state of the art and what further research efforts will likely be needed to close the gaps remaining in current certification practices. The areas addressed include the need for advances in simulation methods, methods to determine learning algorithm stability and convergence rates, the development of better performance metrics for adaptive controllers, the application of formal software assurance methods, the need for on-line software monitoring tools and health assessment, and the development of a certification plan for adaptive systems.

References

[1] K. J. Åström and B. Wittenmark, Adaptive Control, 2nd ed. Reading, MA: Addison-Wesley, 1995.

[2] P. A. Ioannou and J. Sun, Robust Adaptive Control. Englewood Cliffs, NJ: Prentice Hall, 1996.

[3] I. D. Landau, Adaptive Control: The Model Reference Approach. New York, NY: Marcel Dekker, 1979.

[4] K. S. Narendra and A. M. Annaswamy, Stable Adaptive Systems. Englewood Cliffs, NJ: Prentice Hall, 1989.

[5] S. Sastry and M. Bodson, Adaptive Control: Stability, Convergence, and Robustness. Englewood Cliffs, New Jersey: Prentice-Hall, 1994.

[6] Ogata, K., Modern Control Engineering, 4th Edition, Pearson Education, Nov 2001.

[7] Bryson, A. E. and Ho, Y. C., Applied Optimal Control, Taylor and Francis, 1975.

[8] Kaneshige, John, and Gundy-Burlet Karen, "Integrated neural flight and propulsion control system," Proceedings of the AIAA Guidance, Navigation, and Control Conference, AIAA-2001-4386, August 2001.

[9] Williams-Hayes, P.S., "Flight Test Implementation of a Second Generation Intelligent Flight Control System", Technical Report NASA/TM-2005-213669, 2005.

[10] N. Nguyen, K. Krishnakumar, J. Kaneshige, and P. Nespeca. Dynamic and adaptive control for stability recovery of damaged asymmetric aircraft. In Proc. of AIAA Guidance, Navigation and Control Conf., Keystone, CO, Aug. 2006. AIAA 2006-6049.

[11] Kaneshige, J., Bull, J., and Totah, J., "Generic Neural Flight Control and Autopilot System," AIAA-2000-4281.

[12] Hall, R., Barrington, R., Kirchwey, K. and Alaniz, A., "Shuttle Stability and Control during the Orbiter Repair Manuever, AIAA Guidance, Navigation an dControl Conference and Exhibit, 2005, AIAA 2005-5852.

[13] E. N. Johnson and A. J. Calise, "Limited authority adaptive flight control for reusable launch vehicles," Journal of Guidance Control and Dynamics, vol. 26, pp. 906-913, 2003.

[14] Hovakimyan, N., Kim, N., Calise, A.J., Prasad, J.V.R., and Corban, E.J., "Adaptive Output Feedback for High-Bandwidth Control of an Unmanned Helicopter", AIAA Guidance, Navigation and Control Conference, AIAA-2001-4181, 2001.

[15] E. Lavretsky and K. Wise. Adaptive flight control for manned/unmanned military aircraft. In Proc.of American Control Conference, Portland, OR, June 2005.

[16] Rysdyk, R.T. and Calise, A.J., “Fault Tolerant Flight Control via Adaptive Neural Network Augmentation”,AIAA Guidance, Navigation, and Control Conference, AIAA-1998-4483, 1998.

[17] Johnson, E.N., Calise, A.J., El-Shirbiny, H.A., and Rysdyk, R.T., “Feedback Linearization with Neural Network Augmentation Applied to X-33 Attitude Control”, AIAA Guidance, Navigation, and Control Conference, AIAA-2000-4157, 2000.

[18] Calise, A. J., Lee, S., and Sharma, M., "Development of a Reconfigurable Flight Control Law for the x-36 Tailless Fighter Aircraft," Proc. of the AIAA Guidance, Navigation, and Control Conference, Denver, CO, Aug. 2000.

[19] Calise, A. J., Sharma, M., and Corban, J. E. "Adaptive Autopilot Design for Guided Munitions," Journal of Guidance, Control, and Dynamics, vol. 23, pp. 837-843, 2000.

[20] Jacklin, S. A., Schumann, J., Gupta, P., Richard, M., Guenther, K., and Soares, F., “Development of Advanced Verification and Validation Procedures and Tools for the Certification of Learning Systems in Aerospace Applications,” Proc. of the AIAA Infotech@Aerospace Conference, Crystal City, VA, Sept. 2005.

[21] Software Considerations in Airborne Systems and Equipment Certification, Document No RTCA (Requirements and Technical Concepts for Aviation) /DO-178B, December 1, 1992.

[22] Santhanam, V. “Can Adaptive Flight Control Software be Certified to DO-178B Leve A?”, NASA and FAA Software and CEH Conference, Norfolk, VA, July 26-28, 2005.

[23] MATLAB, The MathWorks Inc. http://www.mathworks.com/products.

[24] Nguyen, N., and Jacklin, S. A., “Neural Net Adaptive Flight Control Stability, Verification and Validation Challenges, and Future Research,” IJCNN Conference, Orland Florida, 2007.

[25] Jacklin, S. A., “Comparison of Five System Identification Algorithms for Rotorcraft Higher Harmonic Control,” NASA TP 1998-207687, May 1998.

[26] C. Tomlin, and Greenstreet, M. R., editors. Hybrid Systems: Computation and Control, 5th InternationalWorkshop, HSCC 2002, Proceedings, volume 2289 of Lecture Notes in Computer Science. Springer, 2002.

[27] Clarke, E. M., Fehnker, A., Han, Z., Krogh, B. H., Stursberg, O., and Theobald, M., " Verification of Hybrid Systems based on Counterexample-Guided Abstraction Refinement," in Tools and Algorithms for the Construction and Analysis of Systems, 9th Intl. Conf., TACAS 2003, pages 192–207. Springer, 2003.

[28] Holzmann, G. J., *The Spin Model Checker Primer and Reference Manual*, Addison-Wesley, Boston, MA, 2004.

[29] McMillan, K., *Symbolic Model Checking*. Kluwer Academic Publishers, Boston, MA, 2003.

[30] Visser, W., Havelund, K., Brat, G., Park, S., and Lerda, F., “Model Checking Programs”, Kluwer Academic Publisher, 2002.

[31] Havelund, K., “Using Runtime Analysis to Guide Model Checking of Java Programs,” *SPIN Model Checking and Software Verification*, Vol. 1885 of Lecture Notes in Computer Science. pp. 245–264, Springer, 2000.

[32] Haines, T. B., "Cirrus Gets a New Perspective by Garmin,", AOPA Pilot Reporting Points, May 20, 2008. Available at http://blog.aopa.org/blog/?p=248

[33] Kailath, T., Linear Systems, Prentice-Hall Information and System Sciences Series, 1979.

[34] Belcastro, Christine, and Belcastro, Celeste, "On the Validation of Safety Critical Aircraft Systems, Part I: Analytical & Simulation Methods", Proceedings of AIAA Guidance Navigation and Control Conference, Austin TX, August 2003.

[35] Wise, K. A., Lavretsky, E., and Hovakimyan, N., Robust and Adaptive Control Workshop, American Control Conference, Seattle, WA, June 11-13, 2008.

[36] Khalil, H. K., Nonlinear Systems, Prentice Hall, 3rd Edition, 2001.

[37] J-J. E. Slotine and W. Li, Applied Nonlinear Control, Prentice Hall, New Jersey, 1991.

[38] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, “Introducing SOSTOOLS: A general purpose sum of squares programming solver,” in Proceedings IEEE Conference on Decision and Control, 2002, available at http://www.cds.caltech.edu/sostools.

[39] Prajna, S., Jadbabaie, A., and Pappas, G. J., “Stochastic Safety Verification Using Barrier Certificates,” Proceedings of 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 2004.

[40] S. Prajna and A. Jadbabaie, “Safety Verification of Hybrid Systems Using Barrier Certificates,” in Hybrid Systems: Computation and Control. Heidelberg: Springer-Verlag, 2004.

[41] Bird, R.: RASCLE Version 2.0: Design Specification, Programmer’s Guide, and User’s Guide. Baron Associates, Inc., February, 2002.

[42] Belcastro, Christine, and Belcastro, Celeste, "On the Validation of Safety Critical Aircraft Systems, Part I: Analytical & Simulation Methods", Proceedings of AIAA Guidance Navigation and Control Conference, Austin TX, August 2003.

[43] Schumann, J., Burlet, Pasareanu, C., K. G., Menziers, T., and Barrett, T., “Tool Support for Parametric Analysis of Large Software Simulation Systems,” submitted to Automated Software Engineering Conference, L'Aquila, Italy, Sept. 2008.

[44] Garey, M. R., and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[45] L. G. Crespo and S. P .Kenny, “Robust Control Design for Systems with Probabilistic Uncertainty,” NASA/TP–2005–213531, March 2005.

[46] Jacklin, S. A., Schumann, J., Bosworth, J, Williams, P., and Larson, D., " Test Results of a Tool and Method for In-Flight, Adaptive Control System Verification on a NASA F-15 Flight Research Aircraft,” 7th World Congress on Computational Mechanics, Los Angeles, CA, July 2006.

[47] Bosworth, J. T., and Stachowiak, S. J., "Real-Time Stability Margin Measurements for X-38 Robustness Analysis," NASA/TP-2005-212856, Feb 2005.

[48] Bosworth, J. T. and Burken, J. J., "Tailored Excitation for Multivariable Stability-Margin Measurement Applied to the X-31A Nonlinear Simulation, NASA TM-113085, 1997.

[49] Scherer, S., Lerda, F., Clarke, E., “Model Checking of Robotic Control Systems,” Proceedings of ISAIRAS 2005 Conference, Munich, Germany, Sept. 5-8, 2005
Refs on adaptive flight control applications

[50] C. Cao and N. Hovakimyan. Design and analysis of a novel L1 adaptive control architecture, Part I: Control signal and asymptotic stability. In Proc. of American Control Conference, pages 3397–3402, Minneapolis, MN, June 2006.

[51] Krishnakumar, K., Limes, G., Gundy-Burlet, K., and Bryant, D., ”An Adaptive Critic Approach to Reference Model Adaptation”, AIAA Guidance, Navigation, and Control Conference, AIAA-2003-5790, 2003.

[52] Tao, G., Chen, S. H., Fei, J. T., and Joshi, S. M., “An Adaptive Actuator Failure Compensation Scheme for Controlling a Morphing Aircraft Model,” Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, 2003.

[53] Liu, Y., Tang, X. D., Tao, G., and Joshi, S. M., “Adaptive Failure Compensation for Aircraft Tracking Control Using Engine Differential Model,” Proceedings of the 2006 American Control Conference, Minneapolis, MN, June 2006.

[54] Buffington, J. M., Crum, V., Krogh, B., Plaisted, C., and Prasanth, R., "Verification and Validation of Intelligent and Adaptive Control Systems," 2nd AIAA Unmanned Unlimited Systems Conference, San Diego, CA, Sept. 2003.

[55] J. Totah, K. Krishnakumar, and S. Viken. Stability, maneuverability, and safe landing in the presence of adverse conditions. Report of NASA Integrated Resilient Aircraft Control Project, April 13 2007.

[56] Williams, D.K. and Neilan, P.J., “The Role of Safety Cases in Risk Management,” European Convention on Security and Detection, Brighton, UK, May 1995.

[57] Handbook, Preparation and Evaluation of Safety Cases, Bentham Technical Training Course, published by Balogh International, Inc., May 1994.