Towards Run-time Assurance of Advanced Propulsion Algorithms

Edmond Wong[1]
NASA Glenn Research Center, Cleveland, OH 44114

John D. Schierman[2] and Thomas Schlapkohl[3]
Barron Associates, Inc., Charlottesville, VA 22901

and

Amy Chicatelli[4]
Vantage Partners LLC, Brook Park, OH 44142, USA

This paper covers the motivation and rationale for investigating the application of run-time assurance methods as a potential means of providing safety assurance for advanced propulsion control systems. Certification is becoming increasingly infeasible for such systems using current verification practices. Run-time assurance systems hold the promise of certifying these advanced systems by continuously monitoring the state of the feedback system during operation and reverting to a simpler, certified system if anomalous behavior is detected. The discussion will also cover initial efforts underway to apply a run-time assurance framework to NASA's model-based engine control approach. Preliminary experimental results are presented and discussed.

I. Introduction

Safety and operational requirements for modern aircraft systems call for increasingly advanced and intelligent control capabilities. To meet these demands, researchers have made significant progress towards developing advanced, intelligent control and health management algorithms with the onboard capability to learn, adapt, self-tune and reconfigure. However, before these advanced algorithms can be deployed, it must be assured that the software that implements these systems never instigates instabilities, which can impact the safety of the aircraft. Flight-certification of such systems requires that they undergo thorough verification and validation (V&V) to achieve high confidence in their safety. Unfortunately, rigorous certification of such systems has proven to be challenging and costly using current V&V practices. At present, V&V is still largely based on a process that is dependent upon exhaustive testing. As these algorithms and systems become progressively complex, it is anticipated that test-based V&V methodologies will become prohibitively costly and, ultimately, infeasible at achieving safety confidence.

There are efforts underway to improve the practical applicability of design-time V&V approaches based on formal methods (i.e. theorem proving, model checking, etc.) to provide mathematical proof of the safe execution of highly complex advanced systems. Similarly, advancements in automated testing approaches are expected to have an impact. However, it is uncertain whether these methods will address all of the difficulties associated with achieving the necessary confidence in the use of these algorithms in safety-critical systems. In particular, algorithms that are adaptive, reconfigurable or nondeterministic in nature present the greatest challenge to design-time verification approaches. While there are also current and ongoing efforts to develop analytical proofs and stability/convergence guarantees for some of these algorithms[1, 2, 3], often the assurance results are deemed relatively weak[4] and insufficient in meeting the stringent criteria for safety-critical purposes.

---

[1] Research Engineer, Intelligent Control and Autonomy Branch, 21000 Brookpark Rd., 77-1, AIAA Senior Member.
[2] Principal Research Scientist, 1410 Sachem Place, Suite 202, Charlottesville, VA 22901, AIAA Associate Fellow.
[3] Research Scientist, 1410 Sachem Place, Suite 202, Charlottesville, VA 22901.
[4] Aerospace Engineer, Intelligent Control and Autonomy Branch, 3000 Aerospace Parkway VPL-3, AIAA Senior Member.

There is a growing realization that no single V&V approach, such as, design-time verification or traditional testing, will be sufficient to address all of the challenges associated with providing safety guarantees for these advanced or complex systems. To address this shortfall in V&V capability, some researchers have proposed that run-time monitoring or run-time verification methods can play a complementary and enabling role. The basic premise makes use of "monitors" to observe the execution of an uncertified algorithm in question to insure that resulting system behavior remains constrained within acceptable bounds of stability.

This paper summarizes the preliminary efforts undertaken to investigate run-time assurance methods and their applicability to advanced propulsion control systems. The next section provides background information on the development of advanced engine controls at the National Aeronautics and Space Adminitration (NASA) and the looming certification challenges that provide motivation for investigating run-time approaches as a potential means of providing safety assurance for these algorithms. Next, an overview provides the results of a survey completed on run-time methods, with particular focus on a recently developed run-time assurance framework. Next, a case study devised to apply run-time assurance to NASA's model-based engine control approach is described. Finally, experimental results are presented and discussed.

II. Motivation

In the aircraft propulsion arena, increasing pressures with respect to safety, reliability and cost concerns have placed progressively more demanding requirements upon aircraft engines. These engine requirements include: improving fuel efficiency; extending on-wing engine life; lowering engine maintenance cost by moving from scheduled-based to condition-based engine maintenance; and improved diagnosis and servicing of anomalous behavior. In order to keep pace with these requirements, there is an identified need for improvements in areas such as engine controls and health management[5].

In recent years, engine controls and health management practitioners have increasingly sought to make improvements in turbine engine performance, reliability and cost by turning to onboard estimation as a key enabling technology [6, 7, 8]. In this emerging approach, real-time onboard engine models are employed to provide onboard estimations of unmeasured engine parameters that are relevant to a variety of advanced controls and health management applications[9]. Applications such as model-based controls, model-based diagnostics and onboard engine health estimation have been the focus of much recent activity at NASA Glenn Research Center (GRC).

In the case of diagnostic applications, an onboard engine model can be used to calculate engine parameter outputs based on actual measured inputs to the engine. By comparing the estimated engine parameters to the corresponding measured engine outputs, the resulting differences, or residuals, can be used as indicators of engine faults should they exceed a pre-determined threshold. The residuals can subsequently be used by various fault isolation approaches to determine the fault type.

In the case of control applications, model-based estimation can provide unmeasured engine parameters, such as thrust or stall margins, which are otherwise unavailable due to the lack of onboard sensing technology to measure them. The availability of these parameters allows a control design based on the direct feedback of a desired parameter, such as thrust, rather than indirectly on a less-ideal, but correlated sensed parameter such as fan speed or engine pressure ratio (EPR). The controls based on direct feedback can, generally, be made to be less conservative in design.

However, there are a number of problems related to the use of on-board models to estimate engine parameters. The approach is reliant on the development of high-fidelity engine models with sufficiently high levels of accuracy required for engine performance tracking. Furthermore, over the course of an aircraft engine's life, the level of degradation stemming from normal usage will affect its performance, which may result in model estimations that are inaccurate. Degradation effects are generally described in terms of unmeasurable health parameters such as efficiencies and flow capacities related to each major engine module. Using techniques based on Kalman filters, the level of engine performance degradation can be estimated, given that there are at least as many sensors as parameters to be estimated[10]. Typically, the number of available sensors in an aircraft engine is less than the number of health parameters, which results in an underdetermined estimation problem. Recently, NASA GRC developed an approach based on singular value decomposition that selects a model tuning parameter vector of low enough dimension to be estimated by a Kalman filter[11]. This Optimal Tuner Selection approach[12] has been developed to select the tuning parameters which minimize the estimation error for the unmeasured variables of interest.

NASA GRC and its industry partners are currently incorporating onboard models into a variety of advanced propulsion control and diagnostic algorithms. Onboard models that incorporate onboard parameter tuning, neural network learning[13] or onboard adaptation[14] are either being actively researched or are under consideration for future efforts. Their ability to accommodate system degradation effects, failure effects and other unforeseen system

changes provides advantages over traditional control and diagnostic approaches. Along with their potential advantages, they also present potential challenges. A looming concern being recognized by NASA and its industry partners relates to the feasibility of V&V and subsequent certification of these algorithms for flight critical deployment. One approach under investigation at NASA GRC to address the safety assurance of advanced propulsion algorithms is the application of run-time assurance methods to onboard model-based, closed-loop control.

III. Overview of Run-time Monitoring

Run-time verification is an analysis approach in computer science concerned with observing the execution of a running system (i.e. software program) to detect whether execution behavior satisfies or violates specified correctness properties. This execution checking is performed by constructs called monitors. Run-time monitors have been used extensively to augment design-time model checking of high-level language programs.[15, 16] A number of surveys and overviews[17] of run-time verification may be found in the literature.

Run-time monitoring research has been largely focused on application towards general software. However, some researchers have begun to study the application of run-time monitoring towards real-time software. Application in the context of real-time execution raises the interesting possibility that once a monitor detects a property violation, it can then take some remedial action such as provide an alert or even influence subsequent execution.[18] Similarly, the monitor can be used to actually enforce an expected behavior to avoid violations.[19]

More recently, there have been a number of efforts focused on adapting run-time monitoring to the verification of embedded systems where software interactions are often tightly coupled to the hardware.[20] Related in focus, have been recent investigations into the application of run-time monitoring to distributed real-time systems and safety-critical systems.[21] Recent efforts have also addressed the resource utilization challenges posed by the run-time monitoring of hard real-time systems, where the addition of monitoring functionality must not adversely impact the execution of the target functions.[22]

Increasingly, run-time monitoring is being viewed as an important part of an integrated approach to the V&V and certification of future safety-critical systems—in conjunction with design-time verification and testing.[23] Some researchers have gone so far as to propose that the creation of monitors for a system to guarantee formally specified safety properties will allow certification (at least in part) to be performed at run-time.[24, 25]

A. Run-time Assurance Framework

Of particular interest to NASA GRC is the Run-Time Assurance (RTA) architecture developed by researchers at Barron Associates, Inc. in Charlottesville, VA.[26, 27] This approach is designed to provide safety assurance for systems that consist of adaptive, non-deterministic or other advanced control algorithms that are costly, difficult or impossible to certify using current design-time verification approaches. The RTA framework (**Figure 1**) achieves this assurance by employing a safety monitor that continuously checks the states of the system to ensure that they remain within predetermined safe operating limits. If a violation of the system safety bounds is detected, the RTA framework switches the control to a fully-certified backup system and enables system recovery. This RTA framework was applied to an autonomous, flight-critical auto-land system under an Air Force SBIR Phase II contract.[28, 29]

The primary components of the RTA framework are as follows:

Primary System—the advanced controller that is responsible for achieving performance objectives. Any advanced, adaptive, non-deterministic, etc. algorithms that cannot be certified at design time are a part of the primary system.

Backup System (or Fail-Safe)—a simplified controller where the emphasis is on safety as opposed to performance. Because of the simplicity of the backup system, it is verified at design time and fully certifiable.

RTA Monitor—continually observes the overall state of the system. If it is determined

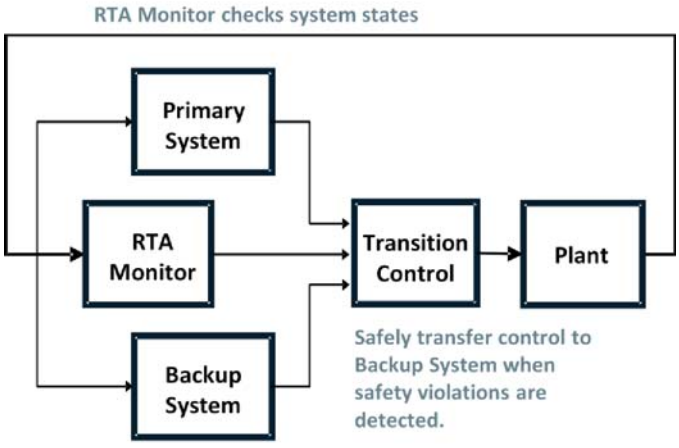

Figure 1. The Run-time Assurance Framework.

that safety violations are impending while under the continued operation of the advanced controller, then the primary system will be disabled and control will be switched to the backup system.

Transition Controller—once enacted by the RTA monitor, transfers control to the backup system. Depending on the application, special logic may be required to ensure that control is transferred in a safe and stable manner to avoid unsafe transients or prevent control rate/position limits from being saturated.

With the exception of the primary advanced system, all components will need to undergo design-time V&V to the required certification criticality level. Although, it may be too difficult or costly to certify the advanced system to the same level as the other components in the feedback system, some V&V analysis should be performed to the degree possible in order to complement the "required certification level equivalency" that is envisioned as being provided by the RTA system during run-time operation.

The determination of when a safety reversion should occur is the key aspect to developing an RTA framework. By definition, this process is highly application specific, as the boundaries for a system's safe operating envelope are defined by safety limits (e.g. structural limits, component limits), performance limits and operational limits. In prior efforts, Barron Associates has investigated application of RTA to flight control and guidance and discovered that safe flight regions are complex functions of many system states and critical parameters of the vehicle's dynamics. In such a case, the safe boundary limits are defined by complex, multi-dimensional operating envelopes.

IV. Advanced Algorithm Case Study

NASA worked with Barron Associates on a limited initial case study to investigate the feasibility of using the RTA approach to supervise the execution of a NASA-developed model-based engine control (MBEC) algorithm incorporating a self-tuning tracking filter for estimating unmeasured engine parameters.[30, 31] The tracking filter includes an optimal tuner estimation routine based on a Kalman filter design.[32] This advanced controller directly feeds back an estimate of thrust to control thrust directly, rather than indirectly through other sensed engine parameters. This methodology has the capability to update tuning parameters, which ensures continued correlation between the engine and model parameter estimates over the life of the engine; thus, overcoming a major concern with directly closing the control loop on unmeasured estimates such as thrust. This allows for the deviations in the controlled thrust that naturally vary over the life of the engine to be controlled within a tighter band. This new tuning approach will provide a more accurate measure of thrust due to its ability to account for deterioration when compared to estimation models developed in the past.

By contrast, traditional engine control designs[33] regulate a sensed parameter, commonly either fan shaft speed (Nf) or engine pressure ratio (EPR), in order to provide power management (**Figure 2**). The sensed parameter is indirectly correlated with thrust and used in the control algorithm and logic, since thrust itself is not measurable. Protection logic is added to ensure engine stability and safe operation throughout the life of the engine by limiting the maximum fuel flow. These limits are typically based on structural and operational constraints. One important constraint is the prevention of high pressure compressor stall during quick accelerations or large changes in thrust demand. This constraint is typically accomplished by implementing an acceleration schedule, which controls engine core acceleration as a function of the current core speed. This schedule is usually determined experimentally to accommodate a range of environmental conditions and engine deterioration levels to provide ample HPC stall margin over all operational conditions and engine conditions, which results in an overly-conservative stall margin for newer engines since it accommodates an end-of-life (EoL) engine.

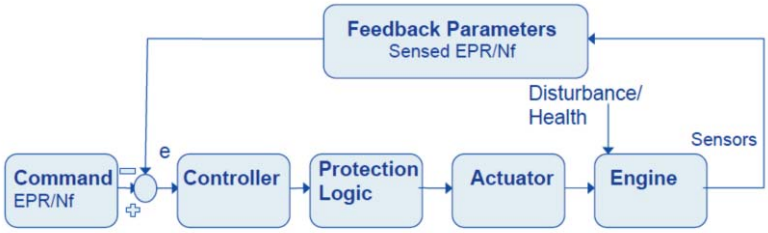

Figure 2. Traditional full authority engine controller that regulates a sensed parameter, either EPR or Nf.

MBEC attempts to reduce the operability margins for newer engines by employing limiter protection logic that uses direct stall margin estimates from the aforementioned optimal tuner Kalman filter (OTKF). This onboard model can provide a more accurate margin for the actual condition of the engine. The resulting benefit is a potential increase in fuel efficiency while maintaining safe operation during transient changes. To enable the simulation study presented in this paper, a MBEC simulation platform architecture was realized that comprises three main components: an engine or "truth" model, shown in **Figure 3** as the Engine block; an on-board estimator model designed to provide real-time estimates of desired unmeasured parameters, labeled as the Estimation block; and a thrust controller with limit protection logic, shown as Thrust Controller. The NASA-developed Commercial Modular Aero-Propulsion System Simulation 40,000 (C-MAPSS40k)[34] high-fidelity engine simulation will serve as the engine to which MBEC is applied.

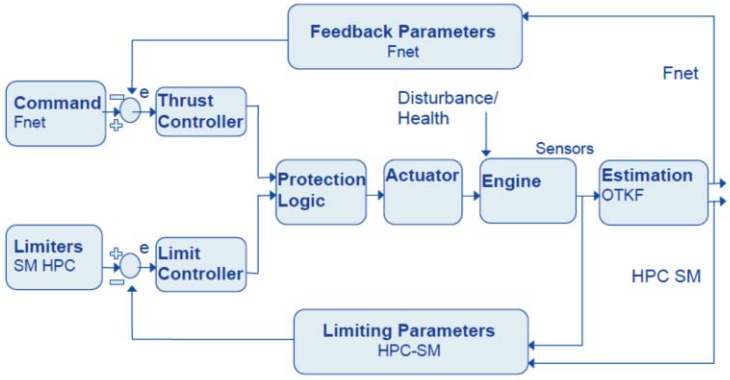

Figure 3. MBEC engine control model architecture.

A. RTA/MBEC Integration

The RTA architecture was integrated with the nonlinear MBEC/C-MAPSS40k simulation model as illustrated in **Figure 4**. A select set of sensed outputs from the engine (discussed in detail in the next subsection) were passed to the RTA system to be continually monitored and checked by the RTA Monitor. The RTA Monitor outputs a boolean flag, labeled "inRAE", to the Transition Control. If all of the sensed values remain within the predefined safety limits, the flag outputs a true value and the engine continues to operate with the advanced thrust controller. In addition, the protection logic continues to use the stall margin limit based on estimated HPC stall margin from the OTKF. When unsafe conditions cause the RTA Monitor to issue a false value on the "inRAE" flag, the Transition Control reverts to the safe backup EPR Controller. In addition, the protection logic switches to using the backup system's traditional acceleration schedule to protect against HPC stall.

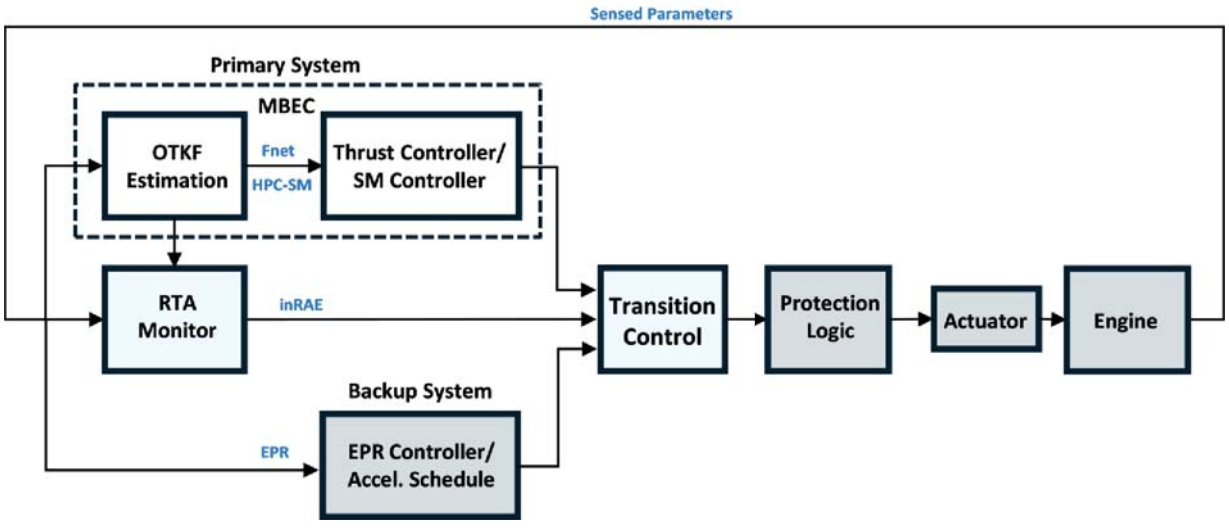

Figure 4. RTA integrated with MBEC architecture.

A note regarding the aforementioned protection logic within the MBEC simulation—it uses similar limits as the RTA to ensure safe operaton of the engine by over-riding the fuel flow rate command from the controller when a prescribed limit is reached. For the purposes of this study, it was decided that the limits for the parameters monitored

by the RTA would be chosen to be more conservative than the limits used by the protection logic, thus ensuring that control mode switching occurs before activation of fuel-flow over-rides. Future work, beyond the scope of this initial investigation, may focus on investigating how these two mechanisms might be better integrated.

B. Monitored Parameters

For the purposes of this preliminary study on the application of the RTA approach to turbofan engine control, the safe operating envelope was defined by the physical limits of the key engine parameters, such as rotor speeds, temperatures and pressures at the various stages through the engine. The boundaries of the safe operating envelope provides the switching conditions that determine whether engine control should be switched from the advanced thrust based controller to the reversionary EPR controller. Future studies may investigate whether additional, possibly more complex, multi-dimensional, limit functions should also be considered.

The key engine parameters chosen for monitoring by the RTA are those associated with operating the engine in a safe manner as well as operating within its performance boundaries. The safety limits that have been identified are:

1. Fan speed less than 4,200 rpm (Nf_max)
2. Core speed less than 12,200 rpm (Nc_max)
3. High pressure compressor discharge pressure less than 433 psi (Ps3_max)

In addition, performance boundaries are identified by operational parameters and their limits:

1. HPC stall margin greater than 15%
2. LPC stall margin greater than 6%
3. RU limit greater than 17% to prevent lean combustor blow out

In addition to the parameters identified from the engine, there are parameters from the OTKF Estimation subsystem that were deemed useful. In this case, a threshold could be established for the error between the true engine values and those estimated by the Kalman filter (i.e.the residuals). When any error exceeded the defined threshold, the RTA would determine that the thrust controller should be switched to the traditional engine controller. The maximum limits for all residuals were chosen at 1% to provide adequate protection for the examples analyzed during this project, without being overly conservative:

1. Core speed – Nc
2. Fan speed – Nf
3. High pressure compressor discharge temperature – T30
4. Low pressure turbine discharge temperature – T50
5. High pressure compressor discharge pressure – Ps3
6. Low pressure turbine exit pressure – P50

C. Switching Logic

During several simulation experiments, the monitored states were often observed to briefly exceed their limits but then quickly return to within their defined safe values. Such spurious events were observed to occur only once during a simulation run or could occur periodically, depending on the parameter and the scenario. Requiring the engine to continue using the reversionary controller for the duration of the mission after its first activation would ensure safety, but might be overly conservative. Allowing the RTA system to switch back to the advanced controller may be better for performance, especially if the limit violation was a one time or rare occurrence (e.g. during some transient event). However, for the situations in which the parameter always tends to violate its limit after switching back to the advanced controller will result in periodic switching between advanced and baseline controllers which can cause oscillatory behavior in the engine responses. The solution implemented here is to allow the RTA system to switch back from the backup controller to the advanced controller only after all limits have been satisfied for a consecutive specified amount of time. Currently a value of one second is used.

In future work, other more complex logic may be developed, such as delaying the switch to the reversionary controller in case the parameter violation is transient, or only allowing a specified number of control mode switches before permanently switching to the baseline controller.

V. Results

The following simulation experiments were performed using operating profiles for take-off and cruise. During the take-off profile, the PLA was linearly increased from 43 to 80 degrees over a 5 second period. Initial conditions for this profile included a Mach number of 0.0, and an altitude of 0 ft. In a typical take-off scenario, the change in PLA takes place over a much shorter period (e.g. 0.15 seconds). This very rapid change always resulted in the

protection logic overriding the control; therefore, the slower change over 5 seconds was chosen for the purposes of RTA testing.

During the cruise profile, the PLA was linearly increased from 60 to 70 degrees over a 5 second period. Initial conditions included a Mach number of 0.7, and an altitude of 30,000 ft.

The experiments undertaken focused on two main investigations which can result in the RTA system enacting a control mode switch to the baseline controller: sensor errors, and coding errors within the OTKF. These experiments are discussed in detail below.

A. Sensor Errors

The RTA system was designed to identify parameter limit violations which can be caused by sensor errors/failures within the engine because such errors may cause poor Kalman filter estimates, resulting in the engine to select an incorrect process model, reach an unsafe operating condition, or have poor performance. To test the impact of sensor errors, a variety of error types were applied to the sensor measurements delivered to the Kalman Estimator and RTA algorithms. The possible sensor errors investigated were:

1. Sensor bias
2. Sensor drift (bias linearly increasing with time)
3. Additional Gaussian noise added to sensor output
4. Sensor output multiplied by gain
5. Sensor output set to constant value
6. Sensor output set to constant value with additional Gaussian noise

Each sensor error can be started at a desired time during the simulation. A select subset of the cases that were analyzed are presented here to demonstrate the benefits of using the RTA system. It soon became apparent that monitoring the residuals provided early detection of abnormal behavior by the thrust controller, and switching to the EPR controller after the 1% residual limit was violated for one or more of the parameters usually kept the system from activating the protection logic. In other cases, residual monitoring identified minor sensor errors that did not cause the engine to reach one of its safety or operational limits, but still resulted in undesired thrust.

1. Bias Error on Fan Speed Sensor

The following case involves a bias error of -500 RPMs in the fan speed sensor, introduced 12 seconds into the cruise profile. In this case, PLA is increased from 60 to 70 degrees over the course of 5 seconds, from 15 to 20 seconds in the simulation run.

For the initial test runs, the RTA system was configured to not monitor the OTKF Estimation residuals, but instead only rely on the safety and operational limits (discussed in Section IV B).

The first plot presents the inRAE Boolean, where,

inRAE = 1 => true, => no parameter has violated its limit

inRAE = 0 => false => at least one parameter has violated its limit

The remaining plots present select engine parameters corresponding to:

(a) the baseline EPR controller, denoted ‘EPR’ in the legend

(b) the advanced thrust controller, denoted ‘Fnet’ in the legend, and

(c) the system with RTA performing parameter monitoring, denoted ‘RTA’ in the legend.

For this experiment, the RTA system was allowed to switch back to the advanced thrust controller once all parameter violations no longer exist for at least one second. It can be seen in **Figure 5** that the HPC stall margin estimate violates its limit first at ~ 16 or 17 seconds, causing the RTA system to switch to the EPR controller. Then, at approximately 18 seconds the RTA system switches back to the advanced controller, as the stall margin is no longer violated. However, in approximately one second the RTA system again switches back to the EPR controller. This “back and forth” switching between the advanced thrust controller and the baseline EPR controller persists throughout most of the simulation run. More advanced switching logic should stop this behavior after a desired number of switches and would then simply remain with the EPR controller running. The plot of the engine’s fan speed (Nf) in **Figure 6** clearly display that this “limit cycle” behavior leads to the undesirable oscillatory response characteristics. Note in **Figure 7** that the “truth” values of the HPC stall margin are plotted and violations of its limit do not necessarily correspond to when the RTA switches to the baseline EPR controller because the RTA system bases its decision on the estimate of the stall margin (not plotted).

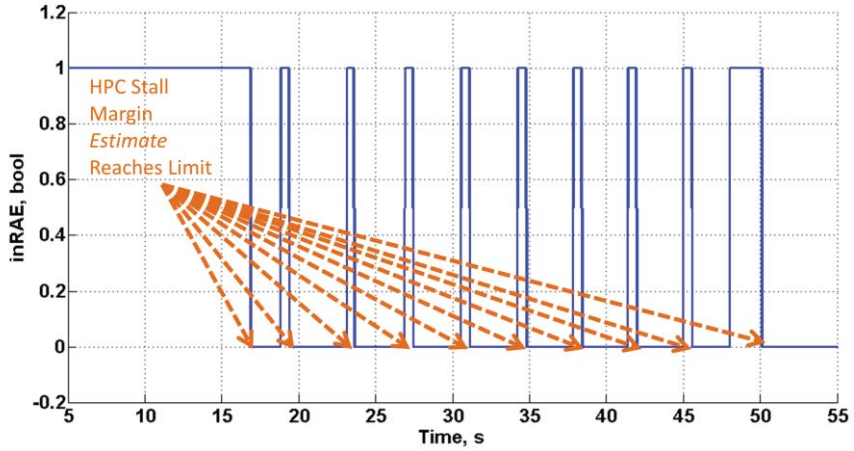

Figure 5. inRAE vs. Time, (Nf Bias Error = -500 rpm), (Residual Monitoring Off)

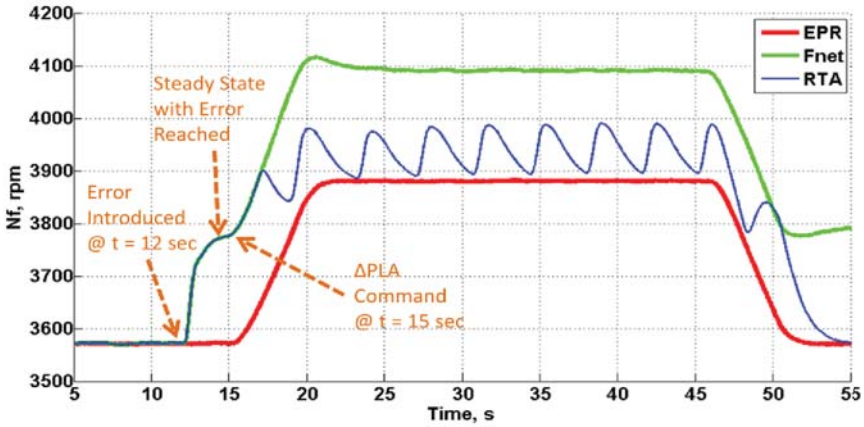

Figure 6. Nf vs. Time, (Nf Bias Error = -500 rpm), (Residual Monitoring Off)

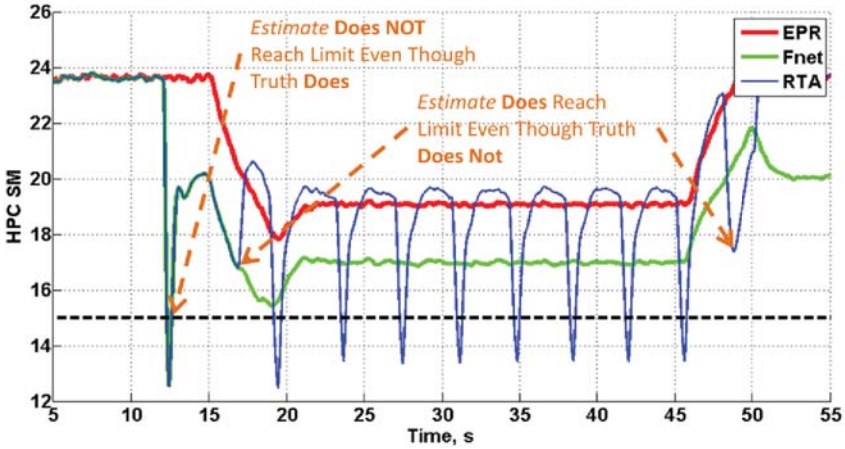

Figure 7. HPC SM vs. Time, (Nf Bias Error = -500 rpm), (Residual Monitoring Off)

The plots in **Figure 8** and **Figure 9** correspond to the case where the RTA system does monitor the Kalman filter residuals. As can be seen, monitoring residuals provides earlier and more consistent indication of anomalies. For much of the simulation, the residuals exceed the 1% limit; therefore, the RTA system keeps the EPR controller running. In this case, there are only two occurrences in which the RTA system switches back to the advanced thrust controller. This occurred once all residuals were less than 1%. Reducing residual limits to less than 1% will keep the RTA from switching back to the advanced controller. Again, additional logic could be introduced such that the EPR controller remains active after say, three or four occurrences of these rapid switching phenomena.

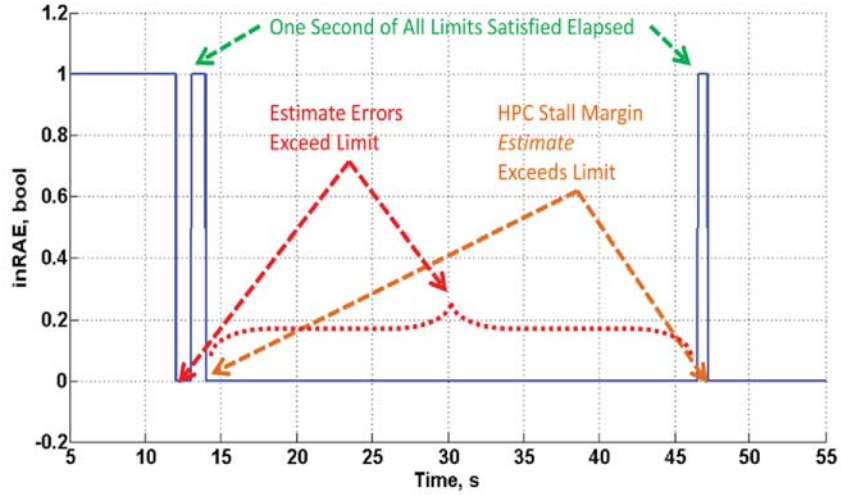

Figure 8. inRAE vs. Time, (Nf Bias Error = -500 rpm), (Residual Monitoring On)

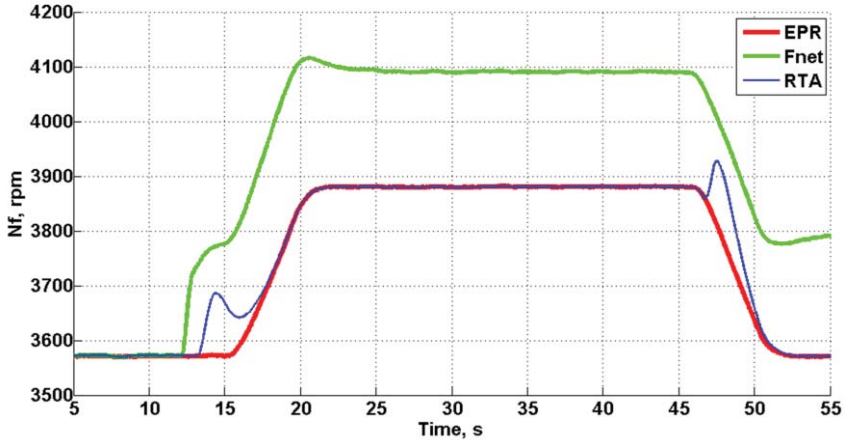

Figure 9. Nf vs. Time, (Nf Bias Error = -500 rpm), (Residual Monitoring On)

B. Coding Errors

Complicated, advanced algorithms are often subject to coding errors. The RTA system should also be able to address coding errors that result in limit violations. To simulate coding errors, sign errors were introduced within the Kalman filter. Although these errors are "contrived," and, in reality, such errors would clearly be identified at design time, they are presented here for demonstration purposes to show the benefit of the RTA system in protecting engine operations under an actual, significantly complex controller.

1. Sign Errors in the Kalman Filter

Sign errors were introduced in select locations within the Kalman filter, and were tested to see what impact they had on engine performance, and how the RTA system responded. **Figure 10** presents the Simulink model screen capture that shows two cases that have been analyzed (sign errors on the *Δy* and *DΔu* terms). For both cases, at 20 seconds into the simulation, the sign starts changing from positive 1 to negative 1 linearly over a 5 second time span.

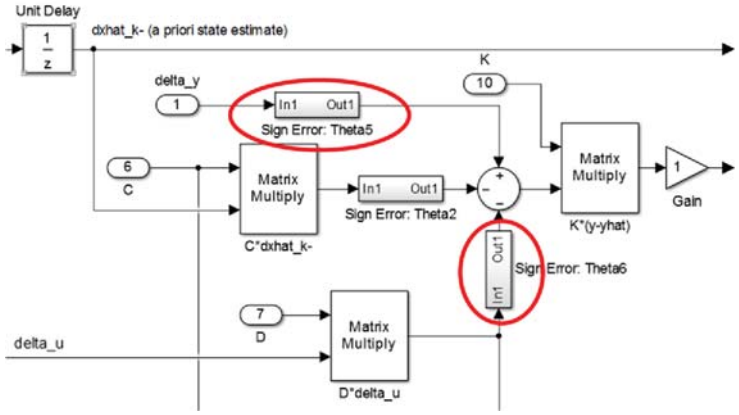

Figure 10. Sign Errors Introduced within Kalman Filter

Take-off (-kΔy)

The following case occurs during the take-off profile and involves a coding error caused by a sign error on the *Δy* term within the Kalman filter. Here, the PLA is increased from 43 to 80 degrees from 25 to 30 seconds into the run. The error is turned on at t = 20 seconds in the simulation. Following the introduction of the coding error, the Kalman filter residuals quickly grow to exceed the 1% limit. Without the RTA system, the coding error causes the states to increase until the protection logic's limiter on the high pressure compressor discharge pressure (Ps3) reaches the safety limit (see **Figure 11**). At that point, the limiter overrides the thrust controller. Therefore, this case shows that the RTA system keeps the protection logic system from being activated.

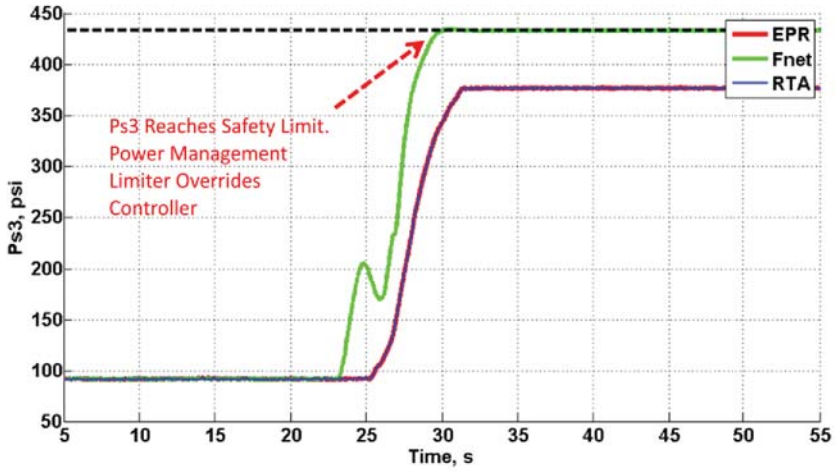

Figure 11. Ps3 vs. Time, Take-off, (-kΔy Coding Error)

Take-off (-kDΔu)

The following case occurs during the take-off profile and involves a coding error caused by a sign error on DΔu term within the Kalman filter. Again, (a) the PLA is increased from 43 to 80 degrees from 25 to 30 seconds into the run, (b) the error is turned on at t = 20 seconds in the simulation.

With the same conditions before the error is introduced, the engine again experiences two false alarms from which it recovers from by returning to the advanced controller. Following the introduction of the coding error, the Kalman filter estimate errors quickly grow to exceed the 1% limit and the RTA system safely switches to the EPR controller. Without the RTA system, it can be seen that the engine parameters exhibit undesirable oscillatory behavior (**Figure 12**). This behavior results in poor estimates of the HPC stall margin, which, activates the stall margin limiter within the protection logic. This limiter, in turn, commands the fuel flow limit override. The poor HPC stall margin estimate causes the maximum fuel flow command to oscillate, resulting in the behavior seen in the plots.

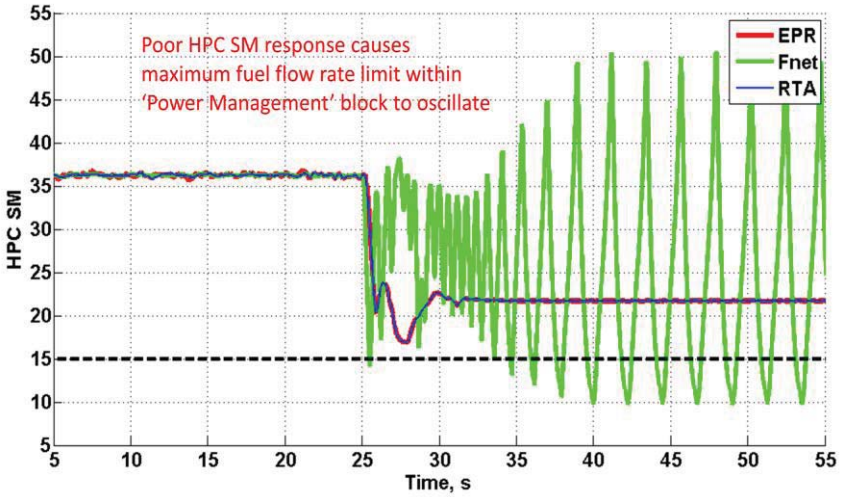

Figure 12. HPC SM vs. Time, Take-off, (-kDΔu Coding Error)

Cruise (-kΔy)

The following case occurs during the cruise profile and involves a coding error caused by a sign error on the Δy term within the Kalman filter. Here, (a) the PLA is increased from 60 to 70 degrees from 15 to 20 seconds into the run, (b) the error is turned on at t = 20 seconds in the simulation. Here too, following the introduction of the coding error, the Kalman filter residuals quickly grow to exceed the 1% limit, and the RTA system safely switches to the EPR controller and continues using it for the duration of the simulation. Without RTA, the coding error causes the engine to abruptly switch between the RU minimum limiter and the maximum limiters within the protection logic. Initially, the RU limit (**Figure 13**) is violated, so the engine uses the override value and decelerates until the HPC stall margin limit (**Figure 14**) is violated. At that point a "bang-bang control" type of behavior is exhibited in which the protection logic switches the override command from the minimum allowed fuel flow rate as defined by the RU limiter to the maximum allowed fuel flow rate as defined by the HPC stall margin limiter. This continues to occur nearly every 10 seconds. It is interesting that the same error during the previously shown take-off profile did not result in this type of response. This indicates that coding errors can result in different responses for different operating conditions using the advanced controller.

Cruise (-kDΔu)

The following case occurs during the cruise profile and involves a coding error caused by a sign error on DΔu term within the Kalman filter. Again, (a) the PLA is increased from 60 to 70 degrees from 15 to 20 seconds into the run, (b) the error is turned on at t = 20 seconds in the simulation. Following the introduction of the coding error, the Kalman filter residuals quickly grow to exceed the 1% limit, and RTA switches to the EPR controller and continues using it for the duration of the simulation. Without RTA, the effect on the thrust controller is much less than what was observed during takeoff. However, high frequency oscillations are seen in several of the parameters, which do not occur in the EPR controller, such as in HPC SM (**Figure 15**) and RU (**Figure 16**).

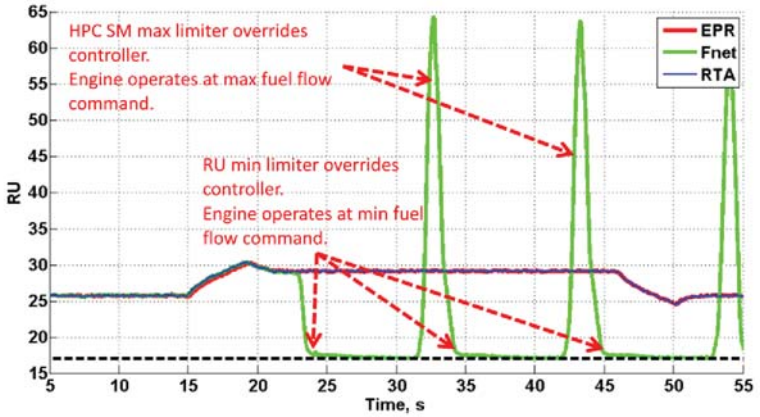

Figure 13. RU vs. Time, Cruise, (-kΔy Coding Error)

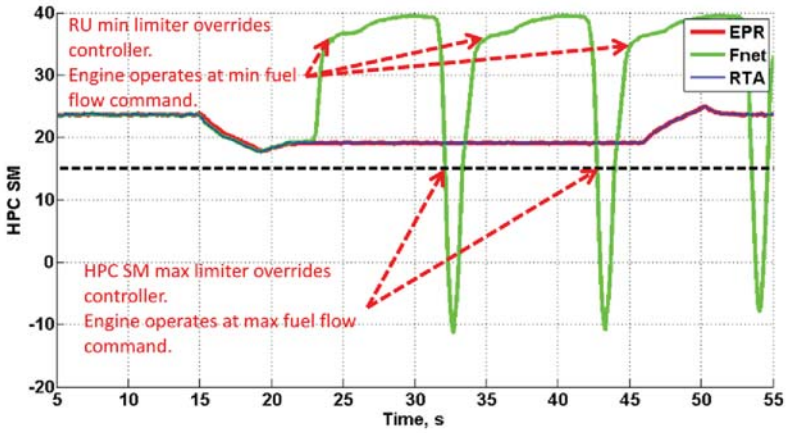

Figure 14. HPC SM vs. Time, Cruise, (-kΔy Coding Error)

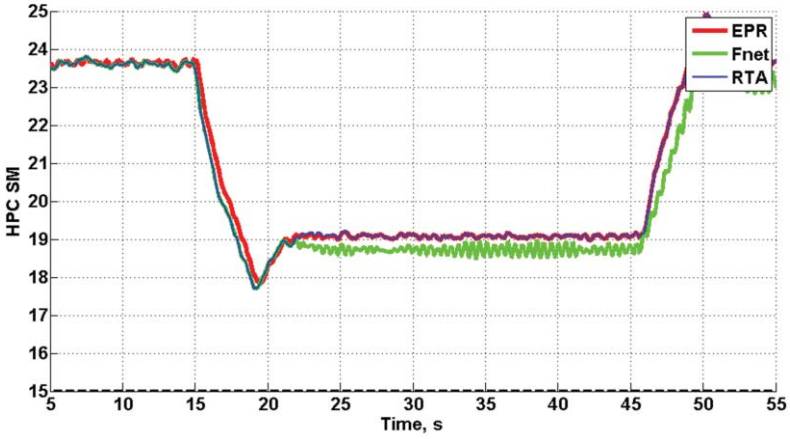

Figure 15. HPC SM vs. Time, Cruise, (-kDΔu Coding Error)

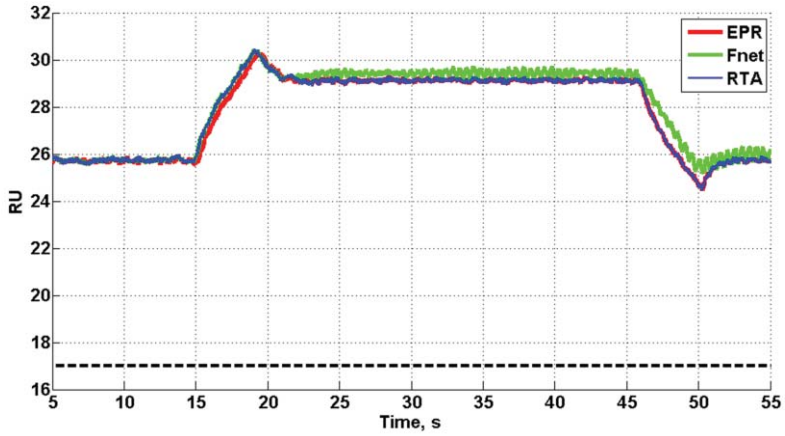

Figure 16. RU vs. Time, Cruise, (-kDΔu Coding Error)

C. Results Discussion

Preliminary simulation experiments were performed to investigate sensor errors and coding errors that would cause the advanced controller to exceed certain parameter limits. It was found that placing limits on the Kalman filter residuals provided critical lead time in identifying errors in the advanced controller because large residuals indicate either poor sensor information or incorrect estimation by the filter. Without monitoring the residuals, operational or safety limit violations are needed to trigger a control mode switch to the baseline EPR controller and in some cases this can lead to triggering the protection logic limiters.

In some cases, temporary limit violations were observed during transient operations. To accommodate these spurious events that trigger false alarms, the RTA framework was allowed to switch back to the advanced controller once limit violations are not detected for a certain length of time. Note that this RTA design decision should be made on an application by application basis. There may be applications where it is preferable for the RTA to never allow switching back to the advanced controller once the system has reverted to the backup controller.

This framework exposed another advantage of monitoring the Kalman filter residuals. If they were not monitored, and there really was a problem with the advanced controller, then scenarios could easily be constructed in which the RTA system would continuously switch between the baseline and advanced controller in a "limit cycle" fashion because the safety and operational limits would be satisfied after a certain period of time under the EPR controller. By monitoring the filter residuals, these parameters would almost always violate their limits under sensor or coding errors, which would stop the continuous switching between the backup and advanced controller.

VI. Future Work

This small study was a preliminary investigation into benefits provided by RTA systems for advanced engine control. It was observed in some experiments that limit violations can occur intermittently due to transient control or other temporary environmental conditions, suggesting a design choice whereby the RTA should be allowed to switch back to the advanced controller if conditions warrant. However, there may be a need for additional logic that characterizes the limit violations so that false alarms are minimized and real unsafe conditions are quickly recognized and recovered from. One rule may be that the EPR controller permanently takes control only after a certain number of control mode switches are recorded. Other logic might analyze the rate at which a limit violation occurs (large rates indicate a severe problem), or the magnitude of the limit violation (large magnitudes would indicate a severe problem). Noisy signals that may be close to a limit, but otherwise nominal, could have a number of violations of small magnitude. Logic will need to be constructed so that such cases do not cause a false alarm.

Another area for further study is determining how much margin should be given in defining parameter limits so that safety is assured, but without excessively restricting the advanced controller's operating envelope. In many cases, it has been observed that introduction of an error can cause rapid, large transients that quickly approach parameter limits, yet it takes time for the parameter values to settle to nominal values once the control mode switch is triggered, causing overshoots that eventually violate the limit. This can be due to the spool up/spool down momentum of the fan and compressors, for example. Further study of this problem is warranted. In other applications of RTA systems, more advanced logic for the transitional control has been studied that stably and safely

takes the system state from the advanced controller to the baseline controller. Such advanced transitional controllers may have valuable benefit in engine control applications as opposed to the simple control switching used in this study.

VII. Conclusion

Advanced aircraft engine controllers are becoming increasingly difficult to certify using current verification practices. This realization has motivated preliminary efforts to investigate run-time assurance methods and their applicability to advanced propulsion control systems. A survey of current run-time monitoring methods was undertaken to assess the potential applicability of this approach as a means of providing safety assurance for advanced propulsion control algorithms. Of particular interest for adoption is the Run-Time Assurance (RTA) framework recently developed for a flight control application. A case study was initiated to investigate the application of this approach to NASA's Model-Based Engine Control (MBEC) architecture. Under this study, a RTA framework was integrated into a MBEC simulation employing NASA's Commercial Modular Aero-Propulsion System Simulation 40,000 (C-MAPSS40k) turbofan engine simulation platform. RTA employs a safety monitor which continually checks that the safety, operational, and Kalman filter residual limits are satisfied. If a limit is violated, the RTA system switches control from the advanced thrust controller to the reversionary Engine Pressure Ratio controller. For this initial effort to apply the RTA methodolgy to turbofan engine control, the simple decoupled physical limits of the key engine parameters were used to define the boundaries of the safe operating envelope by which control mode switching decisions are made. The switching logic comprises simple checks of each monitored parameter to ascertain that they maintain values that remain within their defined upper and lower limits. Future studies may investigate the necessity for additional, possibly more complex, multi-dimensional, limit functions that may require more complex switching logic to search through a complicated operating space. Preliminary simulation experiments performed under this effort show the potenial benefit of the RTA system in protecting engine operations when using advanced controllers that may, otherwise, be too difficult to certify using traditional verification methods. With simple relationships defining the parameter limits, as defined in this initial effort, the code required to construct the RTA system should be fairly straightforward and easily certified at design time.

Acknowledgements

The authors wish to acknowledge Joseph W. Connolly for providing insight regarding the Model-Based Engine Control architecture and Don L. Simon for sharing his expertise on the Optimal Tuner Kalman Filter.

References

[1]Schumann, J., Gupta, P., "Monitoring the Performance of a Neuro-adaptive Controller". Proceedings of the 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, 2004.

[2]Schumann, J., Gupta, P., Jacklin, S., "Toward Verification and Validation of Adaptive Aircraft Controllers", IEEE Aerospace Conference, 2005.

[3]Schumann, J., Liu, Y., "Tools and Methods for the Verification and Validation of Adaptive Aircraft Control Systems", IEEE Aerospace Conference., 2007.

[4]Brat, G., "Verification & Validation for PHM", PHM 2010, 2010

[5]Litt, J. S., Simon, D. L., Garg, S., Guo, T.-H., Mercer, C., Miller, R., et al. "A Survey of Intelligent Control and Health Management Technologies for Aircraft Propulsion Systems." Technical Report NASA/TM 2005-213622 .

[6]Chatterjee, S. and Litt, J. "Online Model Parameter Estimation of Jet Engine Degradation for Autonomous Propulsion Control", AIAA Guidance, Navigation, and Control Conference and Exhibit, 2003.

[7]Viassolo, D., Adibhatla, S., Brunell, B., Down, J., Gibson, N., Kumar, A., Mathews, H., Holcomb, L., "Advanced Estimation for Aircraft Engines", Proceedings of the American Control Conference, 2007.

[8]Kumar, A., and Viassolo, D., "Model-Based Fault Tolerant Control," NASA/CR—2008-215273, 2008.

[9]Behbahani, A., Adibhatla, S., Rauche, C., "Integrated Model-based Controls and PHM for Improving Turbine Engine Performance, Reliability, and Cost", 45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, 2009.

[10]España, M.D., "Sensor Biases Effect on the Estimation Algorithm for Performance-Seeking Controllers", Journal of Propulsion and Power, Vol. 10, No. 4, 1994.

[11]Litt, J.. "An Optimal Orthogonal Decomposition Method for Kalman Filter-Based Turbofan Engine Thrust Estimation", Journal of Engineering for Gas Turbines and Power, 2008.

[12]Simon, D. L., & Garg, S., "Optimal Tuner Selection for Kalman Filter-Based Aircraft Engine Performance Estimation," NASA TM 216076, 2010.

[13]Volponi, A., "Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking," NASA/CR—2008-215272, 2008.

[14]Wood, C. B., "Perspective on Controls and Diagnostics. Pratt and Whitney", 3rd NASA Glenn Research Center Propulsion Control and Diagnostics Workshop, 2012.
[15]Havelund, K., Rosu, G., "Java PathExplorer - A Runtime Verification Tool", The 6th International Symposium on AI, Robotics and Automation in Space, 2001.
[16]Havelund, K., Rosu, G., "An Overview of the Runtime Verification Tool Java PathExplorer", Formal Methods in System Design, 2004.
[17]Leucker, M., Schallhart, C., "A Brief Account of Runtime Verification", Journal of Logic and Algebraic Programming, 2008.
[18]Kim, M., Lee, I., Sammapun, U., Shin, J., Sokolsky, O., Monitoring, Checking and Steering of Real-Time Systems", Proceeding of 2nd International Conference on Runtime Verification, 2002.
[19]Falcone, Y., "You Should Better Enforce than Verify", Proceedings of the First international conference on Runtime verification, 2010.
[20]Zhu, H., Dwyer, M., Goddard, S., "Predictable Runtime Monitoring", 21st Euromicro Conference on Real-Time Systems, 2009.
[21]Goodloe, A., Pike, L., "Monitoring Distributed Real-Time Systems: A Survey and Future Directions", NASA/CR—2010-216724, 2010.
[22]Pike, L., Goodloe, A.,Morisset, R., Niller, S., "Copilot: A Hard Real-Time Runtime Monitor", International Conference on Runtime Verification, 2010.
[23]Callow, G., Watson, G., Kalawsky, R., "System Modelling for Run-time Verification and Validation of Autonomous Systems", 2010.
[24]Rushby, J., "Runtime Certification", Eighth Workshop on Runtime Verification: RV08, 2008.
[25]Rushby, J., "Just-in-Time Certificiation", 12th IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS), 2007.
[26]Bateman, A., Elks, C., Ward, D., Schierman, J., "New Verification and Validation Methods for Guidance/Control of Advancd Autonomous Systems", Proceedings of the AIAA Infotech@Aerospace, 2005.
[27]Schierman, J.D., Ward, D.G., Dutoi, B.C., Aiello, A., Berryman, J., Devore, M., "Run-Time Verification and Validation for Safety-Critical Flight Control Systems", Proceedings of the AIAA Guidance, Navigation and Control Conference, 2008.
[28]Ward, D., Schierman, J., Dutoi, B., Aiello, M., Berryman, J., Grohs, J., DeVore, M., Storm, W., Wadley, J., Tallant, G., "Run-Time Validation and Verification (V&V) For Safety-Critical Flight Control Systems", Barron Associates Final Report to the Air Force Research Laboratory, AFRL-RB-WP-TR-2009-3071, 2009.
[29]Aiello, A., Berryman, J., Grohs, J., Schierman, J., "Run-Time Assurance for Advanced Flight-Critical Control Systems", Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2010.
[30]Connolly, J., Chicatelli, A., and Garg, S., "Model-Based Control of an Aircraft Engine using an Optimal Tuner Approach," 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, No. AIAA 2012-4257, 2012.
[31]Connolly, J., Csank, J., Chicatelli, A., Kilver, J., "Model-Based Control of a Nonlinear Aircraft Engine Simulation using an Optimal Tuner Kalman Filter Approach," 49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference," No. AIAA 2013-4002, 2013.
[32]Simon, D. L., "An Integrated Architecture for On-Board Aircraft Engine Performance Trend Monitoring and Gas Path Fault Diagnostics. NASA TM 216358, 2010.
[33]Csank, J., Ryan, M., Litt, J. S., and Guo, T., "Control Design for a Generic Commercial Aircraft Engine," Technical Report NASA/TM 2010-216811, 2010.
[34]May, R., Csank, J., Litt, J. S., and Guo, T., "Commercial Modular Aero-Propulsion System Simulation 40K," Technical Report NASA/TM 2010-216810, NASA, 2009.