Runtime Assurance Protection for Advanced Turbofan Engine Control*

John D. Schierman,† David A. Neal,‡
Barron Associates, Inc., Charlottesville, VA, 22901

Edmond Wong,§
NASA Glenn Research Center, Cleveland, OH 44135

Amy Chicatelli**
Vantage Partners LLC, Brookpark, OH 44142

Abstract

This paper describes technical progress made in the application of run time assurance (RTA) methods to turbofan engines with advanced propulsion control algorithms that are employed to improve engine performance. It is assumed that the advanced algorithms cannot be fully certified using current verification and validation approaches and therefore need to be continually monitored by an RTA system that ensures safe operation. However, current turbofan engine control systems utilize engine protection logic for safe combustion dynamics and stable airflow through the engine. It was determined that the engine protection logic should continue to be used to provide system safety and should be considered as a part of the overall RTA system. The additional function that an RTA system provides is to perform diagnostics on anomalous conditions to determine if these conditions are being caused by errors in the advanced controller. If this is the case, the RTA system switches operation to a trusted reversionary controller. Initial studies were performed to demonstrate this benefit. The other focus was to improve the performance of the engine protection logic, which was deemed too conservative and reduced engine performance during transient operations. It was determined that the conservative response was due to poor tuning of one of the controller channels within the protection logic. An automatic tuning algorithm was implemented to optimize the protection logic control gains based on minimizing tracking error. Improved tracking responses were observed with no change to the existing protection logic control architecture.

Nomenclature

C-MAPSS40k	Commercial Modular Aero Propulsion System Simulation 40k		
CA	conditionally active	DTA	design time assurance
DTA'd	design time assured	EPR	engine pressure ratio
HPC	high pressure compressor	IFT	iterative feedback tuning
MBEC	model based engine control	OTKF	optimal tuner Kalman filter
PI	proportional-integral	PID	proportional-integral-derivative
PLA	power lever angle	PR	pressure ratio
RTA	runtime assurance	RTA'd	runtime assured
SM	surge margin	V&V	verification and validation
Wf	fuel flow		

I. Introduction

There is now wide interest in run time assurance (RTA) approaches for a number of safety or operational critical systems. In recent years a number of national and international forums and conferences have emerged that are dedicated to software assurance and run time assurance topics and these meetings seem to be growing in size and importance, especially in the computer science and systems engineering fields. Application specific industries are now recognizing the utility and benefits of RTA to realize inclusion of advanced and intelligent systems that would otherwise be too complex to certify for operational use. In general terms, an RTA system provides protection by continually monitoring the overall system's state (or certain critical parameters) to determine if the system is operating correctly (in terms of safety and/or performance). If the monitoring function determines that unsafe or off-nominal

* This paper has been approved for public release by NASA, 12/11/2017.
† Principal Research Scientist, 1410 Sachem Place, Ste. 202, Associate Fellow, AIAA.
‡ Research Scientist, 1410 Sachem Place, Ste. 202.
§ Research Engineer, Intelligent Control and Autonomy Branch, Senior Member, AIAA.
** Aerospace Engineer, Intelligent Control and Autonomy Branch, Senior Member, AIAA.

conditions are ensuing due to undiscovered errors in the untrusted advanced system, then the RTA system activates one or more recovery actions to mitigate the adverse conditions and return operation to a safe/correct state.

Because of the need to increase the operating performance of turbofan engines for new and future aerospace applications, there is currently wide interest in using more advanced control algorithms to achieve these new capabilities. The current investigation is motivated by a number of recent advanced control approaches in the propulsion community to provide better performance over the life cycle of the engine [1,2]. However, it is recognized that certification of such advanced controllers may be difficult because of their inherent complexity with intelligent and possibly nondeterministic elements. Exhaustive software testing used in current verification and validation (V&V) methods cannot cover the nearly infinite states that can be reached by such advanced algorithms. RTA systems can hold the promise of providing safe engine operations under these types of advanced controllers and NASA Glenn Research Center is interested in the application of RTA methods to expedite the certification process.

This paper presents an initial investigation of applying RTA protection to a particular turbofan engine model with an advanced control system. General background on RTA systems is presented next in Section II, followed by discussion of applying RTA to the turbofan engine system in Section III. Case studies demonstrating the benefits of the RTA system in determining faults in the advanced engine controller and subsequent mitigations are then presented in Section IV. Next, work on improving the design of safety critical functions within existing engine protection logic is presented in Section V. This logic is considered to be part of the overall RTA design. Last, conclusions are drawn from the study in Section VI.

II. General Background of RTA Protected Systems

A. Fundamental Concepts and Definitions

Key definitions and fundamental concepts of RTA systems are presented in this subsection. Consider the following definitions:

Trusted software: is a set of software that can be fully V&V'd to its defined required certification level [3,4,5] at design time (before live operation). That is, there is an acceptable level of confidence that the software will operate correctly to the required certification level of the system within which it operates. The required certification level depends on the defined criticality of the software, which is determined by the level of hazardous effects that errors in the software can have on the plant (Level A – catastrophic, Level B,C – major/hazardous, etc., see [3,4]).

Untrusted software: is a set of software that cannot be fully V&V'd to its defined required certification level (due to its complexity, nondeterminism, etc.). That is, there is not an acceptable level of confidence that the software will always operate correctly for any state that can be encountered during live operation of the plant. Untrusted software is typical during developmental testing stages, in which full V&V analysis/testing would be schedule and cost prohibitive at that point in the design cycle.

Design time assurance (DTA): is the process of performing V&V analysis and testing of software offline, at design time (again, before live operation) to the level required for certification of the plant or system the software is intended to operate on. A design time assured (DTA'd) system or subsystem is one in which all its software components are trusted.

Runtime assurance (RTA): is the process of monitoring a system containing untrusted software during runtime or live operation of the plant to determine if the untrusted software is operating correctly. If it is determined that the untrusted software is not operating correctly or it is detected that anomalous or unsafe behavior is ensuing, then control is switched to trusted reversionary software. That is, the RTA system activates some type of recovery action to ensure continued safe or correct operations. Note that the term "correct" implies that the software is performing as it is intended to, defined in some manner, such as by minimum performance requirements. The presumption here is that the reversionary software has equivalent basic functionality as the untrusted software, but without all of the advanced features. The reversionary software would be able to continue safe operation of the plant or system, but at a reduced level of performance, capability, or functionality. In general, the reversionary software provides a "recover and return to base" capability. In the literature, the reversionary system is often referred to as the "backup" system [6,7,8,9,10]. In experimental systems, such as with flight test aircraft, the previously designed and certified production controller can serve as the reversionary system.

The RTA monitor, decision logic, control mode switching code and the reversionary system are all, by definition, DTA'd subsystems since they must be trusted to provide protection from unknown errors in the untrusted advanced

system. A runtime assured (RTA'd) system or subsystem is one which contains untrusted software that is monitored/protected by an RTA system.

It should be noted that RTA protection covers both advanced system design flaws (validation) as well as software coding/implementation flaws (verification). If either type of flaw or error causes the overall system to enter into an anomalous or unsafe state, then the RTA monitoring should detect this and activate the reversionary processes.

There are two fundamental types of RTA systems:

1. Safety-critical RTA systems: monitor system functions whose failure could lead to physical damage or result in catastrophic loss of the system. This class of RTA systems is currently employed on turbofan engines and will be discussed later in the paper.
2. Performance/mission-critical RTA systems: monitor system functions whose failure could result in performance degradation or reduced mission capabilities. In such a case, the RTA system is more akin to a software integrity monitor, performing statistical diagnostics and other analyses to determine if the advanced system is operating as it should, delivering its stated performance. Case studies of a performance critical RTA system are also presented later in this paper.

Last, if the probability of hardware failures is considered sufficiently high enough to warrant inclusion of a hardware health monitor, then this hardware monitor should operate in an integrated fashion with the RTA system. Hardware faults can give rise to similar anomalous conditions as software errors in the advanced system being monitored by RTA. Therefore, if hardware faults are detected, then this information needs to be passed to the RTA decision logic. The RTA system must then make the determination whether to continue with the advanced system or revert to the reversionary system. In some cases it is better to continue operation with the advanced system, especially if it has hardware fault adaptation capabilities. Note that a hardware fault detection algorithm was not included in this study, as this was beyond the scope of the current effort. For all the experiments presented, it was assumed that all hardware (actuators, sensors, turbomachinery, etc.) were working correctly.

B. The Runtime Assured System Framework

There are a number of variants to runtime monitoring or RTA frameworks. A general RTA'd system framework is presented in Figure 1 below, inspired by the original work by Shaw [8] referred to as the Simplex framework.

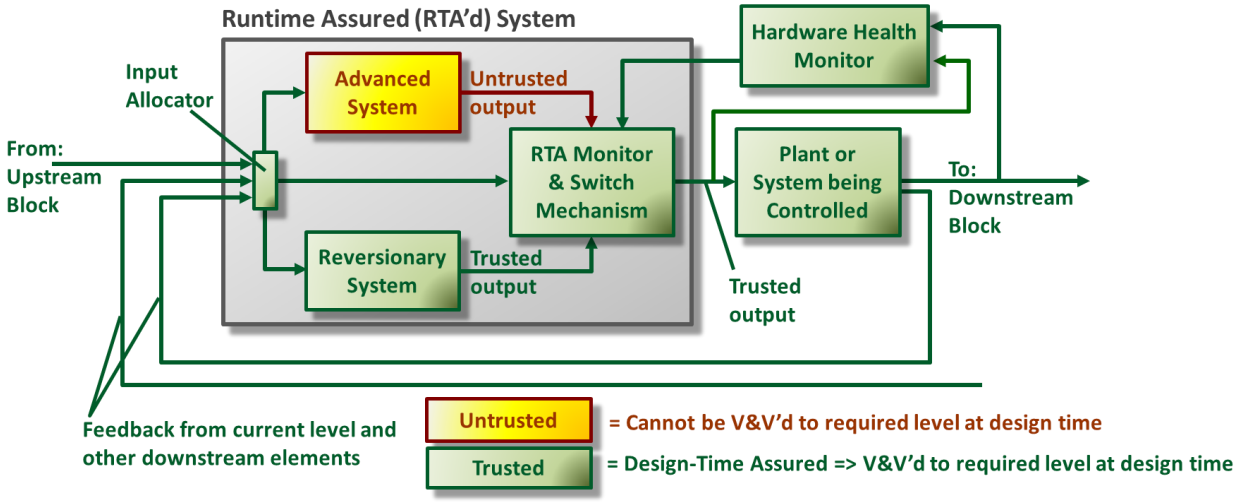

Figure 1. A General Form of a Runtime Assured System

In the figure, input to the RTA protected block can come from upstream sources and from feedback from one or more downstream sources. The advanced system and its output are untrusted. The other blocks in the figure, which include the input allocator, the reversionary system, the RTA monitor and switch mechanism block, the plant being controlled and the hardware health monitor are all DTA'd subsystems. All information flow into and out of the RTA's system is considered trusted even when the output of the advanced system is passed through to the plant. The claim here is that that output is trusted because it has been checked by the trusted RTA monitor. Ultimately, the goal is to

make a safety case argument that the RTA protected system is equivalent in terms of safety and correct operation to a system that is fully V&V'd at design time with accepted analysis/testing practices.

III. RTA Protection Applied to a Turbofan Engine

A. Introduction

This section presents an initial investigation of RTA protection applied to a particular high fidelity turbofan engine model developed by NASA and its contractors known as the Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k) [11,12]. This model was used to develop an advanced model-based engine control (MBEC) system for experimental study [13]. Since thrust cannot be directly measured, traditional engine control indirectly controls thrust through related parameters, such as fan speed or engine pressure ratio (EPR), which is the low pressure turbine discharge pressure P_{50} divided by the inlet pressure, P_2. However, NASA has developed a new approach whereby thrust is estimated using an Optimal Tuner Kalman Filter (OTKF) and this estimate is fed back to the controller, thereby directly controlling thrust. The OTKF can self-tune to account for engine performance variations over its life cycle. Feedback of the thrust estimate allows for tighter control of thrust over the life of the engine, which is not achievable using traditional control methods. The C-MAPSS40k simulation platform used in this study offers an EPR controller, a fan speed controller, and the MBEC OTKF controller as options for experimentation and development by researchers using this simulation platform. Herein, the MBEC OTKF and its associated feedback of estimated thrust defines the advanced controller. The EPR controller offered as part of the simulation package is used for the reversionary controller.

Sensor information used by the controllers include fan and core speed, and temperatures and pressures measured throughout the engine. The single control considered in this study is fuel flow rate. The pilot's throttle input is defined by the power lever angle (PLA).

B. Runtime Assurance Integrated with Engine Protection Logic

Turbofan engines continually operate near their physical limits (in terms of pressures, temperatures, rotation speeds, etc.). Because of this, the turbofan engine control community has long employed engine protection logic that keeps the engine from exceeding such limits, especially during transient operation from one equilibrium point to the next. The protection logic often engages to regulate the fuel flow rate during rapid changes in pilot PLA command input, which can cause transients in critical engine parameters related to structural health boundaries and flow dynamics through the engine. The engine protection function continually monitors the values of these parameters to determine if limit violations are ensuing. If so, the protection logic regulates the fuel flow rate to ensure all allowable bounds defined for the set of critical parameters are not exceeded. Engine protection will also engage during near-idle conditions to ensure combustor blow-out does not occur. Generally, the engine protection engagement is temporary and once the off-nominal conditions are mitigated or the engine settles out at a new equilibrium condition, then the fuel flow rate regulation becomes inactive. In summary, engine protection logic ensures safe operating conditions by preventing the engine from operating in regions where either engine damage or unstable combustion and/or airflow through the engine can occur.

It is key to note here that the engine protection system will often engage as a part of normal operations, regardless of whether an untrusted advanced controller is used, or a trusted reversionary controller is used. Therefore, the current engine protection logic is defined here as the safety critical element in the overall RTA system, as this provides safety to the engine system. No other safety critical RTA function is needed. The engine protection system should operate its monitoring function at all times and take over fuel flow control when necessary, whether control is in advanced or reversionary mode. Transient response performance is a critical factor to the Federal Aviation Administration (FAA) process for certifying aircraft engines. Therefore, design of the engine protection logic is key to certifying the overall engine system and improvements in the design of the engine protection logic is presented in Section V.

Although the engine protection system provides the safety-critical RTA element, additional RTA functionality is required to determine if the advanced controller is operating incorrectly. The other main current effort is development of RTA diagnostic methods that can make this determination. The proposed diagnostic methods constitute the performance-critical part of the RTA system. Here, the RTA monitor continually executes diagnostic algorithms or other tests that can determine if the advanced controller is delivering its required performance or is instead operating in a degraded mode (irrespective of whether the engine protection logic is engaged).

Development of RTA decision protocols is under current study by the authors. It is proposed that performance maps or other criteria that define the expected performance of the trusted reversionary controller be constructed offline

and then efficiently queried online to determine if the advanced controller is at least meeting the expected required performance of the reversionary controller. If not, then this would trigger a control mode switch to the reversionary controller. Such performance criteria could include response rise or settling times, or steady state EPR level versus PLA input, for example. Case studies that compare EPR performance of the reversionary controller to the EPR performance delivered by the advanced controller are presented in the next section. Other criteria may involve sensed temperatures or pressures at various engine stations. These mappings would be a function of flight condition and would be interrogated online based on current sensor inputs.

Other logic may analyze the requested fuel flow from the advanced controller if it triggers activation of the engine protection logic. Parameters such as the rate or magnitude of the fuel flow command when the protection logic is activated may indicate whether an error is present in the advanced controller. Even though the protection logic prevents any adverse effects from occurring, the software integrity of the advanced controller would be in question, triggering a switch to the trusted reversionary controller. A case study of this scenario is presented in the next section.

Fault detection and statistical change detection technologies may prove beneficial to determining more long term, subtle problems, such as in efficient fuel economy or higher than expected operating temperatures causing premature wear and tear to engine components. Such algorithms would run continuously and draw statistical information from a running window of past engine state data and may be periodically analyzed offline during regular maintenance schedules.

C. The Runtime Assured System Framework for the Turbofan Engine

The framework for this architecture is presented in Figure 2. The input to the system is shown to be the pilot's PLA command. The advanced and reversionary controllers are indicated to be the MBEC OTKF and EPR controllers respectively. The RTA monitoring and switch block is expanded to show that it involves both the engine protection logic and associated fuel flow regulation for the safety-critical function as well as the RTA diagnostic function and switch mechanism that determines which control command is delivered to the engine. The plant therefore receives one of three possible control commands, depending on the control mode determined by the RTA diagnostics:

1. Advanced control mode – outputs of the MBEC OTKF controller are passed through to the plant.
2. Reversionary control mode – outputs of the EPR controller are passed through to the plant.
3. Fuel flow rate regulation mode – fuel flow rate is adjusted to ensure no critical parameter violates its upper or lower bound.

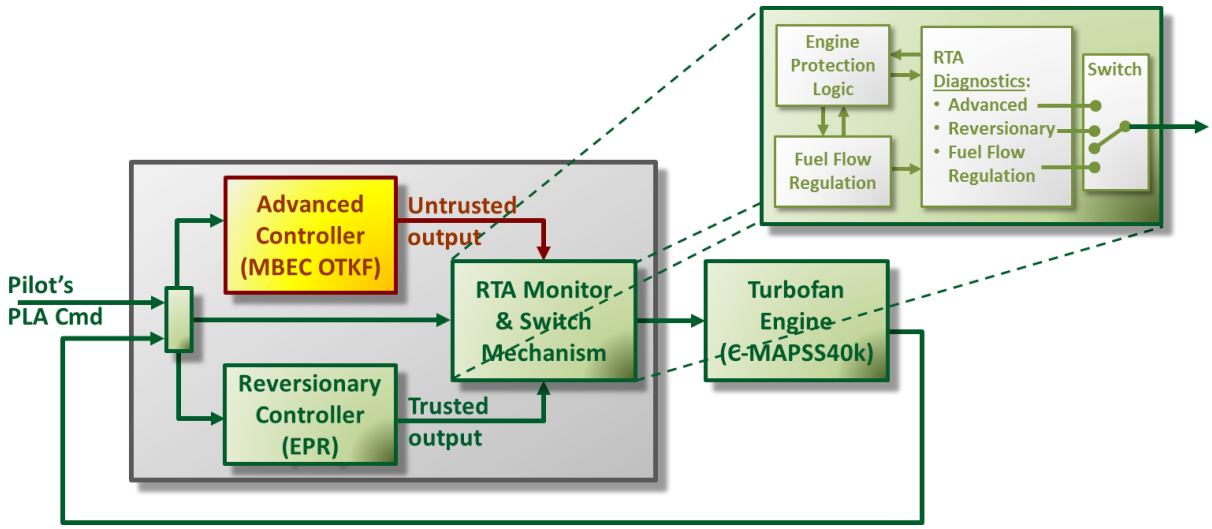

Figure 2. Runtime Assured Engine Protection

Note that the engine protection logic will be active at all times, continually determining if any critical parameter violation is ensuing. The RTA diagnostic algorithm will also be continually active, collecting passed and current output information from certain defined channels and continuously running checks on that data to determine if the advanced controller is faulted in any way. The reversionary controller should also be running at all times in "shadow mode," even if the advanced controller has primary control of the engine system. The objective of shadow mode

operation of the reversionary controller is to minimize transients or other control mode switching issues, such as integrator wind-up, etc.

The next section provides case studies of the RTA system switching control from the advanced controller to the reversionary controller, demonstrating the performance-critical RTA function. Following this, Section V focuses on engine protection logic design improvements, which again defines the safety-critical RTA function. Engine protection logic can be overly conservative in keeping critical parameter values far from their respective limits, causing poor transient response characteristics. Improving on the engine protection performance is an ongoing research focus at NASA and a proposed design procedure enhancement is presented in Section V.

IV. RTA Case Studies

This section presents experimental case studies resulting from initial investigations of a more general RTA decision function design methodology. To narrow the studies, the authors reviewed literature pertaining to incidents resulting from engine malfunctions. In Sallee et al., [14], for example, engine malfunction incidents, their causes, and flight conditions are tabulated and categorized for various types of aircraft engines. From this article, it is evident that engine malfunction incidents are largely due to compressor surge at take-off flight conditions. Loss of power, inappropriate crew responses, and hardware faults are other leading causes. Therefore, case studies were narrowed to take-off flight conditions and to faults that would result in either compressor surge or reduction of power at takeoff. Criteria that could potentially be used to determine triggering a switch to the revisionary controller are also explored.

All results that follow were obtained from the C-MAPSS40k engine simulation. For each case study, the PLA was increased from a ground idle value (PLA = 40 deg.) to maximum take-off power value (PLA = 78 deg.), starting at 5 seconds and reaching the maximum value by 15 seconds. This is considered a typical, not overtaxing power command scenario.

Case 1. Nominal Baseline Results

Figure 3 shows baseline simulation results for this case for the reversionary EPR controller. The upper left hand plot shows the PLA time history, just described. The upper right hand plot shows the fuel flow (Wf) rate command history. Plotted as the blue line is the requested fuel flow (Wf Requested) from the EPR controller. Plotted as the orange line is the fuel flow delivered (Wf Delivered) to the engine. When these two lines differ, this indicates that the engine protection logic is activated. It can be seen that these two lines only slightly differ between 5 and 15 seconds where the PLA changes, indicating that this is a rather benign case.

The high pressure compressor (HPC) surge margin (SM) is plotted in the lower left plot in Figure 3. Also plotted is the surge margin limit (SM Limit), defined for this case study as a constant 11%. There are a number of different models for calculating surge margin. The C-MAPSS40k developers have adopted the following, which is the Society of Automotive Engineers formulation:

$$SM = 100\left(\frac{PR_{\text{surge}} - PR_{\text{operating point}}}{PR_{\text{operating point}}}\right) \tag{1}$$

The pressure ratios (PR) in this equation are specified at a constant corrected mass flow rate. Note that the actual remaining surge margin is usually reduced due to a number of real world conditions, such as compressor blade heat transfer effects, variations in compressor tip clearance, debits due to normal engine deterioration, and debits due to engine-to-engine variations. These effects are not included in these studies, although they are modeled in the C-MAPSS40k simulation and can be considered in higher fidelity studies.

Last, the lower right hand plot in Figure 3 presents the net engine thrust in pounds. It can be seen that there is a significant delay in achieving the requested power, with a 90% rise time of the final thrust value not being achieved until approximately 17 seconds (3 to 4 seconds after the 90% rise time of the max PLA value).

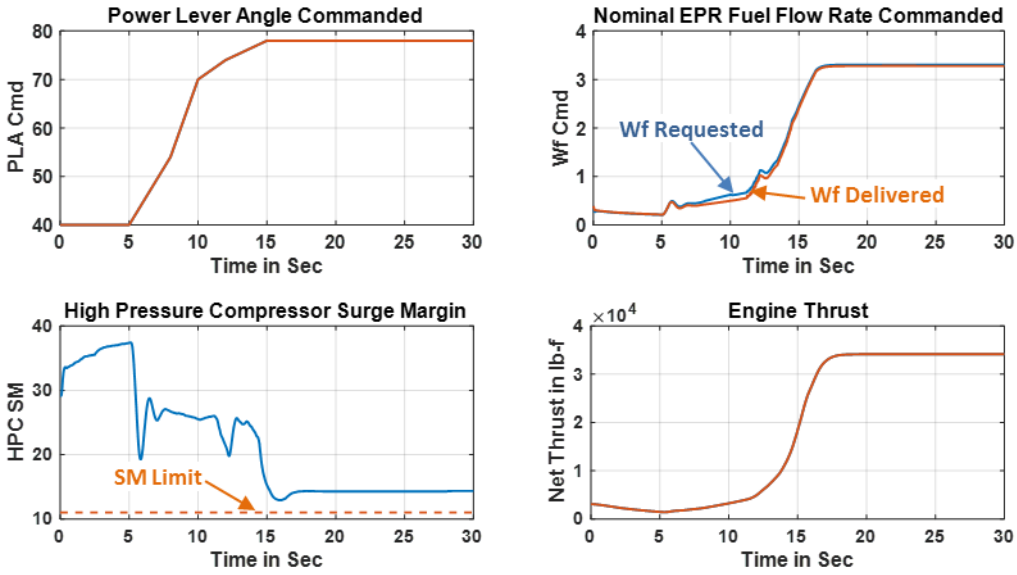

Figure 3. EPR Controller Simulation Results for Take-Off Power Command

Figure 4 shows the analogous results for the nominal (no errors) advanced MBEC OTKF controller. The PLA history is the same as shown in Figure 3. The requested fuel flow, however, is substantially different, with a much more aggressive rise. The requested fuel flow was limited by the protection logic during the PLA change at approximately 6 to 7 seconds due to the acceleration schedule limit in the engine protection logic. This limit is discussed more in Section V. Note that the requested fuel flow was also limited to a maximum value in steady state, due to reaching one or more of the structural upper-bound limits in the engine protection logic. These limits are also discussed more in Section V. As expected, the surge margin is generally lower than that of the EPR controller, and even exhibited a small, short limit violation at around 6 seconds near where the maximum core acceleration occurs. This demonstrates the less conservative nature of this engine controller over that of the EPR controller. The engine thrust response performance is seen to be improved over that obtained by the EPR controller, with a 90% rise time achieved at approximately 10 seconds. This demonstrates a substantial performance improvement over the EPR controller.

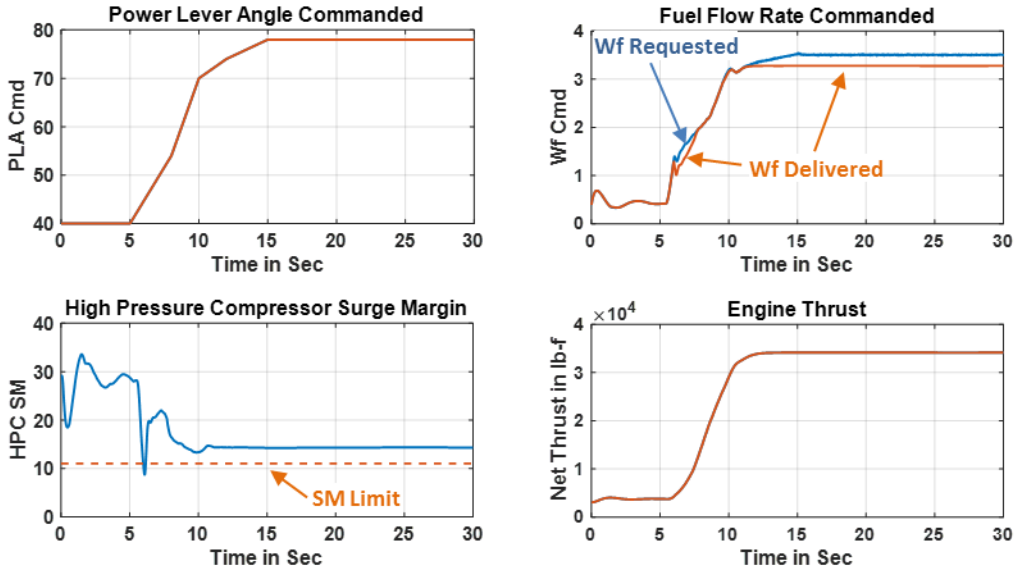

Figure 4. Nominal MBEC OTKF Controller Simulation Results for Take-Off Power Command

Case 2. Seeded Fault during PLA Transient - Increase in Requested Fuel Flow

For the first error case, a number of large step increases were introduced to the fuel flow command generated by the advanced MBEC controller during (and after) the transient region where the PLA command is increased from idle to maximum takeoff value. The results of this case are shown in Figure 5. Here, the large step commands in the requested fuel flow from the MBEC controller are shown in the fuel flow history plot. Without engine protection, these faults would have caused immediate HPC surge, shutting down engine operations. However, it can be seen that the engine protection logic very effectively arrests these faulted commands and the resulting engine operations are largely the same as that shown in the nominal case with no errors in the advanced controller. If the RTA diagnostics do not analyze the fuel flow command history from the MBEC controller, then this case would go unnoticed. This points to the fact that even though the engine protection provides safe, stable combustion, the fuel flow commands generated by the advanced controller should still be analyzed in some manner by the RTA monitor function.

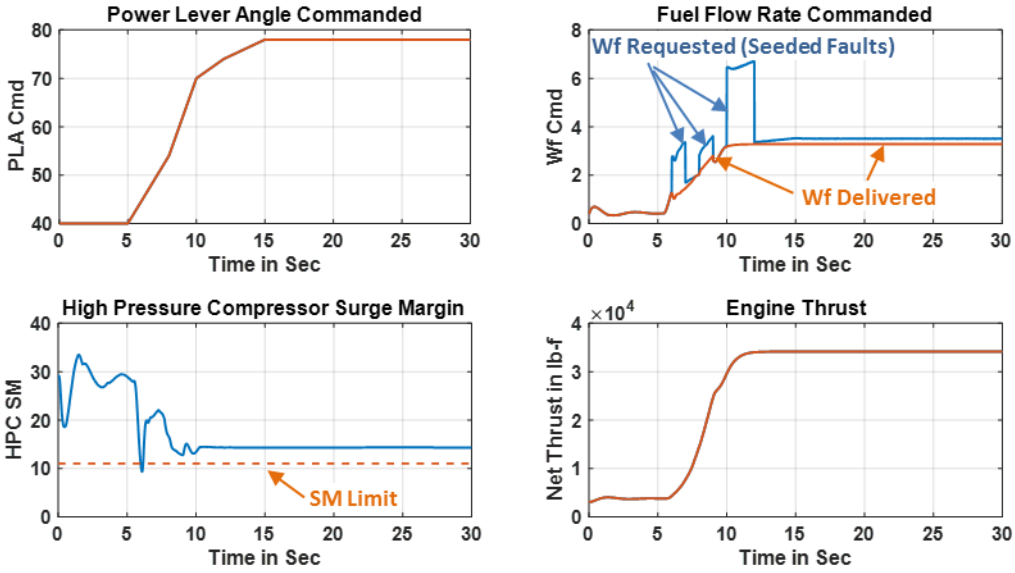

Figure 5. Error in MBEC OTKF Controller – Fuel Flow Increases during PLA Transient

Case 3. Seeded Fault during Steady State - Linear Reduction in Requested Fuel Flow

Since the engine protection logic will arrest large increases in fuel flow command, an error that would not activate the engine protection logic was sought. Here, a simulated fault causing a linear reduction in the fuel flow command generated by the MBEC controller was introduced during the steady state conditions after the peak engine thrust was achieved. This case is shown in Figure 6. The PLA history plot is no longer shown here, as it is the same as shown in the last three figures. The upper left plot shows the fuel flow command history and it can be seen that there is a linear reduction in commanded fuel flow starting at 15 seconds. The key here is that this reduced value in fuel flow causes the engine protection to shut off (the commanded fuel flow neither triggers an upper bound limit nor a lower bound limit of any of the critical parameters). Therefore, this error is allowed to continue. The upper right plot shows the surge margin history and it can be seen that, as expected, the surge margin increases during the phase when the fuel flow is reduced.

The lower left plot shows an example of an RTA check that might prove useful in this case. Here, the EPR history is plotted. The suggested RTA trigger or switching condition is also indicated in this plot. If the EPR achieved by the advanced MBEC controller falls below 90% of the expected steady state EPR achieved by the EPR controller (at this PLA value and flight condition), then the RTA system switches operation to the reversionary EPR controller. This occurs at approximately 24 seconds. A value of 90% is used to account for model uncertainty, environmental variations, sensor noise, etc. That is, to prevent false alarm switches, the RTA switching condition should not be overly sensitive and switch at say, 98% or 99% of the EPR controller value. The actual switching condition is design dependent. The recovery in pressure ratio and net thrust after the switch occurs at 24 seconds is evident in the bottom two plots in Figure 6.

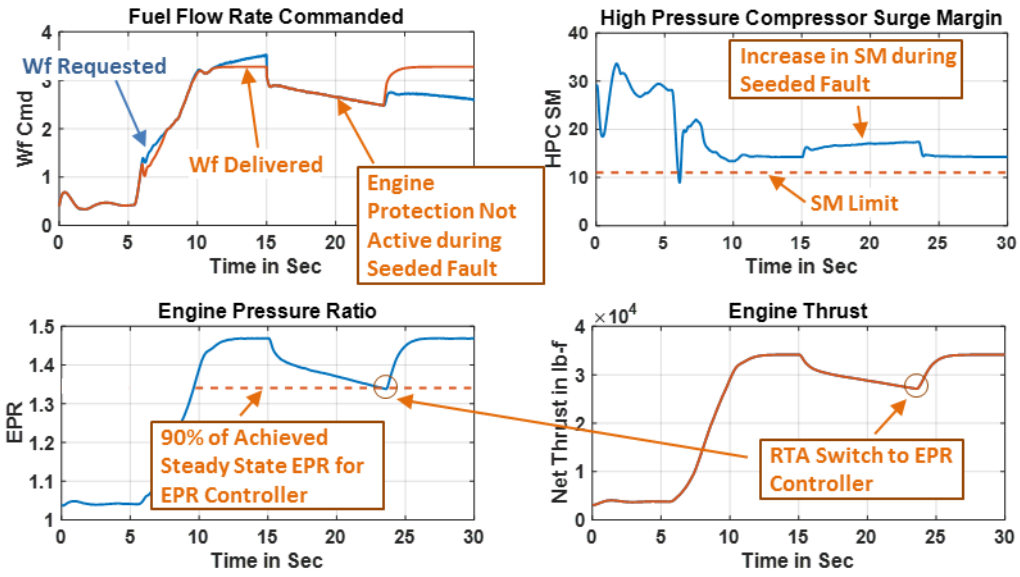

Figure 6. Error in MBEC OTKF Controller – Fuel Flow Decreases Linearly During Steady State

Case 4. Seeded Fault during PLA Transient – Sluggish Response in MBEC Controller

Perhaps a more troublesome error that could occur that also does not trigger activation of the engine protection logic would be if the advanced MBEC controller produced a sluggish response at takeoff. This error was introduced by inserting a second order low pass filter at the output of the MBEC controller. This causes rapid increases in commanded fuel flow to be attenuated, causing the sluggish thrust output performance. This filter also attenuates the steady state magnitude of the commanded fuel flow, causing further performance issues. This case is shown in Figure 7. It can be seen that after 5 seconds, the fuel flow command from the MBEC controller is never adjusted by the engine protection logic. The surge margin is also seen to be much larger than the nominal case, which indicates a lower thrust output. The engine thrust response can be seen to be quite benign compared to the rapid response seen in the nominal case.

The lower left plot in this case also shows an example of an RTA check that might prove useful. Again, the EPR history is plotted. The suggested RTA trigger or switching condition indicated in this plot is when the EPR achieved by the MBEC controller falls below the EPR achieved by the EPR controller during the transient. The slopes of the EPR responses may also be analyzed, indicating a much faster rise time achieved by the EPR controller over that achieved by the MBEC controller. Clearly a loss of performance is indicated here. This switch would be triggered at 15 seconds into the run.

Figure 8 shows the corrected responses achieved by the RTA system switching to the reversionary EPR controller at the trigger time of 15 seconds. The fuel flow command immediately increases to its proper value, resulting in immediate recovery of the EPR and net engine thrust responses.

A key observation seen in both Case 3 and Case 4 is that when the RTA system switches to the reversionary controller, the responses show a very smooth transition, absent of any transients, overshoots, control bumps or other control mode switching issues. It is believed that this is the case because the reversionary EPR controller is being run in "shadow mode," generating control solutions at each step in response to the current state generated by the active advanced MBEC controller. Therefore, when it is commanded to take over operations, its initial conditions are aligned with the current engine and controller states.

In summary, these case studies show the benefits of RTA monitoring and checking of expected engine performance. Erroneous engine responses can occur even when engine protection logic is in place to arrest unsafe conditions. Here, such errors in the advanced controller can potentially cause drastic reduction in engine performance which, even if the engine combustion dynamics remain stable, can cause dangerous conditions to the flight vehicle itself. This is because if the pilot is expecting a certain level of thrust response performance and the error in the advanced controller prevents the engine from achieving its required or expected performance, then this can lead to a number of safety critical scenarios depending on how the pilot responds to the faulted engine.

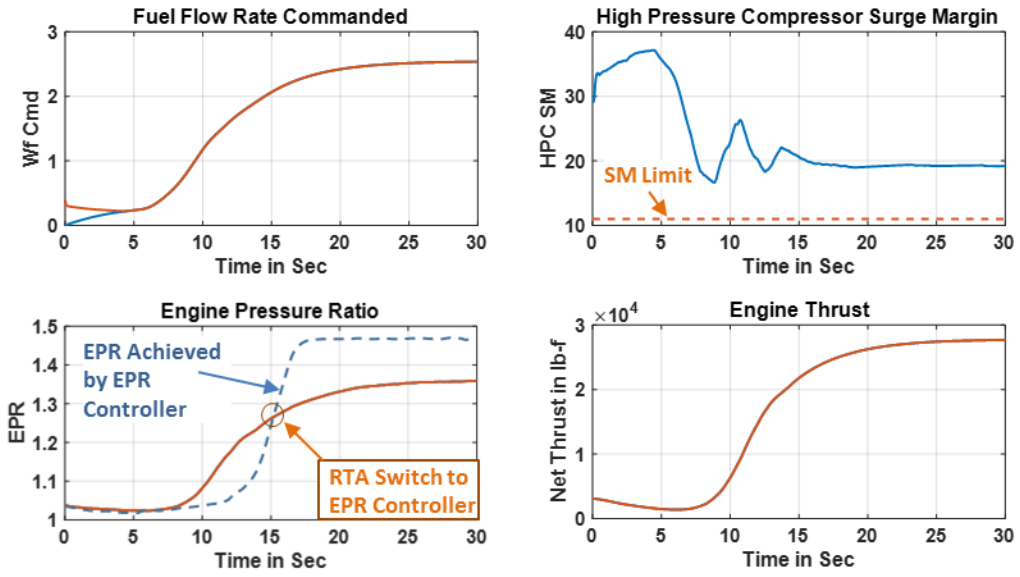

Figure 7. Error in MBEC OTKF Controller – Sluggish Response during Takeoff

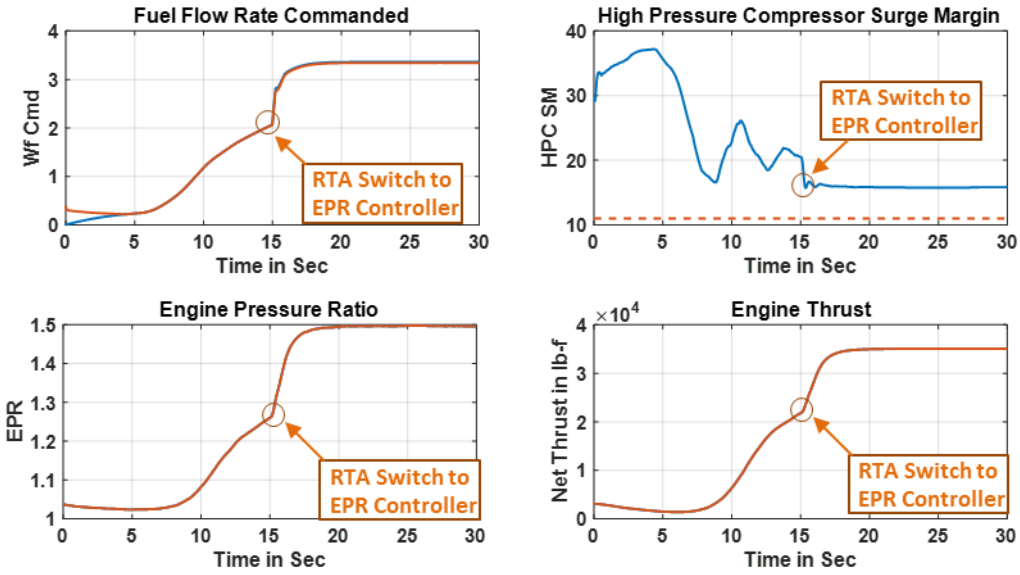

Figure 8. Error in MBEC OTKF Controller – Sluggish Response during Takeoff Corrected by RTA

V. Improvements in Engine Protection Logic Design

This section presents a novel design approach for improving the tracking performance of the engine protection logic. To provide more background, the current engine protection logic is first discussed in more detail than in Section III. Next, an overview of the design challenge is presented, followed by background on the new design approach known as Iterative Feedback Tuning (IFT). Experimental results of using the IFT design method are also presented in this section, showing the benefits of this approach.

A. Engine Protection Logic Background

The Min-Max protection logic, depicted in Figure 9, is a widely accepted feature of engine control for commercial systems. Again, in the C-MAPSS40k simulation, the user can choose one of three controller options –EPR, fan speed, or the advanced MBEC OTKF controller. Depending on the chosen controller, the pilot's "throttle input," or PLA command, is converted to either an EPR, fan speed, or estimated thrust command, which is then mapped to a fuel flow rate command to track the chosen feedback signal. The protection logic uses feedback of critical sensed parameters to ensure that the regulated fuel flow rate command does not cause these critical parameters to exceed specified limits.

The parameters that are limited are listed in Table 1. The fan speed, core speed, burner pressure and exhaust temperature are all upper-bound limited for structural considerations. Exceedances of these limits could result in mechanical or structural component failures. These are generally considered to be steady-state limits. The core acceleration schedule is upper-bound limited for operational considerations. This parameter indicates the remaining surge margin in the HPC. An exceedance of this limit can cause compressor stall during rapid changes in commanded thrust. This is generally considered a transient limit that is active during rapid changes in thrust but is not a consideration once the engine settles at the new commanded thrust. The two lower-bound limits are also for operational considerations and become active when there is not enough fuel flow requested by the controller to maintain stable combustion at the current operational state.

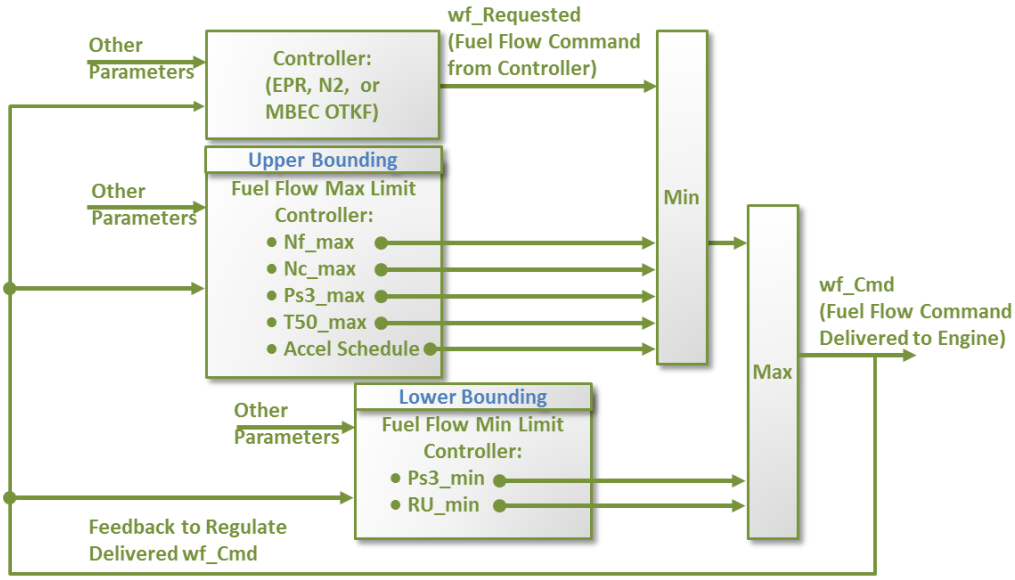

Figure 9. Engine Protection Logic and Fuel Flow Limit Control

Table 1. Critical Parameter Limits in the Engine Protection Logic

Parameter	Description	Bound Type	Engine Control Limit Type
N_f	Fan speed	Max	Structural
N_c	Core speed	Max	Structural
P_{S3}	Burner inlet static pressure	Max	Structural
T_{50}	Exhaust temperature	Max	Structural
$\dot{N}_c / N_c$	Acceleration schedule	Max	Operational – compressor surge
P_{S3}	Burner inlet static pressure	Min	Operational - combustor lean-blow out
w_f / P_{s3}	Ratio Units, RU	Min	Operational - flight idle operation

The upper bounding, lower bounding and min/max blocks shown in Figure 9 are not simply attenuation functions but include control laws, anti-windup logic, lookup tables and other functions – all designed so that, when active, the protection logic tracks the active limit as close as possible for performance considerations. The protection logic uses integral control to calculate the fuel flow rate required so that none of the parameter limits are violated. The closer a parameter comes to its limit, the lower the fuel flow rate allowed for that parameter.

The logic flow displayed in Figure 9 is as follows. First, a distinct fuel flow rate is calculated for each of the parameters listed in the upper bounding block that ensures each respective upper bound is not violated. These fuel flow rate values are then compared against the requested fuel flow rate from the controller. The minimum of all of these values is then passed to the next step in the protection logic. That is, the fuel flow rate value generated by the 'Min' block is either: (1) the requested fuel flow from the active controller if it is less than any of the maximum allowable values calculated in the upper bounding block, or (2) the minimum fuel flow value required so that no upper-

bound limit is exceeded. This fuel flow rate value is then delivered to the 'Max' block and compared to the lower-bound limits generated by the lower bounding block. If it is greater than the lower bounds, then this value is delivered to the engine as the fuel flow command, (wf_Cmd). Otherwise, the maximum fuel flow required so that no lower-bound limit is violated is delivered to the engine. Further details on the engine protection logic design can be found in [11,12].

B. Design Problem Overview

It is well understood that the engine protection logic results in a very conservative design, which can limit engine performance, especially through active transient operations. A preliminary approach was developed by May, et al., [15], in which the engine protection is made conditionally active (CA) only when operating near a parameter limit to help reduce this conservatism. Although increased performance can be obtained by this method, it was noted that the design required significant tuning and the CA limiter bounds were empirically determined in an ad hoc fashion. It was noted more "research needs to be done to develop a more rigorous analytical approach to provide guidelines for the selection of these bounds."

Based on NASA's desire to further mature the CA limit regulators and improve the engine protection logic by reducing its conservatism, the current CA design presented in May, et al. [15] was investigated. However, rather than maturing or augmenting the CA approach, the original protection logic design (i.e., without CA limiting) was assessed to determine if it could be improved. Detailed studies of the acceleration limiter revealed that poor tuning of the protection logic controller was the source of the conservative response in this channel, limiting the engine performance. However, PI controllers can be difficult to tune for nonlinear systems, specifically when undergoing large transients which exceed any linear operating range. Furthermore small changes in the gains can have substantial impact on the performance.

Therefore, instead of manually attempting to tune the proportional-integral (PI) feedback gains in the acceleration tracking controller, the IFT design technique was used, which is an innovative method of automatically tuning controllers assuming no knowledge of the underlying model [16,17]. The IFT method is a theoretically rigorous, robust algorithm capable of optimizing control gains for arbitrary control architectures without requiring an explicit equation for the plant. While the underlying theory and convergence of the algorithm is proved for linear systems, it can be applied to nonlinear systems as well. However the strength of the convergence proofs no longer applies and modifications to the implementation may, in general, be required to achieve desired improvements.

The IFT algorithm was implemented for the acceleration limiter logic and executed in a loop spanning a range of altitude and Mach conditions. The subsequent set of optimal PI gains resulted in reduced conservatism and greatly improved engine response during transient responses over a wide range of altitude and Mach values. It is proposed that these optimized gains replace the original fixed PI gain values in the engine initialization scripts. The entire optimization process is automated and no operator tuning is required to generate the new set of gains. This approach was also investigated for the other channels shown in the upper and lower bounding blocks in Figure 9. However, the approach was found to only improve the design in the acceleration limiter, as this was the most problematic channel.

C. Derivation of the IFT Algorithm

Optimized tracking control objectives can be quantified by a generic cost function parameterized by tracking error, regardless of the specific control architecture. General optimization solutions require a mathematical model of the plant and a specified control structure. However, the optimization process itself often determines the structure of the controller. If the plant is too complex to be modeled mathematically, or the plant is unknown, or there is a prescribed control structure, then optimization is typically achieved by iterative gradient-based minimization. For the acceleration tracking problem it is desired to maintain the existing PI control architecture and obtain the best possible tracking performance. Iterative optimization procedures are often inhibited by either complicated or unknown gradients of the controlled plant with respect to the optimization parameters (i.e., the control gains). It was shown by Hjalmarsson, et al. in [16] that the gradient can be computed directly from the plant input-output signals using specially constructed closed-loop experiments with the controller operating in-the-loop. Once the gradient has been experimentally determined, the tracking cost function can be minimized using standard Gauss-Newton iterations. For a single-degree-of-freedom controller, such as a PI controller, only two closed-loop experiments are required at each iteration step. The first experiment consists of collecting data under normal operating conditions with the controller operating in-the-loop with the plant. The second experiment uses a specially designed reference input to the controller based on outputs from the first experiment and inspires the name iterative feedback tuning. The scheme does not require model identification and can be applied to process systems running online to improve performance.

The theory of the IFT method presented here was initially presented by Hjalmarsson, et al. in [17]. The original presentation treats a more general two-degree-of-freedom controller with disturbances. The presentation here is simplified to consider single-degree-of-freedom controllers that are a function of combined tracking error, like a proportional-integral-derivative (PID) controller and also ignores disturbances.

Assume the following unknown system and single-degree-of-freedom controller:

$$y_t = G_0 u_t \tag{2}$$

$$u_t = C(\rho)(r_t - y_t) \tag{3}$$

It is assumed that the controller, C, has a defined structure and is parameterized by a set of tuning parameters, ρ. For example, C could represent a PID controller, parameterized by the control gains, ρ. It is apparent that for the closed-loop system, both the output and input are functions of the target control parameters, ρ. The closed-loop tracking error can be defined as

$$\tilde{y}(\rho) = y(\rho) - y_d = \left(\frac{CG_0}{1+CG_0} r - y_d \right) = \left(T_0 - T_d \right) r \tag{4}$$

where,

$$y_d \triangleq T_d r, \quad T_0 \triangleq \frac{CG_0}{1+CG_0} \tag{5}$$

The desired output response y_d is defined by some filter applied to the reference input and the closed-loop transfer function is defined as T_0. In order to minimize the tracking error, an optimization cost function can be defined in terms of the summed square of the tracking error over some number of samples, *N*, of a simulation run and weighted by the squared control effort, u_t:

$$J(\rho) = \frac{1}{2N} \left[\sum_{t=1}^{N} \tilde{y}_t(\rho)^2 + \lambda \sum_{t=1}^{N} u_t(\rho)^2 \right] \Rightarrow \frac{1}{2N} \left[\sum_{t=1}^{N} \tilde{y}_t(\rho)^2 \right] \tag{6}$$

For this application, weighting on the control signal was not addressed for the controller optimization so the cost function is simplified as shown. The goal of the algorithm is to determine the set of controller parameters, ρ, which minimize the objective function, $J(\rho)$. The objective function is minimized (locally) when its gradient is zero. So the Gauss-Newton iteration procedure is to determine the gradient of the cost function

$$0 = \frac{\partial J}{\partial \rho}(\rho) = \frac{1}{N} \left[\sum_{t=1}^{N} \tilde{y}_t(\rho) \frac{\partial \tilde{y}_t}{\partial \rho}(\rho) \right] \tag{7}$$

and iterate the control parameters until they satisfy the criteria for the minimum of the function

$$\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \frac{\partial J}{\partial \rho}(\rho_i) \tag{8}$$

where the iteration step size, γ, for the i^{th} step, controls the convergence rate for the control parameters. The update direction at the i^{th} iteration step, R_i, is specified by the user. However, the update direction is typically selected as a Gauss-Newton approximation of the Hessian of the objective function, *J*, which is defined as

$$R_i = \frac{1}{N} \sum_{t=1}^{N} \left[\frac{\partial \tilde{y}_t}{\partial \rho}(\rho_i) \right] \left[\frac{\partial \tilde{y}_t}{\partial \rho}(\rho_i) \right]^T \tag{9}$$

Where the control effort is not included in the current definition of the cost function. Since it's assumed that the process cannot be modeled as a mathematical function, the gradients must be computed from the process (or simulation) outputs. Using the closed-loop transfer function, T_0 the gradient of the tracking error can be defined as

$$\frac{\partial \tilde{y}_t}{\partial \rho}(\rho) = \frac{\partial y_t}{\partial \rho}(\rho) = \left(\frac{G_0}{1 + C(\rho) G_0} - \frac{C(\rho) {G_0}^2}{\left(1 + C(\rho) G_0\right)^2} \right) \frac{\partial C}{\partial \rho}(\rho) r = \frac{1}{C(\rho)} \left(T_0 - {T_0}^2 \right) \frac{\partial C}{\partial \rho}(\rho) r \tag{10}$$

and,

$$T_0^{\,2} r = T_0 y, \qquad \frac{\partial \tilde{y}_t}{\partial \rho}(\rho) = \frac{1}{C(\rho)} \frac{\partial C}{\partial \rho}(\rho) T_0 (r - y) \tag{11}$$

Therefore the tracking error gradient can be computed by multiplying the reciprocal of the controller by the gradient of the control with respect to the tuning parameters by the closed-loop transfer function acting on the input signal (*r*-*y*). This immediately reveals the required process experiments to determine the solution gradient which is broken into two steps.

1. For the i^{th} iteration, a simulation is run for a fixed number of samples, N, with the candidate controller. The input signal is the reference input, *r*, and the output is the nominal closed-loop response y_1.
2. A second simulation is run for the same number of samples, *N*, with the same candidate controller. However, the input signal is now the reference input, *r*, *minus* the nominal closed-loop response, y_1, obtained from the first experiment. Therefore the input is $(r - y_1)$

The output of these two experiments explicitly define the tracking error gradient without requiring any knowledge of the specific plant:

$$\begin{aligned} y_1 &= T_o r \quad \Rightarrow \text{first experiment} \\ y_2 &= T_0 (r - y_1) \quad \Rightarrow \text{second experiment} \\ \frac{\partial \tilde{y}_t}{\partial \rho}(\rho) &= \frac{1}{C(\rho)} \frac{\partial C}{\partial \rho}(\rho) y_2 \quad \Rightarrow \text{gradient calculation} \end{aligned} \tag{12}$$

This treatment is greatly simplified due to the single-degree-of-freedom controller and the control input not being weighted in the cost function. From experiment #1 the tracking error is explicitly defined as

$$\tilde{y}_t(\rho) = y_1 - y_d = y_1 - r \tag{13}$$

where the desired response, y_d, is simply the reference input. Using Eqs. (12) and (13), the gradient of the cost function is now computed according to Eq. (7), the update direction according to Eq. (9), and the control parameters are iterated according to Eq. (8).

D. Implementation of the IFT Algorithm

Conservatism in the tracking response can be reduced by improving the control gains within the existing engine protection logic. The design goal is to maintain the existing tracking control architecture and optimize the gains. In order to apply the IFT algorithm to the engine protection logic, it has to be specified for a *discrete* PI control architecture. The discrete PI controller takes the form

$$C(\rho) = K_p + \frac{K_i T_s}{z - 1} = \frac{K_p z - K_p + K_i T_s}{z - 1} \tag{14}$$

where,

$$\rho \triangleq \left[K_p, K_i \right] \tag{15}$$

The simulation experiments are setup by modifying the C-MAPSS40k model to create a unique test platform. The full nonlinear simulation was modified to start the PLA step change at time 0 so that the acceleration transient would be applied during the initial portion of the simulation to simplify data collection. The simulation was also modified so that the acceleration limiter was always active. The baseline EPR controller and the other protection logic channels were bypassed. By forcing the engine to respond only to fuel flow commands only from the acceleration limiter, there was no min-max control signal switching which would corrupt the algorithm progression. As a desirable

by-product, this also caused the anti-windup logic for the acceleration limiter integral control to become disabled. The final test of the optimized control gains was performed on the unmodified simulation with both the baseline controller and full protection logic active. The acceleration controller was modified to include an auxiliary reference input to collect data for the gradient calculation. For experiment #1 the auxiliary reference input was set to zero and the total reference input to the controller was the nominal acceleration schedule, r. The output of experiment #1 was the nominal closed-loop output data, y_1, which is the sensed engine acceleration. For experiment #2 the same simulation model was used but the auxiliary reference input was set to the output data from experiment #1. Therefore the total reference input for experiment #2 was the nominal acceleration schedule, r, minus the nominal output data from experiment #1, y_1, the original closed-loop acceleration response. The output of experiment #2, y_2, is the closed-loop transfer function acting on the tracking error signal: $T_0(r - y_1)$, which is used to compute the tracking error gradient.

In order to compute the tracking error gradient, total objective function gradient, and search direction, a distinct 'gradient simulation' was created to implement the filter defined by Eq. (12), where Eq. (14) defines the controller. The tracking error in Eq. (13) was incorporated in the gradient simulation using the N-length input and output signals from the two experiments.

E. Results of the IFT Algorithm

While the internal optimization experiments were performed using the modified simulations, the following results correspond to the full, unmodified nonlinear C-MAPSS40k simulation executed at Sea-Level Static Conditions (Mach = 0, Alt = 0) with a step change in PLA from 44 to 80 at time zero. The nominal system response in Figure 10 shows poor closed-loop tracking of the acceleration limiter. Figure 11 shows the corresponding fuel flow rates comparing the EPR control effort with the acceleration limiter logic control effort. The perforated line indicates which of the two control signals is sent to the engine. The protection logic selects the smallest fuel flow magnitude so the acceleration limiter fuel flow rate is initially sent to the engine but at approximately 2.2 seconds, the EPR fuel flow rate falls below the acceleration limiter control effort and the output changes to the EPR control signal. This corresponds to the bounds in Figure 10defining the period of the transient response where the acceleration limiter is active. When the acceleration limiter logic goes inactive, acceleration tracking is not expected because the engine control signal is designated for another priority, in this case the baseline EPR tracking. Fuel flow rates are not shown for the remaining response iterations because the active limit bounds in the tracking response clearly indicate when the acceleration limit logic produces less fuel than the EPR control and is the active channel.

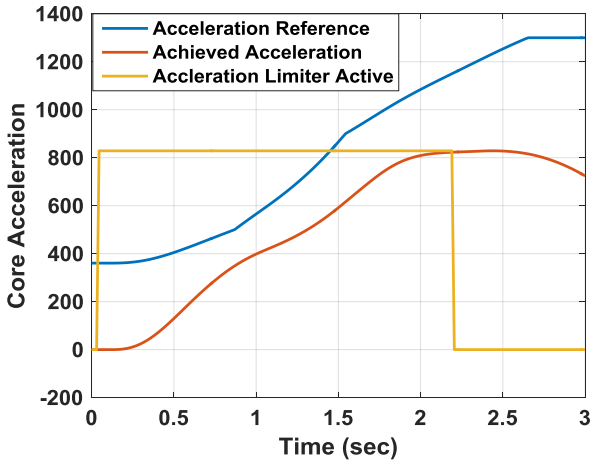

Figure 10. Nominal Acceleration Tracking

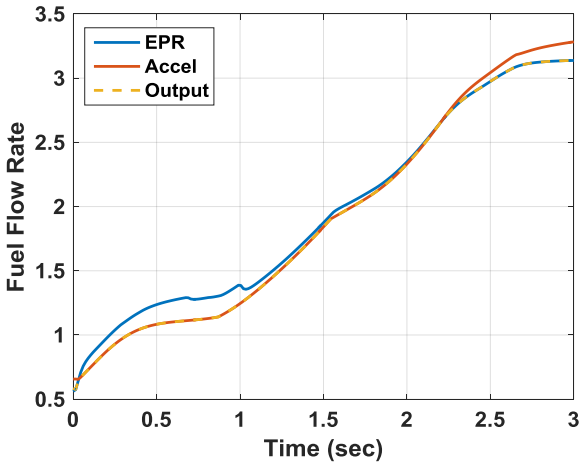

Figure 11. Nominal Fuel Flow Rate

The IFT optimization was performed with a step size of $\gamma = 0.1$ for an incremental number of iterations. The PI control parameters were initialized to their nominal fixed values of $K_p = 0.001823$ and $K_i = 0.003682$ defined in the C-MAPSS40k model. After five iterations of the control parameter optimization, the response improved as shown in Figure 12. The associated decrease in tracking error cost and the evolution of the proportional and integral gains are shown in Figure 13, Figure 14, and Figure 15, respectively.

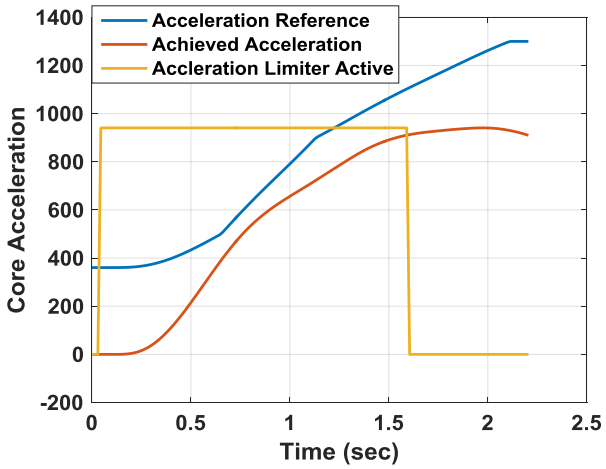

Figure 12. Tracking Response - 5 Iterations

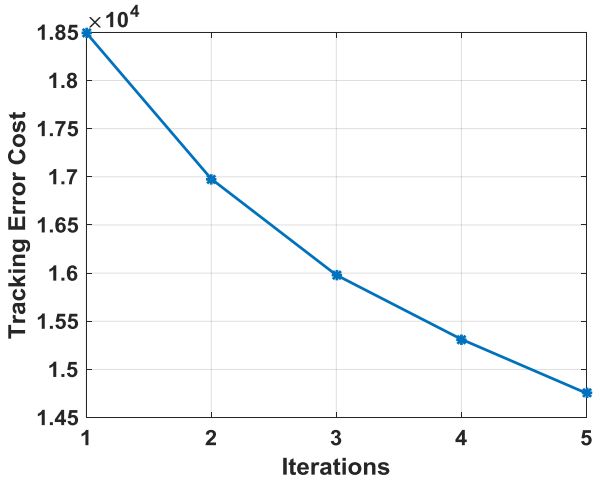

Figure 13. Tracking Error Cost - 5 Iterations

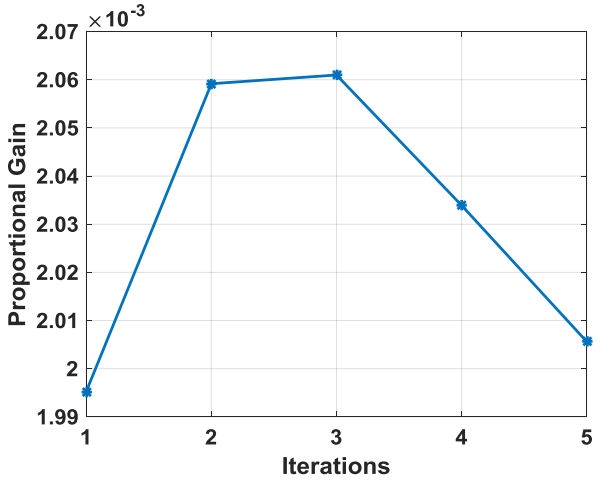

Figure 14. Proportional Gain - 5 Iterations

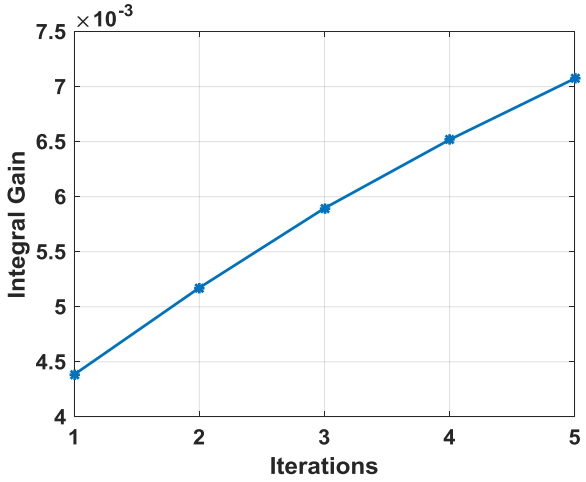

Figure 15. Integral Gain - 5 Iterations

After fifteen iterations of the control parameter optimization, the response improved as shown in Figure 16. The associated decrease in tracking error cost and the evolution of the proportional and integral gains are shown in, Figure 17, Figure 18, and Figure 19, respectively. While the tracking performance is obviously improved with the new gains, neither the tracking cost nor the PI control gains appear to have converged.

After twenty-five iterations of the control parameter optimization, the response improved substantially as shown in Figure 20. The associated decrease in tracking error cost and the evolution of the proportional and integral gains are shown in, Figure 21, Figure 22, and Figure 23, respectively. The tracking performance is greatly improved using the new gains. In addition, the tracking cost appears to be converging as well as the integral gain term. However the proportional term has not converged. Nonetheless, twenty-five iterations was adopted as the convergence point for the acceleration limiter when the algorithm was applied across a range of Mach and Altitude numbers.

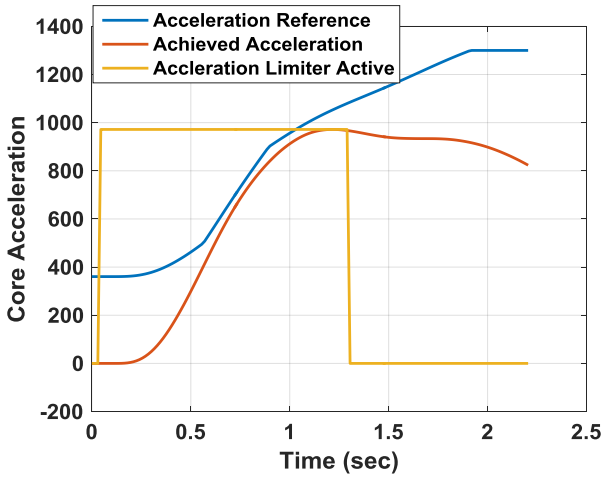

Figure 16. Tracking Response - 15 Iterations

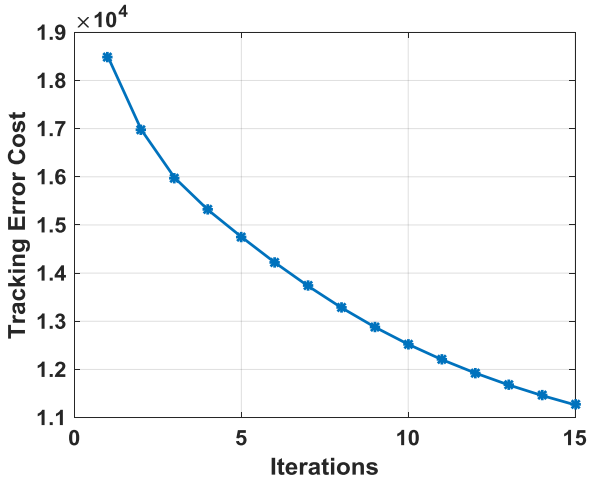

Figure 17. Tracking Error Cost - 15 Iterations

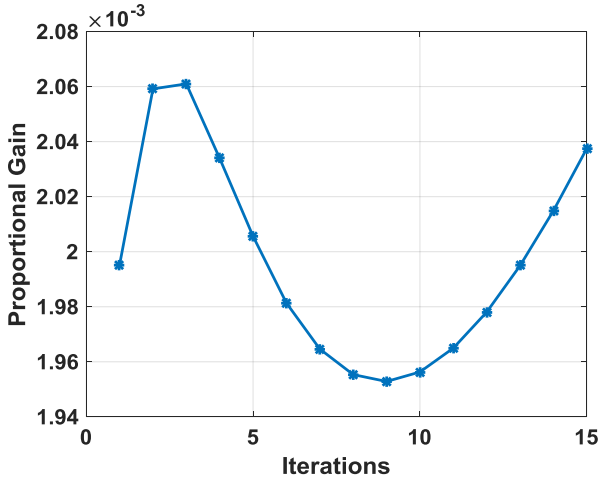

Figure 18. Proportional Gain - 15 Iterations

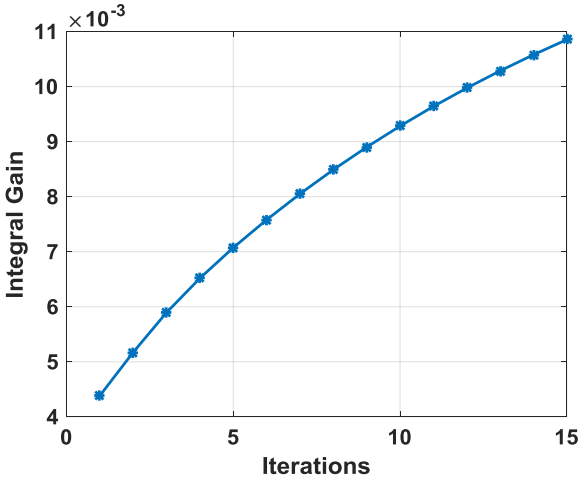

Figure 19. Integral Gain - 15 Iterations

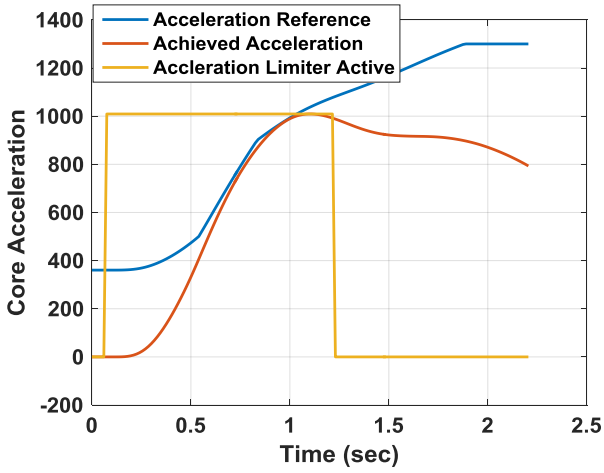

Figure 20. Tracking Response - 25 Iterations

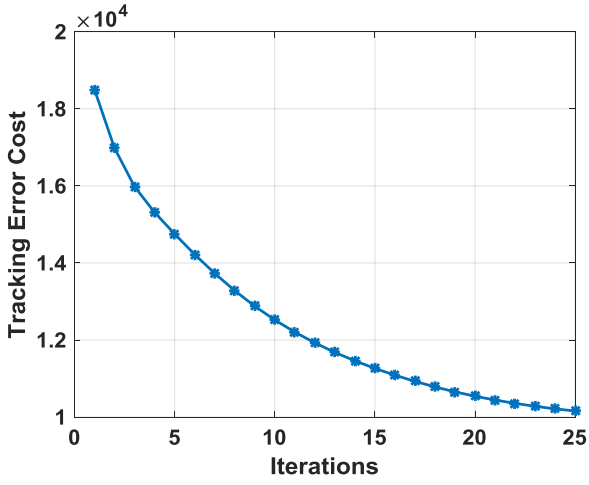

Figure 21. Tracking Error Cost - 25 Iterations

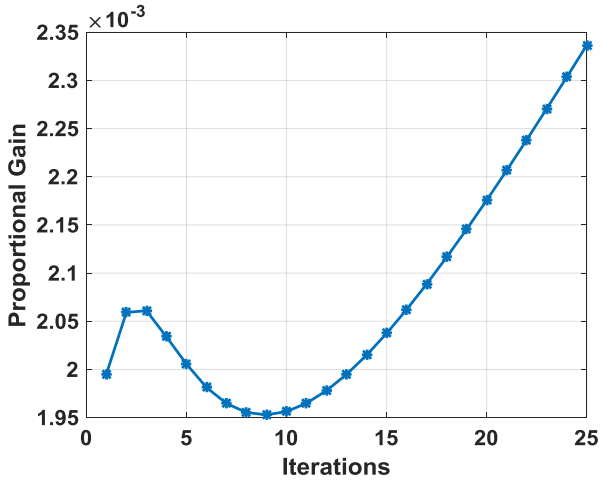

Figure 22. Proportional Gain - 25 Iterations

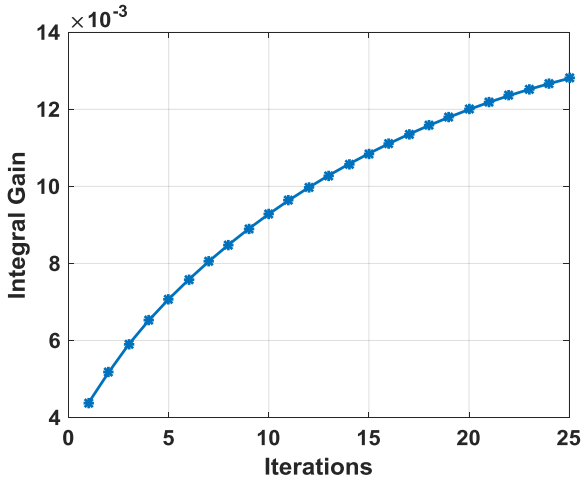

Figure 23. Integral Gain - 25 Iterations

To ensure convergence of the algorithm, the optimization was run for 100 iterations to observe the change in the tracking response and the evolution of the tracking error cost and the optimal control gains. The tracking response in Figure 24 is slightly worse than the twenty-five iteration case which corresponds to the slight uptick in the tracking error cost in Figure 25 as the objective function converges. The proportional control gain in Figure 26 and the integral control gain in Figure 27 both attain their converged values. However, this response might be considered 'over tuned' and the initial minimum of the tracking error cost is a better metric in determining when to stop the gain iteration. As previously stated, the twenty-five iteration case is selected as the ideal response.

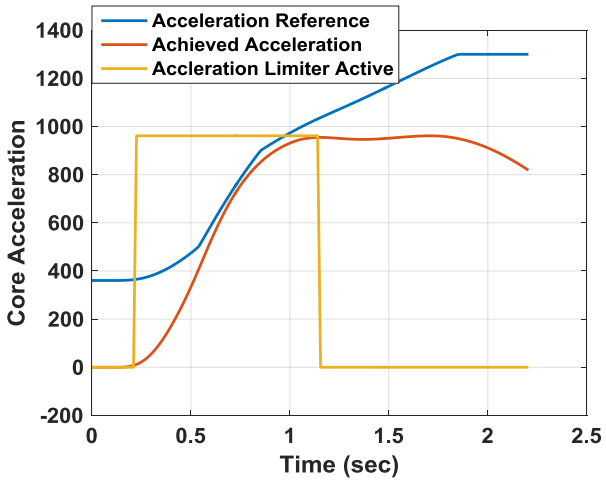

Figure 24. Tracking Response - 100 Iterations

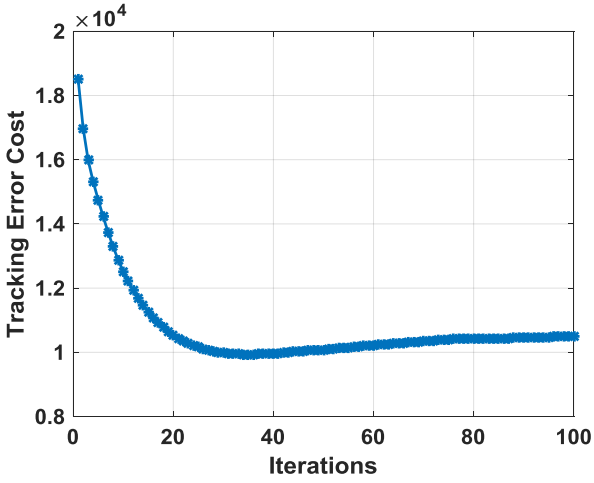

Figure 25. Tracking Error Cost - 100 Iterations

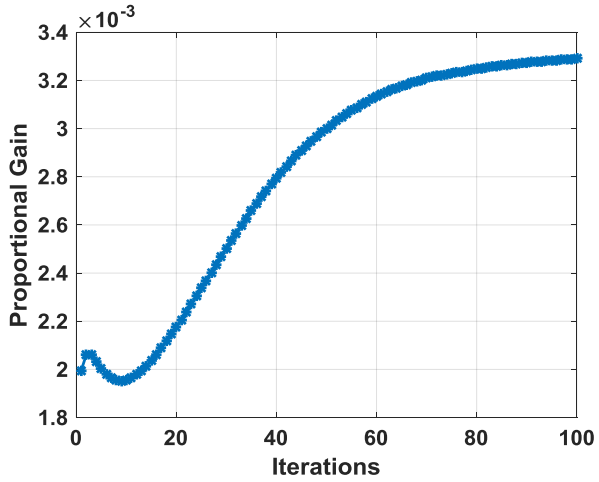

Figure 26. Proportional Gain - 100 Iterations

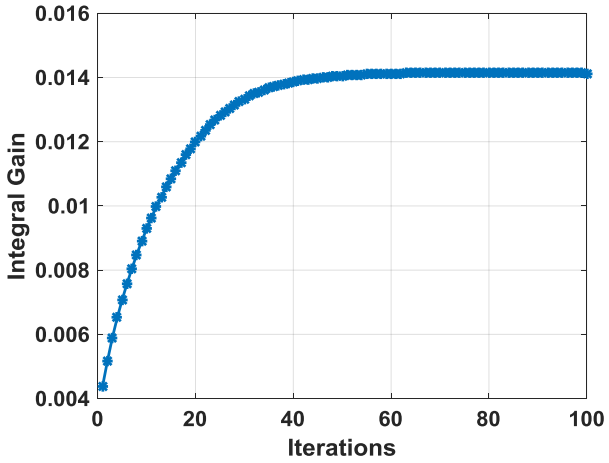

Figure 27. Integral Gain - 100 Iterations

Table 2 summarizes the optimization results using the IFT method. It indicates that the substantial changes in the integral control were most influential on reducing the cost. Table 2 also shows the 'over tuning' corresponding to the increase in cost going from twenty-five up to 100 iterations.

Table 2. Iterated Optimization Results

Iteration	Cost (J x 10^4)	Proportional Gain (K_p x 10^{-3})	Integral Gain (K_i x 10^{-3})
0	1.85	182	368
5	1.42	198	758
15	1.11	206	1111
25	1.01	237	1293
100	1.05	329	1413

F. IFT Application to Varying Environmental Conditions

The IFT algorithm was executed in a loop covering a range of altitude and Mach conditions to evaluate how well the automatic tuning worked over a range of environmental conditions. The same PLA step change was applied at time 0 for each evaluation case. Figure 28, Figure 29, and Figure 30 show the improved tracking response corresponding to 25 IFT iterations for a range of altitude and Mach conditions.

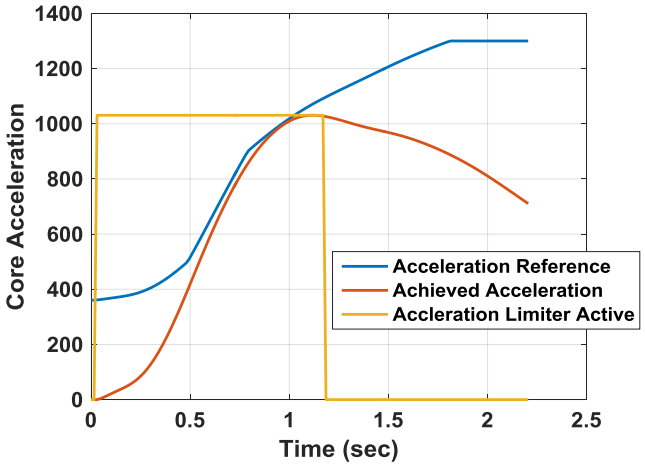

Figure 28. Acceleration Tracking Response – Alt = 10k feet Mach = 0.4

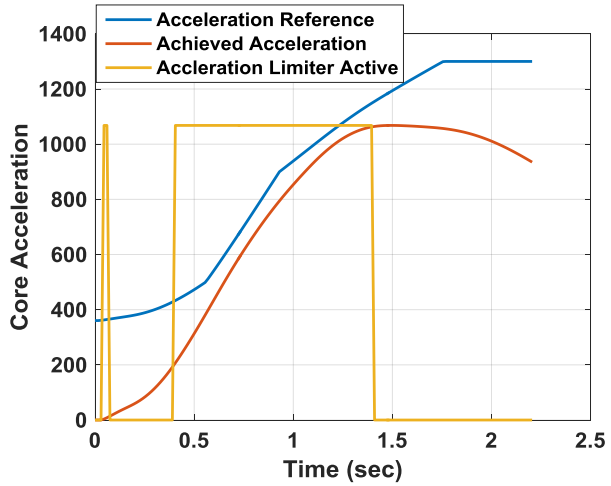

Figure 29. Acceleration Tracking Response – Alt = 10k feet Mach = 0.8

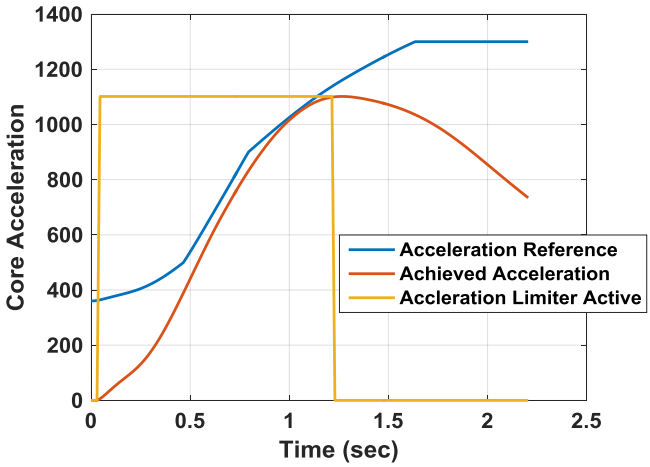

Figure 30. Acceleration Tracking Response – Alt = 20k feet Mach = 0.6

Table 3 shows the optimal integral gain matrix corresponding to variations in altitude and Mach. There is a maximum 15% change in the integral gain which indicates that scheduling over a range of altitude and Mach values might not be necessary for achieving improved response. However, each of these cases was run with a fixed number of 25 iterations. If an intelligent iteration stop was implemented based on achieving the minimum tracking error cost, then improved performance might be attained corresponding to a larger spread in the integral gain values.

Table 3. Integral Gain, K_i Matrix (x 10^{-3})

Altitude (x 1000 ft)	M = 0	M = 0.2	M = 0.4	M = 0.6	M = 0.8
0	127	127	129	137	X
10	140	139	135	136	144
20	X	X	X	147	145

G. IFT Modifications for Acceleration Limiter

The IFT algorithm requires a fixed number of samples to perform the cost computations. Initially, this was selected as the entire length of the acceleration ramp-up due to the PLA step change. However, this did not result in convergence of the algorithm. This is undoubtedly tied to the nonlinearity of the system. A modified form of the IFT algorithm allows the cost functions to be computed over a selected range of the closed-loop simulation time. A 'mask' of length t_0 is defined as

$$J(\rho) = \frac{1}{2N}\left[\sum_{t=t_0}^{N} \tilde{y}_t(\rho)^2\right] \tag{16}$$

where the cost function is summed from sample $t = t_0$ to $t = N$ instead of $t = 1$ to $t = N$. The mask is generally used to bypass the transient response in order to focus the control parameter degrees of freedom on the steady-state response. However, for the acceleration limiter, it was found that setting the mask to the initial portion of the ramp response resulted in excellent convergence. Therefore for the current implementation, all summations are performed over $t = 1$ to $t = t_f$ where t_f is the designated mask length. Several mask lengths were tested before settling on the final value of t_f = 80 samples. Each experiment was run for the same total number of samples, *N*, which covered the entire acceleration ramp. However, the cost computations were only summed over $t = 1$ to $t = 80$. The implications of the limited mask are an area for future research.

VI. Conclusions

This work focused on integration of a runtime assurance (RTA) framework into an existing advanced turbofan engine control system. One part of the RTA system is defined as the set of logic and decision functions required to determine if observed anomalous conditions are due to errors in the advanced controller. Construction of a general method to perform such checks is ongoing, but it is proposed here that the RTA monitoring and decision functions query performance maps and other functions online to determine if the advanced controller is at least delivering the expected performance that can be achieved by the trusted reversionary controller. To demonstrate this idea, a set of experiments were performed comparing expected engine pressure ratio (EPR) responses from the reversionary controller to what was achieved by the advanced controller. It was shown that when the EPR performance did not meet defined minimum standards, the RTA system switched operations to the reversionary controller. These experiments also showed that smooth transitions from the advanced controller to the reversionary controller occurred automatically when it was deemed that the advanced controller was not operating correctly. This was largely due to the reversionary controller being run in "shadow mode" while the advance controller was active. This helped to alleviate transients and other control mode switching issues.

The other main part of the RTA system is defined to be its safety critical function. Engine protection logic, which already exists within the turbofan engine control system, was determined to fill this role. However, the current design of the protection logic is considered conservative and improvement in its tracking performance is desired. To address this problem, the iterative feedback tuning (IFT) algorithm was employed to determine optimal control gains over a range of Mach and altitude values. A matrix of optimal proportional-integral (PI) feedback gains was constructed to cover this range of environmental conditions and resulted in greatly improved engine core acceleration tracking at all operating conditions tested. A major advantage of the IFT approach is that improved performance is attained without modifying the existing protection logic control structure. Therefore, no additional burden is added to the certification process for this well-established protection methodology.

A key result of this investigation is that the benefits of RTA technology are evident for protecting engine systems with untrusted advanced controllers. Turbofan engines have a clearly defined, finite set of physical and performance limit constraints that define the operating envelope and make developing an RTA system quite feasible. The constraints are defined for measurable parameters such as fan and compressor speeds, and temperatures and pressures at various stages through the engine. Safe operation and meeting performance requirements can therefore be clearly defined and checked by the RTA system. This is in contrast to other RTA applications involving systems with complex, multi-dimensional operating envelopes and complex definitions of safety. In such cases, successful RTA designs can be difficult to achieve and quite complex.

Acknowledgements

This work was funded by the NASA Glenn Research Center under contracts NNC15CA27C and NNX16CC99P. The authors would like to acknowledge the important contributions from NASA and express gratitude for their support.

References

1. Simon, D., "An Integrated Architecture for On-Board Aircraft Engine Performance Trend Monitoring and Gas Path Fault Diagnostics," NASA/TM—2010-216358, Prepared for the 57th Joint Army-Navy-NASA-Air Force

(JANNAF) Propulsion Meeting sponsored by the JANNAF Interagency Propulsion Committee Colorado Springs, Colorado, May 3–7, 2010.
2. Connolly, J., Chicatelli, A., Garg, S., "Model Based Control of an Aircraft Engine using an Optimal Tuner Approach," AIAA 2012-4257, *Proc. 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Atlanta, Georgia, July-August, 2012.
3. ARP4754A, anon., Aircraft and System Development Processes, Washington, D.C., SAE Aerospace International, 2010.
4. ARP4761, anon., Safety Assessment Process Guidelines and Methods, Washington, D.C., SAE Aerospace International, 1996.
5. DO-178C, anon., Software Considerations in Airborne Systems and Equipment Certification, Washington, D.C., RTCA, 2011.
6. Seto, D., Krogh, B., Sha, L., Chutinan, A., "The Simplex Architecture for Safe Online Control System Upgrades," Proc. American Control Conference. Philadelphia, PA, June, 1998, pp. 3504-3508.
7. Seto, D., Sha, L., "An Engineering Method for Safety Region Development, Paper 137," Software Engineering Institute, 1999, http://repository.cmu.edu/sei/137 (Current as of 12/2017).
8. Sha, L., "Using Simplicity to Control Complexity," IEEE Software 18(4), pp. 20-28, July/August 2001.
9. Sha, L., Goodenough, J., Pollak, B., "Simplex Architecture: Meeting the Challenges of Using COTS in High-Reliability Systems," CROSSTALK The Journal of Defense Software Engineering, April 1998.
10. Sha, L., Rajkumar, R., Gagliardi, M. "A Software Architecture for Dependable and Evolvable Industrial Computing Systems," Carnegie Mellon University Technical Report CMU/SEI-95-TR-005 ESC-TR-95-005, July, 1995.
11. May, R., Csank, J., Lavelle, T., Litt, J., Guo, T., "A High-Fidelity Simulation of a Generic Commercial Aircraft Engine and Controller," NASA/TM—2010-216810, Oct. 2010, also published by AIAA as paper no. AIAA–2010–6630.
12. Csank, J., May, R., Litt, J., Guo, T., "Control Design for a Generic Commercial Aircraft Engine," NASA/TM—2010-216811, also presented at the 46th Joint Propulsion Conference and Exhibit Nashville, TN, July, 2010, AIAA Paper No. AIAA–2010–6629.
13. Connolly, J., Chicatelli, A., Garg, S., "Model Based Control of an Aircraft Engine using an Optimal Tuner Approach," AIAA 2012-4257, Proc. 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, Atlanta, Georgia, July-August, 2012.
14. Sallee, G., Gibbons, D., "Propulsion System Malfunction Plus Inappropriate Crew Response (PSM+ICR)," Flight Safety Foundation, Flight Safety Digest, Nov.-Dec., 1999, pp. 1-193.
15. May, R., Garg, S., "Reducing Conservatism in Aircraft Engine Response Using Conditionally Active Min-Max Limit Regulators," Paper No. GT2012-70017, Proceedings of ASME Turbo Expo, June, 2012, Copenhagen, Denmark.
16. Hjalmarsson H., Gunnarsson, S., Gevers, M., "A Convergent Iterative Restricted Complexity Control Design Scheme," Proc. of the 33rd CDC, 1994.
17. Hjalmarsson H., Gevers, M., Gunnarsson, S., Lequin, O., "Iterative Feedback Tuning: Theory and Applications," IEEE Control Systems Magazine, 18, 1998.