Transient Optimization of a Gas Turbine Engine

Jonathan L. Kratz*
NASA Glenn Research Center, Cleveland, Ohio, 44135, U.S.A.

Gas turbine engines are the primary power plants for modern commercial aircraft. Transients prompted by significant changes in thrust or power demand are common and unavoidable. Extreme transient scenarios such as those associated with a go-around during a landing attempt are possible and must be accounted for in the design of the engine and its controller. Engine transients tend to cause a reduction in compressor operability margin, which must be addressed by the engine control system and accounted for in the engine design to prevent events such as compressor stall/surge and combustor blow out. Transient operability concerns typically lead to compromises in the engine design that sacrifice efficiency and/or limit responsiveness. Transient operability is typically managed by logic that limits the fuel flow command. If this logic is not optimized, then the potential for valuable performance could be lost. This study presents a strategy for optimizing the transient limit logic and proposes a strategy for updating the control logic over the lifespan of the engine. The results demonstrate significant improvements in transient operability. For example, of the results at sea level static conditions demonstrated a 31% reduction in the usage of the high pressure compressor operability stack during a snap acceleration transient. Furthermore, a reinforcement learning algorithm is demonstrated to modify the transient logic as the engine degrades to minimize response time while respecting a prescribed compressor operability margin limit. A simple demonstration of the reinforcement learning algorithm resulted in a thrust response time reduction of ~11.8%.

I. Introduction

Most modern commercial aircraft are powered and propelled by gas turbine engines. These workhorses of the aviation industry will remain crucial components of aircraft propulsion systems, even as new concepts such as electrified aircraft propulsion (EAP) are pursued. Many EAP concepts, particularly large aircraft concepts, retain the use of gas turbine engines through hybrid gas-electric propulsion. The point is that gas turbine engines will remain the predominate powerplant of the aviation industry for many years to come.

One challenge of gas turbine engines is maintaining operability during engine power transients and throughout a vast operating envelope. The design of transient control logic takes up nearly 75% of the total time dedicated to engine control system development [1]. Engine power transients occur when there is a change in thrust or power demand, often associated with movement of the throttle position by the pilot. Transients result in a change in the shaft speeds of the gas turbine engine. Gas turbine engines often have two shafts that are referred to as the low-pressure shaft (LPS) and high-pressure shaft (HPS). Each shaft is attached to a compressor and a turbine that apply opposing torques. During an acceleration or deceleration, there is a temporary imbalance in torques. This is the result of a change in fuel flow that increases or decreases the amount of energy in the flow path of the turbine and the amount of power that it extracts from the flow to drive the compressor. The non-zero net torque causes a change in shaft speeds. However, the shafts have a considerable amount of inertia, and thus the change in speed is not instantaneous. A mismatch in the internal engine air flow and shaft speeds results in off-incidence flow impinging on the compressor blades. If the off-incidence flow is severe enough, compressor stall or surge can occur. For a two-spool engine, the tendency is for the high-pressure compressor (HPC) operability to degrade during accelerations and the low-pressure compressor (LPC) operability to degrade during decelerations.

* Research Engineer, Intelligent Control & Autonomy Branch, AIAA senior member.

In theory, the transient operability issue could be mitigated by changing the throttle very slowly such that a quasi-steady-state condition is maintained. However, this is not possible from a practical perspective. Aero-engines must be able to accelerate and decelerate relatively quickly to meet Federal Aviation Administration (FAA) and user requirements. FAA regulations state that a commercial engine must be able to accelerate from 15% thrust to 95% thrust within 5 seconds for relevant flight conditions [2]. For some aircraft, especially military aircraft, there are user-imposed requirements to respond even faster. Operability issues must be dealt with through the engine design and its control logic. The transient control logic is typically found in the form of a fuel flow rate limiter the enforces a shaft acceleration/deceleration ($\dot{N}$) limit or a ratio unit (RU) limit in a max-min switch logic structure [3] or command governor (CG) logic [4]. The ratio unit is the ratio of the fuel flow rate and the HPC static discharge pressure (w_f/p_{s3}). Additionally, more advanced control schemes have been studied, including the use of Model Predictive Control (MPC) [5,6] and Linear Parameter Varying control [7].

Regardless of the controls approach, the control logic can only manage the issue, not eliminate it. As a result, the engine must be designed to account for some degree of variability in operability as the engine goes through transients. Thus, transient operability places constraints on the engine design that forces compromises in engine performance. For applications that desire fast responsiveness, this can limit acceleration and deceleration rates. In addition, constraints placed on the engine design can impact efficiency metrics such as fuel burn. Ref. [8] states that roughly one third of the compressor operability allowance can be attributed to transient operation while Ref. [9] states that about half of the stall margin can be devoted to transient operations. Furthermore, high duty compressors tend to achieve maximum efficiency close to the stall line [8]. The highest efficiency of an LPC may occur near or below 10% stall margin (SM) and the shifting of the operating line demanded to maintain compressor stability during transients could cost more than 3% in compressor efficiency [8]. The ability to operate closer to stall could influence the design itself, resulting in different performance maps and a lighter design. Ref. [10] discusses trades between a traditional optimal design and a robust optimal design, which must account for various uncertainties and a wide range of operation that includes transients. Inevitably, sacrifices are made in the design to assure safe and reliable operability.

The methodology for control design could be inherently sub-optimal in application. The NASA-developed Tool for Turbine Engine Closed-loop Transient Analysis (TTECTrA) [11] is a nice tool for preliminary transient control design and dynamic system analysis. It can be used to design transient control schedules. However, it does so assuming a fixed fuel flow input profile, which happens to be a linear ramp. The software utilizes an iterative solver to adjust the slope of the ramp until stall margin and response time constraints are met. The fixed form of the fuel flow input will almost certainly result in a sub-optimal solution. There is plenty of literature and patents related to optimizing the transient control logic. Much of the focus of the literature is on military applications for which engine responsiveness is of great importance. Ref. [12] and [13] are examples of patents in this subject area. The former adapts bleed valve and $\dot{N}$ schedules based upon estimated combustor discharge temperature to minimize engine response time without stalling the engine. The latter has the same goal but utilizes a simulated compressor stall limit signal that is converted to a desired burner pressure limit. That limit is regulated via control either directly or through a proxy measurement. Ref. [14] applies a method referred to as an extrapolation approach to design the transient limit schedule. This method does not require a dynamic model, in contrary to most other methods, but it is known to be less accurate. Ref. [15] performs an optimization of the transients to reduce response time under operability constraints. It also avoids the need for a transient model but tends to be more accurate than the prior mentioned method. It utilizes a noteworthy method referred to as the virtual power extraction method (VPEM) that applies power extraction to the engine via steady-state simulations to shift the operation of the engine and emulate transient conditions. Reference [16] proposes the use of a variable replacement method along with particle swarm optimization to improve transient control performance by modifying the gains of a proportional integral (PI) $\dot{N}$ transient controller. Ref. [17] applies a genetic algorithm to tune the gains of a max-min switch logic controller that includes transient limiters with an objective function that considers the thrust response time and transient fuel consumption. Both approaches are limited as they assume static $\dot{N}$ limit set-point determination. Ref. [18] uses a genetic algorithm to create open-loop fuel flow and nozzle area commands that minimize the thrust response time. The study covered in this paper has similarities to Ref. [18] but differs in several ways including differences in the problem formulation and details about the genetic algorithm, extension of the results to a traditional control schedule, application to a commercial engine, and extension of the strategy toward life-cycle optimization. These are just some examples of the literature on this rich topic of research.

Even if an “optimal” acceleration/deceleration controller is developed and employed on a new engine, it will likely be or become sub-optimal. Reasons for this include factors such as engine-to-engine variations, differences in operation (e.g., varying levels of engine power extraction), secondary effects such as heat soakage, and shifts in performance as the engine ages. In addition, the logic is typically designed with worst case conditions in-mind, and therefore the potential for a faster response or better performance characteristics could be squandered for any given

transient that isn't the worst case. The concept of digital twin could help to improve this situation in the future. A digital twin is a virtual representation of a connected physical asset [19], in this case a gas turbine engine. Gas turbine engines are prime candidates for digital twin given their abundance of data. Ref. [20] approximates that 20TB of data can be collected from an engine per hour. The definition provided by AIAA and the Aerospace Industries Association (AIA) in Ref. [19] is "A set of virtual information constructs that mimic the structure, context and behavior of an individual/unique physical asset, or a group of physical assets, is dynamically updated with data from its physical twin throughout its life cycle and informs decisions that realize value." Numerous potential applications of a digital twin exist including enhancing operational performance through controls. This may include using the digital twin to update control schedules and tuning control gains. It can also extend to using the digital twin model within the control logic, otherwise referred to as model-based engine control (MBEC). Ref. [21], [22], [23] and [24] describe MBEC strategies that utilize a tracking filter for tuning the model to the real system and utilizing the model outputs for controls. Ref. [25] is an example of using machine learning to update the digital twin. A recent application of a digital twin is demonstrated by a software tool released by General Electric to optimize gas turbine operations [26]. The software utilizes artificial intelligence to build a machine learning digital twin model. The model is used to determine the optimal flame temperature and fuel splits that minimize emissions and acoustics. This technology along with the prior work mentioned above provides promising signs of the near-term technology readiness of digital twins to influence engine controls.

The application in this paper is for a commercial single-aisle engine, for which the operations are constrained and predictable compared to military applications, minimizing component life usage during takeoff and landing transients, and minimizing cruise fuel consumption are among the primary concerns [18]. Thus, the goal of this study is to optimize acceleration and deceleration control logic to minimize variations from the steady-state operating line such that the engine design can be improved to enhance efficiency and the transients will be less harsh in terms of the metrics such as compressor operability and peak temperatures. The study considers various optimizations and simulations to draw a variety of conclusions. The topics investigated include: the impact on compression system operability and peak operating temperatures that contribute to engine deterioration, and the ability to generalize the form of an optimal solution for a given flight condition to other flight conditions. A strategy for updating the transient limit logic throughout the engine lifespan is also explored.

The approach involves the use of a nonlinear model of a conceptual advanced geared turbofan and its controller to simulate various engine transients. A genetic algorithm optimizer is given control of the control inputs along with the objective to optimize a measure of compressor operability and to achieve a given response time. The results are used to derive an *RU* limit schedule. Assuming the model to be a digital twin of a specific physical engine, additional optimizations are conducted at different health states of the engine, and those results are used to guide a reinforcement learning algorithm to gradually update the control logic as the engine ages.

The rest of the paper is organized as follows. Section II provides a brief overview of the Advanced Geared Turbofan 30,000lb_f (AGTF30) engine model [27], which is the plant considered in this study. Section III describes the genetic algorithm employed in the study and Section IV covers the transient optimization procedure and results. Section V comments on extending the optimization to updating the control schedules using a digital twin. Finally, Section VI provides a summary.

II. The AGTF30 Propulsion System

The AGTF30 is a model of a conceptual two-spool geared turbofan capable of producing ~30,000 lb_f of thrust at sea level static (SLS) conditions. The engine is envisioned for a single-aisle commercial transport application. The AGTF30 is meant to be representative of technology available in 2035 and includes features such as a compact gas turbine core and a variable area fan nozzle. The engine model is coded in the MATLAB/Simulink® environment using the NASA-developed Toolbox for Modeling & Analysis of Thermodynamic Systems (T-MATS) [28]. T-MATS provides the building blocks for creating a 0-D (component level) model of a gas turbine engine utilizing

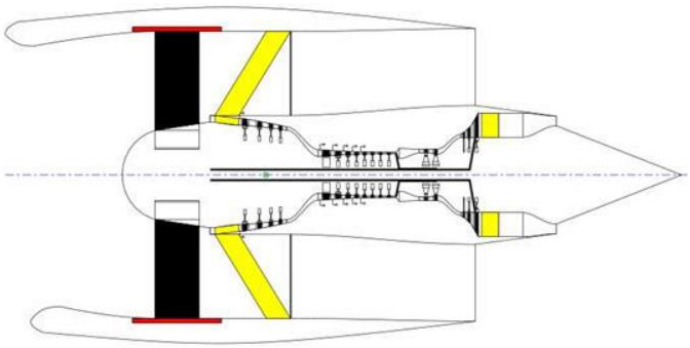

Figure 1. Diagram of the AGTF30 conceptual engine

turbomachinery performance maps, thermodynamic relations, actuator models, and more. The AGTF30 is represented in Fig. 1. The model includes a realistically performing full-flight envelope controller that was developed in Ref. [27]. This controller includes schedules for the variable area fan nozzle and variable bleed valve. It also includes closed-loop gain-scheduled PI controllers for the fuel flow rate that include a nominal corrected fan speed controller and various limit controllers. Among the limit controllers are limiters for over-speed and over-temperature conditions. The model also includes acceleration and deceleration limit logic, which is employed as a maximum *RU* limit schedule for acceleration and a minimum *RU* limit schedule for deceleration. The various fuel flow rate commands go through a max-min decision tree to decide which command to use. Health parameters are used within the engine model to set the health state of the turbomachinery components. Health parameters are modifiers that shift the flow capacity and efficiency of the compressors and turbines based on degradation. The degradation model that relates the health parameters to engine life was taken from another engine model known as the Commercial Modular Aero-Propulsion System Simulation 40,000lb_f engine model [29].

III. The Genetic Algorithm

Genetic algorithms are optimization schemes built upon the biological principles of natural selection and fitness [30]. Genetic algorithms tend to be less likely to get stuck at local minima/maxima than gradient based methods and are substantially more efficient than brute force methods. A population is comprised of numerous solution realizations, each with different parameters. Each individual solution is evaluated based upon a fitness function. The members of the population compete for survival into the next generation and for participation in reproduction. The genetic algorithm utilized in this study has the primary components of elitism, carry-over (replication), reproduction (crossover), and immigration. The primary sub-components of the genetic algorithm are selection, mutation, and duplication removal. Each of these components and sub-components will be described in the following paragraphs. The sub-components are described first to set the foundation for describing the components.

The two selection methods used in this application were random and rank-biased selection. In random selection, all members have the same probability of being chosen. Rank-based selection utilizes the pareto distribution [31] and allows the user to specify parameters that define the exact shape. For instance, the 80-20 rule [31] can be applied by specifying that the probability of selecting a member from the top 20% will be 80%.

Mutation creates a modified version of a member of the population. It will select the number of parameters to mutate based on specified probabilities and then will randomly select parameters to mutate. Finally, those parameters are mutated within specified bounds using a random distribution.

Duplicate removal applies whenever a duplicate shows up in the population. This feature will remove the duplicate and replace it with new member that is generated within the specified parameter bounds using a random number generator.

Elitism involves advancing a set number of the most fit individuals to the next generation. Elitism seeks to preserve the best solutions and enable them to take part in finding better solutions through the functions of reproduction and mutation. In addition to advancing the elite, mutated variants of the elite may also be added to the next generation. This action promotes diversity but also exploits the high fitness of the elite. Inputs include the number of top members of the population to include in the elite, the number of the elite to mutate, the bounds for mutation, and the probability of mutating any number of parameters up to the full number of parameters.

The carry-over component of the algorithm selects members of the population, outside of the elite, to advance to the next generation. Mutation can apply as these members are carried to the next generation. The inputs include the number of members to carry-over, the method of selection, inputs associated with the method of selection, and inputs associated with mutation including the bounds of mutation, the probability of a mutation occurring, and the probability of any number of parameters being mutated.

Reproduction consists of the combining of two members of the population to produce one or more new members of the population in the next generation. The offspring will derive its parameters from its parents. There are 3 methods for assigning parameters. The first is to inherit the parameter from one of the parents. The second is to average the parameters of the parents. The final is to randomly select a value for the parameter between the values of the parameter for the two parents. The probability of each method being used can be specified. In this application, each method had an equal chance of use. The parents are chosen based on the specified selection method, as is the number of offspring that a pair of members will produce. The combined fitness of the parents can be utilized to determine the number of offspring. Limits can be set for the number of times a single member of the population can participate in reproduction. In this application the number of offspring is specified. After the reproduction function is carried out, the offspring can be mutated given inputs about the probability of mutation, the bounds of mutation, and the probability of mutating any number of parameters.

Immigration refers to the introduction of new members to the population that will appear in the next generation. The new members are generated within the specified parameter bounds using a random number generator. This feature helps to explore the solution space more thoroughly.

In this application, the members of the population define a fuel flow command input profile for the transient. The profile is defined by 8 points between the minimum and maximum fuel flow rate. The time at each data point in the profile is constant. The variables in the optimizer include 9 values between 0 and 1, *Y*, that are used to define the change in fuel flow between each of the 8 data points. The fuel flow rate value is a function of *Y*.

$$w_{f,i} = w_{f,i-1} + \frac{Y_i}{\sum_{j=1}^{9} Y_j}\left(w_{f,max} - w_{f,min}\right) \tag{1}$$

In the equation above, w_f is the fuel flow rate, and *i* and *j* refer to the indices of the time interval that each fuel flow change occurs over. The variable *i* differs from *j* by referring only to the time interval for which the fuel flow rate change is being calculated in the equation. The subscripts "*max*" and "*min*" refer to maximum and minimum fuel flow rate. This approach guarantees the fuel flow rate input remains monotonically increasing or decreasing between the minimum and maximum fuel flow rate. The population was initialized using a random number generator for a population of 45 and the genetic algorithm was run for 50-200 generations. For each generation, the fitness of new members was evaluated. This entailed running a transient simulation and calculating fitness for each member. The fitness, *f*, is defined in Eq. (2) where *TSU* is the transient stack usage, t_r is thrust response time, and $t_{r,target}$, is the thrust response target value.

$$f = \frac{1}{TSU} + \frac{1}{10\,max\left(t_r - t_{r,target}, 0\right) + 1} \tag{2}$$

The thrust response time for an acceleration is the time to go from idle thrust to 95% thrust. The response time for a deceleration is the time to go from idle thrust to 20% thrust. The target thrust response time was chosen to match the response time of the engine with the baseline controller. The *TSU* is a metric developed by the author to quantify operability margin. The metric is a single value that quantifies the operability margin for a given transient. The metric is defined below:

$$TSU = max\left(\frac{PR - PR_{SS}}{PR_{stall} - PR_{SS}}\right) \times 100\% \tag{3}$$

PR is the pressure ratio, PR_{SS} is the pressure ratio at the same corrected flow rate along the steady-state operating line, and PR_{stall} is the pressure ratio at the same corrected flow rate along the stall line. Each variable is a vector that varies throughout the transient. The *TSU* metric quantifies what portion of the compressor operability stack is used during the transient. By quantifying the operability margin with a single value for the entire transient using map data that considers the stall line and steady-state operating line, this metric gives a decent summary of the compressor operability without some of the nuances of stall margin. Figure 2 illustrates these terms on a compressor map. W_c on the x-axis is the corrected flow rate at the inlet of the compressor. It is noteworthy that the fitness function could be modified to utilize other operability metrics, such as minimum stall margin, or combinations of various operability metrics.

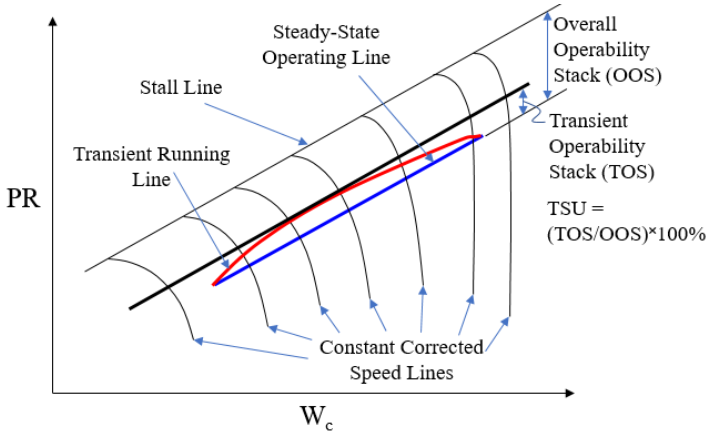

Figure 2. Illustration of *TSU*

IV. Transient Optimization

The transient maneuvers evaluated will be full power range bursts and chops with a new undegraded engine. A burst is characterized by a rapid increase in power/thrust and a chop is characterized by a rapid decrease in power/thrust. Typically, a hot Bodie re-acceleration (hot re-slam) is considered the worst-case scenario [9]. In such a case, the heating of the metal engine components cause the steady-state gas path characteristics to change such that the operating line is shifted closer to the stall line. Given that the model used in this study did not readily capture engine heat soak effects, standard burst and chops were utilized.

The results of the acceleration optimization at SLS conditions are shown for the HPC in Fig. 3 compared to results with the baseline acceleration schedule from Ref. [27]. Fig. 4 shows similar results for the LPC from the deceleration optimization. Referring to Fig. 3, the thrust response time with the optimized fuel flow input is the same as with the baseline schedule, but the *TSU* is reduced significantly from 38.4% to 20.5%, indicating an operability improvement along with observation of the flatter running line on the compressor map in Fig. 3c. Referring to Fig. 4, the response time is slightly faster for decelerations with the optimized schedule and the *TSU* was reduced from 9.1% to 7.4%. The optimized fuel flow input profile is also observed to result in a better trajectory for the LPC running line as it begins to decelerate.

The optimization was conducted for accelerations and decelerations at other flight conditions. Included altitude (Alt), and Mach Number (MN) combinations were: 5000 ft at Mach 0.2, 15000 ft at Mach 0.3, 20000 ft at Mach 0.6,

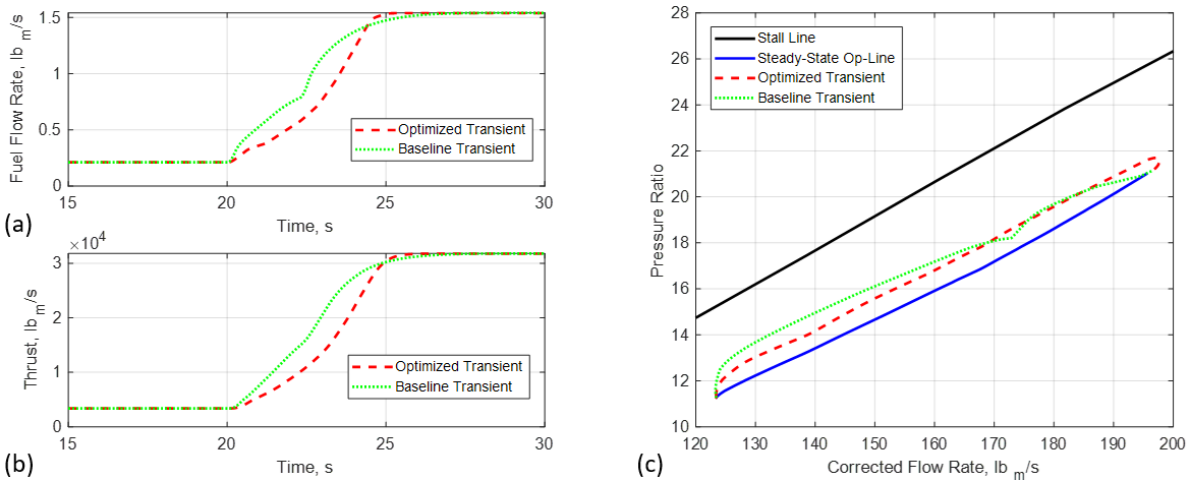

Figure 3. Optimized acceleration results at SLS conditions. HPC data shown in (c).

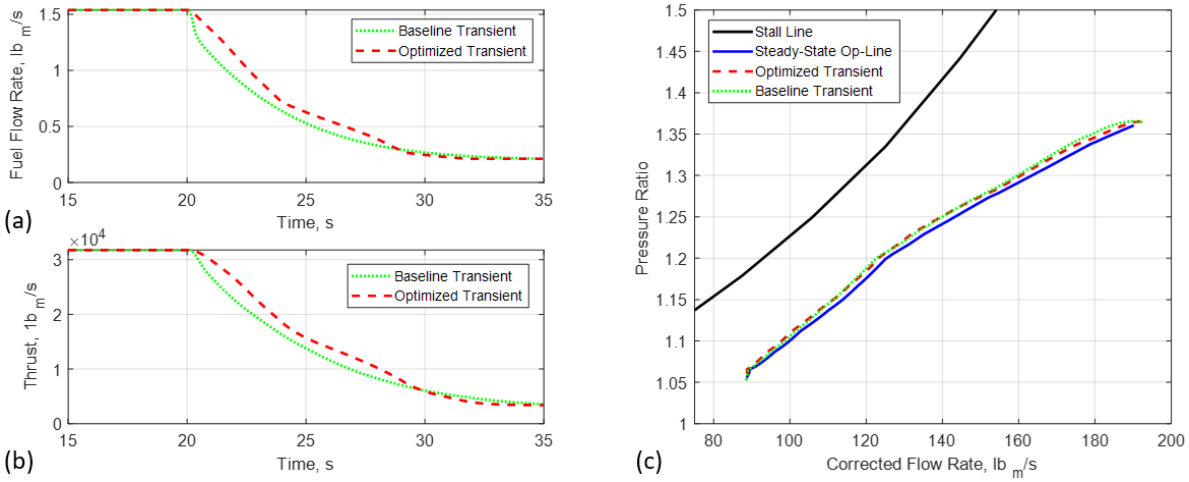

Figure 4. Optimized deceleration results at SLS conditions. LPC data shown in (c)

30000 ft at Mach 0.7, and 35000 ft at Mach 0.8. It was observed that the fuel flow input profile was very similar for all cases. Figure 5 shows the normalized fuel flow input, where $w_{f,norm}$ and t_{norm} are defined below.

$$w_{f,norm} = \frac{w_f - w_{f,min}}{w_{f,max} - w_{f,min}} \tag{4}$$

$$t_{norm} = \frac{t}{t_r} \tag{5}$$

In Eq. (5) t is the time vector during the transient and t_r is the thrust response time. The acceleration fuel flow input profile tends to increase gradually before increasing more rapidly and sharply tapering to the maximum fuel flow rate value. The deceleration fuel flow input profile tends to decease relatively sharply in a nearly linear fashion and then change to a less aggressive trajectory that is nearly linear as the minimum fuel flow rate value is approached. This forms a "kink" in the profile that is associated with the inflection in the LPC running line (see Fig. 4c), which bends to run parallel to the stall line as the variable bleed valve begins to open to manage LPC operability.

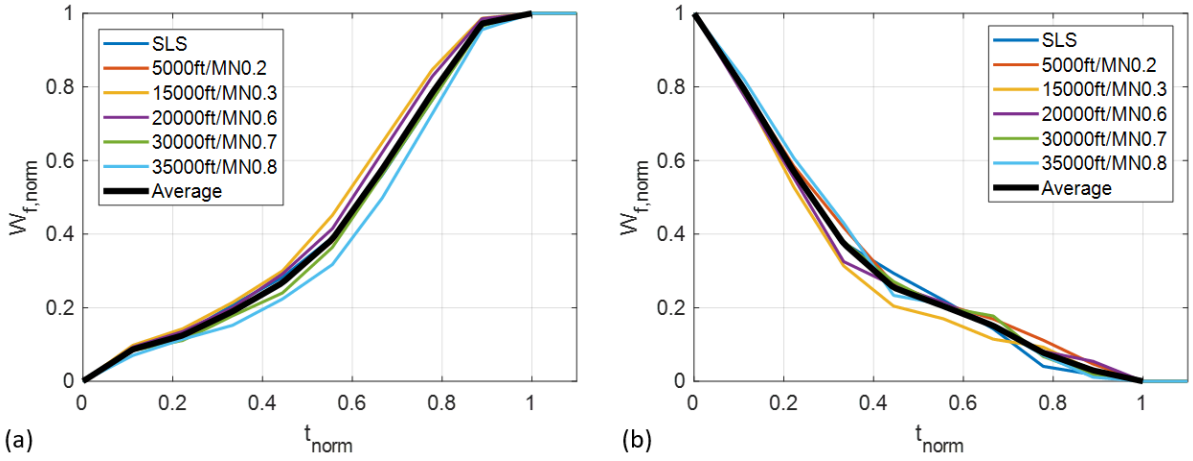

Figure 5. Normalized fuel flow command profiles for burst (a) and chop (b)

It is hypothesized that the fuel flow input profile could be generalized across flight conditions and still achieve near optimal results. Alternatively, the fuel flow input profile could be a function of operating conditions that is derived from optimizations at select operating conditions, such as those listed above. The benefit of either of these choices is a reduction in the workload to perform optimization. With an approximation of the optimal fuel flow input profile an approach similar to that of TTECTrA can be applied throughout the flight envelope to produce acceleration and deceleration schedules. With the form of the fuel flow input profile constant, an iterative solver can be used to stretch or compress the profile to achieve a desired response time while respecting operability constraints. For demonstration, the average normalized fuel flow rate command profile from the optimizations was applied to the transient scenarios at each of the 6 operating conditions investigated prior. An iterative solver was used to achieve a similar thrust response time as the baseline controller.

Figure 6 shows the acceleration results on the HPC compressor map and Fig. 7 shows the deceleration results on the LPC compressor map. Fig. 8 shows a zoomed-in image of the results at 15,000 ft and Mach 0.3 to provide an example of how little variation is observed in the transient running lines. These plots compare the generalized profile to results with the optimized profile for each flight condition. Also present are results that implement acceleration/deceleration schedules that are derived from the generalized profile results. Following the logical and sequential steps laid out in the introduction, those results will be highlighted later in this section. Table 1 and 2 quantify the terms of t_r and *TSU* for accelerations and decelerations respectively. These tables also include results with the derived schedules, which will be discussed later. Note that some of the "optimized" results did not fully converge upon the optimal solution (but are close), as is evidenced by some lower response times, particularly for the deceleration study. This was later improved in the optimization strategy by introducing an iterative solver to achieve the desired thrust response time, rather than incorporating it as a constraint in the fitness function. A slower response with the same fuel flow rate profile should produce more favorable operability results. Overall, the results are very

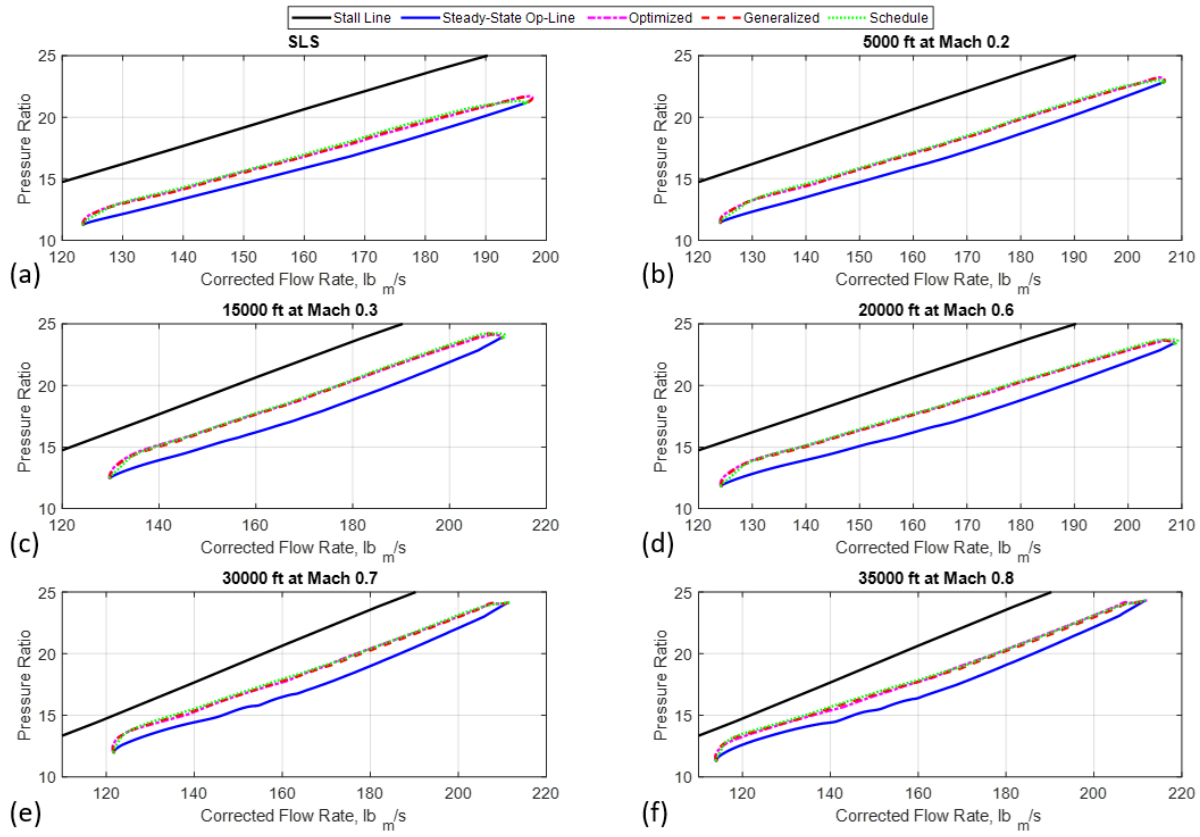

Figure 6. HPC running lines during accelerations transients at various flight conditions

similar and provide support for the theory that the optimal fuel flow input profile can be generalized across various operating conditions. However, it is noted that for the deceleration, the *TSU* with the generalized fuel flow profile is slightly higher than that with the baseline controller at the SLS and 15000 ft Mach 0.3 flight conditions. In general, the generalized input profile provides similar results to the optimized profile but tends to be slightly less optimal in terms of *TSU*. Given the relatively small deviation between the baseline transient running line and the steady-state operating line, the margin for improvement is very small and therefore very small deviations in the running line can make a significant enough change in *TSU* to explain this observation. Observation of the running lines in Fig. 7 provide assurance that the response with the generalized results is quite good, as it is nearly indistinguishable from the optimal solution. The HPC running line during accelerations naturally has much more deviation from the steady-state operating line and in all cases the generalized profile provides a significantly lower *TSU* than the response with the baseline schedule. The *TSU* results and the running lines plotted in Fig. 6 demonstrate similar performance of the optimal and generalized fuel flow input profiles.

Once the fuel flow rate optimization simulation results are obtained, they need to be converted into acceleration and deceleration control schedules that are implementable in practice. To convert the generalized profile results into acceleration and deceleration limit schedules, data from the simulations are utilized. An *RU* limit approach is considered here, but the same idea is applicable to an $\dot{N}$ limit approach. Derived *RU* schedules are constructed as a function of corrected fan speed, $N_{c,Fan}$. To prevent the acceleration and deceleration limit schedules from limiting the fuel flow at the start of the transients, the *RU* limits for accelerations were increased by 3.5% to 5.5% and reduced by 1.5% for decelerations. The resulting schedules are shown in Fig. 9. Figure 6 and 7 show the compressor running lines compared with the optimal and generalized optimal results. It is evident that the results are nearly identical, demonstrating that the limit schedules are working as intended. Table 1 and 2 provide the response time and operability metrics, which help to illustrate the similarity in performance of the engine with the schedule and with an optimized fuel flow input profile. Implementation of the schedule does result in some deviation of the fuel flow input from the data used to produce the schedule and this results in a slight increase in the *TSU*, which may also be affected by slight changes in the thrust response time. Still, the schedule appears to provide a near optimal solution and certainly an improvement over the baseline schedule.

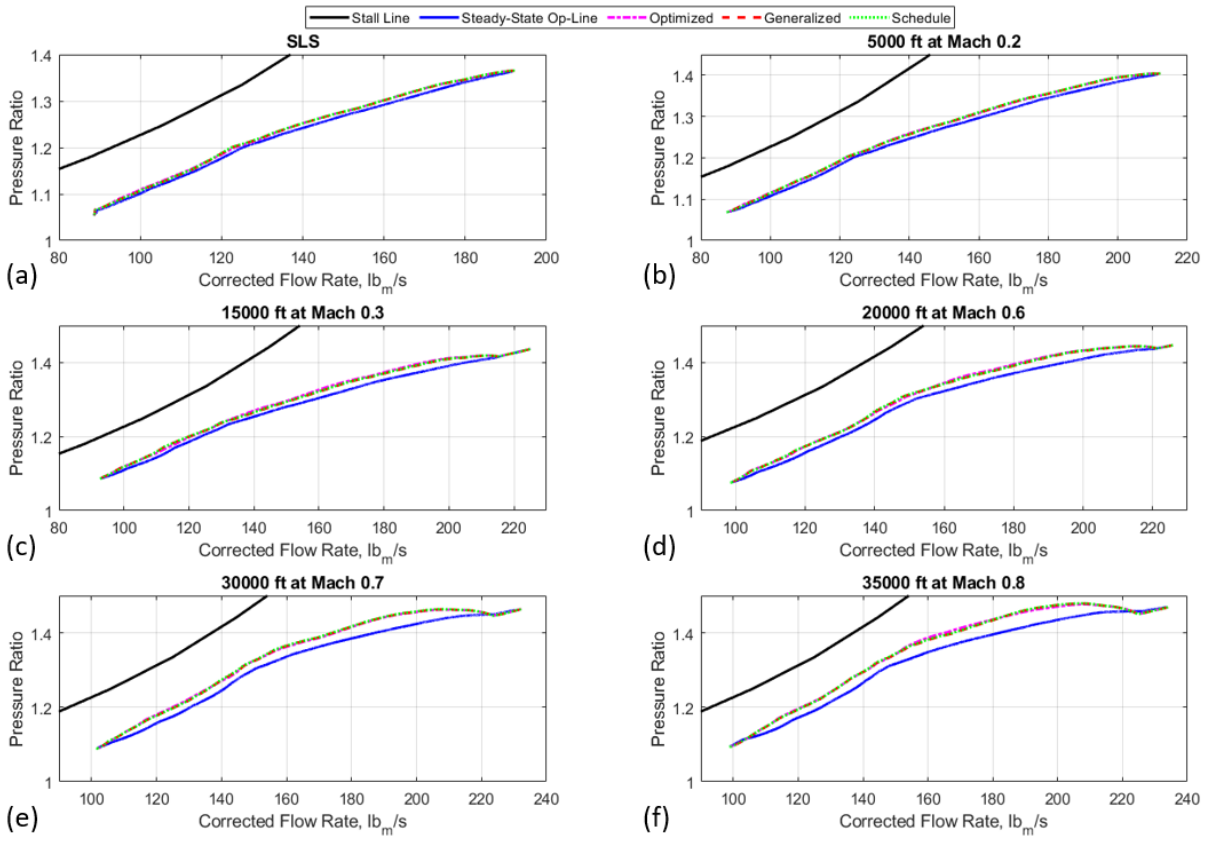

Figure 7. LPC running lines during decelerations transients at various flight conditions

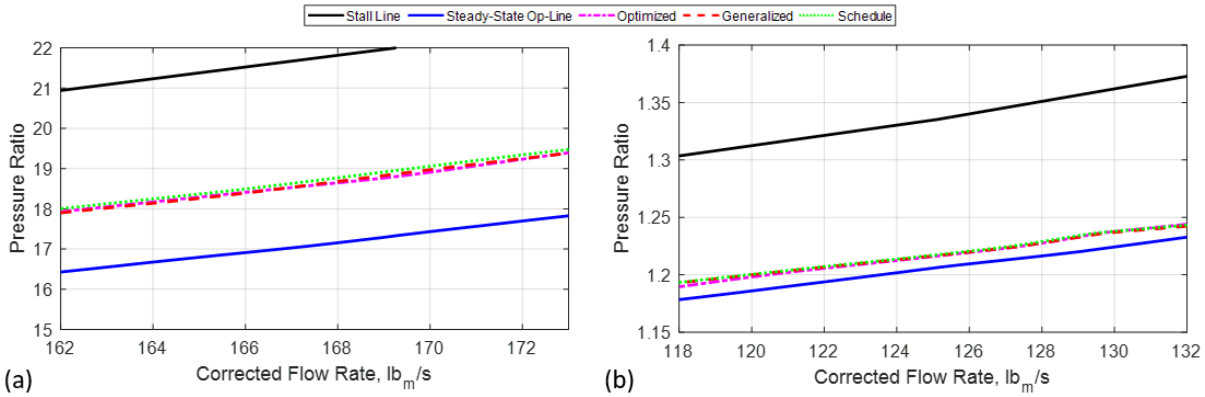

Figure 8. Zoomed in plots of the transient running lines at 15,000 ft and Mach 0.3. (a) shows the HPC data for an acceleration and (b) shows the LPC data for a deceleration.

The next set of simulations demonstrate the impact of the acceleration limit schedule on peak temperatures, which can impact the lifespan and maintenance costs of engine components. These simulations consider an acceleration transient conducted at SLS conditions for a full power rapid burst take-off and a de-rated take-off in which the aircraft only needs ~80% of the maximum thrust. The first two rows of Table 3 compare the peak turbine inlet temperature, $T_{4,peak}$, and T_4 overshoot for the baseline control schedule and a schedule optimized for operability. While the new optimized schedule improves transient operability, peak T_4 is higher. To strike a better balance between operability and peak T_4, the optimization can be redone with a modified fitness function that penalizes T_4 overshoot. An optimization was conducted with the fitness function in Eq. (6).

$$f = \frac{3}{TSU} + \frac{10}{\left(T_{4,peak} - T_{4,SS}\right)/T_{4,SS} + 1} \tag{6}$$

$T_{4,SS}$ is the final steady-state turbine inlet temperature after the acceleration. The algorithm was also modified with a root solver to stretch or contract the fuel flow input profile to achieve the desired response time, and therefore the response time term such as the one used in Eq. (2) becomes unnecessary. Using this fitness function, a new optimized

Table 1. Table of t_r and *TSU* for accelerations

Flight Condition	Baseline Controller		Optimized w_f Input		Generalized w_f Input		Derived RU Schedule	
	t_r, s	*TSU*, %	t_r, s	*TSU*, %	t_r, s	*TSU*, %	t_r, s	*TSU*, %
SLS	4.98	38.4	4.96	20.5	4.98	23.0	4.93	25.8
5000ft, Mach 0.2	5.31	38.7	5.31	24.3	5.31	25.1	5.34	27.3
15000ft, Mach 0.3	5.50	39.6	5.50	33.5	5.50	33.8	5.56	35.4
20000 ft, Mach 0.6	6.01	39.5	6.01	33.0	6.01	33.0	6.07	35.1
30000 ft, Mach 0.7	8.56	46.7	8.53	32.8	8.56	33.9	8.50	37.4
35000 ft, Mach 0.8	10.17	49.5	10.15	31.6	10.17	39.9	10.09	42.7

Table 2. Table of t_r and *TSU* for decelerations

Flight Condition	Baseline Controller		Optimized w_f Input		Generalized w_f Input		Derived RU Schedule	
	t_r, s	*TSU*, %	t_r, s	*TSU*, %	t_r, s	*TSU*, %	t_r, s	*TSU*, %
SLS	11.61	8.41	10.86	6.67	11.61	8.85	11.10	9.24
5000ft, Mach 0.2	11.13	10.08	11.01	7.56	11.13	8.23	10.68	8.38
15000ft, Mach 0.3	11.02	12.74	10.71	9.99	11.02	15.10	10.63	15.33
20000 ft, Mach 0.6	10.27	15.18	10.17	11.38	10.27	12.61	9.99	13.52
30000 ft, Mach 0.7	10.27	19.56	10.15	15.75	10.27	16.47	10.06	17.17
35000 ft, Mach 0.8	10.11	23.97	10.11	21.69	10.11	21.58	9.97	22.06

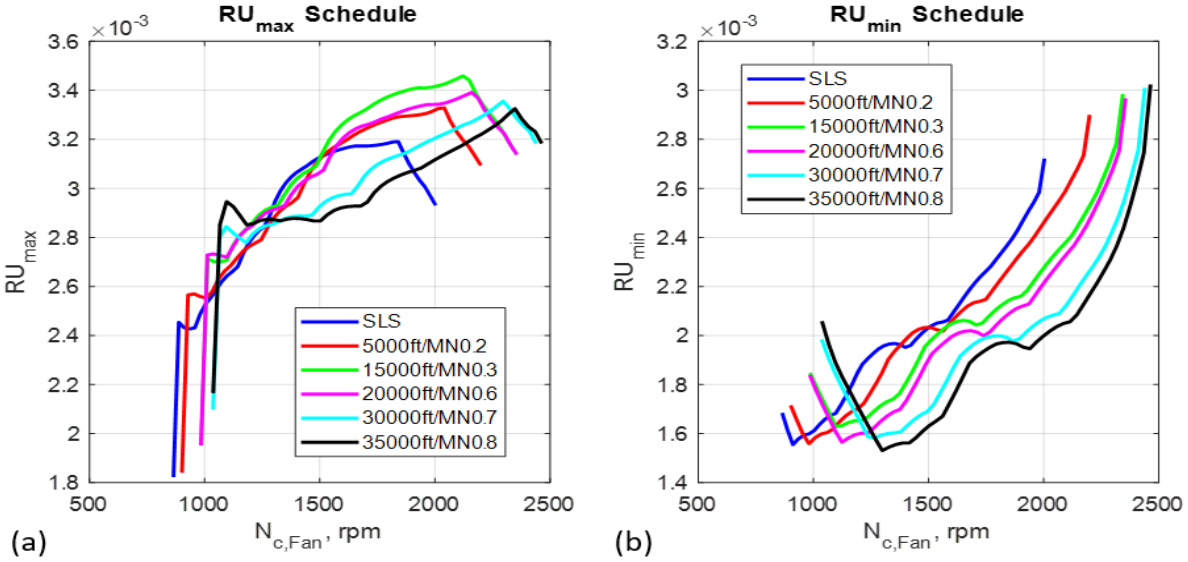

Figure 9. Acceleration (a) and deceleration (b) derived schedules

Table 3. Comparison of transient limit schedules on the basis of *TSU* and peak T_4

Transient Limit Logic	Full Power Burst			Derated Take-Off Burst	
	TSU, %	$T_{4,peak}$, °R	T_4 Overshoot, %	$T_{4,peak}$, °R	T_4 Overshoot, %
Baseline Schedule	38.4	3172	0.1	2951	3.0
Schedule Optimized for Operability	25.8	3181	0.4	2989	4.3
Schedule Optimized with consideration of Operability and T_4	29.76	3182	0.4	2972	3.7

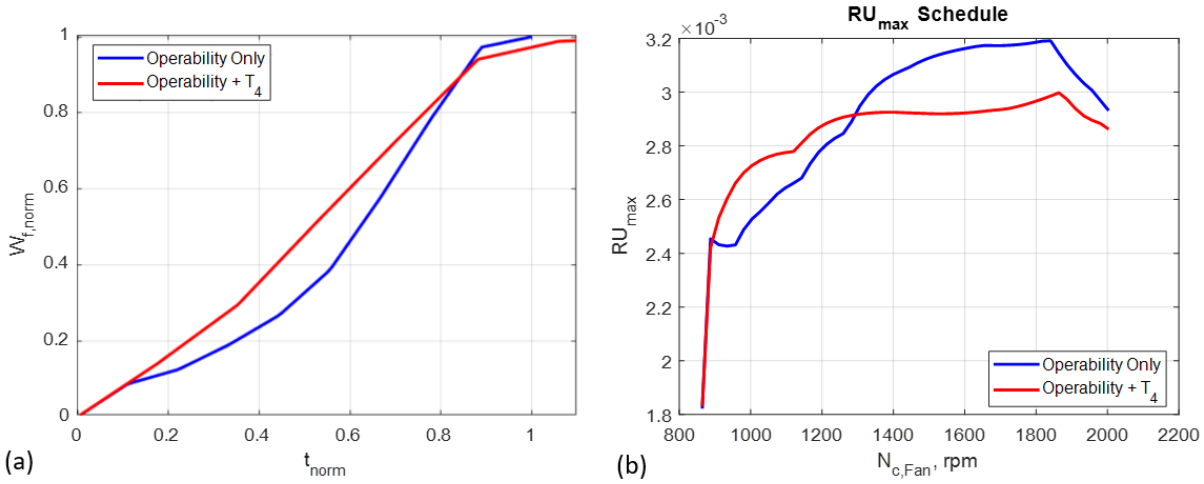

Figure 10. Comparison of optimized fuel flow inputs (a) and acceleration schedules (b)

fuel flow input profile shown in Fig. 10a was obtained for a derated takeoff and a new *RU* limit schedule was extracted. The schedule is plotted in Fig. 10b with reference to the previously optimized schedule. Figure 11 shows the HPC running lines during the acceleration transient. It is evident in these results that the new optimization leads the transient with more fuel flow input to sacrifice some operability such that the maximum fuel flow rate can be approached more gradually to reduce overshoot in T_4. The last row of Table 3 includes the *TSU* and peak T_4 results for the new optimization. During the more common derated takeoff burst scenario, the peak T_4 is reduced by 17°R when compared with the prior optimized schedule. It can also be noted that the while the peak T_4 is still higher than that achieved with the baseline acceleration limit schedule, the transient operability margin is significantly better with a *TSU* of 29.76% vs. 38.4%.

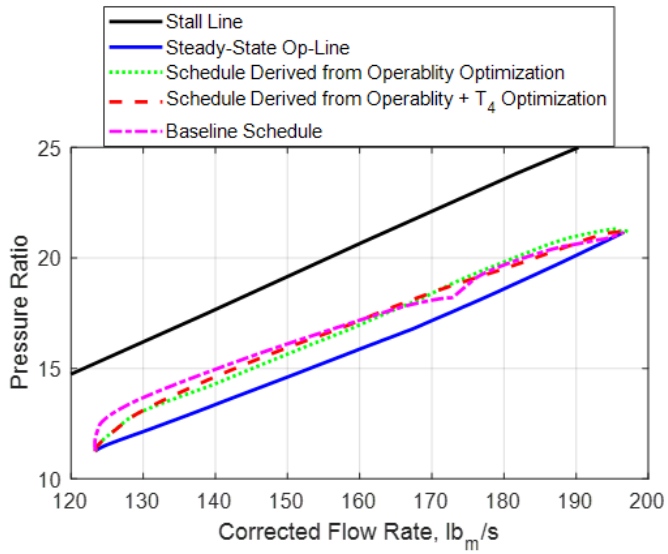

Figure 11. Comparison of the transient running lines of the two optimized acceleration schedules

The conclusion of these analyses is: (1) optimized transient limit logic can have a significant impact on engine operation and influence system design, (2) a generalized or simplified fuel flow profile could be substituted in the control design process to simplify and expedite control design while achieving similar results to the optimal solution, and (3) the optimization objectives can be modified to achieve different goals or to strike a balance between competing goals.

V. Engine Lifespan Optimization

Since the controller is developed to handle engine deterioration, the transient limit logic is overly conservative until the engine is fully deteriorated. This means that the engine will tend to respond slower than it is able. While this may not be a concern for commercial aircraft engines, there are circumstances where better responsiveness is desirable, particularly during emergency scenarios where thrust is needed quickly. Military engine applications would benefit from the ability to improve thrust responsiveness over the lifespan of the engine more so than commercial engines. This goal is in the same spirit of the works referenced prior including Ref. [12], [13], [15], and [18]. In a similar spirit, the example presented in this paper will consider minimizing thrust responsiveness under operability constraints. However, it can be noted that the goals of the lifespan optimization problem formulation could be modified to achieve other goals more relevant to commercial engines. For instance, the objective could be to balance factors that contribute to engine deterioration, such as peak temperatures and pressures, with operability improvements that enable engine design benefits.

MBEC enables the use of closed loop control on unmeasured parameters such as thrust and stall margin. MBEC could also eliminate the need to create extensive schedules for managing acceleration and deceleration. However, the schedule approach considered here is more traditional, and field-tested. Utilizing traditional methods could be argued as being advantageous in the certification process. It is thought that small and gradual updates to the schedule over time, and the ability to revert to the original generic and conservative schedule are favorable qualities with regard to certification and could be viewed as advantageous when compared with advanced MBEC approaches that heavily rely on a model. An optimization scheme with a digital twin will enable the schedule to be reassessed as the engine ages or after changes occur (e.g., performed maintenance and sensed shifts in performance). Coupling this with engine sensor data and a reinforcement learning (RL) strategy could offer a trusted means of maintaining a good balance between transient performance and operability as the engine ages. The RL would leverage the optimization results obtained using the digital twin but would not trust those results. The results would only be used to guide its decisions as it cautiously explores the solution space. The RL algorithm will limit changes to the schedules and penalize any observations of unexpected reductions in operability margin that could be the result of error between the model and physical system. In theory, it could also inform the updating of the digital twin. Figure 12 shows the proposed architecture.

Since engine deterioration occurs gradually, the optimization could occur periodically and may even be conducted off-line with only the results of the optimization being supplied to the control system. The RL algorithm will attempt to adjust the schedule toward the optimal schedule but will limit how much the schedule can change. It will require feedback from the physical system and evaluation of the transient response characteristics prior to allowing further updates. If operability is observed to degrade too much, then the schedule will be shifted back toward the original conservative schedule.

To demonstrate the concept, the genetic algorithm will be employed at four different times during the life span of the engine: new engine (NEW), mid-life engine (MID), three quarters of life engine (3Q), and end-of-life engine

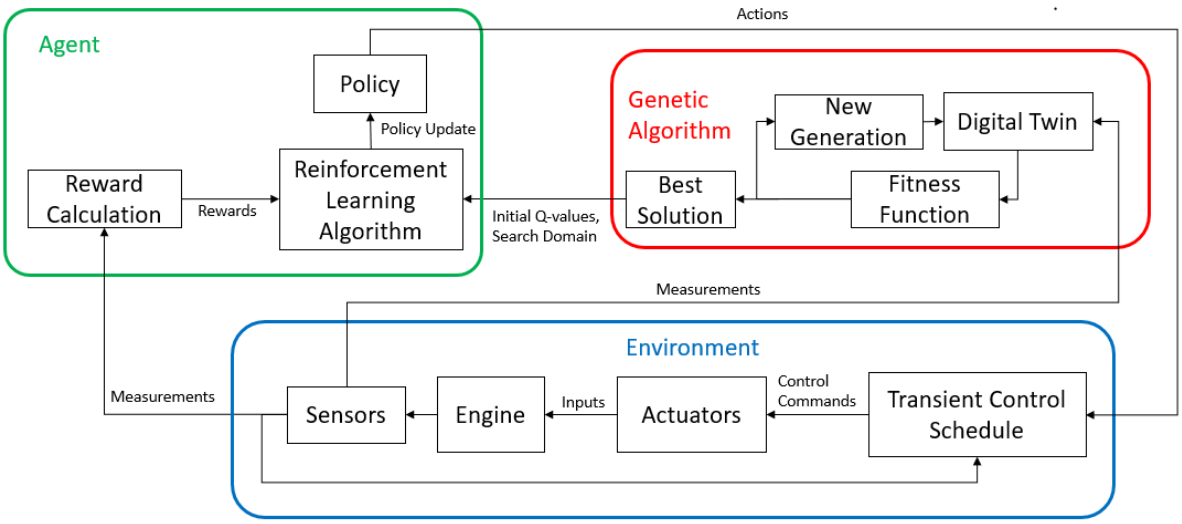

Figure 12. Schematic of the proposed transient control architecture and the iterative process to update it

(EOL). This will be simulated by modifying the health parameters for the turbomachinery components. The degradation states of the engine have been sparsely selected to demonstrate the concept. For the sake of demonstration, all the examples focus on sea level static operation. An RL algorithm is then used to modify the schedule with guidance from the optimization results. The rest of this section is broken into three sub-sections: (1) the RL algorithm, (2) the results of a demonstration, and (3) discussion of some related topics.

A. Reinforcement Learning Algorithm

A Q-learning RL algorithm [32] is utilized to update the schedule. Q-learning uses a quality function, *Q(s,a)*, that describes the value or quality of being in a particular state, *s*, and taking a particular action, *a*. In this case, the state is the enforced limit schedule, and the action is the decision for how to modify it in the future. For demonstration, a discrete set of schedules are initialized and the RL algorithm will have the option to shift the schedule toward the "optimal" transient schedule, retain the same schedule, or shift the schedule toward the original conservative schedule. The quality of the state and action combination is based upon accumulated rewards, *R*, which are a function of the responsiveness and operability of the system. The reward will be the sum of response time and operability rewards (R_r and R_o).

$$R = R_r + R_o$$

$$R_r = \begin{cases} 1 \; if \; t_r - t_{r,old} < -X \\ -0.1 \; if \; \left|t_r - t_{r,old}\right| < X, \\ -3 \; if \; t_r - t_{r,old} > X \end{cases} \quad R_o = \begin{cases} 0 \; if \; \Delta PR \geq 0 \\ 3 \; if \; \Delta PR < 0 \; \& \; (\Delta PR - \Delta PR_{old}) > 0 \\ -5 \; if \; \Delta PR < 0 \; \& \; (\Delta PR - \Delta PR_{old}) \leq 0 \end{cases} \tag{7}$$

$$\Delta PR = min(PR_{max} - PR)$$

The parameter t_r is the response time. *PR* is the pressure ratio and PR_{max} is the maximum allowable pressure ratio given as a function of corrected speed N_c. PR_{max} effectively defines a "do not exceed" line on the compressor map. The subscript "old" refers to quantities resulting from the prior action while variables without the subscript refers the current state and action pair. *X* is a threshold value below which the change in thrust response time is considered negligible. In the presented example $X = 0$, which takes advantage of the idealistic simulation conditions under which the RL is applied. However, it should be noted that variations and sources of uncertainty will be present in real world applications. For that reason, *X* would need to be non-zero. The idea is simple: reward reductions in response time, penalize increases in response time, and penalize reductions in operability beyond the prescribed limit. The second R_o condition shows that if the operability limit is in violation (i.e., $\Delta PR < 0$), an improvement in operability (i.e., $(\Delta PR - \Delta PR_{old}) > 0$) is rewarded. To encourage the search for better solutions, a slight penalty is incurred by staying at the same schedule or achieving the same thrust response time. The relative magnitude of the values chosen in Eq. (7) should be considered to achieve efficient learning.

It is acknowledged that quantifying compressor operability with measurement data remains an open question. Ideally, this would be done without any reliance on a model. Ref. [12] proposes an approach to adapting acceleration schedules based on the combustor discharge gas temperature or an estimate of it. In any case, it is outside of the focused intent of this paper. For demonstration, the maximum pressure ratio constraint fills this role. The compressor pressure ratio will be measured via sensors, as will the corrected HPC speed. While the measurement is feasible, the practical viability of the approach is unknown and beyond the scope of this paper.

In RL terminology, a policy is enacted by an agent. The agent gathers information about the environment and the impact of its actions and uses that information to update the quality function and the policy used to determine future actions. The quality function will be initialized and later updated as the agent applies actions and observes the rewards of those actions. Equation (8) is used to update the quality function at state *s* with action *a*.

$$Q(s,a) = (1-\alpha)Q_{old}(s,a) + \alpha\left(R + \gamma \max_{a'}(Q(s',a'))\right) \tag{8}$$

Q_{old} is the quality function value from the previous update, *R* is the reward for the state and action pair, α is the learning rate, and γ is the discount factor. The learning rate is a value between 0 and 1 with higher values encouraging faster adoption of new data. Since the measured feedback data from the actual engine is assumed to be more accurate than the digital twin, the learning rate will be set to a relatively high value of 0.9. The discount factor is a value between 0 and 1 with lower values favoring immediate rewards over long-term rewards. In the demonstration presented in the next sub-section, γ was set to 0.7. The term multiplied by the discount factor is an estimate of the optimal future value.

The quality function value for each state and action combination can be initialized using information from the genetic algorithm optimization.

A design consideration with any RL application is the trade-off between exploration and exploitation. Exploration refers to the tendency of the policy to explore new state and action pairs in search for a better solution, and exploitation is the tendency of the policy to use known information to maximize reward. Assuming the optimization is decent, initial quality approximations are relatively trustworthy and therefore the RL algorithm can favor exploitation. The epsilon greedy strategy [32] is employed for choosing which action to take. The parameter ε will be set at a value between 0 and 1. A random number generator will output a number between 0 and 1 and if the number is less than ε then the exploration method will be implemented. Otherwise, the exploitation method will be implemented. The exploration method will randomly choose which action to take while the exploitation option will choose the action with highest quality value. For the demonstration presented in the next sub-section, ε was chosen to be 0.05.

The nature of this application allows for some simplifications due to the limited number of action options and the ability to assume trends between action and impact. For instance, a more aggressive acceleration schedule will lead to a faster response time and reduced operability while a less aggressive schedule will lead to a slower response time and improved operability. When the quality function is updated, there are instances when the quality adjustment for other state and action pairs can be inferred. For example, if the schedule was shifted to be more aggressive and the operability was reduced such that *R* was significantly reduced, it can be assumed that any action to make the schedule more aggressive will make the issue worse and will result in more negative rewards. Therefore, the quality function can be updated to discourage such an action.

B. Demonstration

Optimizations were conducted for the NEW, MID, 3Q, and EOL engine. The optimization for the EOL utilized the fitness function provided by Eq. (2). These results were used to create the conservative schedule at which the limit controller will be initialized and will default to if the transient operability were to become a concern. The EOL results were also used to define a "do not exceed" HPC *PR* as a function of HPC corrected speed. This limit was set slightly above the EOL transient running line. Optimizations for the 3Q, MID, and NEW engine sought to minimize the response time while not exceeding the maximum *PR* limit. An iterative root solver was used to just meet the PR limit and the optimizer used the fitness function defined below:

$$f = \frac{1}{t_r} \tag{9}$$

Results from the optimizations were used to create an RU_{max} schedule for each degradation level. The schedules are plotted in Fig. 13a while the transient running lines with the optimized fuel flow input are plotted in Fig. 13b along with a representation of the maximum HPC *PR* limit. It is noted that the normalized fuel flow input profiles have similar shape but do vary significantly when compared to each other in Fig. 14. A newer engine can lead more

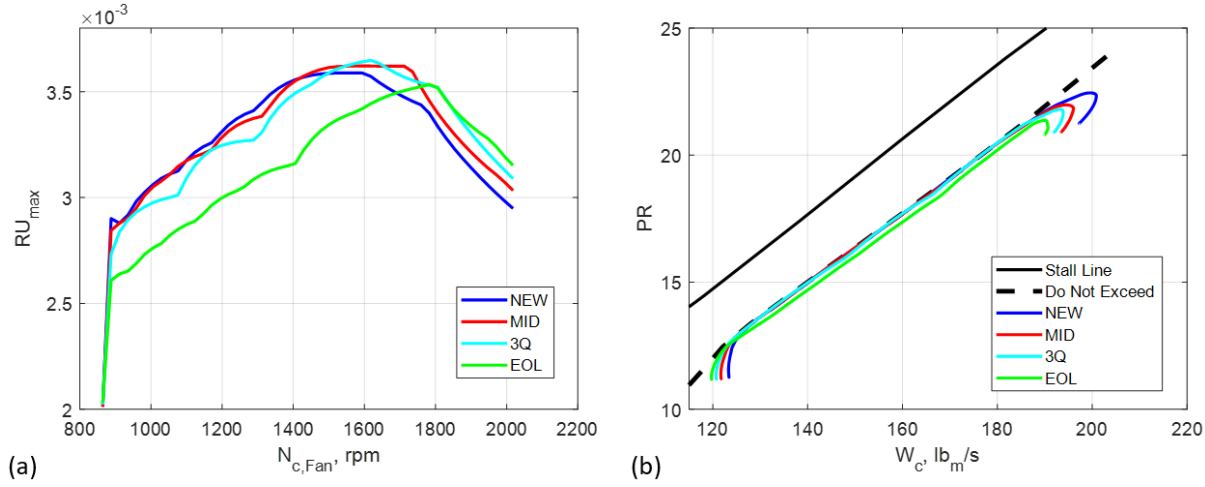

Figure 13. Schedules derived from the optimization results (a) and transient running lines with the optimized fuel flow inputs (b)

aggressively with fuel flow input. However, these profiles appear to collapse to a similar normalized fuel flow input profile if they are stretched or contracted. To demonstrate this Fig. 14 also shows the NEW, MID, and 3Q profiles stretched to better match the EOL profile. The profiles are stretched by factors of 1.2 and 1.15, and 1.1 respectively. Therefore, it is noted that near optimal results are expected by fixing a normalized fuel flow input profile which can be stretched or contracted to just meet the desired operability constraint. This approach would avoid the need to perform additional optimizations.

The RL algorithm was applied assuming the digital twin is an exact match to the physical system. Figure 15 shows the possible acceleration schedules as the controller attempts to adjust from the original conservative schedule to a schedule that minimizes response time for a new engine. Figure 16a shows the progression of the acceleration schedule, and the response time after each transient. Figure 16b shows the running lines for the first, last, and most severe transients during the training period. Since the implementation of the schedule does not perfectly match the optimal results used to derive it, the operability limit is hit prior to reaching the most aggressive schedule (schedule 10) and instead settles on schedule 7, which just meets the operability limit. The thrust response time was reduced from 3.82s to 3.37s, an 11.8% reduction.

Next, the health state of the engine was set to a mid-life state to simulate aging of the engine without updating the optimal schedule solution. This scenario illustrates how the RL algorithm will adapt the schedule to assure adequate operability margin. This scenario could also be viewed as representing a mismatch in the actual system and the digital twin, for which the digital twin is optimistic about the engine's health state. The results are shown in Fig. 17. The RL agent shifted from using schedule 7 to schedule 6 to meet the operability constraint while increasing the thrust response

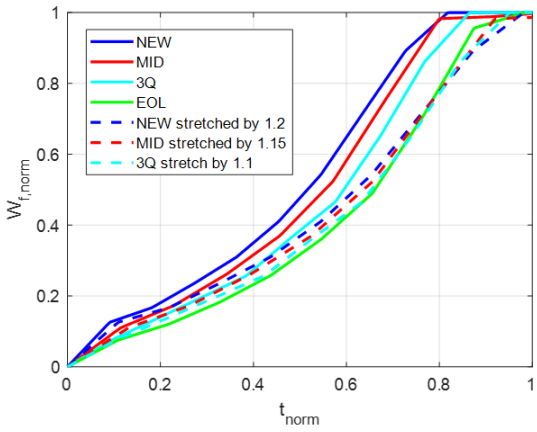

Figure 14. Normalized fuel flow command profiles from the optimizations

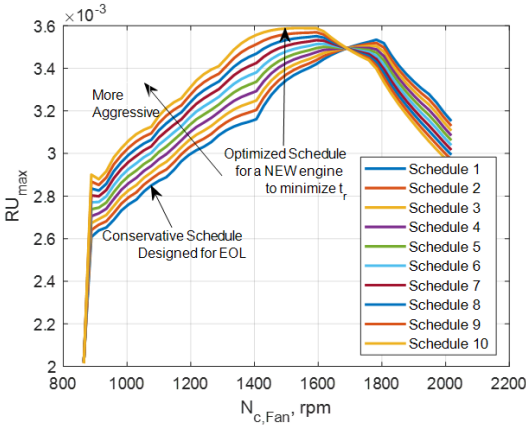

Figure 15. Acceleration schedule options for the RL agent

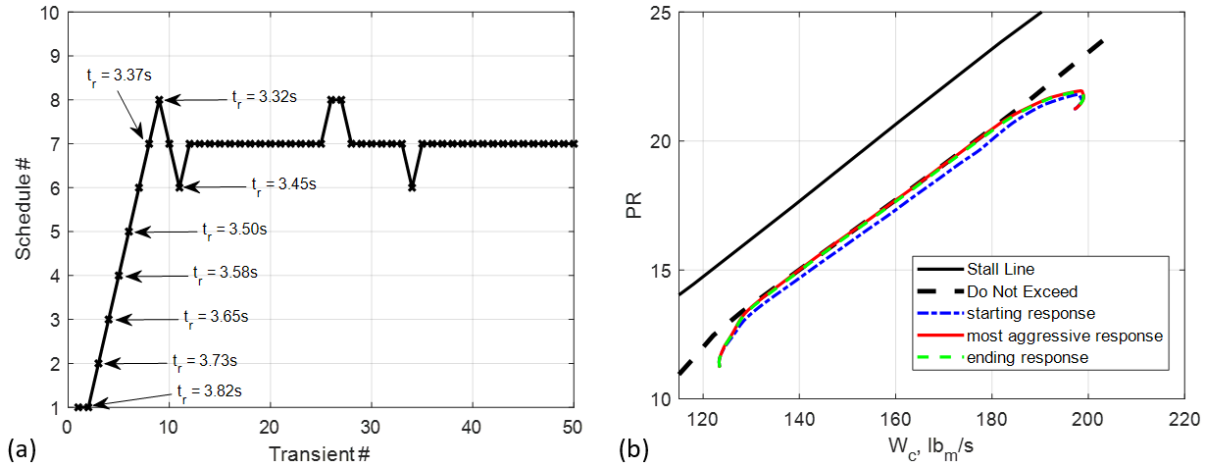

Figure 16. Training of the RL agent from an initialized conservative acceleration schedule to a more aggressive schedule that minimizes response time. (a) shows the schedule selection for each sequential transient and (b) shows the transient running lines

time from 3.68 s to 3.76 s. These results demonstrate the ability of the RL algorithm to learn that the existing schedule is too aggressive and needs to be shifted toward the more conservative original schedule.

The agent can learn and shift the schedule indefinitely as the engine ages, but some additional benefit could be possible if the "optimal" schedule were updated, and the schedule options were refined. To demonstrate this, the engine health state was changed to 3Q and the option of acceleration schedules was refined to exist between the optimal 3Q schedule and the conservative EOL schedule. The Q values were reset, and the schedule was reinitialized on the conservative side of the prior used schedule. The RL agent was observed to shift toward the optimal schedule and settle around the schedule that just meets the imposed operability limit. For comparison, another set of simulations were conducted to observe results that would occur without making these updates. This resulted in shifting from the original schedule 6 to the original schedule 5. It was observed that refinement in the schedule options resulted in a thrust response time reduction of only 0.02 s, which suggests that performing optimizations and deriving new optimal schedules over the lifespan of the engine is unnecessary.

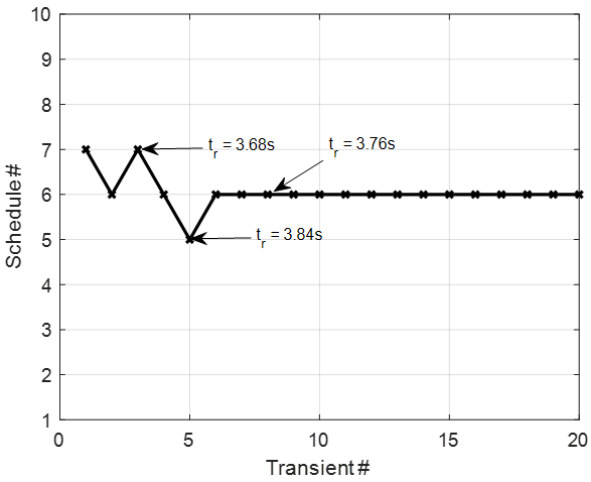

Figure 17. Actions of the RL agent when the engine help state is updated to a MID engine

The results demonstrate the ability of the RL algorithm to minimize acceleration time while respecting operability limits. The RL was also able to slow the response down and prevent the schedule from becoming more aggressive when operability measures indicated an undesirable decrease in operability margin due to engine degradation.

C. Discussion

Results have suggested that a near optimal normalized fuel flow input profile can be used throughout the engine lifespan. Thus, the need to use the genetic algorithm or some other optimization scheme throughout the lifespan of the engine can be reduced or eliminated. However, if the genetic algorithm were employed as the engine ages, optimizer settings could be modified to strike a good balance between exploration and exploitation. For instance, at the beginning of the life of the engine when the controller is representative of the fleet of engines rather than the specific engine, or after maintenance and sensed anomalies, exploration is more favorable. Exploitation is more favorable as the engine slowly degrades.

A benefit to using the RU limit approach over the $\dot{N}$ limiter approach is that it does not require the design of a closed loop controller. However, unlike an $\dot{N}$ limit schedule, an RU limit schedule will not produce a consistent response time as the engine ages. One benefit of updating the RU limit schedule is that it could be used to maintain similar thrust response times for engines on the same aircraft, thus minimizing undesirable asymmetric thrust. This could be done by modifying the acceleration schedule of the faster engine(s) to match the responsiveness of the slowest engine. The acceleration schedule of the slowest engine could be optimized for responsiveness while applying operability constraints, while the operability margin of the faster engine(s) is optimized using thrust responsiveness as a constraint.

To implement the RL method presented here in any practical sense, additional factors will need to be considered. Among them will be strategies for considering variations in operation and secondary effects such as power extraction and heat soak. If the RL were to remain active while the engine is operating in the field, the agent would need to be able to recognize when a transient is occurring and how to characterize it, using sensor feedback to guide the learning process. A variety of factors could make each transient unique including the manner in which the throttle position is moved, the flight condition, the thermal state of the engine components, etc. These factors could become part of the state and accounted for in the quality function, thus allowing the schedule to adapt to these conditions. An alternative could be to build some conservativeness into the operability limit to ensure worst case transient scenarios are accommodated. Another approach could be to put the engine through a "training" period at various intervals within its lifespan so that the agent can adjust the schedule under controlled conditions.

VI. Summary

To optimize engine performance, transient operability must be considered. The proper combination of responsiveness, operability, and efficiency should be sought. To do this, optimization techniques can be applied. A genetic algorithm has been employed to determine the optimal fuel flow input for extreme transient scenarios, and the data has been used to create transient limit schedules. Results have demonstrated significant improvements in operability margin over the original transient limit schedules. An example of this includes a 31% reduction in the amount of operability stack utilized during a snap acceleration at sea level static conditions. Studies have demonstrated, with some success, how the form of the fuel flow input profile can be generalized across various flight conditions to achieve near optimal results. Furthermore, it has been demonstrated how other objectives, such as the reduction of peak turbine inlet temperature, can be considered in the optimization to strike a balance between competing goals. Finally, a method has been proposed and demonstrated for updating engine acceleration limit logic to minimize response time under operability constraints. The method employs reinforcement learning techniques that leverage a digital twin to modify the acceleration schedule while protecting the engine by limiting the changes in the schedule and utilizing sensor feedback to assess the quality of each schedule modification and each action taken to adjust the schedule. A simple application illustrated the ability of the approach to (1) efficiently find the optimal acceleration schedule that minimized thrust response time while respecting operability constraints, and (2) to adapt to changes in engine operation due to component aging. The example illustrated a reduction in thrust response time of ~11.8% compared to a schedule designed conservatively for an end-of-life engine. Results from the example also suggest that it is unnecessary to update the optimal schedule as the engine ages, thus reducing the workload for implementation. Topics for future investigation could include attempts to address the various challenges identified for practical implementation of the reinforced learning approach for updating transient limit schedules. In addition, the approach could be modified to achieve goals more aligned with commercial engine applications.

Acknowledgments

The author would like to acknowledge the Transformational Tools and Technologies (TTT) project under the NASA Aeronautics Research Mission Directorate (ARMD) that has supported this work.

References

[1] Jaw, L., Mattingly, J., *Aircraft Engine Controls: Design, System Analysis, and Health Monitoring*, Washington D.C., USA: AIAA, 2009.

[2] Federal Aviation Administration, "Title 14 of the Code of Federal Regulations", https://www.ecfr.gov/current/title-14/chapter-I/subchapter-C/part-33, accessed May, 2022.

[3] Garg, S., "Aircraft Turbine Engine Control Research at NASA Glenn Research Center," ASCE Journal of Aerospace Engineering, Vol. 26, No.2, 2013.

[4] Garone, E., Di Cairano, S., Kolmanovsky, I., "Reference and command governors for systems with constraints: A survey on theory and applications," Automatica, pp. 75:306–28, 2017.

[5] Montazeri-Gh, M., Rasti, A., Jafari, A., and Ehteshami, M., "Design and implementation of MPC for turbofan engine control system," Aerospace Science & Technology, Vol. 92, 2019.

[6] Richter, H., Singaraju, A., and Litt, J., "Multiplexed Predictive Control of a Large Commercial Turbofan Engine," Journal of Guidance, Control, and Dynamics, 31(2) pp. 273-281. 2008.

[7] Xu, W., Huang, J., Qin, J., and Pan, M., "Limit protection design in turbofan engine acceleration control based on scheduling command governor," Chinese Journal of Aeronautics, 34(10) pp. 67-80. 2021.

[8] Philpot, M., "Practical Consideration In Designing the Engine Cycle," Defense Technical Information Center, 1992.

[9] Spakovszky, Z., "Instabilities Everywhere! Hard Problems in Aero-Engines," GT2021-60864, ASME Turbo Expo, June, 2021.

[10] Ghisu, T., Parks, G., Jarrett, J., and Clarkson, P., "Robust Design Optimization of Gas Turbine Compression Systems," 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia Canada, September, 2008.

[11] Csank, J., and Zinnecker, A., "Application of the Tool for Turbine Engine Closed-loop Transient Analysis (TTECTrA) for Dynamic Systems Analysis," AIAA Propulsion and Energy Forum, July, 2014.

[12] Meisner, R., Goodrich Pump & Engine Control Systems, Inc., West Hartford, CT, U.S. Patent Application for a "Adaptive Acceleration Schedules for Gas Turbine Engine Control Systems," Appl. No. 10/295,571, US 6,820,429, filed 15 Nov. 2002.
[13] Smith, J., United Technologies Corporation, Hartford, CT, European Patent Application for a "Acceleration control for a gas turbine engine with duct pressure loss compensation," Appl. No. 90630111.4, EP 0 401 152 A2, filed on 5, May, 1990.
[14] Xiangxing, K., Xi, W., Daoliang, T., Ai., H., and Yue, L., "An extrapolation approach for aeroengine's transient control law design," Chinese Journal of Aeronautics, 26(5): pp. 1106–1113, 2013.
[15] Yu-chun, C., Si-Yuan, X., Yuan-Ha, C., and Qui-Ye, T., "Virtual Power Extraction Method of Designing Acceleration and Deceleration Control Law of Turbofan," 45th AIAA Joint AIAA/ ASME/SAE/ASEE Joint Propulsion Conference, Denver, CO, August, 2009.
[16] Yu, B., Cao, C., Shu, W., and Hu, Z., "A New Method for the Design of Optimal Control in the Transient State of a Gas Turbine Engine," IEEE Access Multidisciplinary Journal, Vol. 5, 2017.
[17] Montazeri-Gh, M., Jafari, S., "Evolutionary Optimization for Gain Tuning of Jet Engine Min-Max Fuel Controller," AIAA Journal of Propulsion & Power, Vol. 27, No. 5, September, 2011.
[18] Li, J., Fan, D., and Sreeram, V., "Optimization of Aero Engine Acceleration Control in Combat State Based on Genetic Algorithms," International Journal of Turbo Jet-Engines, Vol. 22, pp. 29-36, 2012.
[19] AIAA Digital Engineering Integration Committee, "Digital Twin: Definition & Value. An AIAA and AIA Position Paper", December, 2020.
[20] Badea, V., Zamfiroiu, A., Boncea, R., "Big Data in the Aerospace Industry," Informatica Economică, Vol. 22, No. 1, 2018.
[21] Panov, V., "Auto-Tuning for Real-Time Dynamic Gas Turbine Models", ASME Turbo Expo, Dusseldorf, Germany, GT2014-25606, June, 2014.
[22] Adibhatla, S., and Gastineau, Z., "Tracking Filter Selection and Control Mode Selection for Model Based Control," 30th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Indianapolis, IN, June, 1994.
[23] Adibhatla, S., and Lewis, T., "Model-Based Intelligent Digital Engine Control (MoBIDEC)," 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July, 1997.
[24] Fuller, J., Kumar, A., and Millar, R., "Adaptive Model Based Control of Aircraft Propulsion Systems: Status and Outlook for Naval Aviation Applications," ASME Turbo Expo, Barcelona, Spain, June, 2006.
[25] Kapteyn, M., Knezevic, D., Willcox, K., "Toward predictive digital twins via component-based reduced-order models and interpretable machine learning", AIAA SciTech Forum, Orlando, FL, January, 2020.
[26] Mascellino, A., "GE Digital Introduces a Digital Solution to Autonomously Tune Gas Turbines," Control Automation, https://control.com/news/ge-digital-introduces-a-digital-twin-solution-to-autonomously-tune-gas-turbines/, February 9, 2022.
[27] Chapman, J., Litt, J., "Control Design for an Advanced Geared Turbofan Engine," AIAA Joint AIAA/ ASME/SAE/ASEE Joint Propulsion Conference, Atlanta, GA, 2017.
[28] Chapman, J., Lavelle, T., May, R., Litt, J., and Guo, T.-H., "Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) User's Guide," NASA TM-2014-216638, 2014.
[29] May, R.D., Csank, J., Lavelle, T.M., Litt, J. S., and Guo, T.-H., "A High-Fidelity Simulation of a Generic Commercial Aircraft Engine and Controller," Proceedings of the 46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, AIAA-2010- 6630, Nashville, TN, July 2010.
[30] Brunton, S., and Kutz, N., *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, Cambridge, United Kingdom, Cambridge University Press, 2019.
[31] Newman, M., "Power laws, Pareto distributions and Zipf's law," Contemporary Physics. 46 (5): pp. 323–351.
[32] Sutton, R., and Barto, A., *Reinforcement learning: An Introduction*, Cambridge, MA, MIT Press, 2014.