Investigation of a Verification and Validation Tool with a Turbofan Aircraft Engine Application

Peter Uth [*] and Anshu Narang-Siddarth [†]

University of Washington, Seattle, WA, 98195

Edmond Wong [‡]

NASA Glenn Research Center, Cleveland, OH, 44135

The development of new capabilities in system intelligence and autonomy will require new means of verification and validation (V&V) to ensure safety and performance requirements are satisfied. As a step towards this goal, CoCoSim, a publicly available V&V tool, is being developed to analyze the validity of user-defined assertions on MATLAB/Simulink models. This paper demonstrates CoCoSim's capabilities by applying it to C-MAPSS40k, a 40,000 lbf class turbofan engine model developed at NASA for testing new control algorithms. Due to the current limitations of the CoCoSim V&V software, several modifications are made to C-MAPSS40k to achieve compatibility with CoCoSim. Some of these modifications sacrifice fidelity of the original model, but the analysis is still useful even with that limitation. Several safety and performance requirements typical of turbofan engines are identified and constructed into a series of assertions to be tested as part of the V&V framework. Preliminary results for these requirements using C-MAPSS40k's industry standard turbofan engine controller are presented. While CoCoSim's capabilities are demonstrated, a truly comprehensive analysis will require further development of the tool.

Nomenclature

EPR	Engine pressure ratio
N_c	Core speed (RPM)
N_f	Fan speed (RPM)
PLA	Power lever angle (deg)
P_2	Engine inlet pressure (psi)
P_{50}	Engine exit pressure (psi)
P_a	Ambient pressure (psi)
$Ps3$	High pressure compressor discharge static pressure (psi)
T_2	Engine inlet temperature (°R)
T_3	Compressor temperature (°R)
T_{50}	Engine exit temperature (°R)
VBV	Variable bleed valve
VSV	Variable stator vane
W_f	Fuel flow rate (lb-m/sec)

[*]Graduate Research Assistant, Advanced Dynamics, Validation & Control Research Laboratory, William E. Boeing Department of Aeronautics & Astronautics, AIAA Student Member.

[†]Assistant Professor and Director of the Advanced Dynamics, Validation & Control Research Laboratory, William E. Boeing Department of Aeronautics & Astronautics, AIAA Member.

[‡]Research Engineer, Intelligent Control & Autonomy Branch, AIAA Senior Member.

I. Introduction

Development of increasingly intelligent and autonomous systems has led to a rapid increase of software complexity for cyber physical systems. The need for verification and validation (V&V) schemes that can provide safety and performance assurances for these types of systems has become critical to their acceptance and use. In response to this need, new means of V&V are being developed with the goal of ensuring reliable operation of increasingly complex systems. Among emerging V&V tools is CoCoSim [1], a publicly available automated analysis framework that can use an underlying model checker to verify safety properties of MATLAB/Simulink models. It has been successfully demonstrated on several subsystems of the NASA transport class model (TCM) [2], which is based on a typical twin engine commercial aircraft. CoCoSim currently only supports a subsection of the Simulink library, with support for additional Simulink blocks introduced as the tool undergoes further development. This restricts the direct use of CoCoSim on models containing currently unsupported blocks. However, there are enough supported blocks in this paper's application for an effective study.

Turbofan aircraft engine performance deteriorates over the engine's lifetime. The current standard for engine control is to design the controller conservatively, ensuring that the engine can provide the required thrust response over the entire course of its life [3]. However, this approach results in a performance tradeoff where the engine is inhibited by the controller before the end-of-life condition. More advanced control architectures are being developed that could unlock benefits in fuel efficiency and performance throughout the life of the engine. For example, NASA is developing a model-based engine controller that can target specific stall margins, thus preventing the overly conservative margins that are typical for younger engines using a standard controller [3]. However, more advanced controllers will require new V&V techniques to verify their safety and performance qualities before they can be adopted.

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS40k) [4] is a Simulink model of a typical 40,000 lbf thrust commercial turbofan aircraft engine. The model was developed by NASA for the purpose of testing new control algorithms. It contains a nonlinear engine model and a baseline digital controller that is representative of industry standards. Two control modes are available within this baseline, giving users the option between a fan speed (N_f) and an engine pressure ratio (EPR) controller. Previous development of control algorithms using the C-MAPSS40k model verified safety properties by running simulations for various operating conditions such as in [3,5]. However, this approach provides only a low level of confidence for the safe use of new controllers since simulating every possible operating condition becomes intractable as system complexity grows.

The contribution of this paper is the application of the CoCoSim tool to the C-MAPSS40k engine model to assess CoCoSim's capabilities for performing V&V on a realistic engine propulsion system. Integration challenges with the tool and model are discussed, since the C-MAPSS40k model contains several Simulink blocks currently unsupported by CoCoSim that require conversion to CoCoSim-compatible elements to enable the analysis. Typical turbofan engine safety and performance criteria are identified for verification, and some preliminary V&V results using the baseline controller are produced. Concepts presented could be applied to other system models, and alternate control designs can be used in lieu of the baseline. Results and current limitations of the software are discussed, and further areas of development are identified towards the goal of a fully autonomous, advanced V&V scheme for modern systems.

II. Verification Software

CoCoSim [1] is a verification tool that can check properties of top level Simulink models using an underlying model checker. Given an existing Simulink model, an "observer" [6, 7] is added to the system using the installed CoCoSim menu. All of the inputs of the system, identified by Simulink inport blocks, must be routed as inputs to the observer. System outputs, identified by Simulink outport blocks, can be routed as inputs to the observer as desired by the user. Any properties to be verified are then constructed as assertions within the observer subsystem block. Assertions are statements that must always be true for a system to be valid. For example, the assertion $x > 0$ is always true for the system $x = 1$. When the verification process is initiated, CoCoSim varies the system inputs, observes the system outputs, and checks the validity of the assertions within the observer. The output of CoCoSim is "valid" or "invalid." Valid implies that the assertions within the observer are satisfied for all input values. Invalid implies that input values were found that falsify the assertions. If the solution is invalid, the counterexample is displayed.

The internal process between initiation and verification output is as follows. First, the system model and the observer's assertions are compiled into Lustre [8], a synchronous data flow language well suited for verification. The compiled Lustre code is then passed to a back-end solver, which performs the verification. CoCoSim is compatible with several back-end solvers. JKind [9–11], an infinite-state model checker for safety properties, was chosen for the work described in this paper for its compatibility with the Windows operating system. JKind verifies Lustre code using k-induction [12] and property directed reachability [13–15] via a satisfiability modulo theories (SMT) solver [16]. The analysis in this paper uses the Z3 SMT solver [17] for its state-of-the-art efficiency. If JKind determines that a safety property is valid, that property is guaranteed to be true for all system operating points. If a property is found to be invalid, the falsified property is reported along with an explicit counterexample demonstrating the violation.

A simple example of the CoCoSim verification setup in Simulink is shown in Figure 1. The model being verified is seen with an observer block attached to the two inputs and two outputs of the system. The interior of the observer is also shown. The assertion within the observer states that for any two input values $In1$ and $In2$,

$$Out1 \geq In1 \quad OR \quad Out1 \geq In2. \tag{1}$$

Given that the system defines $Out1$ as equal to the larger of the two inputs, this case was determined to be valid since $Out1$ will always be greater than or equal to at least one of the inputs. The green color exhibited by the observer in Simulink indicates a valid, or safe, solution. Had the solution been invalid or unsafe, the color of the observer would have exhibited a red color.

The CoCoSim development team has previously used CoCoSim to verify certain component properties of the NASA transport class model (TCM) [2]. This was done by creating stand-alone models for the TCM subsystems of interest with property observers. For example, a stand-alone altitude controller model was created with its own observer. The property to be verified, which stated that "the guidance shall be capable of climbing at a defined rate, to be limited by minimum and maximum engine performance," was taken from [18] and constructed as an assertion within this observer.

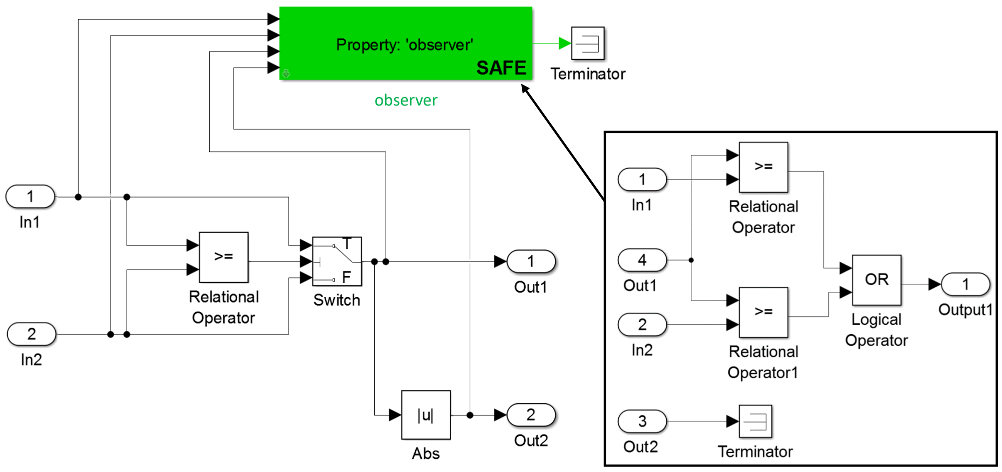


Figure 1. Simple example of the verification setup.

III. Turbofan Engine and Controller Model

C-MAPSS40k [4] is a 40,000 lbf class turbofan engine Simulink model based on dynamic flight test data of a highly instrumented engine during steady state and transient maneuvers. The model's main purpose is to provide a detailed component-level engine model that can be used as a test bed for control algorithms. The main components of the C-MAPSS40k model are an open-loop engine and a baseline controller that is representative of industry standards [5]. Figure 2 shows a schematic of the C-MAPSS40k engine. Engine components are modeled in C-code and accessed via Simulink S-functions, which allows for fast execution. Sensors available for turbofan engine control architectures include core speed (N_c), fan speed (N_f), compressor pressure ($Ps3$), compressor temperature (T_3), inlet pressure (P_2), inlet temperature (T_2), exit pressure (P_{50}), and exit temperature (T_{50}). The baseline controller uses either the engine pressure ratio (EPR $= P_{50}/P_2$) or N_f as the control target. These values are commonly used since they can be measured with typical sensors, unlike variables such as engine thrust or stall margin. Fuel flow rate (W_f) command is the main control input, and control options also exist for a variable bleed valve (VBV) and a variable stator vane (VSV). The baseline controller contains protection logic that limits the W_f command for safety purposes.

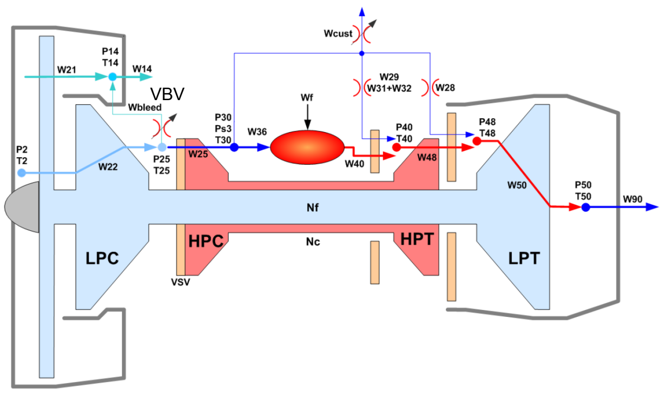


Figure 2. C-MAPSS40k engine schematic.

Since it is important to design an engine controller that performs acceptably throughout the life of the engine, the C-MAPSS40k model includes health parameters that can be adjusted to represent engine deterioration. These parameters use data from a fleet averaged profile of engine deterioration versus number of flight cycles, allowing various stages of a typical engine life cycle to be simulated.

IV. C-MAPSS40k – CoCoSim Compatibility

This section describes modifications that are required to integrate CoCoSim with C-MAPSS40k at the component, subsystem, and top levels of the model. CoCoSim currently only supports a subsection of the Simulink library, and the C-MAPSS40k model contains several blocks that are unsupported. These include continuous time functions, zero-order holds, derivatives, transfer functions, rate limiters, transport delays, interpolating lookup tables, grounds, and clocks. The unsupported blocks must therefore be converted into supported elements to enable the use of CoCoSim on the C-MAPSS40k model. Section IV.A details compatibility conversions for these unsupported Simulink blocks. Note that many of these conversions involve discretization and other operations that estimate rather than replicate the original model content. A discrete sampling time of 0.001 seconds is used for the analysis discussed in this paper, but this value can be adjusted depending on the desired accuracy versus computation speed trade-off (e.g., increased sampling rate yields increased accuracy and decreased computation speed). Section IV.B applies the compatibility conversions to the baseline controller model and tests the converted model's accuracy against the original model. S-function blocks are unsupported and a viable conversion has not been identified; the consequences of this for the engine model are discussed in Section IV.C. Modifications to the top level C-MAPSS40k model are discussed in Section IV.D.

A. Simulink Block Compatibility Conversions

1. State Space (continuous)

Continuous time state-space models are not currently compatible. However, discrete state-space models are compatible. To achieve compatibility, continuous state-space blocks are discretized. This is done by taking the A, B, C, and D matrices from the continuous block, using the discretizing "c2d" MATLAB command with a user specified sampling time, and inserting the resulting discrete A, B, C, and D matrices into a discrete state-space block.

2. *Integrator (continuous)*

Continuous time integrator blocks are replaced with discrete time integrator blocks.

3. *Zero-Order Hold*

A zero-order hold block is replaced with a discrete state-space block that acts as a pass-through with the desired sampling time specified by the user. In some cases, the sampling time of the block downstream of the zero-order hold is defined within the block itself, in which case the zero-order hold can be removed entirely.

4. *Discrete Derivative*

Discrete derivative blocks are replaced with the integrator loop configuration depicted in Figure 3. The discrete state-space block acts as a zero-order hold as previously described. The memory block is used if an initial condition is specified in the original derivative block. A gain G is incorporated into the integrator loop as shown in Figure 3 and manipulated to improve the derivative response.

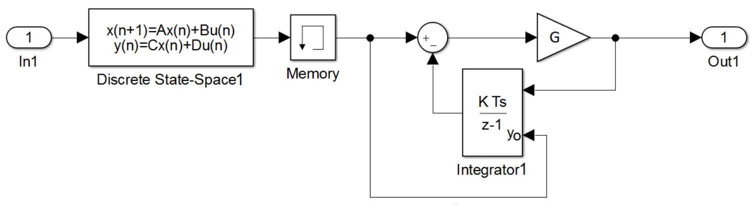


Figure 3. CoCoSim compatible derivative.

5. *Derivative (continuous)*

Continuous time derivative blocks are replaced with the discrete derivative described in the previous section.

6. *Discrete Transfer Function*

Neither continuous nor discrete time transfer functions are currently compatible. However, they can be modeled as discrete state-space blocks. For discrete time transfer functions, this is done by applying the "tf2ss" command with the numerator and denominator values of the transfer function block to create the A, B, C, and D matrices of a state-space representation. These matrices are then inserted into a discrete state-space block.

7. *Transfer Function (continuous)*

Continuous time transfer functions first require discretization. This is done by taking the numerator and denominator values from the transfer function block and using the "c2d" command to generate a discrete transfer function. The rest of the procedure follows the discrete transfer function process previously discussed.

8. *Rate Limiter*

Rate limiter blocks are replaced using the integrator loop with the desired limits defined within the saturation block shown in Figure 4. A gain G is incorporated into the integrator loop as shown in Figure 4 and manipulated to improve the rate limiter response. Note that this G is unrelated to the gain previously discussed in the discrete derivative conversion.

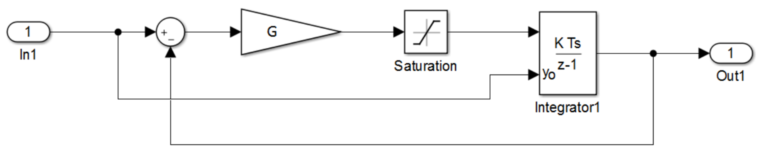


Figure 4. CoCoSim compatible rate limiter.

9. Transport Delay

Transport delay blocks are replaced with unit delay blocks. However, since unit delay only works for one time step, the sampling time must be set to the desired delay time. Since delay time is typically longer than sampling time, this often results in a slower sampling rate than desired. If the delay does not affect the verification output of the system, the transport delay block can be removed for V&V analysis to avoid affecting the sampling rate.

10. 1D Lookup Table

The engine controller model contains several lookup tables, which require only a few data points that Simulink will use to actively interpolate based on the block input value. These are converted into direct lookup tables for compatibility. Direct lookup tables do not interpolate and require that the inputs are index values for an embedded data set. Therefore, data points in the original lookup table must be interpolated prior to insertion into a direct lookup table. This operation is performed with a user-defined resolution to generate a new data grid based on the original data, which is then inserted into the direct lookup table block. Additionally, the table input values must be translated into index values of the new interpolated data grid. This translation is defined as

$$index = \left(\frac{input - lower\ limit}{resolution}\right), \tag{2}$$

with the first index value for a direct lookup table being zero. Note that this setup only works within the upper and lower limits of the interpolated data grid. A value outside of the limits will round to the nearest limit. Error is introduced since the input is rounded to the nearest predetermined data point rather than being actively interpolated as with the original lookup table. This error is reduced by increasing the resolution of the data grid interpolation. Figure 5 shows the converted setup.

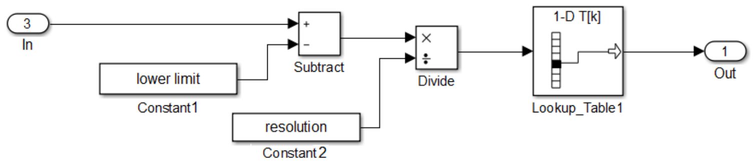


Figure 5. CoCoSim compatible 1D lookup table.

11. 2D Lookup Table

2D lookup tables are converted into direct lookup tables. This process is the same as described for 1D lookup tables, except with two inputs, each with a conversion operation as defined by (2).

12. Ground

Ground blocks are used to prevent warnings about unconnected input ports. These functions are not necessary for the analysis discussed in this paper. Therefore, they are removed from the model.

13. Clock

Since CoCoSim can freely vary the values of the inputs provided to it, clock blocks are replaced with a CoCoSim input that allows the clock time to be varied.

B. Controller Model Conversion

The conversions described in the previous section are applied to the C-MAPSS40k baseline controller model. Since many of these conversions provide an estimate and not an identical translation of the original blocks, the accuracy of the converted model to the original model must be assessed. To this end, both controllers are given sinusoidal inputs representative of actual operating conditions (e.g., altitude was varied between 0 and 40,000 ft) to observe any differences in their responses. The outputs of these controllers are compared in Figure 6. Note that Figure 6 depicts the responses to one set of inputs and therefore serves as a sanity check rather than an exhaustive comparison of the two controllers.

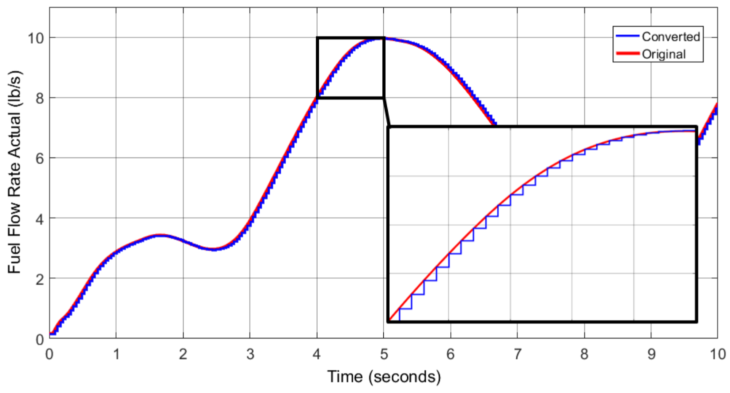


Figure 6. Converted vs. Original controller fuel flow output.

The fuel flow actuator within the C-MAPSS40k controller contains a transport delay that is converted into a unit delay. As previously discussed, since a unit delay only works for one time step, this requires the new sampling rate to equal the desired delay time. The result is the staircase response of the converted model seen in Figure 6, with the width of each step being equal to the delay time. However, each sampling point is accurate to the original controller output, indicating that the converted model provides an accurate estimate of the original model.

C. Engine Model Conversion

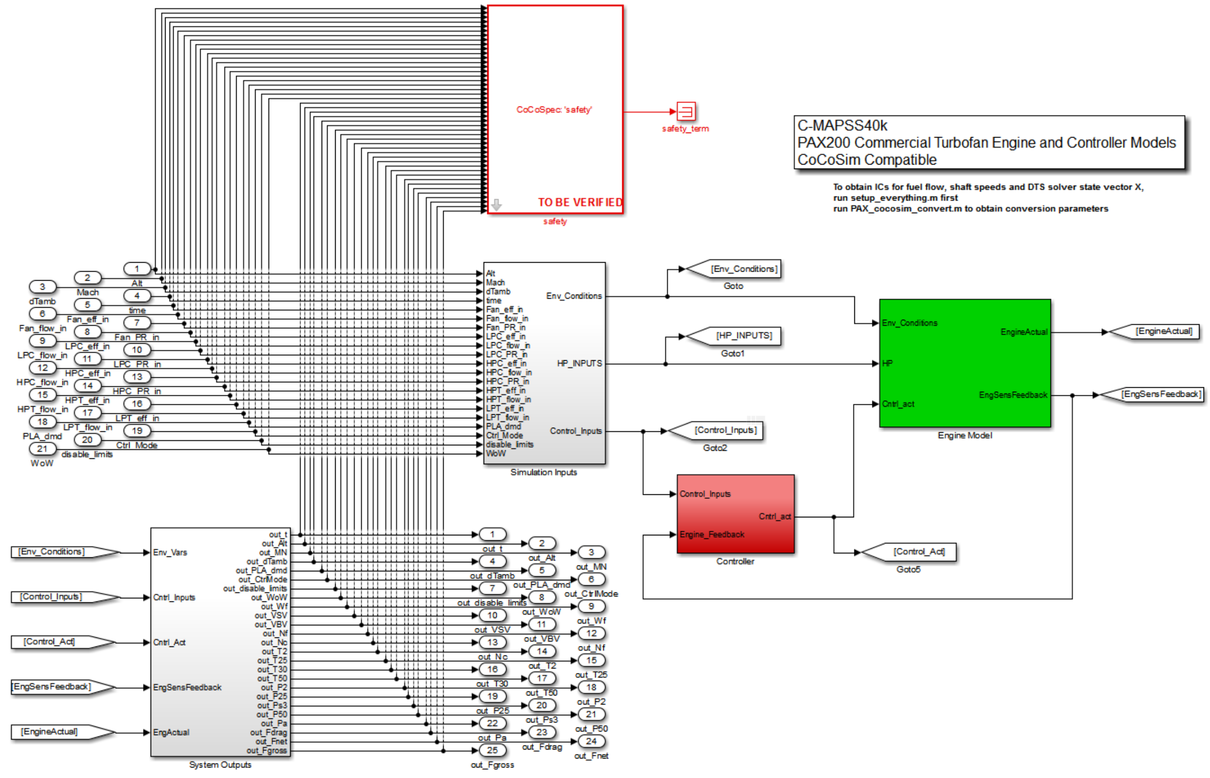
With the controller model now compatible with the CoCoSim verification tool, a compatible engine model is required. The original C-MAPSS40k nonlinear engine model is composed of several S-functions, which are currently unsupported by CoCoSim. Therefore, the nonlinear model cannot be used. However, a linearized version of the engine model is compatible. C-MAPSS40k contains a routine to determine a linear representation of the open-loop engine at a user-specified operating point [4] and will be used to develop the compatible engine model. The generated linearized system is of the form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du. \end{aligned} \tag{3}$$

The original nonlinear engine model can then be replaced by a discrete state-space Simulink block constructed using the A, B, C, and D matrices from (3). Since this linear model is determined for one set of operating conditions at a time, it is important to note that its accuracy to the nonlinear model will deteriorate as the conditions stray from the chosen linearization point. Two S-functions exist outside of the nonlinear engine model that perform calculations to determine ambient and engine inlet conditions. Given their relatively simple mathematical calculations, these blocks are manually translated from the root C code into Simulink subsystems containing CoCosim-compatible blocks.

D. C-MAPSS40k Top Level Conversion

With the controller and engine models now compatible with CoCoSim, some additional modifications to the top level C-MAPSS40k model are required to perform the verification analyses. First, inputs and outputs of the system are restructured to use inport and outport Simulink blocks. This is necessary since CoCoSim observers must be attached to these block types. Second, user specified upper and lower limits are imposed on the system inputs via saturation blocks to ensure that verification is performed using realistic values. Otherwise, CoCoSim would be able to assert unrealistically high or low values (e.g., a negative Mach number). The limits used for the analysis discussed in this paper are 0 to 40,000 ft altitude, 0 to 0.8 Mach number, -30°F to +50°F ambient temperature difference, and 40° to 80.5° power lever angle (PLA). PLA defines the physical angle at which the throttle lever is actuated with 40° as engine idle and 80.5° as max throttle. Third, an empty observer block is added to the top level model, with the inputs to this block being all the inputs and outputs of the system. The resulting CoCoSim-compatible top level C-MAPSS40k model with observer is shown in Figure 7. Any properties to be verified are constructed in the observer, and the verification process is initiated from the CoCoSim interface within Simulink. Additionally, selected inputs can be fixed as constants and any unused outputs can be removed from the observer input. This flexibility can increase process efficiency and allow for verification of more specific scenarios, such as fixed altitude.



C-MAPSS40k
PAX200 Commercial Turbofan Engine and Controller Models
CoCoSim Compatible

To obtain ICs for fuel flow, shaft speeds and DTS solver state vector X,
run setup_everything.m first
run PAX_cocosim_convert.m to obtain conversion parameters

Figure 7. CoCoSim-compatible C-MAPSS40k top level model with observer (outlined in red).

V. Engine Requirements

This section discusses the identification and definition of engine requirements for verification with CoCoSim. The engine requirements to be verified are separated into two categories: safety and performance. Safety requirements are defined such that certain properties of the engine model output remain within safe operating limits over the entire operational envelope. Performance requirements are defined such that given a set of operating conditions, the engine output must be within specific boundaries.

A. Safety Requirements

Typical safety requirements for turbofan engines are shown in Table 1. Requirements for shaft speeds and combustor pressures are taken from [5]. Requirements for engine stability are taken from Federal Aviation Regulation (FAR) §33 [19], which defines air worthiness standards for aircraft engines. Safety requirements for fuel flow are set by component limitations (e.g., max flow through a fuel metering valve). The safety limit values specific to C-MAPSS40k are taken from safety limiters within the baseline controller model. These are then used to construct a CoCoSim observer for the verification process as depicted in Figure 8. Within this observer are logical operators that assert that fan speed and core speed stay below a safe limit, combustor pressure stays below a safe limit but above a stall limit, stall margin stays above a desired minimum, and fuel flow stays below a safe limit. Given any inputs within the user specified limits, CoCoSim then checks that the fan speed, core speed, combustor pressure, stall margin, and fuel flow outputs of the C-MAPSS40k system satisfy the assertions for all scenarios. If they are always satisfied, the CoCoSim output is valid, or safe. If CoCoSim finds a value that lies outside of the limits, the output is invalid, or unsafe.

Table 1. Engine safety requirements.

Requirement	**Description**.
Maximum Shaft Speeds	Max fan and core speeds should be 110% of the speeds at normal operation on a standard day (15,000ft, 0.8 Mach, std. temperature, full power).
Maximum Combustor Pressure	Max combustor pressure is the greatest pressure in the flight envelope at any temperature (2,000ft below sea level, 0.5 Mach, delta ambient temperature -30°F).
Minimum Combustor Pressure	Minimum combustor pressure should be such that a stable, steady response is seen at a minimum idle at the full range of operational altitudes.
Surge and Stall FAR §33.28(b)	The engine does not surge, stall, or experience unacceptable thrust or power changes or oscillations or other unacceptable characteristics.
Maximum Fuel Flow	Max fuel flow does not exceed a safe limit.

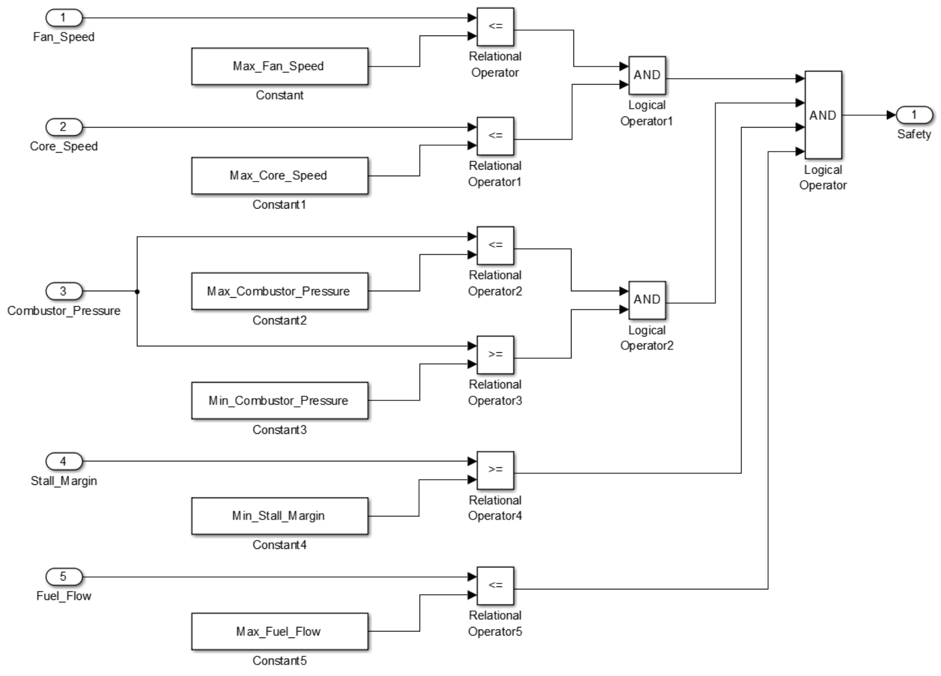


Figure 8. Safety requirements CoCoSim observer.

B. Performance Requirements

To identify engine performance requirements, expected performance data must first be obtained. This is done by simulating the original C-MAPSS40k model with the nonlinear engine for multiple operating conditions. Baseline data are gathered at the boundaries of the operational envelope by performing simulations at all combinations of minimum, maximum, and nominal values for Mach number, ambient temperature difference, and altitude while PLA is ramped from idle to maximum throttle. The values used for operating conditions are shown in Table 2.

Table 2. Simulation input values.

Condition	Minimum	Nominal	Maximum.
Mach number	0.0	0.4	0.8
Ambient ΔT [°F]	-30	0	50
Altitude [ft]	0	20,000	40,000

The results of these simulations for

$$\frac{Ps3}{P_a}, \qquad \frac{W_f}{P_a\sqrt{T_2}}, \qquad \frac{N_f}{\sqrt{T_2}}, \qquad \frac{N_c}{\sqrt{T_2}}, \tag{4}$$

versus EPR are shown in Figure 9. These ratios can be used as performance measurements for turbofan engines, such as in [20]. By observing data for these ratios against EPR, which correlates to thrust, expected engine performance can be mapped. As one might expect, Figure 9 shows increases in EPR yielding increases in compressor pressure, fuel flow, and engine speeds. Additionally, engines at higher Mach numbers require greater amounts of compressor pressure, fuel flow, and engine speeds to produce the same EPR as at lower Mach numbers, as depicted by the three different curves in each panel of Figure 9. This behavior is expected since turbofan engines experience a decrease in thrust efficiency at higher forward velocities.

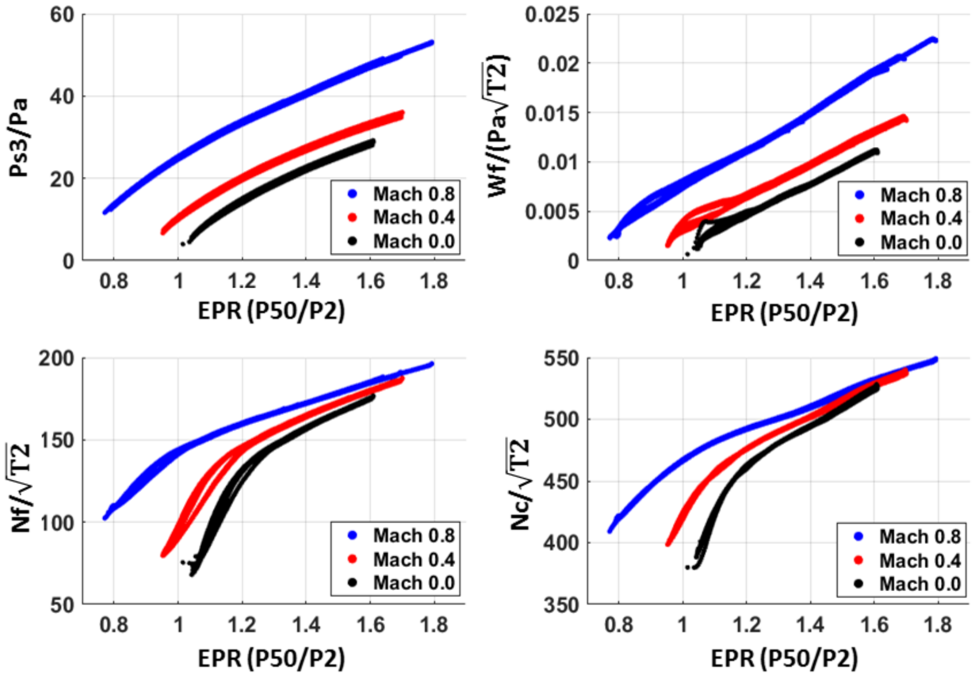


Figure 9. C-MAPSS40k simulation results.

To build a performance requirement from these data, upper and lower limits are defined as offsets from the average of the data curve. Figure 10 shows the $Ps3/P_a$ vs. EPR data at Mach 0.8 with example upper and lower performance limits. By modeling the limits as lookup tables, an observer can be formulated to check that for a given condition (e.g., EPR), certain properties (e.g., $Ps3/P_a$) stay within defined limits. The block diagram for a $Ps3/P_a$ vs. EPR observer is depicted in Figure 11, which shows the observer inputs $Ps3$, P_a, and EPR being compared to performance limit lookup tables via relational operators.

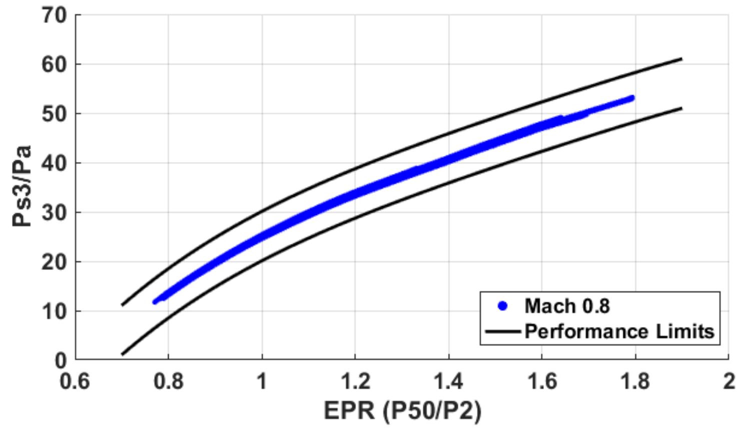


Figure 10. $Ps3/P_a$ **performance data with imposed limits.**

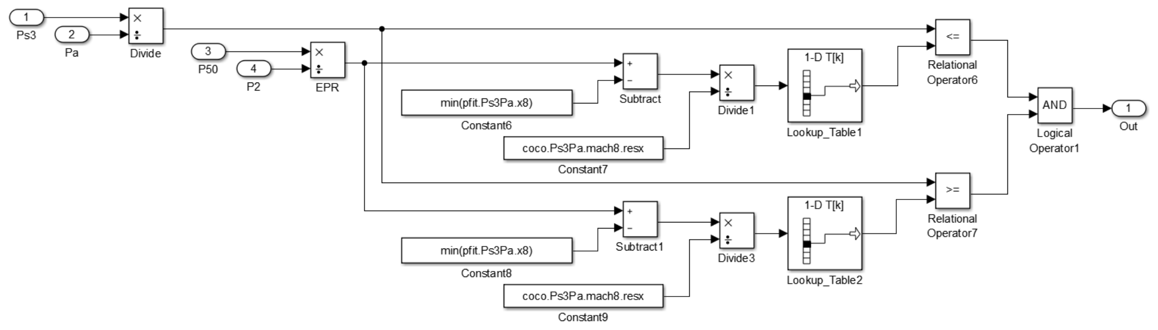


Figure 11. $Ps3/P_a$ **performance CoCoSim observer.**

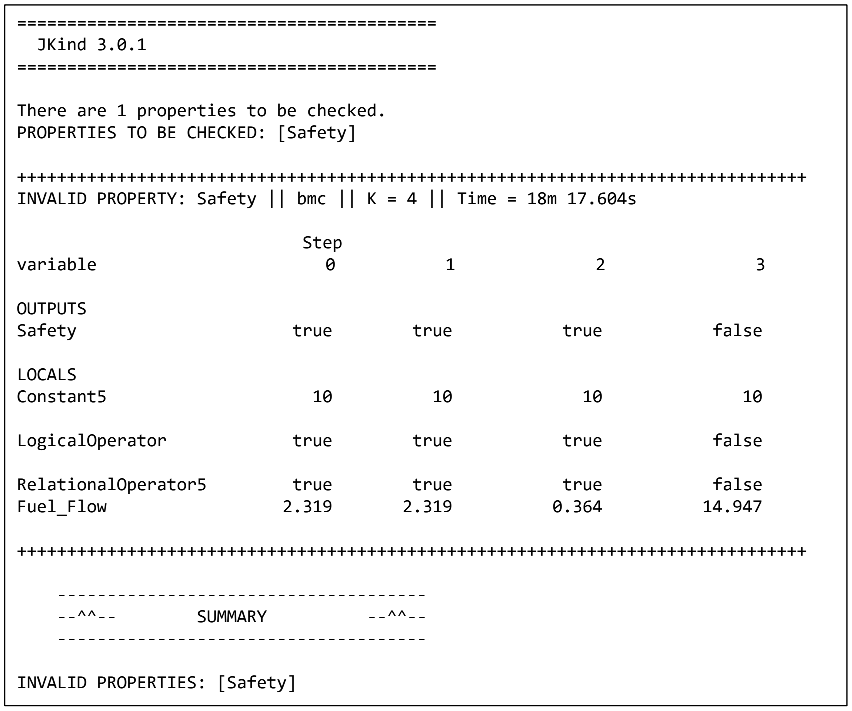
VI. Results

This section verifies engine safety using the converted C-MAPSS40k model and requirements defined in Sections IV and V. The maximum fuel flow safety requirement discussed in Section V and shown in Figure 8 was intentionally lowered below the expected maximum output to observe whether or not CoCoSim could find an appropriately invalid case. For this analysis, engine health parameters are defined such that there is no performance degradation due to engine usage (i.e., a new engine).

The CoCoSim verification process was initiated on the converted C-MAPSS40k model with the safety requirements observer in Figure 8. The terminal output is depicted in Figure 12, truncated to show only the values of interest. The four inductive steps of the JKind verification process that resulted from this analysis can be seen with properties of interest. The entire observer is "Safety," the intentionally lowered upper fuel flow limit is "Constant5," the operator that checks for all safety assertions to be valid is "LogicalOperator," the operator that checks for maximum fuel flow is "RelationalOperator5," and the output fuel flow is "Fuel_Flow." It can be seen that for the first three verification steps the safety assertions were satisfied (i.e., true). However, in the fourth step "Fuel_Flow" exceeded the limit "Constant5," resulting in "RelationalOperator5" and thus "LogicalOperator" to be falsified. This resulted in the observer "Safety" being false. Therefore, CoCoSim successfully determined the observer assertions to be invalid as seen in the summary section of Figure 12. The process time for this analysis was 18 minutes and 17.6 seconds on an

Intel Xeon E5-1650 3.20GHz processor with 16GB of RAM.

This process was repeated while lowering the upper limits of various requirements to provoke an invalid response. A successful invalid response was typically found in under an hour of processing time. However, when the requirement limits were realistic and a valid response was expected, the verification process did not converge to a solution. A separate analysis using the performance requirement depicted in Figure 10 also failed to converge to a solution. It is suspected that the non-convergent behavior was caused by a circular dependency within the linearized engine that was not encountered for the invalid test cases. Circular dependencies occur when two or more components are mutually dependent on each other (e.g., component 1 requires component 2 to be defined first and vice versa). In this case, the verification process cannot converge to a valid solution but also cannot find any invalid cases, as expected, to break the cycle. To test if the linear engine model was the source of the non-convergent behavior, the controller block was analyzed without the engine with a fuel flow command requirement. Applying the same realistic limits as in the non-convergent case, CoCoSim successfully determined the observer to be valid within 3 seconds. Therefore, modifications to the linearized engine model may be necessary to enable valid solution outputs.


```
==========================================
  JKind 3.0.1
==========================================

There are 1 properties to be checked.
PROPERTIES TO BE CHECKED: [Safety]

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INVALID PROPERTY: Safety || bmc || K = 4 || Time = 18m 17.604s

                             Step
variable                       0           1              2               3

OUTPUTS
Safety                      true        true           true           false

LOCALS
Constant5                     10          10             10              10

LogicalOperator             true        true           true           false

RelationalOperator5         true        true           true           false
Fuel_Flow                  2.319       2.319          0.364          14.947

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    -------------------------------------
    --^^--        SUMMARY          --^^--
    -------------------------------------

INVALID PROPERTIES: [Safety]
```

Figure 12. Example verification process output (invalid).

VII. Discussion

New capabilities are being incorporated into CoCoSim as the V&V tool continues to be developed. This includes the addition of supported Simulinks blocks. Future work will focus on reverting the converted portions of the C-MAPSS40k model back to their original forms as more blocks are supported to restore fidelity to the original model. In particular, support for S-functions would allow the use of the original nonlinear engine. Ideally, the verification process would be applied directly to the original C-MAPSS40k top level model without any compatibility conversions.

Future work will also involve adding the capability to verify transient response requirements. For example, the FAR §33.73(b) thrust transient requirement states that "the design and construction of the engine must enable an increase from the fixed minimum flight idle power lever position when provided, or if not provided, from not more than 15 percent of the rated takeoff power or thrust available to 95 percent rated takeoff power or thrust in not over 5 seconds." A means to verify requirements such as the FAR §33.73(b) will be necessary for a comprehensive V&V scheme.

Despite the current challenges, the CoCoSim verification process was successfully demonstrated on certain properties of the converted C-MAPSS40k model with a baseline controller. It is expected that this will serve as a foundation for a safety verification framework for aircraft engine control. After a comprehensive analysis using the baseline controller is performed, the framework can be used to verify new control algorithms. Additionally, much of the work presented in this paper is not specific to aircraft engines and can be applied to other system models.

VIII. Conclusion

This paper demonstrated the CoCoSim verification and validation (V&V) process and capabilities on the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS40k) turbofan engine and controller model. Since CoCoSim is still under development, only a subsection of the Simulink library is currently supported. Therefore, unsupported components within the C-MAPSS40k model had to be converted to achieve compatibility with the verification tool. Some of these conversions resulted in reduced model fidelity, including linearization of the nonlinear engine. Typical safety and performance requirements for turbofan engines were identified to serve as the basis for the CoCoSim analysis. The safety requirements were formulated as assertions within a verification observer that was attached to the inputs and outputs of the top level converted C-MAPSS40k model. The verification process was then performed for several cases. While CoCoSim successfully found solutions for invalid cases, non-convergent behavior within the linearized engine model prevented verification of valid cases. Future work will include modifying the linear engine model to enable the verification of valid cases.

Acknowledgments

The authors gratefully acknowledge Misty D. Davies (NASA Ames Research Center) and Temesghen Kahsai Azene (Stinger Ghaffarian Technologies, Inc) for their invaluable assistance with the CoCoSim software. The research documented in this paper was sponsored by the NASA Airspace Operations and Safety Program under the System-Wide Safety Project.

References

[1]CoCoSim, [online] https://github.com/coco-team/cocoSim.

[2]Hueschen, R. M., "Development of the Transport Class Model (TCM) Aircraft Simulation From a Sub-Scale Generic Transport Model (GTM) Simulation," August 2011, NASA/TM2011-217169.

[3]Connolly, J. W., Csank, J., Chicatelli, A., and Kilver, J., "Model-Based Control of a Nonlinear Aircraft Engine Simulation using an Optimal Tuner Kalman Filter Approach," *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2013, AIAA-2013-4002.

[4]May, R., Csank, J., Lavelle, T., Litt, J., and Guo, T.-H., "A High-Fidelity Simulation of a Generic Commercial Aircraft Engine and Controller," *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2010, AIAA-2010-6630.

[5]Csank, J., May, R., Litt, J., and Guo, T.-H., "Control Design for a Generic Commercial Aircraft Engine," *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2010, AIAA-2010-6629.

[6]Dieumegard, A., Garoche, P.-L., Kahsai, T., Taillar, A., and Thirioux, X., "Compilation of Synchronous Observers as Code Contracts," *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ACM, 2015, pp. 1933–1939.

[7]Champion, A., Gurfinkel, A., Kahsai, T., and Tinelli, C., "CoCoSpec: A Mode-Aware Contract Language for Reactive Systems," *International Conference on Software Engineering and Formal Methods*, Springer, 2016, pp. 347–366.

[8]Halbwachs, N., Caspi, P., Raymond, P., and Pilaud, D., "The Synchronous Data Flow Programming Language LUSTRE," *Proceedings of the IEEE*, Vol. 79, No. 9, 1991, pp. 1305–1320.

[9]JKind, [online] http://loonwerks.com/tools/jkind.html.

[10]Ghassabani, E., Gacek, A., and Whalen, M. W., "Efficient Generation of Inductive Validity Cores for Safety Properties," *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM, 2016, pp. 314–325.

[11]Gacek, A., Katis, A., Whalen, M. W., Backes, J., and Cofer, D., "Towards Realizability Checking of Contracts Using Theories," *NASA Formal Methods Symposium: 7th International Symposium*, Springer, 2015, pp. 173–187.

[12]Sheeran, M., Singh, S., and Stålmarck, G., "Checking Safety Properties Using Induction and a SAT-Solver," *International Conference on Formal Methods in Computer-Aided Design*, Springer, 2000, pp. 127–144.

[13]Een, N., Mishchenko, A., and Brayton, R., "Efficient Implementation of Property Directed Reachability," *Formal Methods in Computer-Aided Design (FMCAD), 2011*, IEEE, 2011, pp. 125–134.

[14]Bradley, A. R., "SAT-Based Model Checking without Unrolling," *International Workshop on Verification, Model Checking, and Abstract Interpretation*, Springer, 2011, pp. 70–87.

[15]Bradley, A. R. and Manna, Z., "Checking Safety by Inductive Generalization of Counterexamples to Induction," *Formal Methods in Computer Aided Design, 2007. FMCAD'07*, IEEE, 2007, pp. 173–180.

[16]Biere, A., Heule, M., van Maaren, H., and Walsh, T., *Handbook of Satisfiability*, IOS Press, 2009.

[17]De Moura, L. and Bjørner, N., "Z3: An Efficient SMT Solver," *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2008, pp. 337–340.

[18]Brat, G., Bushnell, D., Davies, M., Giannakopoulou, D., Howar, F., and Kahsai, T., "Verifying the Safety of a Flight-Critical System," *International Symposium on Formal Methods*, Springer, 2015, pp. 308–324.

[19]Electronic Code of Federal Regulations, Title 14, Chapter I, Subchapter C, Part 33 - Airworthiness Standards: Aircraft Engines, [online] https://www.ecfr.gov/cgi-bin/text-idx?node=pt14.1.33&rgn=div5#se14.1.33_128.

[20]United Aircraft Corporation, Pratt & Whitney Aircraft Division, *The Aircraft Gas Turbine Engine and its Operation*, 1974.