NASA/TM—2012-217432

User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)

Version 2

商用模块化航空推进系统仿真用户指南

Yuan Liu
N&R Engineering, Parma Heights, Ohio

Dean K. Frederick
Saratoga Control Systems, Inc., Saratoga Springs, New York

Jonathan A. DeCastro
ASRC Aerospace Corporation, Cleveland, Ohio

Jonathan S. Litt and William W. Chan
Glenn Research Center, Cleveland, Ohio

March 2012

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 443–757–5803

- Telephone the NASA STI Help Desk at 443–757–5802

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2012-217432

User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)

Version 2

Yuan Liu
N&R Engineering, Parma Heights, Ohio

Dean K. Frederick
Saratoga Control Systems, Inc., Saratoga Springs, New York

Jonathan A. DeCastro
ASRC Aerospace Corporation, Cleveland, Ohio

Jonathan S. Litt and William W. Chan
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

March 2012

Acknowledgments

The authors gratefully acknowledge the contributors to and testers of the code, as well as those who otherwise made the creation of C-MAPSS possible: Donald L. Simon, Ryan D. May, Jeffrey T. Csank, John A. Sekora, Kimberly A. Lemon, Gregory E. McGlynn, Sanjay Garg, and Vinod Nagpal.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Contents

User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)

Version 2

Yuan Liu
N&R Engineering
Parma Heights, Ohio 44130

Dean K. Frederick
Saratoga Control Systems, Inc.
Saratoga Springs, New York 12866

Jonathan A. DeCastro
ASRC Aerospace Corporation
Cleveland, Ohio 44135

Jonathan S. Litt and William W. Chan
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

1.0 Introduction

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) is a nonlinear, dynamic, component-level model (CLM) of a commercial, high-bypass, dual-spool turbofan engine. The overall simulation is implemented in the MATLAB/Simulink environment, thus providing flexible interaction with the software user. One can take full advantage of the features included in C-MAPSS through a straightforward graphical user interface (GUI). C-MAPSS has built-in capabilities (e.g., Newton-Raphson solver for calculation of steady-state operating points, generation of linear engine models, and different engine control system configurations) that are directly accessible through the top-level GUI. Furthermore, advanced users can readily modify and customize the model to their specific requirements using the graphical syntax of the Simulink environment.

C-MAPSS is a unique addition to the resources of the general aeronautical research community because it is a realistic, non-proprietary simulation of both the engine and its control system. The CLM simulates a modern, commercial turbofan engine, and is presented in a user-friendly, visually oriented manner. Moreover, the control system is representative of the control architecture in a Full Authority Digital Engine Control (FADEC) system used with many present-day aircraft engines.

This guide is written for Version 2 of the C-MAPSS software, which has a number of important differences from Version 1. The original version of C-MAPSS is described in Reference 1. As explained in Reference 2, Version 2 includes the addition of:

- Variable stator vanes (VSV) in high pressure compressor (HPC)
- Variable bleed valve (VBV) for controlling inter-compressor bleed flow
- Actuator dynamics for fuel-metering valve (FMV), VSV, and VBV
- Sensor dynamics
- Combustion delay
- Reynolds Number effects in HPC

- Controllers and limit regulators with second-order incremental portions
- Extensively revised GUI with more intuitive and compact presentation of program features and model parameters

The aim of this user's guide is to familiarize the new user with the C-MAPSS software through descriptive instructions and examples. It is also meant to serve as a reference manual for the more seasoned user for new features and/or definitions and descriptions of model variables.

Nomenclature

Alt	Altitude
CLM	Component Level Model
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
degF	Degrees Fahrenheit
DTamb	Deviation from standard temperature at given altitude
DVBV	Deviation from variable bleed valve schedule
DVSV	Deviation from variable stator vane schedule
EPR	Engine Pressure Ratio
error_tol	Error tolerance for the iterative solver
FADEC	Full Authority Digital Engine Control
Fn	Net thrust
GUI	Graphical User Interface
HPC	High Pressure Compressor
HPT	High Pressure Turbine
LEM	Linear Engine Model
LPC	Low Pressure Compressor
LPT	Low Pressure Turbine
max_iter	Maximum number of iterations allowed per time step for the iterative solver
MN	Mach Number
Nc	Core speed
Nc_dot	Core acceleration
Nf	Fan Speed
PCNfR	Percent corrected fan speed
Phi	Ratio of fuel flow to Ps30 (pph/psi)
pph	Pound-mass per hour
pps	Pound-mass per second
PR	Pressure Ratio
Ps30	HPC outlet static pressure
RMS	Root Mean Square
T48	HPT outlet total temperature
TRA	Throttle Resolver Angle
VBV	Variable Bleed Valve
VSV	Variable Stator Vanes
Wf	Fuel flow (pph)

2.0 Engine Model Description

This section describes the engine and control system and their implementation in C-MAPSS. The C-MAPSS engine is a 90,000-lb thrust class turbofan engine with a dual-spool configuration and a bypass ratio of approximately 8.4. Figure 1 shows a schematic diagram of the upper half of the engine, which consists of an inlet, bypass nozzle, fan, low pressure compressor (LPC), high pressure compressor (HPC), combustor, high pressure turbine (HPT), low pressure turbine (LPT), and core nozzle. The HPT powers the HPC, and the LPT drives both the LPC and fan. The actuators that are modeled consist of the fuel metering valve, which controls the fuel flow into the combustor, variable stator vanes (VSV) within the HPC, and a variable bleed valve (VBV) at the outlet of the LPC.

The engine is modeled as a nonlinear dynamical system with two state variables: fan speed and core speed. The engine model is component-level; each engine component is modeled independently, with the overall simulation connecting them together. Component-level performance is calculated using thermodynamic relationships and data interpolation from the fan, compressor, and turbine maps. The system inputs are fuel flow, deviation from scheduled VSV angle (DVSV), deviation from scheduled VBV position (DVBV), and a set of 13 health parameters that simulate engine degradation. The health parameters (listed in Table 2 in the Appendix) consist of flow, efficiency, and pressure-ratio modifiers for the fan, LPC, and HPC, and flow and efficiency modifiers for the HPT and LPT. The model outputs are listed in Table 3 and Table 4 in the Appendix, though any simulation variable is accessible with minor modifications to the Simulink model. At each simulation time step, the engine model is iterated until the flow rate into and out of each rotating component is consistent. The iterative solver is similar to that in C-MAPSS40k, a 40,000-lb thrust class turbofan engine simulation (Ref. 3), and adapted for the C-MAPSS engine model.

To control the engine, the three input parameters—fuel flow, DVSV, and DVBV—must be specified. These values can be directly supplied to the engine, in which case the engine is said to be in “open-loop” operation. Alternatively, C-MAPSS provides a variety of control systems to utilize with the engine. In this more conventional “closed-loop” scenario, the engine control system calculates the appropriate fuel flow command from a user-specified throttle input. VSV and VBV actuator positions are generally scheduled based on current flight and engine conditions, though they may be offset using DVSV and DVBV inputs. The default engine control system is shown in Figure 2. This configuration represents the most complete control architecture C-MAPSS offers. The `Power management` block converts throttle resolver angle (TRA, in degrees) to desired fan speed, Nf. The `scheduled Nf controller` block then calculates the change in fuel flow (Wf) necessary to achieve the desired fan speed by comparing it with measured fan speed. The controller gains are scheduled on flight conditions to ensure operation across a wide flight envelope. A set of limit regulators (located within the `High-limit regulators` and `Low-limit regulators` blocks) protects against combustor blowout, mechanical failure from

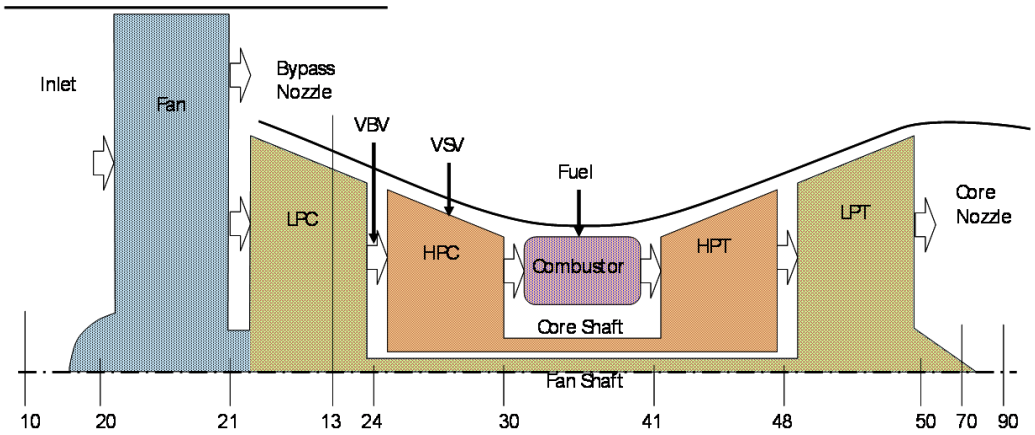

Figure 1.—Schematic diagram showing the components and station numbers of the C-MAPSS engine.

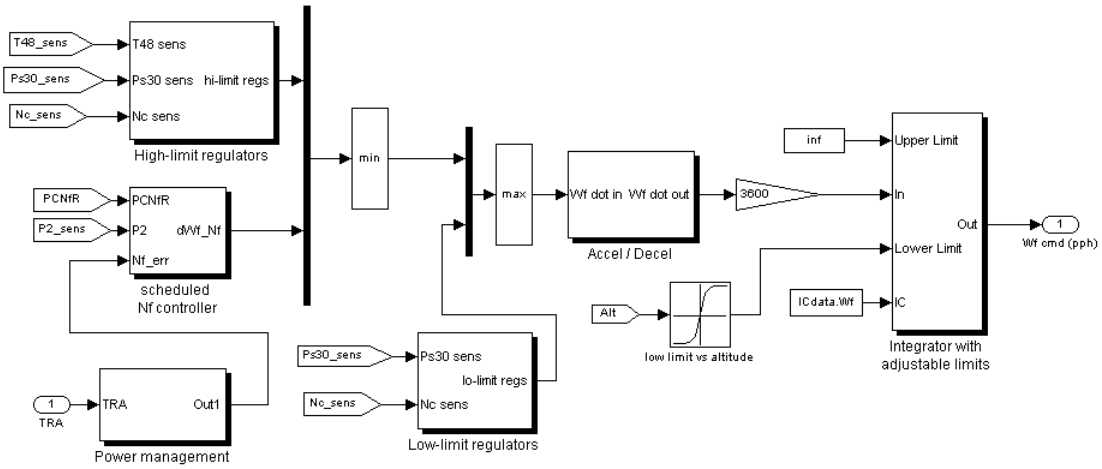

Figure 2.—Default C-MAPSS engine control system.

excessive core speed, combustor failure due to high pressure, and turbine damage due to prolonged high temperature exposure. There are also limits on spool acceleration and deceleration (the `Accel/Decel` block), which protect the engine against operability issues such as compressor stall by preventing the fuel flow from increasing or decreasing too rapidly. Other configurations available (see Section 4.1) are essentially control systems with different subsets of the components shown in Figure 2. Details of the C-MAPSS engine control design are available in Reference 2.

3.0 Simulation Setup

Once properly set up, C-MAPSS grants a large amount of flexibility in modeling the engine responses to various flight conditions, input profiles, and health and degradation scenarios. This section provides detailed instructions for setting up the software and initializing a session and summarizes the operations available using the GUI.

3.1 Initialization 介绍软件

When initially downloaded from the NASA Glenn Research Center Software Catalog,[1] the C-MAPSS software is compacted as a single .zip file. Uncompressing this file will yield a top-level folder containing all the files and the directory structure necessary to run the software. This folder, henceforth referred to as the root directory, is portable and can be placed in any directory, provided that there are no blank spaces in the name of the root directory or the path to it.

Figure 3 shows the contents and structure of the root directory. On a cautionary note, it is not advisable to place .MAT files in the root directory; their contents may be unintentionally loaded by C-MAPSS into the MATLAB workspace during the course of running the simulation. The names of most folders in the root directory are representative of their content. For instance, the `HPC` directory contains performance tables and scalars for the high pressure compressor (HPC). The Simulink model files and various built-in model analysis tools are stored in `CLsim`. `CLM` consists of plotting routines and pre-saved flight conditions, flight profiles, linear models, and regulators (described in detail in subsequent sections). The `DLL` directory consists of compiled and source code for various engine components. The aerothermodynamic calculations for the engine components are written in C. Instructions for recompiling the C code are contained within each source file. (Note: C-MAPSS has been tested and shown to function properly on MATLAB version 2010a with Simulink and Control System Toolbox; the source files have been compiled for a computer system running the Windows operating system).

[1] http://sr.grc.nasa.gov/

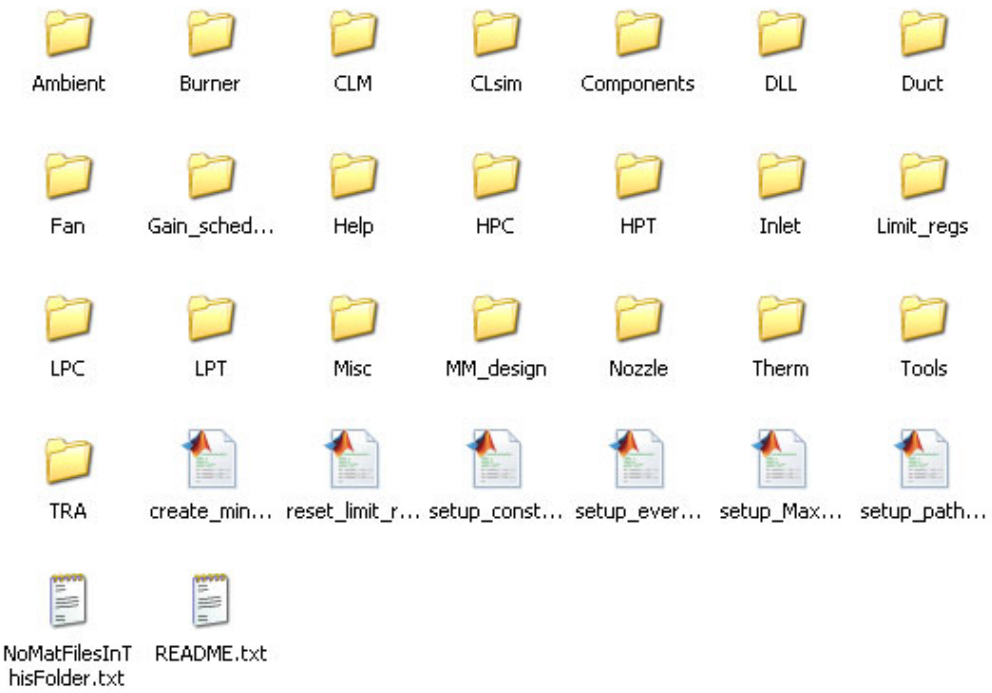

Figure 3.—C-MAPSS root directory structure.

3.2 Top-Level Graphical User Interface 顶层图形用户界面

The C-MAPSS graphical user interface (GUI) facilitates the primary interactions between the user and the software. To open the main GUI window, initiate MATLAB and ensure that the Current Directory is set as the C-MAPSS root directory. Then, execute `setup_everything.m`. This will initialize the simulation in the MATLAB workspace and open up the main GUI, which will be clean of any simulation parameters (Figure 4). The top-level GUI can be resized, if desired, with its components remaining generally proportional to the overall window size. However, certain components will overlap if the window is made too small.

The main menu options are located under the title bar of the top-level GUI. They are, from left to right:

- `Model/Controller Selection`: select model (open-loop, closed-loop) and controller type (if applicable)
- `Flight Condition`: load, create, and/or save flight conditions
- `Input Profile`: load, create, and/or save input profiles
- `Linear Model`: load, create, and/or save linear engine models (LEM); analyze LEM; compare LEM and CLM responses
- `Plot`: generate a variety of plots after running the simulation
- `Help`: access quick help with this GUI and information about C-MAPSS

While setting up the simulation, the main GUI will update with the latest model status and summary. Once the simulation is sufficiently defined, pressing the `Run Simulation` button runs the model. Pop-up windows will notify the user of any errors in the setup process.[2] The `Restart` button clears the MATLAB workspace and reloads the main GUI. The `Close` button will exit the program.

[2] Alternatively, clicking the play button ( ) on the Simulink window will also run the simulation. However, this method will bypass the GUI's error-checking process.

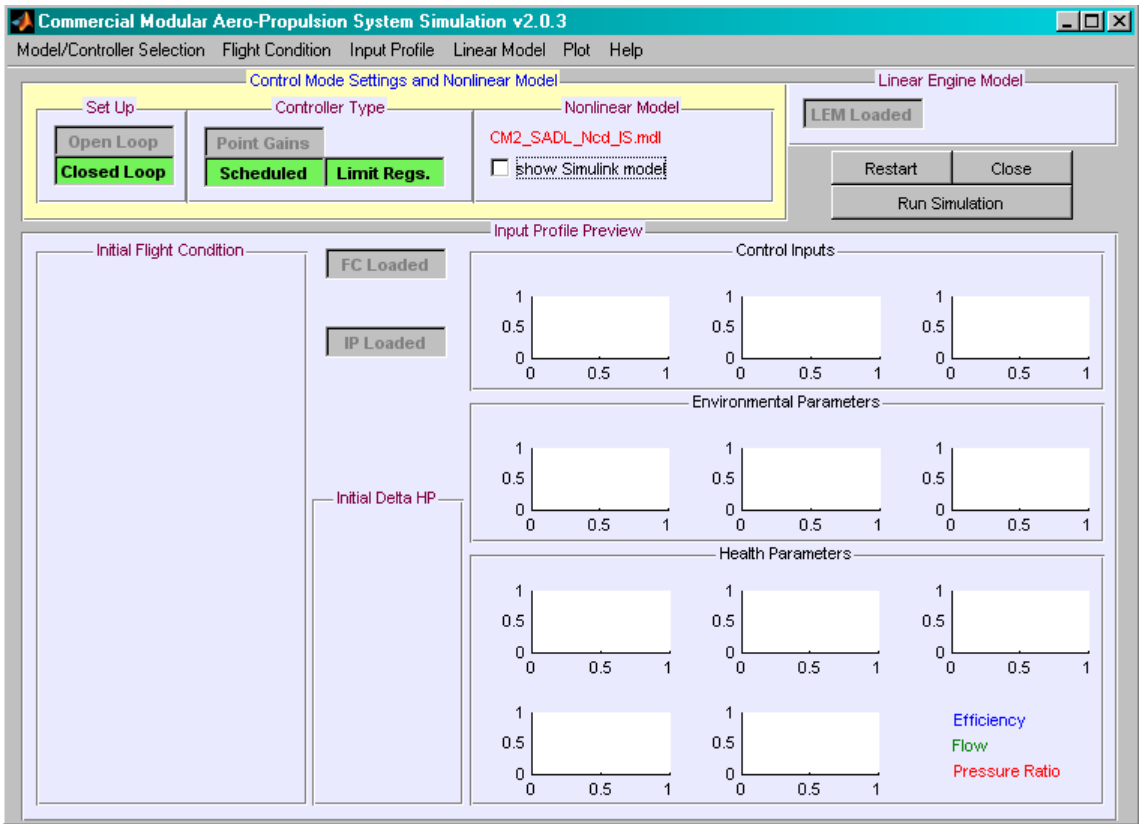

Figure 4.—Top-level GUI for C-MAPSS, after running `setup_everything`.

4.0 Running a Simulation

4.1 Model Selection

The first step when using C-MAPSS is to decide which Simulink model to use. Select the desired model using the `Model/Controller Selection` menu option from the main GUI. Several different model files are included with C-MAPSS. Each model consists of a distinct control system configuration (the engine components are identical). Table 1 lists the available model files and their associated control system architecture. To select a user-defined model file not included in the list, select `Custom Model` from the dropdown menu. The model selected by default is `CM2_SADL_Ncd_IS`. The `Nonlinear Model` panel in the main GUI window will display the name of the currently selected model file. Checking the `show Simulink model` box will open the model file. The following describes the control configurations of the different model files.

The nomenclature for the models listed in Table 1, though not rigorous, generally indicates which control system components are present:

- S: gain-scheduled fan speed controller
- AD: acceleration/deceleration limiters
- L: limit regulators for burner static pressure (Ps30), high pressure turbine exit total temperature (T48), and core speed (Nc)
- Ncd: limit regulator for core acceleration
- OL: open-loop, i.e., no control system

The default model, CM2_SADL_Ncd_IS, is the most complete configuration since it contains all the available control system components. The model includes the full control system, which consists of a gain-scheduled fan speed controller and the complete suite of protection logic. CM2_SAD_IS consists of the gain-scheduled fan speed controller and acceleration/deceleration limiters, but no limit regulators. CM2_OL_IS is an open-loop engine model. Since the model does not have a control system to calculate fuel flow from throttle resolver angle, fuel flow must be provided directly (see Section 4.3). Finally, CM2_switched_ctrl_IS is a "switched" control system configuration. This model contains flags that activate or deactivate various control system components. The user can choose to: (1) disable the limit regulators and acceleration/deceleration limiters, (2) select among the gain-scheduled fan speed controller, single-gain controller, or no controller. The available configurations of CM2_switched_ctrl_IS are shown in Table 1. Note that the no controller option essentially duplicates the performance of the open-loop engine model, CM2_OL_IS.

TABLE 1.—SUMMARY OF SIMULINK MODELS AND AVAILABLE CONFIGURATIONS

Model name	Nf controller	Ps30/T48/Nc limit regulators	Nc_dot limit regulator	Accel/Decel limiter
CM2_OL_IS	No (open-loop)	No	No	No
CM2_SAD_IS	Scheduled	No	No	Yes
CM2_SADL_Ncd_IS (default selection)	Scheduled	Yes	Yes	Yes
CM2_switched_ctrl_IS (4 available configurations)	Scheduled	Yes	No	Yes
	Scheduled	No	No	No
	Point gains	No	No	No
	No (open-loop)	No	No	No

4.2 Flight Condition

Once the desired model file has been selected, the user must specify the initial flight condition for the simulation—power setting (TRA or Wf), altitude, Mach number, standard ambient temperature offset (DTamb)—by loading an existing flight condition file or creating a new one. The design ranges for these parameters are:

一旦选择了所需的模型文件，用户必须指定模拟功率设定(tra或wf)、高度、马赫数、标准环境温度的初始飞行条件。
fset(DTamb)-通过加载现有的飞行条件文件或创建一个新的文件。这些参数的设计范围如下

- TRA: 0° to 100°
- Altitude: 0 to 40,000 ft
- Mach number: 0 to 0.9
- DTamb: –60 to 103 °F

To load a flight condition file, select Load from the Flight Condition menu option located at the top of the main GUI window. A new window will open (Figure 5), displaying a list of available flight conditions along with a concise description for each (filename, TRA, altitude, Mach number, DTamb). Note that the list is only representative of the flight condition files in the directory, CLM\FC_files. This directory contains 16 preset flight conditions: FC01.mat through FC16.mat. User-created flight conditions located in other directories may be selected by choosing the Change Directory option and clicking Refresh List. Figure 5 shows three user-created files in addition to the 16 preset flight conditions with the default naming style. To load one of the flight conditions, select the file from the list. This will load the selected .MAT file onto the workspace and the selection window will close automatically. The main GUI window will update with a summary of the flight condition (Figure 6).

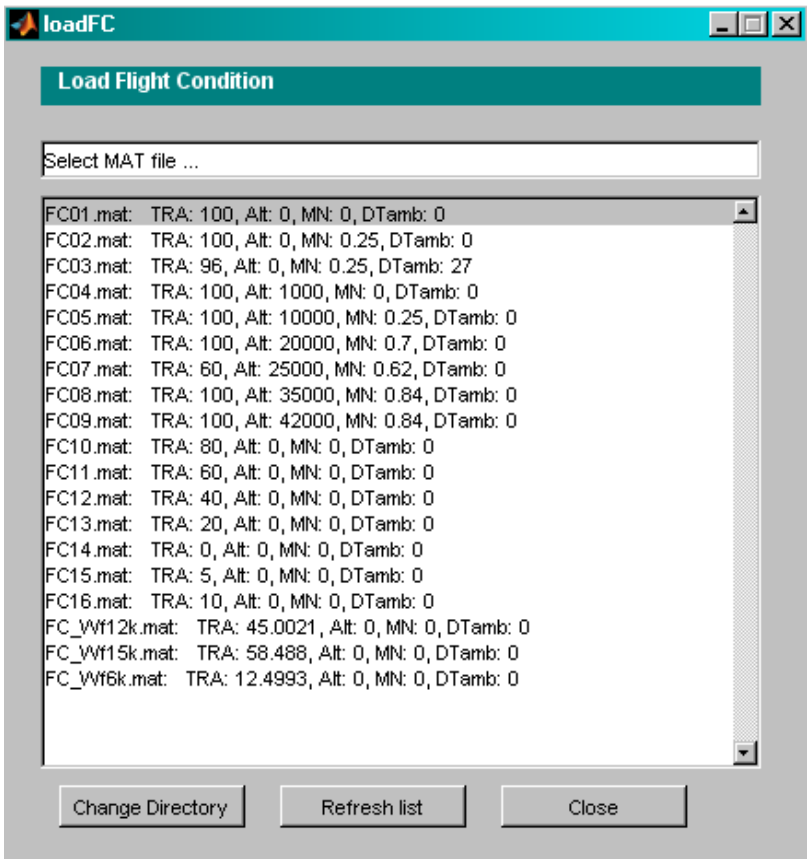

Figure 5.—Loading a flight condition.

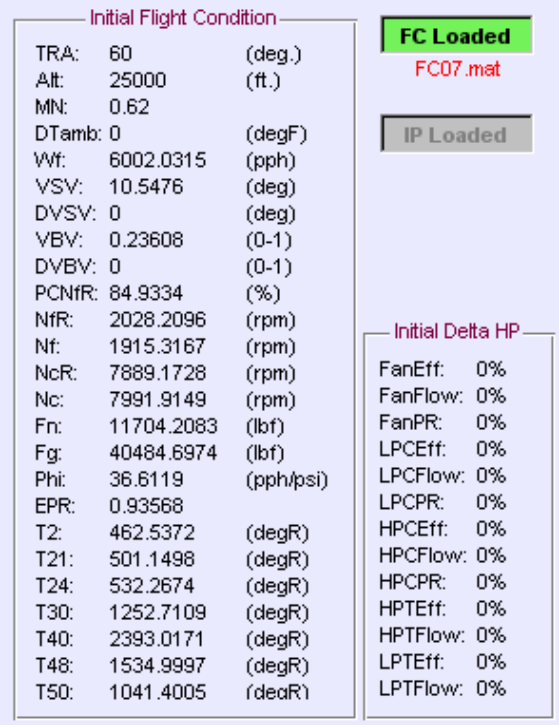

Figure 6.—Flight condition panel on the main GUI window after loading flight condition file.

C-MAPSS includes a steady-state solver that enables users to create new flight conditions. This solver, which applies the Newton-Raphson method, enables the engine to be balanced at a user-specified operating point. To create a new flight condition, begin by first loading an existing flight condition file. Next, select `Flight Condition > Create > Using steady-state solver` to open the solver GUI (Figure 7). Enter the desired flight condition parameters, including engine health parameters, into the provided fields. It is important to note that only one of the three power setting variables (fuel flow, TRA, net thrust) may be specified at a time. Given one power setting variable, the solver has enough information to calculate corresponding values for the other two. Therefore, flight conditions created by the steady-state solver are usable by both open and closed-loop engine configurations.

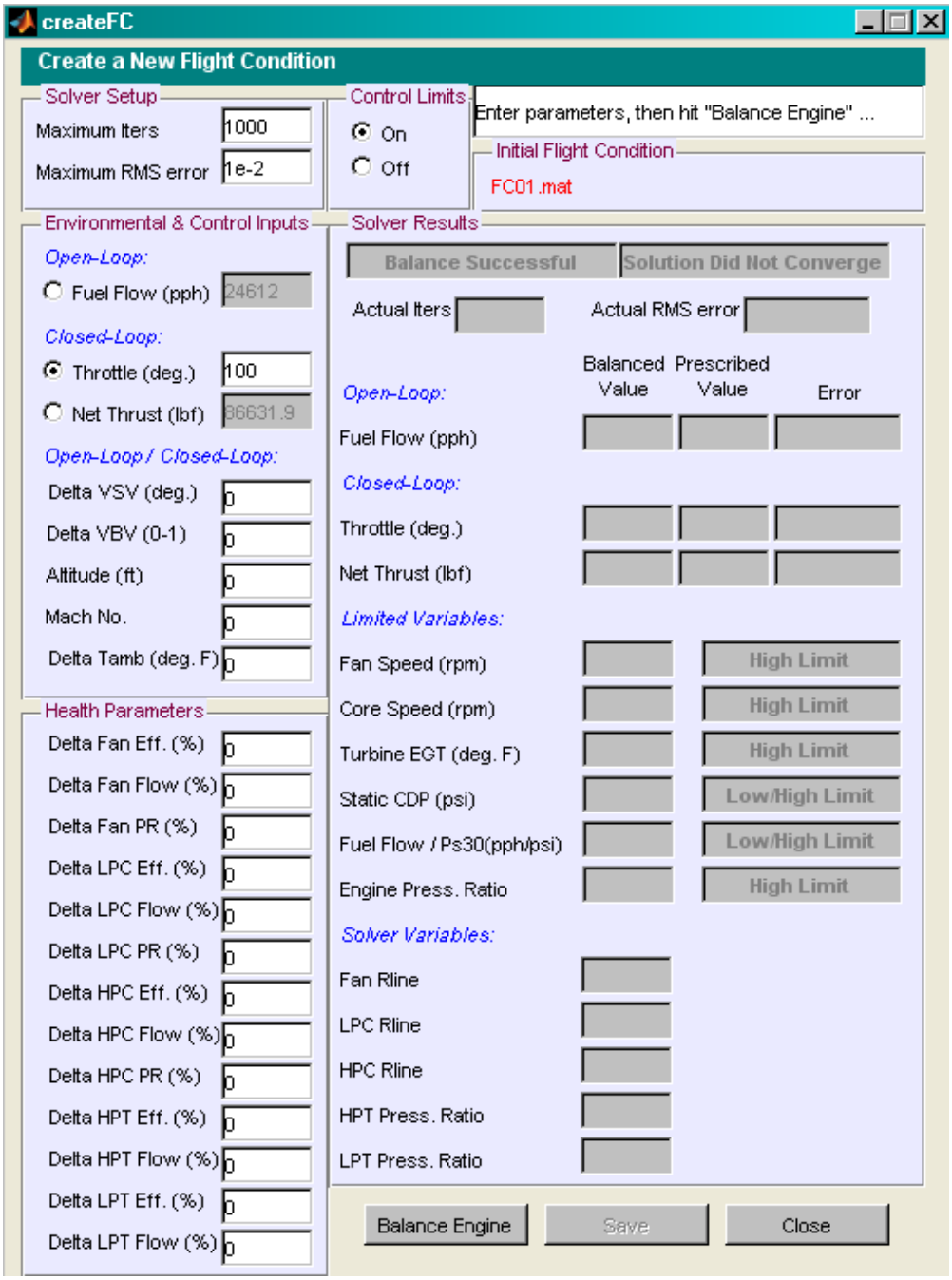

Figure 7.—Solver GUI for creating new flight conditions.

The Control Limits panel activates or deactivates the limit regulators during the balancing procedure. The Solver Setup panel contains fields for the solver termination parameters, maximum iterations and allowable RMS error. Click the Balance Engine button to start the iteration process. The convergence status and calculation results are displayed in the Solver Results panel.

There are several options available if the solver does not converge to a solution on the first attempt. A common reason for unsuccessful convergence is a large discrepancy between the initial and desired flight condition parameters. Thus, consider separating the problem into multiple steps to reduce the difference between the initial and final flight conditions for the solver. For example, if the solver does not converge to a solution for 30,000 ft altitude from an initial condition of 0 ft altitude, try first solving to 10,000 ft, then from there to 20,000 ft, and so on. Occasionally, the solver may also fail to converge because several control limits are actively enforced during the balancing procedure. Therefore, another option is to turn off the limit regulators. It is important to note that, with the regulators deactivated, the solver may converge to flight conditions that violate certain engine safety limits. For example, it is possible to converge to a flight condition that exceeds the maximum fan speed limit by setting Wf to a value higher than that corresponding to a TRA of 100°. However, if the limit regulator option were active, the solver would fail to converge to a solution. Therefore, caution should be exercised when using the steady-state solver without the limit regulators activated. Furthermore, if the solver is not converging successfully because the maximum allowable number of iterations (i.e., Maximum Iters in the Solver Setup panel) is reached, consider increasing this value. Finally, the termination condition for convergence can be relaxed by increasing the maximum allowable RMS error (i.e., Maximum RMS error in the Solver Setup panel).

Flight conditions are stored in a MATLAB structure called ICdata. Click the Save button in the solver GUI to save ICdata to a .MAT file into the default directory, CLM\FC_files. If the solver GUI has been closed, the loaded flight condition can be saved by selecting Save current values under the Flight Condition menu. There is also an option (Save final values) to save the flight condition at the end of a simulation run once it has been completed. However, if this last option is exercised after an open-loop simulation, the TRA value may be invalid. This is because, unlike the steady-state solver, the open-loop engine model does not “back-calculate” TRA given fuel flow rate. Therefore, subsequent use of such a flight condition with closed-loop configurations may result in errors or unexpected results since the closed-loop engine model utilizes TRA as an input parameter.

4.3 Input Profiles

An input profile is the time sequence of the environmental, control, and engine health parameters for the simulation run. C-MAPSS allows the user to both load existing input profiles and create new ones. The profiles, stored as binary .MAT files, contain initial condition data (ICdata) as well as two structures called timevec and datavec. The two structures contain row vectors that define a set of time values and data values, respectively, for all of the model’s inputs (altitude, Mach number, DTamb, fuel flow, DVSV, DVBV, and the 13 health-parameter modifiers). C-MAPSS includes GUIs for creating relatively simple input profiles and loading existing ones. However, since there are no restrictions on the length of an input profile, more complex profiles can be manually created as desired.

To load an existing profile, open the dedicated GUI (Figure 8) by selecting Load from the Input Profile menu option on the main GUI. Available profiles are listed in the first drop-down menu. The profiles are categorized as open-loop or closed-loop and further subdivided as constant or variable. Open-loop profiles utilize fuel flow as the primary input whereas closed-loop profiles utilize TRA. An error dialog will appear if a simulation run is attempted with a profile that is incompatible with the loaded model (e.g., an open-loop profile for a closed-loop engine). Variable input profiles contain inputs to the engine that change relative to time. Conversely, the inputs in constant profiles do not change with respect to time.

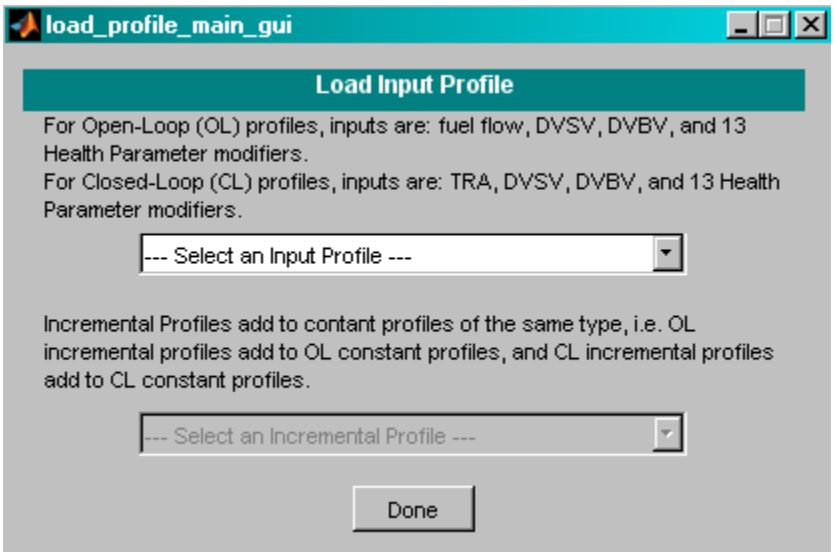

Figure 8.—GUI for loading existing input profiles.

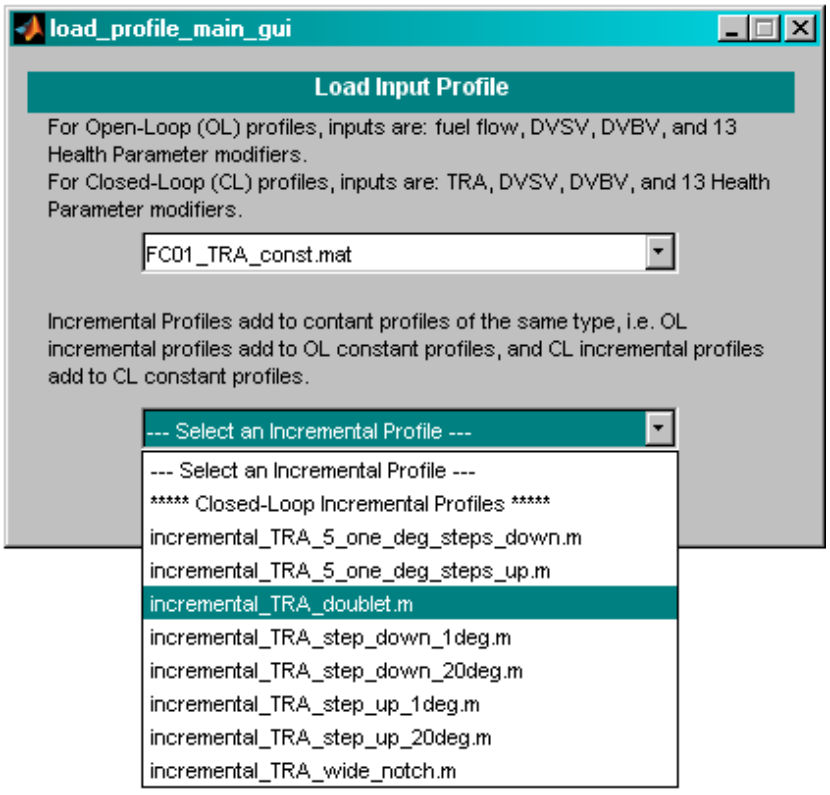

Figure 9.—Selecting incremental profile on top of a constant profile.

Constant profiles are designed to be used in conjunction with so-called “incremental profiles.” By selecting a constant profile, the second drop-down menu, which contains the available incremental profiles, is enabled (Figure 9). Incremental profiles add variation to a selected constant profile. For instance, by selecting a constant profile of 50° TRA and an incremental profile of a step increase of 20° TRA, the resultant profile is a step increase in TRA from 50° to 70°. The numerous permutations achievable by combining constant profiles with incremental profiles provide the user with easy access to a rich variety of input profiles.

To create a profile using the GUI, first select a flight condition. Next, select `Create` from the `Input Profile` menu option. Two options are available: (i) Open-loop profile and (ii) Closed-loop profile. Both choices result in the same GUI appearing, but with different initializing data. Figure 10 shows the GUI if `Closed-loop profile` is selected. Note that the fields for fuel flow are unavailable. Accordingly, fields for throttle will be unavailable if `Open-loop profile` were selected. The user can choose among three types of inputs: synchronous step, asynchronous step, and synchronous ramp. Synchronous inputs mean all engine inputs change at the same time. For an asynchronous step, the user is able to select when each input changes using the `Step Time` fields. The starting values correspond to the flight condition selected. The user is free to change the ending values of the environmental, control, and health parameter inputs. Once the settings have been accepted, the user can save the input profile into a .MAT file. By default, this file will be stored in `CLM\FT_files` directory.

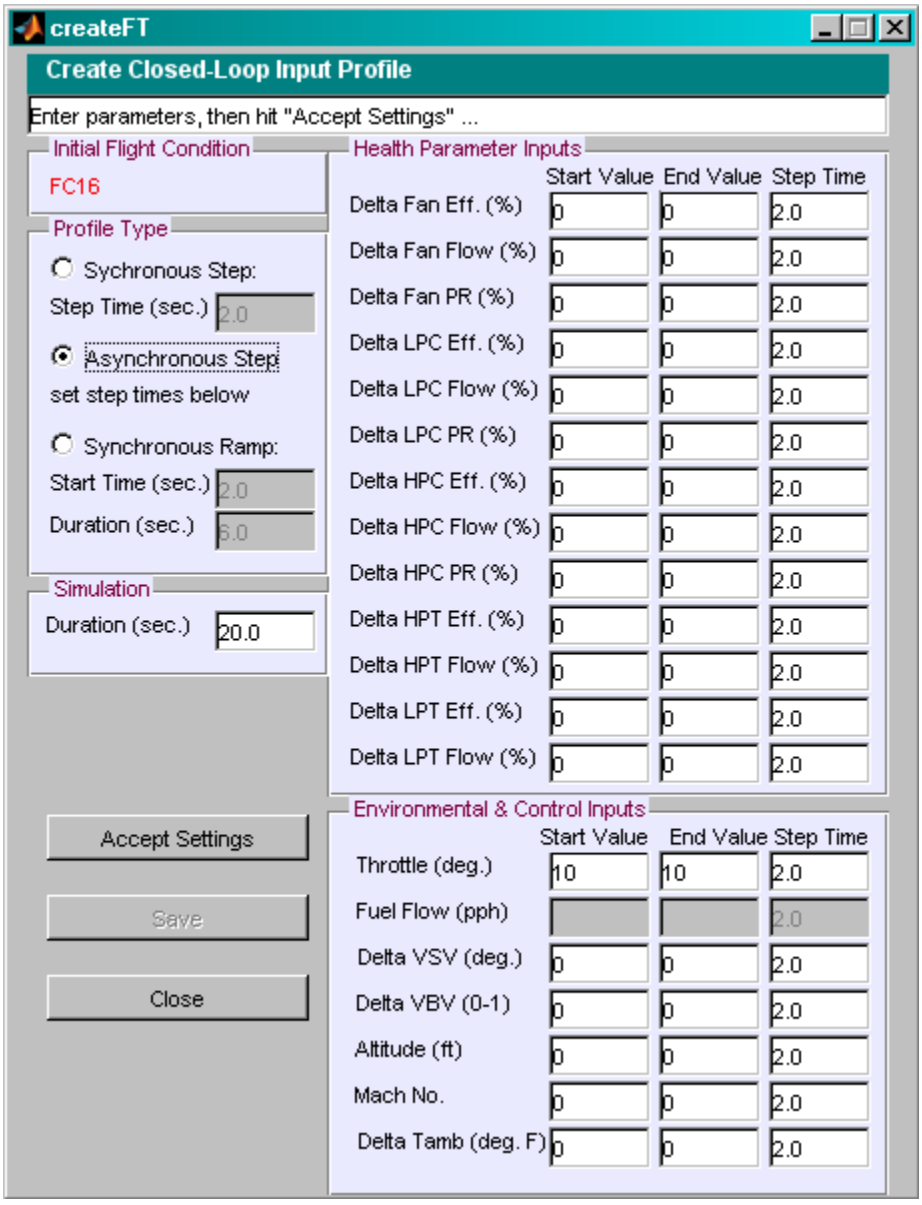

Figure 10.—GUI for creating new input profiles.

In order to create more complex input profiles that are beyond the capabilities of the GUI, the relevant arrays in the MATLAB workspace must be directly modified. An input profile is stored as two separate structure arrays: `timevec` and `datavec`. The `timevec` structure contains arrays of time values for the various input parameters, while the `datavec` structure contains the corresponding values of these parameters. For example, consider a 60-sec input profile with a step increase in TRA from 50 to 100 at 10 sec, then a 10-sec ramp decrease in TRA back to 50 at 30 sec. Such a profile would be implemented by: `timevec.TRA = [0,10,10.015,30,40,60]` and `datavec.TRA = [50,50,100,100,50,50]`. Note that a step input is implemented by incrementing time by a value equal to or less than the simulation time step (0.015 sec).

4.4 Viewing Simulation Results

Once all parameters necessary for executing a run have been initialized, run C-MAPSS by clicking the `Run Simulation` button or the "Play" icon on the Simulink diagram. Once the run is completed, there are several methods of accessing the simulation results. Several built-in plot routines are available from the `Plot` menu option on the main GUI:

- `6 variables`: fuel flow, fan speed, engine pressure ratio, core acceleration, fuel flow derivative, net thrust
- `9 variables`: throttle, fan speed, fuel flow, ratio of fuel flow to burner inlet static pressure, core speed, HPT exit temperature, burner inlet static pressure, engine pressure ratio, net thrust
- `Fan speed`: demand, actual, limit
- `Fuel flow and derivative`: fuel flow, fuel flow derivative, core acceleration
- `Map parameters`: fuel flow, fan/LPC/HPC R-line map parameters, HPT/LPT PR map parameters
- `Solver errors`: fuel flow, component inlet/exit flow errors
- `Iterations`: fuel flow, iterations per time step
- `Stall margins`: fan/LPC/HPC stall margin values

There are several other methods of accessing the model outputs. For real-time viewing of the engine parameters as the model runs, displays and/or scopes can be connected to any of the signals within the Simulink diagram. Additionally, since many simulation outputs are saved to the MATLAB workspace after a run, the output variables may be directly accessed via the command prompt, functions, or scripts. Since all output variables are functions of time, they are stored as independent, one-dimensional, column vectors. A list and description of important variables are provided in the Appendix. Finally, it is important to note that any signal or mathematical combination of signals in the C-MAPSS model can be saved to the MATLAB workspace. Native Simulink blocks (e.g., `To Workspace`) are available to save model output data to a file or the MATLAB workspace in several different formats. More information is available through the MATLAB/Simulink help files and The MathWorks website (Ref. 4).

5.0 Linearization and Controller Design

In addition to the simulation of a commercial-type aircraft engine, the C-MAPSS software consists of several tools for model linearization and controller design. These tools are accessed through the main GUI using the `Linear Model` menu option. This section assumes the user has a basic understanding of linear dynamical systems theory.

5.1 Linear Engine Models

Linear models are commonly used for control system design. C-MAPSS allows the user to linearize the nonlinear open-loop engine simulation into a continuous linear state-space model. Let *x* be a vector containing the state variables, *u* be a vector containing the input variables, and *y* be a vector containing the output variables. Moreover, let the subscript 0 denote the equilibrium condition corresponding to the point at which the state-space model has been linearized. The state-space parameters are then defined as perturbations about their respective equilibrium points:

$$\Delta x = x - x_0$$
$$\Delta u = u - u_0$$
$$\Delta y = y - y_0$$

With some abuse of notation, drop Δ and redefine *x*, *u*, and *y* as the perturbed values about the equilibrium condition. Noting that $\Delta\dot{x} = \dot{x} - \dot{x}_0 = \dot{x}$, the linear state-space model can then be written in its familiar form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Thus, a linear model is represented by its state-space matrices (*A*, *B*, *C*, *D*) and its equilibrium (trim) point (x_0, u_0, y_0). For C-MAPSS, the trim point is the flight condition. The state vector consists of fan speed and core speed. The input vector consists of fuel flow, VSV, VBV, and 13 health parameters (Table 2 in the Appendix). The output vector elements are listed in Table 3 in the Appendix.

C-MAPSS includes several linear engine models that are readily accessible. To load an existing linear model, select `Load` from the `Linear Model` menu option, and select from the list in the GUI window that opens (Figure 11). The linear models are located in the directory `CLM\LEM_files`. Each file consists of two MATLAB objects: `SSeng_16x27_unsc` contains the four state-space matrices and `ICdata` contains the trim conditions. When the user selects a file, it will be loaded into the workspace. The GUI window will be closed, and the name of the linear model will appear in the `Linear Engine Model` panel in the upper-right corner of the top-level GUI.

To create a new linear model, first load or create a flight condition about which the linearization will take place. Once flight conditions are set, select `Create` from the `Linear Model` menu option. This will open the linearization GUI (Figure 12). The linearization routine consists of a dedicated Simulink model (Figure 13) that contains a copy of the nonlinear open-loop engine model. The state-space matrices essentially consist of partial derivatives relating various pairs of engine variables. For example, the elements of *C* are partial derivatives of the output variables with respect to the state variables. The linearization algorithm computes these partial derivatives using the nonlinear engine model by perturbing the variables from their equilibrium values one at a time, and then determining the resulting effects on the rest of the variables once mass flows are balanced throughout the engine. The linearization GUI allows the user to modify the perturbation magnitudes (in units of the respective variable, not percentage change) and the simulation duration to ensure mass flow balance. However, the default values should be satisfactory for most applications.

Click the `Linearize` button to start the linearization routine. The MATLAB command window will update with the progress of the linearization process. When the process has completed, the elements of the state-space matrices and the eigenvalues of *A* will be displayed in the command window. The message `Completed build of linear model` will appear in the status box near the top of the GUI window. The `Save` button on the GUI will also become active at this point. Clicking `Save` will create a binary .MAT file (in `CLM\LEM_files` by default) containing the state-space matrices (as a state-space system MATLAB object) and the equilibrium condition (`ICdata` structure).

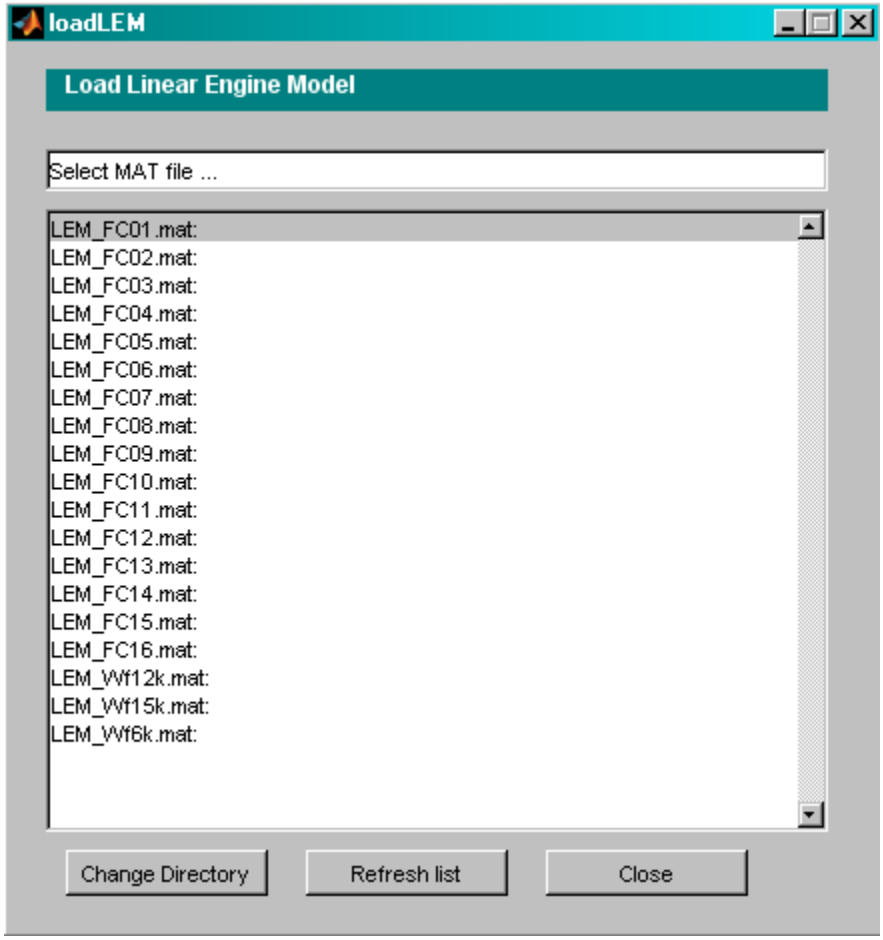

Figure 11.—Loading an existing linear model.

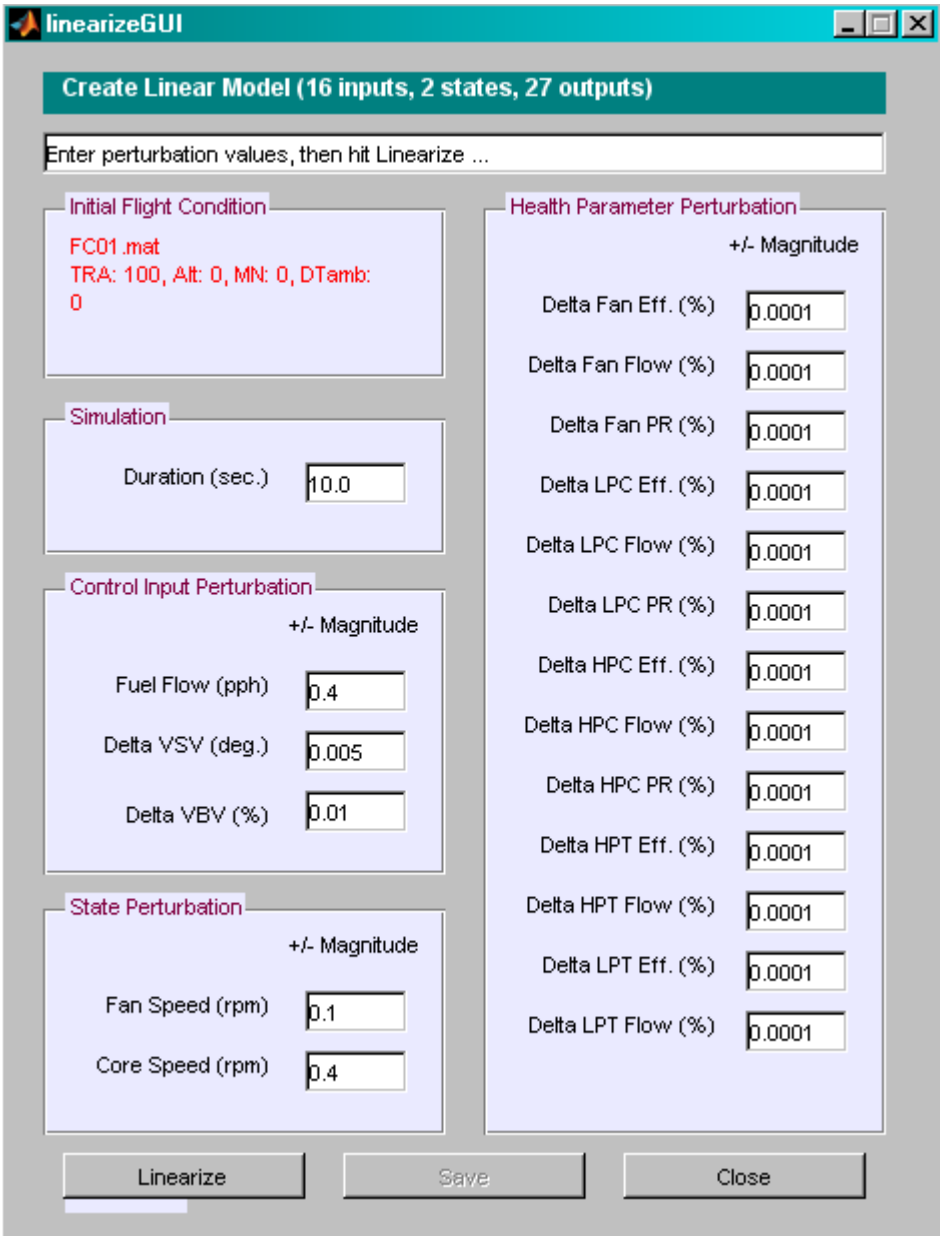

Figure 12.—Creating a new linear engine model.

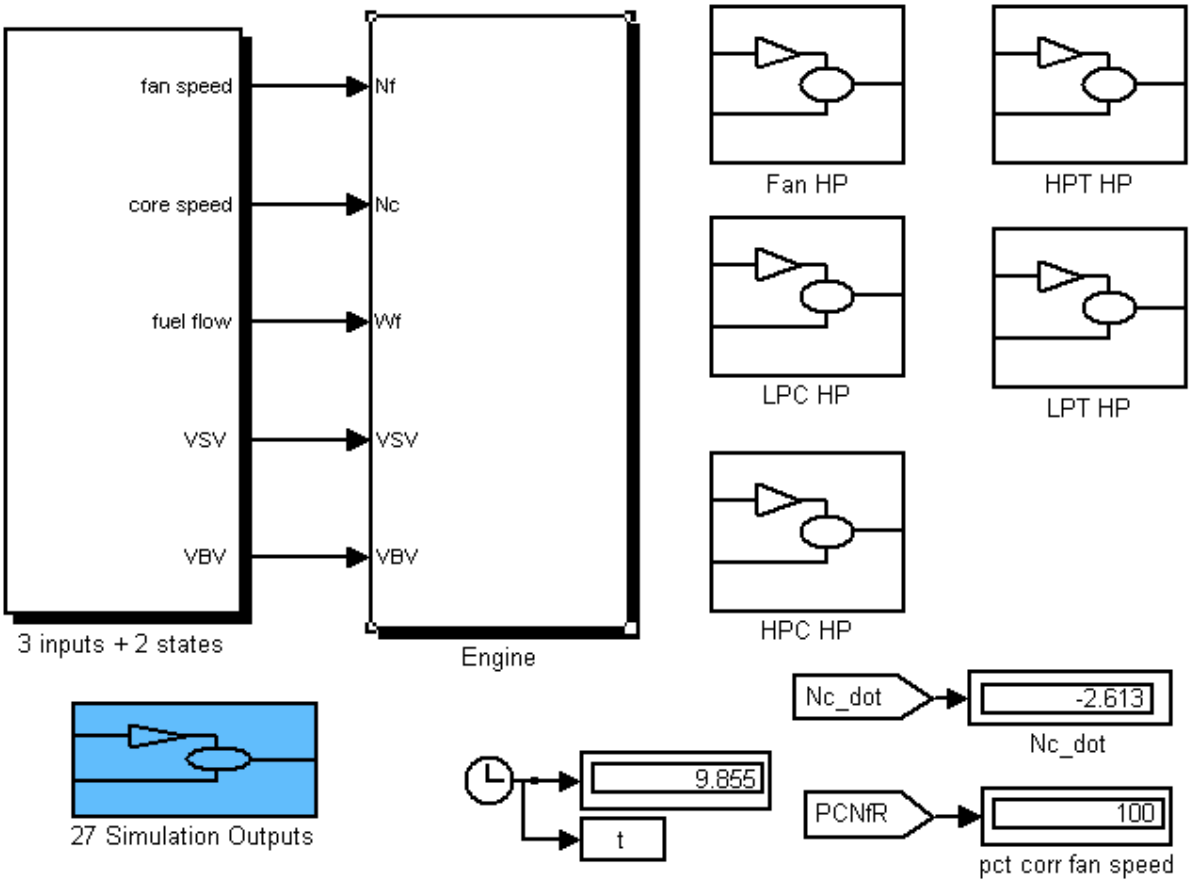

Figure 13.—Simulink model used for linearization routine.

Once a new linear model has been created or an existing one has been loaded, the user can perform some basic analysis by choosing `Analyze LEM` from the `Linear Model` menu option. This will open the GUI shown in Figure 14. The upper-left panel contains buttons for carrying out five different types of analysis for a simplified single-input/single-output (SISO) version of the linear model (with incremental fuel flow as input and incremental fan speed as output). Text results are shown in the display panel on the right half of the window. The five options are:

- `state-space model`: Display the *A*, *B*, *C*, and *D* matrices of the SISO linear model.
- `step response`: Compute and plot the step response.
- `bode plot`: Draw a Bode plot for the magnitude and phase angle of the linear model's frequency response.
- `poles and zeros`: Display the poles and zeros of the transfer function from fuel flow to fan speed.
- `transfer function`: Display the transfer function in numerical form.

The drop-down menu in the middle of the GUI shown in Figure 14 can be used to analyze the full multi-input/multi-output (MIMO) linear model. Select one of the 16 inputs (fuel flow, DVSV, DVBV, health parameters). Click `run simulation` to generate the linear model response to a doublet-shaped input (step-increase followed by step-decrease of the input parameter about its initial value). The bottom panel has four buttons that produce plots of the inputs and key outputs when clicked. It is important to note that, when dealing with linear models, all of these variables (both inputs and outputs) are incremental relative to the respective flight condition.

C-MAPSS also provides tools for comparing the linear model with the nonlinear engine model. To do so, select `Compare with CLM` from the `Linear Model` menu option, which will open the GUI shown in Figure 15. The drop-down menu functions in the same way as the one in Figure 14 to select the input variable for the comparison. Clicking `run simulation` will run the Simulink file shown in Figure 16, which can be opened by checking the `show Simulink model` box. This Simulink diagram contains both linear and nonlinear engine models. Furthermore, the input to the nonlinear model is the sum of the trimmed values and the incremental values seen by the linear model. The comparison plot routines are written such that the linear model outputs (incremental variables) are added to the respective trimmed values. Hence, by examining these comparison plots, the user can quickly determine whether the linear model is a good representation of the nonlinear model.

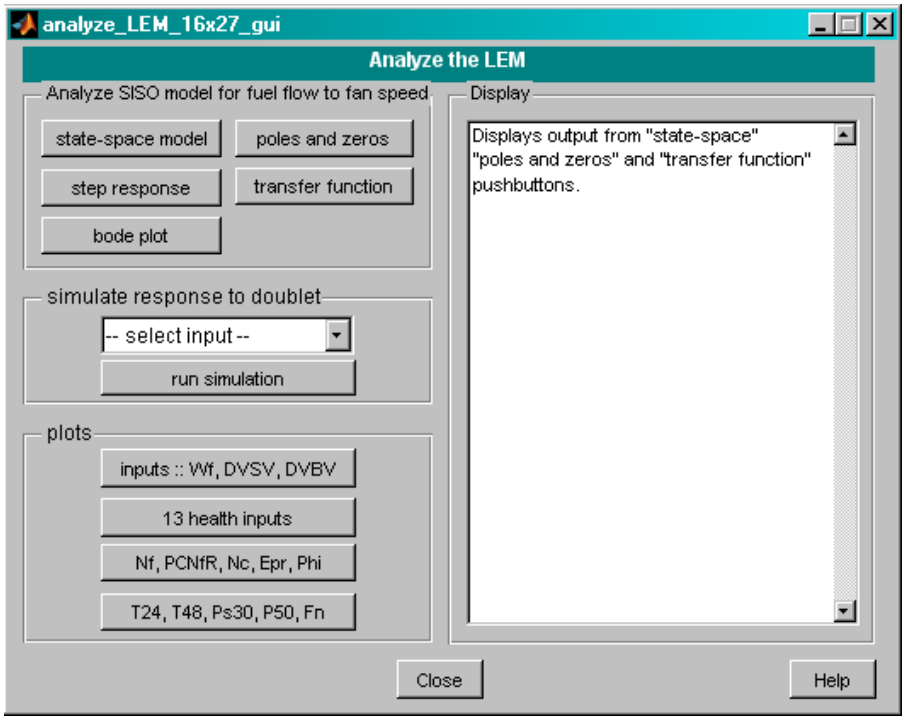

Figure 14.—Analyze a linear model.

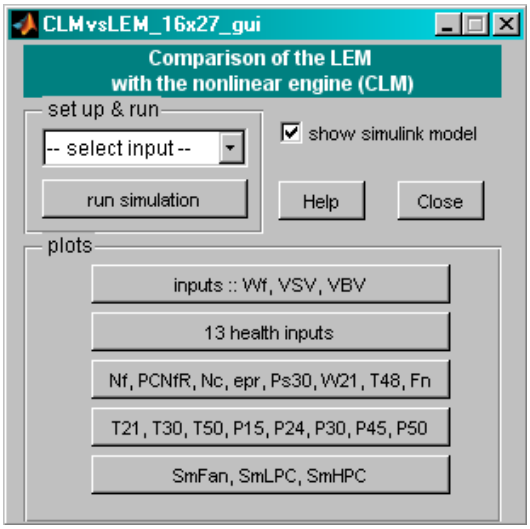

Figure 15.—Comparing linear model and nonlinear model.

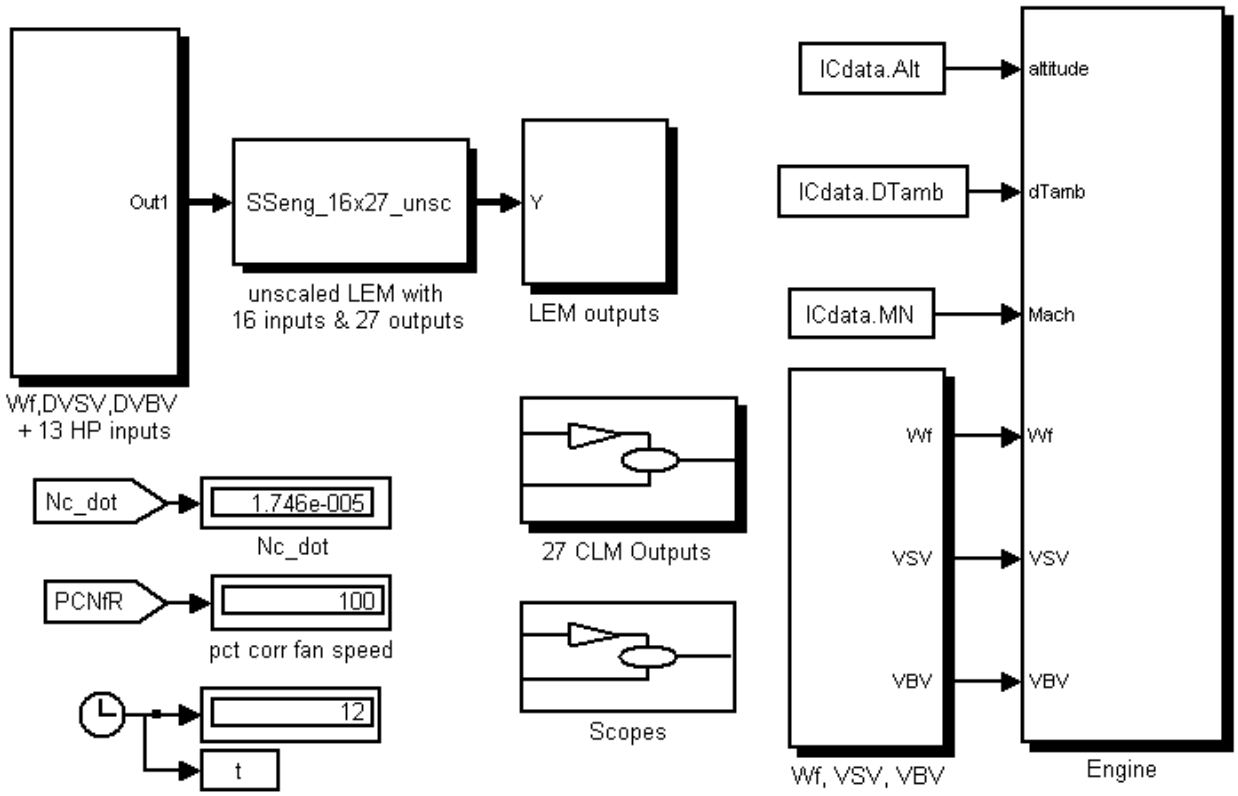

Figure 16.—Simulink model used to compare linear model with nonlinear engine model.

5.2 Controller Design

With a newly created or existing linear engine model, C-MAPSS can be used to design customized controllers for the engine. Since these "point" controllers are designed from a linear model, their intended usage is localized around the same flight condition as the linear model. To use the GUI to create a new point controller, a flight condition must have been defined and a linear model corresponding to that flight condition either loaded or created, though not necessarily saved. Once the linear model is in the MATLAB workspace, select `Design Controller` from the `Linear Model` menu to open the controller design GUI (Figure 17). Using the drop-down menu, select the engine variable to be controlled:

- Fan speed (Nf)
- Core speed (Nc)
- Core acceleration (Nc_dot)
- HPT exit temperature (T48)
- Max. combustor static pressure (high Ps30)
- Min. combustor static pressure (low Ps30)
- Engine pressure ratio (EPR)

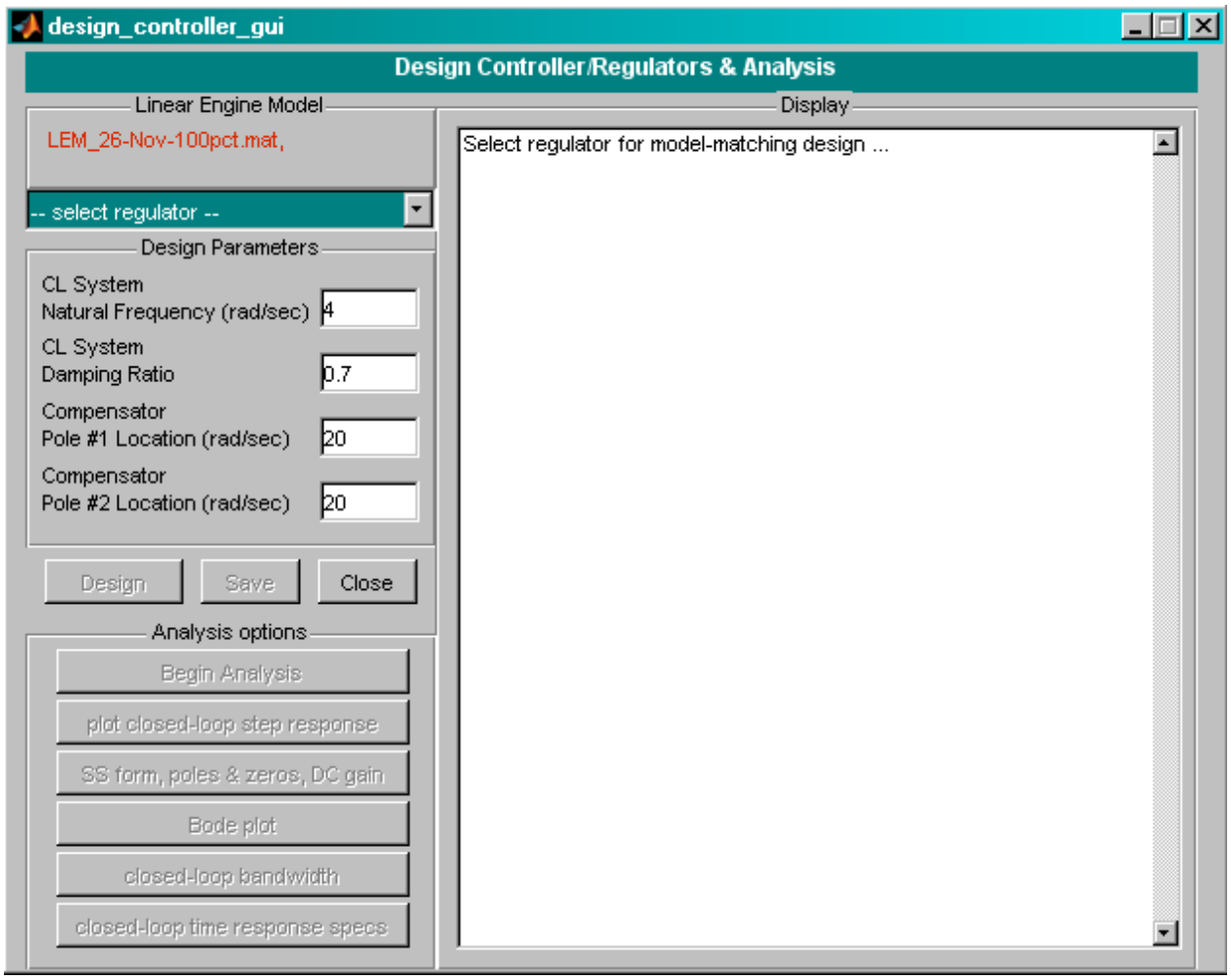

Figure 17.—Designing a custom controller.

Controller design parameters, such as the natural frequency and damping ratio of the closed-loop system and compensator pole locations, can be modified. Once set, click the `Design` button to start the controller design process. The controller is designed using Edmunds' model-matching method, which is described in Section 7.2 of Reference 5. Figure 18 shows the GUI after executing a typical design process. In this case, a T48 regulator was selected, with the desired natural frequency and damping ratio set to 3 rad/s and 0.9, respectively. Once the design process is complete, the display window (which mirrors the MATLAB command prompt), shows a summary of the controller parameters. The first part of the message is the value of the norm of the error in the design calculations. The lower this number is, the better the step response of the actual closed-loop linear model matches that of the ideal closed-loop system that the user specified by selecting the natural frequency and damping ratio values. Also shown in the command window is the structure `DesignData`, which contains key information about the design:

- Flight condition of the linear model that was used
- Natural frequency (Wn) and damping ratio (zeta) of the ideal closed-loop system
- Poles of the controller, as selected by the user
- Norm of the error in the design
- Type of regulator

The display also shows the names of the controller in the MATLAB workspace.

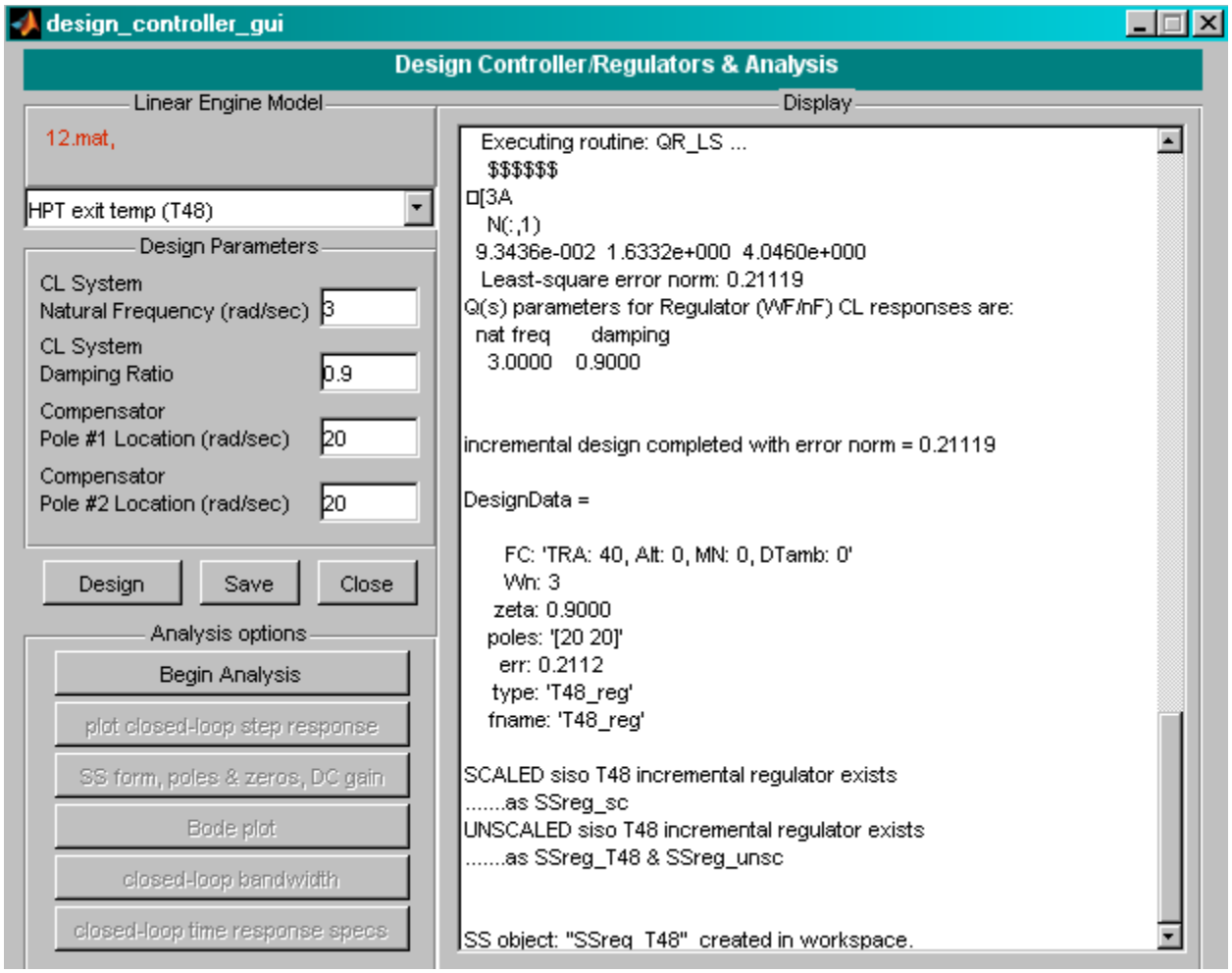

Figure 18.—Sample controller design. Display window shows status of controller design.

With the design process completed, the user can save the controller, close the controller design GUI, or perform additional controller analysis. To conduct further analysis, first click the `Begin Analysis` button on the `Analysis options` panel. This will set up the analysis process and activate the remaining five buttons, which will allow the user to quickly determine the open- and closed-loop properties of the controller in both numerical and graphical form. The results will be plotted and shown on the status display as appropriate. If the design is satisfactory, the controller can be saved with the `Save` button. If the controller design is not satisfactory (e.g., the overshoot of the step response is too high, the response is too slow, etc.), the design parameters (natural frequency, damping ratio, controller poles) can be changed for another iteration. The saved regulator .MAT files are located in separate folders for each type of regulator in the directory `CLsim\Reg_files`. For example, all of the fan-speed controllers are located in `CLsim\Reg_files\fan_speed`.

5.3 Using Custom Controllers

Using the C-MAPSS GUI, custom-designed linear controllers can be readily applied to the nonlinear engine model. A fan speed controller designed using the process described in Section 5.2 can be used with the nonlinear engine model by selecting the “switched” model and choosing the `Point fan speed controller` option (refer to Section 4.1 for details on using the switched model). However, this option can be exercised with any linear controller regardless of the design process so long as the

controller parameters are saved in a compatible format. Namely, the controller must be stored in the same manner as those included in the directory `CLsim\Reg_files`. The controller must be a single-input/single-output (SISO) MATLAB state-space object saved as a single .MAT file. The .MAT file should also include a structure called `DesignData` that contains a description of the controller (see Section 5.2). Although not required, this structure is used by the C-MAPSS GUI to display the controller parameters. The controller can then be applied to the nonlinear engine model using the aforementioned process.

The C-MAPSS engine models are designed with a modular structure to facilitate modification or replacement of the default control system. The engine-only portion of the simulation requires a fixed number of inputs to operate properly: three environmental parameters (altitude, Mach number, deviation from standard ambient temperature), 13 health parameters, and three control parameters (fuel flow, VSV, VBV). Engine operation is independent of the method by which these variables are specified. Thus, the advanced user can easily modify the default control system or replace it with a variety of custom-designed control architectures (e.g., nonlinear control, model-predictive control, etc.). In the default control system, fan speed is controlled using fuel flow, and VSV and VBV are scheduled. An advanced user might want to modify the fan speed control, leaving VSV and VBV scheduled. This could be accomplished by modifying the Simulink diagram shown in Figure 2 in the `CM2_SADL_Ncd_IS` model. To introduce a different method of controlling VSV and VBV into C-MAPSS, the `scheduled VSV and VBV` block in the model `CM2_SADL_Ncd_IS` must be replaced. Alternatively, it may be more convenient to work from the open-loop engine model `CM2_OL_IS` if one wants to connect the engine to a novel control system that is significantly different from the default one. Development of these sophisticated customized models requires direct manipulation of Simulink files and creation of new Simulink models, and is clearly beyond the scope of the C-MAPSS GUI. However, if these custom models contain the requisite inputs, the GUI can still be used to load and run the simulation by selecting the `Custom Model` option from the `Model/Controller Selection` menu.

6.0 Examples

This section provides examples of how to use most of the features of C-MAPSS. For each case, it is assumed that the user has appropriately set up C-MAPSS by running the `setup_everything` script. Thus, the top-level GUI should be open with the default model active.

6.1 Load Input Profiles

The quickest and simplest way to run a simulation using C-MAPSS is to load an existing input profile. To do so:

1) Select the `Input Profile` menu option, then `Load`.
2) Select the desired input profile from the drop-down menu. In this case, we will select `FC10_ramp_TRA77_4Kft_M0p25` under the closed-loop variable profiles category. Note that the main GUI has updated automatically to reflect the features of this profile.
3) Click `Done` to close the load profile GUI.
4) Click `Run Simulation` to start the run.
5) Once completed, use the `Plot` menu option to quickly access the results. Figure 19 shows the nine-variable plot as an example.

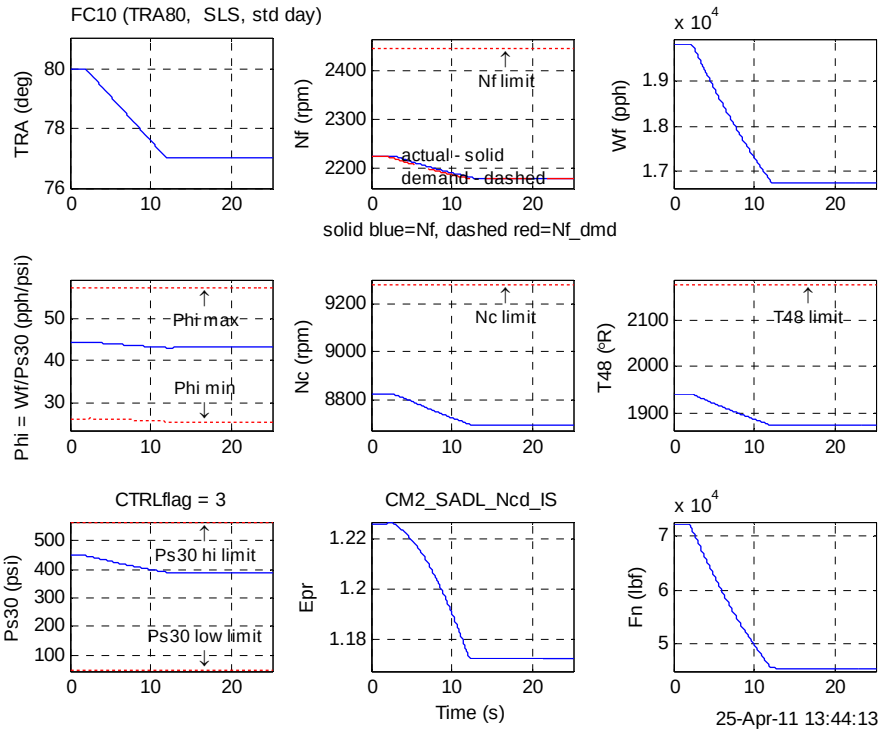

Figure 19.—Nine-variable plot after running an existing input profile (`FC10_ramp_TRA77_4Kft_M0p25`).

6.2 Create Flight Conditions and Input Profiles

Suppose now that we wish to run a transient case that is not available from the list of included input profiles. Additionally, we wish to begin the transient at a flight condition that is not included with C-MAPSS. In this example, let the starting point of the profile be at 85° TRA, 24000 ft altitude, Mach 0.60. The conditions then change linearly (i.e., ramp) to 75° TRA, 22000 ft altitude, Mach 0.65 in 4 sec, starting at the 5-sec mark.

1) To create a flight condition, first load a (preferably similar) flight condition. Select `Load` from `Flight Condition`, and choose `FC06`, which corresponds to 100° TRA, 20000 ft altitude, Mach 0.70.
2) Now, open the steady-state solver GUI by selecting `Create > Using steady-state solver` from the `Flight Condition` menu option.
3) Enter the desired initial values of TRA, altitude, and Mach Number into the appropriate fields. Click `Balance Engine`. The results should be similar to those shown in Figure 20. The results can be saved for future use, though this is not a necessary step in creating an input profile.
4) To create the profile, close the solver GUI, and choose `Create > Closed-loop profile` from the `Input Profile` menu.
5) Select `Synchronous Ramp`. Set start time and duration as desired; in this case, 5 and 4 sec, respectively. We may leave the total simulation duration at 20 sec.
6) Set the end values of TRA, altitude, and Mach Number. Note that the start values correspond to the newly created flight condition.
7) Click `Accept Settings`. Note that the main GUI updates with the newly created profile (see Figure 21).
8) Save the profile if so desired and click `Close`. Run the simulation and view results. Figure 22 shows the `9 variables` plot.

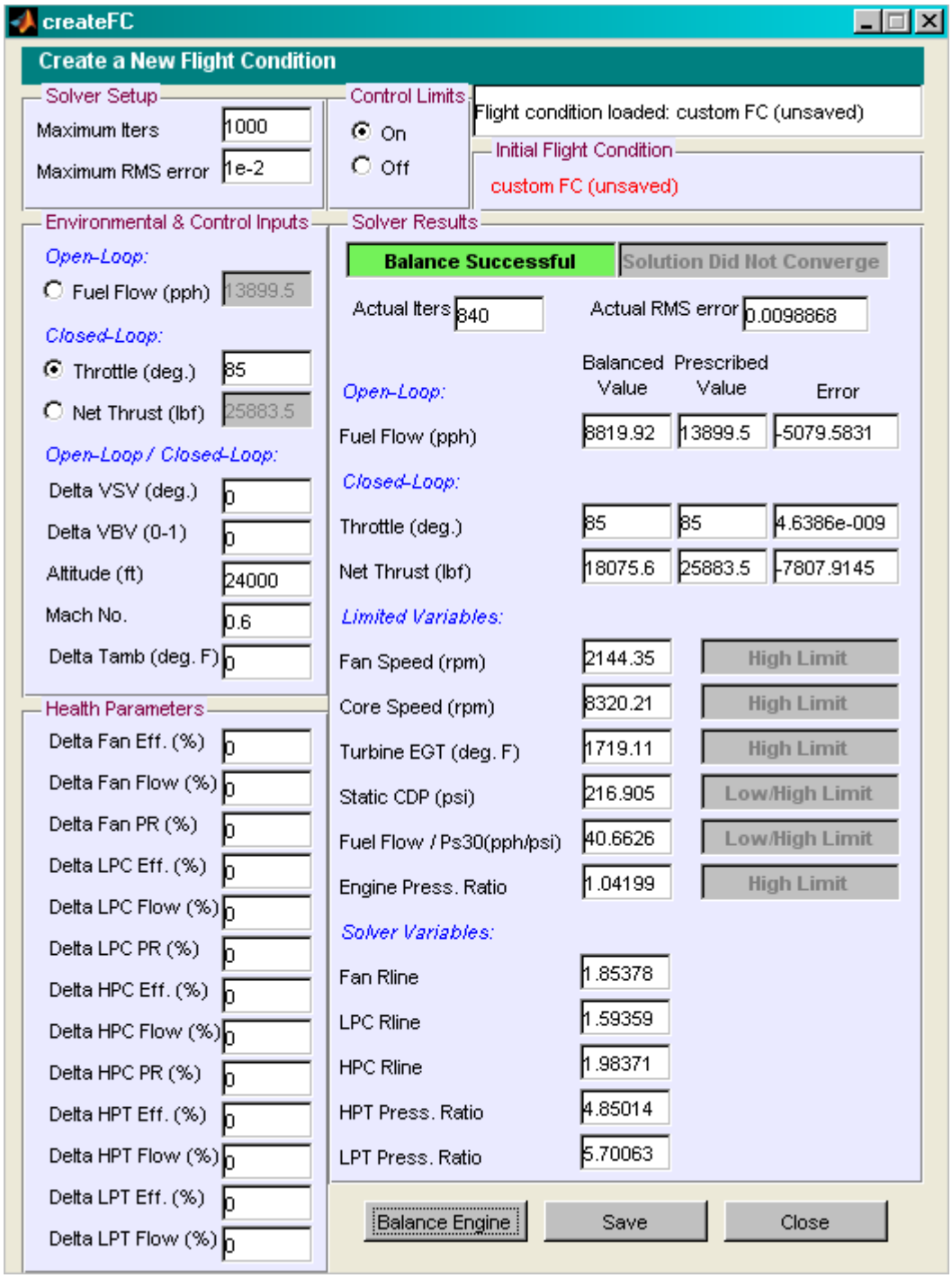

Figure 20.—Results after creating a new flight condition using the steady-state solver GUI.

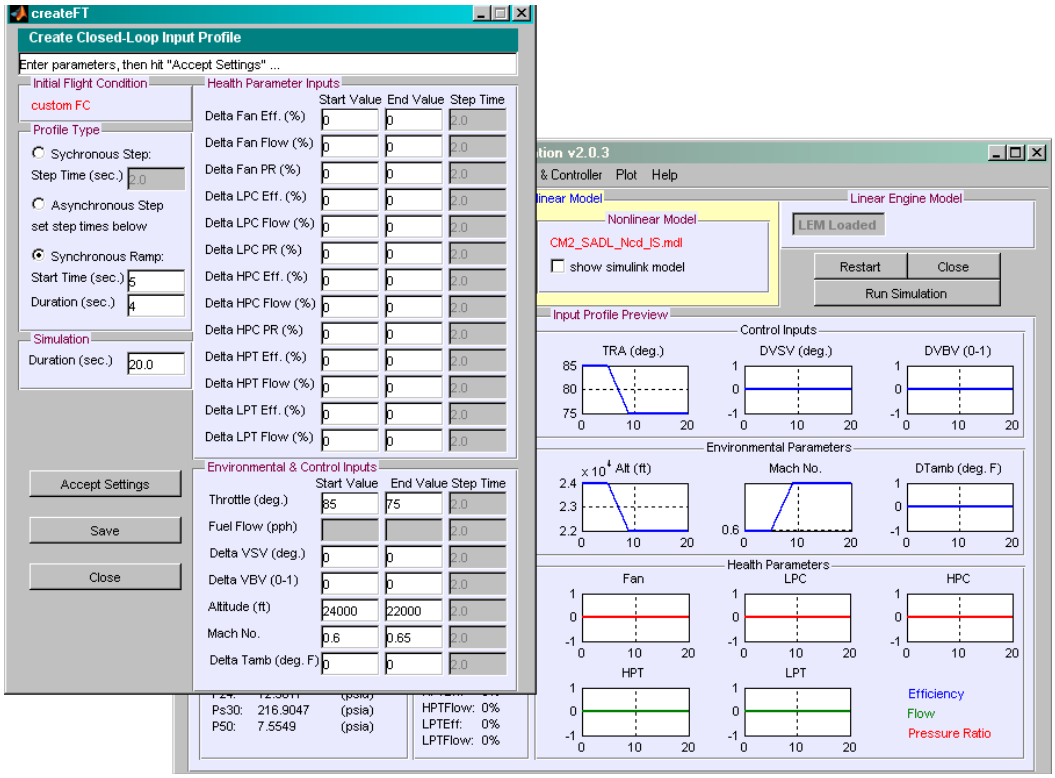

Figure 21.—Results after creating a new input profile.

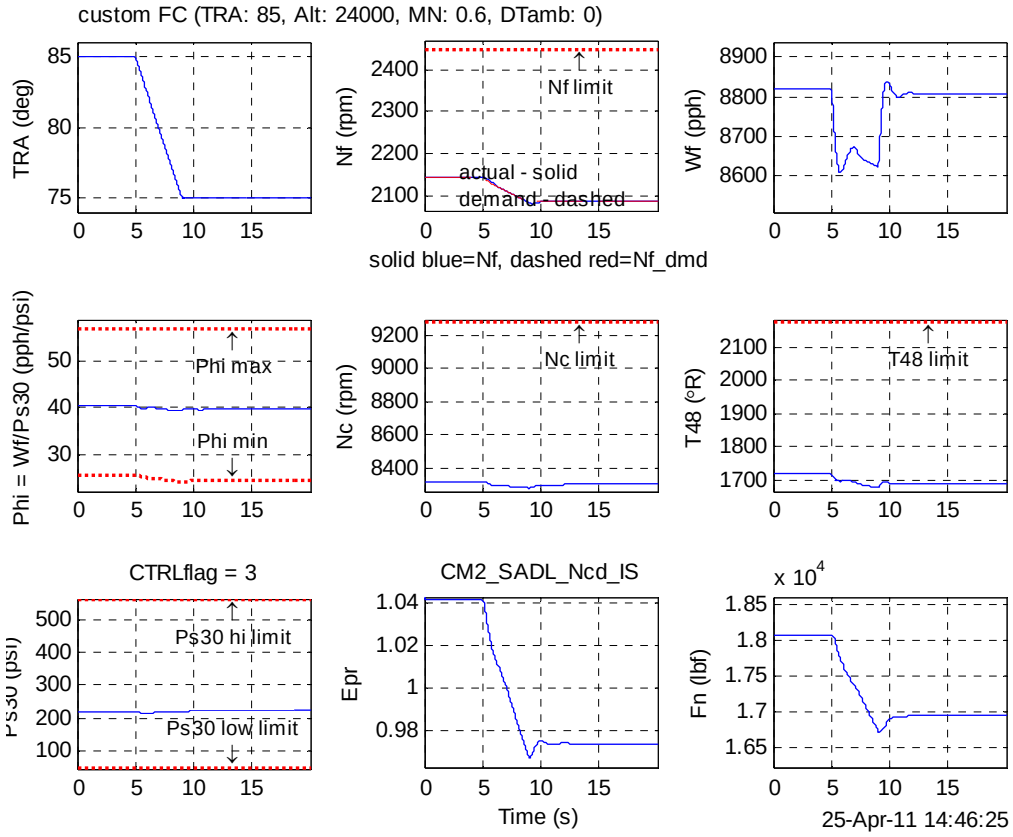

Figure 22.—Nine-variable plot after running newly created input profile.

6.3 Incremental Profiles

This section describes how to utilize the incremental profile functionality in C-MAPSS. The process will demonstrate the versatility of using incremental profiles over fixed input profiles in certain applications. For this example, suppose we wish to evaluate how a doublet fuel flow input affects an open-loop engine for two different flight conditions.

1) First, we click the `Restart` button. This will close and reopen the main GUI and, more importantly, clear the workspace of results from the previous run to prevent confusion.
2) Select the open-loop engine model (`CM2_OL_IS`) from the `Nonlinear Model & Controller` menu option.
3) Select `Load` from the `Input Profile` option.
4) Choose a constant, open-loop profile. In this case, we will use `FT_Wf12K_const`.
5) Choose an incremental profile that will add to this constant profile. In this example, we choose the `incremental_Wf_doublet.m` incremental profile. Click `Done`. Notice from the updated plots shown in the top-level GUI that the doublet increment is superimposed onto the constant profile.
6) Use the `Run Simulation` button to run the engine model with the specified input. Then, use the `Plot` menu option to view results. Figure 23 shows the plots created using the `6 variables` submenu selection.
7) Now, suppose we wish to analyze the same doublet command, but around a different fuel flow. To demonstrate, load the `FT_Wf15K_const` constant profile. There is no need to restart C-MAPSS.
8) Again, select the `incremental_Wf_doublet.m` incremental profile. Note that the doublet is now around the 15,000 pph fuel flow level.
9) Run and view the results. Figure 24 shows the `6 variables` plot for this second case.

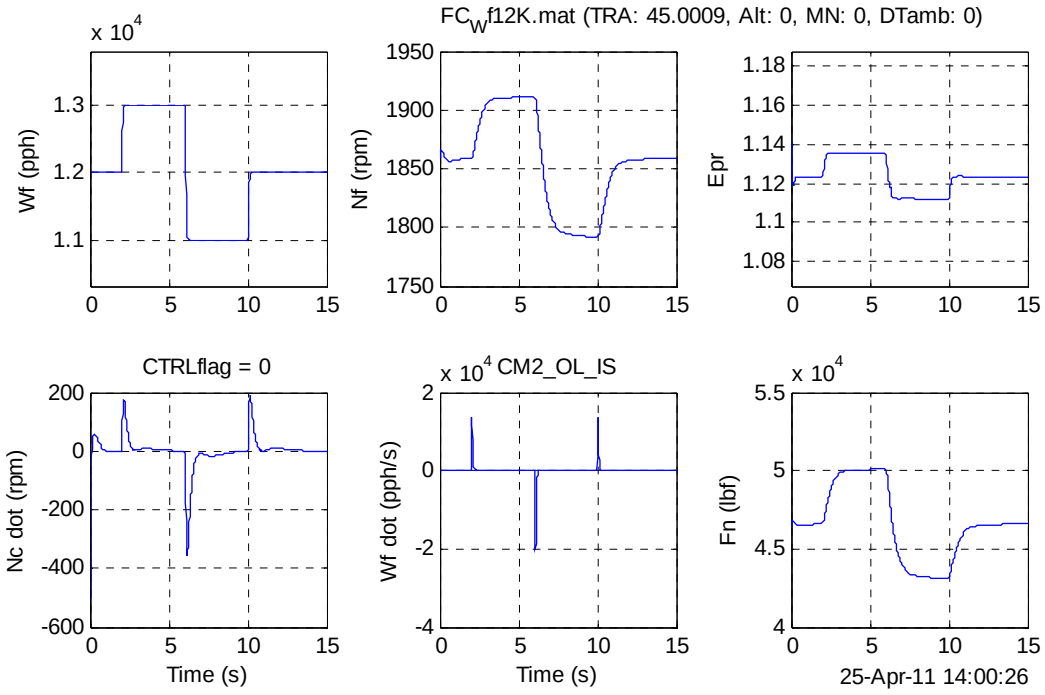

Figure 23.—Six-variable plot for open-loop incremental profile (doublet at 12,000 pph fuel flow).

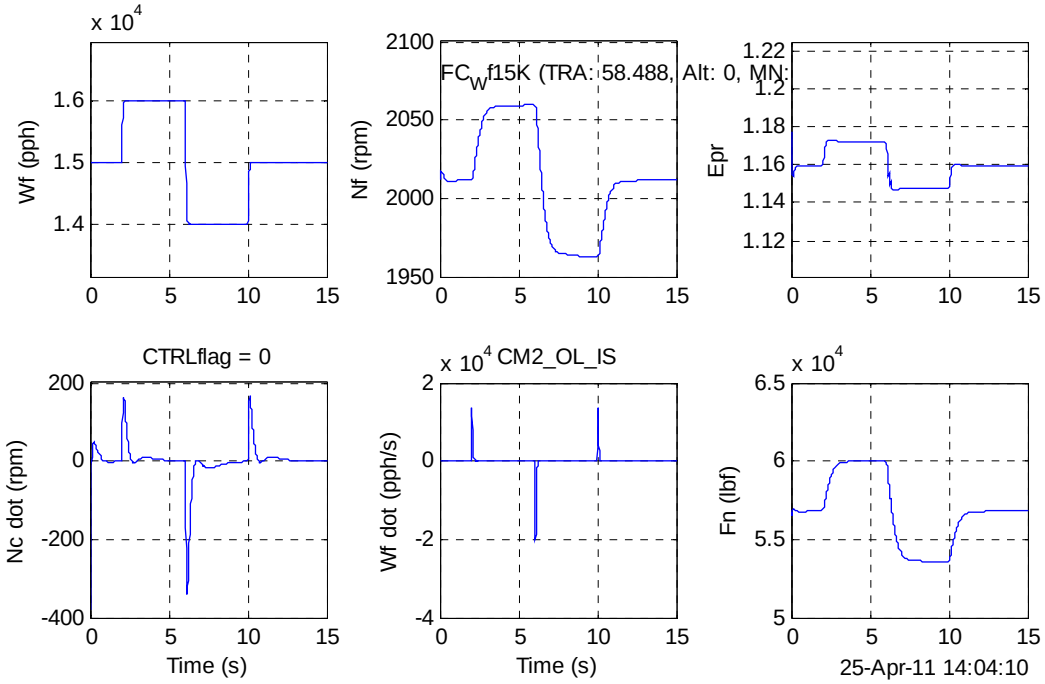

Figure 24.—Six-variable plot for open-loop incremental profile (doublet at 15,000 pph fuel flow).

As this example shows, the incremental profile function provides an efficient method to simulate many transient runs that have similar input profiles but different flight conditions. If the desired flight conditions do not exist, they simply need to be created and saved using the steady-state solver. There is no need to additionally create a new variable profile each time.

6.4 Iterative Solver Settings

As described previously, the engine portion of each C-MAPSS Simulink model is iterated within each time step to maintain consistency of flow rates into and out of each of the five rotating components. By default, the maximum number of iterations per time step is 100 and the allowable flow rate error for each rotating component is 2 percent. For the majority of cases, the default settings are sufficient. Nevertheless, the more advanced user may deem it necessary to change the iterative solver settings to better suit particular applications. This section describes how to analyze iterative solver performance and change solver settings.

To the illustrate the process, we use a transient profile of 20-sec duration at sea-level, static conditions, and include a large step change in TRA from 0° to 100° at the 5-sec mark. Since this profile does not exist as a pre-packaged profile, it must first be created. Therefore, use the procedures described in Section 6.2 to create this input profile. The flight condition, `FC14` (sea-level static), may be used as the starting point. Once the input profile is created, run the simulation and select the `Solver errors` option from the `Plot` menu. A set of plots similar to those shown in Figure 25 will appear. These plots show the evolution of the flow errors. It is important to note that the flow errors are plotted against solver iterations, which is *not* proportional to time since the number of iterations may vary with each time step. The oscillatory behavior is due to the flow errors being large at each time step before any iterations occur, and then driven down as the solver iterates. In this case, since the default solver settings are unchanged, all errors (most notably error 3, which represents difference in flow rates at the LPT exit and core nozzle exit) are eventually driven to below 2 percent.

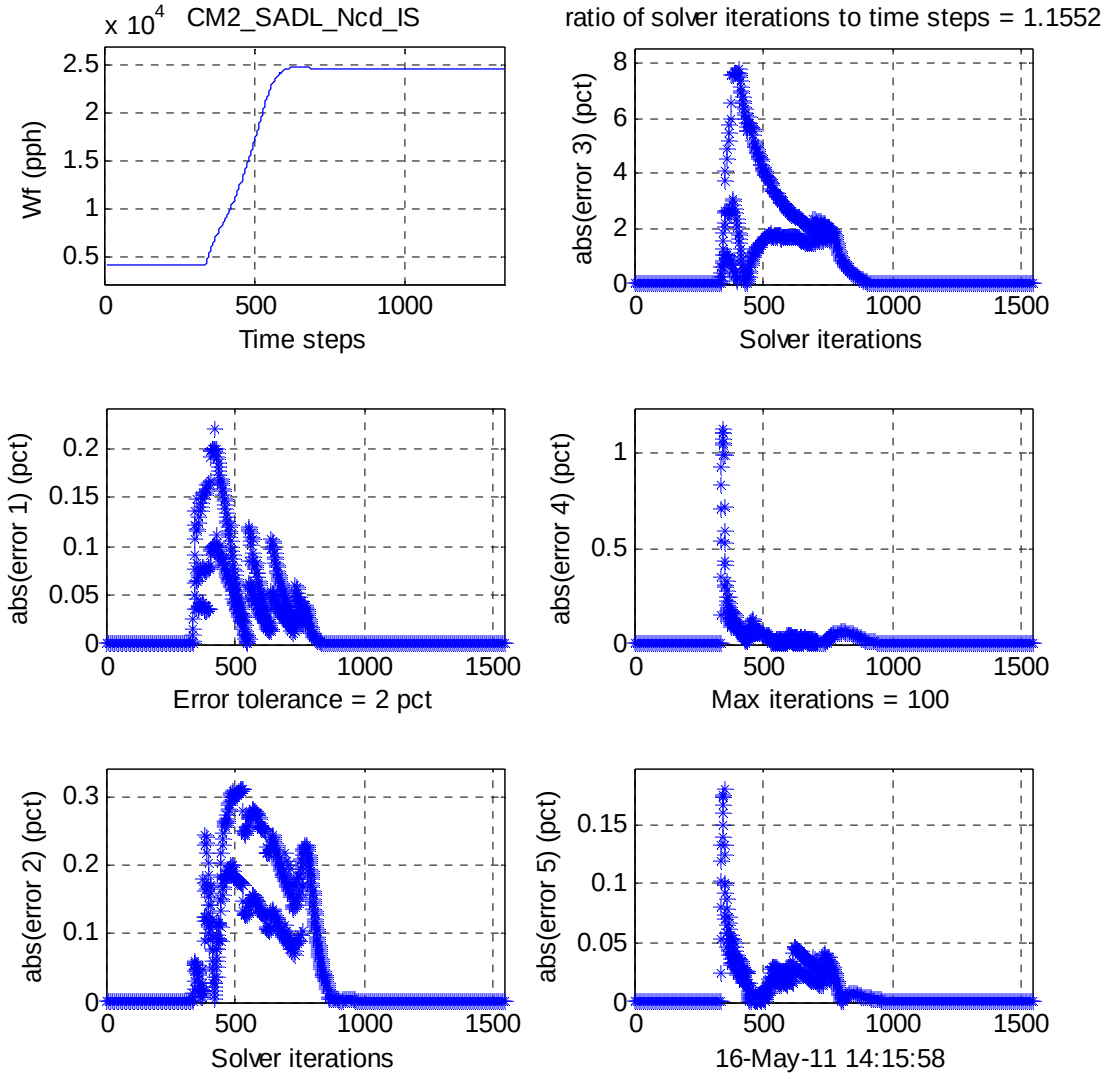

Figure 25.—Flow rate errors for 2 percent allowable error.

However, suppose we wish to have all flow errors to be within 0.5 percent. From Figure 25, we see that such a requirement would affect error 3 and error 4 (consistency among HPC, burner, and HPT flow rates), which remain above 0.5 percent at certain time steps with the default settings. To change the allowable error, modify the variable `error_tol` from the MATLAB command line. The unit of `error_tol` is percent. Thus, change allowable error to 0.5 percent by entering `error_tol = 0.5` and run the simulation again. Figure 26 shows the results of this change. All flow errors are now iterated until they are less than 0.5 percent. To illustrate the error convergence in more detail, the magnitude of error 3 is plotted for 20 solver iterations (Figure 27). Lines connecting consecutive data points are shown to elucidate the progression of the error. Each "peak" represents the error at the start of a new time step (hence, these 20 iterations represent 6 time steps). The solver iterates the engine model to achieve lower error values until the magnitude is less than 0.5 percent. The time step is then incremented, causing another peak in error magnitude, and the process is repeated.

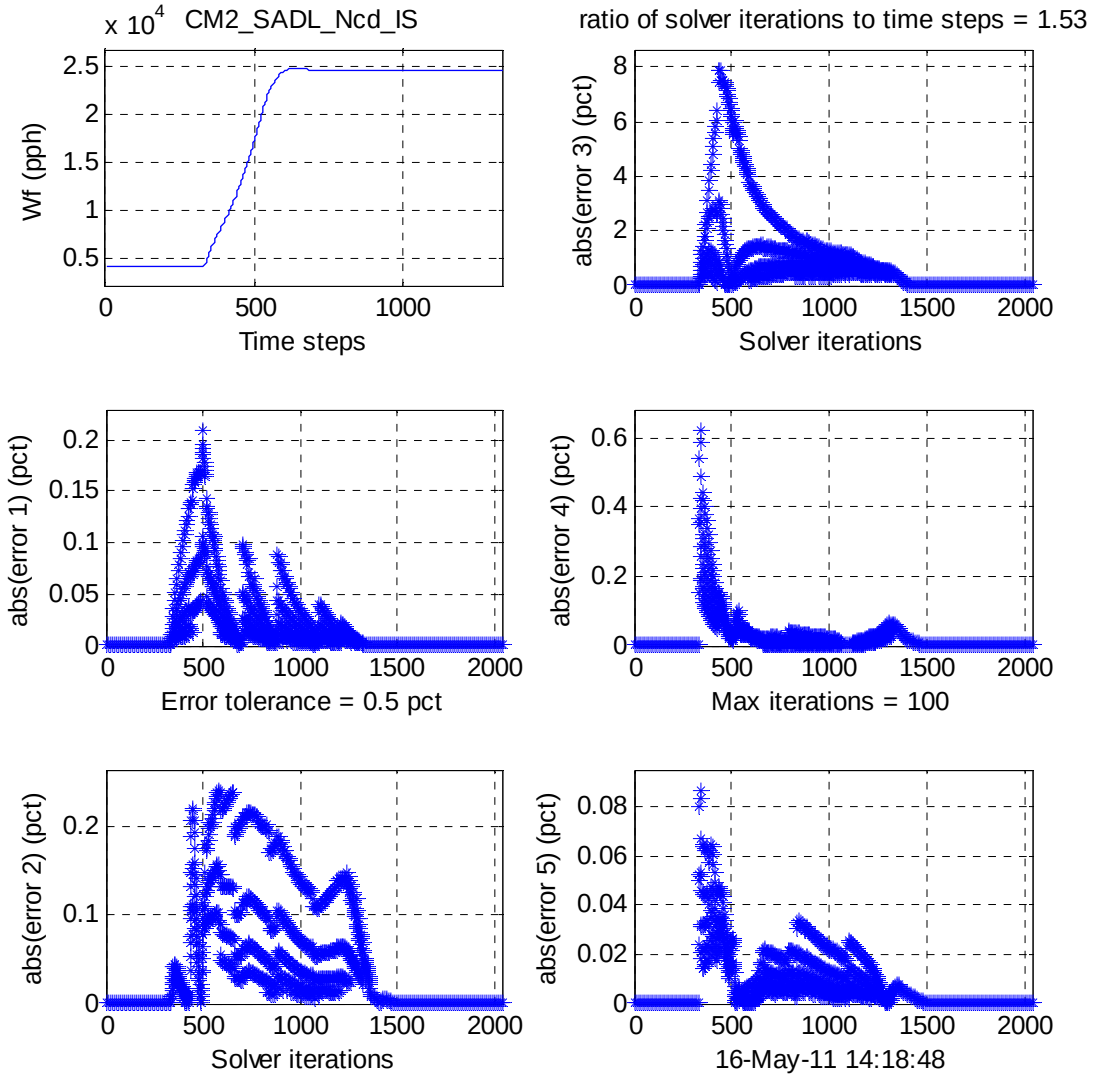

Figure 26.—Flow rate errors for 0.5 percent allowable error.

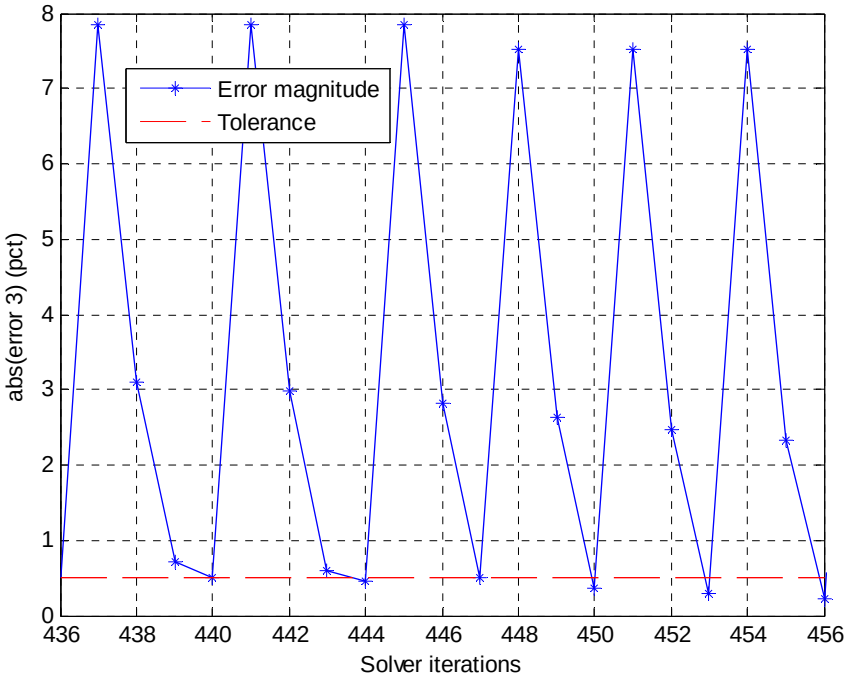

Figure 27.—Error 3 magnitude for 20 solver iterations.

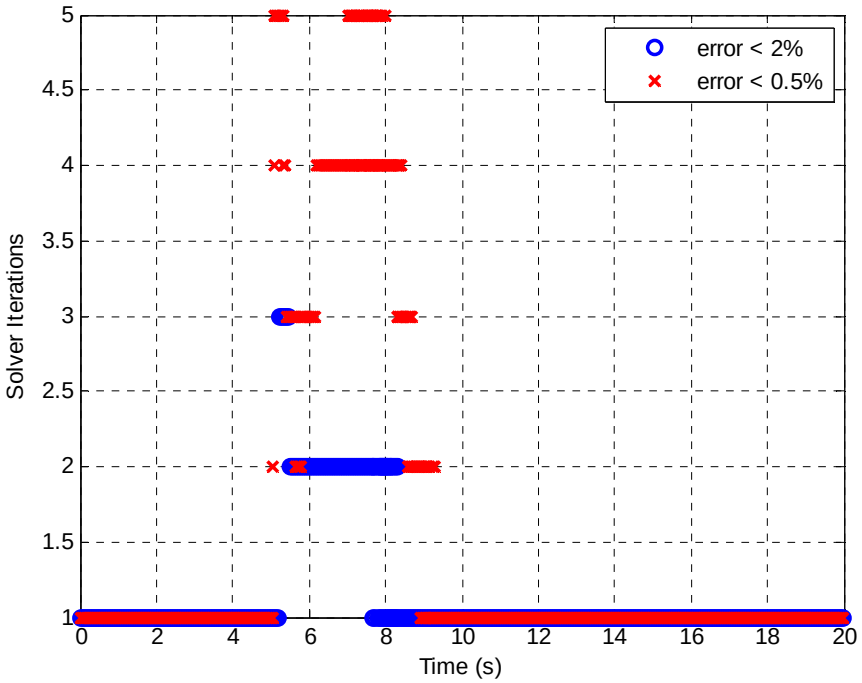

Figure 28.—Comparison of solver iterations per time step for 2 and 0.5 percent allowable errors.

The output variable `Iter_count` records the number of iterations at each simulation time step. Figure 28 shows the time evolution of iterations per time step for the cases of 2 and 0.5 percent allowable error. Expectedly, the case of the smaller allowable error requires more solver iterations per time step. Nonetheless, for both cases, the number of iterations per time step remains well below the default maximum of 100. If this ceiling is reached, it may be changed by editing the variable `max_iter` from the MATLAB command line.

6.5 Create and Analyze Linear Models

This section describes the steps for creating and analyzing a linear model. We will use the flight condition created in Section 6.2: 85° TRA, 24000 ft altitude, Mach 0.6.

1) Once this flight condition is loaded or created, select `Create` from the `Linear Model` menu option.
2) The default values in the linearization GUI should be sufficient. Therefore, start the process by clicking `Linearize`. The MATLAB command prompt will display the linearization progress and the final state-space matrices once completed.
3) Save, if necessary, and close the linearization GUI. The newly created linear model is now loaded into the main GUI (Figure 29).
4) To analyze the linear model, select `Analyze LEM` from the `Linear Model` menu option.
5) Utilize the built-in analysis functionalities in the GUI. For instance, Figure 30 shows the results of clicking the `transfer function` button. The display window shows the transfer function of the linear model from fuel flow to fan speed.
6) Utilize the drop-down menu in the `simulate response to doublet` panel to obtain the linear model's response to a doublet command in one of the inputs. Use the adjacent `plots` panel to view results. Figure 31, generated using the third plot button, shows the responses of several parameters of the linear model to a doublet fuel flow input.
7) Finally, use the `Compare with CLM` option from the `Linear Model` menu to compare responses of the linear model with those of the nonlinear model. Figure 32 shows the results of comparing the two models with a doublet fuel flow input. As the plots show, the linear model is a good localized representation of the nonlinear model.

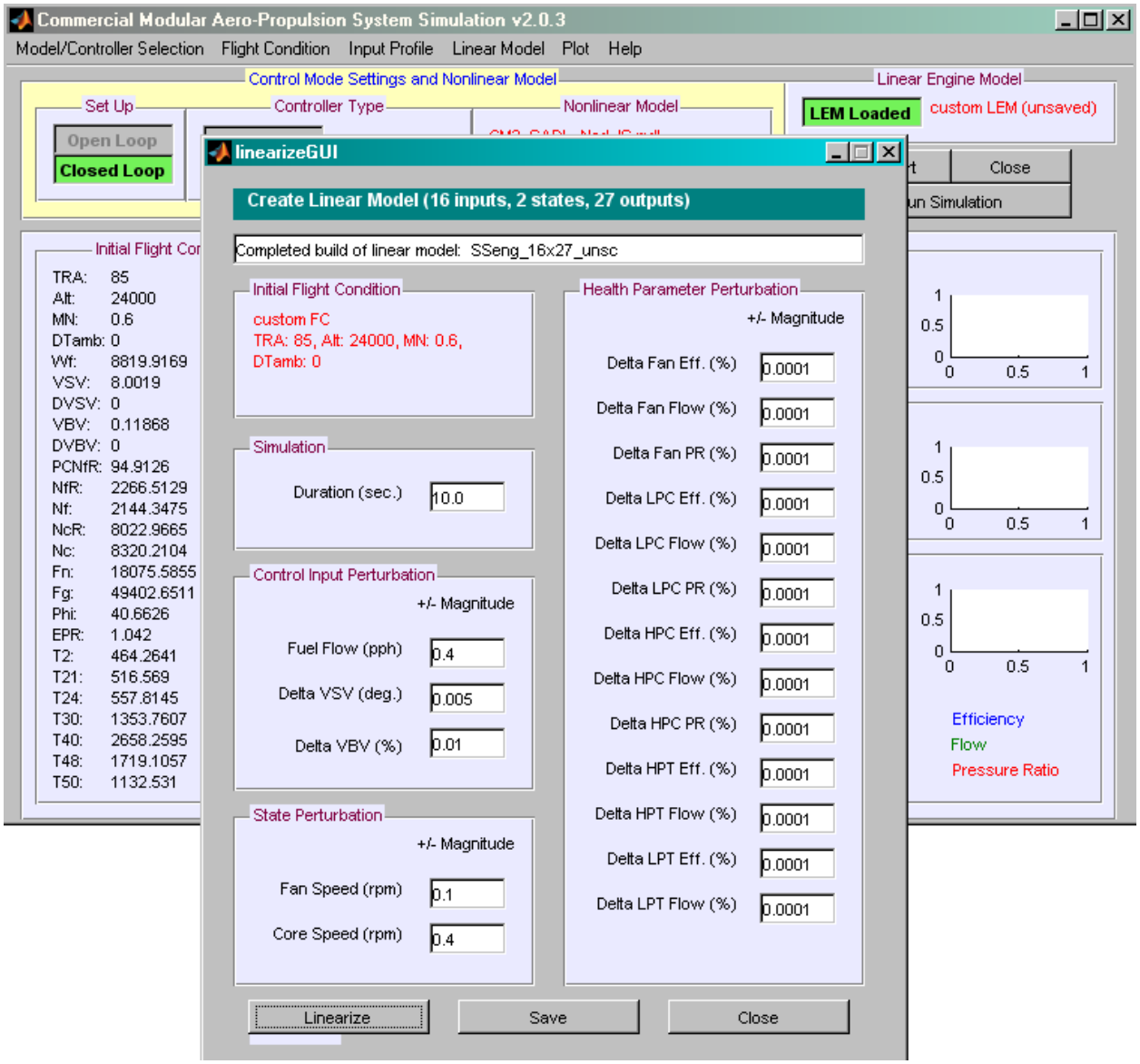

Figure 29.—Newly created linear model.

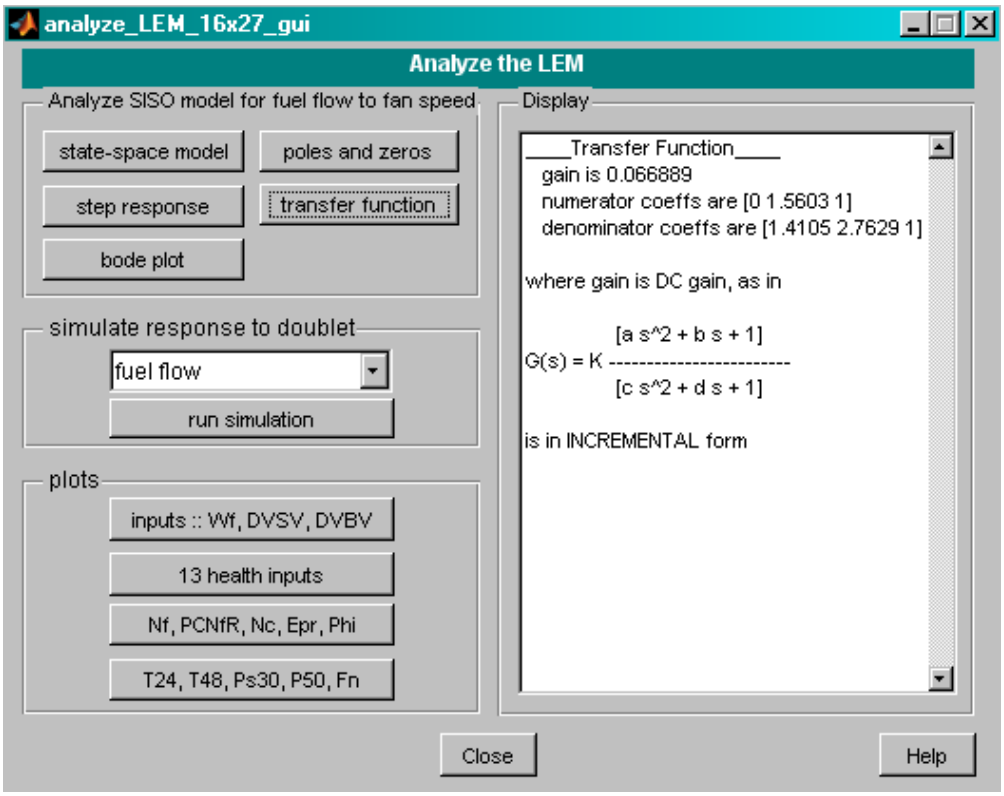

Figure 30.—Analysis of linear model.

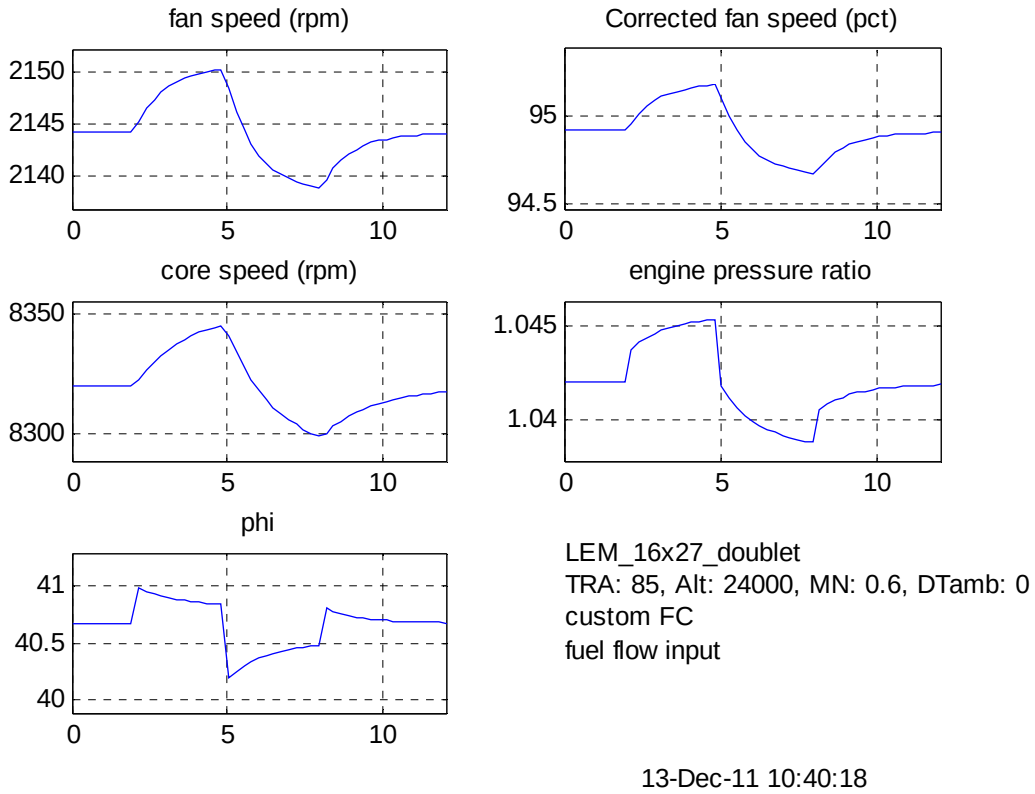

Figure 31.—Response of linear model to doublet command in fuel flow.

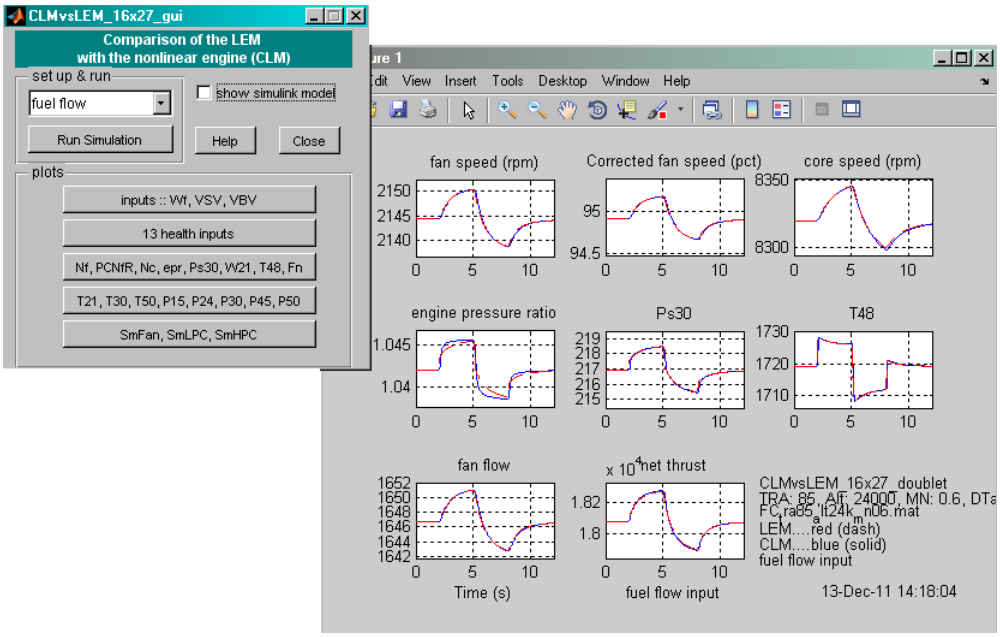

Figure 32.—Compare linear model with nonlinear model.

6.6 Design, Analyze, and Use Custom Controllers

This example shows the steps to design, analyze, and utilize custom controllers. In this case, we will design a fan speed controller for a certain flight condition, analyze it, and use it with the nonlinear engine model.

1) Load a flight condition into C-MAPSS. In this case, we will use 60° TRA, sea-level static, standard day (`FC11`).
2) Load the corresponding linear model (`LEM_FC11`).
3) Select `Design controller` from the `Linear Model` menu option.
4) Select `Fan speed` from the drop-down menu. For this design, we will change the parameters from the default values to: 5 rad/s natural frequency, 0.8 damping ratio, and two poles at –25. Note that the poles must be entered into the GUI as the negative of the actual desired value. Hence, for –25, we enter 25 into the appropriate fields in the GUI.
5) Click `Design`. Note the parameters are summarized in the display window and MATLAB command prompt window. Figure 33 shows the results of the design.
6) Click `Begin Analysis`. The five plot buttons should now be available for the user to analyze the controller. For instance, Figure 34 shows the plot created by clicking `plot closed-loop step response`. The plot shows the step response of the fan speed when using the newly created controller with the linear model.
7) To utilize the controller with the nonlinear model, first save the controller. In this example, we will save the controller as `reg_Nf_FC11_new`. Close the controller design GUI.
8) From the main GUI, select the “switched” model option, and choose `Point fan speed controller (no limit regulators)`
9) Load the newly created controller, `reg_Nf_FC11_new`. This file should be shown in the list of available controllers if it was saved to the default directory (see Figure 35). If not, select `Change Directory` and direct C-MAPSS to load the appropriate file.
10) Load a closed-loop input profile. Click `Run Simulation` to run the nonlinear model with the fan speed controller.

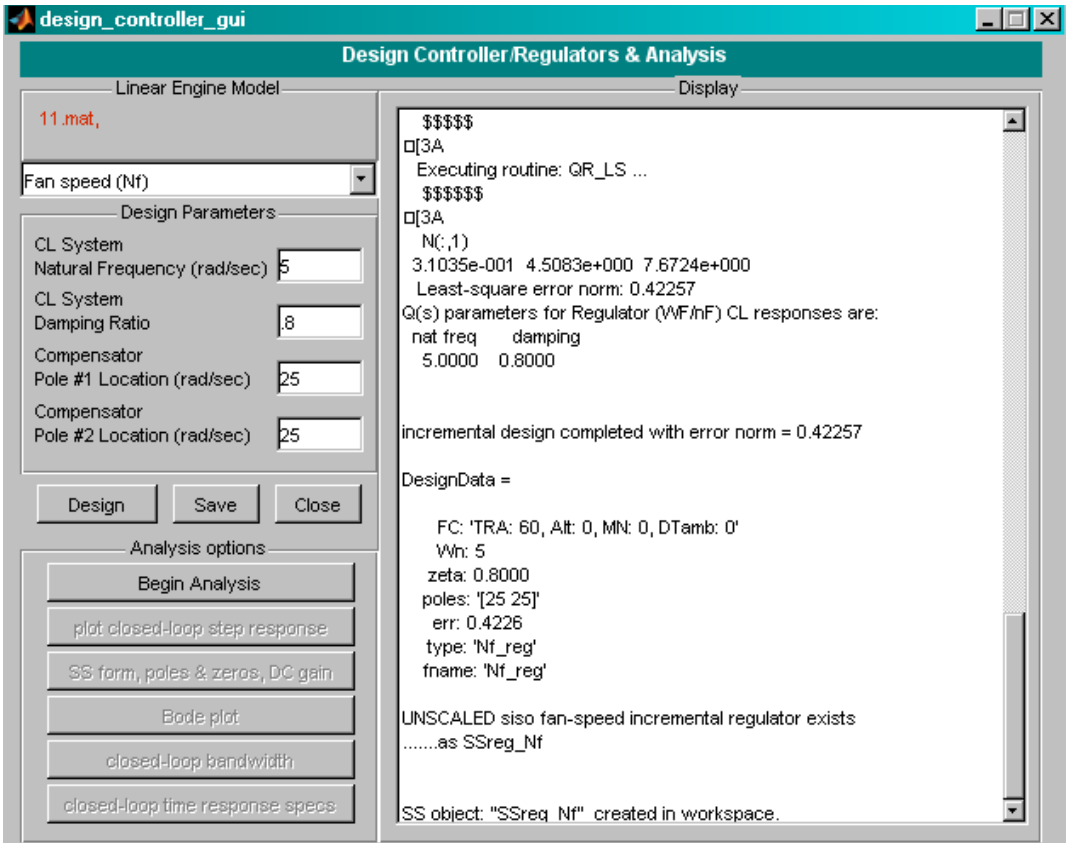

Figure 33.—Designing a custom fan speed controller.

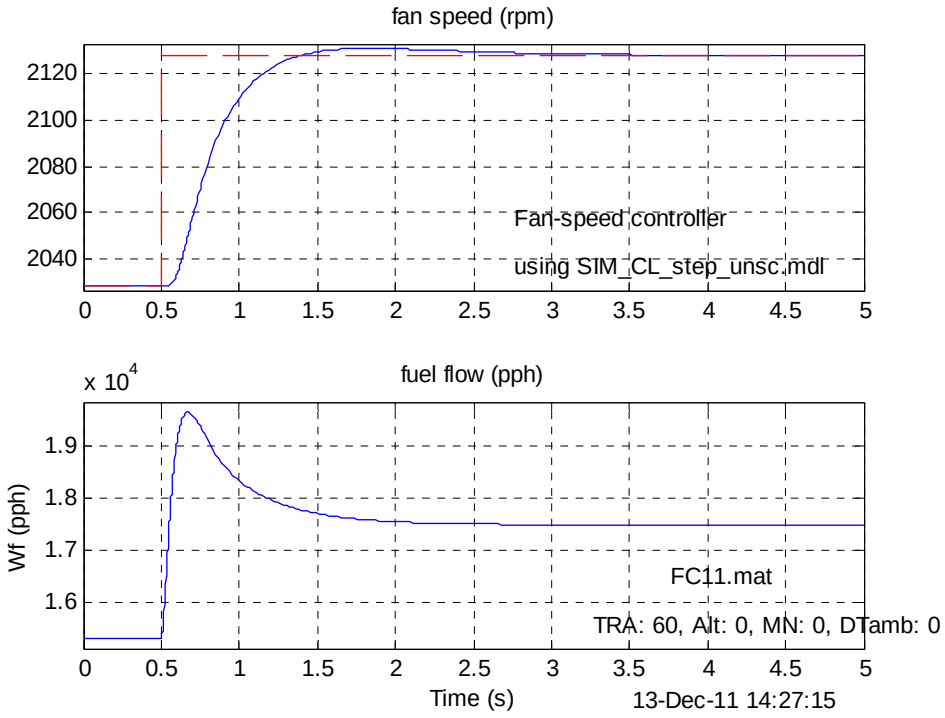

Figure 34.—Step response of linear model with newly designed fan speed controller.

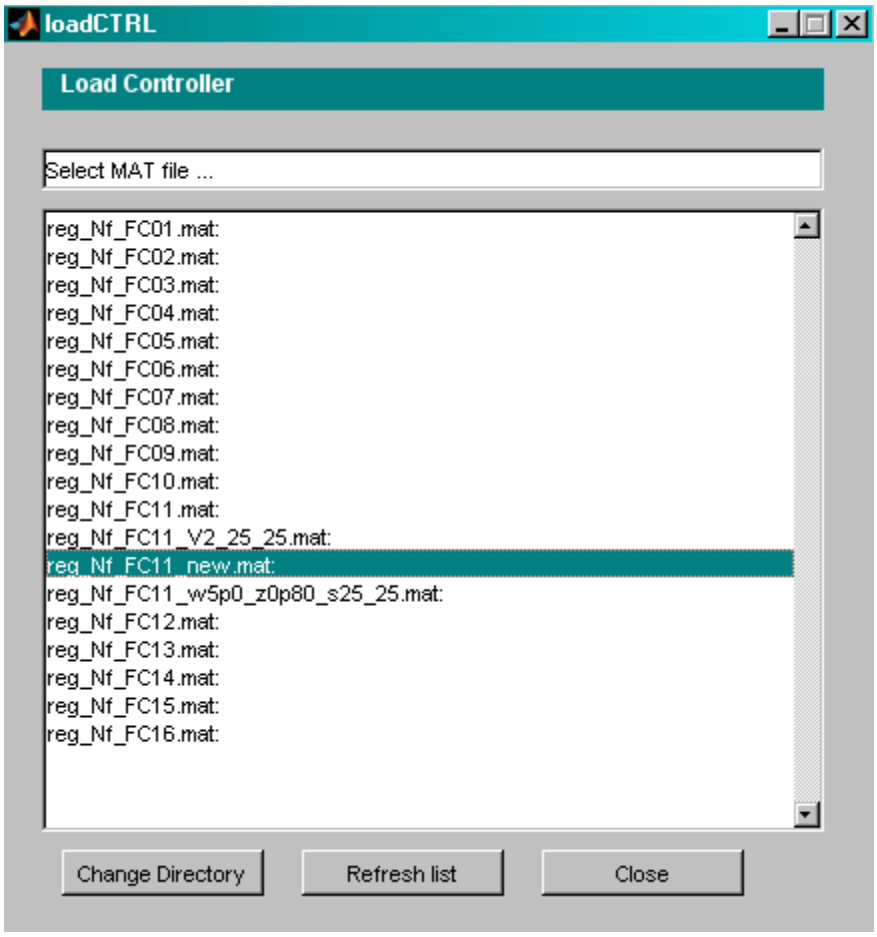

Figure 35.—Utilize newly created controller with nonlinear engine model.

To show the difference between the full scheduled controller and the newly designed controller, an existing input profile is loaded: `FC06_ramp_TRA_60`, which ramps TRA from 100° to 60° at 20000 ft and Mach 0.7. It is important to note that this flight condition is not a good match for the flight condition at which the fan speed controller was created. Figure 36 and Figure 37 (generated using the `Fan speed` option under the `Plot` menu) show the responses of the nonlinear engine model with the point fan speed controller and the full engine control system, respectively. As expected, the latter system, which utilizes a scheduled fan speed controller, produces a better fan speed response (with less “under-shoot” across the desired fan speed) than the point fan speed controller.

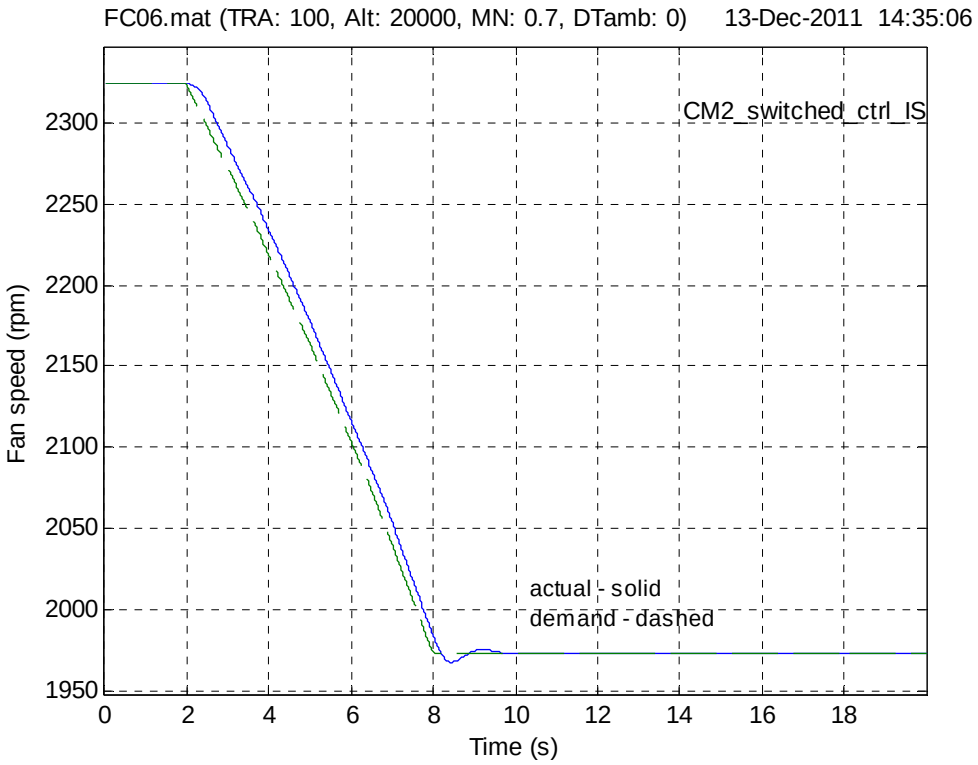

Figure 36.—Nonlinear model response with point fan speed controller.

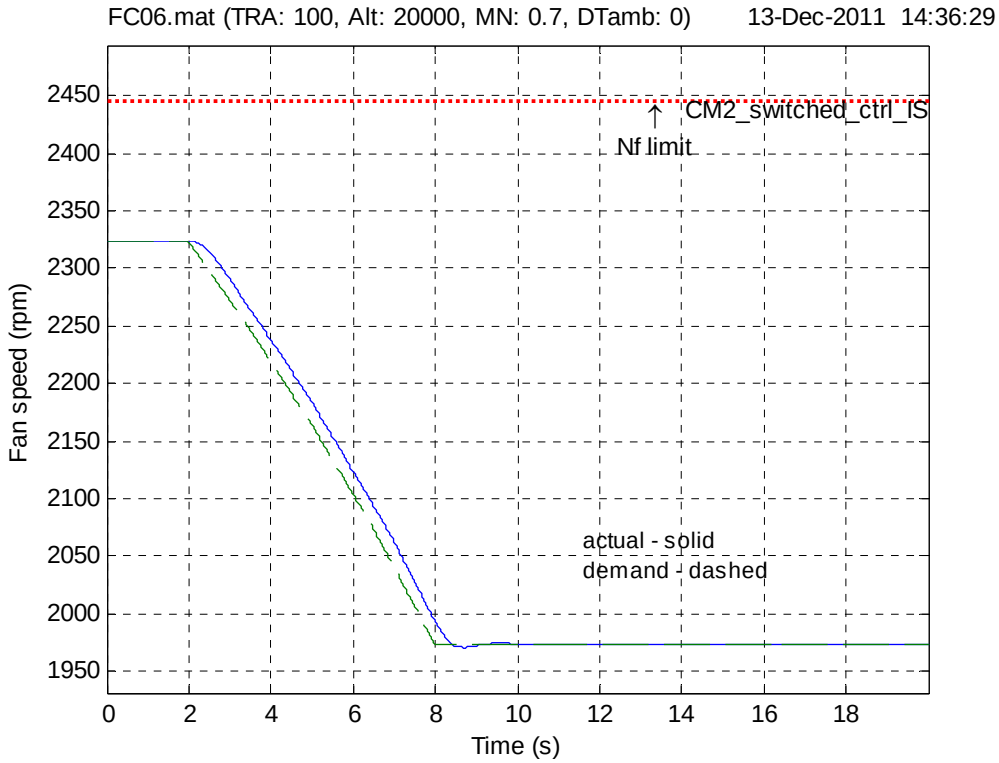

Figure 37.—Nonlinear model response with full engine control system.

Appendix—C-MAPSS Input/Output Variables

TABLE 2.—C-MAPSS INPUT VARIABLES
[Index refers to position in linear model input vector.]

Index	Description	Variable name	Units
1	Fuel flow	Wf	pph
2	Delta VSV	DVSV	deg
3	Delta VBV	DVBV	--
4	Fan efficiency modifier	fan_eff_mod	%
5	Fan flow modifier	fan_flow_mod	%
6	Fan pressure ratio modifier	fan_PR_mod	%
7	LPC efficiency modifier	LPC_eff_mod	%
8	LPC flow modifier	LPC_flow_mod	%
9	LPC pressure ratio modifier	LPC_PR_mod	%
10	HPC efficiency modifier	HPC_eff_mod	%
11	HPC flow modifier	HPC_flow_mod	%
12	HPC pressure ratio modifier	HPC_PR_mod	%
13	HPT efficiency modifier	HPT_eff_mod	%
14	HPT flow modifier	HPT_flow_mod	%
15	LPT efficiency modifier	LPT_eff_mod	%
16	LPT flow modifier	LPT_flow_mod	%

TABLE 3.—C-MAPSS OUTPUT VARIABLES
[Index refers to position in linear model output vector.]

Index	Description	Variable name	Units
1	Physical fan speed	Nf	RPM
2	Physical core speed	Nc	RPM
3	Engine pressure ratio (P50/P2)	EPR	--
4	Total pressure at fan outlet	P21	psia
5	Total temperature at fan outlet	T21	R
6	Total pressure at LPC outlet	P24	psia
7	Total temperature at LPC outlet	T24	R
8	Total pressure at HPC outlet	P30	psia
9	Total temperature at HPC outlet	T30	R
10	Total pressure at burner outlet	P40	psia
11	Total temperature at burner outlet	T40	R
12	Total pressure at HPT outlet	P45	psia
13	Total temperature at HPT outlet	T48	R
14	Total pressure at LPT outlet	P50	psia
15	Total temperature at LPT outlet	T50	R
16	Fan flow	W21	pps
17	Net thrust	Fn	lbf
18	Gross thrust	Fg	lbf
19	Fan stall margin	SmFan	%
20	LPC stall margin	SmLPC	%
21	HPC stall margin	SmHPC	%
22	Corrected fan speed	NfR	RPM
23	Corrected core speed	NcR	RPM
24	Total pressure in bypass-duct	P15	Psia
25	Percent corrected fan speed	PCNfR	%
26	Static pressure at HPC outlet	Ps30	psia
27	Ratio of fuel flow to Ps30	Phi	pph/psia

TABLE 4.—ADDITIONAL C-MAPSS OUTPUT VARIABLES
[Not included in linear model.]

Description	Variable name	Units
Fan acceleration	Nf_dot	rpm/s
Core acceleration	Nc_dot	rpm/s
Demanded fan speed	Nf_dmd	rpm
Pressure at fan inlet	P2	psia
Corrected fan speed demanded	PCNfRdmd	%
Output of PCNfR filter for gain scheduling	PCNfR_filtered	%
Throttle resolver angle	TRA	deg
Fan inlet total temperature	T2	R
Derivative of fuel flow	Wf_dot	lbm/s^2

References

1. Frederick, D.K., DeCastro, J.A., and Litt, J.S., "User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)," NASA/TM—2007-215026, October 2007.
2. DeCastro, J.A., Litt, J.S., and Frederick, D.K., "A Modular Aero-Propulsion System Simulation of a Large Commercial Aircraft Engine," 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 21-23, 2008, Hartford, CT.
3. 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," AIAA–2010–6630, 46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 25-28, 2008, Nashville, TN.
4. http://www.mathworks.com/
5. Maciejowski, J.M., Multivariable Feedback Design, Addison Wesley, 1989.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-03-2012	Technical Memorandum	

4. TITLE AND SUBTITLE
User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)
Version 2

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Liu, Yuan; Frederick, Dean, K.; DeCastro, Jonathan, A.; Litt, Jonathan, S.; Chan, William, W.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
WBS 457280.02.07.03.03.01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-18125

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITOR'S ACRONYM(S)
NASA

11. SPONSORING/MONITORING REPORT NUMBER
NASA/TM-2012-217432

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category: 07
Available electronically at http://www.sti.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 443-757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report is a Users' Guide for version 2 of the NASA-developed Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) software, which is a transient simulation of a large commercial turbofan engine (up to 90,000-lb thrust) with a realistic engine control system. The software supports easy access to health, control, and engine parameters through a graphical user interface (GUI). C-MAPSS v.2 has some enhancements over the original, including three actuators rather than one, the addition of actuator and sensor dynamics, and an improved controller, while retaining or improving on the convenience and user-friendliness of the original. C-MAPSS v.2 provides the user with a graphical turbofan engine simulation environment in which advanced algorithms can be implemented and tested. C-MAPSS can run user-specified transient simulations, and it can generate state-space linear models of the nonlinear engine model at an operating point. The code has a number of GUI screens that allow point-and-click operation, and have editable fields for user-specified input. The software includes an atmospheric model which allows simulation of engine operation at altitudes from sea level to 40,000 ft, Mach numbers from 0 to 0.90, and ambient temperatures from -60 to 103 °F. The package also includes a power-management system that allows the engine to be operated over a wide range of thrust levels throughout the full range of flight conditions.

15. SUBJECT TERMS
Turbofan engine simulation; Control; Actuation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email:help@sti.nasa.gov)
U	U	U	UU	48	19b. TELEPHONE NUMBER *(include area code)* 443-757-5802