NASA/TM—2010-215840

Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide

Donald L. Simon
Glenn Research Center, Cleveland, Ohio

January 2010

NASA/TM—2010-215840

Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide

Donald L. Simon
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

January 2010

Acknowledgments

The support of the following individuals in creating the ProDiMES software is graciously acknowledged: Jeff Bird, Craig Davison, Al Volponi, Gene Iverson, Link Jaw, Richard Eustace, Tak Kobayashi, Jonathan Litt, Jon DeCastro, Eric Gillman, and Christopher Heath.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

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
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

Contents

Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide

Donald L. Simon
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

1.0 Introduction to ProDiMES

The Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) provides a standard benchmarking problem and a set of evaluation metrics to enable the comparison of candidate aircraft engine gas path diagnostic methods. This User's Guide provides an overview description of the software tool along with installation and operating instructions.

1.1 Overview

Recent technology reviews have revealed that while Engine Health Management (EHM) related research and development has increased significantly in recent years, there exists a fundamental inconsistency in defining and representing EHM problems (Ref. 1). Currently many of the EHM solutions published in the open literature are applied to different platforms, with different levels of complexity, addressing different problems, and using different metrics for evaluating performance. As such, it is difficult to perform a one-to-one comparison of candidate approaches. Furthermore, these inconsistencies create barriers to effective development of new algorithms and the exchange of EHM-related ideas and results. To help address these issues, the ProDiMES software tool has been specifically designed with the intent to be made publicly available. In this form it can serve as a reference, or theme problem, to aid in propulsion gas path diagnostic technology development and evaluation. The overall goal is to provide a tool that will serve as an industry standard, and will truly facilitate the development and evaluation of significant EHM capabilities. Additional details regarding the motivation for creating the ProDiMES tool can be found in Reference 2.

ProDiMES has been constructed based on feedback provided by various individuals within the aircraft engine health management community. A key design decision in the development of ProDiMES was the decision to construct the tool around an aircraft engine simulation. Gas path diagnostic algorithm development and validation requires access to engine models and data. Ideally, this would include a rich database of information collected from engines over a broad range of operating conditions, deterioration levels, and known fault and no-fault conditions. However, to facilitate a public benchmarking approach, it was decided to generate simulated engine data utilizing a steady-state version of the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) high-bypass turbofan engine model (Ref. 3), referred to as C-MAPSS Steady-State. C-MAPSS Steady-State applies conventional aerothermal performance cycle modeling practices to construct a nonlinear component level model of a generic turbofan engine. Its use within ProDiMES avoids the use of engine data and analytical models that contain proprietary information. While an engine simulation will never fully capture all the nuances contained in actual engine data, it does provide some advantages. For example, it will allow the simulation of a broader range of fault types and magnitudes occurring over a broader range of engine operating conditions. It will also provide unambiguous knowledge of an engine's true fault/no-fault state, or "ground truth" condition. However, users are cautioned to realize that the simulated gas path fault test cases generated by ProDiMES have not been validated against actual engine fault data. They are simply representative of, not identical to, the types of faults an actual aircraft engine may experience. As such ProDiMES serves as a tool to enable the initial development and evaluation of candidate gas path diagnostic methodologies. It is readily acknowledged that additional maturation and development would be required to further mature gas path methods to the level of practical implementation.

The ProDiMES architecture for benchmarking aircraft engine gas path diagnostic methodologies is presented in Figure 1. It is coded in the Matlab (The MathWorks, Inc.) environment and specifically focuses on diagnostic methods applied to "snap shot," or discrete, engine measurements collected each flight at takeoff and cruise operating points. The intent is to provide users a publicly available toolset to enable the development, evaluation, and side-by-side comparison of candidate diagnostic solutions. This includes providing the functionality to evaluate analytical and/or empirical diagnostic approaches. The top half of Figure 1 shows the capabilities provided by ProDiMES that enable end users to independently develop and evaluate diagnostic solutions. This includes an Engine Fleet Simulator, user provided diagnostic solution algorithms, and a metrics evaluation routine. Each is briefly summarized below:

- ***Engine Fleet Simulator (EFS)***: The EFS generates and archives sensed parameter histories as if collected from a fleet of engines over multiple flights. Through an associated Matlab Graphical User Interface (GUI) users are able to specify the type and quantity of the simulated engine data produced by the EFS. This includes selecting the number of engines in the fleet, the number of flights that data will be collected over, the number of occurrences of 18 different gas path fault types, the flight of fault initiation, and the fault evolution (propagation) rate. The EFS incorporates stochastic elements that will result in the random generation of representative engine operating conditions, deterioration profiles, fault magnitudes, and sensor noise. Once generated and archived, the EFS sensed parameter histories serve as diagnostic test cases for the development and evaluation of user provided diagnostic solutions.
- ***User Provided Diagnostic Solutions***: Diagnostic solution(s) to be developed by the user. They are designed to interpret EFS generated parameter histories and diagnose the occurrence of any faults.
- ***Metrics***: A provided routine to automatically evaluate the diagnostic performance of candidate diagnostic solutions against a predefined set of metrics. This includes detection metrics (i.e., true positive rate, true negative rate, false positive rate, and false negative rate), classification metrics (i.e., correct classification rate, and misclassification rate), and diagnostic latency metrics. All metrics are generated by comparing the diagnostic assessments against the true fault / no-fault condition, (also known as the "ground truth" condition,) of the engines as generated by the EFS. Metric results are automatically archived to a Microsoft Excel (Microsoft Corporation) spreadsheet.

After diagnostic methods have been independently developed and evaluated, users will be invited to apply their diagnostic methods to a provided set of blind test case data that is also distributed as part of ProDiMES. This will enable the side-by-side comparison of diagnostic solutions developed by multiple users as shown in the bottom half of Figure 1. The blind test case data is also generated via the EFS, and all users will receive the same identical set of blind test case data. However, users will not be provided the associated ground truth fault information associated with the blind test case data. Diagnostic assessments produced based on the blind test case data are to be submitted to NASA, which will evaluate the results against the ground-truth information, and in return provide the metric results back to the end user along with the anonymous results of other participants.

This User's Guide will describe the elements of ProDiMES in more detail, along with installation and operating instructions. It is organized as follows. First, a version note and software installation instructions are provided. Next, elements of the EFS are more fully described and EFS operating instructions are provided. This is followed by a description of the output format that user provided diagnostic solutions must adhere to. Next, a description of the provided diagnostic metrics are given, along with operating instructions for the metrics evaluation routine. This is followed by instructions for evaluating and submitting diagnostic assessment results based on the blind test case data. 0 provides the ProDiMES software parameter list, and Appendix B presents an example diagnostic solution, that is distributed with the ProDiMES software.

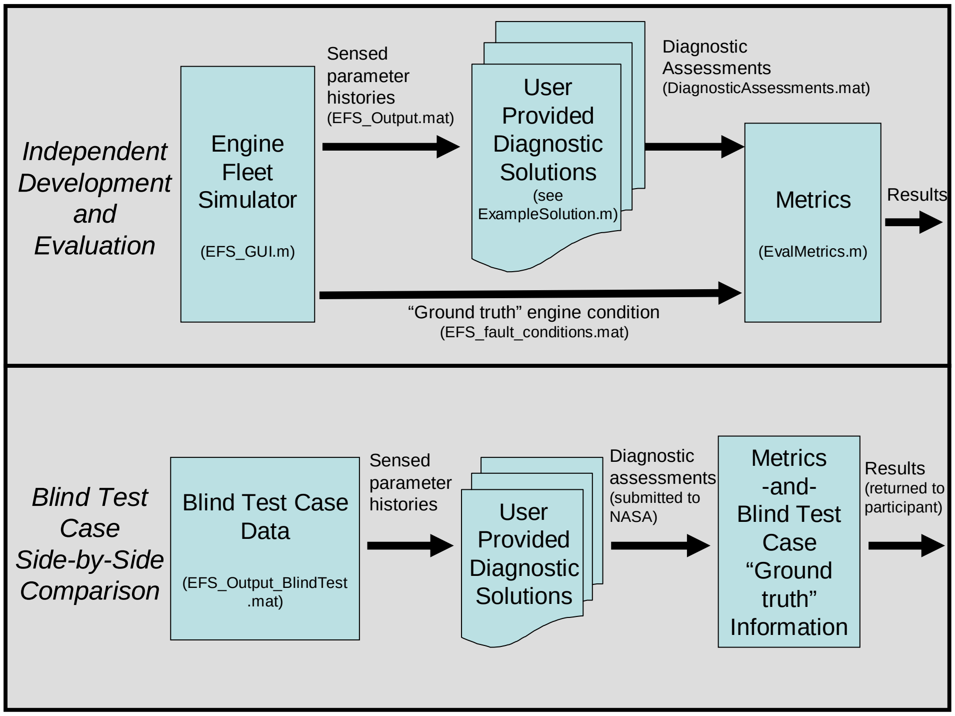

Independent Development and Evaluation
Engine Fleet Simulator
(EFS_GUI.m)
Sensed parameter histories
(EFS_Output.mat)
User Provided Diagnostic Solutions
(see ExampleSolution.m)
Diagnostic Assessments
(DiagnosticAssessments.mat)
Metrics
(EvalMetrics.m)
Results
"Ground truth" engine condition
(EFS_fault_conditions.mat)
Blind Test Case Side-by-Side Comparison
Blind Test Case Data
(EFS_Output_BlindTest.mat)
Sensed parameter histories
User Provided Diagnostic Solutions
Diagnostic assessments
(submitted to NASA)
Metrics -and- Blind Test Case "Ground truth" Information
Results
(returned to participant)

Figure 1.—ProDiMES Architecture.

1.2 Version Note and Software Requirements

This guide is intended to be used with the initial release of the ProDiMES software (version 1.0). This software has been developed using MATLAB Version 7.2.0.232 (R2006a) under the Microsoft Windows XP operating system. On other releases of Matlab the software will not run unless the associated executable code (Matlab MEX files) is recompiled from the provided C source code to generate platform specific executables. Procedures for doing this are discussed in more detail in Section 1.3. With recompilation it is expected that ProDiMES will be able to run on any current Matlab version, on any Matlab supported operating system, although this has not been completely verified. With recompilation of the provided C source code it is also expected that the software will run on future releases of Matlab. The ProDiMES metrics evaluation routine archives results to a Microsoft Office Excel 97-2003 worksheet. Users must have access to this version, or a newer version, of Microsoft Excel in order to run and interpret the metric results.

1.3 Installation Instructions

Unzip the contents of the provided ProDiMES.zip file to a local folder on your computer. This will create a directory structure as shown in Figure 2.

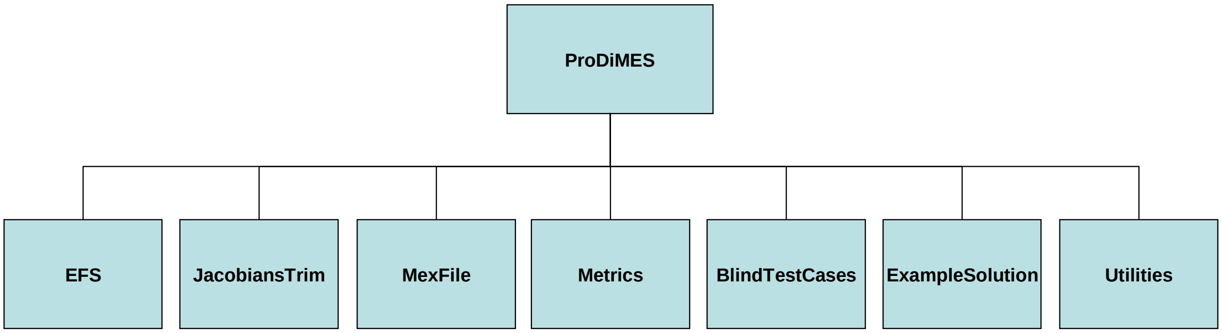


Figure 2.—ProDiMES Directory Structure.

A brief description of the contents of each subdirectory is given below:

ProDiMES: Top level directory. Contains User's Guide.

EFS: Engine Fleet Simulator Matlab software. The main program is contained in `EFS_GUI.m`. Upon execution it creates sensed parameter histories (`EFS_Output.mat`) and ground truth fault information (`EFS_fault_conditions.mat`).

JacobiansTrim: Contains Jacobian matrices and operating point trim files necessary to operate the engine model.

MexFile: C source code, a make file, and the Matlab *.mex file of the C-MAPSS Steady-State engine model.

Metrics: Evaluation Metrics Matlab routine (`EvalMetrics.m`) that compares diagnostic assessments against ground truth fault information to assess diagnostic performance metrics. This routine archives metric results to a Microsoft Excel spreadsheet (`EvalMetrics.xls`).

BlindTestCases: Contains a blind test case data set (`EFS_Output_BlindTest.mat`).

ExampleSolution: Contains an example user provided diagnostic solution (`ExampleSolution.m`). This routine illustrates how to acquire and parse EFS produced parameter histories (`EFS_Output.mat`), process the data, and archive diagnostic solutions in the required format (`DiagnosticAssessments.mat`).

Utilities: Contains routines to assist the user in correcting and plotting EFS generated sensed parameter histories (`plot_corrected_data.m`), and plotting true versus diagnosed engine fault condition information (`plot_true_vs_diagnosed_condition.m`).

1.3.1 Mex File Generation Steps

The engine model used in the ProDiMES software is coded in C and compiled into a Matlab *.mex function. This is done to improve the execution speed of the program. The *.mex function is located in the MexFile subdirectory and is named `cmapss_ss_function_C.mexw32`.

This file has been **compiled to run on a Windows PC platform using Matlab Version 7.2.0.232 (R2006a)**. In order to run the EFS on a different computer platform the *.mex file will need to be regenerated following these steps:

1. In Matlab navigate to the ProDiMES subdirectory titled “`\MexFile`”
2. At the Matlab command prompt, type “`make_file`”. This will produce a new mex file of the appropriate extension named “`cmapss_ss_function_C.*`”

2.0 EFS Description and Operating Instructions

The Engine Fleet Simulator (EFS) enables users to generate simulated engine output data for the development and evaluation of propulsion gas path diagnostic methodologies. The following subsections will provide a functional description of the EFS software, followed by a description of EFS operating instructions and output files.

2.1 Engine Fleet Simulator Functional Description

The EFS produces simulated “snap-shot” engine measurements, with relevant noise levels, as if collected from a fleet of engines at takeoff and cruise operating points each flight. In order to emulate realistic behavior, each engine within the fleet will experience unique operating and deterioration profiles. Users can choose for engines to encounter relevant gas path faults including sensor, actuator and component faults. The EFS architecture, shown in Figure 3, is implemented in the Matlab environment. This architecture consists of: a graphical user interface (GUI) that accepts user specified inputs regarding the number of engines and the number of occurrences of each fault type; a case generator designed to produce unique faults, deterioration profiles, and operating profiles for each engine in the fleet; and the nonlinear C-MAPSS Steady-State turbofan engine simulation that produces the “snap-shot” measurement parameter histories for each engine, at takeoff and cruise of each flight. Each component of the EFS is further described below.

2.1.1 EFS Graphical User Interface

The EFS GUI is designed to provide flexibility in generating data sets for diagnostic development and validation purposes. Through this interface, the type and number of faults that occur within the fleet of

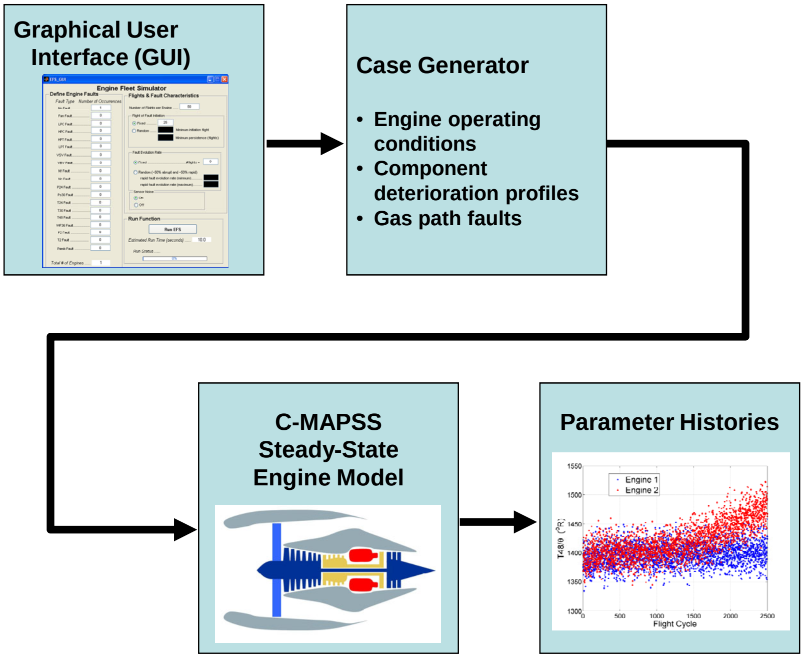


Figure 3.—EFS architecture.

engines are defined. It should be emphasized that the EFS has been designed assuming that an individual engine can only experience a single fault—it will not simulate multiple faults occurring within the same engine. There are 18 possible fault scenarios plus the no-fault scenario. The sum of the number of occurrences of each scenario determines the total number of engines in the fleet, and thus the number of test cases presented to the end user. The interface also allows the user to specify the following: the number of flights over which output data will be collected for each engine; the flight of fault initiation (either at a fixed flight number or randomly within a specified window of flights); the rate at which faults evolve, either abruptly (instantaneously) or rapidly (over a number of flight cycles); and sensor noise turned on or off.

2.1.2 EFS Case Generator

After the EFS inputs have been specified through the GUI, the user selects the "Run EFS" button. This initiates the process of generating engine parameter histories according to the number and type of scenarios specified by the user. The Case Generator randomly assigns a unique operating history and deterioration profile to each engine in the fleet. This includes assigning the city pairs that an engine will alternate takeoffs between, and the calendar date that engine data collection will commence. Pressure altitude, Mach number, ambient temperature, and power setting parameters at the takeoff and cruise operating points where data will be collected during each flight are all randomly generated from defined distributions representative of commercial aircraft operations. Histograms illustrating the distributions in operating parameters implemented within the EFS Case Generator are shown in Figure 4 (takeoff) and Figure 5 (cruise). At takeoff the power reference parameter is established by Power Lever Angle (PLA) that will either be 100 percent, or a fixed de-rated takeoff power setting of 90 or 80 percent as commonly applied in commercial aviation. At cruise the power reference parameter is specified by net thrust.

The EFS Case Generator also specifies a level and rate of gradual performance deterioration for each engine. These effects are included to emulate the gradual performance deterioration that an aircraft engine will naturally undergo over its lifetime of use due to fouling, erosion, and corrosion of turbomachinery blades and vanes. It should be emphasized that in ProDiMES gradual performance deterioration is not considered a fault, and it evolves on a much slower timescale than faults do. Diagnostic methods will not be required to diagnose gradual performance deterioration, but they should be designed to be robust to these effects. In ProDiMES, deterioration effects are simulated via adjustments to 10 health parameters within the engine simulation so that engines will continuously degrade over time. These health parameters include an efficiency and flow capacity modifier for each of the five major modules in the engine (Fan, Low Pressure Compressor (LPC), High Pressure Compressor (HPC), High Pressure Turbine (HPT), and Low Pressure Turbine (LPT)). The fleet average deterioration profile implemented within the EFS is representative of the information provided in (Ref. 4), although adjustments have been made to the turbine health parameters to cause them to deteriorate on a time scale consistent with the rest of the engine. An actual aircraft engine will deteriorate as a function of its usage, and the environment it operates in. Therefore, no two engines will deteriorate at the same rate or along the same profile. To emulate this the EFS Case Generator includes variations to the fleet average deterioration profile to produce a unique deterioration profile for each individual engine including: 1) more/less rapid overall engine deterioration; 2) more/less rapid individual module deterioration; 3) more/less module flow deterioration relative to efficiency deterioration; and 4) initial engine-to-engine manufacturing variation. Figure 6 shows the baseline, or average, deterioration profile for each health parameter (in red) and example variations (in cyan). The cyan points were generated by running the EFS to generate data from 100 engines over 5000 flights. 5000 flights is the maximum number of flight cycles that can be defined for any engine in the EFS. However, the user can choose fewer flights. If so, an engine's starting condition will be randomly placed somewhere along the deterioration profiles defined in Figure 6.

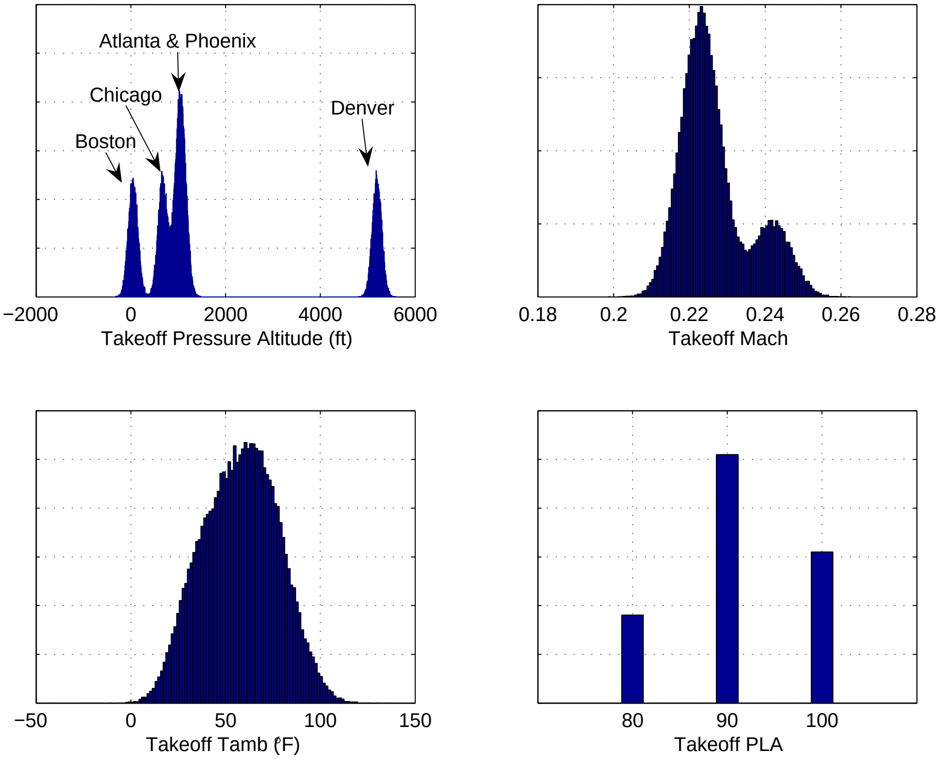


Figure 4.—Takeoff operating condition distributions.

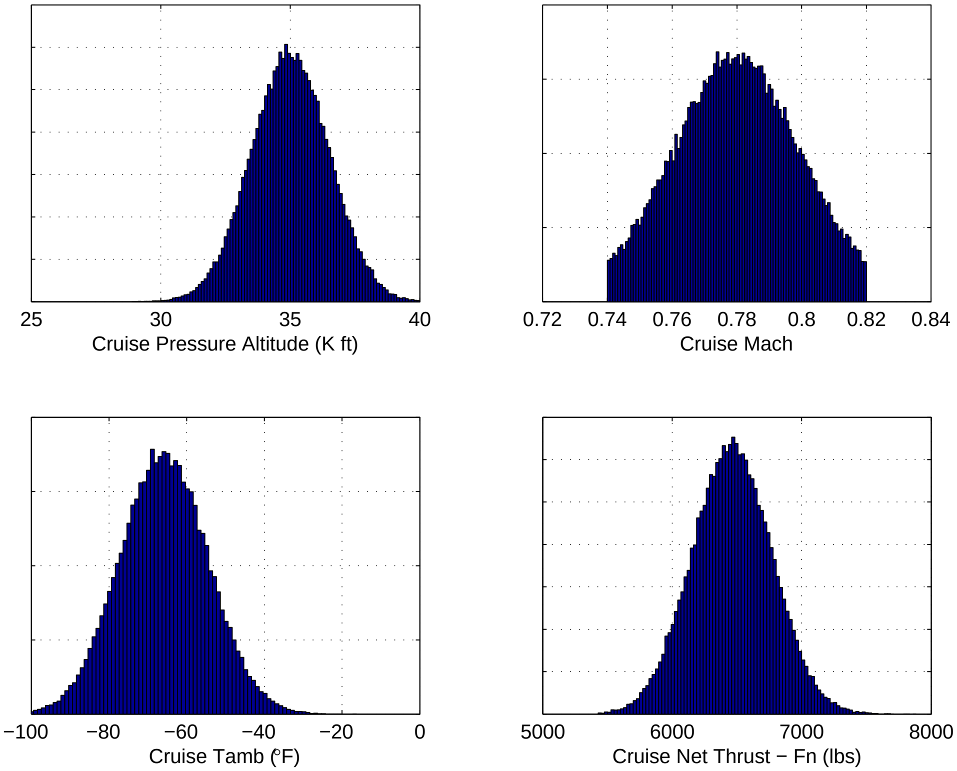


Figure 5.—Cruise operating condition distributions.

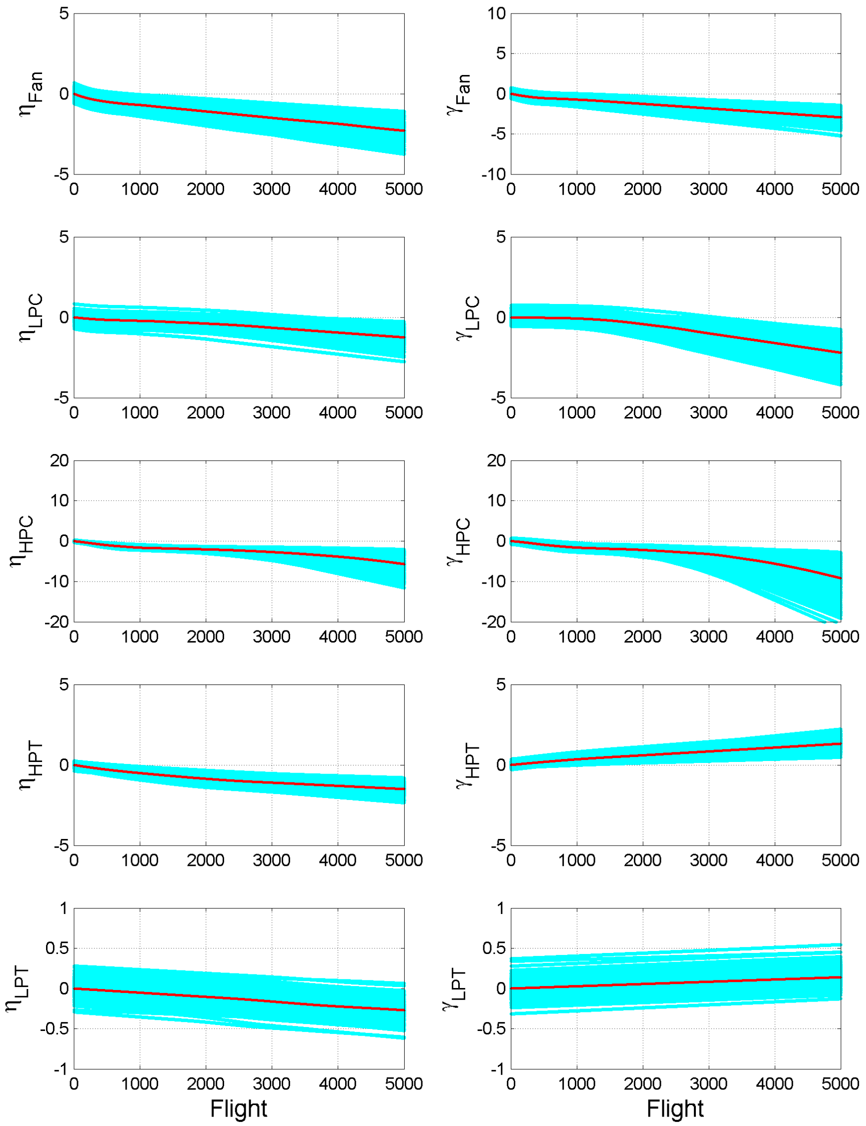


Figure 6.—Health parameter deterioration profiles, average (red) and distribution (cyan)

In addition to generating the operating history and deterioration profile for each engine, the Case Generator will also define the flight of fault initiation, the fault magnitude, and the fault evolution rate for those engines experiencing faults. A summary of the fault types and their uniformly distributed fault magnitudes is provided in Table 1. Module faults (i.e., fault ID's 1 through 5 corresponding to Fan, LPC, HPC, HPT and LPT faults) are simulated by simultaneously adjusting the faulty module health parameters: the efficiency, η, and flow capacity, γ. Module fault magnitude distributions shown in the table are in terms of the root-sum-square value of the combined η and γ deviations. The uniformly distributed ratios of flow capacity to efficiency health parameter adjustment, $\gamma:\eta_{ratio}$, are also shown. For Fan, LPC, and HPC faults, both γ and η are reduced. For HPT and LPT faults, η is reduced while γ is increased. Module faults are simulated by adjusting efficiency and flow capacity as follows:

$$\begin{aligned}\eta\ \text{adjustment} &= \frac{\text{fault magnitude}}{\sqrt{1+\left(\gamma:\eta_{\text{ratio}}\right)^{2}}}\\ \gamma\ \text{adjustment} &= \eta\ \text{adjustmen}t\cdot\left(\gamma:\eta_{\text{ratio}}\right)\end{aligned} \tag{1}$$

Actuator faults in the variable stator vanes, VSV, or the variable bleed valve, VBV, (fault ID 6 and 7) will result in a misscheduling between the commanded and actual actuator position. A fuel flow (Wf) actuator fault is not specifically included, but a fuel flow sensor fault is. The overall effect of a fuel flow actuator or sensor fault would be the same—a mismatch between the sensed and actual fuel flow.

There are 11 different sensor faults (fault ID’s 8 through 18) implemented within the EFS. Sensor fault magnitudes are implemented in units of average measurement noise standard deviation, σ, at takeoff and cruise. As a point of clarification it should be emphasized that if sensor noise is activated in the GUI, it will cause zero-mean normally distributed noise to be added to the sensed parameter histories generated by the EFS. This noise is applied as a percentage of the actual sensed parameter. Sensor fault magnitudes do not vary as a function of actual sensed parameter, they are simply defined as a multiple of the average noise within a sensed parameter at takeoff or cruise. Table 1 shows both percent (%) values of σ, as well as average σ values in engineering units at takeoff and cruise. Based on this table, it can be seen that a $+1\sigma$ fan speed (Nf) sensor fault will result in a +5.59 rpm bias existing between the sensed and actual Nf at takeoff, and a +4.67 rpm bias existing between the sensed and actual Nf at cruise. Users should be aware that faults in certain control feedback sensors will also result in actuator misscheduling. Specifically, VBV position is scheduled based on corrected fan speed, and VSV position is scheduled based on corrected core speed (corrected to station 24 temperature). Therefore a sensor fault in Nf, Nc, T2 or T24 will also result in actuator misscheduling.

TABLE 1.—EFS FAULT TYPES

Fault ID	Fault description	Fault magnitude	Fault* $\gamma : \eta_{ratio}$	σ (shown in %)	Takeoff average σ (actual units)	Cruise average σ (actual units)
0	No-fault	---	----------	-----	------------	------------
1	Fan fault	1 to 7%	2 to 1	-----	------------	------------
2	LPC fault	1 to 7%	2 to 1	-----	------------	------------
3	HPC fault	1 to 7%	2 to 1	-----	------------	------------
4	HPT fault	1 to 7%	–0.5 to –1	-----	------------	------------
5	LPT fault	1 to 7%	–0.5 to –1	-----	------------	------------
6	VSV fault	1 to 7%	----------	-----	------------	------------
7	VBV fault	1 to 19%	----------	-----	------------	------------
8	Nf sensor fault	± 1 to 10 σ	----------	0.25	5.59 rpm	4.67 rpm
9	Nc sensor fault	± 1 to 10 σ	----------	0.17	15.04 rpm	13.22 rpm
10	P24 sens. fault	± 1 to 10 σ	----------	0.50	0.134 psia	0.040 psia
11	Ps30 sens. fault	± 1 to 10 σ	----------	0.20	0.896 psia	0.224 psia
12	T24 sensor fault	± 1 to 10 σ	----------	0.16	1.01 °R	0.816 °R
13	T30 sensor fault	± 1 to 10 σ	----------	0.16	2.45 °R	1.92 °R
14	T48 sensor fault	± 1 to 10 σ	----------	0.50	10.07 °R	7.48 °R
15	Wf sensor fault	± 1 to 10 σ	----------	0.60	0.035 pps	0.0069 pps
16	P2 sensor fault	± 1 to 10 σ	----------	0.15	0.023 psia	0.0077 psia
17	T2 sensor fault	± 1 to 10 σ	----------	0.16	0.838 °R	0.707 °R
18	Pamb sens fault	± 1 to 19 σ	----------	0.15	0.022 psia	0.0052 psia

*The fault $\gamma : \eta_{ratio}$ column reflects the ratio of the change in flow capacity divided by the change in efficiency. For Fan, LPC, and HPC faults both efficiency and flow capacity are decreased (negative change), and thus their ratio is always positive. Conversely, for HPT and LPT faults flow capacity is increased (positive) while efficiency is decreased (negative), thus their ratio is always negative.

C-MAPSS Steady-State Engine Model

The outputs of the Case Generator are provided as inputs to a steady-state version of the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) high-bypass turbofan engine model. C-MAPSS is a transient nonlinear aerothermodynamic engine model developed for controls and diagnostics research and development purposes and implemented in the Matlab/Simulink environment (Ref. 3). It has two states, fan and core speed, and three actuators including fuel flow, variable stator vanes, and a variable bleed valve. A modified steady-state version of C-MAPSS, denoted as C-MAPSS Steady-State, is used by the EFS. The C-MAPSS Steady-State engine model has been validated against the original C-MAPSS model. The steady-state version of C-MAPSS has several

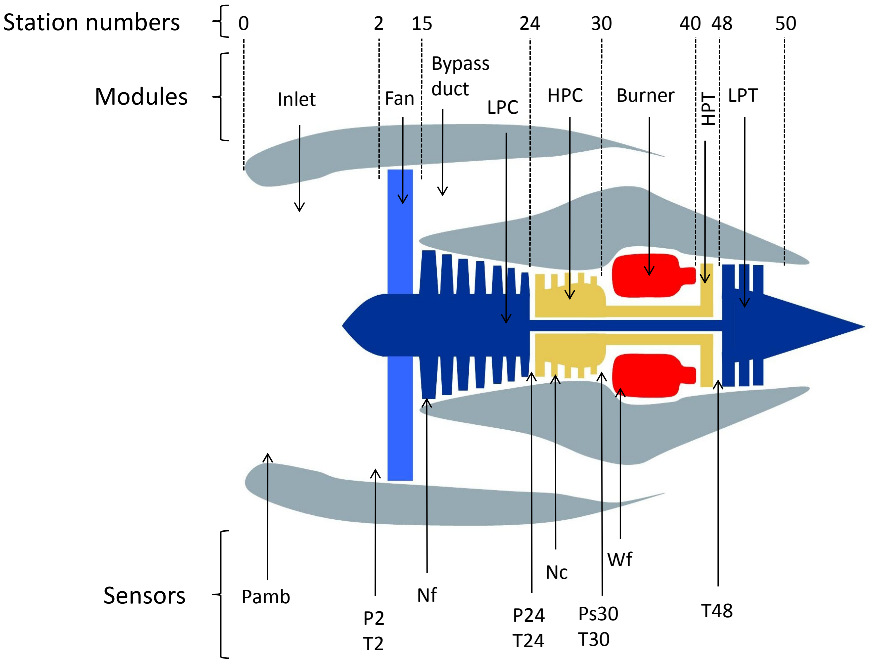


Figure 7.—C-MAPSS Steady-State station numbers, modules, and sensors.

notable differences from the original: the closed-loop control system has been removed, the code has been converted from Simulink to a Matlab Mex file, and the transient solver has been replaced by a steady-state solver to enable quick convergence to steady-state operating points. C-MAPSS Steady-State is executed within the EFS to generate simulated output data for each engine, at each flight, for each takeoff and cruise operating condition specified by the Case Generator. Although C-MAPSS Steady-State does not include closed-loop control logic, it does apply logic to capture actuator misscheduling due to control feedback sensor faults, and to ensure that operating limits are not violated in generating the steady-state solutions. End users are provided access to the C-MAPSS Steady-State model for diagnostic solution development purposes. For example, fault influence coefficient matrices can be extracted from C-MAPSS Steady-State for use in the design of model-based diagnostic methods as described in the example solution provided in Appendix B. Figure 7 shows C-MAPSS Steady-State station numbers, along with module and sensor locations.

2.2 EFS Operating Instructions and Output Files

The EFS operating instructions and EFS output files are described below.

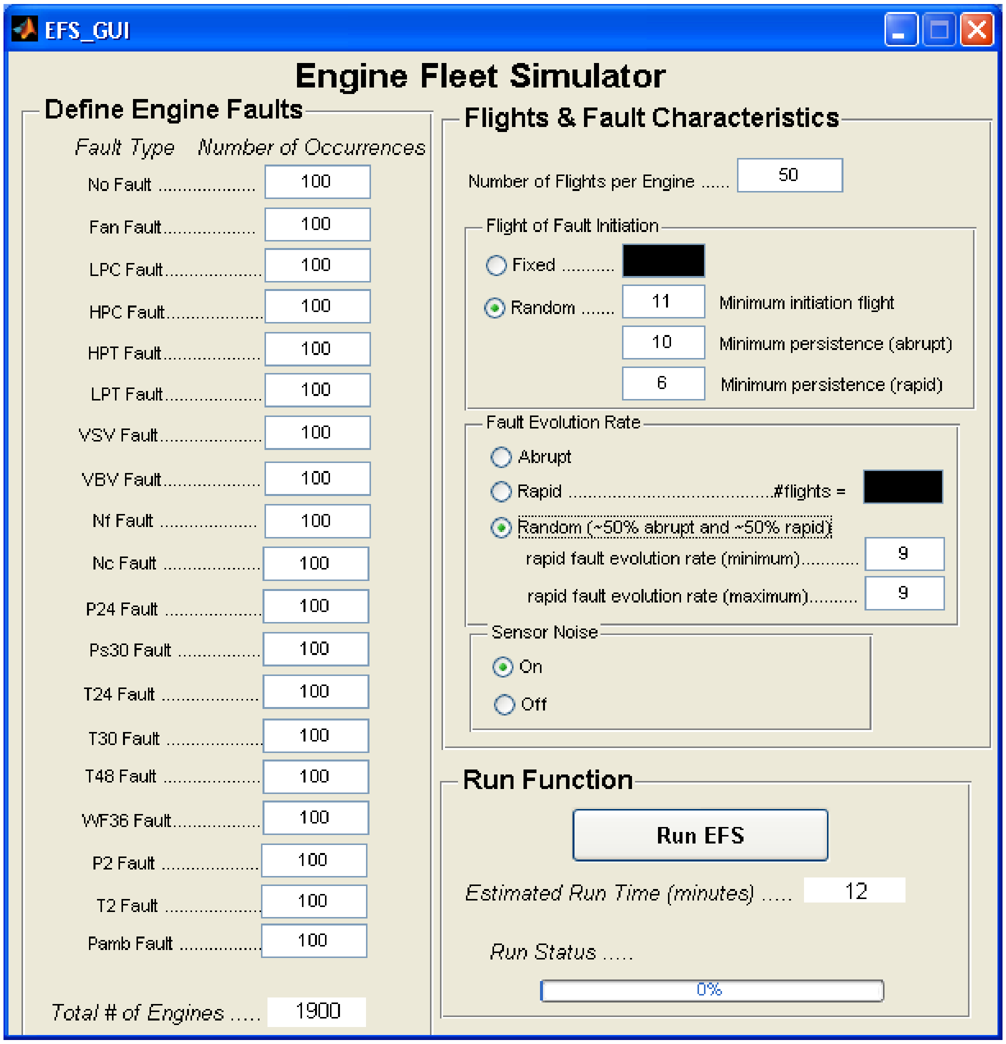
2.2.3 EFS Operating Instructions

The steps for running the EFS are as follows:

1. In the Matlab command window navigate to the `EFS` directory, and type the command `EFS_GUI`. This will open the EFS graphical user interface shown in Figure 8.
2. Using the boxes on the left side of the GUI, specify the number of occurrences of the No Fault case, plus the number of occurrences of each of the 18 possible fault types. The sum of these inputs defines the total number of engines in the fleet, which is displayed at the bottom left hand side of the GUI.

3. Using the box labeled “Number of Flights per Engine,” specify the number of flights of data that will be generated for each engine. Acceptable entries range from a minimum of one flight, to a maximum of 5000 flights.
4. Select the flight of fault initiation. Users can select “fixed” flight of fault initiation that will result in all engines in the fleet experiencing faults on the same flight, or “random” flight of fault initiation, that will result in the flight of fault initiation being randomly assigned for each engine.
   a. If fixed flight of fault initiation is chosen the user will be required to specify the flight number.
   b. If random flight of fault initiation is chosen, the user must specify the minimum possible flight of fault initiation, and the minimum possible number of flights the faults will persist. These inputs bound the window of flights over which faults can initiate. Note: minimum fault persistence information is specified for both “abrupt” and “rapid” evolution fault types, which are defined below.

   It should noted that ProDiMES does not currently provide the capability to introduce and then later remove a fault from an engine’s sensed parameter history. Once a fault initiates, it will persist over the remaining duration of that particular engine’s flight history.
5. The fault evolution rate refers to how rapidly faults will evolve, or grow in magnitude. The ProDiMES software provides three different fault evolution rate options: abrupt, rapid, or random



Engine Fleet Simulator

Define Engine Faults

Fault Type	Number of Occurrences
No Fault	100
Fan Fault..................	100
LPC Fault...................	100
HPC Fault...................	100
HPT Fault...................	100
LPT Fault...................	100
VSV Fault....................	100
VBV Fault...................	100
Nf Fault	100
Nc Fault	100
P24 Fault	100
Ps30 Fault	100
T24 Fault	100
T30 Fault	100
T48 Fault	100
WF36 Fault..................	100
P2 Fault	100
T2 Fault	100
Pamb Fault	100

Total # of Engines 1900

Flights & Fault Characteristics

Number of Flights per Engine 50

Flight of Fault Initiation

○ Fixed ....

● Random .... 11 Minimum initiation flight

10 Minimum persistence (abrupt)

6 Minimum persistence (rapid)

Fault Evolution Rate

○ Abrupt

○ Rapid ....#flights =

● Random (~50% abrupt and ~50% rapid)

rapid fault evolution rate (minimum)............ 9

rapid fault evolution rate (maximum).......... 9

Sensor Noise

● On

○ Off

Run Function

Estimated Run Time (minutes) 12

Run Status .....

0%

Figure 8.—EFS Graphical User Interface

fault evolution rates. Faults that appear instantaneously, but do not grow in magnitude over time, are considered abrupt faults. Faults that initiate and grow in magnitude over time are considered rapid faults. All rapid faults will linearly increase in fault magnitude each flight, from the flight of fault initiation up until the flight where the fault "plateaus" at its assigned fault magnitude. Both abrupt and rapid fault magnitudes are assigned based on the fault magnitude distributions shown in Table 1. The options for specifying fault evolution rates are:

   a. Abrupt fault evolution: select the "Abrupt" fault evolution rate button.
   b. Rapid fault evolution: select the "Rapid" fault evolution rate button, and enter an integer ≥ "1" in the corresponding "# of flights =" box. This specifies the number of flights after fault initiation required for faults to grow to their assigned fault magnitude. For example, entering a value of "3" will produce rapid faults that are 25 percent of their assigned fault magnitude on the flight of fault initiation, and then linearly increase in magnitude to 50, 75 and 100 percent over the next three flights.
   c. Random fault evolution: select the "Random" fault evolution rate button. This will result in approximately 50 percent of the engines in the fleet experiencing abrupt faults, and approximately 50 percent experiencing rapid faults. If random fault evolution rates are chosen, the user must also specify the minimum and maximum number of flights over which rapid faults can evolve.

6. Choose the appropriate "Sensor Noise" button to turn sensor noise on or off. If sensor noise is selected, each engine will experience zero-mean, normally distributed sensor noise (measurement nonrepeatability). The noise standard deviation varies as a percentage of the actual sensed value as shown in Table 1.
7. After all the necessary inputs have been specified, Users can run the EFS by pressing the "Run EFS" button. Upon pressing "Run EFS" users will be queried to determine whether or not any existing EFS output files in the current directory should be overwritten. Selecting "Yes" will result in any existing files being overwritten. Selecting "No" will halt the program providing users the opportunity to archive any previously generated EFS output files that they wish to retain. The EFS software also includes error logic to ensure that users have not specified incompatible entries within the GUI. For example, requesting faults that will not initiate or fully evolve until after the last flight. If incompatible entries are found an error message box will be displayed, and the user will be required to modify the GUI entries. The "Estimated Run Time" provides an estimate of the time required to generate the requested data set, and the run status bar will automatically update to show progress towards completion. **Note:** It is recommended that users begin by generating data for a small number of engines to ensure that no errors are encountered when first running the EFS. Occasionally, engine operating points will be requested that the model has difficulty converging to. If this occurs a message will be displayed to the Matlab command prompt such as:

   ```
   Did not converge: Take Off - Engine #2 flight #5
   ```

   Any engine that experiences a nonconvergence operating point will be identified and removed from the archived EFS output files.

The ProDiMES evaluation metrics, which will be fully described in Section 4.0 of this guide, apply specific assumptions regarding the minimum flight of fault initiation and minimum fault persistence. The metrics assume that:

- No fault will initiate prior to flight 11
- Abrupt faults will persist for a minimum of 10 flights
- Rapid faults will persist for a minimum of 6 flights at their final fault magnitude

EFS warning logic is included to notify users if they specify GUI entries that will violate these assumptions. Upon receiving the warning, users will have the flexibility to either revise their GUI entries to be compatible with the metrics routine, or to continue generating EFS data as specified. A summary of the GUI entry requirements in order to maintain compatibility with the metrics routine is given in Table 2.

TABLE 2.—EFS GUI ENTRY REQUIREMENTS TO MAINTAIN COMPATIBILITY WITH METRICS ROUTINE

If "Flight of fault initiation" is …	If "Fault evolution rate" is …	Then metrics compatibility requirements are …
Fixed	*Abrupt*	• Flight of fault initiation ≥ 11 • Number Flights per Engine—Flight of fault initiation + 1 ≥ 10
	Rapid	• Flight of fault initiation ≥ 11 • Number Flights per Engine—Flight of fault initiation—Fault evolution rate + 1 ≥ 6
	Random	• Flight of fault initiation ≥ 11 • Number Flights per Engine—Flight of fault initiation + 1 ≥ 10 • Number Flights per Engine—Flight of fault initiation—Rapid fault evolution rate (maximum) + 1 ≥ 6
Random	*Abrupt*	• Random minimum initiation flight ≥ 11 • Minimum persistence (abrupt) ≥ 10 • Number Flights per Engine ≥ Random minimum initiation flight + Minimum persistence (abrupt) – 1
	Rapid	• Random minimum initiation flight ≥ 11 • Minimum persistence (rapid) ≥ 6 • Number Flights per Engine ≥ Random minimum initiation flight + Fault evolution rate + Minimum persistence (rapid) – 1
	Random	• Random minimum initiation flight ≥ 11 • Minimum persistence (abrupt) ≥ 10 • Minimum persistence (rapid) ≥ 6 • Number Flights per Engine ≥ Random minimum initiation flight + Minimum persistence (abrupt) – 1 • Number Flights per Engine ≥ Random minimum initiation flight + Rapid fault evolution rate (maximum) + Minimum persistence (rapid) – 1

EFS Output Files—Fault Conditions, Operating Conditions, and Parameter Histories

As the EFS runs it will generate and store three *.mat Matlab files in the `\EFS` directory consisting of fault conditions, operating conditions, and parameter histories. The contents of these files are further described below.

`EFS_fault_conditions.mat`: Contains the associated fault information for each engine in the fleet. This includes the fault type, fault magnitude, flight of fault initiation and fault evolution rate. This information comprises the actual, or "ground truth," fault information pertaining to the faults present within each engine generated by the EFS. The "ground truth" fault information will be used by the diagnostic metrics evaluation routine (to be described later) to evaluate diagnostic performance. This information is stored in the parameter "`fault_params`"—a 2D matrix of size #Engines × 5. The 5 columns of `fault_params` are:

1. Fault ID as listed in Table 1
2. Flight of fault initiation
3. Fault evolution rate
4. Fault magnitude
5. Ratio of module flow capacity to efficiency (only used for the 5 module faults).

`EFS_operating_conditions.mat`: Contains "`sensor_noise`" and "`engine_params`". `sensor_noise` is a scalar that represents noise on (1) or noise off (0). `engine_params` is a 3D matrix of dimension #Engines × #Flights × 18. The 18 parameters for each engine, each flight define:

1 to 4: Pressure altitude, Mach, ΔTamb, and Nf (fan speed) on takeoff
5 to 8: Pressure altitude, Mach, ΔTamb, and Fn (net thrust) at cruise
9 to 18: 10 module health parameters used to represent deterioration

`EFS_Output.mat`: Contains "`efs_output_to`" and "`efs_output_c`". These are both 3D matrices of dimension #Engines × #Flights × 11 sensed measurements. `efs_output_to` corresponds to the engine sensed measurements collected at takeoff, and `efs_output_c` corresponds to the engine sensed outputs collected at cruise. The 11 sensed parameters are shown in Table 3.

TABLE 3.—EFS ENGINE OUTPUT PARAMETERS

Index	Symbol	Description	Units
1	Nf	Physical fan speed	rpm
2	Nc	Physical core speed	rpm
3	P24	Total pressure at LPC outlet	psia
4	Ps30	Static pressure at HPC outlet	psia
5	T24	Total temperature at LPC outlet	°R
6	T30	Total temperature at HPC outlet	°R
7	T48	Total temperature at HPT outlet	°R
8	Wf	Fuel flow	pps
9	P2	Total pressure at fan inlet	psia
10	T2	Total temperature at fan inlet	°R
11	Pamb	Ambient pressure	psia

Figure 9 shows an example of the raw simulated engine output parameters produced by the EFS, and the corresponding features that can be extracted via a user provided diagnostic solution. Only 7 of the 11 sensed output parameters are shown as the other four, Nf, P2, T2 and Pamb, are used for power reference and parameter correction purposes in this example. The left hand side of this figure shows time history plots of the raw uncorrected cruise data collected for a single engine over 4000 flights. The raw data does reflect a gradual performance trend change over time due to the inclusion of deterioration effects. What is not as readily apparent from the raw data is the fact that this particular engine experiences a 5 percent fan fault on flight 2000. On the right, corrected residual data for the same engine over the same flight history is shown. This data was generated by correcting the EFS output parameters and referencing them against a fleet average engine model. Here the occurrence of the anomaly at flight 2000 is more readily apparent. The corrected residual data is not output directly by the EFS but is shown here to illustrate how such processed information can assist in anomaly detection. This is an example of the type of information users may choose to include within their individually developed diagnostic solutions. The example solution provided in Appendix B provides addition details to assist users with parameter correction and residual calculation techniques. Additionally, the routine `plot_corrected_data.m`, which is located in the `\Utilities` folder, provides an example of parameter correction and plotting for an individual engine.

3.0 User Provided Diagnostic Solutions

ProDiMES will enable end users to independently design and evaluate various gas path diagnostic methodologies. An example user provided diagnostic solution coded in Matlab is located in the folder `\ExampleSolution`, and further described in Appendix B of this guide. The User Provided Diagnostic Solution takes the output file from the EFS and generates an assessment file that can be analyzed to determine various metrics. In order to do this, it must correctly parse the EFS output file as well as output the assessment file in the specified format. Additionally, since the EFS output file contains all data from all of the engines and all of the flights, rules must be followed to ensure that causality of the diagnostics are respected.

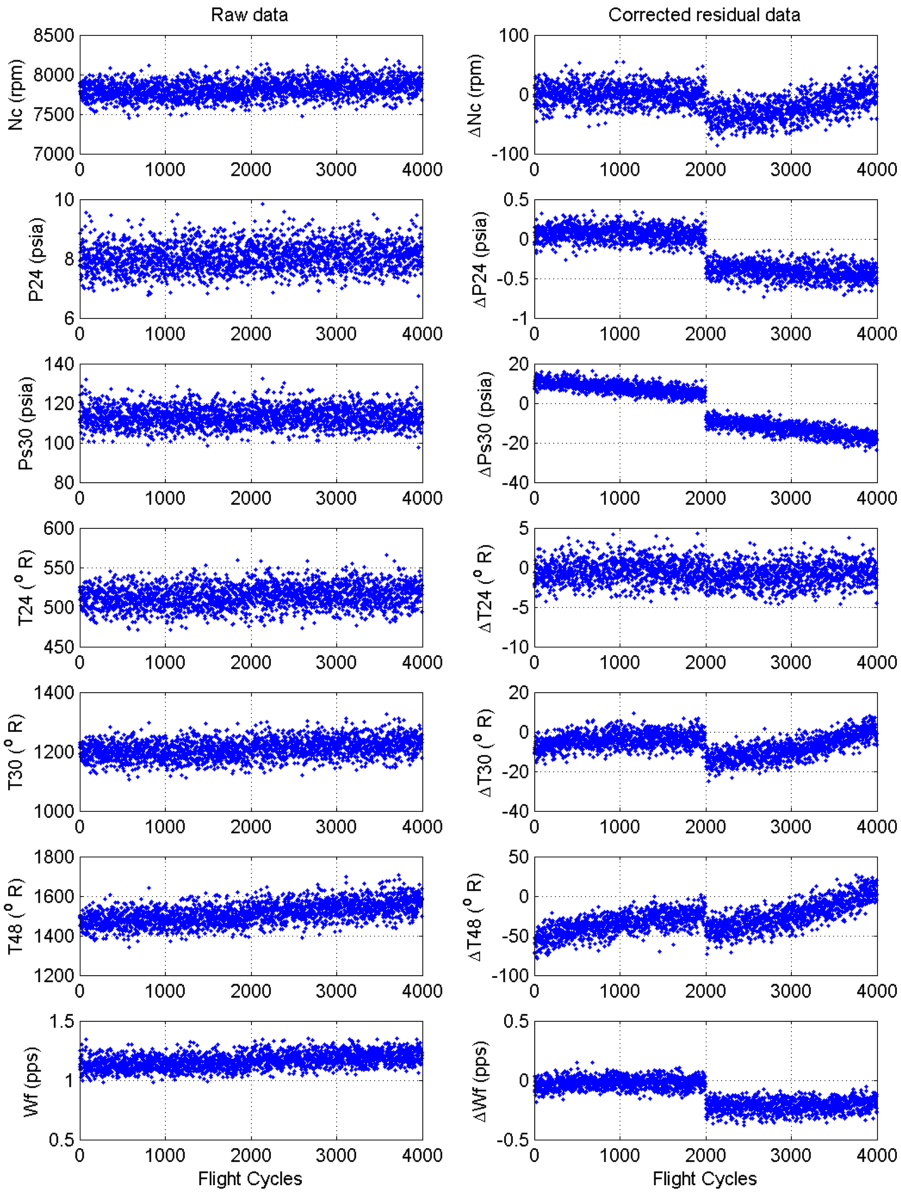


Figure 9.—Example of EFS output parameters (left) and corrected residual data (right)

3.1 Required Output Format

ProDiMES has been designed to provide users a large degree of flexibility to design individual diagnostic methods of their choice. However, there are certain input/output format and design restrictions that all methods must adhere to. Obviously, methods must be designed to read, parse, and process the simulated engine parameter histories produced by the EFS. Additionally, all diagnostic solutions must output diagnostic assessments in a standard format. The defined output format must be followed in order to maintain compatibility with the evaluation metrics routine, which will be described in the next section. The defined output format is as follows: Given EFS generated parameter histories for a fleet of engines, diagnostic solutions are required to produce a diagnostic assessment for each engine, at each flight. The diagnostic assessment for the fleet of engines is to be stored in a 2D Matlab matrix (named `diagnostic_assessment`) of dimension #Engines (rows) × #Flights (columns). For engines and flights where no fault is found a zero (0) is to be recorded in the matrix. On those engines and flights where a fault is diagnosed, the appropriate fault identification number (fault ID), as defined in Table 1, shall be recorded in the matrix. A notional example of the `diagnostic_assessment` matrix is shown in Figure 10. This reflects diagnostic assessments for a fleet of three engines over a ten flight history. For the first engine, shown in row one, no fault is diagnosed and thus a zero is recorded in each column. For the second engine, a fan fault (fault ID of 1) is diagnosed on flights three through ten. The third engine's diagnostic assessment is found to alternate between several assessments. A LPC fault (fault ID of 2) is diagnosed on flight six. The diagnosis then reverts back to no-fault (fault ID of 0) on flight seven, changes to Nf fault (fault ID of 8) on flight eight, and then LPC fault once again on flights nine and ten.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 8 & 2 & 2 \end{matrix}$$

Figure 10.—Example diagnostic_assessment matrix

The `diagnostic_assessment` matrix should be archived to a file named `DiagnosticAssessments.mat`. Users can specify a different filename for storing diagnostic assessments if they so choose. However, they should be aware that the metrics evaluation routine, which will be described in the next section, must also be modified to recognize the new filename. The example diagnostic solution provided in Appendix B illustrates the creation and archiving of the `diagnostic_assessment` matrix.

3.2 Design Restrictions

Some additional critical design restrictions that all users must be cognizant of when designing diagnostic solutions are the following: 1) all diagnostic solutions must be designed to function relying only on the available engine measurement information collected at and prior to the current flight cycle; and 2) once a diagnostic assessment is made at a given flight cycle, it cannot be retroactively adjusted based upon new knowledge gained from follow-on flights. The justification for these restrictions is to ensure that solutions maintain real-world relevance by producing timely diagnostic inferences after each flight. Although simulated engine measurement information over all flights will be available to end users, it is not permissible to use future flight information to produce diagnostic assessments on the current flight as such prophetic knowledge is not available in real-world applications. Furthermore, retroactively adjusting an assessment after the fact is problematic as presumably it could have already resulted in an unnecessary or incorrect maintenance action.

4.0 Evaluation Metrics

In order to enable a side-by-side comparison of candidate diagnostic methods, it is not only necessary to apply the methods to a common diagnostic problem, but also to evaluate their respective performance against a standard set of evaluation metrics. The ProDiMES software includes a uniform set of metrics that will enable solution developers to independently evaluate the performance of their individual solutions given the ground-truth fault output information produced by the EFS. This is performed by the Matlab routine `EvalMetrics.m`, which can be found in the `\Metrics` directory. It will automatically evaluate and store metric results in the Microsoft Excel spreadsheet `EvalMetrics.xls`, which can also be found in the `\Metrics` directory. These metrics are applied and categorized based on fault type, fault magnitude, and fault evolution rate (abrupt or rapid). The following subsections will provide a description of these metrics, along with operating instructions for evaluating diagnostics solutions against these metrics using the provided evaluation routine.

4.1 Description of Evaluation Metrics

The ProDiMES software assesses the overall detection and classification performance of candidate diagnostic methods through the following metrics:

1. True Positive Rate (number of **correct fault detections** divided by the number of fault cases)
2. False Negative Rate (number of **incorrect no fault detections** divided by the number of fault cases)
3. False Positive Rate (number of **incorrect fault detections** divided by the number of no fault cases)
4. True Negative Rate (number of **correct no fault detections** divided by the number of no fault cases)
5. Correct Classification Rate (number of **correct classifications** of a fault divided by the number of cases of that fault)
6. (Incorrect) Misclassification Rate (number of **incorrect classifications** of a fault divided by the number cases of that fault)
7. Detection latency
8. Classification latency
9. Kappa Coefficient

The detection metrics (1 through 4 in the above list) will be captured within the elements of a Detection Decision Matrix, and the classification metrics (5 and 6 in the above list) will be captured within the elements of a Classification Confusion Matrix. A further definition of these matrices and the applied metrics is provided below:

1. **Detection Decision Matrix:** A 2×2 matrix that reflects an algorithm's ability to discriminate between fault and no-fault cases. Its main diagonal reflects the number of correct predictions (true positives and true negatives) and its off-diagonal elements reflect the number of incorrect predictions (false negatives and false positives). The `EvalMetrics.m` routine generates a normalized detection decision matrix by dividing the elements of the matrix by the sum of its rows. This will reflect the **True Positive Rate** and **True Negative Rate** along the main diagonal, and the **False Positive Rate** and **False Negative Rate** on the off diagonal as shown in Figure 11. For example, correctly detecting 999 out of 1000 presented fault cases would result in a 0.999 in the upper left hand corner of the detection decision matrix.
2. **Classification Confusion Matrix:** Reflects an algorithm's ability to classify between multiple fault types as shown in Figure 12. The confusion matrix, denoted here as *C*, is a square matrix whose rows reflect the true fault condition and whose columns reflect the diagnosed fault condition. The main diagonal of the confusion matrix reflects correct classifications for each fault type, and any nonzero off-diagonal elements reflect incorrect classifications. As with the detection decision matrix, the `EvalMetrics.m` routine generates a normalized confusion matrix by dividing each element by the sum of its corresponding row. This results in the **Correct Classification Rate** for each fault type being displayed along the main diagonal, and fault **Misclassification Rate** being displayed on off-diagonal elements.
3. **Detection latency**: The average number of flights a fault must persist prior to true positive detection by the diagnostic algorithm.
4. **Classification latency**: The average number of flights a fault must persist prior to correct fault classification by the diagnostic algorithm.
5. **Kappa Coefficient**: Provides a measure of an algorithm's ability to correctly classify a fault, which takes into account the expected number of correct classifications occurring by chance. The Kappa Coefficient, denoted here as κ, is calculated from the elements of the un-normalized confusion matrix, *C*, as shown in the equation below. The two subscript indices represent the row and column corresponding to individual confusion matrix elements.

$$\kappa = \frac{N(\text{correctly classified}) - N(\text{expected correct by chance})}{N(\text{total}) - N(\text{expected correct by chance})}$$

where

$$\begin{aligned} N(\text{correctly classified}) &= \sum_{p=1}^{n} C_{pp} \\ N(\text{total}) &= \sum_{p=1}^{n}\sum_{q=1}^{n} C_{pq} \\ N(\text{expected correct by chance}) &= \sum_{p=1}^{n}\left\{\sum_{q=1}^{n}\frac{C_{pq}}{N(\text{total})}\cdot\sum_{q=1}^{n}C_{qp}\right\} \end{aligned} \tag{2}$$

If a diagnostic method achieves perfect fault classification performance then κ = 1. If its classification performance is worse than that expected by chance then κ < 0.

		Predicted State	
		Fault	**No Fault**
True State	**Fault**	*True Positive Rate*	*False Negative Rate* *(missed detections)*
	No Fault	*False Positive Rate* *(false alarms)*	*True Negative Rate*

Figure 11.—Detection decision matrix

		Predicted State			
		Fault 1	**Fault 2**	**. . .**	**Fault n**
True State	**Fault 1**	C_{11}	C_{12}	. . .	C_{1n}
	Fault 2	C_{21}	C_{22}	. . .	C_{2n}
	:	:	:	. . .	:
	Fault n	C_{n1}	C_{n2}	. . .	C_{nn}

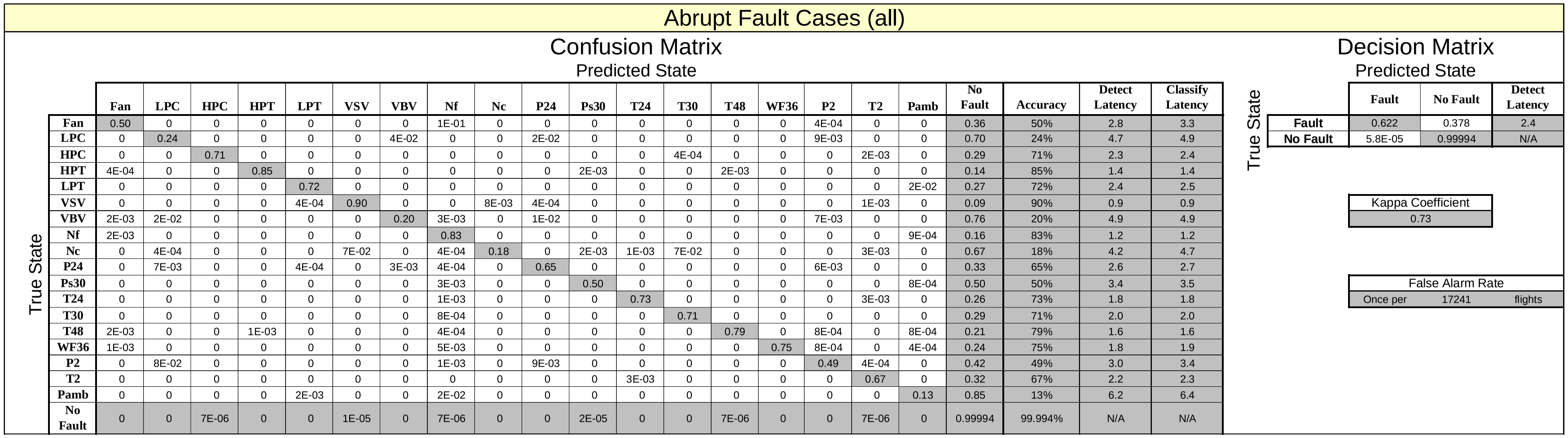
Figure 12.—Classification confusion matrix

An example portion of the `EvalMetrics.xls` spreadsheet produced by the ProDiMES metrics evaluation routine is shown in Figure 13. (An example of the entire metrics spreadsheet can be found in 0). The False Alarm Rate, given in number of flights, is automatically calculated within the Excel spreadsheet by inverting the False Positive Rate (lower left corner of the Detection Decision Matrix).

Partitioning of engine flight histories

Due to the format of the ProDiMES test cases and their associated diagnostic assessments, the application of the metrics described above is not a straightforward task. Consideration has to be given to the fact that over their operating history engines can experience both nominal and faulty behavior. Furthermore, as the diagnostic assessment is able to change from flight-to-flight, there may be flights where the assessment is "correct" and other flights where it is "incorrect". To account for these issues, the evaluation metrics routine partitions the flight history of each engine into separate operating regions, or "windows," and each flight within those windows is treated as an individual test case when applying the metrics. The applied engine flight history diagnostic window partitioning is shown in Figure 14. Here a notional example of the true condition (red dots) and the diagnosed condition (black circles) of a single engine over its provided flight history is shown. Any flight where the dot and circle are not concentric represents an incorrect assessment. This notional engine experiences a fault on flight *K*. The flight history is divided into the following four windows:

1. **Initial window (Flights 1 through 10)**: The ProDiMES metrics have been defined assuming that all engines will be fault-free for the first 10 flights of their flight history. This is done to provide a finite number of initial flights to allow solution providers to establish a performance baseline for each individual engine if they choose to do so. Consequently, the first 10 flights are excluded from metric calculations. If users intend to use the metrics evaluation routine, they should be aware of the no-fault assumption for the first 10 flights, and select the flight of fault occurrence accordingly. Furthermore, every engine in the provided blind test cases, to be described later, is guaranteed to have no fault present for the first 10 flights.
2. **Prefault window (Flights 11 through *K*–1)**: No fault is present. Diagnostic assessments are assessed for True Negative and False Positive detections. Note: Any engine that experiences a fault initiating on flight 11 will transition directly from the initial window to the fault window—it will not have a prefault window.
3. **Fault window**: This is a finite window of flights at, and immediately after, the flight of fault occurrence *K*. The fault window for abrupt faults is defined to be 10 flights long (flights *K* through *K*+9). The fault window for rapid faults is defined to be flights *K* through 5 flights after the flight where the fault is fully evolved (flights *K* through *K* + fault evolution rate + 5). In other words, the rapid fault window will encompass all flights from the flight of fault occurrence through the flight were the fault has persisted at its final fault magnitude for 6 flights. The fault window for abrupt and rapid faults is illustrated in Figure 15.
4. **Post-fault window**: Diagnostic assessments produced on all flights after the "fault window" are excluded from metric assessments. This is done to place more emphasis on the early detection and classification of faults.



Abrupt Fault Cases (all)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detect Latency	Classify Latency
Fan	0.50	0	0	0	0	0	0	1E-01	0	0	0	0	0	0	0	4E-04	0	0	0.36	50%	2.8	3.3
LPC	0	0.24	0	0	0	0	4E-02	0	0	2E-02	0	0	0	0	0	9E-03	0	0	0.70	24%	4.7	4.9
HPC	0	0	0.71	0	0	0	0	0	0	0	0	0	4E-04	0	0	0	2E-03	0	0.29	71%	2.3	2.4
HPT	4E-04	0	0	0.85	0	0	0	0	0	0	2E-03	0	0	2E-03	0	0	0	0	0.14	85%	1.4	1.4
LPT	0	0	0	0	0.72	0	0	0	0	0	0	0	0	0	0	0	0	2E-02	0.27	72%	2.4	2.5
VSV	0	0	0	0	4E-04	0.90	0	0	8E-03	4E-04	0	0	0	0	0	0	1E-03	0	0.09	90%	0.9	0.9
VBV	2E-03	2E-02	0	0	0	0	0.20	3E-03	0	1E-02	0	0	0	0	0	7E-03	0	0	0.76	20%	4.9	4.9
Nf	2E-03	0	0	0	0	0	0	0.83	0	0	0	0	0	0	0	0	0	9E-04	0.16	83%	1.2	1.2
Nc	0	4E-04	0	0	0	7E-02	0	4E-04	0.18	0	2E-03	1E-03	7E-02	0	0	0	3E-03	0	0.67	18%	4.2	4.7
P24	0	7E-03	0	0	4E-04	0	3E-03	4E-04	0	0.65	0	0	0	0	0	6E-03	0	0	0.33	65%	2.6	2.7
Ps30	0	0	0	0	0	0	0	3E-03	0	0	0.50	0	0	0	0	0	0	8E-04	0.50	50%	3.4	3.5
T24	0	0	0	0	0	0	0	1E-03	0	0	0	0.73	0	0	0	0	3E-03	0	0.26	73%	1.8	1.8
T30	0	0	0	0	0	0	0	8E-04	0	0	0	0	0.71	0	0	0	0	0	0.29	71%	2.0	2.0
T48	2E-03	0	0	1E-03	0	0	0	4E-04	0	0	0	0	0	0.79	0	8E-04	0	8E-04	0.21	79%	1.6	1.6
WF36	1E-03	0	0	0	0	0	0	5E-03	0	0	0	0	0	0	0.75	8E-04	0	4E-04	0.24	75%	1.8	1.9
P2	0	8E-02	0	0	0	0	0	1E-03	0	9E-03	0	0	0	0	0	0.49	4E-04	0	0.42	49%	3.0	3.4
T2	0	0	0	0	0	0	0	0	0	0	0	3E-03	0	0	0	0	0.67	0	0.32	67%	2.2	2.3
Pamb	0	0	0	0	2E-03	0	0	2E-02	0	0	0	0	0	0	0	0	0	0.13	0.85	13%	6.2	6.4
No Fault	0	0	7E-06	0	0	1E-05	0	7E-06	0	0	2E-05	0	0	7E-06	0	0	7E-06	0	0.99994	99.994%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detect Latency
Fault	0.622	0.378	2.4
No Fault	5.8E-05	0.99994	N/A

Kappa Coefficient
0.73

False Alarm Rate		
Once per	17241	flights

Figure 13.—Example portion of metric results archived by ProDiMES.

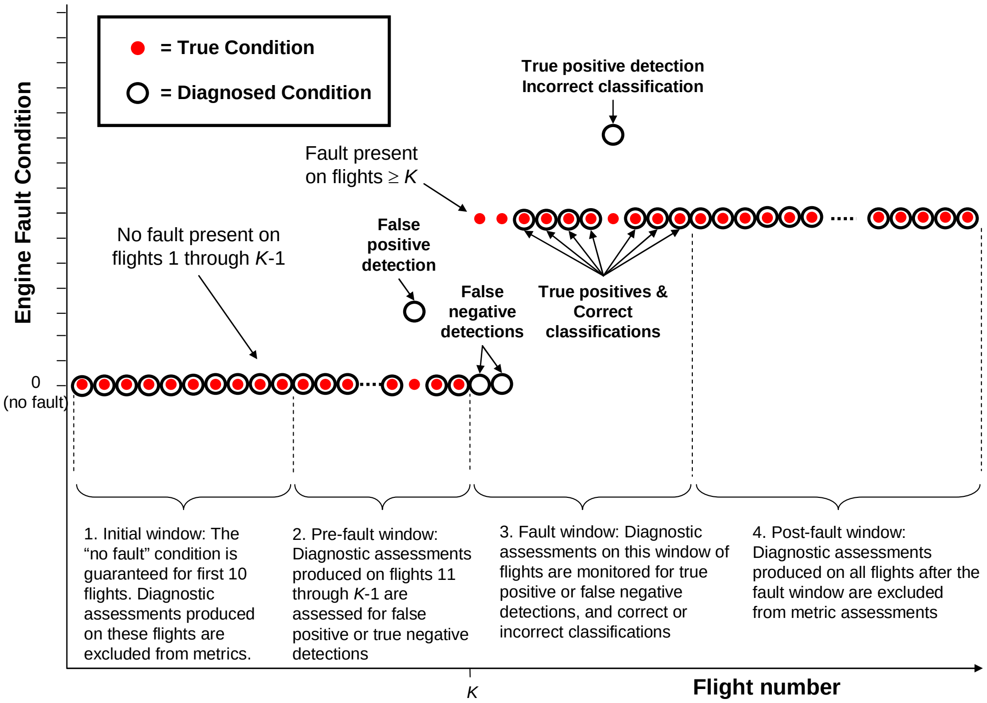

● = True Condition
○ = Diagnosed Condition
Engine Fault Condition
0
(no fault)
True positive detection
Incorrect classification
Fault present
on flights ≥ K
No fault present on
flights 1 through K-1
False
positive
detection
False
negative
detections
True positives &
Correct
classifications
1. Initial window: The "no fault" condition is guaranteed for first 10 flights. Diagnostic assessments produced on these flights are excluded from metrics.
2. Pre-fault window: Diagnostic assessments produced on flights 11 through K-1 are assessed for false positive or true negative detections
3. Fault window: Diagnostic assessments on this window of flights are monitored for true positive or false negative detections, and correct or incorrect classifications
4. Post-fault window: Diagnostic assessments produced on all flights after the fault window are excluded from metric assessments
K
Flight number

Figure 14.—Diagnostic window partitioning of engine flight history.

Note that the fault window and post-fault window only apply for those engines that experience a fault. For those engines that experience no faults, the prefault window will consist of all flights after flight 10.

As a further illustration of the application of diagnostic performance metrics, and the partitioning of engine flight histories into the defined windows, refer to Figure 16 which shows the true (red dots) and the diagnosed condition (black circles) for a ***single*** engine over 50 flights. (Note: This single engine example is simply given to illustrate flight window partitioning and the resultant metric calculations. Obtaining an accurate assessment of a given diagnostic method's performance will require evaluation against a suitably rich database encompassing numerous instances of each fault type). On flight 25 this engine experiences an abrupt fan fault (fault ID 1). On flight 27 the fault is correctly detected, but incorrectly classified as fault ID 8. On flights 28 through 50 the fault is both correctly detected and correctly classified as fault ID 1. This would produce the following flight history windows and metric results:

- **Initial window**: Flights 1 through 10 are excluded from metric calculations
- **Prefault window**: Consists of flights 11 through 24. During these 14 flights no false positive detections occur. Thus the true negative rate is 14 out of 14 flights (1.0) and the false positive rate is 0 out of 14 (0.0).
- **Fault window**: Consists of flights 25 through 34. During these 10 flights:
  - True positive fault detection occurs 8 out of 10 times (0.8)
  - Correct fault classification occurs 7 out of 10 times (0.7)
  - False negatives occur on 2 of 10 flights (0.2)
  - Misclassification of fault 1 as fault 8 occurs once (0.1)
  - Fault detection latency is 2 flights
  - Fault classification latency is 3 flights
- **Post-fault window**: Flights 35 through 50 are excluded from metric calculations.

The file `plot_true_vs_diagnosed_condition.m`, located in the `\Utilities` folder, contains the Matlab code used to generate Figure 16.

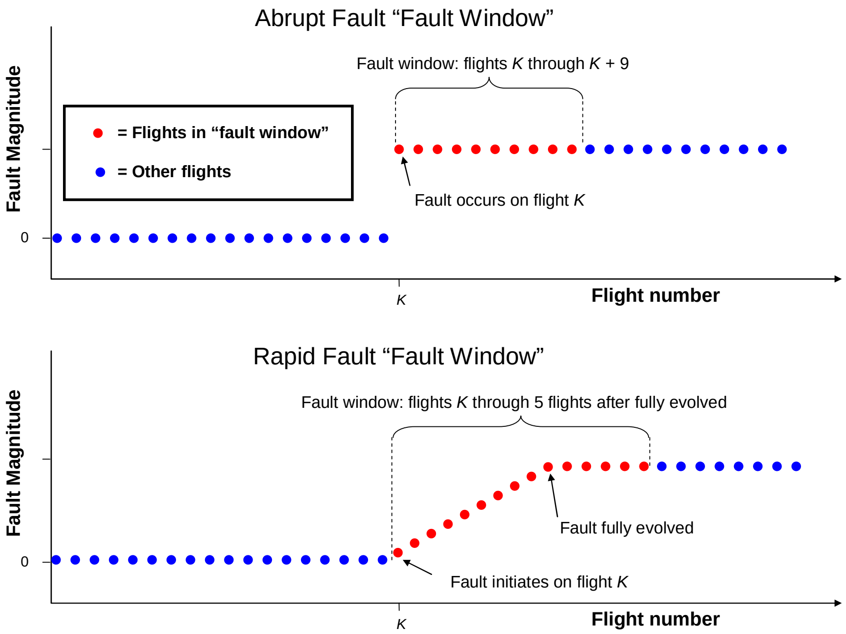

Abrupt Fault “Fault Window”
Fault window: flights K through K + 9
● = Flights in “fault window”
● = Other flights
Fault Magnitude
0
Fault occurs on flight K
K
Flight number
Rapid Fault “Fault Window”
Fault window: flights K through 5 flights after fully evolved
Fault Magnitude
0
Fault fully evolved
Fault initiates on flight K
K
Flight number

Figure 15.—Defined "fault window" for abrupt and rapid faults.

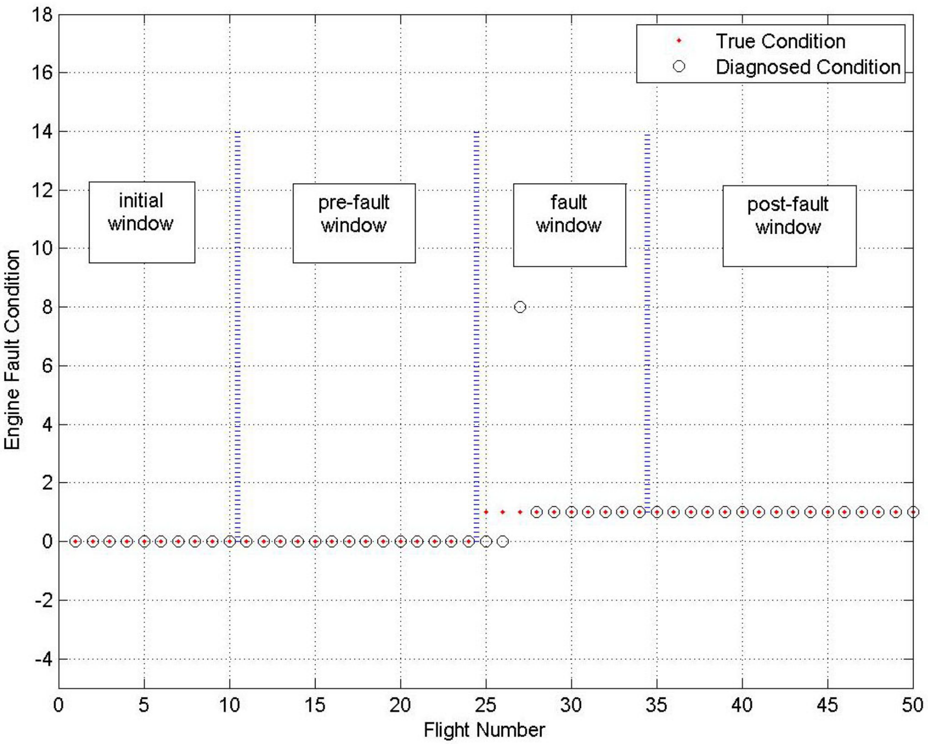


Figure 16.—Example: True versus diagnosed engine condition.

When using the supplied metric evaluation routine, users should remain cognizant of the flight window partitioning described above, and define test case inputs through the EFS GUI accordingly. For example, no faults should be specified to initiate prior to flight 11, all abrupt faults should persist for at least 10 flights, and all rapid faults should persist for at least 6 flights. If users specify EFS GUI entries that violate any of the metrics compatibility requirements, appropriate warning messages will be displayed to notify the user of the incompatibility. For a complete list of the EFS GUI entry requirements necessary to maintain compatibility with the metrics evaluation routine users are referred to Table 2.

Metrics: Fault evolution rate and fault magnitude

The degree of difficulty in correctly diagnosing a fault not only varies by fault type, but also by fault magnitude and fault evolution rate. In general, large abrupt faults will be easier to diagnosis than small rapid faults. To account for this, the evaluation metrics are categorized based on the fault evolution rate (abrupt or rapid), and fault magnitude (small, medium or large) as defined in Table 4. (Note: Fault magnitudes vary over a uniform continuous distribution and can be any value within the defined ranges shown in Table 4. There are no discrete steps within the defined fault magnitude distributions). Diagnostic performance metrics will be calculated and archived for each of the following eight categories:

1. Abrupt faults (all magnitudes)
2. Abrupt faults (small)
3. Abrupt faults (medium)
4. Abrupt faults (large)
5. Rapid faults (all magnitudes)
6. Rapid faults (small)
7. Rapid faults (medium)
8. Rapid faults (large)

TABLE 4.—DEFINED GAS PATH FAULT MAGNITUDES

Fault ID	Fault description	Fault magnitude	Small	Medium	Large
0	No-fault	----------	----------	----------	----------
1	Fan fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
2	LPC fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
3	HPC fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
4	HPT fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
5	LPT fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
6	VSV fault	1 to 7%	1 to 3%	3 to 5%	5 to 7%
7	VBV fault	1 to 19%	1 to 7%	7 to 13%	13 to 19%
8	Nf sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
9	Nc sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
10	P24 sens. fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
11	Ps30 sens. fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
12	T24 sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
13	T30 sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
14	T48 sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
15	Wf sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
16	P2 sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
17	T2 sensor fault	± 1 to 10 σ	1 to 4σ	4 to 7σ	7 to 10σ
18	Pamb sens fault	± 1 to 19 σ	1 to 7σ	7 to 13σ	13 to 19σ

4.2 Evaluation Metrics Routine—Operating Instructions

The evaluation metrics software is located in the directory `\Metrics`. The steps for using this software are as follows:

1. Prior to running the evaluation metrics routine, users are required to generate and place a `diagnostic_assessment` matrix file, and an EFS generated “ground truth” fault information file in the `\Metrics` directory. The evaluation metrics routine is setup to run assuming these two files, named `Diagnostic_Assessment.mat` and `EFS_fault_conditions.mat` respectively, are present in the `\Metrics` directory. Users can use different file names if they so chose, but must edit the evaluation metrics routine, `EvalMetrics.m`, accordingly.
2. In Matlab, navigate to the `\Metrics` directory, and type the command `EvalMetrics`.
3. Upon completion the metrics evaluation routine will archive results to the Microsoft Excel spreadsheet `EvalMetrics.xls`.

Note: Each time the `EvalMetrics.m` routine is run it will check for the existence of `EvalMetrics.xls` within the `\Metrics` directory. If this file exists users will be asked if they would like to overwrite its contents. If a user specifies “Yes,” the contents of `EvalMetrics.m` will be overwritten. Specifying “No” will terminate the routine, giving users the opportunity to archive the existing file. A template version of the metrics spreadsheet, `EvalMetrics_template.xls`, is also located in the `\Metrics` directory. This file contains formatting necessary to interpret metric results. It should not be deleted or renamed.

5.0 Blind Test Cases

The ProDiMES software is distributed with a set of blind test case data to enable a comparison between diagnostic methods developed by multiple participants. The following subsections provide additional information regarding blind test data to assist solution providers in developing and submitting diagnostic solutions.

5.1 Blind Test Case Information

The blind test case data is provided in the file `EFS_Output_BlindTest.mat` located in the `\BlindTestCases` directory. This data set has the following characteristics:

1. Not every engine will experience a fault.
2. Only single fault scenarios are included. No individual engine will experience more than one fault during its provided time history.
3. The blind test case data was generated using the EFS although the engines have been randomly re-ordered so that they are not arranged sequentially according to fault_ID. The EFS GUI inputs specified to generate the blind test case data were identical to those previously shown in Figure 8, with the exception that the number of occurrences of each of the no fault/fault types is different. The remaining GUI inputs are the same. This means that blind test case data will have the following characteristics:
   a. Both abrupt and rapid fault types will be included in the test cases. Abrupt faults will appear instantaneously, while rapid faults will occur and evolve linearly to a randomly selected fault magnitude. The number of flights required for the rapid faults to fully evolve (after fault initiation) will be exactly 9 flights.

b. No fault will be present for the first 10 flights of every engine. This is done to provide an initial window of time for establishment of a performance baseline for each individual engine if solution providers choose to do so.
c. If an abrupt fault does occur in an engine, it will be fully evolved for at least 10 flights prior to the end of the provided time history. If a rapid fault occurs, it will be fully evolved for at least 6 flights prior to the end of the provided time history.
d. 50 flights of data are provided for each engine. Abrupt faults can initiate on any flight between flights 11 and 41. Rapid faults can initiate on any flight between flights 11 and 36.

4. Diagnostic assessments are to be based on the data collected at, and prior to, the current flight. Future flight data cannot be utilized to improve the diagnostic assessment on the current flight. Furthermore, previously made diagnostic assessments cannot be retroactively adjusted based on new data.

5.2 Submitting Blind Test Case Results

Participants should generate their blind test case diagnostic assessments in the appropriate format, as previously defined in Section 3.0. These diagnostic assessments should be submitted to Don Simon of the NASA Glenn Research Center:

Email: Donald.L.Simon@nasa.gov
Phone: 216–433–3740

NASA will evaluate provided results against the ground truth fault information using the metrics evaluation routine. In an attempt to maintain a level of uniformity in the diagnostic solutions applied, a target false positive detection rate (false alarm rate) of once per 1,000 flights is specified. As initial feedback, submitters will only be provided their false alarm rate metrics. If a submission fails to meet the target false alarm rate of once per 1,000 flights, the submitter will be made aware of this and asked to resubmit their results applying less aggressive thresholds. Similarly, those participants whose results have a much lower false alarm rate than the target rate will have the opportunity to resubmit results applying tighter thresholds. This is done in an attempt to maintain uniformity in the submitted results. Once the final collection of blind test case results have been submitted, participants will receive the complete set of evaluation metric results for their submission, in addition to the anonymous results of other participants.

6.0 Future Workshop

A future workshop will be convened to share solution results and lessons learned from this process. The intent is not to formulate this as a competition, but rather as a means for the engine health management community to share diagnostic approaches. For additional information, and updates on the current schedule and status of this effort, individuals are directed to visit the following website:

www.grc.nasa.gov/WWW/cdtb/software/ehmbenchmark.html

7.0 Requested Feedback

Questions, comments, and suggestions for improving ProDiMES are welcomed. This includes the reporting of any problems or errors found in the operation of the code. Please refer all questions and comments to:

Email: Donald.L.Simon@nasa.gov
Phone: 216–433–3740

Appendix A.—ProDiMES Software Parameter List

This appendix provides a summary of the software parameters used within ProDiMES. This information is organized below according to software routines.

EFS_GUI.m: This is the main EFS GUI function. It reads in user entries specified through the GUI, and generates several outputs used in the generation of EFS sensed parameter histories. It is located in the `\EFS` directory.

TABLE A.1.—EFS_GUI.m PARAMETER LIST

	Parameter name	Description
Outputs	engine_condition	Specifies the number of occurrences of each no-fault / fault condition (19 element vector).
	fault_evolution_rate	Specifies fault progression rate: 0: abrupt. > = 1: number of flights (after fault initiation) over which rapid faults evolve. –1: Random (produces ~ 50% abrupt and 50% rapid faults).
	fault_initiation_flight	> = 1: specifies flight of fault initiation. –1: specifies random flight of fault initiation.
	flight_cycles	Specifies the number of flight cycles for each engine.
	min_fault_initiation_flight	Specifies the earliest flight of fault occurrence (random initiation only).
	min_fault_persist_abrupt	Specifies the minimum number of flight cycles an abrupt fault will persist before end of recorded flight history (random initiation only).
	min_fault_persist_rapid	Specifies the minimum number of flight cycles a rapid fault will persist at its final magnitude before end of recorded flight history (random initiation only).
	num_engines	Specifies the number of engines in the fleet.
	rapid_fault_evolution_max	Specifies maximum rapid fault evolution rate (random initiation only).
	rapid_fault_evolution_min	Specifies minimum rapid fault evolution rate (random initiation only).
	sensor_noise	Scalar that specifies sensor noise on (1) or off (0).

EFS_FaultParams.m: This subroutine generates fault parameters (fault type and magnitude) for each engine in the fleet. It also assigns the ratio of flow capacity to efficiency for component faults. It is located in the `\EFS` directory.

TABLE A.2.—EFS_FaultParams.m PARAMETER LIST

	Parameter name	Description
Inputs	engine_condition	Specifies the number of occurrences of each no-fault / fault condition (19 element vector).
	fault_evolution_rate	Specifies fault progression rate: 0: abrupt. > =1: number of flights (after fault initiation) over which rapid faults evolve. –1: Random (produces ~ 50% abrupt and 50% rapid faults).
	fault_initiation_flight	> = 1: specifies flight of fault initiation. –1: specifies random flight of fault initiation.
	flight_cycles	Specifies the number of flight cycles for each engine.
	min_fault_initiation_flight	Specifies the earliest flight of fault occurrence (random initiation only).
	min_fault_persist_abrupt	Specifies the minimum number of flight cycles an abrupt fault will persist before end of recorded flight history (random initiation only).
	min_fault_persist_rapid	Specifies the minimum number of flight cycles a rapid fault will persist at its final magnitude before end of recorded flight history (random initiation only).
	num_engines	Specifies the number of engines in the fleet.
	rapid_fault_evolution_max	Specifies maximum rapid fault evolution rate (random initiation only).
	rapid_fault_evolution_min	Specifies minimum rapid fault evolution rate (random initiation only).
Outputs	fault_params	2D array (dimension #engines × 5) defining engine conditions including: 1. Fault ID 2. Flight of fault initiation 3. Fault evolution rate 4. Fault magnitude 5. Ratio of flow capacity to efficiency (used for component faults only)

EFS_OperatingConditions_Deterioration.m: This subroutine generates random operating conditions at takeoff and cruise for each engine in the fleet. It also assigns a random deterioration profile for each engine in the fleet. It is located in the `\EFS` directory.

TABLE A.3.—EFS_OperatingConditions_deterioration.m PARAMETER LIST

	Parameter name	Description
Inputs	num_engines	Specifies the number of engines in the fleet.
	flight_cycles	Specifies the number of flight cycles for each engine.
Outputs	engine_params	3D array (number_engines × flight_cycles × 18 outputs). 18 outputs are: 1. Altitude (takeoff) 2. Mach (takeoff) 3. dTamb (takeoff) 4. NfR (takeoff) 5. Atlitude (cruise) 6. Mach (cruise) 7. dTamb (cruise) 8. Fn (cruise) 9-18. 10 health parameters reflecting engine performance deterioration

EFS2CMAPSS_C8.m: Processes EFS specified engine operating points through the C-MAPSS Steady-State Model to generate sensed parameter histories. This routine contains the steady-state solver used to balance the C-MAPSS Steady-State engine model at each engine operating point. It is located in the `\EFS` directory.

TABLE A.4.—EFS2CMAPSS_C8.m PARAMETER LIST

	Parameter name	Description
Inputs	Jinv_c_Fn, Jinv_c_Nc, Jinv_c_Nf, Jinv_c_T48, Jinv_to_Nc, Jinv_to_Nf, Jinv_to_T48	C-MAPSS Steady-State inverse Jacobian matrices used by the steady-state solver. These matrices are loaded from the file `\JacobiansTrim\Jacobians_CMAPSS.mat`.
	fault_params	2D array of dimension (number_engines × 5 outputs) that defines fleet fault conditions. This array is generated by `EFS_FaultParams.m` and stored in the file `EFS_fault_conditions.mat`.
	engine_params	3D array of dimension (number_engines × flight_cycles × 18 outputs) that defines fleet operating conditions. This array is generated by `EFS_OperatingConditions_Deterioration.m` and stored in the file `EFS_operating_conditions.mat`.
	sensor_noise	Scalar that specifies sensor noise on (1) or off (0). This parameter is specified through the EFS_GUI and stored in the file `EFS_operating_conditions.mat`.
	Fn_zro, MN_zro, Nc_zro, Nf_zro, P2_zro, P24_zro, Ps30_zro, Pamb_zro, T2_zro, T24_zro, T30_zro, T48_zro, Wf_zro, X_zro, alt_zro, dTamb_zro	Fleet average engine parameter trim values used for initializing C-MAPSS Steady-State engine model and specifying sensor fault magnitudes. These parameters are stored in the files `\JacobiansTrim\FC_cruise.mat` (cruise) and `\JacobiansTrim\FC_takeoff.mat` (takeoff).
Outputs	efs_output_c	Engine fleet sensed parameter histories generated at cruise (stored to `EFS_Output.mat`).
	efs_output_to	Engine fleet sensed parameter histories generated at takeoff (stored to `EFS_Output.mat`).

city_pairs.m: This function will randomly select the city pairs each engine will fly between. It is called by `EFS_OperatingConditions_Deterioration.m`, and is located in the `\EFS` directory.

TABLE A.5.—city_pairs.m PARAMETER LIST

	Parameter name	Description
Inputs	num_engines	Specifies the number of engines in the fleet.
Outputs	start_end_city	2D array that defines city pairs (airports) each engine in the fleet will fly between.

calculate_HPs.m: This subroutine calculates and returns a deterioration profile for each engine. Deterioration is represented by 10 health parameters that capture efficiency and flow capacity shifts in each of the 5 major modules of the engine. It is called by `EFS_OperatingConditions_Deterioration.m`, and is located in the `\EFS` directory.

TABLE A.6.—calculate_HPs.m PARAMETER LIST

	Parameter name	Description
Inputs	num_engines	Specifies the number of engines in the fleet.
	flight_cycles	Specifies the number of flight cycles for each engine.
Outputs	hp_array	3D array that defines health parameter deterioration profile for each engine (dimension #engines × #flights × 10 health parameters).

cmapss_ss_function_C.mexw32: C-MAPSS steady-state engine model MEX function located in the `\MexFile` directory.

TABLE A.7.—cmapss_ss_function_C.mexw32 PARAMETER LIST

	Parameter name	Description
Inputs	x	Vector of eight (8) independent variables used to balance steady-state model at specified operating conditions. Includes five module operating line points, two state variables (fan and core speed), and fuel flow.
	op_cond	Three (3) element vector specifying engine environmental operating conditions (altitude, dTamb, and Mach).
	hp_vector	Ten (10) element vector specifying engine health parameters.
	bias_vector	Five (5) element vector used to specify the fault magnitude in variable geometry actuators (VBV, VSV) or sensors that schedule variable geometry actuators (T2, T24, Nf, Nc).
	Fn_target	Scalar specifying target net thrust (used to balance cruise steady-state points)
Outputs	z	Vector of eight (8) error terms used by the steady-state solver to balance the C-MAPSS steady-state engine model. Includes five (5) flow errors, two (2) state derivatives, and error between actual and target net thrust.
	y	Vector of eleven (11) C-MAPSS Steady-State engine model outputs.

EvalMetrics.m: Compares user provided diagnostic assessments against ground truth fault information to assess performance against defined metrics. This file is located in the `\Metrics` directory. All outputs are archived to the Microsoft Excel spreadsheet `EvalMetrics.xls`. The information is partitioned into two separate worksheets within the Excel file; one representing abrupt fault diagnostic metrics, and the second representing rapid fault diagnostic metrics.

TABLE A.8.—EvalMetrics.m PARAMETER LIST

	Parameter name	Description
Inputs	fault_params	2D array of dimension (number_engines × 5 outputs) which defines ground truth fault conditions. Outputs are: 1. Fault type 2. Fault initiation flight 3. Fault evolution window 4. Fault magnitude 5. Ratio of flow capacity to efficiency (for component faults only)
	diagnostic_assessment	2D array of dimension (num_engines × flight_cycles) that contains the diagnostic assessment for a given fleet of engine data. This assessment is to be generated by the diagnostic solution provided by the user, and stored in the file `DiagnosticAssessments.mat`.
Outputs	confusion_matrix_abrupt confusion_matrix_abrupt_small confusion_matrix_abrupt_medium confusion_matrix_abrupt_large confusion_matrix_rapid confusion_matrix_rapid_small confusion_matrix_rapid_medium confusion_matrix_rapid_large	Classification confusion matrices
	decision_matrix_abrupt decision_matrix_abrupt_small decision_matrix_abrupt_medium decision_matrix_abrupt_large decision_matrix_rapid decision_matrix_rapid_small decision_matrix_rapid_medium decision_matrix_rapid_large	Detection decision matrices
	tp_latency	True positive detection latency
	cc_latency	Correct classification latency
	KappaCoefficient_abrupt KappaCoefficient_abrupt_small KappaCoefficient_abrupt_medium KappaCoefficient_abrupt_large KappaCoefficient_rapid KappaCoefficient_rapid_small KappaCoefficient_rapid_medium KappaCoefficient_rapid_large	Kappa coefficients

Examples of the EvalMetrics.xls worksheets produced by the ProDiMES metrics evaluation routine are shown in Figure A.1 (abrupt faults) and Figure A.2 (rapid faults).

Abrupt Fault Cases (all)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.50	0	0	0	0	0	0	1E-01	0	0	0	0	0	0	0	0	0	0	0.40	50%	2.9	3.3
LPC	0	0.28	0	0	0	0	3E-02	0	0	2E-02	0	0	0	0	0	2E-02	0	0	0.65	28%	4.6	4.9
HPC	0	0	0.72	0	0	0	0	2E-03	0	0	0	0	4E-03	0	0	0	0	7E-04	0.28	72%	2.3	2.3
HPT	0	0	0	0.86	0	0	0	0	0	0	2E-03	0	0	0	0	0	0	9E-04	0.14	86%	1.4	1.4
LPT	0	0	0	0	0.69	0	0	0	0	0	0	0	0	0	0	0	0	9E-03	0.31	69%	2.6	2.6
VSV	0	0	0	0	3E-03	0.89	0	0	2E-02	0	0	0	0	0	9E-04	0	9E-04	0	0.08	89%	0.8	1.0
VBV	0	2E-02	0	0	0	0	0.21	2E-03	0	4E-03	0	0	0	0	0	1E-02	0	0	0.75	21%	5.4	5.4
Nf	7E-04	0	0	0	0	0	0	0.81	0	0	0	0	0	0	0	0	0	0	0.19	81%	1.4	1.4
Nc	0	0	0	0	0	1E-01	0	1E-03	0.18	0	0	0	5E-02	0	0	0	4E-03	0	0.65	18%	4.0	4.7
P24	0	4E-03	0	0	0	0	7E-04	1E-03	0	0.56	0	0	0	0	0	6E-03	0	0	0.43	56%	2.8	2.8
Ps30	0	0	0	0	0	0	0	8E-04	0	0	0.54	0	0	0	0	0	0	0	0.46	54%	3.5	3.5
T24	0	0	0	0	0	0	0	0	0	0	0	0.73	0	0	0	0	2E-03	0	0.26	73%	1.7	1.7
T30	0	0	0	0	0	0	0	2E-03	0	0	0	0	0.66	0	0	0	1E-03	0	0.34	66%	1.9	2.0
T48	4E-03	0	0	9E-04	0	0	0	2E-03	0	0	0	0	0	0.83	0	0	0	0	0.16	83%	1.6	1.5
WF36	3E-03	0	0	0	0	0	0	2E-03	0	0	0	0	0	0	0.78	6E-04	0	6E-04	0.21	78%	1.6	1.7
P2	0	8E-02	0	0	0	0	6E-04	0	0	7E-03	0	0	0	0	0	0.48	6E-04	0	0.43	48%	3.1	3.4
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.66	0	0.34	66%	2.1	2.1
Pamb	2E-03	0	0	0	4E-03	0	0	3E-02	0	0	0	0	0	0	0	0	0	0.11	0.86	11%	6.2	6.4
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.610	0.390	2.5
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.73

False Alarm Rate		
Once per	37564	flights

Abrupt Fault Cases (small)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.23	0	0	0	0	0	0	7E-02	0	0	0	0	0	0	0	0	0	0	0.71	23%	5.4	5.4
LPC	0	0.02	0	0	0	0	0	0	0	5E-03	0	0	0	0	0	2E-03	0	0	0.97	2%	7.0	6.3
HPC	0	0	0.46	0	0	0	0	4E-03	0	0	0	0	1E-02	0	0	0	0	0	0.53	46%	4.3	4.4
HPT	0	0	0	0.70	0	0	0	0	0	0	3E-03	0	0	0	0	0	0	0	0.30	70%	3.0	3.0
LPT	0	0	0	0	0.42	0	0	0	0	0	0	0	0	0	0	0	0	2E-02	0.57	42%	4.7	4.8
VSV	0	0	0	0	8E-03	0.79	0	0	1E-02	0	0	0	0	0	0	0	0	0	0.19	79%	1.9	2.0
VBV	0	0	0	0	0	0	0.00	2E-03	0	0	0	0	0	0	0	9E-03	0	0	0.99	0%	6.5	
Nf	2E-03	0	0	0	0	0	0	0.54	0	0	0	0	0	0	0	0	0	0	0.46	54%	3.5	3.5
Nc	0	0	0	0	0	0	0	0	0.00	0	0	0	3E-03	0	0	0	0	0	0.99	0%	7.5	7.0
P24	0	6E-03	0	0	0	0	2E-03	3E-03	0	0.17	0	0	0	0	0	1E-02	0	0	0.81	17%	6.2	6.3
Ps30	0	0	0	0	0	0	0	0	0	0	0.11	0	0	0	0	0	0	0	0.89	11%	7.1	7.1
T24	0	0	0	0	0	0	0	0	0	0	0	0.36	0	0	0	0	5E-03	0	0.63	36%	4.2	4.2
T30	0	0	0	0	0	0	0	2E-03	0	0	0	0	0.24	0	0	0	0	0	0.76	24%	4.9	4.9
T48	5E-03	0	0	0	0	0	0	5E-03	0	0	0	0	0	0.66	0	0	0	0	0.33	66%	3.2	3.1
WF36	1E-02	0	0	0	0	0	0	7E-03	0	0	0	0	0	0	0.49	2E-03	0	2E-03	0.49	49%	3.7	3.9
P2	0	2E-02	0	0	0	0	0	0	0	7E-03	0	0	0	0	0	0.16	0	0	0.82	16%	6.2	6.3
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.29	0	0.71	29%	4.5	4.5
Pamb	0	0	0	0	0	0	0	5E-03	0	0	0	0	0	0	0	0	0	0.00	1.00	0%	7.0	
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.312	0.688	4.3
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.46

Abrupt Fault Cases (medium)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.68	0	0	0	0	0	0	8E-02	0	0	0	0	0	0	0	0	0	0	0.25	68%	2.5	2.9
LPC	0	0.25	0	0	0	0	3E-02	0	0	2E-02	0	0	0	0	0	4E-02	0	0	0.67	25%	5.7	5.9
HPC	0	0	0.80	0	0	0	0	2E-03	0	0	0	0	0	0	0	0	0	2E-03	0.19	80%	1.9	2.0
HPT	0	0	0	0.91	0	0	0	0	0	0	3E-03	0	0	0	0	0	0	3E-03	0.08	91%	0.8	0.9
LPT	0	0	0	0	0.78	0	0	0	0	0	0	0	0	0	0	0	0	9E-03	0.22	78%	2.2	2.3
VSV	0	0	0	0	0	0.94	0	0	3E-03	0	0	0	0	0	3E-03	0	0	0	0.06	94%	0.6	0.6
VBV	0	3E-02	0	0	0	0	0.16	2E-03	0	0	0	0	0	0	0	9E-03	0	0	0.80	16%	6.9	7.0
Nf	0	0	0	0	0	0	0	0.91	0	0	0	0	0	0	0	0	0	0	0.09	91%	0.9	0.9
Nc	0	0	0	0	0	4E-02	0	2E-03	0.21	0	0	0	3E-02	0	0	0	1E-02	0	0.71	21%	6.0	6.3
P24	0	5E-03	0	0	0	0	0	0	0	0.78	0	0	0	0	0	0	0	0	0.21	78%	2.1	2.1
Ps30	0	0	0	0	0	0	0	0	0	0	0.63	0	0	0	0	0	0	0	0.37	63%	3.6	3.6
T24	0	0	0	0	0	0	0	0	0	0	0	0.86	0	0	0	0	0	0	0.14	86%	1.4	1.4
T30	0	0	0	0	0	0	0	2E-03	0	0	0	0	0.85	0	0	0	4E-03	0	0.15	85%	1.5	1.5
T48	0	0	0	3E-03	0	0	0	0	0	0	0	0	0	0.90	0	0	0	0	0.10	90%	1.0	1.0
WF36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.89	0	0	0	0.11	89%	1.1	1.1
P2	0	1E-01	0	0	0	0	2E-03	0	0	1E-02	0	0	0	0	0	0.54	2E-03	0	0.31	54%	3.1	3.5
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.81	0	0.19	81%	1.9	1.9
Pamb	0	0	0	0	1E-02	0	0	4E-02	0	0	0	0	0	0	0	0	0	0.07	0.88	7%	6.8	7.7
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.686	0.314	2.5
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.79

Abrupt Fault Cases (large)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.71	0	0	0	0	0	0	2E-01	0	0	0	0	0	0	0	0	0	0	0.14	71%	1.4	2.1
LPC	0	0.55	0	0	0	0	7E-02	0	0	3E-02	0	0	0	0	0	1E-02	0	0	0.34	55%	3.4	4.2
HPC	0	0	0.91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.09	91%	0.9	0.9
HPT	0	0	0	0.98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	98%	0.2	0.2
LPT	0	0	0	0	0.90	0	0	0	0	0	0	0	0	0	0	0	0	3E-03	0.09	90%	0.9	1.0
VSV	0	0	0	0	0	0.95	0	0	3E-02	0	0	0	0	0	0	0	2E-03	0	0.02	95%	0.2	0.5
VBV	0	3E-02	0	0	0	0	0.53	3E-03	0	1E-02	0	0	0	0	0	2E-02	0	0	0.42	53%	4.2	4.3
Nf	0	0	0	0	0	0	0	0.99	0	0	0	0	0	0	0	0	0	0	0.01	99%	0.1	0.1
Nc	0	0	0	0	0	3E-01	0	1E-03	0.33	0	0	0	1E-01	0	0	0	4E-03	0	0.29	33%	2.7	3.3
P24	0	0	0	0	0	0	0	0	0	0.90	0	0	0	0	0	5E-03	0	0	0.10	90%	1.0	1.0
Ps30	0	0	0	0	0	0	0	2E-03	0	0	0.79	0	0	0	0	0	0	0	0.20	79%	2.0	2.1
T24	0	0	0	0	0	0	0	0	0	0	0	0.93	0	0	0	0	0	0	0.07	93%	0.7	0.7
T30	0	0	0	0	0	0	0	2E-03	0	0	0	0	0.92	0	0	0	0	0	0.08	92%	0.8	0.8
T48	6E-03	0	0	0	0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	0.01	98%	0.1	0.2
WF36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.97	0	0	0	0.03	97%	0.3	0.3
P2	0	1E-01	0	0	0	0	0	0	0	3E-03	0	0	0	0	0	0.75	0	0	0.15	75%	1.5	2.0
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.91	0	0.09	91%	0.9	0.9
Pamb	7E-03	0	0	0	0	0	0	4E-02	0	0	0	0	0	0	0	0	0	0.32	0.63	32%	5.7	5.7
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.850	0.150	1.5
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.88

Figure A.1.—Example abrupt fault metric results archived by ProDiMES

Rapid Fault Cases (all)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detect Latency	Classify Latency
Fan	0.31	0	0	0	5E-04	0	0	1E-01	0	0	0	0	0	0	0	5E-04	0	5E-04	0.59	31%	7.7	8.7
LPC	0	0.17	0	0	0	0	2E-02	0	0	8E-03	0	0	0	0	0	4E-03	0	0	0.80	17%	9.6	10.0
HPC	0	0	0.50	0	0	0	0	0	0	0	0	0	8E-04	0	0	0	8E-04	4E-04	0.50	50%	6.5	6.6
HPT	0	0	0	0.60	0	0	0	0	0	0	0	0	0	1E-03	0	0	0	0	0.40	60%	5.7	5.7
LPT	0	0	0	6E-04	0.46	0	0	1E-03	0	0	6E-04	0	0	0	0	0	0	1E-02	0.52	46%	7.0	7.2
VSV	0	0	0	0	0	0.69	0	0	8E-03	5E-04	0	0	0	0	0	0	5E-04	0	0.30	69%	4.4	4.5
VBV	9E-04	6E-03	0	0	0	0	0.12	4E-03	0	6E-03	0	5E-04	0	0	0	1E-02	0	0	0.85	12%	9.5	10.0
Nf	3E-03	0	0	0	0	0	0	0.60	0	0	0	0	0	0	0	0	0	5E-04	0.40	60%	5.0	5.1
Nc	4E-04	0	0	0	8E-04	2E-02	0	0	0.13	0	0	0	2E-02	0	0	0	3E-03	8E-04	0.83	13%	9.7	10.0
P24	0	3E-03	0	0	0	0	1E-03	0	0	0.44	0	0	0	0	0	5E-03	0	1E-03	0.55	44%	6.6	6.8
Ps30	4E-03	0	0	0	0	0	0	1E-03	0	0	0.29	0	0	0	0	0	0	5E-04	0.71	29%	8.6	8.7
T24	0	0	0	0	0	0	0	5E-04	0	0	0	0.52	0	0	0	0	4E-03	0	0.48	52%	5.9	6.0
T30	0	0	0	0	0	0	0	5E-04	0	0	0	0	0.48	0	0	0	0	1E-03	0.52	48%	6.6	6.6
T48	0	0	0	0	0	0	0	3E-03	0	0	0	0	0	0.61	0	1E-03	0	0	0.39	61%	5.4	5.4
WF36	9E-04	0	0	0	0	0	0	6E-03	0	0	0	0	0	0	0.56	2E-03	0	0	0.43	56%	5.7	5.8
P2	0	6E-02	0	0	0	0	4E-04	2E-03	0	7E-03	0	0	0	0	0	0.28	0	0	0.64	28%	8.0	8.2
T2	0	0	1E-03	0	0	0	0	0	0	0	0	1E-03	0	0	0	0	0.44	0	0.56	44%	6.8	6.8
Pamb	1E-03	0	0	0	0	0	0	2E-02	0	0	0	0	0	0	0	0	0	0.07	0.91	7%	10.4	10.6
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.419	0.581	6.8
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.56

Rapid Fault Cases (small)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detect Latency	Classify Latency
Fan	0.11	0	0	0	0	0	0	5E-02	0	0	0	0	0	0	0	0	0	0	0.83	11%	10.5	10.7
LPC	0	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0	6E-03	0	0	0.99	1%	10.0	10.0
HPC	0	0	0.22	0	0	0	0	0	0	0	0	0	1E-03	0	0	0	1E-03	0	0.77	22%	9.6	9.7
HPT	0	0	0	0.44	0	0	0	0	0	0	0	0	0	3E-03	0	0	0	0	0.56	44%	7.7	7.8
LPT	0	0	0	2E-03	0.23	0	0	3E-03	0	0	2E-03	0	0	0	0	0	0	2E-02	0.75	23%	9.9	10.3
VSV	0	0	0	0	0	0.55	0	0	8E-03	0	0	0	0	0	0	0	2E-03	0	0.44	55%	6.6	6.6
VBV	0	0	0	0	0	0	0.00	0	0	0	0	0	0	0	0	0	0	0	1.00	0%		
Nf	0	0	0	0	0	0	0	0.35	0	0	0	0	0	0	0	0	0	0	0.65	35%	7.5	7.5
Nc	0	0	0	0	0	4E-03	0	0	0.00	0	0	0	0	0	0	0	0	0	0.99	0%	11.5	11.5
P24	0	2E-03	0	0	0	0	2E-03	0	0	0.10	0	0	0	0	0	3E-03	0	0	0.90	10%	10.5	10.7
Ps30	0	0	0	0	0	0	0	0	0	0	0.03	0	0	0	0	0	0	0	0.97	3%	12.1	12.1
T24	0	0	0	0	0	0	0	0	0	0	0	0.24	0	0	0	0	8E-03	0	0.76	24%	9.2	9.3
T30	0	0	0	0	0	0	0	1E-03	0	0	0	0	0.22	0	0	0	0	1E-03	0.78	22%	9.7	9.8
T48	0	0	0	0	0	0	0	2E-03	0	0	0	0	0	0.36	0	0	0	0	0.63	36%	8.3	8.3
WF36	0	0	0	0	0	0	0	1E-02	0	0	0	0	0	0	0.32	5E-03	0	0	0.66	32%	8.3	8.5
P2	0	1E-02	0	0	0	0	0	6E-03	0	2E-03	0	0	0	0	0	0.06	0	0	0.92	6%	11.0	10.9
T2	0	0	4E-03	0	0	0	0	0	0	0	0	1E-03	0	0	0	0	0.17	0	0.83	17%	9.9	9.9
Pamb	0	0	0	0	0	0	0	2E-03	0	0	0	0	0	0	0	0	0	0.00	1.00	0%	13.0	
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.193	0.807	9.0
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.31

Rapid Fault Cases (medium)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detect Latency	Classify Latency
Fan	0.37	0	0	0	1E-03	0	0	1E-01	0	0	0	0	0	0	0	0	0	0	0.51	37%	7.7	8.9
LPC	0	0.13	0	0	0	0	3E-02	0	0	1E-02	0	0	0	0	0	4E-03	0	0	0.83	13%	11.0	11.4
HPC	0	0	0.55	0	0	0	0	0	0	0	0	0	0	0	0	0	1E-03	0	0.45	55%	6.8	6.8
HPT	0	0	0	0.66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.34	66%	5.1	5.1
LPT	0	0	0	0	0.55	0	0	0	0	0	0	0	0	0	0	0	0	2E-02	0.43	55%	6.5	6.8
VSV	0	0	0	0	0	0.73	0	0	1E-02	2E-03	0	0	0	0	0	0	0	0	0.25	73%	3.8	4.0
VBV	1E-03	1E-02	0	0	0	0	0.07	4E-03	0	1E-03	0	1E-03	0	0	0	1E-02	0	0	0.89	7%	10.5	11.1
Nf	8E-03	0	0	0	0	0	0	0.66	0	0	0	0	0	0	0	0	0	1E-03	0.33	66%	4.9	5.0
Nc	0	0	0	0	0	1E-02	0	0	0.12	0	0	0	3E-03	0	0	0	1E-03	0	0.86	12%	10.8	11.1
P24	0	6E-03	0	0	0	0	1E-03	0	0	0.51	0	0	0	0	0	8E-03	0	1E-03	0.47	51%	7.1	7.3
Ps30	1E-02	0	0	0	0	0	0	3E-03	0	0	0.34	0	0	0	0	0	0	2E-03	0.64	34%	9.5	9.6
T24	0	0	0	0	0	0	0	0	0	0	0	0.62	0	0	0	0	2E-03	0	0.38	62%	5.7	5.8
T30	0	0	0	0	0	0	0	0	0	0	0	0	0.57	0	0	0	0	0	0.43	57%	6.4	6.4
T48	0	0	0	0	0	0	0	6E-03	0	0	0	0	0	0.65	0	2E-03	0	0	0.34	65%	5.1	5.2
WF36	0	0	0	0	0	0	0	3E-03	0	0	0	0	0	0	0.64	0	0	0	0.35	64%	5.3	5.3
P2	0	8E-02	0	0	0	0	0	0	0	2E-02	0	0	0	0	0	0.32	0	0	0.59	32%	8.9	8.9
T2	0	0	0	0	0	0	0	0	0	0	0	2E-03	0	0	0	0	0.55	0	0.45	55%	6.7	6.7
Pamb	0	0	0	0	0	0	0	8E-03	0	0	0	0	0	0	0	0	0	0.03	0.97	3%	11.3	11.6
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.458	0.542	7.0
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.61

Rapid Fault Cases (large)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detect Latency	Classify Latency
Fan	0.46	0	0	0	0	0	0	1E-01	0	0	0	0	0	0	0	2E-03	0	2E-03	0.40	46%	6.0	7.4
LPC	0	0.36	0	0	0	0	3E-02	0	0	1E-02	0	0	0	0	0	2E-03	0	0	0.59	36%	8.8	9.2
HPC	0	0	0.68	0	0	0	0	0	0	0	0	0	1E-03	0	0	0	0	1E-03	0.32	68%	4.7	4.8
HPT	0	0	0	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	75%	3.8	3.8
LPT	0	0	0	0	0.66	0	0	0	0	0	0	0	0	0	0	0	0	9E-03	0.33	66%	4.9	5.0
VSV	0	0	0	0	0	0.80	0	0	3E-03	0	0	0	0	0	0	0	0	0	0.19	80%	2.9	3.0
VBV	1E-03	8E-03	0	0	0	0	0.28	8E-03	0	1E-02	0	0	0	0	0	2E-02	0	0	0.67	28%	9.1	9.6
Nf	2E-03	0	0	0	0	0	0	0.77	0	0	0	0	0	0	0	0	0	0	0.23	77%	3.4	3.4
Nc	1E-03	0	0	0	3E-03	5E-02	0	0	0.27	0	0	0	7E-02	0	0	0	9E-03	3E-03	0.61	27%	8.9	9.0
P24	0	2E-03	0	0	0	0	1E-03	0	0	0.65	0	0	0	0	0	5E-03	0	2E-03	0.34	65%	5.1	5.2
Ps30	2E-03	0	0	0	0	0	0	0	0	0	0.54	0	0	0	0	0	0	0	0.46	54%	6.9	6.9
T24	0	0	0	0	0	0	0	1E-03	0	0	0	0.72	0	0	0	0	2E-03	0	0.28	72%	4.2	4.2
T30	0	0	0	0	0	0	0	0	0	0	0	0	0.70	0	0	0	0	2E-03	0.30	70%	4.5	4.5
T48	0	0	0	0	0	0	0	1E-03	0	0	0	0	0	0.74	0	1E-03	0	0	0.26	74%	3.8	3.9
WF36	3E-03	0	0	0	0	0	0	1E-03	0	0	0	0	0	0	0.74	0	0	0	0.26	74%	3.9	3.9
P2	0	1E-01	0	0	0	0	1E-03	0	0	2E-03	0	0	0	0	0	0.48	0	0	0.41	48%	6.1	6.8
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.68	0	0.32	68%	4.7	4.7
Pamb	3E-03	0	0	0	0	0	0	4E-02	0	0	0	0	0	0	0	0	0	0.18	0.78	18%	10.1	10.4
No Fault	8E-06	0	0	0	0	8E-06	0	0	0	0	0	0	0	0	0	4E-06	0	8E-06	0.99997	99.997%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.610	0.390	5.6
No Fault	2.7E-05	0.99997	N/A

Kappa Coefficient
0.7

Figure A.2.—Example rapid fault metric results archived by ProDiMES

Appendix B.—Example Diagnostic Solution

ProDiMES provides a process for developing and benchmarking candidate gas path diagnostic methodologies. An example solution has also been included to demonstrate how a user defined diagnostic solution would integrate into the process. This Matlab program, `ExampleSolution.m`, is located in the `\ExampleSolution` directory. ***This example should not be interpreted as the recommended or required approach for solving the problem***. It is simply provided as an example solution to illustrate the overall diagnostic process and to serve as a template for the development of additional diagnostic solutions. The following subsections provide a functional description of the example solution, followed by a description of the example solution operating instructions.

B.1 Example Diagnostic Solution Description

The example diagnostic solution is partitioned into a four-step process shown in Figure B.1 consisting of: (1) parameter correction; (2) trend monitoring; (3) anomaly detection; and (4) event isolation. This process is applied to each engine in an EFS generated data set. Each of these steps is further discussed below.

B.1.1 Step 1—Parameter Correction

As an initial step within the example solution, all engine parameters are corrected to standard day operating conditions. This is performed to reduce scatter in the engine outputs in preparation for subsequent analysis (Ref. 5). Parameter correction is performed by applying the following parameter correction equation

$$\text{Parameter}_{\text{corrected}} = \frac{\text{Parameter}}{\theta^{a}\delta^{b}} \tag{B1}$$

where the temperature and pressure correction terms, θ and δ respectively, are defined as

$$\begin{aligned} \theta &= \frac{T2\left(^{\circ}\text{R}\right)}{518.67} \\ \delta &= \frac{P2\left(\text{psia}\right)}{14.696} \end{aligned} \tag{B2}$$

Parameter correction exponents and correction equations for individual parameters are shown in Table B1.

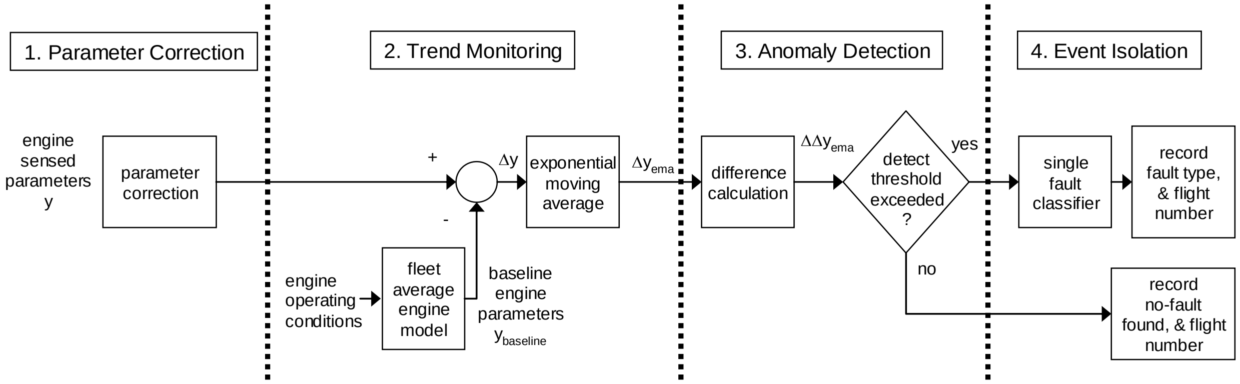

1. Parameter Correction
2. Trend Monitoring
3. Anomaly Detection
4. Event Isolation
engine sensed parameters y
parameter correction
+
-
Δy
exponential moving average
Δy_{ema}
engine operating conditions
fleet average engine model
baseline engine parameters $y_{baseline}$
difference calculation
$\Delta\Delta y_{ema}$
detect threshold exceeded ?
yes
no
single fault classifier
record fault type, & flight number
record no-fault found, & flight number

Figure B.1.—Example diagnostic solution process

TABLE B.1.—EFS OUTPUT PARAMETER CORRECTION EXPONENTS

Parameter	a	b	Corrected parameter
Nf	0.5	0	$NfR = \frac{Nf}{\sqrt{\theta}}$
Nc	0.5	0	$NcR = \frac{Nc}{\sqrt{\theta}}$
P24	0	1	$P24R = \frac{P24}{\delta}$
Ps30	0	1	$Ps30R = \frac{Ps30}{\delta}$
T24	1	0	$T24R = \frac{T24}{\theta}$
T30	0.94	0	$T30R = \frac{T30}{\theta^{0.94}}$
T48	1	0	$T48R = \frac{T48}{\theta}$
Wf	0.63	1	$WfR = \frac{Wf}{\theta^{0.63}\delta}$

B.1.2 Step 2—Trend Monitoring

ProDiMES has been constructed assuming that engine performance changes can manifest themselves in two ways: a) gradual (long-term) deterioration, or b) abrupt or rapid (short-term) faults. The former is due to *all* of the engine modules deteriorating slowly over time and is included in an attempt to emulate physical causes such as erosion, corrosion, fouling, and increased clearances within the turbomachinery. All engines will naturally undergo gradual performance deterioration over their lifetime of use, and this should not be diagnosed as a fault. Conversely, an abrupt or rapid performance change is due to a *single* fault occurring within the engine and thus should be diagnosed as a fault. An effective gas path diagnostic solution must be able to function with both of these processes occurring and interacting simultaneously, and must be able to discriminate between the two cases, without corrupting the overall diagnostic approach.

In the example solution presented herein, a trend monitoring approach is applied to capture gradual performance changes in the form of residuals, or measurement deltas, relative to a fleet average engine. Although the inclusion of trend monitoring functionality within a user diagnostic solution is not required, it is expected to improve overall diagnostic performance, and is thus included as part of this example. The trend monitoring approach implemented within the example solution utilizes a three-dimensional table lookup model representative of a fleet-average, or 50 percent deteriorated engine. This model was generated by running the C-MAPSS Steady-State engine model over a range of altitude, Mach number, and corrected fan speed settings while setting module health parameters to represent a 50 percent deteriorated engine (based on the deterioration profiles previously shown in Figure 6). Engine output parameters produced at each operating point were then used to create the lookup model. (Note: The Matlab program used to generate the fleet average table lookup model, `CMAPSS_FA_TLM.m`, can be found in the `\ExampleSolution` directory. End users can edit this file to generate table lookup models at other operating points or deterioration levels if they so desire. The table lookup model is stored in the file `CMAPSS_FA_TLM.mat`, also located in the `\ExampleSolution` directory). For an individual engine corrected data collected during each flight are then referenced against the fleet average engine model to calculate measurement deltas, Δy_i's, as:

$$\Delta y_i(k) = y_i(k) - y_{i\_\text{baseline}}(k) \qquad \text{(B3)}$$

where $y_i(k)$ is the corrected value of the i^{th} measurement collected during the k^{th} flight, and $y_{i\_baseline}(k)$ is the fleet average engine value for the i^{th} measurement at the corresponding pressure altitude, Mach number, and corrected fan speed values of the k^{th} flight. (Note: pressure altitude is the altitude corresponding to Pamb as defined within Standard Atmosphere tables. Pressure altitude and Mach number can be calculated from Pamb and P2 as shown in the example solution source code). A $y_i(k)$ value is only calculated for seven of the 11 measurements available. The parameters Nf, Pamb, P2, and T2 are used for establishing the engine operating point and parameter correction purposes and are thus excluded from the $y_i(k)$ calculations. The calculated measurement delta values are trended over time by applying an exponential moving average approach as described in (Ref. 6) and given as

$$\Delta y_{i\_ema}(k) = \alpha \cdot \Delta y_{i\_ema}(k-1) + (1-\alpha) \cdot \Delta y_i(k) \tag{B4}$$

where $\Delta y_{i\_ema}(k)$ is the exponential moving average of the i^{th} measurement delta on flight k. The moving average weighting between previous and current data is established by the constant α (where $0 < \alpha < 1$). In the example solution α was chosen to be 0.8844.

B.1.3 Step 3—Anomaly Detection

The previous subsection presented an example approach for monitoring gradual engine performance measurement deltas over time. Next, anomaly detection logic is applied to detect discrete events causing a rapid shift in observed measurement deltas and to extract the corresponding measurement delta signature of those events. It is important to recognize that it would be prudent to incorporate a time latency inherent in the detection process to avoid false alarms caused by statistical outliers in the measurement data. The approach applied herein for anomaly detection uses a backwards difference calculation of the exponential moving average (EMA) of each measurement delta shown as

$$\Delta\Delta y_{i\_ema}(k) = \Delta y_{i\_ema}(k) - \Delta y_{i\_ema}(k-\beta) \tag{B5}$$

where $\Delta\Delta y_{i\_ema}(k)$, or the measurement delta-delta, is the change in the EMA of the i^{th} measurement delta between flight k and some previous flight, $k - \beta$. Choosing a $\beta = 10$ flight cycle distance between the compared EMA values was found to provide detection capability for both abrupt as well as rapid faults. It is anticipated that improved detection robustness for abrupt as well as rapid faults could be obtained by combining multiple detection filters that apply different β distances within the measurement delta-delta calculations. Anomaly detection logic is applied that monitors for $\Delta\Delta y_{i\_ema}(k)$ exceeding a specified threshold. Based upon trial an error, anomaly detection thresholds of $\pm 1.31\ \sigma$ were applied to each measurement delta-delta EMA. (Note: Here, σ refers to analytically calculated measurement delta standard deviation values. The Matlab program used to analytically calculate these σ values, `delta_y_covariance.m`, can be found in the `\ExampleSolution` directory). If an anomaly is detected, the example diagnostic solution logic proceeds to event isolation, which is discussed below.

B.1.4 Step 4—Event Isolation

The final step in the diagnostic process is event isolation, or classifying the root cause for any detected anomalies. Numerous approaches towards aircraft gas turbine engine fault classification can be found in the literature. The example diagnostic solution distributed with the ProDiMES software applies a backwards difference calculation to extract the measurement signature of the detected anomaly, and then assigns a fault classification through a simple random guess. The backwards difference calculation produces residuals between the EMA measurement deltas collected on subsequent flights and the EMA measurement deltas on the flight of initial anomaly detection, k_{anomaly}. This difference calculation is computed as

$$\Delta\Delta y_{i\_ema\_\text{anomaly}}(k) = \Delta y_{i\_ema}(k) - \Delta y_{i\_ema}(k_{\text{anomaly}}) \tag{B6}$$

The measurement delta-deltas, $\Delta\Delta y_{i\_ema\_\text{anomaly}}(k)$, produced by (B6) can be directly used by many fault classification approaches. Through Equation (B6) an anomaly signature vector in the measurement delta-delta space due to the underlying fault event can be obtained as:

$$\Delta\Delta Y(k) = \begin{bmatrix} \Delta\Delta y_{1\_ema\_\text{anomaly}}(k) \\ \Delta\Delta y_{2\_ema\_\text{anomaly}}(k) \\ \vdots \\ \Delta\Delta y_{m\_ema\_\text{anomaly}}(k) \end{bmatrix} \tag{B7}$$

where m is the number of measurement deltas. Once the anomaly signature vector, $\Delta\Delta Y(k)$, is obtained, the classification problem becomes one of selecting the fault type most likely to be the cause of the observed anomaly vector. This is often performed through least squares estimation techniques and the application of a fault influence coefficient matrix that relates engine fault types to observed changes in engine outputs. The ($m \times n$) fault influence matrix is denoted as H, where m = number of measurements, and n = number of single fault types. Assuming $x(k)$ is an n×1 vector representing the magnitudes of the n single fault types under consideration, the interrelationship between faults to measurement delta-delta changes can be written as:

$$\Delta\Delta Y(k) = Hx(k) \tag{B8}$$

Generally speaking the equation shown in (B8) presents an underdetermined estimation problem as there are more unknowns (i.e., fault types) than available measurements. However, applying the single fault assumption, the problem becomes tractable and reduces to one of choosing the fault type most likely for producing the observed signature. A fault influence coefficient matrix can be generated by running the C-MAPSS Steady-State engine model to a fixed closed-loop operating condition (specified by altitude, Mach, ΔTamb, corrected Nf, and deterioration level) and individually introducing each of the 18 fault types. The elements of the matrix consist of the partial derivatives relating the change in corrected measured engine outputs (fault condition vs. nominal condition) to the magnitude of the implanted fault. Users are referred to the Matlab routine `CMAPSS_IC_generation.m`, located in the `\ExampleSolution` directory, which contains code for generating fault influence coefficient matrices at takeoff and cruise operating points for a 50 percent deteriorated engine.

As previously mentioned, the example diagnostic solution applies a random guess to classify, or isolate, the fault type. As such it is expected to exhibit relatively poor overall classification performance. However, this routine does provide a template to illustrate how diagnostic assessments are to be generated for each engine each flight, and then archived within the `diagnostic_assessment` matrix.

B.2 Example Diagnostic Solution Operating Instructions

In order to run the example solution, users must navigate to the `\ExampleSolution` directory, and type the command `ExampleSolution`. Upon execution, the example solution routine will load a previously generated EFS parameter history stored in the file named `EFS_Output.mat`. ProDiMES comes with a pregenerated `EFS_Output.mat` file stored in the `\ExampleSolution` directory. If users wish to evaluate different EFS output files with the example solution they will be required to copy and paste a different `EFS_Output.mat` file into the `\ExampleSolution` directory.

Upon completion, the example solution archives a diagnostic assessment produced for each engine, each flight, to the file `DiagnosticAssessments.mat`. This diagnostic assessment is stored in the standard format as previously described in Section 3.0 of this guide.

B.3 Example Diagnostic Solution—Software Parameter List

The software parameters used within `ExampleSolution.m` are listed in the table below.

TABLE B.2.—EXAMPLESOLUTION.M PARAMETER LIST

	Parameter name	Description
Inputs	efs_output_c, efs_output_to	EFS generated sensed parameter histories loaded from `EFS_Output.mat`.
	FA_TLM_grid4_c, FA_TLM_grid4_to, Mach_grid4_c, Mach_grid4_to, NfR_grid4_c, NfR_grid4_to, alt_grid4_c, alt_grid4_to, sensor_grid4_c, sensor_grid4_to	Fleet average table lookup model matrices loaded from the file `CMAPSS_FA_TLM.mat`. (Note: the program used to generate the fleet average engine model is `CMAPSS_FA_TLM.m`).
	Fn_zro, MN_zro, Nc_zro, Nf_zro, P2_zro, P24_zro, Ps30_zro, Pamb_zro, T2_zro, T24_zro, T30_zro, T48_zro, Wf_zro, X_zro, alt_zro, dTamb_zro	Fleet average engine parameter trim values used for approximating average cruise and takeoff sensor measurement noise. These parameters are loaded from the files `\JacobiansTrim\FC_cruise.mat` (cruise) and `\JacobiansTrim\FC_takeoff.mat` (takeoff).
	H_c, H_to	Fault influence coefficient matrices loaded from the file `CMAPSS_IC_matrix.mat`. (Note: See `CMAPSS_IC_generation.m`)
	R_c, R_to	Delta measurement covariance matrices loaded from `delta_y_covariance.mat`. (Note: The R_c and R_to matrices are generated by the routine `delta_y_covariance.m`).
Outputs	diagnostic_assessment	Matrix of diagnostic assessments for each engine, each flight stored to `DiagnosticAssessments.mat`.
Internal parameters	R	Delta measurement covariance matrix (takeoff and cruise concatenated).
	Mach_interp_c, Mach_interp_to, NfR_interp_c, NfR_interp_to, alt_interp_c, alt_interp_to, sensors_interp_c, sensors_interp_to	Input matrices to fleet average table lookup model.
	H	Fault influence coefficient matrix.
	Mach_hat_c, Mach_hat_to	Estimated Mach number.
	Nc_sigma, Nf_sigma, P2_sigma, P24_sigma, Pamb_sigma, Ps30_sigma, T2_sigma, T24_sigma, T30_sigma, T48_sigma, Wf_sigma	Sensor measurement noise standard deviations.
	NormFactor_c, NormFactor_to	Measurement normalization vectors.
	alt_hat_c, alt_hat_to	Estimated pressure altitudes.
	anomaly_flag	Flag set if an engine anomaly is detected.
	delta, theta	pressure and temperature correction factors.
	delta_y_corrected_c, delta_y_corrected_to	Corrected delta measurements.
	ndyc	Normalized corrected delta y measurements (takeoff and cruise).
	ndyc_ema10	Exponential moving avg. of normalized corrected delta measurements.
	ndyc_ema10_old	Exponential moving average of normalized corrected delta measurements, preanomaly detect values.
	ndyc_gradient	Exponential moving average normalized corrected delta-delta measurements (referenced against values 10 flight prior).
	ndyc_gradientanomaly	Exponential moving average normalized corrected delta-delta measurements (referenced against prefault values).
	normalized_delta_y_corrected_c, normalized_delta_y_corrected_to	Normalized corrected delta measurements (entire fleet).
	num_engines	Number of engines in the fleet.
	num_flight_cycles	Number of flight cycles each engine experiences.
	sigma_c, sigma_to, sigma_norm_c, sigma_norm_to	Measurement noise standard deviation vectors.
	y_corrected_c, y_corrected_to	Sensed parameter history corrected measurements (cruise and takeoff).

References

1. Jaw, L.C., (2005), “Recent Advances in Aircraft Engine Health Management (EHM) Technologies and Recommendations for the Next Step,” ASME Paper GT2005-68625.
2. Simon, D.L, Bird, J., Davison, C., Volponi, A., Iverson, R.E., (2008), “Benchmarking Gas Path Diagnostic Methods: A Public Approach,” NASA/TM-2008-215271, ASME GT2008-51360, ASME Turbo Expo 2008, Berlin, Germany.
3. Frederick, D.K., DeCastro, J.A., Litt, J.S., (2007), “User’s Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS),” NASA Technical Memorandum TM-2007-215026.
4. Sallee, G.P., (1978) “Performance Deterioration Based on Existing (Historical) Data – JT9D Jet Engine Diagnostics Program,” NASA Contractor Report CR-135448, United Technologies Corporation, Pratt & Whitney Aircraft Group Report PWA-5512-21.
5. Volponi, A.J., (1999), “Gas Turbine Parameter Corrections,” Journal of Engineering for Gas Turbines and Power, Vol. 121, pp. 613–621.
6. DePold, H.R., Gass, F.D., (1999), “The Application of Expert Systems and Neural Networks to Gas Turbine Prognostics and Diagnostics,” Journal of Engineering for Gas Turbines and Power, Vol. 121, pp. 607–612.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-01-2010	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Simon, Donald, L.	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER WBS 645846.02.07.03.03.01
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-17100
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-2010-215840

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category: 07
Available electronically at http://gltrs.grc.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 User's Guide for the Propulsion Diagnostic Method Evaluation Strategy (ProDiMES). ProDiMES is a standard benchmarking problem and a set of evaluation metrics to enable the comparison of candidate aircraft engine gas path diagnostic methods. This Matlab (The Mathworks, Inc.) based software tool enables users to independently develop and evaluate diagnostic methods. Additionally, a set of blind test case data is also distributed as part of the software. This will enable the side-by-side comparison of diagnostic approaches developed by multiple users. The Users' Guide describes the various components of ProDiMES, and provides instructions for the installation and operation of the tool.

15. SUBJECT TERMS
Aircraft engines; Systems health monitoring; Gas turbine engines; Flight safety

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** STI Help Desk (email:help@sti.nasa.gov)
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	44	**19b. TELEPHONE NUMBER** ***(include area code)*** 443-757-5802