



Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking

*Al Volponi
Pratt & Whitney, East Hartford, Connecticut*

NASA STI Program . . . in Profile

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

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

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONFERENCE PUBLICATION.** Collected

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

- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

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

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

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at 301-621-0134
- Telephone the NASA STI Help Desk at 301-621-0390
- Write to:
NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320



Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking

Al Volponi
Pratt & Whitney, East Hartford, Connecticut

Prepared under Contract NAS3-01138, Task order 26

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

Acknowledgments

NASA Glenn Research Center and Pratt & Whitney are gratefully acknowledged for their funding support for this work. In particular we wish to thank the COTR, Don Simon, NASA Glenn for his continued interest, support, and program flexibility over the years as this technology evolved. Of special note are the numerous contributions made by Tom Brotherton and Rob Luppold of Intelligent Automation Corporation (IAC) in defining and maturing the hybrid model eSTORM methodology and implementation described in this report.

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 NASA 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>

Table of Contents

ABSTRACT	1
1. INTRODUCTION	2
1.1 TEAMING ARRANGEMENT	2
1.2 CHRONOLOGY	3
1.3 PROGRAM GOALS	3
1.4 BACKGROUND: STORM OVERVIEW	3
1.5 BACKGROUND: ESTORM OVERVIEW	7
2. CURRENT PROGRAM ACTIVITIES	11
2.1 INITIALIZING BASIC ESTORM CONFIGURATION	11
2.1.1 PW6000 STORM System	11
2.2 MODEL REFINEMENTS	14
2.2.1 Temperature Thermocouple Modeling	14
2.3 EMPIRICAL ELEMENT MODEL BUILDING	16
2.3.1 Empirical Model Building Overview	19
2.3.2 GMM Clustering Process	23
2.3.3 MLP Training Using Clusters	27
2.4 APPLICATION EXAMPLE	27
2.5 REAL-TIME CODE DEVELOPMENT	37
3. SUMMARY	41
APPENDIX	43
REFERENCES	49

List of Figures

1-1 STORM Architecture	4
1-2 Simulated SVM/Engine Mismatch: Without KF	5
1-3 Effect on STORM Tuners of SVM/Engine Mismatch	6
1-4 Simulated SVM/Engine Mismatch: With KF	6
1-5 Constructing the Empirical Element of the Hybrid Model	7
1-6 Implementing the Empirical Model Element to Form the Hybrid Model	8
1-7 Simulated SVM/Engine Mismatch: With Hybrid Model eSTORM	9
1-8 Effect on Tuners of SVM/Engine Mismatch Using eSTORM	9
1-9 Increased Performance Deviation Visibility Using eSTORM Hybrid Model	10
2-1 Performance Tuners During High Power Steady State Operation	13
2-2 PW6000 Block 4 Flight Test Take-Off Run	14
2-3 PW6000 Block 4 Flight Test Take-Off Run	15
2-4 PW6000 Block 4 Flight Test Take-Off Run	15
2-5 Parameter Residual Empirical Model MLP Structure	17
2-6 Partitioning the Flight Envelope to Create Sub-Models	18
2-7 Two-Stage Empirical Modeling Approach	20
2-8 MLP Training Overview	21
2-9 eSTORM Overview Process Architecture	22
2-10 GMM Cluster Process	25
2-11 Altitude Profile for Example Data	28
2-12 Segment 1 Input Parameters	28
2-13 Segment 1 Measurement Residual Parameters (Percent Δ) Without KF	29
2-14 Segment 1 Measurement Residual Parameters (Percent Δ) With KF	30
2-15 Segment 1: Effect on STORM Tuners of SVM/Engine Mismatch	30
2-16 Segment 1: GMM Centers	31
2-17 Segment 1 Measurement Residuals (Percent Δ) With Hybrid Model Trained With 23 GMMs	32
2-18 Segment 1 STORM Tuners Using Hybrid Model Trained With 23 GMMs	32
2-19 Segment 2 Input Parameters	33
2-20 Segment 2 Measurement Residual Parameters (Percent Δ) Without KF	34

2-21	Segment 2 Measurement Residual Parameters (Percent Δ) Without KF and Segment 1 MLP Enabled	34
2-22	Segment 2 STORM tuners Using Hybrid Model Trained With 23 GMMS	35
2-23	Segment 1 and 2: GMM Centers	36
2-24	Segment 2 Measurement Residuals (Percent Δ) With Hybrid Model Trained With 59 GMMS	36
2-25	Segment 2 STORM Tuners Using Hybrid Model Trained With 59 GMMs	37
2-26	Hardware System Summary	38
2-27	Development Station	38
2-28	Operating System Architecture	39
2-29	Simulink to Embedded System Development	39
2-30	Demonstration Software to Hardware Mapping	40
A-1	PW6000 Block 4: Measurement Comparison—N1	43
A-2	PW6000 Block 4: Measurement Comparison—N2	43
A-3	PW6000 Block 4: Measurement Comparison—P25	44
A-4	PW6000 Block 4: Measurement Comparison—Pb	44
A-5	PW6000 Block 4: Measurement Comparison—P5	45
A-6	PW6000 Block 4: Measurement Comparison—T25	45
A-7	PW6000 Block 4: Measurement Comparison—T3	46
A-8	PW6000 Block 4: Measurement Comparison—T5	46

List of Tables

1-1	Participating RASER Task Order No. 26 Companies	2
1-2	Major Responsibilities Per Company	2
2-1	Error Between SVM and Current PW6000 Engine	13
2-2	Error Between SVM and Current PW6000 Engine With Tuners Enabled	14

Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking

Al Volponi
Pratt & Whitney
East Hartford, Connecticut 06108

ABSTRACT

A key technological concept for producing reliable engine diagnostics and prognostics exploits the benefits of fusing sensor data, information, and/or processing algorithms. This report describes the development of a hybrid engine model for a propulsion gas turbine engine which is the result of fusing two diverse modeling methodologies; a physics-based model approach and an empirical model approach. The report describes the process and methods involved in deriving and implementing a hybrid model configuration for a commercial turbofan engine. Among the intended uses for such a model is to enable real-time, on-board tracking of engine module performance changes and engine parameter synthesis for fault detection and accommodation.

1. INTRODUCTION

A practical consideration for implementing a real-time on-board engine component performance tracking system is the development of high-fidelity engine models capable of providing a reference level from which performance changes can be trended. Real-time engine models made their advent as state variable models (SVM) in the mid-80s and used a piecewise linear model that granted a reasonable representation of the engine during steady-state operation and mild transients. Increased processor speeds over the next decade allowed more complex models, which were a combination of linear and non-linear physics-based elements, to be considered. While the latter provided greater fidelity over transient operation and the engine operational flight envelope, these models could be improved to provide the high level of accuracy required for long-term performance tracking, as well as address the issue of engine-to-engine variation. Over time, these models may deviate enough from the actual engine being monitored, as a result of improvements made during an engine’s life cycle such as hardware modifications, bleed and stator vane schedules alterations, cooling flow adjustments, which cause the module performance estimations to be inaccurate and often misleading. Thus, the challenge is to find a modeling approach that will address all of these shortcomings, while maintaining the execution speed required for real-time implementation.

To address this challenge, an alternate approach to engine modeling has been introduced; wherein, a hybrid engine model architecture incorporates physics-based and empirical components. This methodology provides a means to automatically tune the engine model to a particular configuration as the engine evolves over the course of its life, and, furthermore, aligns the model to the particular engine being monitored to insure accurate performance tracking, while not compromising real-time operation.

Overall benefits that can be derived from hybrid models include reduction of health management system false alarms and missed detections, improvement of engine diagnostics for the accurate isolation of faults, as well as increased engine prognostic capabilities. These enhancements would directly support NASA and industry aeronautic strategic goals of reduced operating cost, increased safety, and increased reliability.

1.1 TEAMING ARRANGEMENT

The vast majority of the work contained in this report was accomplished by members of three companies (*Table 1-1*) with support from Glenn Research Center (GRC).

Table 1-1. Participating RASER Task Order No. 26 Companies

<i>Company/Agency</i>	<i>Role</i>	<i>Principal Investigator</i>
Pratt & Whitney (P&W)	Prime Contractor	Al Volponi
Intelligent Automation Corporation (IAC)	Sub-Contractor	Tom Brotherton
		Rob Luppold
		Jesse Ma
NASA GRC	Program Support	Don Simon (COTR)

Although there was considerable synergy and technical contribution overlap between the primary members of the team in all areas during this development, the major responsibilities are illustrated in *Table 1-2*:

Table 1-2. Major Responsibilities Per Company

<i>Company</i>	<i>Responsibility</i>
P&W	Program Management, Technical Direction, Analytical Support, Algorithm Development
IAC	Empirical Modeling, Algorithm Development, System Integration, Real-Time Software Development

1.2 CHRONOLOGY

This program was a follow-on development effort to develop, refine, and demonstrate a real-time hybrid engine model building strategy to enable accurate engine module performance tracking. The original exploration and development of the hybrid engine model approach were initiated under previous programs and funded by NASA GRC and NASA Dryden Flight Research Center (DFRC) as Small Business Innovative Research (SBIR) grants. These (Phase I and Phase II) grants were awarded to IAC and the development (which took place from 2001 through 2004) was carried out under close collaboration with P&W. These earlier programs provided a proof of concept for using a hybrid model approach with simulated engine data for a P&W F117 military transport (C17) engine. Real-time operation of an F117 hybrid engine model (developed off-line) was demonstrated on a target 300 MHz processor using streaming (20 Hz) simulated Aeronautical Radio, Incorporated (ARINC) 429 F117 flight data and was the culmination of the SBIR efforts.

These efforts were based on a P&W real-time engine model strategy referred to as the self-tuning onboard real-time model (STORM). The enhancements developed under these SBIRs (as well as under the current program) are referred to as enhanced STORM (eSTORM).

1.3 PROGRAM GOALS

The present program was initiated by NASA GRC as a follow-on effort with the end goal of refining the hybrid engine model building methodology to be used entirely in an on-board configuration and demonstrating the real-time software on the target processor driven by actual streaming flight data. This included both model building across the flight regime (previously done off-line on a PC) and operation of the model in real-time using actual engine flight data.

To accomplish these goals many sub-tasks had to be addressed. First, a suitable target engine for the study needed to be identified having the following attributes:

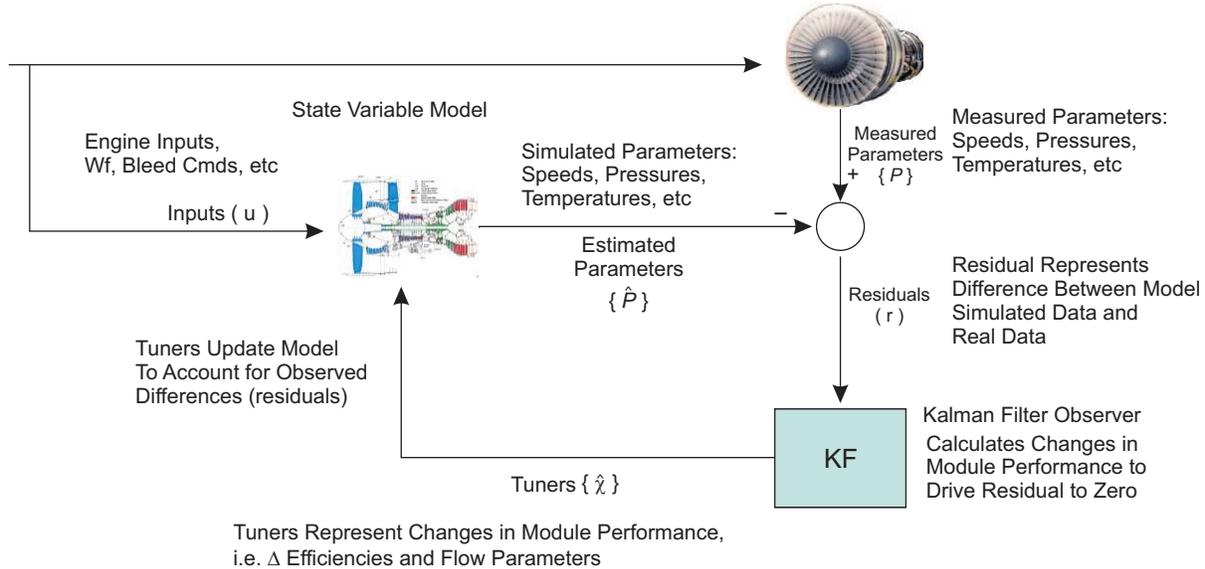
- Be a high-bypass engine used for transport applications
- Have sufficient flight and/or test cell data available to P&W to support initial eSTORM creation and development
- Have an operational STORM model that already exists, which can serve as the primary physics-based portion of the eSTORM hybrid model.

The P&W commercial PW6000 engine was chosen as the application engine for this development. It is a high-bypass engine that had recently (September through October 2004) completed flight certification and there was a large database of flight-recorded data available. It also had an on-board operational STORM system.

Before discussing the specific tasks and achievements of the current program, a brief overview of the STORM and eSTORM methodologies is provided. Further detail can be found in references 1 through 5, (see Section 4, *References*).

1.4 BACKGROUND: STORM OVERVIEW

The first attempts at constructing on-board engine models used a simple real-time engine model, typically a SVM coupled with a Kalman filter (KF) observer that together provided an adaptable engine model capable of tracking the monitored engine during steady state and transient operation. The KF observer would act upon the residuals formed by the output of the SVM and the actual observed measurements to provide a set of tuners that would adapt the SVM to match the actual observations (driving the residuals to zero, on the average). The tuners consist of a set of engine module performance parameters, such as efficiencies and flow parameters, that would allow the engine states and output parameters to be adjusted to provide a more faithful match to the actual engine. A typical architecture for such a model is depicted in *Figure 1-1*.



193441.cdr

Figure 1-1. STORM Architecture

The measured engine parameters, denoted by P in **Figure 1-1**, typically consist of some set of engine speeds, and inter-stage temperatures and pressures. The input set consists of the independent engine and flight condition parameters required to drive the actual engine and the engine model. These are typically fuel flow, bleed commands, variable stator vane positions, altitude, Mach number, and the like. The engine model produces a noise free estimation of the measured engine parameters, denoted by \hat{P} , which represent the nominal baseline levels and are compared to the actual measured values to produce a residual parameter vector r :

$$r = P - \hat{P}$$

The residual vector contains the effects of module deterioration, (denoted by \hat{x}), measurement noise, sensor bias (if present) and any attendant modeling error. The vector \hat{x} consists of some subset of Module performance parameters, such as changes in adiabatic efficiency ($\Delta\eta$), flow capacity ($\Delta\Gamma$), and nozzle areas (ΔA) for the compressor(s) and turbine(s). This vector (\hat{x}), sometimes referred to as the set of tuners, is made available to the engine model (SVM), where they are used to adjust the output engine parameters to more closely match the observed values. In the closed loop operation, these tuners attain the required level to force the residuals to an average level of zero. The result of such a system is the creation of a set of virtual sensors, (\hat{P}), that can be used for fault detection and accommodation typically performed by the engine control [full-authority digital electronic control (FADEC)].

The primary goal of this engine model-KF system (depicted in **Figure 1-1**) was to increase the accuracy of engine parameter synthesis and provide engine module performance tracking. The module performance changes calculated by the KF are used to adapt or tune the SVM to the engine being monitored by forcing the model estimated parameters to more closely match (and track) the measured engine parameters as the engine underwent various power maneuvers and degrading over time.

For this architecture to work, an implicit assumption is being made, i.e., that the physics-based model (SVM) is a fairly good match of the particular engine being monitored. Unfortunately, this assumption is seldom satisfied due to engine-to-engine variation, the simplicity of the SVM itself (needed for real-time operation), not to mention that engine modifications/improvements occurring over the engine's life cycle typically do not find their way into

revised models for on-board usage in a timely fashion (if at all). The results of an engine/model mismatch will cause corruption in the performance measures (tuners) that are being tracked. In effect, the tuners (as their name suggests) become unwilling mathematical artifacts, assuming whatever value is required to drive the residuals to zero. This is illustrated in *Figures 1-2* and *1-3*.

In *Figure 1-2*, there is a simulated (very) slow transient (2,000 seconds) from idle to takeoff back to idle power. The blue traces represent the outputs from a non-linear aero-thermal engine simulation of the monitored gas path parameters (P) and the red traces the output of the SVM (\hat{P}) without the KF in the loop. The mismatch is intentional to show what the effects would be on the tuners when the KF is enabled to drive the mismatch as close to zero as possible. This is depicted in *Figure 1-3*. As can be seen, the performance measures meander between ± 10 percent to effect closure, thereby rendering them ineffective as true performance tracking measures. Since the residual is being driven as close to zero (as possible with the tuners) when the KF is enabled, the attendant estimated parameters (\hat{P}) in this case will be closer to those observed on the actual engine being monitored, as can be seen in *Figure 1-4*. The closure is not perfect, but is certainly better than that observed in *Figure 1-2*. Thus, the analytical redundancy provided by these virtual sensors (\hat{P}) could still be of benefit to the fault detection and accommodation (FDA) logic typically used in modern FADECs to facilitate the control adaptation in the case of dual channel mismatches. However, for enabling meaningful performance tracking, a more faithful model representation would be needed.

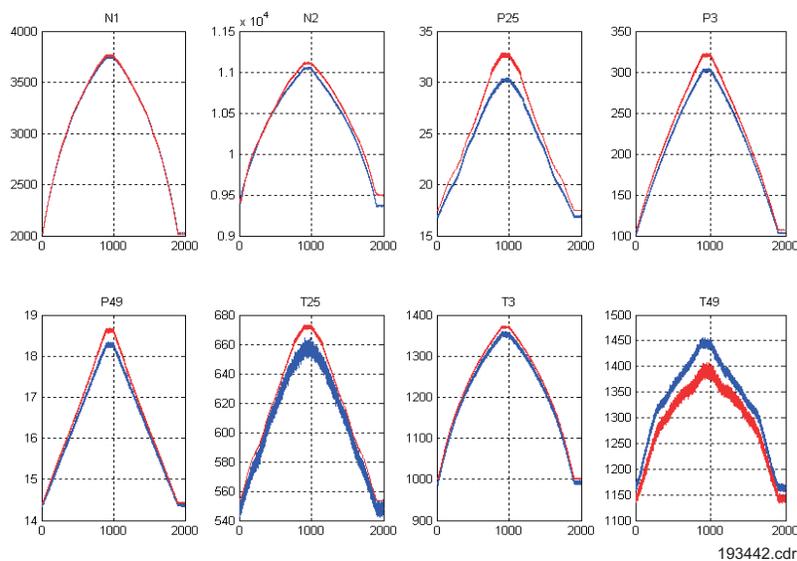


Figure 1-2. Simulated SVM/Engine Mismatch: Without KF

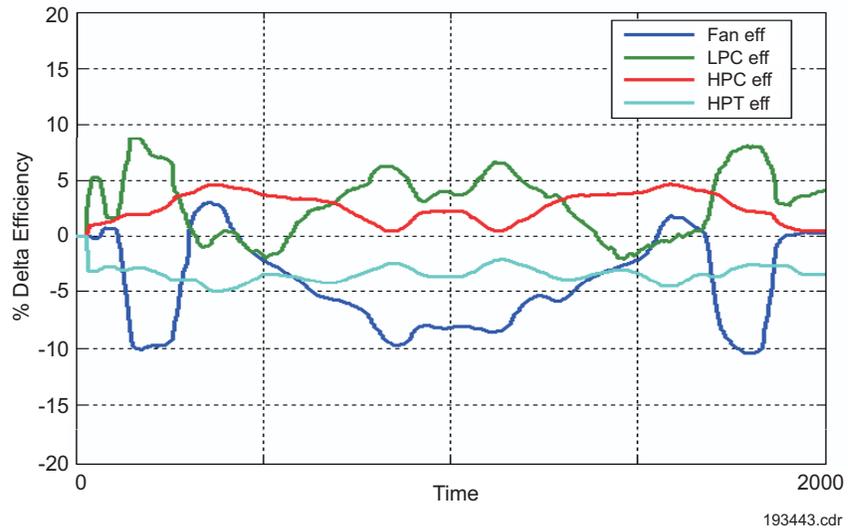


Figure 1-3. Effect on STORM Tuners of SVM/Engine Mismatch

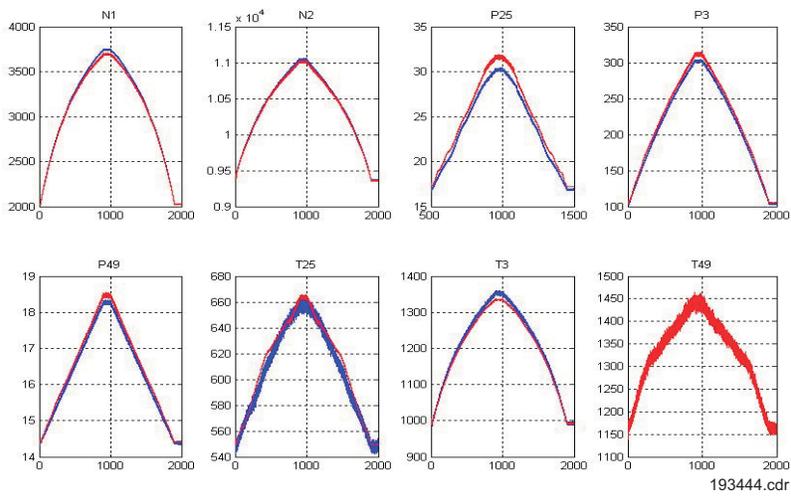
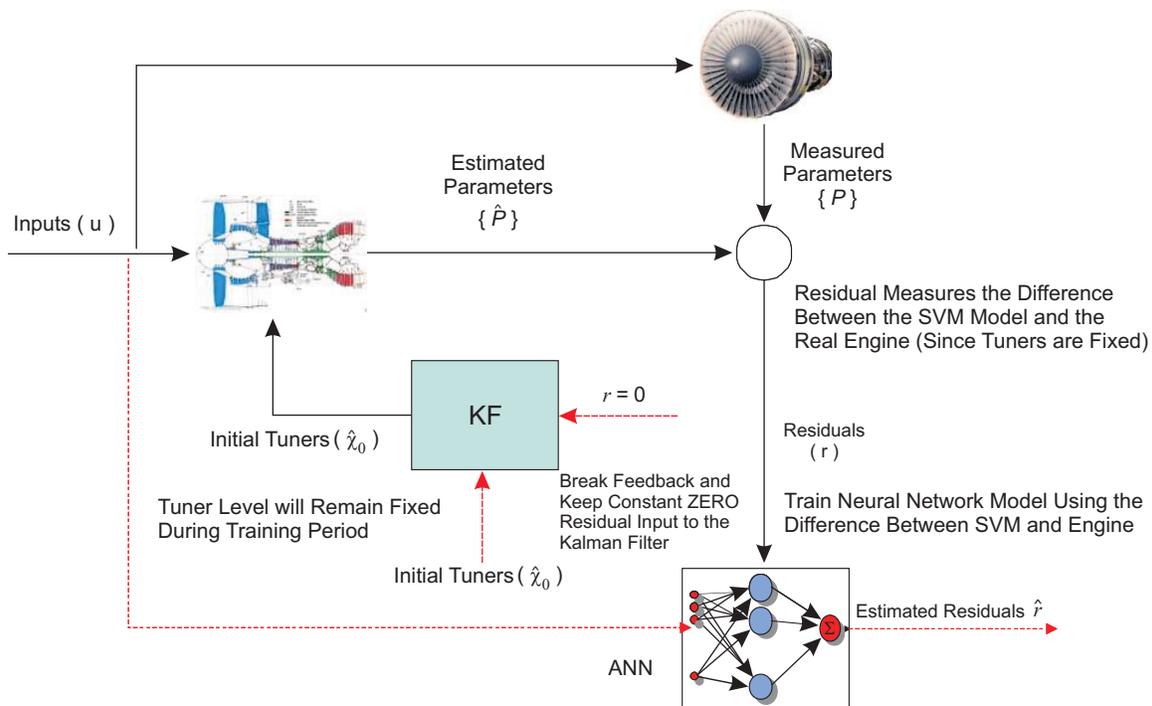


Figure 1-4. Simulated SVM/Engine Mismatch: With KF

1.5 BACKGROUND: ESTORM OVERVIEW

To provide a more accurate tracking of the module performances changes of the engine being monitored, a modification to the original STORM system is required. To prevent any simple engine model (SVM) deficiencies from being absorbed into the tuners (efficiencies, flow capacities, etc.), it is necessary to create a more accurate model for the particular engine being monitored. Engine-to-engine variation along with the real-time constraints imposed on the system would all but rule out a physics-based model approach to the problem. Instead, a hybrid model solution will provide the requisite accuracy and speed. This system is referred to as eSTORM.

A hybrid gas turbine engine model consists of physics-based and empirically derived constituents. Physics-based models would consist of piecewise linear or non-linear aero-thermal models of varying complexity, wherein the SVM is a simple example. In contrast, empirical models are derived solely on the basis of collected data. A typical architecture for such a hybrid model that might be used for the purpose of engine performance tracking is depicted in *Figures 1-5* and *1-6*.



193445.cdr

Figure 1-5. Constructing the Empirical Element of the Hybrid Model

Figure 1-5 illustrates a typical configuration wherein an empirical modeling process captures the difference between the physics-based engine model and the actual engine being monitored. The empirical element can take many forms, however, it has been convenient to use a multi-level perceptron (MLP) artificial neural network (ANN) for this purpose. When the empirical model is complete, the hybrid structure would take the general form shown in *Figure 1-6*.

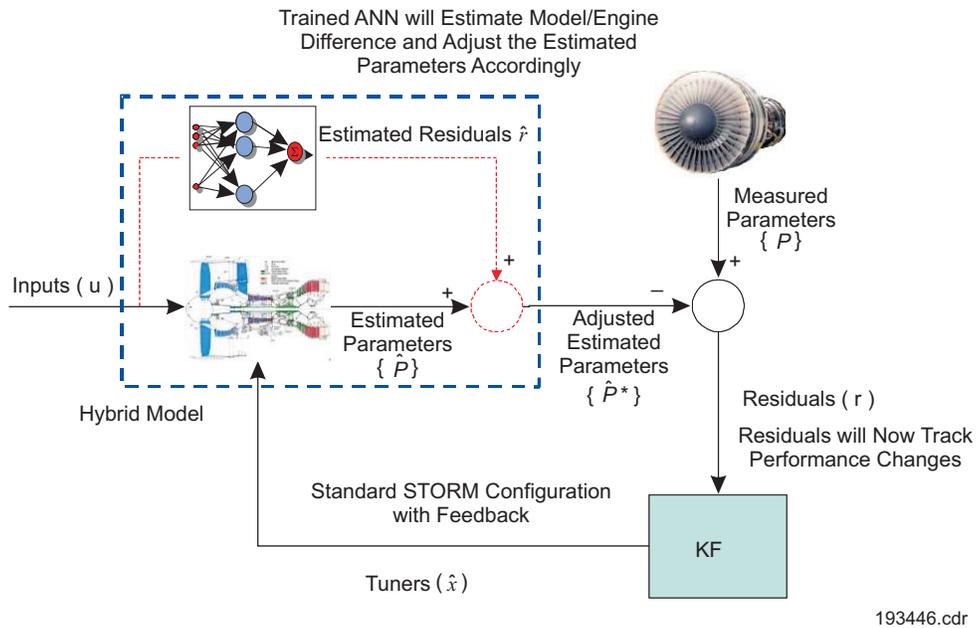


Figure 1-6. *Implementing the Empirical Model Element to Form the Hybrid Model*

The combination of the empirical element and the physics-based engine model provides a more faithful representation for the particular engine being monitored. This provides more meaningful residual information from which an engine performance change assessment can be performed since potential (physics-based) model inaccuracies and shortcomings have been effectively removed by virtue of the empirical element.

This strategy accomplishes two things. It will improve the parameter synthesis accuracy beyond that which is achieved through the basic STORM approach (previously illustrated in **Figure 1-4**) and it provides a zero reference for the estimation of the module performance changes (tuners). These two effects are illustrated in **Figures 1-7** and **1-8**, respectively, for the 2,000-second power transient under consideration.

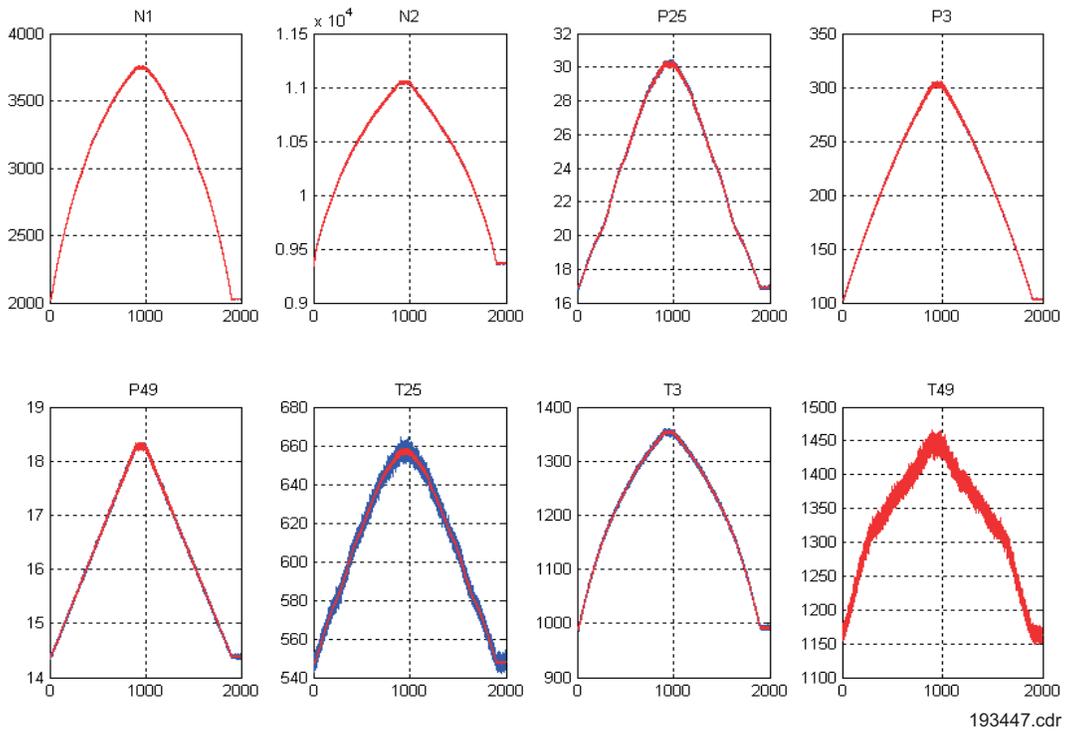


Figure 1-7. Simulated SVM/Engine Mismatch: With Hybrid Model eSTORM

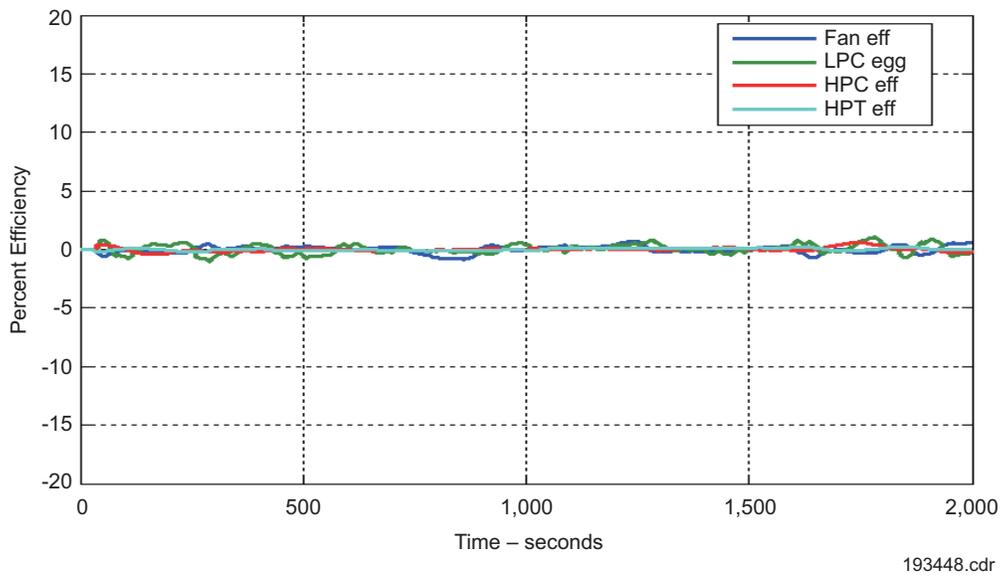


Figure 1-8. Effect on Tuners of SVM/Engine Mismatch Using eSTORM

The approximate zeroing out of the module performance deltas (tuners) in the eSTORM system as seen in **Figure 1-8**, ultimately supports the (visible) tracking of these parameters as deterioration occurs over time. An ill-performing module will now stand out among its neighbors, as illustrated in **Figure 1-9**.

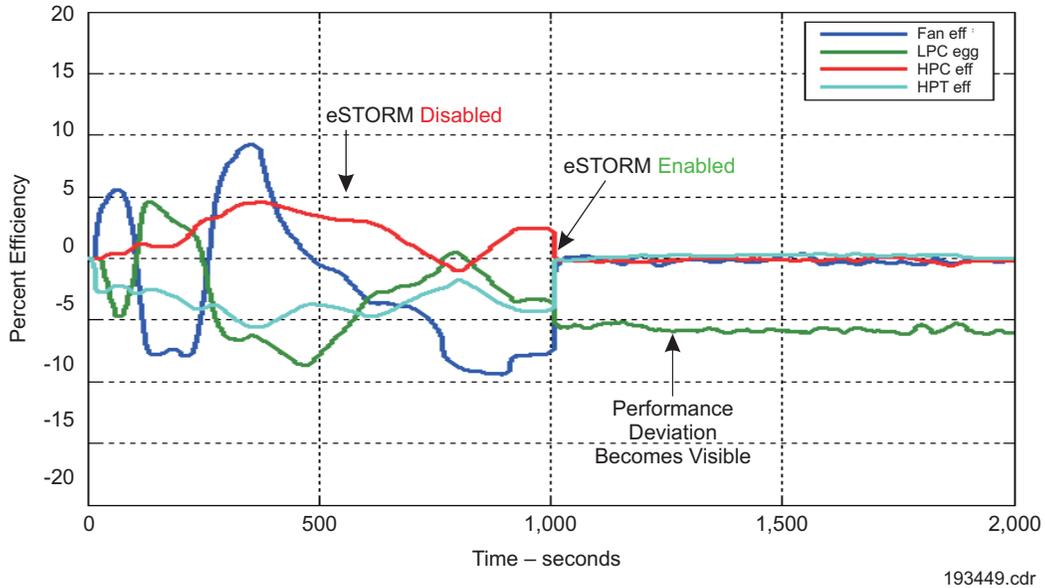


Figure 1-9. Increased Performance Deviation Visibility Using eSTORM Hybrid Model

This increased visibility in module performance tracking in addition to the increased accuracy in the engine parameter synthesis substantiates the use of the hybrid model approach.

2. CURRENT PROGRAM ACTIVITIES

The goals of the present program required refining the hybrid engine model building methodology to be used entirely in an on-board configuration and demonstrating the real-time software on the target processor driven by actual streaming flight data. The PW6000 engine was chosen as the target for this activity since it had an operational STORM system and a large repository of flight data was available from recent flight certification activities.

To accomplish these goals, the following general tasks were identified:

1. Initialize basic eSTORM configuration for chosen target engine.
 - a. This entailed a re-configuration of the PW6000 STORM system to standards developed under previous eSTORM activities mentioned earlier in this report.
2. Model refinements entailing
 - a. Enhancements required for various ambient and flight envelope excursions during quasi-steady-state operating conditions.
 - b. Accommodations required for engine transients.
 - c. Data normalization methods supporting model development.
 - d. Extension of the basic engine SVM to include thermocouple modeling for all temperature measurements (T25, T3, and T49).
 - e. Redesign of the KF to incorporate the thermocouple model changes and accommodate the associated state dynamics.
 - f. Validation of model and filter design against PW6000 flight data.
3. Develop incremental empirical model training methods.
 - a. Refinement of current incremental modeling training methodology for real-time operation
 - b. Develop means of invoking segmented model elements and smoothly transitioning between individual model elements over the flight regime.
 - c. Develop an adequate data clustering process to support model regime recognition in real-time.
 - d. Develop adequate neural network training methods for real-time or near real-time implementation.
4. Real-time implementation development
 - a. Determination of bandwidth requirements.
 - b. Generation of software code.
 - c. Demonstration of real-time system in a real-time test bench facility using hardware representative of flight hardware, along with streaming flight data.

In following sections, P&W will provide an overview description of the activities under these tasks, highlighting the technical issues, algorithm development, and implementation challenges.

2.1 INITIALIZING BASIC ESTORM CONFIGURATION

2.1.1 PW6000 STORM System

The current on-board PW6000 STORM consists of an engine SVM with two KF observers. The two-KF STORM system is somewhat unique to the PW6000 and was originally developed to aid in detecting and accommodating measurement error. A Matlab/Simulink version of this STORM system was also available. Several initial tasks were required to baseline the eSTORM system for the PW6000, as follows:

- a. Re-configure the PW6000 STORM to a one-KF STORM system amenable to subsequent eSTORM development.
- b. Confirm that the ground-based Matlab/Simulink SVM is the same as the in-flight SVM embedded in the on-board system.
- c. Locate PW6000 flight-data database
 - Identify required engine/flight parameters required for eSTORM development
 - Prepare automated scripting to download required parameters and populate Matlab/Simulink environment
- d. Prepare Matlab/Simulink environment to accept recorded flight data and process through re-configured STORM.

All of the above tasks were accomplished in the period of performance extending from January 2005 through September 2006. Several observations were made during this investigation:

- a. The Matlab/Simulink PW6000 SVM is essentially identical to the on-board SVM. Several plots of a typical flight indicate some differences; however, they are explainable as follows:
 - For temperatures, the on-board system has thermocouple lags enabled. The Matlab/Simulink system did not have thermocouple lags modeled, and thus there was an expected lead observed during transients.
 - The on-board SVM parameter outputs (on ARINC) were recorded at a substantially lower sample rate (approximately ½ to 1 second) as opposed to the Matlab/Simulink model which has a 100 ms output.
- b. There is a steady-state mismatch between the actual PW6000 engine (recorded data) and the model outputs (SVM). The considerable difference will have to be absorbed by the empirical modeling within eSTORM.

Appendix A contains plots depicting the differences between the actual engine (blue), the on-board STORM (green), and the Matlab/Simulink eSTORM (red). This data represents a sea-level power transient, idle-T/O-idle in 190 seconds, wherein the STORM tuners were turned off. Thus, the output observed is essentially directly out of the on-board model SVM (green). The Matlab/Simulink (eSTORM) was run with ANN off and the tuners disabled so that the output (red) is essentially the model SVM output. This way a direct comparison can be made between the two STORM (SVM) systems and with the actual engine.

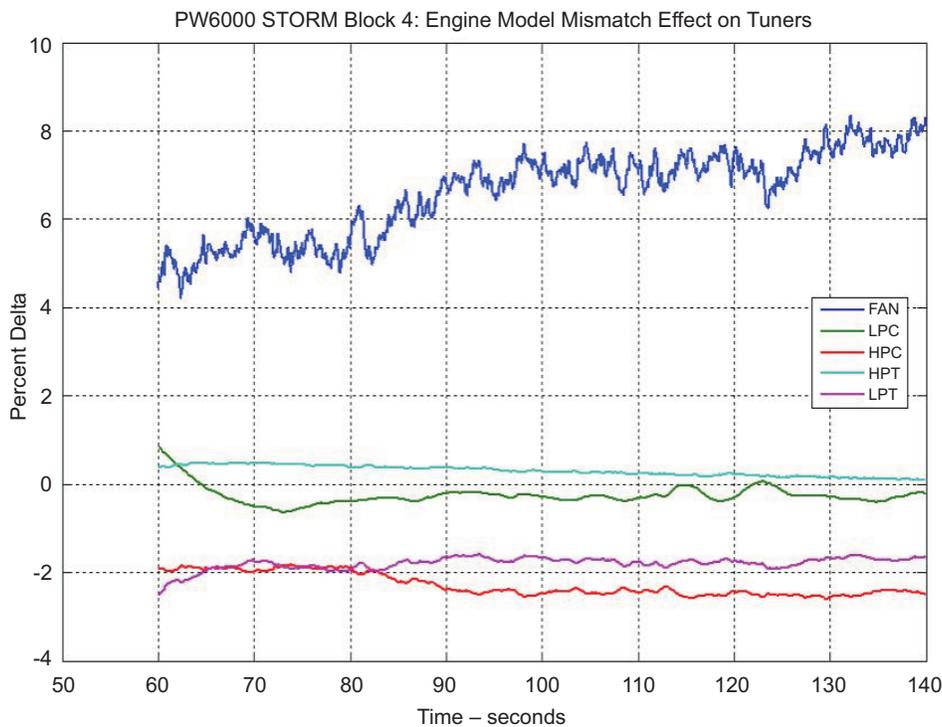
The red and green plots coincide fairly well (except for the temperatures and fast acceleration/deceleration areas for the reasons noted above). P&W has concluded from these that the SVM models (on-board and Matlab) were essentially the same.

It was also evident that the actual engine did not seem to match either model, especially at steady state, with the exception of P5 engine pressure ratio (EPR). The disparity is quite large as noted in **Table 2-1**.

Table 2-1. Error Between SVM and Current PW6000 Engine

<i>Location</i>	<i>Percent Error</i>
N1	2.14
N2	3.78
T25	3.6
P25	11.8
T3	1.05
Pb	3.96
P5	1.54
T5	1.94

This level of error would have a dramatic effect on the STORM tuners, if they had been enabled. The disparity is believed to have been due to the use of a Block 4 fan module during certification testing. A re-run of the STORM model with tuners enabled supported this hypothesis. **Figure 2-1** depicts the performance tuners during the high power steady-state area (approximately 60 to 140 seconds).



193450.cdr

Figure 2-1. Performance Tuners During High Power Steady State Operation

The presence of the tuners reduces the model/engine mismatch error dramatically in all parameters except Pb and P5¹. These errors are tabulated in the **Table 2-2**.

Table 2-2. Error Between SVM and Current PW6000 Engine with Tuners Enabled

Location	Percent Error
N1	0.25
N2	0.12
T25	0.31
P25	0.30
T3	0.51
Pb	2.11
P5	1.38
T5	0.18

2.2 MODEL REFINEMENTS

As noted above, there was an unacceptably high level of disagreement between the current MatLab/Simulink PW6000 SVM and the data obtained during flight certification of the engine. There were two areas that were investigated:

1. Thermocouple modeling to align the temperature dynamics of the SVM to the actual engine
2. A re-generation of steady-state base points from the P&W non-linear engine simulation (state-of-the-art performance program [SOAPP]) to address the steady-state errors.

2.2.1 Temperature Thermocouple Modeling

The transient response of the gas temperatures (T25, T3, and T5) from the PW6000 SVM was faster than the observed engine temperatures. This was due to the lack of thermocouple models in the SVM. Thermocouple models were added to the basic SVM and the resultant transient lag now matches the actual data response with greater fidelity (**Figures 2-2 through 2-4**).

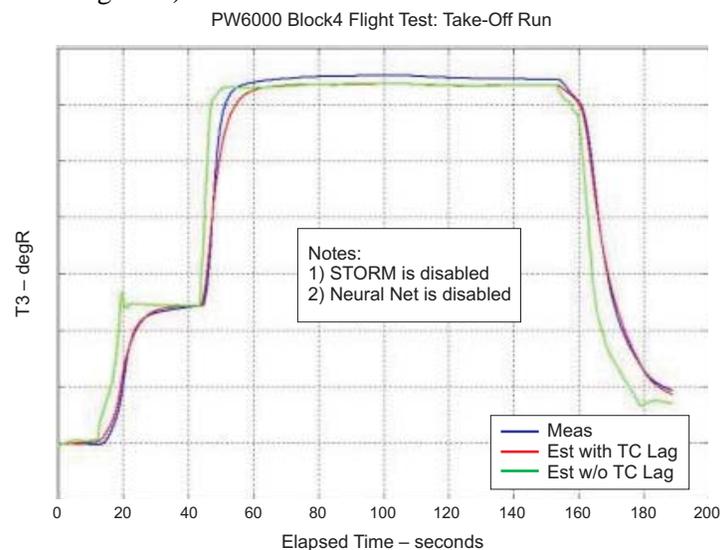


Figure 2-2. PW6000 Block 4 Flight Test Take-Off Run

¹ The lack of closure relative to these particular parameters is due to the reduced informational content of these parameters (relative to the other gaspath parameters) for the specific health parameter set chosen that is estimated by the Kalman Filter.)

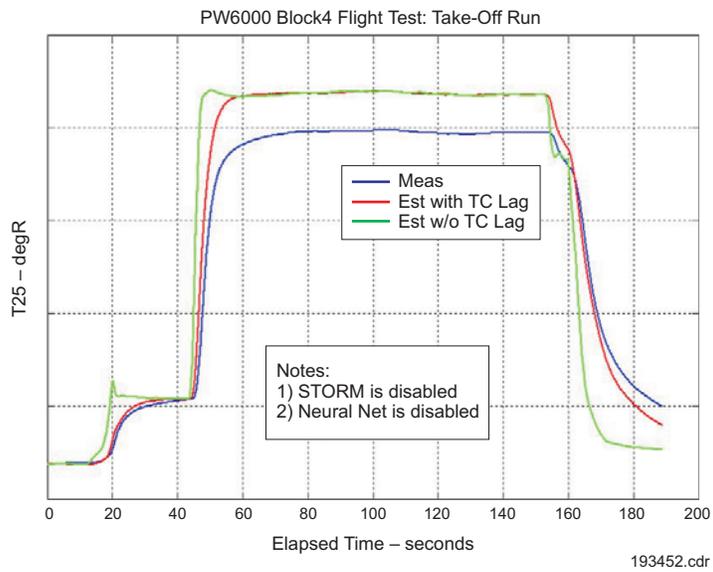


Figure 2-3. PW6000 Block 4 Flight Test Take-Off Run

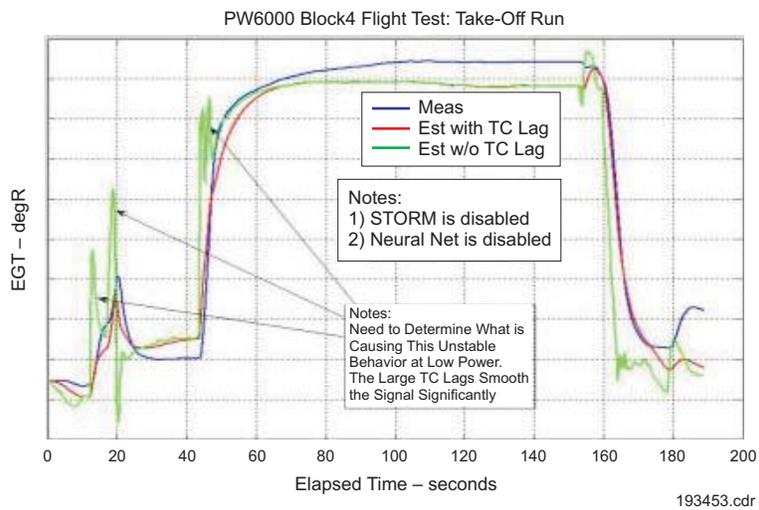


Figure 2-4. PW6000 Block 4 Flight Test Take-Off Run

In *Figures 2-2* through *2-4*, blue represents the actual temperature transient values, green the model value without the thermocouple model, and red with the thermocouple model. Because these values impacted the residual values and dynamics, the KF had to be redesigned to accommodate this addition. This was accomplished by augmenting the state equations in the SVM by adding three additional temperature states representing the thermocouple temperatures. The basic model is a 1st-order lag:

$$\dot{T}_{TC} = \frac{1}{\tau}(T - T_{TC})$$

where:

$$\begin{aligned} T &= \text{gas temperature} \\ T_{TC} &= \text{thermocouple temperature} \\ \tau &= \text{thermocouple time constant} = f(Pb) \end{aligned}$$

The time constants for T25, T3, and T5 were scheduled as a function of burner pressure ($Pb \approx P3$). The addition of the 1st-order lag for these three temperatures is accomplished by augmenting the state vector to include the TC temperatures as states as follows:

$$\begin{bmatrix} \dot{x} \\ \dots \\ \dot{T}_{TC} \end{bmatrix} = \underbrace{\begin{bmatrix} A & 0 \\ (1/\tau)C_T & -(1/\tau) \end{bmatrix}}_{A_{augmented}} \underbrace{\begin{bmatrix} x \\ \dots \\ T_{TC} \end{bmatrix}}_{x_{augmented}} + \underbrace{\begin{bmatrix} B \\ (1/\tau)D \end{bmatrix}}_{B_{augmented}} u$$

$$\dot{x}_{augmented} = A_{augmented} x_{augmented} + B_{augmented} u$$

This formulation incorporates the 1st-order lag dynamics directly into the state equations, so that the resultant system with KF observes the appropriate parameter dynamics. More details can be found in Appendix A.1, *Accommodating Thermocouple Dynamics*.

2.3 EMPIRICAL ELEMENT MODEL BUILDING

P&W has introduced the basic hybrid model concept, but has not discussed any of the many implementation issues associated with this type of an approach. Since this methodology is predicated on developing an empirical description of the difference between a (simple) fixed physics-based engine model and the actual engine being monitored, the method employed for determining the empirical element, and the associated data collection requirements play a central role in this development. Other issues, such as the configuration management of the hybrid model, accommodating the effects of simple line maintenance activities, instrumentation problems and replacements, and flight domain excursions, also contribute to the feasibility of a real-time performance diagnostic tracking system, but fall outside the scope of the present study and will not be addressed.

The empirical model that will be used in this application will take the form of a series of n node hidden layer multi-layer perceptron neural networks (MLP NNs) for each of the parameter residuals being modeled. For this study P&W chose $n=25$, which appears to be adequate for the task at hand. For the PW6000, there were eight such MLPs, one each of the measured parameters (N1, N2, T25, P25, T3, P3, T5, and P5). The inputs for each of these MLPs consisted of engine and flight parameters:

1. Altitude
2. Mach number
3. Normalized low rotor speed
4. Stator vane angle
5. B25 bleed
6. B8 stability bleed
7. Active clearance control (ACC)
8. Air-oil cooling (AOC).

The architecture for each MLP is depicted in *Figure 2-5*.

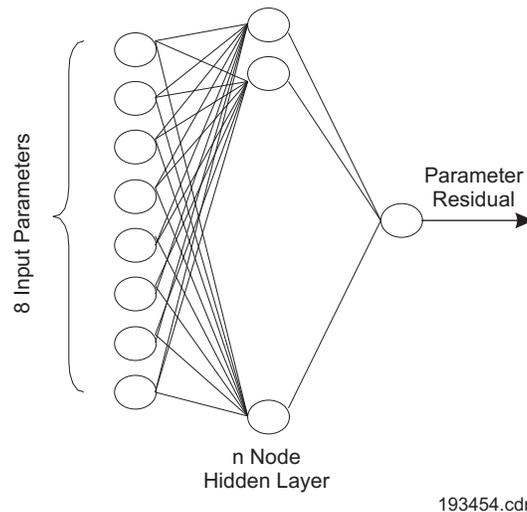


Figure 2-5. *Parameter Residual Empirical Model MLP Structure*

An empirical model of this type (represented by an MLP), trained at a fixed flight condition, does exhibit a degree of robustness in the sense that it encapsulates a region surrounding that operating point with sufficient accuracy such that the parameter synthesis estimates and the module performance tracking are left uncorrupted. However, it could not be expected to cover an entire flight regime. Thus, it is natural to consider a partitioning of a typical flight envelope into contiguous regions and developing individual empirical models for each region in order to provide adequate coverage. The resultant configuration would require a means to smoothly transition between models as the aircraft’s flight trajectory progressed across the flight envelope. A suitable partition might entail having regions as depicted in *Figure 2-6*.

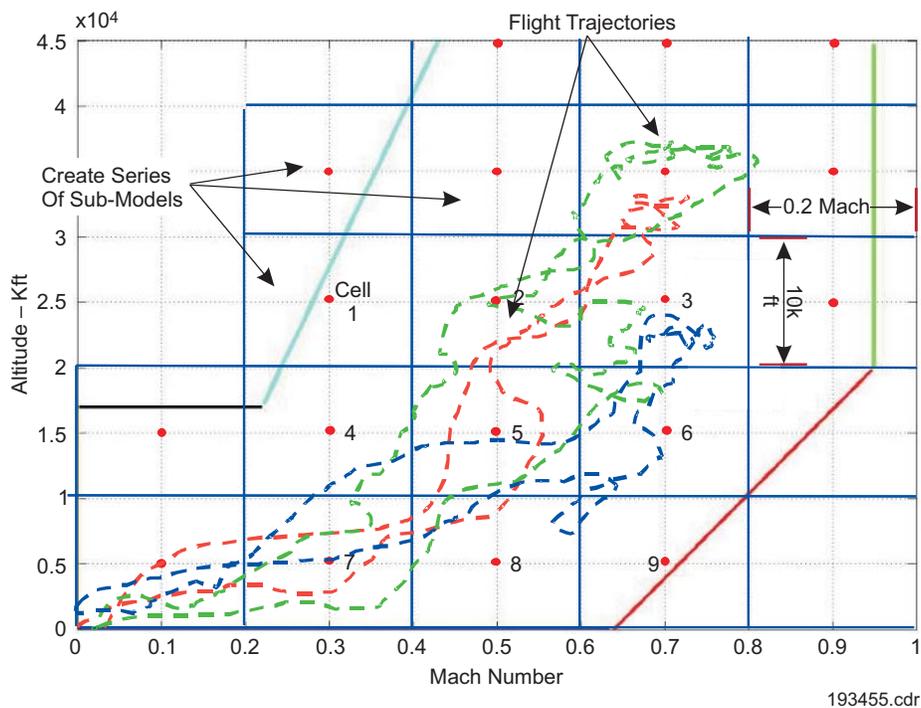


Figure 2-6. Partitioning the Flight Envelope to Create Sub-Models

Each region is defined by its center point (red dot) and a \pm distance in altitude and Mach number (about that point). Empirical models (MLPs) would be derived for each measured parameter residual and each region in the partition. For this project, a partition of 12 cells was used, hence the number of (sub-)models required would be 96 ($=8 \times 12$). This may seem like a large number, however, given the small size of the MLPs and their computational efficiency, their implementation in real-time operation is easily achievable as will be seen in the later sections. The more difficult task to address is how to collect sufficient data and train the networks to provide sufficient flight envelope coverage.

Traditionally, MLPs are trained offline in a batch mode once sufficient data has been collected for that purpose. This would become a burdensome proposition to apply across a fleet of engines. It would also defeat the goal of providing the level of autonomy P&W seeks in an on-board diagnostic system. Since P&W is tracking to what degree component performance may change on a given engine, what is needed is a methodology for training the system onboard without the need to download engine and flight data to a ground station for batch processing. It is also evident that this methodology must support sequential building of the empirical model (ANN) over many flights, since it is unlikely that sufficient data would be collected on any single flight to capture the entire flight envelope, or for that matter, any given partition of the flight envelope. Such a method would need to be performed within the computational bandwidth of present day microprocessors, not require excessive amounts of memory for storing data of the interim models and have a means for self-determining when a given model is complete (in order to end the model building phase).

To achieve autonomy, it is necessary to develop a strategy and process that supports model building to be done in an incremental fashion as data is collected during flight as opposed to the traditional method of collecting all the necessary data and then developing the model. Earlier in 2004, a bootstrap methodology was developed that provided the first introduction to such an incremental empirical model building strategy. A recent algorithmic breakthrough in this development has achieved (from benchmark testing) a real-time computational reduction of

1,000 to 1¹, with an attendant reduction in operating memory requirements relative to the original bootstrap concept.

The new methodology takes advantage of the source of data, (i.e., a gas turbine engine in flight). The first step in the bootstrap process was to partition the data into groups that would be subsequently represented by a set of radial basis functions used to generate the pseudo-data for modeling purposes². Most self-organizing strategies for partitioning data are generic in the sense that the data processed by these algorithms can be quite varied in its form and magnitude. This is certainly the case with batch processing a set of data collected over a long time period. However, in formulating an incremental strategy advantage can be taken of the fact that from discrete time k to $k+1$ ³, the inputs (altitude, mach number, etc.) and gas turbine parameter outputs (speed, temperature, pressure residuals) are not changing radically. This fact enables a simple (and effective) ad hoc self-organizing strategy to be incorporated (which will be described momentarily). The second advantage comes from the realization that there is nothing gained (from a model information perspective) by using random (pseudo) data above and beyond the statistical moments (mean and variance) used to generate them. Thus, it would be sufficient to use the mean and variance (weighted by sample size) directly in the model building process. To do this, the MLP training algorithm needed to be changed to use these quantities directly in such a way that the quantities $\mu, \sigma, n_{\text{sample}}$ would have the same effect as a sample of random data of size n_{sample} generated from the normal distribution $N(\mu, \sigma)$. Across a flight envelope partition (cell) this would yield a tremendous reduction in the training set and is largely responsible for the 1,000 to 1 reduction in computation time. The algorithm, which will be described in the sequel, makes use of Gaussian mixture models (GMMs) which are essentially multi-dimensional normal distributions (or radial basis functions) consisting of (n-dimensional) quantities $\{\mu, \sigma, n_{\text{sample}}\}$.

2.3.1 Empirical Model Building Overview

To accomplish the empirical model building task that would be suitable for a real-time, on-board environment, operating in a totally autonomous operational mode, a two-stage model building strategy was adopted. This strategy incorporates a real-time process and a non-real-time (off-line) process. Both processes are envisioned to be accomplished on-board, automatically, without manual intervention.

The first stage of the process is performed in real-time. It is the process of forming the GMMs alluded to in the preceding section. In essence, this can be thought of as a data compression phase, wherein clusters of input and residual parameter data are formed, in real-time during flight and temporarily stored for subsequent processing in the second (off-line) stage of the model building procedure. In addition to forming the (compressed) data clusters, the first stage of this process also performs:

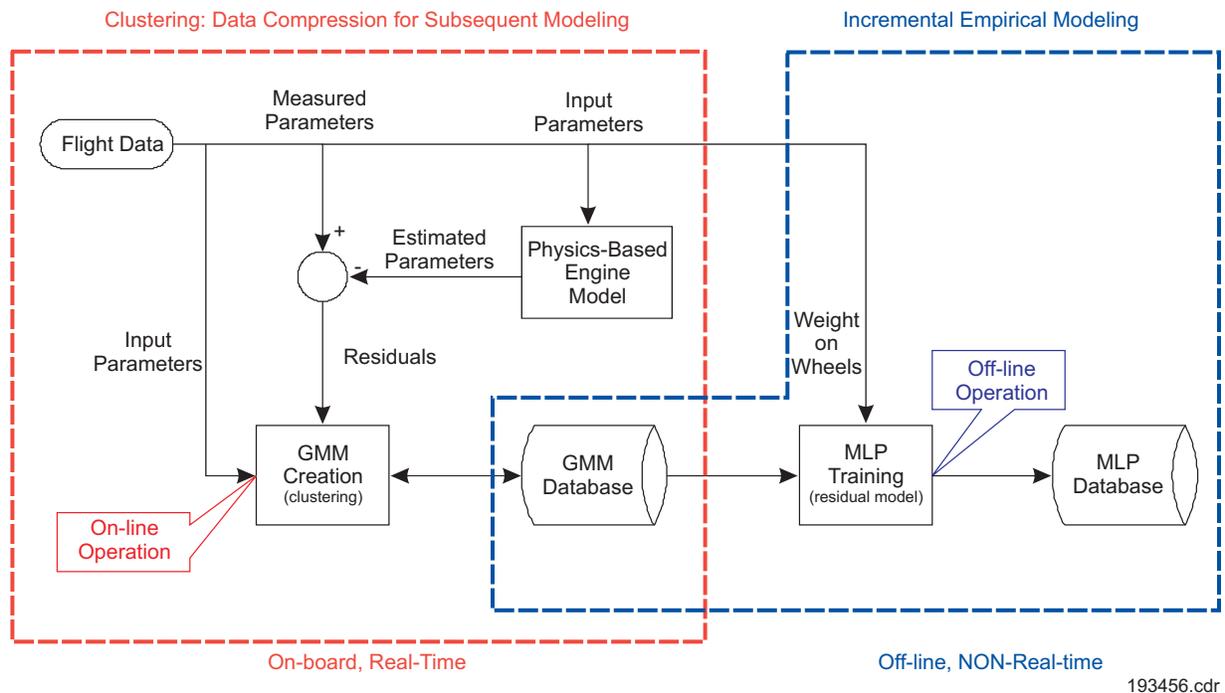
- a. Regime recognition (i.e., knowing what cell partition, and hence sub-model is being built)
- b. Whether the current data point should be part of a new cluster or is indeed already covered by an existing cluster from a previous flight (in which case the current data point is already modeled and the eSTORM configuration in *Figure 1-6* can be executed using existing MLPs to track performance changes).

¹ Time reduction depends on sample size of the data being processed and is non-linear with sample size.

² *A Bootstrap Data Methodology for Sequential Hybrid Model Building*, Proceedings of the 2005 IEEE Aerospace Conference. Big Sky, MT, March 2005

³ Assuming typical sampling rates of in the range of 10-20 Hz.

The overview structure of this two-stage process is depicted in *Figure 2-7*.



193456.cdr

Figure 2-7. Two-Stage Empirical Modeling Approach

Figure 2-7 depicts *only* the overview process of GMM creation (real-time) and the sequential MLP training (off-line, but on-board) and does not indicate the intricacies involved in steps a) and b) and the general control of the eSTORM program. Relative to *Figure 2-7*, the steps enclosed within the red dashed lines form the first stage of the process that is performed on-board in real-time. As each data point is received, (typically at 10 to 20 Hz), its cell classification is determined using altitude and mach number. This defines where in the flight envelope the data resides and what sub-model is in effect. Using the input parameters (N1, N2, T25, P25, T3, P3, T5, and P5) (of this data point), a determination is made as to whether the data point should be processed by an existing MLP sub-model (as depicted in overview in *Figure 1-6*) or whether the current data point represents an area in the flight regime that has not been previously modeled. This determination is made, by establishing whether or not the n-dimensional input vector falls within a neighborhood of an already existing GMM residing in the GMM database. A nearest neighbor criterion, to be described later, is used for this determination. If the data point falls outside any already established GMM, it is a candidate for the formation of a new GMM. The formation of a new GMM is essentially a data clustering process that is performed in real-time.

The steps enclosed in the blue dashed lines in *Figure 2-7* are performed off-line (i.e., non-real-time), and are performed on-board at the end of the flight. This can be launched by a weight-on-wheels (WOW) signal. This is the second stage of the incremental model building process. The GMM database is perused to determine if new GMMs have been added to the database. In this case, on a cell-by-cell basis, new MLPs are generated for each residual parameter (8 total), for each cell (or sub-model) that has new GMMs, and become the current incremental model in the MLP database. The training of the MLPs are done individually. Relative to traditional MLP training methodology that would train directly on stored flight data, the proposed method trains on the GMMs. The effect (on the residual) model is essentially the same. *Figure 2-8* highlights the departure from traditional batch training.

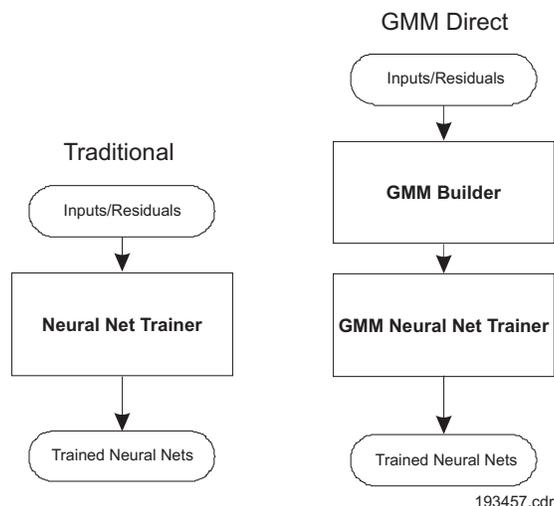
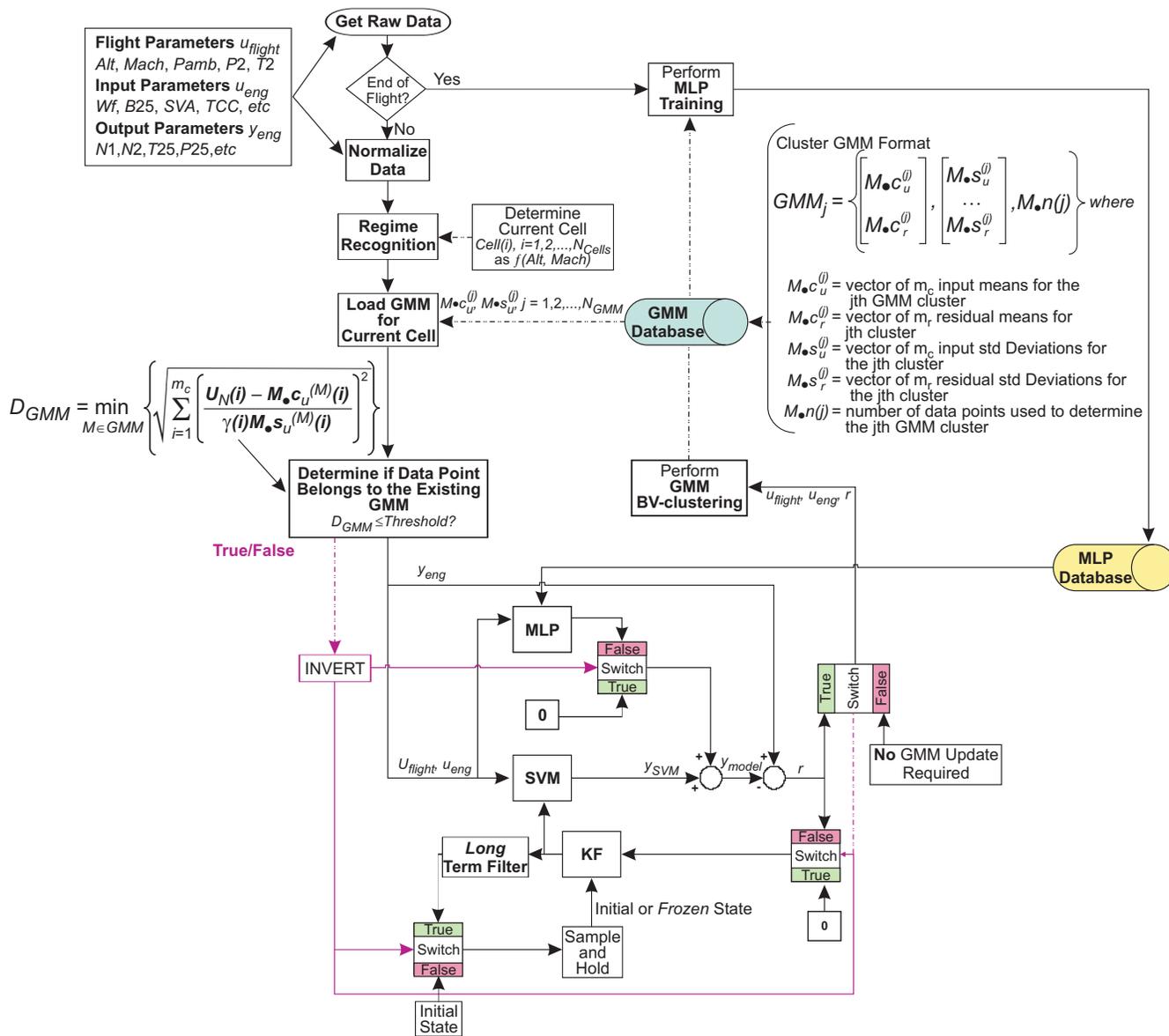


Figure 2-8. MLP Training Overview

To summarize, for each incoming data point:

1. Determine the flight envelope partition (*cell*) to which the data point belongs. This is accomplished by choosing the nearest neighbor cell (Altitude and Mach number).
2. Determine if the current data point belongs to a previously defined *cluster* (within this cell [i.e., GMM]). Note: These *clusters* include mean and standard deviation of the *input* vector and the measurement *Residual* vector. The *input* vector (means) are used to determine the current point's proximity to existing *clusters*, while the *residual* (means) are the compressed data that provide (training) definition for the subsequent empirical modeling (MLP).
3. If the point lies within an existing cluster, process the point through eSTORM, obtain the next data point and go to 1. If the point is outside any existing cluster, begin the GMM cluster process, obtain next data point, and go to 1.
4. When the flight has ended (queued off of WOW), peruse Empirical Model (GMM) Database for new information. MLP training and updates are accomplished during time between WOW and engine shutdown.

Since the training of the empirical model is done incrementally, it is necessary for the eSTORM system to be able to process data that resides in *already trained* regions and to process new training whenever required without compromising the STORM tuners (module health parameters). The overall process (in overview) is depicted in **Figure 2-9**.



193458.cdr

Figure 2-9. eSTORM Overview Process Architecture

2.3.2 GMM Clustering Process

The empirical model database contains information relating to the reduced order encapsulation of the physics-model/engine residual difference data determined in the on-board stage of the process referred to as GMMs and the actual empirical model of the residual difference determined from the GMMs. This latter model is determined in the second stage of the process (off-line).

The GMMs are multi-dimensional Gaussian distributions that are characterized by two statistical quantities, a mean and a measure of dispersion, the standard deviation. These GMMs are multi-dimensional quantities and are determined for the vector of independent input parameters u and the vector of dependent engine parameter residuals r . The residual vector r is determined as the difference between measured engine parameters y , and estimations of these parameters from a physics-based model component (**Figure 1-5**). The input parameters, u , includes altitude, mach number, fuel flow demand, engine bleed commands, active clearance control commands, variable geometry commands and other parameters as needed. The measured engine parameters, y (and hence the residuals r), can include engine speeds, inter-stage temperatures and pressures, flows, etc. A GMM will consist of a vector of means and a vector of standard deviations for both the *inputs* and *residuals* that are computed recursively during the GMM clustering process as follows:

Inputs:

$$\bar{u}_N(i) = \left(\frac{N-1}{N}\right)\bar{u}_{N-1}(i) + \left(\frac{1}{N}\right)u_N(i)$$

$$\sigma_N^{(u)}(i) = \sqrt{\left(\frac{N-2}{N-1}\right)[\sigma_{N-1}^{(u)}(i)]^2 - \left(\frac{N}{N-1}\right)(\bar{u}_N(i) - \bar{u}_{N-1}(i))^2 + \left(\frac{1}{N-1}\right)(u_N(i) - \bar{u}_{N-1}(i))^2}$$

$$\forall i = 1, 2, \dots, m_c$$

Residuals:

$$\bar{r}_N(i) = \left(\frac{N-1}{N}\right)\bar{r}_{N-1}(i) + \left(\frac{1}{N}\right)r_N(i)$$

$$\sigma_N^{(r)}(i) = \sqrt{\left(\frac{N-2}{N-1}\right)[\sigma_{N-1}^{(r)}(i)]^2 - \left(\frac{N}{N-1}\right)(\bar{r}_N(i) - \bar{r}_{N-1}(i))^2 + \left(\frac{1}{N-1}\right)(r_N(i) - \bar{r}_{N-1}(i))^2}$$

$$\forall i = 1, 2, \dots, m_r$$

where m_c and m_r are the number of inputs and measured engine parameter residuals respectively, with \bar{u}_N , $\sigma_N^{(u)}$ representing the $m_c \times 1$ vectors of (running) input means and standard deviations and \bar{r}_N , $\sigma_N^{(r)}$ representing the vectors of engine parameter residuals means and standard deviations after N data points have been processed. When the GMM cluster process terminates (to be described below), these mean and standard deviation vectors, along with the sample size (N) (at termination) are stored in the GMM database.

The process of generating the GMMs during flight and the subsequent determination of the incremental residual model offline will be described in a series of process steps. The system continuously processes engine data as it transcends a trajectory within the flight envelope. The first step is to determine if the flight has terminated. This can be achieved by monitoring a number of possible parameters. As an example, WOW would provide such an indication. If the flight is determined to be in progress, the system will continue to monitor and build the GMM database;

if the flight has ended then the system ceases to monitor any further engine data and performs an empirical model (EM) update by re-evaluating the MLP models in the EM database. P&W describes both phases in the sequel.

In the former case, as a data point is received (vectors u and r), some form of data correction and normalization is performed to help reduce the variability in the data. Standard corrections such as gas path standard-day corrections are applied to remove the effects of ambient temperature and pressure. This in itself reduces the order of the empirical modeling effort. Once this is accomplished, the normalized input parameters, altitude and mach are used to determine which cell is in effect. This can be done by calculating the distance from the pre-stored cell centers and selecting the cell that yields the shortest distance.

$$Cell_Number = \arg \min_{i=1,2,\dots,N_{Cell}} \left(\sqrt{\frac{(Alt_i^* - Alt^*)^2}{\sigma_{Alt}^2} + \frac{(Mach_i^* - Mach^*)^2}{\sigma_{Mach}^2}} \right)$$

where

N_{Cell} = Number of Cells

superscript * denotes a normalized quantity, e.g. $Alt_i^* = Alt_i / 50000\text{ft}$

σ = assumed variability in the measurement (standard deviation)

Once the cell is determined, the sub-model element is defined and the current GMM models are loaded from the database for that cell. The next step is to determine if the current point is adequately represented by an existing GMM (in which case no update is necessary) otherwise, a new GMM model element is initiated. The initiation and determination of a new GMM model element entails a number of process steps including an exit criterion to determine when to end the GMM building process. Since the GMM effectively represents a (multi-dimensional) cluster, the process needs to determine when to exit the cluster-building calculation. The criterion depends on the number of points collected to represent the current GMM cluster (which itself depends on certain stability calculations) as well as a distance migration from cluster initiation. A simplified process diagram appears in **Figure 2-10**.

To form a GMM cluster, a minimum number of data points (N_{MIN}) is required. This number depends on whether or not the data is acquired in steady state (where the input parameters are not migrating) or transient conditions. Thus there are two minimum number specifications, one for steady state (N_{MIN_SS}) and one for transient (N_{MIN_Trans}), where $N_{MIN_SS} > N_{MIN_Trans} \cdot N_{MIN}$ is set to one of these two numbers according to a transient operation calculation. This test looks at the rate of change of inputs (e.g., *Altitude*), and determines if the rate exceeds a pre-defined Limit in which case N_{MIN_Trans} will be chosen, otherwise N_{MIN_SS} is chosen for N_{MIN} . Once N_{MIN} data points have been collected, the initial GMM cluster mean and standard deviation vectors (for the input vector u), μ_0 and σ are determined and temporarily stored for reference. As each new data point is acquired, a running exponential average of the input vector is calculated as follows:

$$\bar{u}_N^\alpha(i) = \alpha \bar{u}_{N-1}^\alpha(i) + (1 - \alpha)u_N(i) \quad \text{for } i = 1, 2, \dots, m_c$$

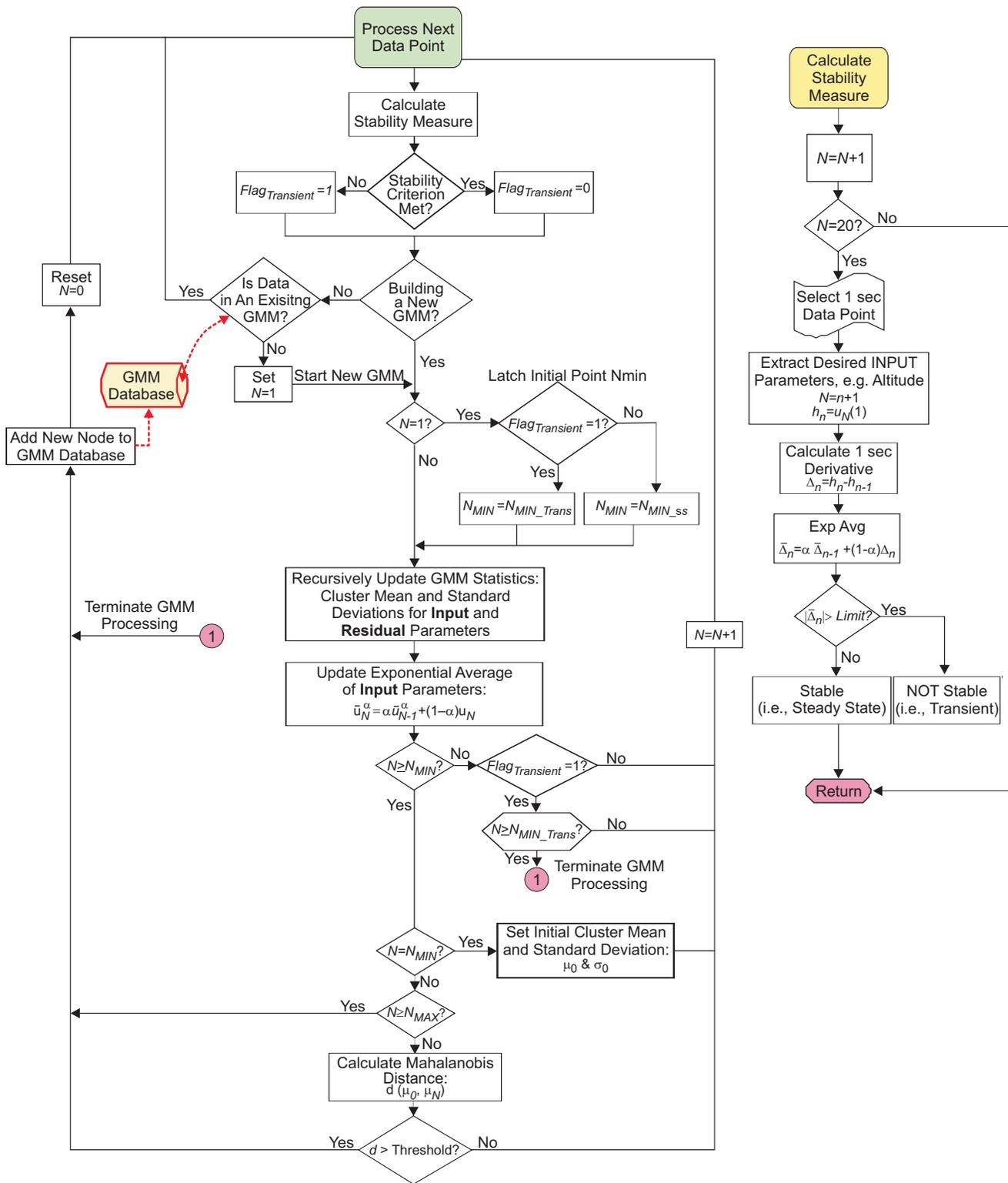
where

$\bar{u}_N^\alpha(i)$ = running exponential average of the i^{th} element of the input vector

$u_N(i)$ = current (N^{th}) data point for the i^{th} element of the input vector

α = pre-defined exponential averaging constant

m_c = number of elements in the input vector



193459.cdr

Figure 2-10. GMM Cluster Process

These statistics are updated for each incoming data point until an *exit criterion* is satisfied to end the GMM clustering process and establish a new GMM cluster point for the database. There are two exit criteria that are applied. If N represents the current number of data points in the current GMM cluster which is being formed then the process ends when one of two conditions are met, i.e. either N is larger than some pre-defined maximum N_{MAX} , ($N \geq N_{MAX}$) or the running average \bar{u}_N^α has migrated sufficiently far from the initial average μ_0 . This latter condition is tested by calculating a Mahalanobis-type of vector distance $d(\mu_0, \bar{u}_N^\alpha)$ and comparing to a pre-defined *Threshold*. The distance measure is calculated by:

$$d(\mu_0, \bar{u}_N^\alpha) = \sqrt{\sum_{i=1}^{m_c} \left(\frac{\bar{u}_N^\alpha(i) - \mu_0(i)}{\gamma(i) \sigma_0(i)} \right)^2}$$

where $\gamma(i)$ is a weighting factor for the i^{th} input parameter noise sensitivity. Thus, the clustering process will end if either of the following conditions are satisfied:

$$d(\mu_0, \bar{u}_N^\alpha) \geq \text{Threshold}$$

or

$$N \geq N_{MAX}$$

When the current cluster (GMM creation) terminates, the newly formed GMM is added to the database and the next data point received follows the same overall process to determine if it is within the domain of an existing GMM and if not, a new GMM formation is initiated.

Referring back to **Figure 2-9**, when a data point is determined to be within the domain of an existing GMM, it is processed by the (existing) hybrid model as depicted in **Figure 1-6** using current MLP empirical model elements. Since the data point is already represented by an existing GMM, an existing MLP empirical model provides the requisite compensation to allow performance estimation to be determined without corruption. If the data point is not within the domain of an existing GMM, the hybrid model is run in frozen mode and a new GMM cluster process is initiated as described above. Running the hybrid model in frozen mode simply means to process the point as indicated in **Figure 1-5** where the Initial Performance Level input to the KF is frozen at the last known value estimated (from data within the current GMM database domain). This will insure that the residuals that are calculated (and used in the GMM cluster formation) do not contain presently known deterioration effects. This is important since we only want to capture (empirically) the differences between the actual engine and physics-based model representation and not *absorb* any deterioration we might have accrued since installation.

Once the flight has terminated, (e.g. with a WOW indication), the second phase of the empirical model process is initiated. The GMM database is perused for newly generated clusters on a cell- by-cell basis. If a given cell (sub-model) is found to have newly generated GMMs, the entire collection of GMMs for that cell (sub-model) are used to generate a new MLP empirical model for each of the residual measurements. The collection of GMMs within a cell contain a compact representation of the flight data (within that cell) in the form of Input parameter and Residual parameter averages, standard deviations and associated sample sizes. This compact data representation can now be modeled with an MLP (or some other representation). This final modeling process is performed offline after weight-on-wheels indication and the subsequent MLPs generated are stored in the EM database, replacing (and hence updating) the previously stored MLPs. These MLPs will constitute the EM element in the hybrid model representation depicted in **Figure 1-5** for subsequent analysis of future flight data.

2.3.3 MLP Training using Clusters

In the second stage of the process, the empirical sub-models are sequentially updated to reflect the new data that was collected during the current flight. In this stage, the GMMs are used as the input data in the training procedure. As mentioned previously, the MLP training process had to be adjusted to use the GMM information directly in such a way as to yield essentially the same residual model result that would have been obtained if the actual residual data had been stored and accumulated over time and used directly in the training in a batch process mode. In order to describe this procedure we will introduce a slightly different nomenclature wherein M represents the GMM model with component centers $M.c$, standard deviation $M.s$, and the number of points included in the GMM cluster $M.n$.

The MLP training is performed by adjusting the various parameters in the neural net to optimize some performance functional. For standard MLP training, this is simply the sum-of-squares-error (SSE) defined as:

$$SSE = \sum_{k=1}^N |r(k) - F(u(k))|^2$$

where $r(k)$ is the k -th point of the residual sequence we are trying to predict, $u(k)$ is the k -th point of the multi-dimensional input data set, and $F(\bullet)$ is the mapping function approximated by the MLP neural net. There are N points in the training data set.

Assume that the GMM model is made up of C clusters. $M.n(i)$ is the number of points represented by cluster i which has center $M.c(i)$ and standard deviation $M.s(i)$. Taking into account the cluster parameters, an equivalent SSE error function to be used for training can be derived as follows:

$$SSE = \frac{1}{N} \sum_{i=1}^C M.n(i) \left| \frac{M.c_r(i) - \hat{F}(M.c_u(i))}{M.s_r(i)} \right|^2$$

where:

$$N = \sum_{i=1}^C M.n(i) = \text{Total number of points}$$

and the ' r ' and ' u ' subscripts refer to the components in the i -th cluster that correspond to the residual and inputs respectively:

$$M.c(i) = \{M.c_u(i) \ M.c_r(i)\}.$$

This is the only change required in the trainer.

2.4 APPLICATION EXAMPLE

To illustrate the algorithms used in this methodology, P&W will apply actual PW6000 engine flight data. In this example, P&W uses some flight data that was collected during a several hour ferry flight that contains stable data at several altitudes and climbs and descents between these altitudes. To illustrate the process we will focus our attention on two segments of this flight. The first segment will contain a level flight at an altitude of 36k ft. The second segment will contain 36k ft altitude level flight, followed by a climb to 38k and level flight at this new altitude. **Figure 2-11** shows the altitude profile for this data.

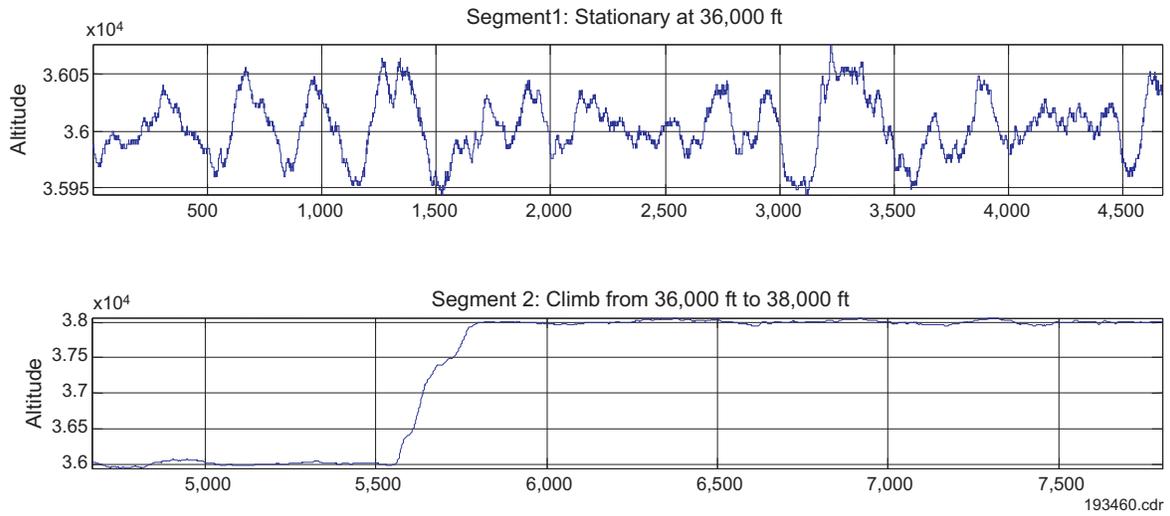


Figure 2-11. Altitude Profile for Example Data

Figure 2-12 shows the input parameters for Segment 1.

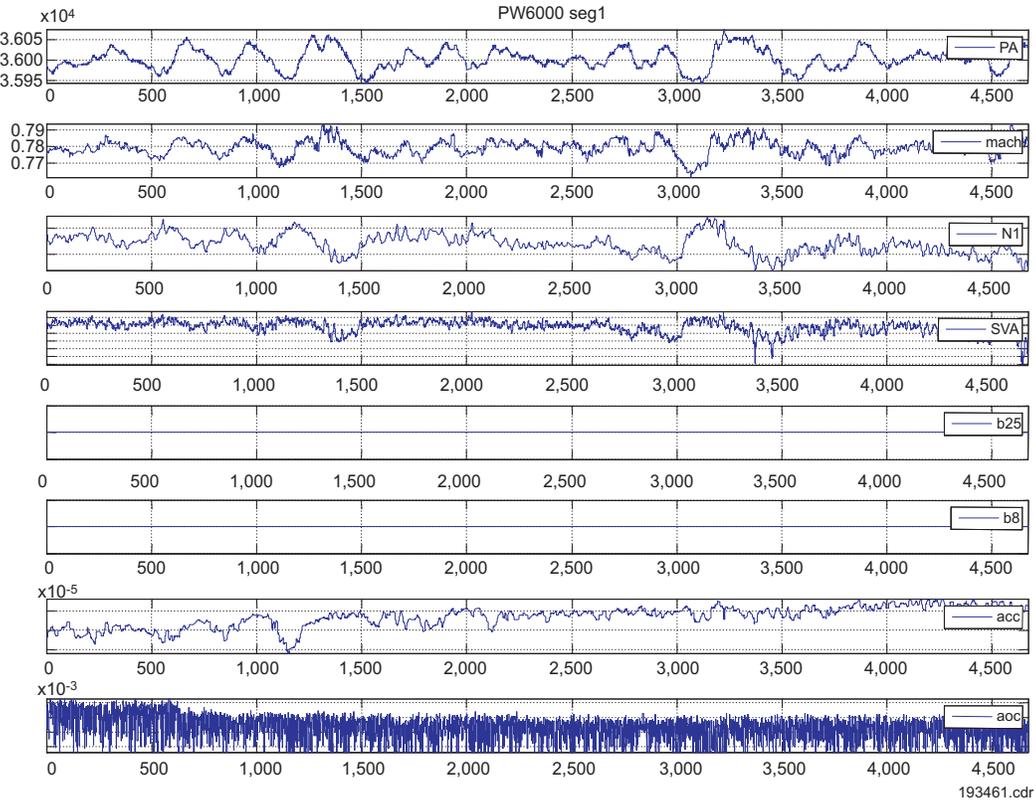


Figure 2-12. Segment 1 Input Parameters

Since the ordinate axis values in *Figure 2-12* have been removed (to preserve the proprietary nature of the data), it is difficult to gauge the stability and non-repeatability of the data. *Figure 2-13* depicts the measurement residuals (in percent) between the main gaspath measurements and the model (SVM) estimates. This plot illustrates that the variation in the measured parameters is a couple of percent, indicating a fairly stable flight condition. It can also be observed that all parameters have a mean value other than zero (which implies that is not a good model-engine match).

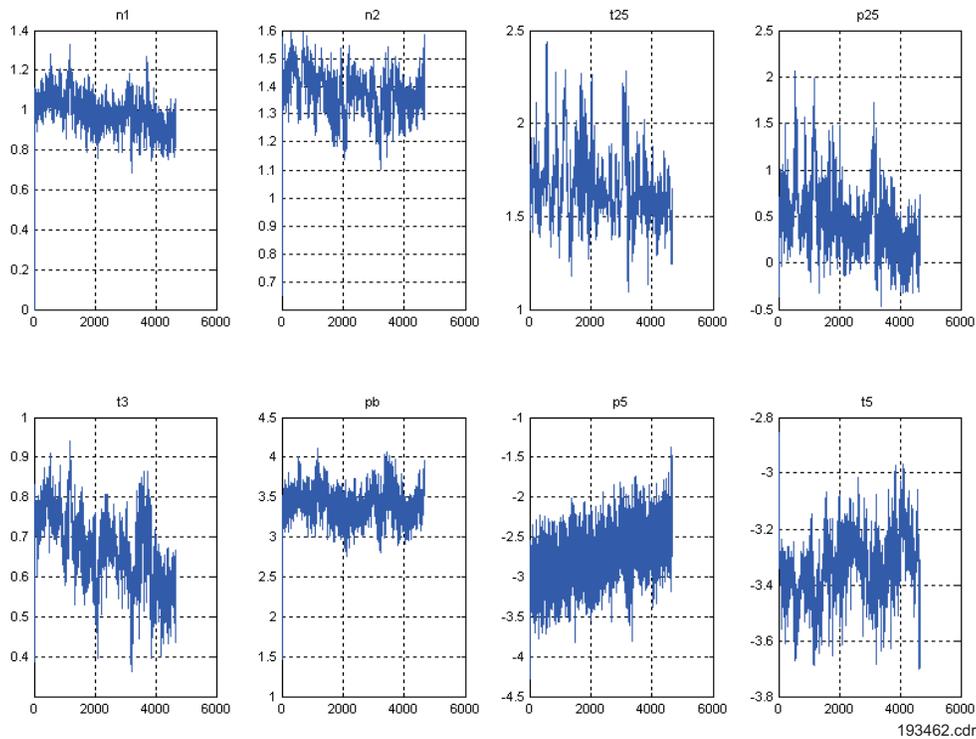


Figure 2-13. Segment 1 Measurement Residual Parameters (Percent Δ) Without KF

If the KF is enabled, then the residuals drive closer to zero as shown in *Figure 2-14*, however total closure is not produced (i.e., there are several parameter residuals exhibiting a non-zero mean). The lack of closure is primarily due to the diminished visibility of the estimated health parameter faults in these particular parameters, i.e., the informational content of these measured residuals is weak relative to the other gas path parameters in tracking (estimating) the fault set selected.

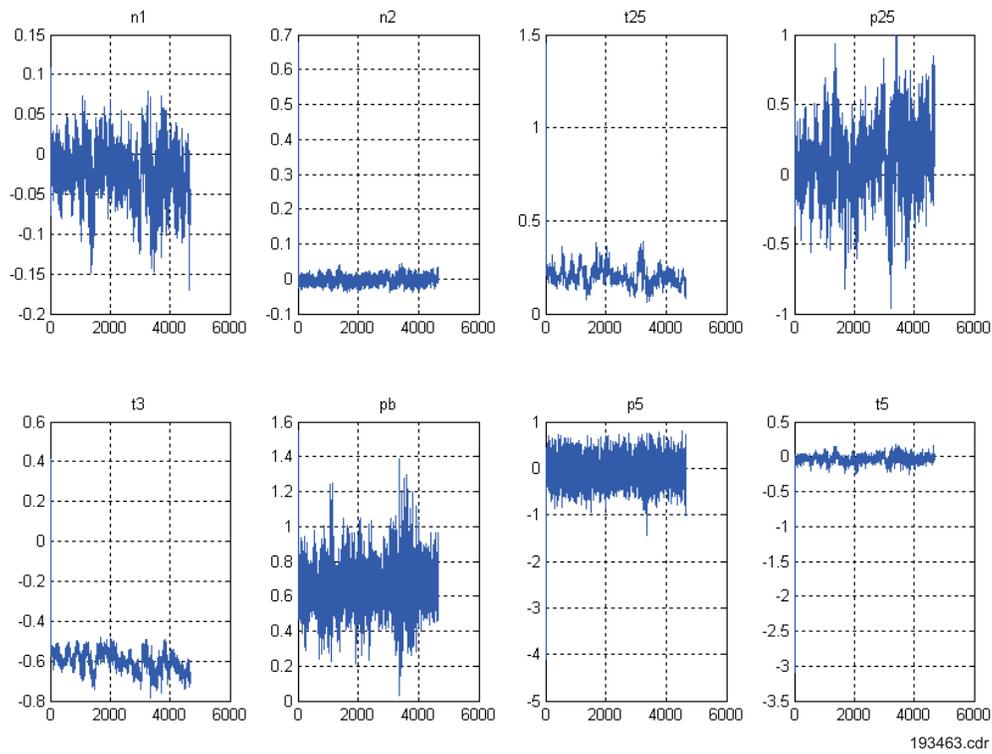


Figure 2-14. Segment 1 Measurement Residual Parameters (Percent Δ) With KF

This partial closure in the residuals comes at the price of corrupting the tuners, which have now absorbed the engine-model difference, plotted in **Figure 2-15**.

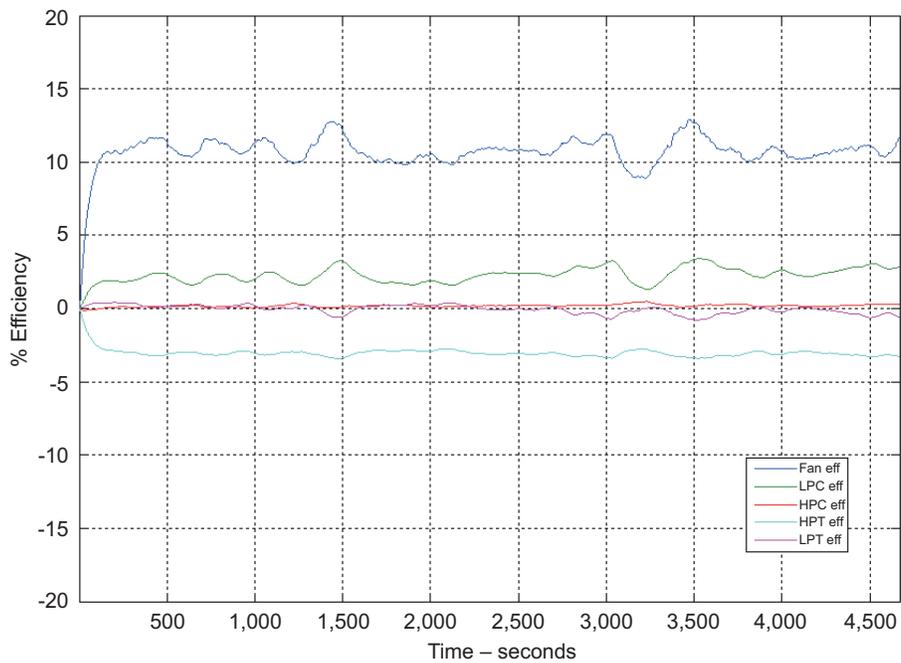


Figure 2-15. Segment 1: Effect on STORM Tuners of SVM/Engine Mismatch

Using the process described above a series of GMM clusters are created in the first stage of the empirical model building procedure. This process uses the input data (*Figure 2-12*) and the residual difference between the SVM model and the engine measurements (*Figure 2-13*) to create this compressed representation. For the Segment 1 data, 23 GMM vectors were created to represent the 93,395 data points in this segment. This represents approximately a 4,000:1 compression. The *location* of GMMs created are plotted in *Figure 2-16*. In this plot the green points represent the altitude-mach location of the Segment 1 data and the pink points represent the GMM center locations with a one sigma band superimposed.

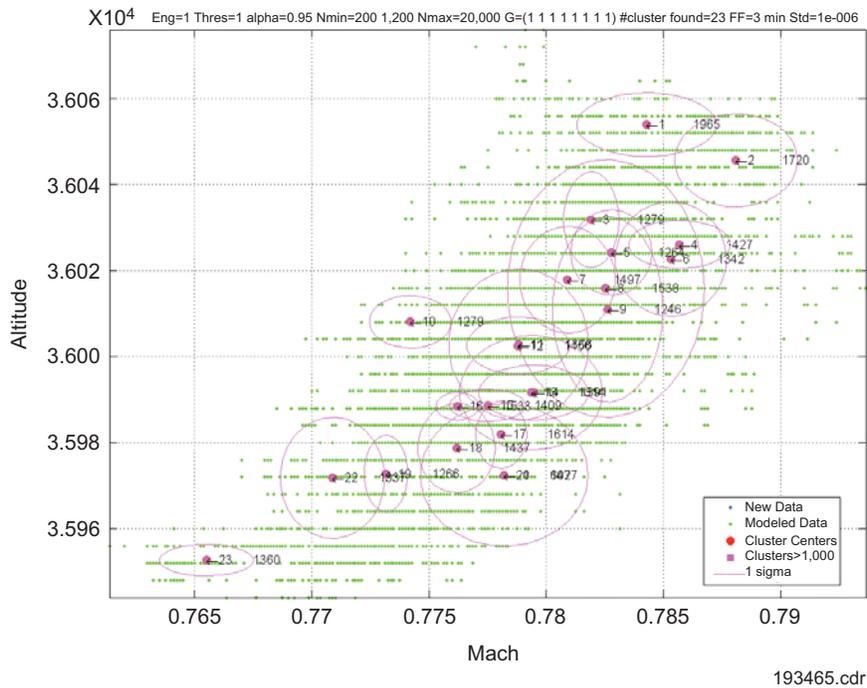


Figure 2-16. Segment 1: GMM Centers

The appearance of the GMM distribution relative to the actual data might appear strange at first glance in that they seem to be clustered together in certain regions and sparse in others. The reason for this is that the GMM centers are multi-dimensional vectors and only altitude and mach are depicted. The (seemingly) dense distribution in altitude and mach is really due to the other six input parameters not shown. In actuality the GMM centers are uniformly dispersed throughout the eight-dimensional input space in Segment 1. The 4,000:1 compression represented in this segment is rather dramatic and it would be natural to question the suitability of the GMM formulation and whether or not training an MLP on 23 GMMs (in the second stage of the process) would yield essentially the same result as training on the original 93,395 data points directly. The efficacy of the GMM compression can be tested by running the Segment 1 data back through the hybrid model system depicted in *Figure 1-6*, where the MLP has been trained using the 23 GMMs. The effect on the residuals and the tuners are depicted in *Figures 2-17* and *2-18*, respectively.

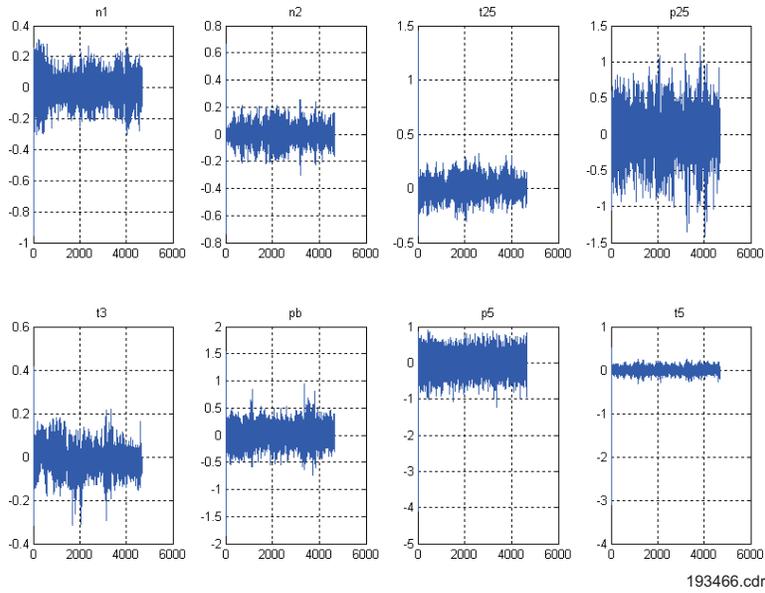


Figure 2-17. Segment 1 Measurement Residuals (Percent Δ) With Hybrid Model Trained With 23 GMMs

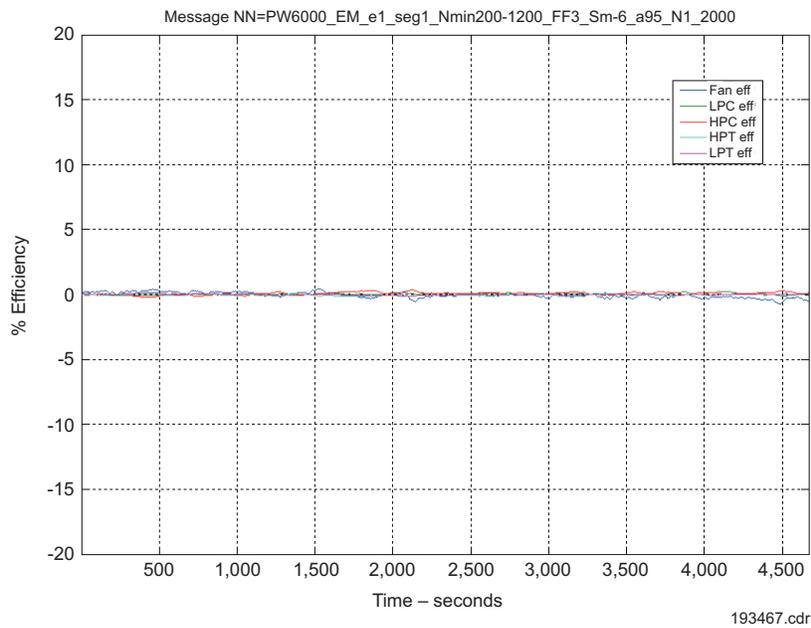


Figure 2-18. Segment 1 STORM Tuners Using Hybrid Model Trained With 23 GMMs

Segment 2 (*Figure 2-11*) contains a level flight at 36k feet altitude, followed by a climb to 38k feet and then level flight at that altitude. The input parameters for Segment 2 are depicted in *Figure 2-19*.

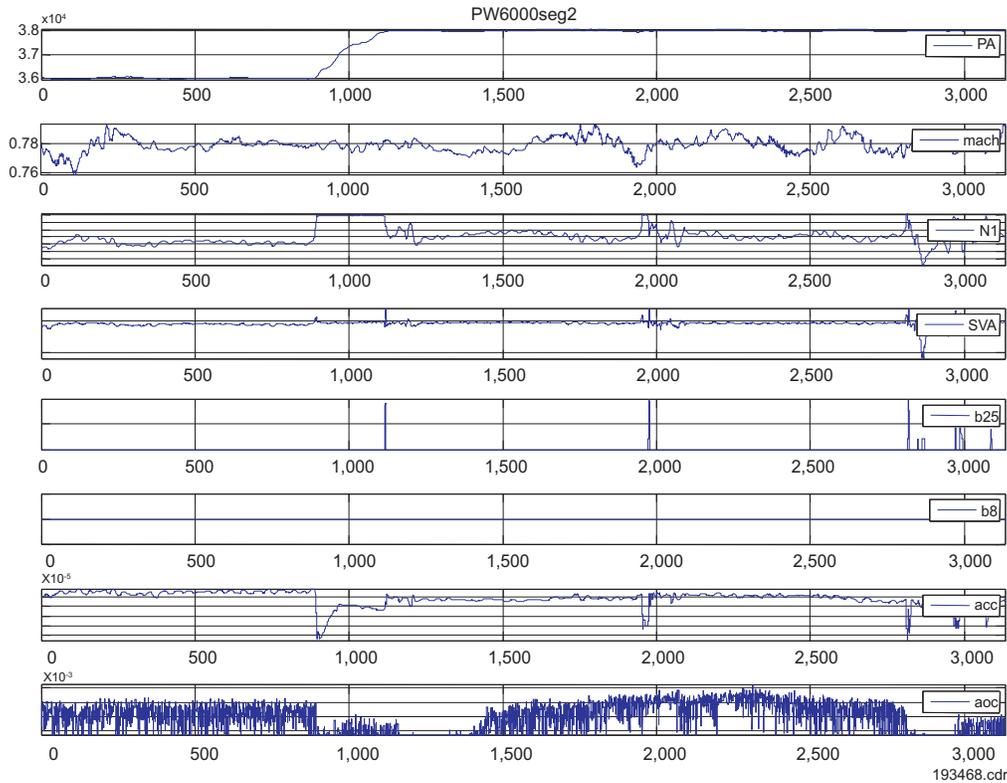


Figure 2-19. Segment 2 Input Parameters

Figure 2-20 depicts the measurement residuals (in percent) between the main gaspath measurements and the model (SVM) estimates. It can be seen from this plot that all parameter residuals have a mean value other than zero (which implies that there is not a good model-engine match).

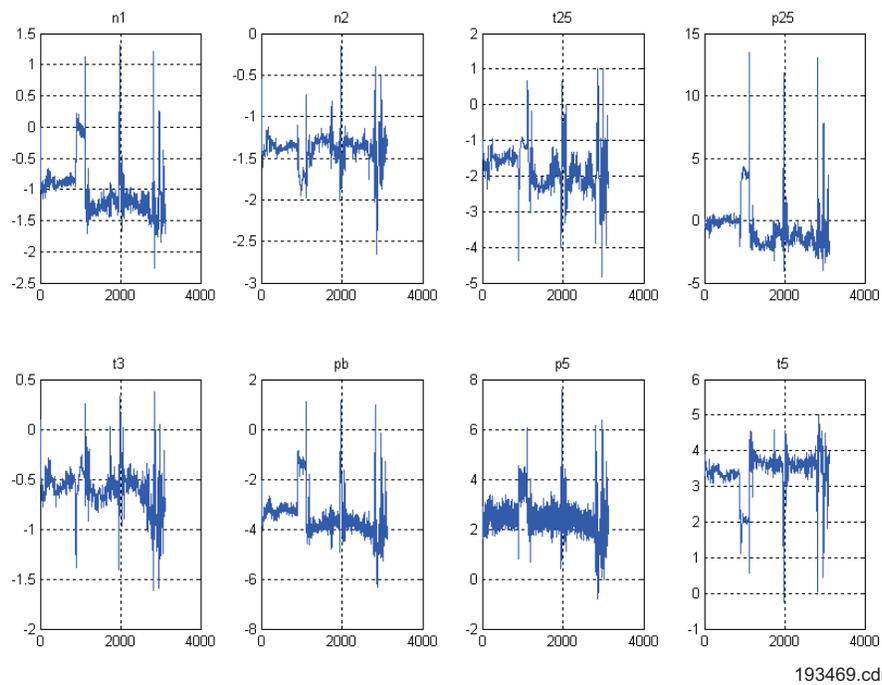


Figure 2-20. Segment 2 Measurement Residual Parameters (Percent Δ) Without KF

If the model was run with the NN enabled at this point (but with the KF disabled), a different set of residuals would be generated. The NN at this stage would be the MLP trained on the 23 GMMs formed during Segment 1 of the flight. The residuals that would be formed would be the difference between the incremental hybrid model (i.e., the SVM plus Segment 1 MLP) and the actual engine. These residuals are depicted in **Figure 2-21**.

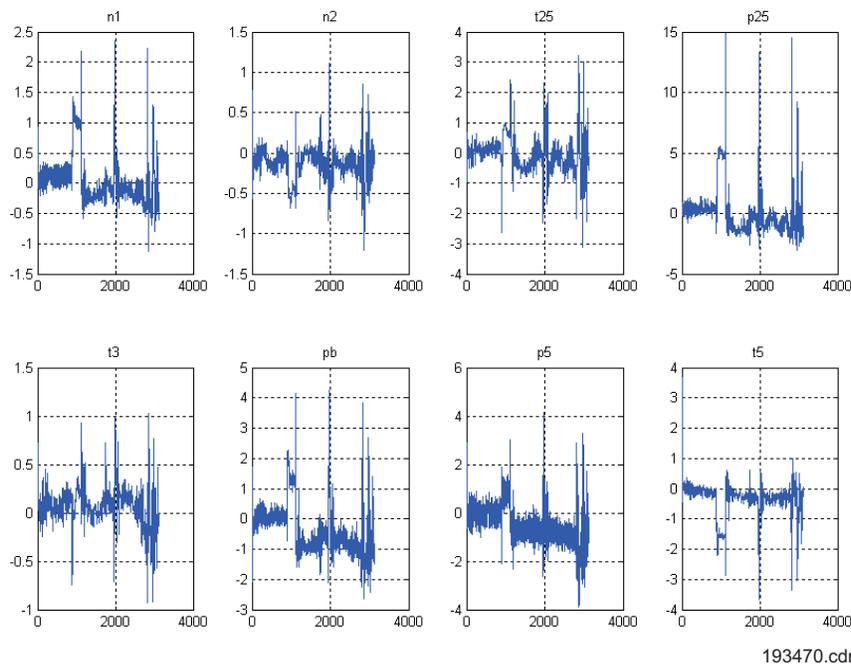


Figure 2-21. Segment 2 Measurement Residual Parameters (Percent Δ) Without KF and Segment 1 MLP Enabled

Since Segment 2 begins at 36k feet altitude, much of that portion of the segment is *covered* by the previous flight training (which was also at 36k feet). Differences in the other seven input parameters could yield differences between this incremental hybrid model and the engine (at 36k feet); however, it can be observed in **Figure 2-21** that the residuals are all approximately zero during the 36k feet portion of Segment 2. The short portion attributed to the climb to 38k feet can be seen to affect all residuals. If we were to engage the KF along with this incremental hybrid model we would produce the following set of STORM tuners as depicted in **Figure 2-22**.

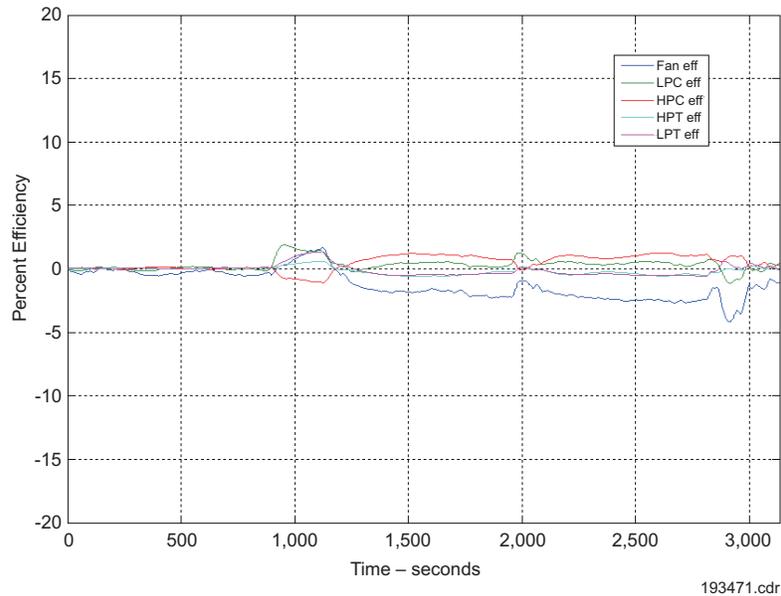


Figure 2-22. Segment 2 STORM Tuners Using Hybrid Model Trained With 23 GMMs

During the Segment 2 flight, 36 additional GMMs were formed, which represent any un-modeled portion of the 36K feet flight, the climb to 38K feet and the stable 38K feet portion. Between Segments 1 and 2, a total of 59 (23 plus 36) GMMs were formed. These are depicted in **Figure 2-23**.

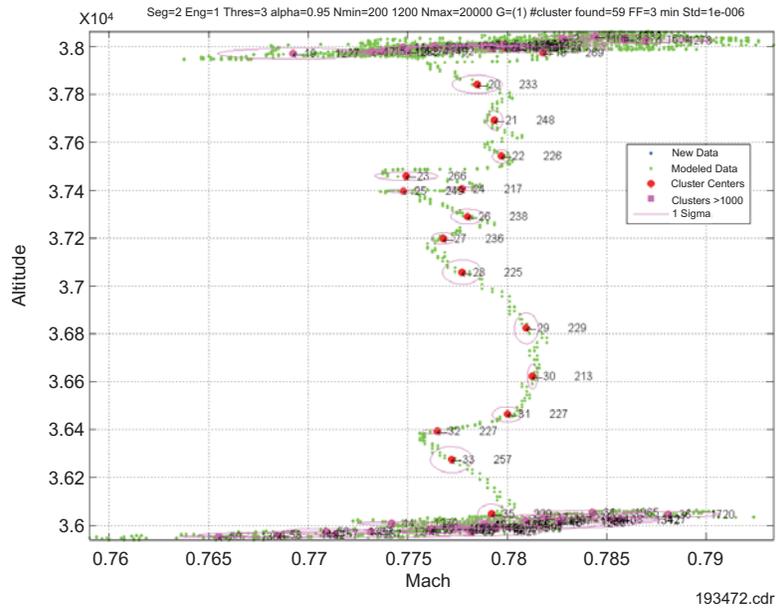


Figure 2-23. Segment 1 and 2: GMM Centers

Training the MLP, in the second stage of the process, using all 59 GMMs, will now produce the next incremental change in the empirical portion of the hybrid model. Processing the Segment 2 data relative to the new model would yield residuals that are now zero on the average as depicted in **Figure 2-24**.

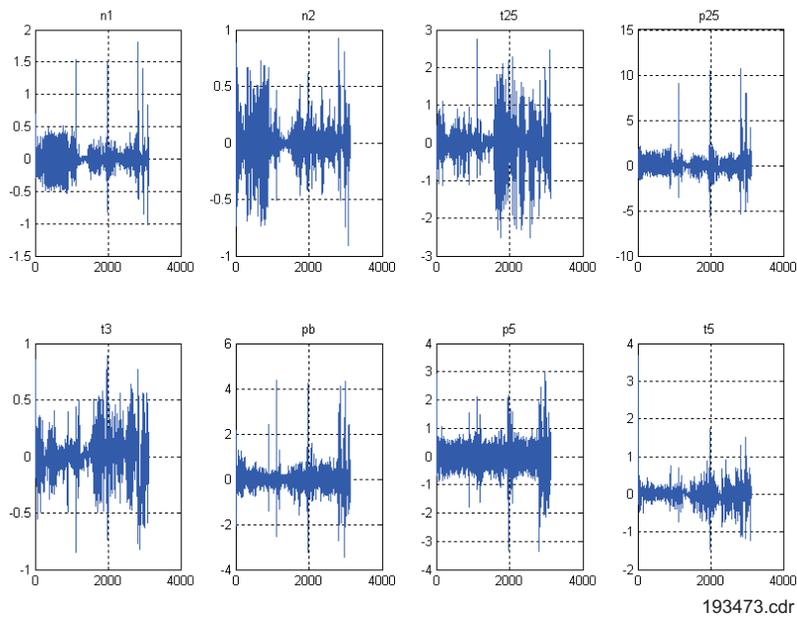


Figure 2-24. Segment 2 Measurement Residuals (Percent Δ) With Hybrid Model Trained With 59 GMMs

The resultant performance Δ s, now appear as depicted in *Figure 2-25*. As can be observed, the performance Δ s are now essentially at a zero level.

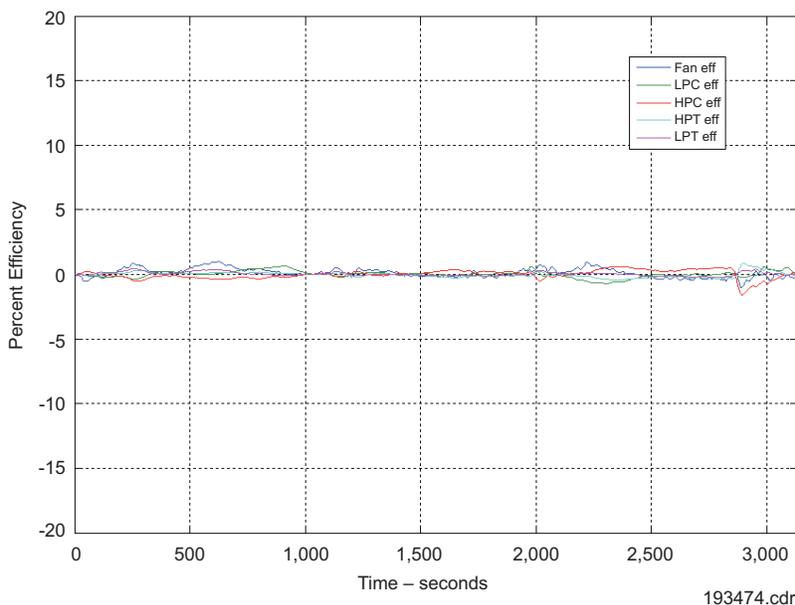


Figure 2-25. Segment 2 STORM Tuners using Hybrid Model Trained With 59 GMMs

This process would continue with every new flight that the vehicle would take. At some point in time, on a cell-by-cell basis, no new GMMs would be found. In which case, the real-time assessment would be to run the hybrid eSTORM model residing in the database. The second stage processing would not be invoked. In essence, the data itself, determines when the training process is complete.

2.5 REAL-TIME CODE DEVELOPMENT

The eSTORM algorithm has been transitioned onto a hardware platform for real-time implementation and demonstration. The hardware platform used was based on the IAC-1209 vibration management enhancement program (VMEP) on-board data collection and processing unit developed for Army helicopters (Section 4, *References, 6 and 7*). The hardware system includes an Ethernet interface for input of eSTORM parameters, data, and output of results. The IAC-1,209 real-time software architecture includes different processing modules as stand-alone data processing units (DPUs). The eSTORM software was developed as a stand-alone DPU that fits within this structure.

The hardware system was originally purchased for IAC’s NASA SBIR where the original real-time demonstration of the eSTORM concept was implemented. To facilitate a cost effective demonstration of the recent advances in the eSTORM algorithm, NASA transferred that hardware system to the current project. The embedded eSTORM DPU was developed directly from the Simulink implementation, using tools available from the MathWorks. The hardware system, real-time software architecture, and real-time system demonstration are described below.

Figure 2-26 shows the hardware of the embedded target system. This system is a variation of the IAC-1209 VMEP hardware developed by IAC for U.S. Army applications. The system hardware includes two COTS PC-104 cards that support an ARINC 429 interface card and a CPU/Ethernet card, respectively. The ARINC 429 interface

card can be used for reading data from the aircraft's ARINC 429 bus. In the eSTORM SBIR Phase II Demonstration, P&W used this card to process C-17 data for an embedded eSTORM algorithm. In this program, P&W did not use that bus; instead, data was input to the eSTORM algorithm from RAM disk (DiskOnChip).

The CPU card forms the computation platform within the IAC-1209 box. This card includes an Ethernet port for real-time output of eSTORM results and to allow the user to simulate component degradations. The computational core is supplied by a 233 MHz Pentium II compatible processor. In terms of computational speed, this box's throughput is comparable to a high end 1997 desktop PC. The inclusion of an Ethernet interface is a departure from the standard VMEP configuration. However, for the demonstration on this project, the Ethernet interface allows us to set processing parameters, input PW6000 data, and display eSTORM processing results immediately.



Temperature Range: -40 through 55° C
Input Power: 6-40VDC, MIL-STD-704D
Power: <18 W
Dimensions: 3 in. × 5 in. × 9 in. without mounting ears
Weight: 4 lbs 5 oz.
Mounting: via mounting plate
Interconnection: MIL-STD 38999 connectors
Remote front panel interface
Processor: Pentium compatible 233 MHz
SDRAM: 128 Mbyte
Disk on Chip: 96 Mbyte
Compatibility: PC/AT compatibility
Communication Channels: Ethernet, ARINC 429

193475.cdr

Figure 2-26. Hardware System Summary

For embedded software development on the target platform, we used a VxWorks development station shown in **Figure 2-27**. The development station has the stack of PC-104 cards and interfaces that allow easy access to both the hardware and software components of the system. VxWorks supplies debugging tools to facilitate the embedded system development process. After the embedded system has been verified and validated on the development station, it is then transitioned to the target platform.



193476.cdr

Figure 2-27. Development Station

The real-time eSTORM software runs in the VMEP real-time system environment (Section 4, *References, 6 and 7*). To fit within that architecture, the entire eSTORM algorithm was implemented as a stand-alone DPU. **Figure 2-28** shows the real-time embedded system architecture with the eSTORM DPU as the lone DPU executing in the system. For the final demonstration on this project, the ARINC-429 interface was not used. Instead, the eSTORM input data was obtained through the Ethernet interface. Management of the input data is handled by the Measurement Executive. The Measurement Executive then schedules the eSTORM DPU to run after all of its inputs have been read. Outputs from the eSTORM DPU are sent to the File Manager and are available to the user via the Comms module, which transmits this information back out over Ethernet to a PC client.

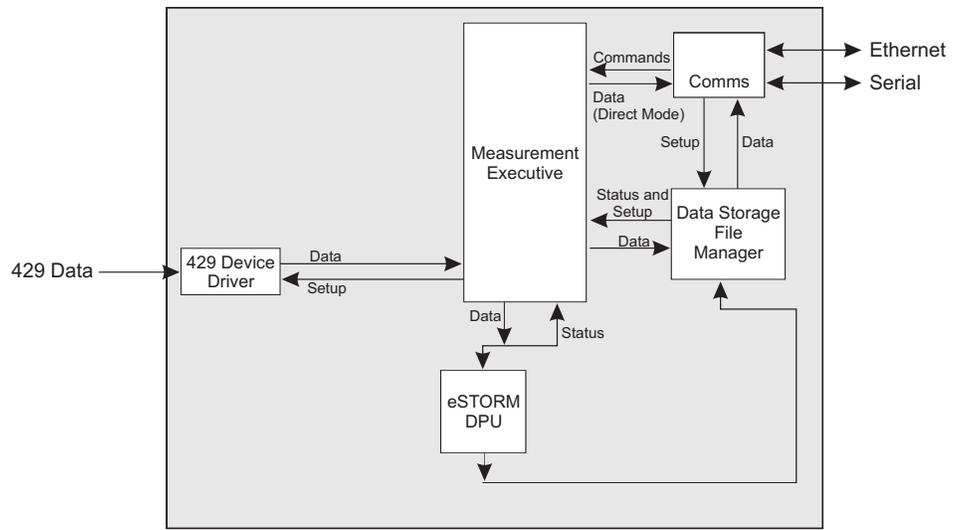


Figure 2-28. Operating System Architecture

193477.cdr

Management of the input data is handled by the Measurement Executive. The Measurement Executive then schedules the eSTORM DPU to run after all of its inputs have been read. Outputs from the eSTORM DPU are sent to the File Manager and are available to the user via the Comms module, which transmits this information back out over Ethernet to a PC client.

Transition of the eSTORM algorithm written in terms of Simulink block diagrams to C-code embedded in the real-time target system was performed using tools available from The MathWorks. More specifically, the Real Time Workshop (RTW) with the Embedded Coder extension was used to automatically generate production quality C source code from the Simulink block diagrams. **Figure 2-29** shows a high level flow diagram for the steps required in performing that transition. Extensive use of the Target Language Compiler (TLC) to inline all five of the S-functions was used in the eSTORM Simulink implementation. In-lining Simulink S-functions is a necessary step for real-time code generation if the advantages of the RTW Embedded Coder are to be realized. By using the Embedded Coder extension of RTW, all of the overhead code that mimics the Simulink simulation environment is removed in auto-generated code. Hence, the eSTORM code produced for the real-time demonstration has a reduced memory footprint and an accelerated execution time relative to that produced by the conventional RTW auto-code generator.

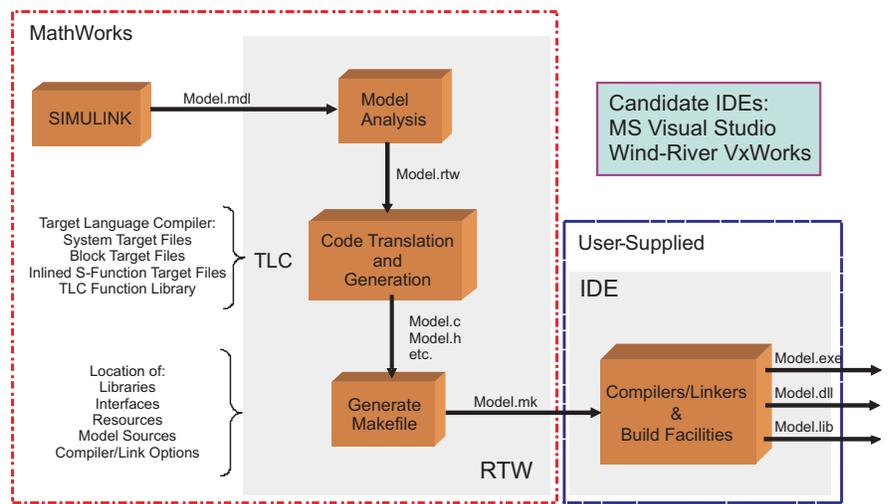


Figure 2-29. Simulink to Embedded System Development

193478.cdr

Figure 2-30 shows the overall processing flow for the real-time demonstration. In a previous demonstration, the ARINC-429 interface was used to simulate input of aircraft bus data to the real-time system. Since P&W has already demonstrated that the ARINC-429 interface on the real-time system works, for this demonstration we elected to process the data as fast as the real-time system could run, as a matter of convenience.

PW6000 data collected from flight test was read from the demonstration computer disk. Rather than play that data back in real-time, an entire segment of data was loaded to the disk-on-a-chip memory within the real-time system via the Ethernet connection. The demonstration computer then commands the real-time system to run. The real-time system then executes the eSTORM code as fast as it can to process that data. Data is read from the disk-on-a-chip memory, processed, and the results transmitted to the demonstration computer via the Ethernet connection as they become available. Those results are then displayed on the demonstration computer.

The C source code developed with RTW was compiled into a run-time module in the Microsoft Visual Studio Integrated Design Environment (VS IDE). The executable module produced by VS IDE required only 136 kilobytes of memory on a PC laptop running the Windows XP operating system. That PC had a 1.2 GHz Pentium III processor and 512 megabytes of RAM. The code ran approximately 120 times real-time on the laptop. Note that the eSTORM execution time estimate was not aided by assigning the application a high priority within the Windows operating system.

With the 233 MHz Pentium II processor in the IAC-1209, P&W expected the code to run about 5 times slower than on the 1.2 GHz machine; (i.e., about 24 times real-time). However, P&W found that the actual code cycle time on the IAC-1209 was about 4 times real time. This timing result is most likely due to the disk-on-a-chip and ISA busses employed by the IAC-1209 box. The ISA bus is particularly slow when compared to the PCI bus used by the PC.

This will not be an issue in future applications with the latest version of the IAC real-time hardware system (the IAC-1134). That hardware includes a Xilinx Virtex-IV FPGA processor. Embedded in that chip are two PowerPCs each rated at 450 MHz. In addition the IAC-1134 will include PCI bus structure for passing data between memory and processing elements.

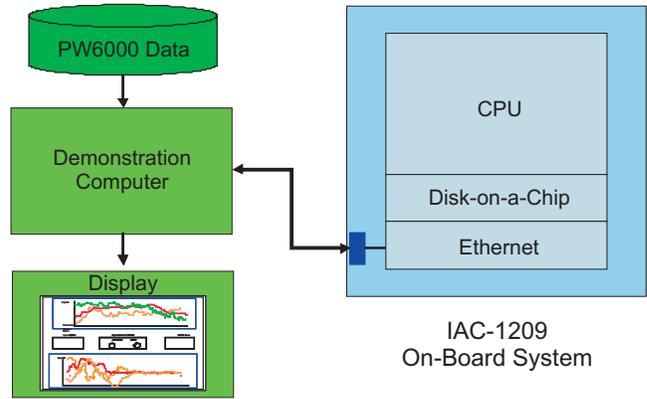


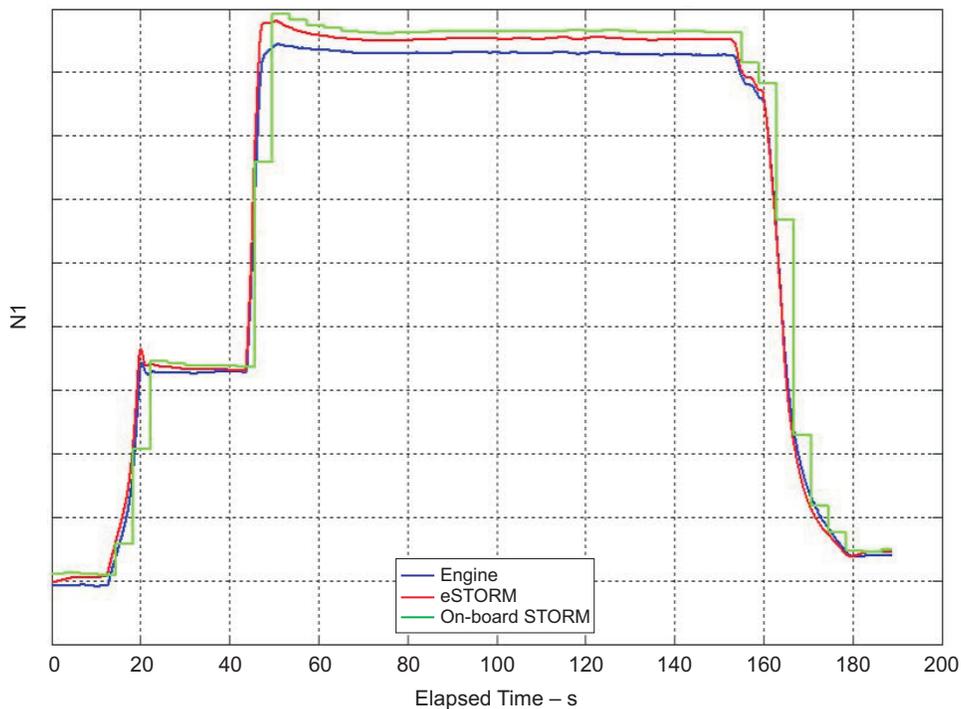
Figure 2-30. *Demonstration Software to Hardware Mapping*

193479.cdr

3. SUMMARY

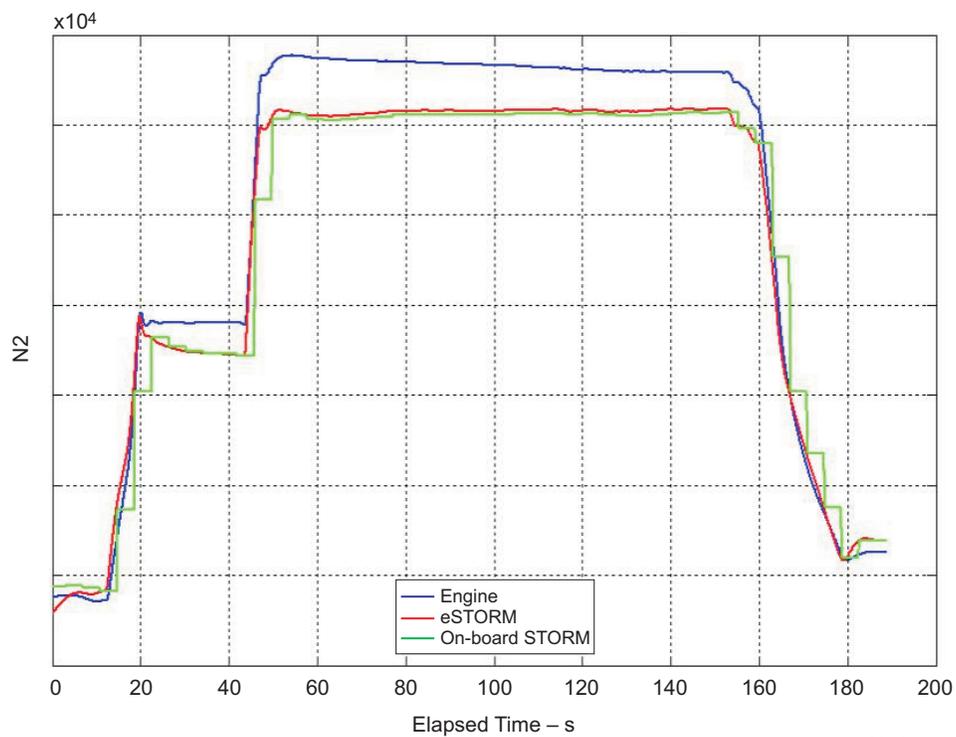
The concept of a hybrid engine model has been introduced and discussed. A pivotal requirement for successful implementation of such systems (containing empirical model elements) is the ability to perform the empirical modeling task in a manner that imposes little additional burden in terms of data manipulation, infrastructure, memory, and computation. A two-stage on-board empirical modeling strategy as been presented in this report that supports autonomous real-time model derivation and subsequent model operation. Hybrid models show great promise in providing increased accuracy for parameter synthesis and module performance tracking.

APPENDIX A



193480.cdr

Figure A-1. PW6000 Block 4: Measurement Comparison — N1



193481.cdr

Figure A-2. PW6000 Block 4: Measurement Comparison — N2

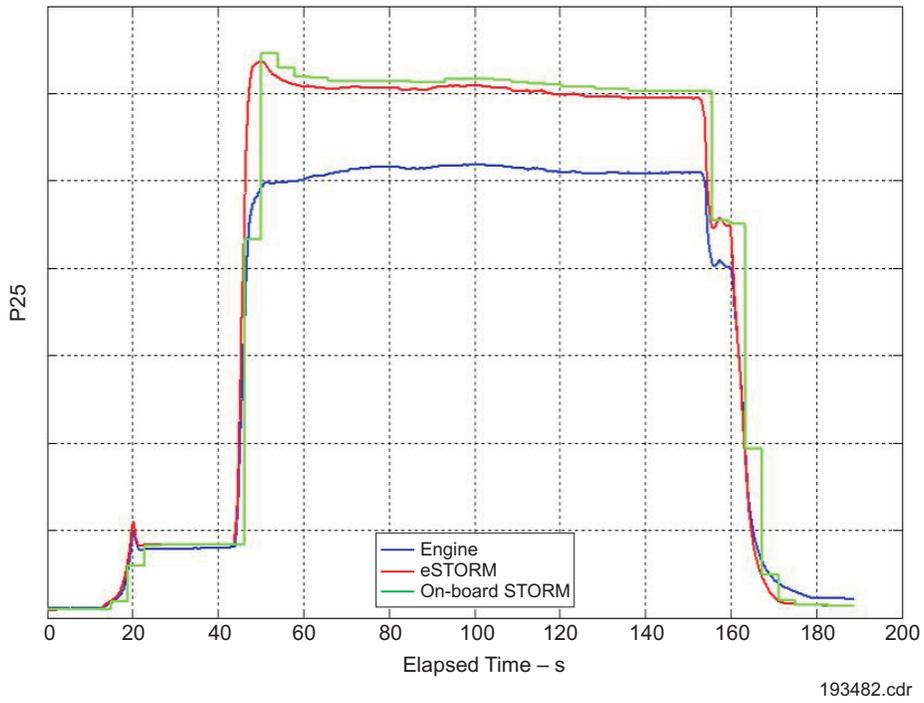


Figure A-3. PW6000 Block 4: Measurement Comparison — P25

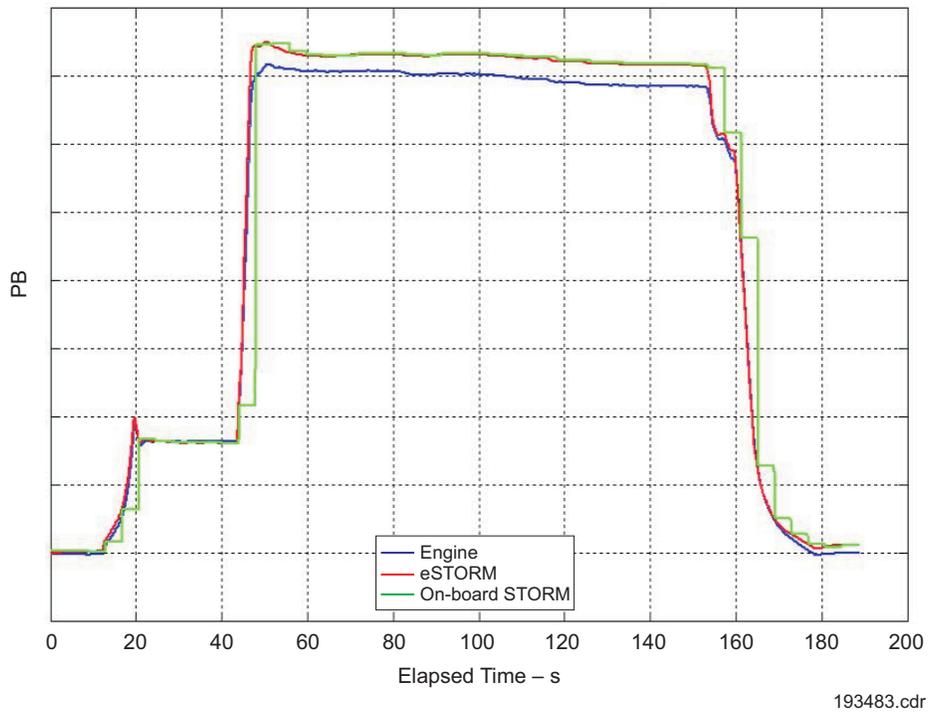
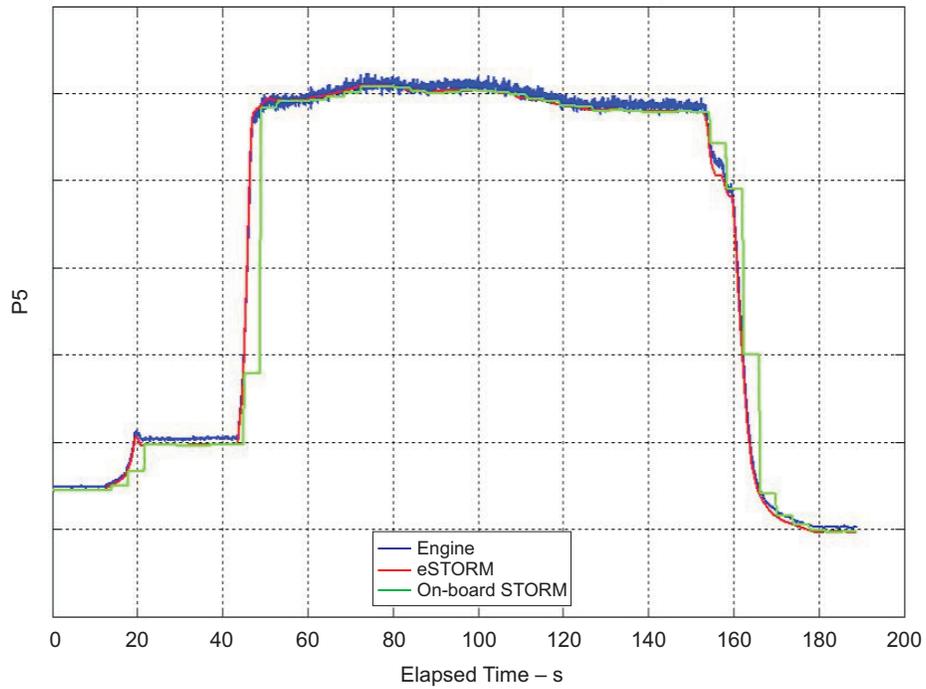
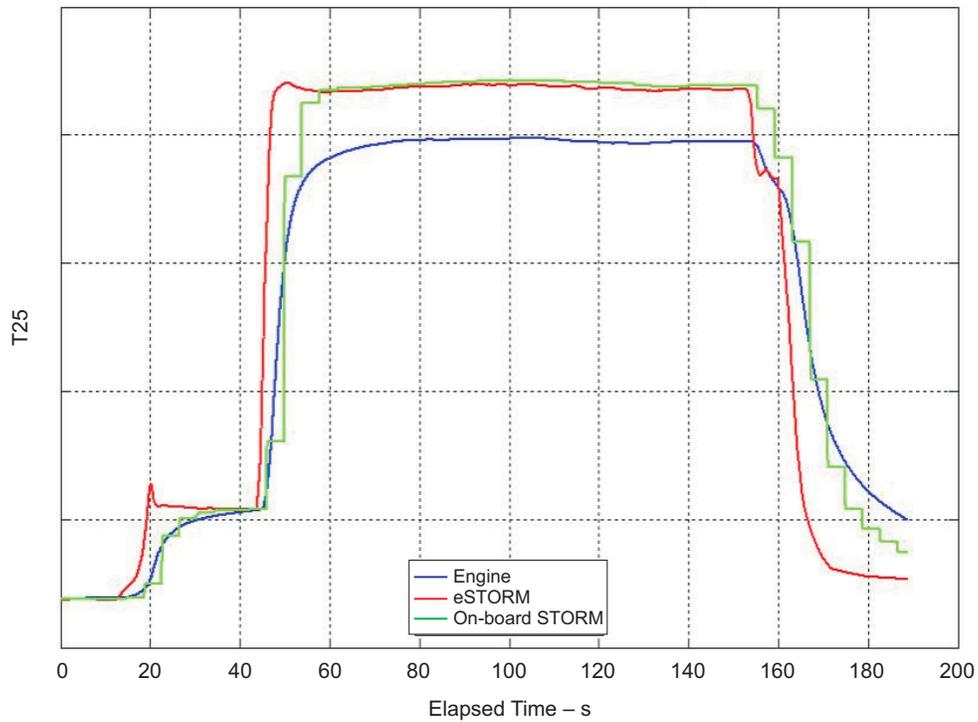


Figure A-4. PW6000 Block 4: Measurement Comparison — Pb



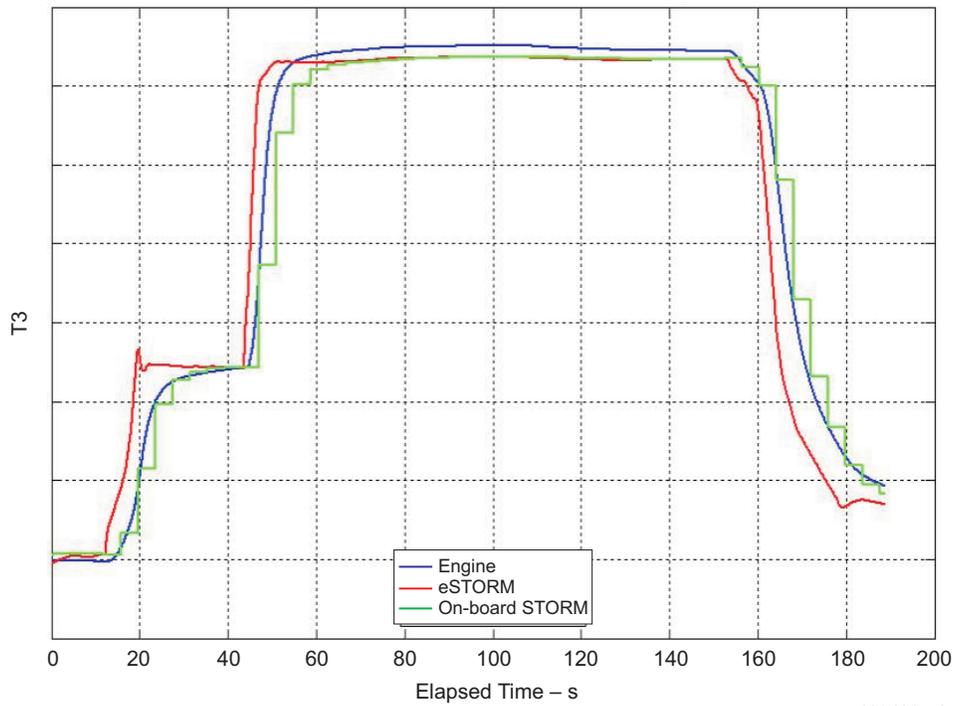
193484.cdr

Figure A-5. PW6000 Block 4: Measurement Comparison — P5



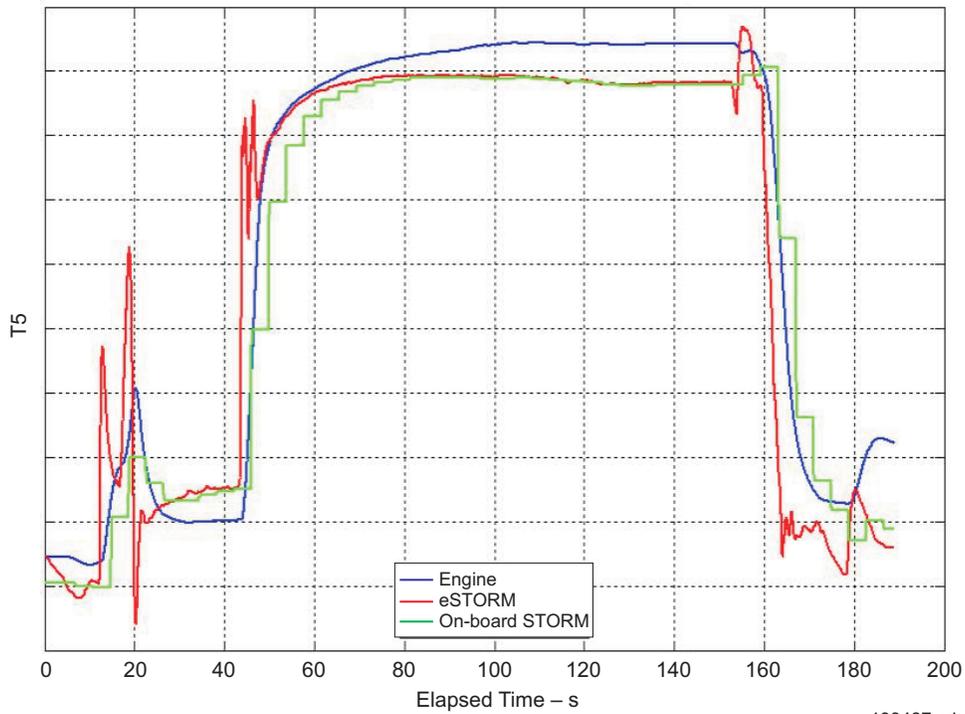
193485.cdr

Figure A-6. PW6000 Block 4: Measurement Comparison — T25



193486.cdr

Figure A-7. PW6000 Block 4: Measurement Comparison — T3



193487.cdr

Figure A-8. PW6000 Block 4: Measurement Comparison — T5

A.1 ACCOMMODATING THERMOCOUPLE DYNAMICS

The PW6000 instrumentation suite contains three temperature measurements, i.e., $T_{2.5}$, T_3 , and T_5 (EGT). The sensor transducers used to make these measurements are thermocouples. The dynamic response of these measuring devices can be characterized with a first order lag system whose transfer function is given by:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{\tau s + 1} \quad (1)$$

where τ is the so-called dominant sensor time-constant, s is the Laplace Transform variable, $u(\cdot)$ is the sensor input, and $y(\cdot)$ is the sensor output. The engine's rotor speed and pressure transducers could also be characterized in the same manner, but the dominant time constants of these transfer functions are significantly greater than the open-loop engine dynamics (i.e., their transfer functions can be approximated as unity in the frequency range associated with the PW6000 SVM). The thermocouple time-constants, however, are well within the bandwidth of the SVM and should be addressed in the design of the eSTORM KF to minimize transient estimation error when the system is driven with real engine data. Extensive empirical studies at P&W have shown that the PW6000 thermocouple time-constants vary with engine power level (i.e., the time-constants are larger [sensor response is more sluggish] at low power and become smaller at the higher engine power levels). To reflect this observed phenomenon, the thermocouple time constants for each temperature sensor are tabulated as a function of uncorrected burner pressure over the nominal engine operating line.

To augment the dynamic SVM with the thermocouple dynamics, the following approach was used. At a specific engine power condition, the dynamic SVM can be expressed mathematically as:

$$\begin{aligned} \dot{\mathbf{x}}_{Eng}(t) &= \mathbf{A}_{Eng} \mathbf{x}_{Eng}(t) + \mathbf{B}_{Eng} \mathbf{u}_{Eng}(t) \\ \mathbf{y}_{Eng}(t) &= \mathbf{C}_{Eng} \mathbf{x}_{Eng}(t) + \mathbf{D}_{Eng} \mathbf{u}_{Eng}(t) \end{aligned} \quad (2)$$

In this particular state-space realization of the engine model, the state vector, $x_{Eng}(t)$, contains the high and low rotor speeds and four engine core metal temperatures. The engine input vector, $u_{Eng}(t)$, contains control system commands, installation effects, and incremental variations in gas path rotating machinery efficiencies and airflow capacities. The engine output vector, $y_{Eng}(t)$, contained in the PW6000 measurement suite is composed of eight elements: two rotor speeds, three pressures, and three temperatures. For the purposes of this discussion, let's assume that the three metal temperatures are located in the last three elements of y_{Eng} .

A state-space realization of the scalar transfer function (single-input, single output) given by Equation 1 can be written in a manner similar to the notation used in Equation 2. That is:

$$\begin{aligned} \dot{x}_{TC}(t) &= -(1/\tau_{TC})x_{TC}(t) + (1/\tau_{TC})u_{TC}(t) \\ y_{TC}(t) &= x_{TC}(t) \end{aligned} \quad (3)$$

where $u_{TC}(t)$ denotes a single element of the engine output vector, $y_{Eng}(t)$, corresponding to a specific measured temperature, e.g., $T_{2.5}$, the thermocouple state vector, $x_{TC}(t)$, is a scalar reflecting the thermal capacity of the transducer, and the output, $y_{TC}(t)$, represents the measured temperature time history. Note, that the individual thermocouple states are not dynamically coupled to each other. Using our definition of the thermocouple input vector and the fact that the x_{TC} states are decoupled from each other, we can express the three thermocouple measurement

system in terms of engine states, thermocouple states, and the engine input vector. To illustrate the validity of that last point, let's integrate the engine model with the thermocouple measurement system which results in an expanded state-space system that merges Equation 2 with the vector (multi-input, multi-output) version of Equation 3 such that:

$$\begin{bmatrix} \dot{\mathbf{x}}_{Eng}(t) \\ \dot{\mathbf{x}}_{TC}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{Eng} & \mathbf{0} \\ \mathbf{A}_{TC} & \mathbf{C}_{Temp} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{Eng}(t) \\ \mathbf{x}_{TC}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{Eng} \\ \mathbf{A}_{TC} \mathbf{D}_{Temp} \end{bmatrix} \mathbf{u}_{Eng}(t)$$

$$\begin{bmatrix} \mathbf{y}_{Fast}(t) \\ \mathbf{y}_{Sen}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{Fast} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{Eng}(t) \\ \mathbf{x}_{TC}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{D}_{Fast} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_{Eng}(t)$$

$$\mathbf{C}_{Eng} = \begin{bmatrix} \mathbf{C}_{Fast} \\ \mathbf{C}_{Temp} \end{bmatrix} \quad \mathbf{D}_{eng} = \begin{bmatrix} \mathbf{D}_{Fast} \\ \mathbf{D}_{Temp} \end{bmatrix} \quad \mathbf{A}_{TC} = \begin{bmatrix} 1/\tau_{T_{2.5}} & 0 & 0 \\ 0 & 1/\tau_{T_3} & 0 \\ 0 & 0 & 1/\tau_{T_5} \end{bmatrix} \quad (4)$$

Note that all of the matrices must have compatible dimensions to perform the pre- and post-multiplication operations represented in Equation 4. With the engine/sensor dynamics expressed in the form of Equation 4, the task of writing a MATLAB script or function for designing a KF that includes all significant dynamical effects present in the model is quite simple. The current baseline PW6000 Kalman design tool was updated so that the user could performed the design procedure with or without the thermocouple dynamics. However, I highly recommend using the dynamic temperature sensor model if the time-constant data is available and the real-time software application possesses the computational resources to support the increased number of arithmetic operations imposed by the expanded system.

After designing the new filter, the Simulink code that implements the PW6000 SVM was updated to support the thermocouple measurement model. These coding changes reflect the variations of the thermocouple time constants with engine power level (table lookups based on uncorrected PB), and the implementation of the state equation augmentations specified by Equation 4. Care was taken to compartmentalize (modularize) these changes so that switching back to the SVM given by Equation 2 can be effected through user input to a simulation setup script and Configurable Subsystem blocks embedded within the eSTORM Simulink block diagram. The employment of a data-driven specification of the PW6000 eSTORM Simulink block diagram closely mimics the techniques developed in the F117 eSTORM work done on the NASA PHM and Data Fusion programs. This approach provides the user with large degrees-of-freedom in changing the baseline configuration and avoids the proliferation of spin-off models to accommodate these off-nominal configuration changes.

4. REFERENCES

1. Berry, J., Branhof, B., Brotherton, T., Grabill, P., and Grant, L., *Rotor Smoothing and Vibration Monitoring Results for the US Army VMEP*, American Helicopter Society 59th Annual Forum, Phoenix, AZ, May 6-8, 2003.
2. Berry, J., Brotherton, T., Grabill, P., and Grant, L., *The US Army and National Guard Vibration Management Enhancement Program (VMEP): Data Analysis and Statistical Results*, American Helicopter Society 58th Annual Forum, Montreal, Canada, June 10-12, 2002.
3. Brotherton, T., Volponi, A., Luppold, R., Simon, D., *eSTORM: Enhanced Self Tuning On-board Real-time Engine Model*, 2003 IEEE Aerospace Conference Proceedings, Big Sky, MT, March 2003.
4. Luppold, R.H., et.al., 1989, *Estimating In-Flight Engine performance Variations Using Kalman Filter Concepts*, AIAA Paper 89-2584.
5. Volponi, A., 2005, *Use of Hybrid Engine Modeling for On-Board Module Performance Tracking*, ASME GT2005-68169, IGTI Turbo Expo 2005, Reno, NV, June 2005.
6. Volponi, A., *The Use of Kalman Filter and Neural Network Methodologies in Gas Turbine Performance Diagnostics: A Comparative Study*, ASME 2000-GT-547, IGTI Turbo Expo 2000; Munich; accepted for Transactions of the ASME, Journal of Engineering for Gas Turbines and Power.
7. Volponi, A., Brotherton, T., 2005, *A Bootstrap Data Methodology for Sequential Hybrid Engine Model Building*, 2005 IEEE Aerospace Conference Proceedings, Big Sky, MT, March 2005.

REPORT DOCUMENTATION PAGE

*Form Approved
OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 01-07-2008		2. REPORT TYPE Final Contractor Report		3. DATES COVERED (From - To) October 2004-September 2006	
4. TITLE AND SUBTITLE Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking				5a. CONTRACT NUMBER NAS3-01138	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Volponi, Al				5d. PROJECT NUMBER	
				5e. TASK NUMBER 26	
				5f. WORK UNIT NUMBER WBS 645846.02.07.03.03.01	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Pratt & Whitney 400 Main Street East Hartford, Connecticut 06108				8. PERFORMING ORGANIZATION REPORT NUMBER E-16543	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001				10. SPONSORING/MONITORS ACRONYM(S) NASA	
				11. SPONSORING/MONITORING REPORT NUMBER NASA/CR-2008-215272; FR-26751	
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, 301-621-0390					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT A key technological concept for producing reliable engine diagnostics and prognostics exploits the benefits of fusing sensor data, information, and/or processing algorithms. This report describes the development of a hybrid engine model for a propulsion gas turbine engine, which is the result of fusing two diverse modeling methodologies: a physics-based model approach and an empirical model approach. The report describes the process and methods involved in deriving and implementing a hybrid model configuration for a commercial turbofan engine. Among the intended uses for such a model is to enable real-time, on-board tracking of engine module performance changes and engine parameter synthesis for fault detection and accommodation.					
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 56	19a. NAME OF RESPONSIBLE PERSON STI Help Desk (email:help@sti.nasa.gov)
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (include area code) 301-621-0390

