ISABE 97-7144
Development of a Near Real-Time Turbine Engine Testing Diagnostic System Using Feature Extraction Algorithms

Donald J. Malloy
Sverdrup Technology, Inc., AEDC Group
Arnold Engineering Development Center
Arnold Air Force Base, Tennessee 37389

Csaba Biegl
Vanderbilt University
Department of Electrical Engineering
Nashville, TN

June F. Zakrajsek
NASA Lewis Research Center
Cleveland, OH

Claudia M. Meyer
NYMA Inc.
Brook Park, OH

Christopher E. Fulton
Analex Inc.
Brook Park, OH

XIII INTERNATIONAL SYMPOSIUM
ON
AIR BREATHING ENGINES

September 7 -12, 1977
Chattanooga, Tennessee

DEVELOPMENT OF A NEAR REAL-TIME TURBINE ENGINE TESTING DIAGNOSTIC SYSTEM USING FEATURE EXTRACTION ALGORITHMS*

Donald J. Malloy
Sverdrup Technology, Inc., AEDC Group
Arnold Engineering Development Center
Arnold AFB, TN

Csaba Biegl
Vanderbilt University
Department of Electrical Engineering
Nashville, TN

June F. Zakrajsek
NASA Lewis Research Center
Cleveland, OH

Claudia M. Meyer
NYMA Inc.
Brook Park, OH

Christopher E. Fulton
Analex Inc.
Brook Park, OH

ABSTRACT

The Arnold Engineering Development Center and the NASA Lewis Research Center are collaborating on a system which will provide automated data validation and fault identification during developmental turbine engine testing. The system is based on a set of feature extraction algorithms developed at the NASA Lewis Research Center to aid in reusable and expendable launch vehicle data analysis. With minor enhancements, these algorithms have been successfully used to detect both scheduled and unscheduled events in online turbine engine test data acquired at the Arnold Engineering Development Center. The algorithms enable automation of many of the data validation and fault identification procedures typically performed by data analysts, thereby greatly reducing the labor-intensive manual inspection required. The system operates in near real-time on a parallel distributed computer network consisting of five workstations. The approach is shown to be successful in detecting and identifying sensor anomalies and engine component events in a timely manner.

INTRODUCTION

The Arnold Engineering Development Center (AEDC) and the NASA Lewis Research Center (NASA LeRC) are working together to develop, validate, and implement condition monitoring technologies and real-time computing techniques for test articles and facilities. Condition monitoring technologies maximize the return from Research, Development, Test and Evaluation (RDT&E) investments, minimize the more expensive and often hazardous flight test phase of systems acquisition, and save time and resources in the overall development, acquisition, and deployment process. AEDC is the largest and most advanced complex of flight simulation facilities in the world with some 53 aerodynamic and propulsion wind tunnels, rocket and turbine engine test cells, space environmental chambers, arc heaters, ballistic ranges and other specialized units (Fig. 1). NASA LeRC defines and develops advanced technology for high-priority national needs. The work of the Nasa Lewis Research Center is directed toward new propulsion, power, and communications technologies for application to aeronautics and space, so that U.S. leadership in these areas is ensured (Fig. 2).

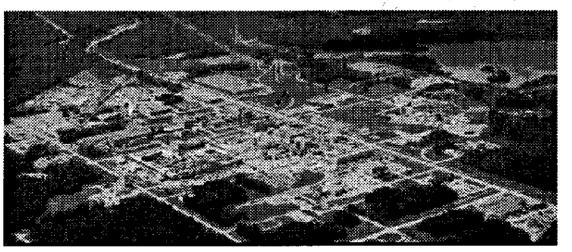

Fig. 1. Arnold Engineering Development Center.

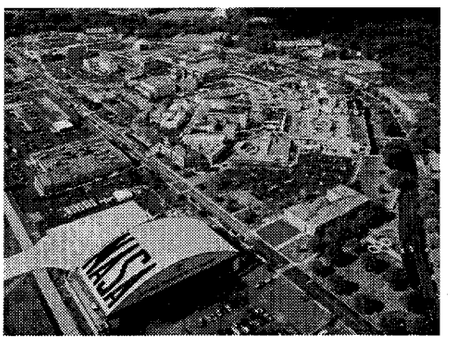

Fig. 2. NASA Lewis Research Center.

* The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Materiel Command. Work and analysis for this research were performed by personnel of Sverdrup Technology, Inc., AEDC Group, technical services contractor for AEDC. Further reproduction is authorized to satisfy needs of the U. S. Government.

This paper is declared a work of the U. S. government and not subject to copyright protection in the United States.

Approved for public release; distribution unlimited.

AEDC performs research to develop new technology for advanced test facilities, test techniques, and measurement methodologies associated with ground testing. During a typical year at AEDC, more than 10,000 hours of test data are acquired in AEDC's flight simulation facilities. With projects like the Pratt & Whitney commercial 4084 engines and the next generation of fighter engines for the F-22 and the Navy F/A-18E/F, more than 2,500 hours of turbine test data are acquired annually in AEDC's eleven engine and aeropropulsion system test facilities. Thousands of individual sensors in the engine and test facility produce an enormous amount of data during developmental turbine engine testing. The static data system alone can acquire more than one million samples of digital data per second using up to 2,950 data channels.*

Because it is not feasible to screen all of the data manually during test operations, automated techniques are required to ensure timely identification of problems that might otherwise go undetected. Improvements in the overall development, acquisition, and deployment process can be realized by modernization of test articles and facilities and the implementation of automated, comprehensive data analysis tools. Providing automated data analysis tools to meet these challenges for developmental turbine engine testing was the motivation for this work.

The techniques described herein are based on a set of feature extraction algorithms developed at NASA LeRC. This work focuses on development and application of technologies capable of near real-time determination of turbine engine propulsion system and test facility sensor and component failures using feature extraction algorithms. Feature extraction enables this automation by identifying both scheduled and unscheduled events in the data stream.

Collaborating partners for development, validation, and standardization of the algorithms (and related real-time test data analysis tools) include AEDC, NASA LeRC, and Vanderbilt University, with support from the Common High Performance Computing Software Support Initiative (CHSSI) component of the High Performance Computing Modernization Program (HPCMP). Timely application of the algorithms for defense-specific problems is facilitated through a tri-service application group consisting of Army, Air Force, and Navy representatives (Fig. 3). An institutional commitment has been made by NASA LeRC, AEDC, and Vanderbilt University (through a formal Memorandum of Understanding) to ensure durability and maintenance of the scaleable software products; and porting and upgrades of the models through evolving hardware, operating systems, language developments, and scaleable architectures.

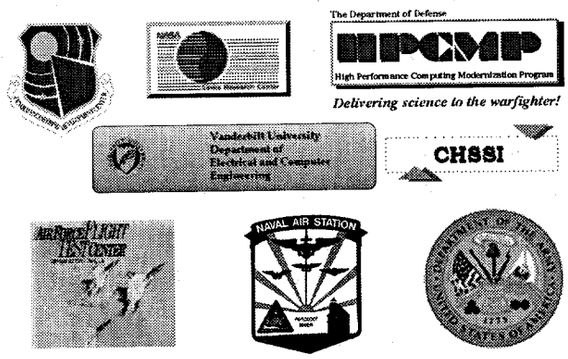

Fig. 3. Collaborating partners.

APPROACH

Test efficiency can be increased and lost test time decreased by providing tools for continuous and comprehensive monitoring of facility and test article data. Near real-time monitoring of test data moves the observation of detectable faults from posttest (after testing has been completed) to pretest and online (prior to and during test operations). This approach minimizes the time required to observe faults, permits correction of anomalies before on-line failures can occur, increases test efficiency, and minimizes lost test time. In addition, labor costs and data transmittal time can be drastically reduced by decreasing the amount of time analysis engineers need to review the data.

In order to meet the requirements for a fast, thorough system, AEDC has undertaken a development program which uses a number of automated analysis tasks to maximize test efficiency.[1-3] Automated analysis tasks include an event detection system, a rule-based expert system with more than 150 checks, and a model-based fault detection and diagnostic system (Fig. 4). Since no single automated analysis technique can provide a thorough analysis of all the data, the results of each analysis task are fused together to provide the engineer with

* The dynamic data system can acquire 144 or more analog measurements at a bandwidth of 32 kHz. A computer-aided dynamic data analysis and monitoring system provides real-time monitoring of dynamic data measurements.

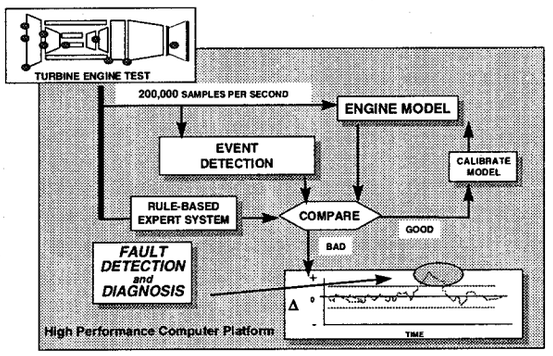

Fig. 4. Online test diagnostic system.

a comprehensive diagnostic capability. As currently configured, the rule-based expert system and the model-based fault detection and diagnostic system analyze a limited number of crucial engine and test facility measurements. In contrast, the event detection system continuously analyzes all engine and test facility measurements. Automated data analysis requires high-performance computing platforms that work together efficiently to support test operations. As a result, a large percentage of the AEDC's computational resources are devoted to support online test activities.

The event detection system is based on feature extraction algorithms which the NASA Lewis Research Center developed and demonstrated as part of a Post-Test Diagnostic System (PTDS) under NASA's Office of Space Flight. The PTDS is an offline aid to engineers who are responsible for detecting and diagnosing data anomalies. The first PTDS application was the Space Shuttle Main Engine (SSME).[4-6] Feature extraction algorithms are at the root of PTDS.[7] The algorithms identify deviations in the data stream during both steady-state and transient operation and provide these events to the rule-based system in PTDS. The algorithms reduce sensor traces into peaks, level shifts, spikes, and drifts, and detect excessive noise, exceedance violations, erratic and flat regions, and check for consistency among redundant channels.

The feature extraction algorithms, as developed for the SSME PTDS and later applied to the Atlas/Centaur propulsion system,[8] are inherently independent of the system to which they are applied. It was proposed in this work to apply these algorithms to test data acquired in propulsion test cells. Specifically, the feature extraction algorithms were applied to 1,000 critical sensors in the engine and test facility sampled at rates between 20 and 500 samples/sec (~100,000 samples of data per second). After completion of this work, the feature extraction algorithms could be applied to the full sensor suite (2,950 sensors) and data rates (one million samples of data per second), and ultimately to test data acquired in wind tunnels, space chambers, and hyperballistic ranges.

Several enhancements were required to transition the feature extraction algorithms from a posttest non-real time to an online, near real-time diagnostic system. In the following sections, a brief description of the algorithms and the associated graphical user interface is provided followed by a summary of the enhancements required to achieve near real-time execution on the Center's existing computers. Computational time is quantified and typical test results are presented.

ALGORITHM DESCRIPTIONS

The feature extraction routines are written in the 'C' programming language and have been incorporated into a real-time graphical user interface. To function efficiently and to minimize demands on analysis engineers, the algorithms are automatically executed as data are acquired. If an event is detected, the analysis engineer is notified of the event via a computer-generated voice message and the event features and data are visually displayed for review. All events are logged and permanently recorded on disk.

The algorithms use statistical information obtained from the time-dependent data stream, along with user-defined tolerances and thresholds, to detect features in the data. To minimize demands on analysis engineers, user-defined tolerances were automatically generated. User-selectable options permit detection of all events, detection of only new events (to exclude failed sensors), and verification of scheduled events. Feature extraction algorithms are applied during both steady-state and transient engine operation. Detailed descriptions of each of the feature extraction routines summarized below can be found in Ref. 7. Sample features are shown in Fig. 5.

DRIFTS

Drifts are general upward or downward trends in the data. In order to detect a drift, the data are first broken into periods of constant linear behavior. The slope of each period is monitored, and a drift is

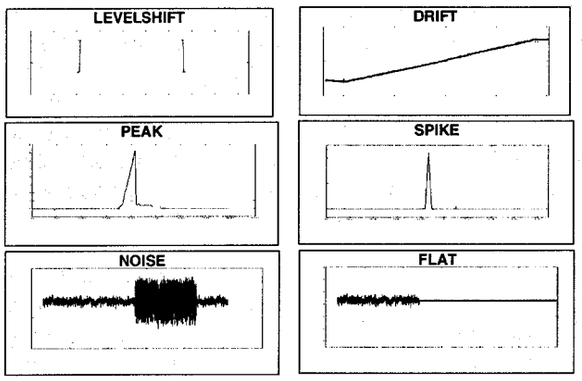

Fig. 5. Data features.

flagged when one or more consecutive periods exhibits a slope that falls outside predefined bounds.

LEVEL SHIFTS

Level shifts are detected by fitting linear equations to windows of data and monitoring the fit parameters for excursions. User-definable parameters are used to constrain the routine to the detection of level shifts that are important to the user.

PEAKS

Peaks are detected by monitoring the slope of successive windows of data for significantly non-zero values. Once a significant slope is detected, it is monitored until it returns to a near-zero state. Peaks which do not meet the minimum duration and height requirements specified by the user are rejected.

SPIKES

The spike routine uses curve fits of the data to look for positive or negative data excursions which occur within three data samples. The standard deviation of the curve fit error is used to identify possible spikes. The maximum time over which a spike can occur and the minimum height are set by the user to exclude insignificant features in the data.

NOISE

Since most sensors exhibit what could be described as "noise," the definition of noise in this application is a larger-than-expected variance in the signal for a prescribed length of time. This routine makes a distinction between noisy and excessively noisy parameters based on the expected variance of the signal.

FLAT REGIONS

In order to detect a flat signal, a curve fit is performed. If the resulting slope falls within predefined limits, the difference between each data point and the fitted line is computed. If the user-defined majority of difference values are less than four times the nominal standard deviation of the signal, a flat signal is declared.

GRAPHICAL USER INTERFACE

A Graphical User Interface (GUI) has been developed to permit efficient setup and operation of the feature extraction algorithms by the analyst. The GUI supports initial setup of the feature extraction algorithms, configuration of the scaleable parallel computing environment, and interactive execution of the feature extraction algorithms (including data visualization, monitoring, and diagnostics).

The main design goals of the GUI were universal applicability and user extensibility. This was achieved by providing a set of core functions for database services and basic plots and an Application Program Interface (API) for implementation of customized functions. The API contains the entry points through which user software can be registered into the user interface; C, C++, and FORTRAN entry points are accommodated. The GUI was written in C++ using common interface elements adapted from public domain toolkits. Most of the custom functions are implemented using user-defined C or FORTRAN software which can be registered into the basic GUI framework.

The GUI interface has been implemented on a variety of platforms including IBM PC's running DOS (full screen) or 32-bit Windows applications (WIN32S, Windows 95®, and Windows NT™), an IBM RS/6000, and several UNIX workstations with the X window environment (Linux, Silicon Graphics, Inc. INDY, Dec Alpha running OSF-1, an IBM RS 6000, and Hewlett Packard). Standard data file scanners provide the capability to scan standardized data file formats. These include binary data files and tabulated data. More customized file formats, such as the Parameter Oriented Data (POD) files used at AEDC, are registered and accessed using the Scanner Library.

NEAR REAL-TIME EXECUTION

Effective use of automated posttest data analysis tools in an online test environment often requires significant improvements in execution speed (often by several orders of magnitude). Distributed workstation networks have been chosen as the target architecture for this work at the AEDC. Benefits of using distributed workstation networks include better availability, increased flexibility (workstations can be used independently by several users during periods when no testing is performed), affordability, and scaleability of distributed architectures. In the following sections, a brief summary of the computational and hardware enhancements required to achieve near real-time execution on the Center's existing computers is presented, and computation time is quantified for a developmental multiprocessor Pentium®-based system and a combination of SGI Challenge® L and INDY® workstations used for online data processing.

COMPUTATIONAL ENHANCEMENTS

Near real-time execution on the Center's existing computational platforms was accomplished by enhancing the operation of the feature extraction algorithms and through scaleable parallel processing. Execution time was reduced significantly by optimizing data transfer to the feature extraction algorithms and by streamlining memory allocations.

"Task level" parallel processing was used to provide a near real-time computing capability. The advantage of explicit parallelization is that, frequently, a much higher level of parallelization can be obtained than with instruction-level parallelizing compilers. Explicit calls to the underlying communication library are made at the points where synchronization and/or data exchange are necessary. A high level of parallelization and load balancing was possible by distributing the data and event detection commands to the various processors. When executed in parallel, one processor is dedicated to the tasks of generating event detection commands, load balancing, distribution of the commands and results to and from the worker processors, and viewing the results. In order to minimize data transfer to the feature extraction algorithms, each processor identifies all feature types for a given sensor.

The parallelized code enables run-time tuning of communication-related parameters, including queue size. To further reduce the communication overhead between the parallel processes, the main program can issue commands to the distributed "worker" tasks to read and process several parameters at once. This allows optimizing the code for different processor speeds and network bandwidths. For a small test program, one processor can perform all tasks. A typical test program with 1,000 measurements sampled at 100 samples/sec (100,000 samples of data per second) requires six processors with four of the six dedicated to execution of the algorithms. For a large test program, twelve or more processors are required to execute the feature extraction algorithms and review the results.

HARDWARE ENHANCEMENTS: PENTIUM® PLATFORM

An eight-processor parallel distributed machine[9] was built to provide an inexpensive development environment. The machine consists of eight 150-MHz Pentium® PC clones interconnected via a high-speed (100 Mb/sec) Ethernet network (Fig. 6). Each node has four network ports: one of these is a standard 10 base-2 (coaxial) port and the other three are high-speed 100 base-T (twisted pair) ports. The coaxial ports are used for "standard" network traffic while the high-speed ports are used for application-level communications. The high-speed ports are connected in a hubless point-to-point scheme to build an eight-node hypercube. To keep things simple, off-the-shelf hardware and software were used. Every processor runs a copy of the Linux® operating system. The parallel machine also includes system management tools (a small collection of shell scripts added to the standard Linux distribution) to facilitate control of the distributed processors. The Message Passing Interface (MPI)[10] and Parallel Virtual

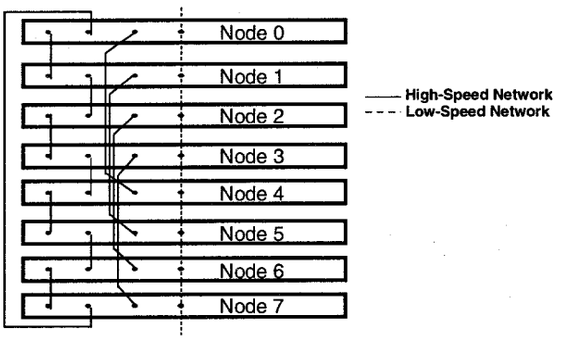

Fig. 6. Schematic of eight-processor parallel distributed machine.

Machine (PVM)[11] have been ported to this environment to provide application-level communication services over the high-speed ports. High-speed user-level network device drivers were developed for the MPI to control the high-speed Ethernet ports without the need to use operating system calls. However, better performance is obtained using a simple TCP/IP based communication protocol which has been optimized for the communication patterns of the parallelized feature detection algorithms.

The feature extraction algorithms processed fewer than 30,000 of the 100,000 samples of data per second acquired using a single 150-MHz Pentium processor (three times slower than real time). The effectiveness of the Pentium® platform and parallelized software was evaluated using a 46-sec test maneuver. Figure 7 shows the average processing speed as a function of the number of worker processors employed. The results convincingly show that the 8-processor Pentium system is fully capable of continuous execution of the feature extraction algorithms. Since a finite amount of time is required to process all measurements, critical measurements are processed prior to noncritical measurements to permit identification of critical sensor failures and engine events with minimal delay. With five or more processors, the analysis can be performed faster than real time.

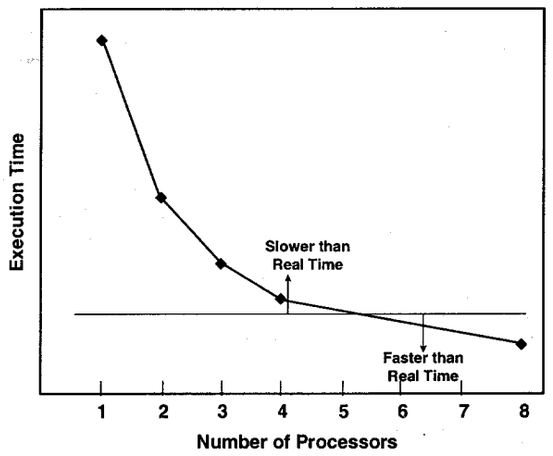

Fig. 7. Pentium® and online platform effectiveness.

HARDWARE ENHANCEMENTS: ONLINE PLATFORM

AEDC test data are distributed over a number of high-speed networks. The low latency "real-time" reflective memory network is used to distribute data with minimal latency. It features a bandwidth > 10 Mb/sec, latency < 10 msec, and no processor overhead. The low latency "real-time" network has not been fully implemented. In the interim, most data analysis is performed in near-real time using a combination of SGI Challenge® L's, M's, and S's, SGI Indy® Workstations, SGI Indigo® II's, Sun Sparc10 Workstations, HP 715/50 Workstations, and PC's running X emulation on the data analysis network. In an online test configuration, an FDDI network is used to transmit data at a bandwidth of 100 Mbps to switching hubs and at a bandwidth of 10 Mbps to each workstation via dedicated Ethernet. For this application, a simple TCP/IP based communication protocol with error checking is used for communication between the workstations.

The feature extraction algorithms processed fewer than 30,000 of the 100,000 samples of data per second acquired using a single 200-MHz SGI Indy® Workstation (three times slower than real time). Once again, the effectiveness of the online analysis network and parallelized software was evaluated using a 46-sec test maneuver. Execution time on the analysis network was nearly identifcal to the Pentium platform (Fig. 7). However, execution time was seen to vary up to 10 percent, depending on network traffic.

RESULTS

The feature extraction algorithms were evaluated using data from two distinctly different modern military turbofan engines undergoing ground test development at simulated altitude test conditions. Measured and calculated data from approximately 1,000 sensors located in the engine and test facility were recorded and analyzed at rates between 20 and 500 samples/sec (~100,000 samples of data per second). Typical events detected during developmental turbine engine testing are discussed in the following sections. For discussion purposes, these events are categorized as either sensor or data acquisition system failures or unplanned or planned engine events.

SENSOR AND DATA ACQUISITION SYSTEM FAILURES

Sensor failures are equally likely to occur during steady-state and transient engine operation. The diagnostic system was successful in detecting sensor failures over a wide range of operating conditions, while maintaining a low false alarm rate.

Most sensor failures occur independently of other events, with the majority of the sensor failures occurring in the harsh environment of the engine.

Timely identification of test facility problems is essential to ensure safe operation of the engine and test facility, and to minimize lost test time. Figure 8 shows two views of a faulty facility cooling water pressure. The y-axis on the upper graph has been rescaled to show the small spikes and peaks that occurred in the data prior to complete loss of the sensor. These spikes and peaks were successfully detected by the feature extraction routines and provided an early warning of the incipient sensor failure.

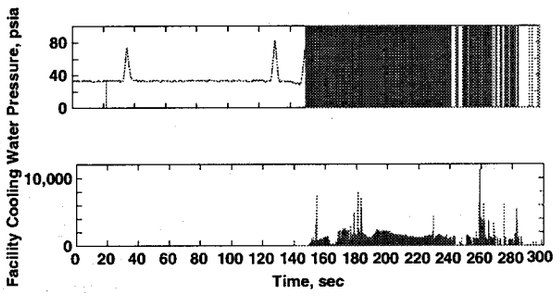

Fig. 8. Cooling water pressure sensor fault.

Of fundamental importance to test operations is the ability to accurately set test conditions during both steady-state and transient test operations. Sensor failures, if undetected, can compromise the safety of test operations and prevent completion of critical test objectives. Figure 9 illustrates an anomalous ambient temperature measurement. Spikes and level shifts were detected early in the test run, prior to the onset of the drift (180 - 250 sec), and permitted use of alternate measurements to verify test conditions.

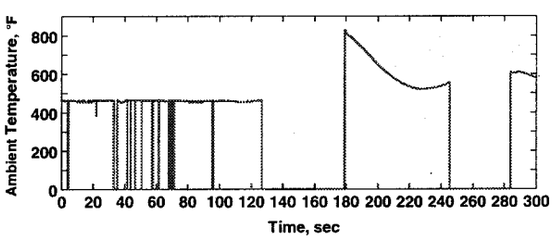

Fig. 9. Ambient temperature fault.

As described previously, all engine and test facility measurements are analyzed by the event detection system. In addition, rule- and model-based analyses are performed for some key measurements. Figure 10 is an example of an invalid fan exit measurement. The event detection, model-based, and rule-based analyses all corroborated the conclusion that this sensor had failed. The model-based system identified the fan exit temperature sensor fault after the measurement level had shifted approximately 1 deg. The event detection system and rule-based expert system identified the sensor fault after the measurement level had shifted approximately 5 deg.

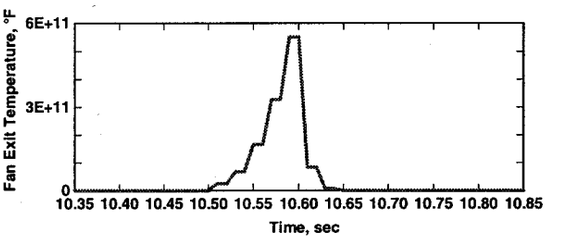

Fig. 10. Fan exit temperature sensor fault.

Figure 11 is an example of a noisy exhaust nozzle liner pressure identified by the event detection system. For this particular test, neither the model-based fault detection and diagnostic system nor the rule-based event detection system was configured to provide a comprehensive analysis of the exhaust liner nozzle pressure. More detailed analysis was required to verify that the indicated pressure variations could be attributed to measurement noise, as opposed to combustion instability.

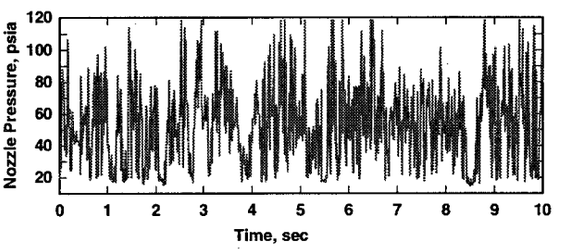

Fig. 11. Nozzle pressure sensor fault.

Events are detected in both measured and computed data. Figure 12 depicts faulty measured engine inlet and ambient pressures in addition to calculated bleed flow rate, gross thrust, engine inlet airflow, and total engine fuel flow. All three analysis modules - event detection, rule-based and model-based - detected an off-nominal condition. However, when multiple anomalies occur simultaneously, the diagnostic capability of the model-based fault detection and diagnostic system and the rule-based event detection system can be

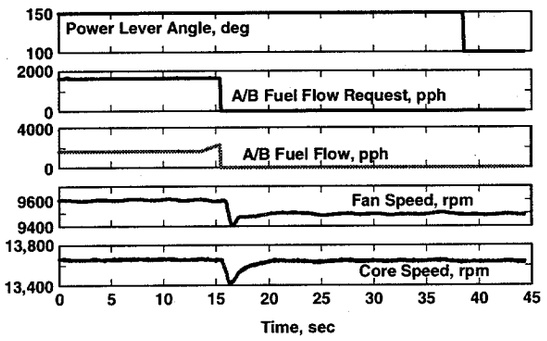

Fig. 15. Augmenttor fuel valve failure.

power. Here, the augmentor fuel flow request remains constant while the actual augmentor fuel flow increases as a result of the forced failure of the augmentor fuel valve. Augmentor fuel flow was cut off by the engine control after the fuel valve failure was detected. The feature extraction routines detected the drift in the augmentor fuel flow, the subsequent shifts in augmentor fuel flow request and fan and core speed shown in Fig. 15, and associated changes in measurements and control variables. Following the event, engine power was reduced to intermediate power. More detailed posttest anlaysis is typically performed for planned events. However, the feature extraction routines help pinpoint the time of the event and provide an initial assessment of system operation and initial insight into system impact.

SUMMARY

The Arnold Engineering Development Center, NASA Lewis Research Center, and collaborating partners have developed a near real-time diagnostic system to provide automated data validation and fault identification during developmental turbine engine testing. With minor enhancements, the feature extraction algorithms developed by the NASA Lewis Research Center have been successfully used to detect both scheduled and unscheduled events in online turbine engine test data acquired at the Arnold Engineering Development Center. The approach is shown to be successful in detecting and identifying sensor faults and engine component events in a timely manner.

The algorithms enable automation of many of the data validation and fault identification procedures typically performed by data analysts, thereby greatly reducing the labor-intensive manual inspection required. The feature extraction algorithms have been applied to 1,000 critical sensors in the engine and test facility sampled at rates between 20 and 500 samples/sec (~100,000 samples of data per second). The system operates in near real-time on a parallel distributed computer network consisting of five workstations. Future efforts will build on the scaleable architecture to accommodate the full sensor suite of nearly 3,000 sensors, higher data rates approaching one million samples of data per second, and the low-latency reflective memory "real-time" data network. The work has demonstrated the flexibility and adaptability of feature extraction techniques for ground-based automated data analysis.

REFERENCES

1. Malloy, D. J., "Improved Data Validation and Quality Assurance in Turbine Engine Test Facilities," AIAA-93-2178, June 1993.

2. Malloy, D. J., Chappell, M. A., and Biegl, C., "Real-Time Fault Identification for Developmental Turbine Engine Testing," ASME Report 97-GT-141, June 1997.

3. Davis, J., Bapty, T., Karsai, G., Sztipanovits, J., Tibbals, T., and Malloy, D., "A Model-Based Data Validation System," Society for Machinery Failure Prevention Technology, 1996 Technology Showcase on Integrated Monitoring, Diagnostics and Failure Prevention.

4. Zakrajsek, June F., "The Development of a Post-Test Diagnostic System for Rocket Engines," AIAA-91-2528, June 1991.

5. Surko, Dr. Pamela and Zakrajsek, June F., "PTDS: Space Shuttle Main Engine Post Test Diagnostic System for Turbopump Condition Monitoring," SAE Aerotech '92, SAE Report 922059, October 1992.

6. Bickmore, T. W. and Maul, W. A., "A Qualitative Approach to Systematic Diagnosis of the SSME," 31st Aerospace Sciences Meeting, AIAA-93-0405, January 1993.

7. Zakrajsek, J. F., Fulton, C. E., and Meyer, C. M., "Feature Extraction for Post-Test Diagnostics," Advanced Earth-to-Orbit Propulsion Technology Conference 1994, May 1994.

8. Erickson, Thomas J., et al., "Post-Test Diagnostic System Feature Extraction Applied to Martin Marietta Atlas/Centaur Data," AIAA-94-3224, June 1994.

9. Biegl, C., et al., "Automated, Real-Time Validation of Turbine Engine Test Data Using Explicit Parallelization on an Eight Processor Pentium® Platform," to be presented at the Thirteenth International Symposium on Airbreathing Engines, September 1997

10. MPI Standard, version 1.1, MPI-2 Forum, June 12, 1994.

11. *PVM: Parallel Virtual Machine -- A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press, Cambridge, 1994.