USC-SIPI REPORT #416

A Signal Processing Approach to Robust Jet Engine Fault Detection and Diagnosis

by

Martin Gawecki

August 2014

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

A SIGNAL PROCESSING APPROACH TO ROBUST JET ENGINE FAULT DETECTION AND DIAGNOSIS

by

Martin Gawecki

A Dissertation Presented to the

FACULTY OF THE USC GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(ELECTRICAL ENGINEERING)

August 2014

Doctoral Committee:

Professor C.-C. Jay Kuo, Chair

Professor Keith Jenkins

Professor Aiichiro Nakano

Abstract

As in any mechanical system, entropy is continually fighting our best efforts to preserve order. Engineers, mechanics, and pilots have all helped in the process of engine health management by perceiving and identifying faults in aircraft. The complexity of these systems has gradually increased, necessitating the evolution of novel methods to detect engine component problems. As airlines and manufacturers have begun to develop capabilities for the collection of ever more information in the age of "Big Data," an opportunity for such a method has presented itself to the signal processing community.

This work will address the development of reliable fault detection and diagnosis algorithms, built around the collection of various types of engine health data. Engine Health Management (EHM), has so far relied on rudimentary readings, the diligence of maintenance crews, and pilot familiarity with expected equipment behavior. While the majority of EHM advances are inexorably tied to the field of mechanical and aerospace engineering, signal processing approaches can make unique contributions in effectively handling the oncoming deluge of complicated data.

During the scope of this work, two broad approaches are taken to address the challenges of such an undertaking. First, the feasibility of vibration and acoustic sensors is examined in controlled experimental conditions to determine if such information is useful. This in turn will be used to develop modern detection/diagnosis algorithms and examine the importance of sampling frequency for EHM systems in this context. Here,

this work offers several important contributions, chief among which are: excellent results for "stationary" phases of flight, a consistent fault detection rate for synthetic abrupt changes, fast responses to component failures in high-frequency data, and well-defined clustering for nominal samples in lower-frequency (1Hz) data.

Second, this work describes an improved Gas Path Analysis (GPA) approach that utilizes information from traditional sensors (pressures, temperatures, speeds, etc.) to produce relevant high-quality simulated data, develop a correspondence between simulated and real-world data, and demonstrate the feasibility of fault detection in these scenarios. Here, the chief contribution is the establishment of a close agreement between synthetically simulated faults and nominal data from real engines. Building on this, a reliable fault detection and diagnosis system for "stationary" and "transient" flight phases is developed, while adapting high quality simulated full flight data to low-frequency (1Hz) real world correspondences.

to my supportive family, to my incredible friends

Acknowledgements

This dissertation is an accumulation of a several years of work, which would not have been possible if not for the support of a large community of family, friends, coworkers, and colleagues.

Dr. C.-C. Jay Kuo has been the single most important driving force in my time at USC. Through his encouragement in joining the Media Communications Lab, help throughout the PhD process, and sage advice I have learned a lot about engineering and life. His weekly "fireside chats" have taught me countless lessons in being a better person, researchers, and contributing member of society; I am eternally in your debt.

I would also like to sincerely thank my colleagues at the USC Media Communications Lab, without whom this journey would not have been as rewarding. I am grateful for their counsel and guidance, their continued insights into research that is not their own, and their tolerance of my endless questions. I would especially like to thank Steve Cho, Jiangyang Zhang, Sanjay Purushotham, Abhijit Bhattacharjee, Joyce Liang, and Sachin Chachada - their stoic actions in the face of adversity have been invaluable examples of doing things right.

I would like to thank my long-time roommate, Syed Ashrafulla, whose excellence as a researcher, a friend, and a lifelong learner have helped shape me into a more inquisitive and insightful human being. I am eternally grateful for the times we have had ping-ponging around LA and growing as PhD students. To my friends in LA and where I grew up - thank you for helping me relax and find ways to enjoy life outside of school.

Much of this work was performed as part of the ongoing Pratt-Whitney Insitute for Collaborative Engineering. Pratt-Whitney donated considerable resources and time in order to assist with work that had been ongoing when I joined older projects and to continue helping with what is discussed within this dissertation. In a special way, I would like to sincerely thank Dr. Allan J. Volponi for his invaluable insight and tutelage during this time. Your humor, and sincerity greatly amplified the knowledge I was able to glean from your years of experience. I would also like to mention the indirect help of Turbine Technology, who donated one of the datasets used in this work.

I must also acknowledge the help and resources I received from NASA, and in particular Don Simon, who assistend in the aquisition and troubleshooting of various simulation software that are utilized for the latter part of this dissertation. I am happy to

have had the opportunity to interact with Don, and hope that my contributions can help improve upon the excellent work being done at NASA, for the good of mankind.

Computation for the work described in this paper was supported by the University of Southern California Center for High-Performance Computing and Communications (www.usc.edu/hpcc).

Lastly, but most importantly, I would like to recognize the support that my family has given me - not only while at USC, but throughout my life. My parents, Anna and Wojtek Gawecki, were there to celebrate my victories and lend a helping hand when I stumbled; I will always remember your sincere love. To my siblings, Peter and Kate, thanks for everything you have done to help me along this journey.

Table of Contents

List of Figures

Chapter 1

Introduction

Since the start of the modern age of commercial flight, safety has been a top priority for manufacturers and carriers. While extremely rare, in-flight disasters still do occur and are sometimes attributed to mechanical failures despite best efforts to mitigate their cause. In the nascent days of commercial flight, prevention of such mechanical faults was primarily a result of rigorous maintenance schedules devised by manufacturers and knowledgeable mechanics who were trained to spot possible issues visually. With advances in digitization and the information age, more measurements could be used to assess the state of an engine's health, especially with the added complexity of turbofan jet engines over the older propeller types.

Unsurprisingly, a real push for understanding this newly collected information did not occur until the expertise of manufacturers was paired with a need to drive down the bottom line. Originally, engine OEMs (Original Equipment Manufacturers) contractually relinquished responsibility for problems with the product upon delivery to customers, making it the responsibility of airlines to prevent mechanical failure. This was generally limited to following suggested maintenance guidelines and trying to utilize spot measurements by a growing array of digital sensors. Because of the long life cycle of airplanes (coupled with several economic crises that prevented many carriers from replenishing their fleet) and the reasonable prevention of failure by the traditional combination of maintenance and visual inspections, there was a limited drive towards

utilizing, or even collecting, much of the information these new sensors were capable of generating.

As a result of some of the latest financial hardships many commercial airlines have experienced, manufacturers began offering services to complement the products they were selling, indirectly spurring research into the capabilities of this new source of data. OEMs realized that it was profitable for them to sell maintenance services on a fleet-wide basis, and airlines were all too happy to not only sign up for a more centralized and experienced form of engine maintenance but to relinquish the legal responsibility for the operational state of their engines.

One of the more interesting supply chain issues is the disconnect that exists between engines and the rest of the plane, since the primary jet engine manufacturers (General Motors, Pratt-Whitney, and Rolls-Royce) do not work on delivery of the whole plane. Carrier airlines and aircraft manufacturers (chief among them Boeing and Airbus) both act as clients for engine manufacturers, who specialize in producing this key component in the context of the desired airframe. This means that any module that the OEM would like to include as part of the main body (such as data collection storage or transmission circuitry) becomes optional if it is not essential to proper engine function. Understandably, many carriers have opted out of these add-ons for cost-reduction purposes and to simplify the workflow around their fleet. Fortunately, with the recent rise in fleet management programs, OEMs can require these add-ons be included in the service package and the stagnation in data collection seems to be subsiding.

It should be noted that there is also an important shift taking place as this, more active approach to engine health management, develops. Under the direction of OEM guidelines, airlines were obligated to perform routine maintenance that was scheduled on a "worst-case-scenario" basis; it was generally overly conservative in order to account

for the most extreme conditions that engines may operate in. As this responsibility, and the associated costs, are moving back to the OEMs, more liberal and profit-maximizing strategies are being developed. Rather than blindly overhauling every engine assuming the most brutal degradation profiles, it is economically preferable to tailor each maintenance schedule to actual usage and let engines degrade to the farthest operational point possible. New methods will need to incorporate this type of risk assessment into algorithms that handle individual component problems.

Methods for monitoring relevant engine health information, particularly in the case of jet engines that operate at extremely high speeds within a variety of harsh environments, have progressed to include a collection of metrics such as pressures, temperatures, speeds, and vibrations. Acoustic information has also recently been tested as a means of evaluating engine fitness, owing to the ability of mechanics to hear when an engine just doesn't "sound" right. The typical approach to evaluating these mounts of information is for a maintenance technicial to scan for glaring abnormalities and do a physical inspection of the engine itself, trusting that the trained eye is worth more than a confusing jumble of numerical values. [Vol13]

Up until the turn of the century, the standard protocols for airlines were to gather the instantaneous values that the gagues for certain parameters (pressure, temperature) were indicating and either have the pilots monitor them during the flight, or periodically transmit them to maintenance crews on the ground [TB99]. Cited among the chief problems with these approaches were the difficulties associated with transmitting the data and the unrealiable conclusions to be drawn from its rudimentary analysis (mostly comparing values to manufacturer-provided operating norms). In recent years, increasingly successful approaches to analysis of this data have been making headway in improving the capabilities of detecting and diagnosing problems that may occur during or after use

of such engines [RK00] [RI05] [BOS09] [BSOA09]. While each method has had its mixture of benefits and drawbacks, the general trend continues to be towards a more data-driven and automated approach to EHM.

1.1 Motivations for Research

Airlines operate vast fleets of planes whose health is carefully monitored and recent global financial woes have put a strain on an already-complex industry. With the historically low costs of processing power and physical hardware, the abundance of data being collected by a multitude of devices in on each plane is consistently more of a problem than a benefit without proper tools to interpret an overwhelming floot of information. New maintenance solutions that can supplement and potentially replace a large portion of the mechanical work done to keep planes in the air are sorely needed. [Saf13]

Despite having an extremely low chance of breaking down, commercial arilines have a high burden of proof to meet in order to placate worried customers. Airline operators themselves can only put forward their best effort in keeping their fleets of planes in reasonable working condition while meeting demand. In order to ensure the safety of these planes, a more objective approach to maintenance requirements and necessities is needed. Although automated safety systems in many fields are and should be secondary to human judgement, they tend to serve as a rational and rigid boundary on what will and won't cause harm, or lead to catastrophes. An automated fault detection and diagnosis system, whether it operates in real-time or offline, will significantly improve the identification of issues that may be missed by routine maintenance and may even anticipate problems before they fully manifest themselves. [Joh11]

Maintenance is a tricky issue - almost as much an art as it is a science, when it comes to the best mechanics. As discussed by Tumer [TB99], the industry of jet engine

maintenance is fraught with inconsistent standards, vague documentation, and a largely ad-hoc approach to determining the fitness of a particular engine. A system that supplements the traditional approach - suggesting places to look, automatically finding patterns while checking for anomalies, and one that does this consistently without variation built up from years of habits - is an increasingly necessary innovation.

The safety of thousands of daily air passengers is determined by the proper upkeep of the components that keep their planes in the air - upkeep that is constantly at odds with the bottom line. Because of the decline of airline solvability as a result of the collapse of financial institutions, unexpected fluctuations in the number of customers, and increasing costs of operation tied to the rising price of oil, airlines have looked for any place they can in order to stay above water. An automated system built from commercially available components that significantly reduces the tedious work done by maintenance crews will allow them to be more cost effective when working on plane maintenance. A system which detects minor component defects and identifies their nature before they lead to a general breakdown will quickly become an invaluable tool for the efficient upkeep of an airline's fleet [VBL08].

1.2 Overarching EHM Industry Challenges

There are a series of practical issues that uniquely define this problem, and make it a difficult tradeoff between what needs to and can be done. Chief among these are [Vol13]:

- Accessibility of Live Data - samples collected from real flights are still difficult to come by, because they are generally the intellectual property of airlines.

- Sampling Rate - where data is collected, the sampling rate is generally no greater than 1 Hz, while it is known that significant harmonics occur at frequencies much greater than this

- Small Fault Sample Size - the number of live flight records from engines that experienced significant faults or failures is on the order of 10-20 such cases in the history of the field. Significant problems are, thankfully, extremely rare, but even smaller faults often go unnoticed until a degradation trend can be established over many days and many records were not kept to a degree of detail that would allow current research the benefit of exploiting such prior knowledge

- Complexity of Real-Life Flights - traditional methods only consider averaged behavior during a steady state (or "stationary") phase of flight, such as when the plane is at cruising altitude and experiences no turbulence. Modern approaches, especially when data is collected throughout the course of the entire flight, will have to address those "transient" moments that actually comprise a significant portion of every flight

1.3 Organization and Contributions of Research

The proposed research will address many of the motivating problems discussed in the previous section, as well as draw attention to the theoretical and practical challenges of mechanical failure detection and diagnosis. The general contributions of this work are introduced below, as a supplement to what is outlined in the Abstract, and will be justified throughout the course of the dissertation.

Chapter 3 introduces the experimental test of acoustic and vibration sensors in a controlled test-cell environment, using basic engine fault detection and diagnosis, where a

variety of approaches and tradeoffs are analyzed. Extensions to this in the form of real-time engine breakdown detection are introduced in Chapter 4 and a discusion of low-frequency vibration sensor analysis follows in Chapter 5. Chapter 6 includes extensive experiments on advanced Gas Path Analysis (GPA) approaches, which primarily utilize existing temperature, pressure, and rotational sensors. Data is simulated from the perspective of a real-world flight and subsequent detection/diagnosis results are synchronized with live jet engine data.

As mentioned in Section 1.1 of this chapter, the key needs for an efficient maintenance system will be addressed through a combination of offline and real-time approaches to fault detection and diagnosis. These methods will range from identifying the moment that a problem occurs, checking the behavior of the engine in the context of the whole flight, and discussion of a framework for attempting to predict when problems may occur as a result of wear-and-tear based on historical risk analysis.

It is important to distinguish between the task of detecting a problem (which entails discriminating between normal and abnormal behavior) and identifying it (which implies an additional differentiation between different kinds of problems). The proposed methods will address both of these aspects, cofirming the intuition that detection is much easier than diagnosis, but also verifying the optimistic notion that automated identification of a problem via passive sensing is also possible.

Mechanical engines are periodic in nature; though they respond to external stimuli of thrusts and pressures, they largely operate as systems with surprisingly clear harmonics - unlike the sound of a human voice, for example. During the development of detection and diagnosis methods, the frequency characteristics of engines and ways to leverage them will be discussed.

One of the major practical issue when implementing any modern digital system is the complexity with which the system operates, both in terms of the amount of space needed to store data and the time it takes to process that data into useful information. While only a portion of the methods proposed in this work are intented to operate in real time, speed and space efficiency are important tradeoffs with performance nonetheless. In the following Chapters, the methods introduced will be scrutinized from this perspective.

Chapter 2

Background

A review of background materials will cover some of the foundational research that has been done in the field of engine fault detection and introduce the fundamental concepts used to approach this problem.

2.1 A Primer on Engine Health Management (EHM)

Engine maintenance is about 6% of the total cost of operating an airplane, with a trend of rising costs as well as an increasing share of the total cost. EHM systems can potentially save about 5 percent of this, which translates to a huge aggregate savings. EHM systems increase operational reliability, turn unscheduled maintenance into scheduled upkeep, reduce and predict shop visits, and help avoid secondary damage.

Older systems were much more passive, with fixed service intervals potentially being drastically conservative, but the advent of digital circuits in the 1970-1980s saw a shift to active control and monitoring systems. The initial success of these trending approaches drove airlines to add more sensed parameters, and in the 1990s the paradigm for maintenance services shifted to the OEMs in order to more easily distribute and predict monthly costs (fleet management programs). [Vol13]

This in turn fostered innovation and renewed interest, especailly in Gas Path Analysis approaches and extensions, and is now focusing on prognostics, transient engine models, and information fusion [SMSR12] [ZTD12]. There's an important difference between "information" and "data" in EHM philosophy, because much more than just

sensor readings can provide useful information for GPA metrics, given the appropriate background.

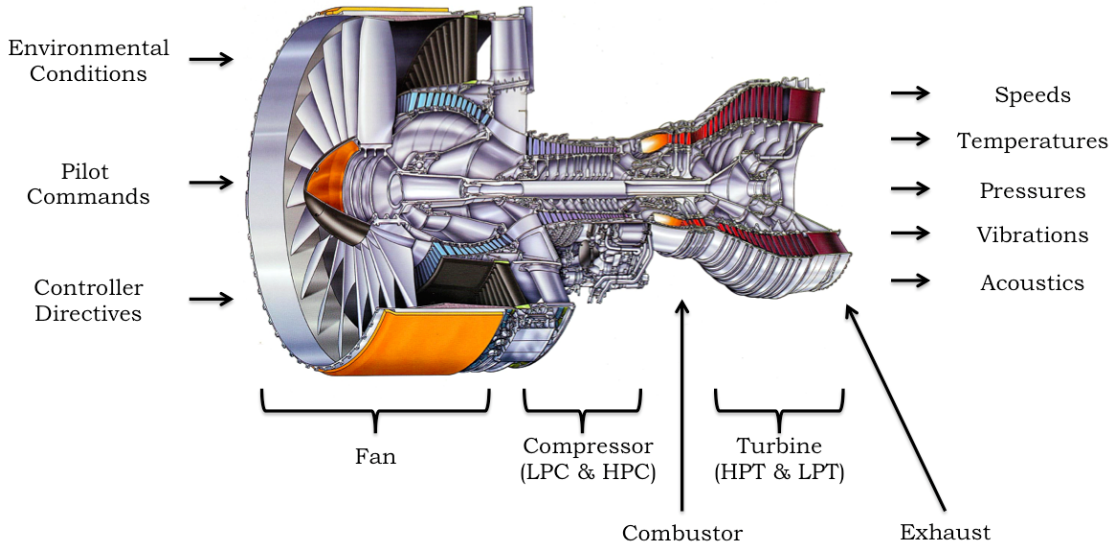


Figure 2.1: **Turbofan Jet Engine Diagram** - this is a typical turbofan jet engine, along with the main components labeled along the bottom and (rough) input/output parameters specified on either side. The outputs (right) listed here are those that will be of primary interest in this research, though input and intermediate parameters will also play an important role.

Figure 2.1 shows a general jet engine that will be widely referenced throughout this work. Tranditional gas path approaches focus on measuring a variety of variables related to pressures, temperatures, and rotational speeds of different components in order to establish a corresponance between current and baseline behavior in the steady state (usually cruising altitude, when there is little change in the behavior of the engine and all thermodynamic processes have stabilized).

Modern systems are taking advantage of various sources of real-world data that already exist in limited amounts. Most frequently, high fidelity data is collected in "test cells" maintained by engine manufacturers or maintenance facilities. A test cell is a

warehouse-like room where an engine can be mounted, fitted with a wide array of diagnostic tools, and monitored during simple usage scenarios (ramp up, ramp down, spike, steady, etc.). This type of data usually has a high-frequency profile and is used as a baseline for units undergoing maintenance with a reference to past history or an original precursor. Because these tests are performed in highly controlled and sea-level conditions, they are generally no substitute for performance characteristics measured while in flight. Live data is extremely difficult to collect because of the limits on collection, storage, and retransmission, but is becoming more common with ongoing replacement of fleets with newer models. Current work focuses on extending the understanding of data and development of algorithms that can appropriately identify different engine conditions [ZGW11] [TZD11] [CSR11].

The future will likely focus on the "intelligent engine" ideal, which will combine wireless and energy-harvesting sensors, MEMS, high temperature materials, and information fusion into a control system that can adapt the engine's operation to a variety of environmental and fault conditions. This will require a great deal of data infrastructure, verification, and validation of a variety of systems - not likely to happen in the next 20 years. [Vol13]

2.2 Overview of Relevant Feature Sets

The solution to this challenging problem highly depends on the mathematical tools that are used. To extract features from measured data, tools such as the short-time Fourier transform (STFT), the discrete wavelet transform (DWT), or filter-bank decompositions can viable depending on the nature of the information. Some methods may be improper because the underlying signals have nonlinear and non-stationary properties. When performing initial analysis of vibration and acoustic data, several methods were tested.

Matching Pursuit (MP) decomposition [Mal93] has been effectively used to analyze sounds for time-domain signature extraction. MP-based features have been proven to be effective in classifying sounds where frequency-domain features fail [CNK09]. The Hilbert-Huang Transform (HHT) [HSL+98], which was introduced by Huang in 1998, is based on an empirical mode decomposition (EMD) scheme and as the pre-processing step for the Hilbert transform. This tool has received a lot of attention in the last decade for nonlinear and nonstationary signal analysis.

The above two mathematical tools have proven successful in certain areas but preliminary results showed that they did not accurately diferentiate between different test cases for jet engine data. Believing them to be too general in their approach to these particular signals, we turned to two other technologies that are more specific to acoustics: MFCCs and CELP.

2.2.1 Mel Frequency Cepstral Coefficients (MFCC)

Mel-Frequency Cepstral Coefficients are a well-known signal processing tool that has had huge success in the speech community, in applications like speech recognition and speaker identification. In this research, the original MFCC design is slightly modified and adapted to engine acoustics (an increase in the number and density of analysis filters in the filter bank). This modification is necessary because the sampling rate of the test cell sensors on an engine (see Appendix A) is much higher than that used in traditional speech applications.

Mel-Frequency Cepstral Coefficients, in general, are proportional to the response of the human ear at different frequency locations along the spectral range that human ears can detect (20 Hz to 20 kHz). Each of the coefficients corresponds to a representative amplitude of a frequency range to which our ears are sensitive, but it is a combination of

the Mel Scale (which approximates the nonlinearities of our hearing) and the separating properties of the cepstrum which make MFCCs truly powerful. The cepstrum is a representation used in homomorphic signal processing, whereby the generalized principle of signal superposition is extended to convolution, rather than just addition. The method converts signals combined by convolution into sums of their cepstra, allowing for linear separation, and in particular, the power cepstrum is often used as a feature in further processing.

Many modern signal processing applications, especially those that deal with audio, operate on a perceptual basis; regardless of the mean-square error of a system, the true test of its usefulness is how mcuh it hides the errors that human ears are keen on perceiving. Audio compression and encoding methods remove much of the information from an audio signal solely based on the convenient fact that the human ear would not be able to perceive all of it anyway. MFCCs are the most widely used perceptual representation of audio signals, and allow for the construction of systems that accurately reflect the information we are able to subconsciously extract from audio; they are robust and versatile set of easily calculated features.

MFCCs are generally derived as follows:

1. **Windowing** - create a windowed version of part of the signal by appropriate application of a Hamming Window and overlap (usually 1/2 window length) with previous windows.

2. **DFT** - take the Discrete Fourier Transform (DFT) of the windowed signal in order to extract the frequency domain information from the signal.

3. **Mel-Scale Mapping** - calculate the powers of the spectrum, using triangular overlapping bandpass filters. It turns out that the human auditory system is not equally

sensitive to each frequency, and not able to perceive changes in volume or pitch equally well along the entire audible spectrum (20Hz - 20kHz). The Mel Scale was devised in order to more closely approximate the sensitivity of an average human ear to different frequencies, so the normal spectrum must be mapped into this alternate spectrum where some frequencies are emphasized and others are de-emphasized.

4. **Logarithm** - take the logs of the powers at each of the mel frequencies. This operation is necessary because of the nature of the ear's perception of differences in amplitude. The auditory system determines changes in volumen on more of a logarithmic scale, rather than a linear scale.

5. **DCT** - take the discrete cosine transform of the array of mel log powers. The DCT produces highly uncorrelated features from an input signal and can be efficiently implemented in hardware and software, much like the FFT.

2.2.2 Code Excited Linear Prediction (CELP)

Code Excited Linear Prediction is a well-known technique widely used in telecommunications and is the most successful technique among those used in speech coding, primarily because it is simple, fast, and effective. It is model based, and therefore has a much higher compression rate than any other speech coding technique, leading to an efficient signal representation. The breakthrough that increases performance in the detection of engine faults is the adoption of CELP model parameters as features to classifier. In this work, LPC coefficients and pitch information will be incorporated into the feature set, using the original settings of the CELP coder.

The original CELP standard adopts 10 Linear Predictive Coding (LPC) coefficients to model the data at the subframe level. In each subframe, the optimal LPC coefficients are found by efficiently solving the Durbin-Levinson recursion. Since the models assume data to be largely linear, the mean of LPC coefficients from each of the 4 subframes is taken as the final representative set of coefficients for each frame.

The original CELP standard estimates the pitch every 2 subframes with an open-loop and a closed-loop estimation method. The mean of two closed-loop estimates is taken as the final pitch information for one frame in this implementation. The pitch information and LPC coefficients contain the most important parts of the original signal.

The code excited linear prediction (CELP) technique is a mature and widely-adopted speech coding algorithm, which was first proposed by Schroeder and Atal **??**. It outperforms several other vocoders such as the linear prediction coder (LPC) and the residual-excited linear predictive (RELP) for its better quality at low-bit rates. It has been adopted as ITU-T G.723.1 [**?**] with two coding rates (namely, 5.3 kbps and 6.3kbps). After the introduction of the CELP codec, several variants (ACELP, RCELP, LD-CELP and VSELP) have been developed. CELP and its variants offer an effective low-bit-rate speech coding tool, which serve as the core of all modern speech codecs.

The CELP codec encodes speech or audio signals based on linear predictive analysis-by-synthesis coding with frame-level processing. Each frame consists of 240 samples, which are further decomposed to four 60-sample subframes. One set of CELP-based feature is obtained from each frame. For a sampling rate at 8 kHz, each frame has a duration of 30 ms. The block-diagram of a CELP consists of two input signals obtained from an adaptive and a fixed codebook, while their sum serves as the excitation to a sythesis fiter whose coefficients are updated dynamically. The excitation from the adaptive codebook is used to synthesize the main signal while the excitation from the fixed

codebook is used to account for the residual signal. The fixed codebook contains 5 to 6 fixed-position pulses as an enhancement to the excitation.

The CELP codec extracts four types of information from each frame and optimizes the decoded audio signal in a closed loop perceptually.

- Linear Prediction Coefficients (LPC)

  Each frame is first filtered to remove the DC component and decomposed into four sub-frames, each of which has 60 samples. Every sub-frame is masked by a Hamming window, after which we compute the 10-th order LPC using the Levinson-Durbin recursion and use these coefficients to construct the synthesis and formant weighting filter in each subframe. The 10 dimension LPC is sufficient to represent the regularity of audio signals in a short frame.

  In the implementation of a CELP system, only LPC parameters of the last subframe are quantized and transmitted since the LPC parameters of all other subframes can be interpolated under the linearity assumption of adjacent subframes. The LPC coefficients are employed to construct the short-term perceptual weighting filter, which is used to filter the entire frame and to obtain the perceptually weighted speech signal.

  The LPC is differentially quantized in the line spectral frequency (LSF) domain using a Predictive Split Vector Quantizer (PSVQ) and encoded according to the built-in adaptive codebook. Since the reference code uses an approximation method to acquire LSF, the average LPC parameters of 4 subframes are chosen as the CELP features. In addition to the coefficients, the following values are also used as features:

- Pitch Lag (PITCH)

For every two subframes (120 samples), an open loop pitch period is computed based on the weighted audio signal by the correlation method. The open pitch lag, L, takes on 128 values using 7 bits. The values range from 18 to 145, which correspond to the frequency range from 55Hz to 444Hz, respectively, under the 8KHz sampling rate. This is enough for most speech and audio applications since the fixed codebook can compensate for prediction residuals. Since the pitch lags in adjacent subframes are close to each other. We choose the average of the open pitch lag L in one frame as the desired feature.

- Gain of Pitch Filter (GAIN)

  The close-loop pitch lag is computed by optimizing the 5-tap pitch filter gain, b, within the codebook of the system by searching from $L-1$ to $L+2$.

- Pulse Position in Fixed Codebook (POS)

  The pulse position information is acquired by minimizing the residual error in a nested loop for each subframe. This piece of information occupies the largest portion of the bit stream.

2.3 Statistical Tools

Statistical methods are often used in studies to measure the likelihood of differences or similarities between datasets. The statistics of this dissertation revolve around the hypothesis test, which determines whether a given hypothesis can or cannot be rejected. When there is no clear representation of the probability distributions under a hypothesis, we must use resampling techniques to form enough surrogate data to adequately test the hypothesis. In this section, we will assume we are looking for the differences between two random variables X and Y .

2.3.1 Hypothesis Testing

A hypothesis test [CB01] is a procedure to reject a hypothesis so that we can verify its complementary statement. The null hypothesis, usually labelled H_0, is the hypothesis that $X = Y$. In this context, a test statistic summarizes the effect of the null hypothesis, for example a t-statistic is defined as:

$$t = \frac{\mu_X - \mu_Y}{\sqrt{\frac{\sigma_X^2}{N_X} + \frac{\sigma_Y^2}{N_Y}}}$$

where the means of the random variables are μ_X and μ_Y, the standard deviations of the random variables are σ_X^2 and σ_Y^2, and the sample sizes of the random variables are N_X and N_Y, respectively. The null distribution, describes the distribution of t under the null hypothesis. In our example, if X and Y are Gaussian random variables with unit variance, we can assume the null distribution of t to be Gaussian with variance $\frac{1}{N_X} + \frac{1}{N_Y}$ and zero mean (since under the null hypothesis, $X = Y$ so $\mu_X = \mu_Y$).

The hypothesis test is then an attempt to reject a null hypothesis ($X = Y$) because one required condition ($\mu_X = \mu_Y$) is unlikely. We use the statistic $\hat{t}$ calculated from the actual samples. The null distribution provides a probability that the statistic is above this actual value given the null hypothesis: $p = P\left(t > \hat{t} | H_0\right)$. If this *p-value* falls under a standard rate of error called the false positive rate α (in neuroscientific experiments, often $\alpha = 0.05$), we reject the null hypothesis.

Fundamentals of hypothesis tests will be used to motivate the CUSUM algorithm (Section 4.3.3) and Goodness of Fit (GoF) tests (Section 5.3.3).

Chapter 3

Fault Detection and Diagnosis via MFCC and CELP

3.1 Introduction

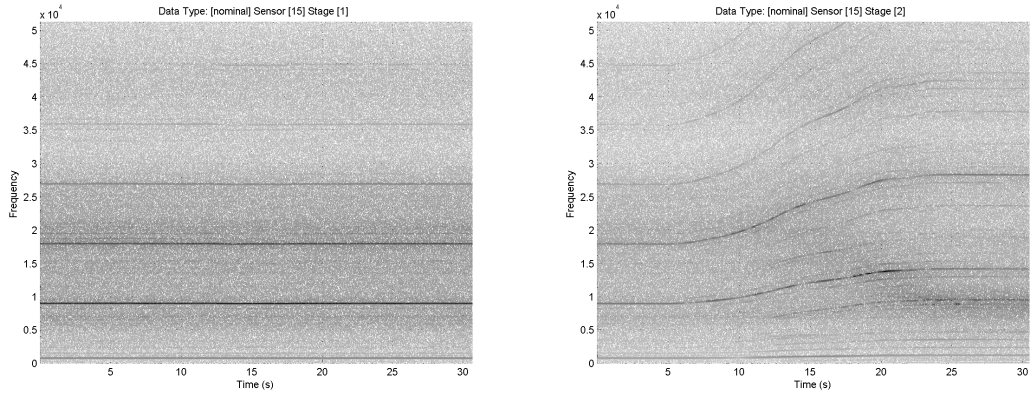
As mechanical engines have become more complex, monitoring their behavior and finding or analyzing potential problems has become a hot research topic. This development has drawn a lot of attention from researchers in academia and industry, as the availability of fast processors and efficient algorithms continues to facilitate the development of non-invasive system analysis.

For jet engines in particular, the tiniest defect may cause a malfunction, or worse, result in a catastrophe. Preventative maintenance is generally the only way to counteract such consequences, but this is a very indirect and costly approach. The airline industry, perhaps uniquely amongst industries that regularly operate engines, has come under additional scrutiny with respect to the conflicting goals of flight safety and profitability - this has put pressure on airlines worldwide to develop effective inspection methods for finding and identifying problems at the earliest possible time. It is therefore not only of academic interest, but of great practical importance to construc an easily-applicable, robust detection and diagnosis mechanism that can quickly discover abnormalities and accurately identify their causes. This would not only fundamentally improve flight safety,

but also prevent damage to the machine and substantially reduce the cost of engine repair or replacement.

Engine problems may occur as a result of a variety of internal defects, environmental conditions, abnormal structural strain, or simple wear-and-tear. Because engines by nature consist of moving parts, they have certain acoustic and vibrational properties that tend to become abnormal when integrity issues manifest themselves; as a consequence, the analysis of sound or vibration data, especially in the frequency domain, becomes an important tool for potential fault analysis systems. The frequency domain is additionally useful because the harmonic profile of most mechanical systems is very well behaved - the lack of biological components and the repetitive nature of even a system as dynamic as a jet engine means consistent peaks in the frequency spectrum of any such system. The key problem in analyzing this type of environment over the course of the system's operation is the non-stationarity of the signal as the engine operates. Here, stationarity is taken to mean wide-sense stationarity - a property of a time indexed signal which indicates that the first and second order moments (mean and variance) of a signal do not change with time. While ideal idle and cruising speeds have this stationarity property, any change in engine speed breaks this guarantee and makes subsequent analysis extremely difficult. This phenomenon is shown in Figure 3.1 below

Here, we discuss an approach to this problem that uses a joint feature extraction technique based on Mel-Frequency Cepstral Coefficients (MFCC) and Code Excited Linear Prediction (CELP), in conjunction with a Support Vector Machine (SVM) classifier. Experimental results demonstrate that the use of these two feature sets, an appropriately chosen classification algorithm, as well as proper pre- and post-processing steps successfully finds and identifies problems during cruise and idle phases of flight. Although the system does not fully compensate for the time-varying characteristics of acceleration



(a) Stationary Spectrogram

(b) Non-Stationary Spectrogram

Figure 3.1: **Sample SR-30 Spectrograms** - spectrograms of engine idle (a) and acceleration (b) data, demonstrating the inherent frequency stationarity of constant engine operation and the breakdown of stationarity when conditions within the engine (here, the speed), change.

and deceleration scenarios, it does provide some insight into how these can be analyzed. While the majority of our work is performed in the context of a jet engine, the methods listed in our approach can, to a large degree, be used in any engine analysis.

3.2 Discussion of Data Acquisition

The datasets we worked with came from two different engines, but both sets were collected under controlled laboratory conditions so as to maximize their fitness in terms of reproducability and representation. In both cases, a "nominal" data set was first collected, characterizing the engine's behavior under normal operating conditions. Next, the engine was tampered with in order to simulate a breakdown of one of its components, and one or more of such "failure" datasets were collected (more details are available in Appendix A).

3.2.1 Sensors Challenges

Since engine faults usually arise from a variety of sources whose statistical characteristics are mostly non-linear and non-stationary (such as high rotor dynamic forces or bearing interactions), data analysis in the time and frequency domains plays an important role in the fault diagnosis system. Due to the non-stationary of engine signals, traditional analytic tools have limitations when discovering useful features.

Additional constraints on any such system are a consequence of the processing capabilites of the on-board system which will be used to collect and (potentially) analyze this data. Rather than introducing new and costly systems into the already complex workings of an airplane, we aim at developing methods that could be integrated into existing systems as seamlessly as possible.

In summary, the problem of engine vibration analysis has the challenges of a complex vibration environment with little obvious time-domain information, and potential non-stationarity in the vibration readings

Additionally, any developed solution must meet the following constraints, which are specific to a real-time on-board detection and diagnosis system: an arbitrarily long set-up time, real-time data collection, real-time data processing (low computational complexity), and low memory complexity.

As is discussed in Section 3.2.2, the setup for collecting data from the SR-30 included an assortment of acoustic sensors in addition to vibration sensors. At the time, the performance of these two types of sensors was uncertain, and it was decided to test both. After the tests, however, vibration sensors proved to be far more desirable as a way of detecting engine problems, for two main reasnons:

1. *Consistency* - some of the readings from the acoustic sensors exhibit such erratic behavior that we concluded they had either malfunctioned or been placed in an

inappropriate position. As we did not have access to the test-cell setup where the data originated, we did not have the means to reproduce these experiments and collect new datasets.

2. *Performance* - discarding the readings deemed erroneous, a comparison of the detection error for similar operating conditions demonstrates a vast difference between acoustic and vibration sensors. Vibration sensors outperformed the acoustic counterparts (both were active during identical data collection) in nearly every test, and usually by a large margin

In light of these points, acoustic data will not be considered in further analysis. Acoustic sensors may be useful in other applications, but do not appear to be as effective as vibration sensors are for jet engines. The combination of variable altitudes and the volatility of propulsion seem to be at odds with the desired qualities of an experimental environment. Due to the nature of the signal these sensors detect and the indirect way in which they represent the state of the engine (engine operates > noise propagates through air > pressure waves collected by sensor), it is far more appropriate to use vibration sensors, which are directly recording the state of the engine.

3.2.2 SR-30 Dataset

The SR-30 data set used in this dissertation was acquired by Turbine Technology, and it includes a collection of data from 15 sensors mounted on an SR-30 jet engine (see Appendix A for more details. 11 sensors were used to capture acoustic engine signals (1 at the edge of the compressor blade, 9 around the engine, and 1 at the edge of the exhaust). The other four sensors were vibration sensors, mounted at 90 degrees intervals

around the outside of the engine casing. The sampling rate for all the sensors was 102.4KHz, and three separate tests were performed to acquire different data sets:

1. Nominal (N) represents the engine state in a standard configuration and balance
2. Blade Damaged (BD) represents the engine state when there is a notch cut from a blade of the compressor
3. Bearing Failure (BF) represents the engine state when one of the bearings is damaged so that it causes a vibration to occur throughout the engine

Each test consists of a sequence of sub-tests according to the typical stages of flight, which are summarized below for this particular engine. Note that speeds are given in kRPM (thousands of revolutions per minute).

Flight Stage	**Speed**	**Duration**
Idle	43 kRPM	30 seconds
(Slow) Acceleration	From 43 to 70 kRPM	15-20 seconds
(Fast) Acceleration	From 43 to 70 kRPM	~3 seconds
Cruise	70 kRPM	30 seconds
(Slow) Deceleration	From 70 to 43 kRPM	15-20 seconds
(Fast) Deceleration	From 70 to 43 kRPM	~3 seconds

Figure 3.2: **SR-30 Flight Stages** - List of flight stages and associated engine speeds for SR-30 engine experiments. Data was collected in a laboratory test cell setup with 11 acoustic sensors and 4 vibration sensors (see Appendix A.1 for more details).

It is useful to point out that, with respect to the above table and the discussion of stationary and non-stationary states of operation in Section 1.1, the cruise and idle stages are most amenable to analysis because they are the most stable in the stationary sense. Aside from brief interruptions, the engine is operating in a relatively static manner and we will show that results from these two states of flight are the best. Our methods

applied to the other four stages of flight will fare worse, but potential improvements will be discussed in Section 3.6.

3.2.3 PW4000 Dataset

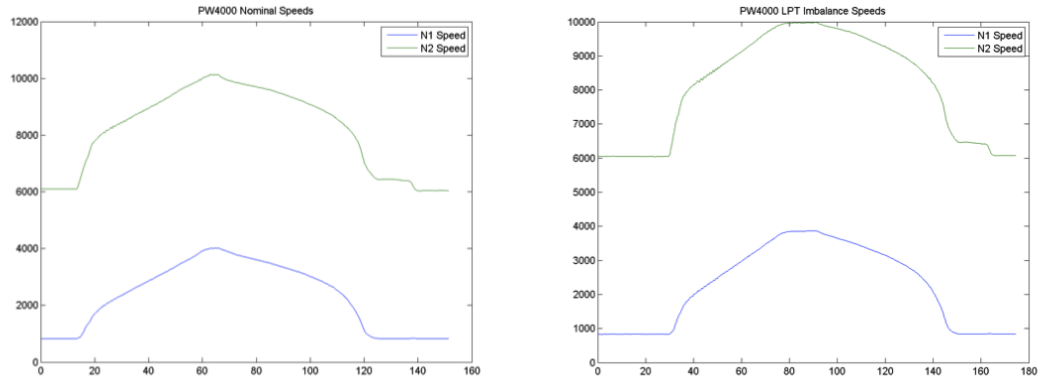
The PW4000 data is a set of real jet engine data collected using vibration sensors. They operate at a relatively low sampling rate of 25 kHz, compared to the SR-30 data. There are 55 vibration sensors or "channels" collecting approximately 5 minutes data. However, for the performance evaluation, only channels with no error were used: channel 72 - 80 and channel 99 - 113 (a total of 24 channels).

Similarly, three separate tests were performed to acquire different data sets as shown below:

1. Nominal (N) - engine operating under normal conditions
2. Fan Imbalance (FI): The fan at the air-intake of the engine is out of balance and causes a wobble
3. LPT Imbalance (LI): There is an imbalance in the low presure turbine

This dataset offers an additional chance to study the performance of our algorithm, this time using a full-sized engine. While there are no specific phase of flight indicators as seen in Figure 3.3, the data contains three different engine state parameters (nominal and two faulty operations). From the plots of the rotor speeds we can see that the entirety of the engine data constitutes several stages of operation. Because the distinctions between them are not explicit, we make the most conservative labels in order to preserve the integrity of the data.

From Figure 3.3, we can see that with high certainty, the period of 50 seconds starting at time 0 could be classified as an idle stage, so we take this portion of the data to



(a) Nominal Test Speeds

(b) Fan Imbalance Test Speeds

Figure 3.3: **Turbine Speed Profiles for PW4000 Data**, showing a test run lasting several minutes during which the engine accelerates from idle to a maximum speed and decelerates back to idle speed. The above plots demonstrate the similarity in speed profiles for nominal (a) and fan fault (b) speeds. More details about this dataset can be found in Appendix A.

be the representative of that phase. The portion which most seems to correspond to an acceleration is that between around 80 and 130 seconds, yielding a 50 second segment of acceleration data. Finally, the segment from roughly 200 to 250 seconds is characteristic of deceleration, so we label this is our last set of sample points. Conservatively speaking, no other portions of the data can be safely labeled, because:

1. The final 50 seconds of the test also seem like they could be an idle/cruise stage, but the behavior of the N2 shaft makes this assumption questionable

2. The initial portions of acceleration (around 60 seconds) and the later portions of deceleration (around 270 seconds) could be construed as rapid, but it is really uncertain whether these are just the ramping up and down of normal acceleration or truly rapid rotations

3. We have chosen to assume that the initial portion is idle, rather than cruise, but this would make little difference if the two were switched.

The original data has variations of length close to 8,750,000 samples, which roughly correspond to 350 seconds of data taken at a sampling rate of 25 kHz. The extracted features, which were generated with no overlap using windows of 240 samples, have corresponding length of around 36,400 data points. At this sampling rate, each second of raw data corresponds to approximately 104 feature points.

3.3 Experimental Setup

The general approach for this system is typical of a supervised pattern classification problem; a training phase uses known data to develop a classifier that is then used on test data in order to measure the system's performance at distinguishing between the classes of data. Here, these classes are modes of engine operation - one normal state and two (or more) failure states corresponding to the breakdown of specific engine components.

The overall system is summarized in Figure 3.4 and further discussed in the rest of this section.

3.3.1 Feature Set Selection

Since engine faults usually arise from a variety of sources whose statistical characteristics are mostly non-linear and non-stationary, such as high rotor dynamic forces or bearing failures, data analysis in time and the frequency domain plays an important role in the fault diagnosis system. Due to the non-stationary characteristics of engine signals, traditional analytic tools have limitations in discerning useful information within such complicated signals. Therefore, in this project, the joint feature extraction technique based on Mel-Frequency Cepstrum Coefficient (MFCC) and Code Excited Linear

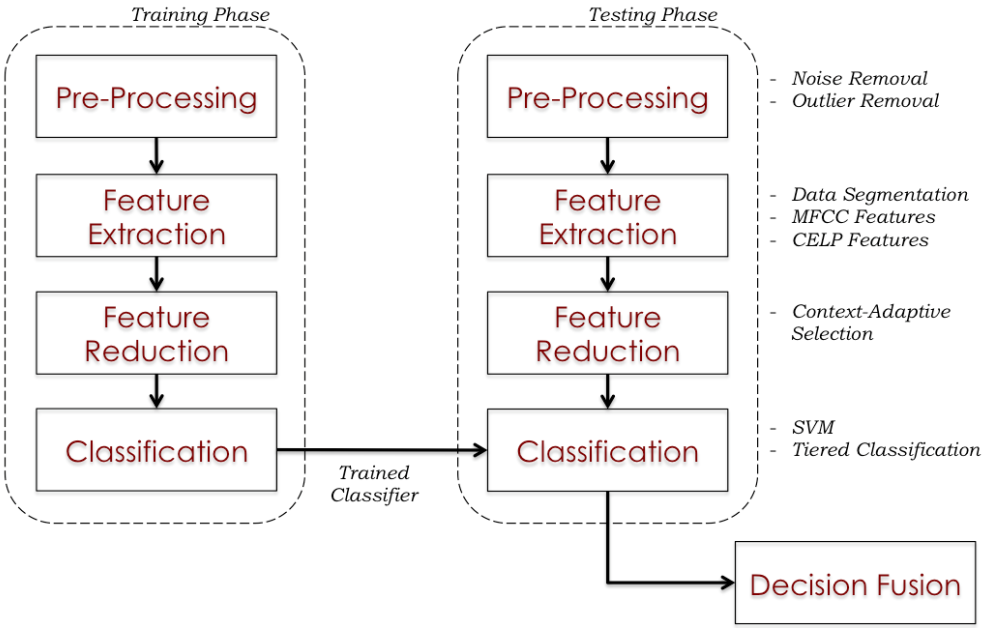


Figure 3.4: **Overview of the Proposed System**, with the training and testing phases separated for convenience and an overview of major operations. Decision fusion can be accomplished at the level of sensors, windows (after segmentation), and different classifiers.

Prediction (CELP) is used for an efficient feature representation. MFCC is a successful feature set in auditory recognition and is amenable to compensation for convolution channel distortion (Section 2.2.1). CELP is a state-of-the-art coding technique for audio signals, which provides linear prediction coefficient (LPC) and pitch information of input signals (Section 2.2.2). The experimental results, shown in Section 3.4, demonstrate that the proposed method yields fair performance for fault detection and diagnosis.

A frame is defined as a sequence of data from which a set of features will be extracted, so that one frame generates exactly one set of features. If each feature is taken as one axis of a multiple dimensional feature space, the feature set generated by one frame will be exactly one sample point in this space. The standard CELP algorithm segments the data every 240 samples, regardless of sampling rate. These original CELP setting are adapted in order to conform to existing CELP applications. In order to make

the framing of MFCC consistent with CELP, the normal MFCC frames of 512 samples (30ms at the audio sampling rate of 44.1 kHz) are also changed to be 240 samples.

Windowing is a preprocessing step for feature extraction, which employs one of a variety of techniques (rectangular window, Hamming window, Hanning window) and serves to further divide the data for processing with finer resolution. For CELP, the original settings for LPC analysis are kept, which means that each frame is equally divided into 4 subframes with 60 samples. An appropriately defined Hamming window of width 140 samples is used on every subframe to estimate the LPC coefficients. This has an overlap of 40 samples into the two subframes adjacent to the subframe being processed to provide better smoothing of the entire stream of data. For MFCC, a Hamming window with the same length as the frame is used.

3.3.2 Pre-Processing: Dimensionality Reduction

The huge amount of available engine data obstructs a real-time implementation of the system, even if the classifier training can be done offline. Because there are many different sensors and the time-resolution of the sensors is relatively high, the amount of information must be reduced or aggregated if it is to be processed by on-board systems in a timely manner. The following section presents some introductory ideas into feature space dimensionality reduction, although they are only partially explored because the system is initially being designed as an offline analysis tool.

In a typical classification setting, there is a trade-off between the number of features extracted from a dataset and the correct classification rate. With relatively few features, the classifier is prone to frequent mistakes. Additional features provide extra information and the error rate decreases. Yet at some point, not only does this trade-off reach a

point of diminishing marginal returns, but the returns become negative; the classifier's performance actually degrades with additional features.

In this examination of the engine failure diagnosis problem, it was found that reducing features from the full set of 32 MFCC and CELP features does not improve performance, which implies that with this raw view, 32 features have not "saturated" the classifier's hunger for information and more features could conceivably be derived and included to boost performance. Due to power, time, and space complexity, this is an undesirable action. However, when viewed from the perspective of context-adaptive features, a reduction below 32 features yields performance degradation at a much slower pace, which implies that other methods could be used to reduce dimensionality to something manageable while still guaranteeing a high quality classifier.

Initially, Principal Components Analysis (PCA) was employed in order to reduce the dimensionality of the underlying data. The assumption was that combined MFCC and CELP features had redundancies that could be easily eliminated by including the most highly correlated features in the data that was to be input to our classifier. However, two characteristics of PCA make it unsuitable for these purposes:

1. **Sensitivity to Scaling Variables** - for two features with equal variances that are positively correlated, PCA will result in a rotation of the feature space by 45 degrees with equal weights for the two principal components. But if the value of the first feature is multiplied by a constant K » 1, then the resulting principal components will be oriented and weighted more heavily towards the first original feature. This means that whenever different underlying variables are measured (such as temperature and pressure), PCA is a somewhat arbitrary method of analysis. In this scenario, MFCC and CELP features are fundamentally different and the resulting principal components are distorted.

2. **Representation, not Differentiation** - PCA computes the weights and orientations of the features in the direction that has the greatest variance among all the classes. It is therefore representative of all the classes rather than providing features which differentiate the classes from each other.

An alternative approach is motivated by the observation that features, which can differentiate between different classes, tend to have small standard deviations, or small variances. It is therefor helpful to use representative features with the smallest variances while avoiding those whose values are not as concentrated. Because some features may have disproportionate magnitudes, it is also useful to normalize all of the features into a common range of values and then compute standard deviations. The proposed algorithm is as follows:

1. Normalize the data for each feature separately (zero mean, unit total inter-class variance), regardless of class assignments

2. Calculate the intra-class standard deviation for each class within a given feature

3. Rank features based on the smallest intra-class standard deviation

The intuition behind this technique is that normalized standard deviations will preserve the information provided by the underlying features while identifying features that delineate class clusters. Small standard deviations will indicate that other classes are further away and each class is more concentrated, while standard deviations close to the normalized inter-class variance of 1 will mean that each cloud of data is very spread out and mixed with other data.

3.3.3 Pre-Processing: Down-Sampling

A secondary practical concern (also discussed below) is the difference between the exeperimental data, sampled at 22.5 kHz and 102.4 kHz, and real engine data, which is typically sampled at much lower rates. In order to bridge this gap, we introduce this problem below and address it fully in Section 3.5.

Practical flight monitoring systems collect data at relatively low sampling rates - typically far below the fidelity and precision of those seen in lab-collected experimental data sets. In order to validate these findings when less precise data is available, a critical down-sampling ratio, which keeps fair classification performance but maximally reduces the number of necessary samples, was investigated. From extensive experiments on the sample data, there appears to be a good trade-off between the error rate and computational complexity if the samples are decimated by certain ratios. A full treatment of this matter is available in Section 3.5, where down-sampled results are discussed in the context of fully-sampled findings from Section 3.4.

In addition, the design of a low-pass filter to be used as an anti-aliasing filter before down-sampling, since there are frequency components with considerable energy outside of the Nyquist frequency which might give a rise to an aliasing problem. In this system, an 11th order Chebyshev IIR low pass filter is used to perfectly reject the harmonics in the stop band.

3.3.4 Classifier Selection

The selection of a pertinent classification algorithm is an important step in any recognition problem, and determines a lot of what kinds of design decisions will need to be made in the subsequent tweaking of the algorithm. Because of the nature of the MFCC and CELP data, the classifier chosen for this project was the Support Vector Machine

(SVM), which provides a way to deal with a large set of data separated into relatively few classes. SVM is a technique that maps sample points into a higher dimension so that it may create a linear decision boundary with a hyperplane, rather than constructing complicated decision rules in the original dimension that tend to either severely under-represent the complexity of the data or over fit the classifier to the specifics of the training samples.

As discussed in the Introduction, we are interested in performing both detection and diagnosis of engine faults. The detection portion differentiates between nominal and a fault condition, while the diagnosis part serves to identify the type of fault.

		Decision		
		Nominal	**Fault 1**	**Fault 2**
Ground Truth	**Nominal**	Correct Detection	Detection Error Type I	Detection Error Type I
	Fault 1	Detection Error Type II	Correct Diagnosis	Diagnosis Error Type I
	Fault 2	Detection Error Type II	Diagnosis Error Type II	Correct Diagnosis

Figure 3.5: **Confusion Matrix, Non-Tiered Classifier** - this table shows a typical confusion matrix in a 3-class problem, with detection and diagnosis errors labeled. For a corresponding non-tiered confusion matrix, see Figure 3.10.

While trying to limit our choices to linear classifiers (because of their speed), we looked at the Logistic, Fisher's Linear Discriminant, and K-Nearest-Neighbors classifiers, with results shown below in Figure 3.6:

The KNN classifier is clearly infeasible, but the Logistic and Fisher classifiers performed particularly well for some of the sensors, leading us to believe that they are also good candidates for an on-line implementation. Although the trend of the SVM is that it will most likely outperform the two alternatives given enough training samples, these

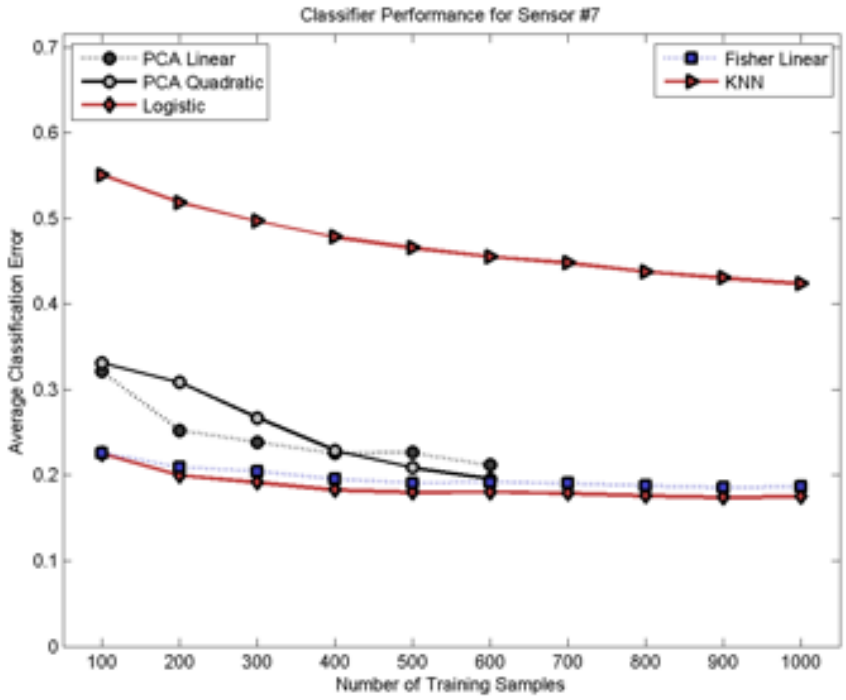


Figure 3.6: **Summary of Classifier Performance** - Behavior of various classifiers with respect to training set size (a major concern to optimize performance on limited data). Results indicate that after about 500 training samples, performance for all classifiers will converge, making individual training/testing speeds a bigger factor in classifier selection.

results were not collected because the SVM classifier requires prohibitive amounts of time and memory to finish the training stage. Their reasonable performance and the extremely short time it takes to train these classifiers make both of them good alternatives for further study.

In addition to pure detection performance, Type I (false positive) and Type II (false negative) errors are also important metrics for classifier selection. In the context of the afore-mentioned classifiers these errors were examined and are presented in the tables below.

	Detection Rate		Diagnosis Rate	
	Type I	**Type II**	**Type I**	**Type II**
SVM Linear	99.5%	100%	89.3%	75.4%
SVM Quadratic	99.6%	99.9%	79.4%	81.0%
Logistic	99.8%	100%	83.4%	88.1%
Fisher Linear	99.5%	100%	84.1%	85.8%

Figure 3.7: **Comparison of Classifier Performance, 300 Training Samples** - Type I and Type II performance (as percentages, inverse of detection errors) for both detection and diagnosis given 300 training samples for a selection of viable classifiers. Note that in each case, the detection stage performs with few errors, but there is a high level of misclassification with respect to which problem was detected.

	Detection Rate		Diagnosis Rate	
	Type I	**Type II**	**Type I**	**Type II**
SVM Linear	99.5%	100%	87.6%	81.1%
SVM Quadratic	99.8%	100%	83.4%	87.4%
Logistic	99.9%	100%	84.4%	88.7%
Fisher Linear	99.5%	100%	85.4%	86.4%

Figure 3.8: **Comparison of Classifier Performance, 600 Training Samples** - Type I and Type II performance (as percentages, inverse of detection errors) for both detection and diagnosis given 300 training samples for a selection of viable classifiers. Again, we see a very high detection rate, but low identification percentages, indicating that most classifiers will perform similarly.

As seen from the results in the above tables, the majority of these classifiers perform extremely well at the detection stage - there is usually less than 1% total error when determining whether the engine is operating at a normal state, and the false negative (faulty operation that is detected as normal) is less around 0.01%. The problem is still in the diagnosis component - here the SVM classifiers do slightly better, but the amount

of error is still around 20-30% total. Unsurprisingly, there is little difference between Type I and II errors at this stage, since both states are equally problematic to identify.

Kernels, in the context of classification, are functions of the form shown in Equation 3.1 that are substituted for the inner product operator so that linear classification techniques can be used to solve a non-linear problem. Underlying data, which is assumed to be non-linear in nature, is thereby mapped into higher dimensions so that a linear hyper-plane can be used as a discriminant function between different classes.

$$k(x_i, x_j) = f(x_1, x_2, ..., x_N) \tag{3.1}$$

For the majority of the results in this chapter, a simple polynomial kernel of order 1 is used, but this section presents results from a variety of different kernel functions. These may provide more simple or efficient mappings into an appropriate higher-dimensional space, allowing for additional performance boosts to the method. Here, we examine several of the other available kernel functions built into PRTools in order to determine the tradeoffs between performance and complexity for our problem. The following kernels are tested:

$$\begin{array}{ll} \text{Polynomial Kernel} & k(x_i, x_j) = (x_i, x_j)^p \\ \text{Gaussian Radial Basis Kernel} & k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2p^2}\right) \\ \text{Exponential Kernel} & k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|}{2p^2}\right) \\ \text{Minkowski Kernel} & k(x_i, x_j) = exp|\sum_{i=1}^{N} \|x_i - x_j\|^p|^{1/p} \end{array} \tag{3.2}$$

In each of the above, parameter p values of 1,3, and 5 were examined. Radial basis kernels or exponential kernels of higher order would have been relatively simple to do,

but the high exponents in the polynomial and Minkowski kernels would make these parameter values computationally prohibitive.

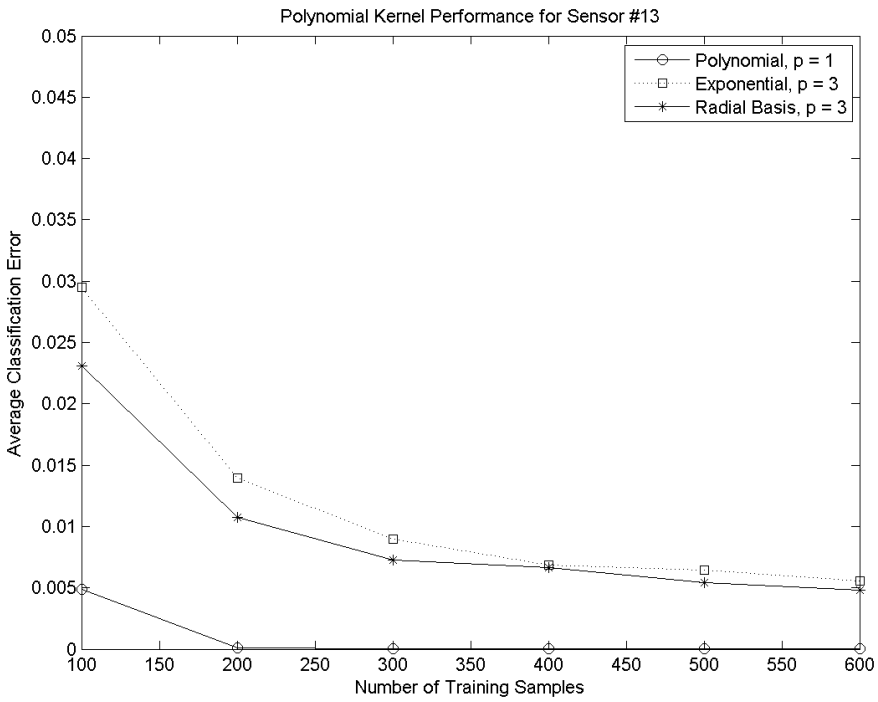


Figure 3.9: **Performance Comparison of Three Kernels** - kernels and parameters based on discussion in the above section, for a Support Vector Machine. Note that the results for the Minkowski Kernel are not displayed because of the prohibitively large training computational complexity.

The first observation is that Minkowski Kernels are not at all suited for this problem - no results are shown because even the fastest realization (parameter value of 1) not only took at least twice as long as the slowest of the other examined kernels (polynomial with p = 1), but had an error rate of around 80% even for larger training sample sizes.

Next, it appears, mainly from Figure 3.9, that the polynomial kernel with p = 1 is the best suited for the purposes of this data set, since it outperforms the other settings and actually achieves 100% detection rate in cases where at least 200 training samples are included. The only downside, as mentioned above, is that the training time is much

greater than that of the other kernels. In our given setting, where training can be done offline, this is not as much of a hindrance, but it is interesting to note that Radial Basis and Exponential kernels can approach a 99% detection accuracy while having a significant savings with respect to runtime (around 1/3 of the speed of a polynomial kernel with p = 1).

The testing times, which are not reported here, were similar for the three algorithms examined here. The polynomial times tended to increase a bit, because slightly more computation was needed to calculate the exponents, but the Radial Basis and Exponential times were nearly the same on average, owing to the fact that changes of the test parameter result in a multiplicative scalar, rather than a power operation.

3.3.5 Tiered Classification (Fusion)

Training the classifier is the most complex step in the classification phase of a pattern recognition problem. Assuming that the data has been dimensionally and semantically decomposed to its essential components, a number of factors can still influence the success or failure of the algorithm, some of which are discussed below.

The first consideration that was undertaken for robust classifier training was the size of the training set. The two main constraints on the size are the time it takes for the classifier to be trained and the amount of data that can be stored in the memory of the computer doing this calculation. Although more training data theoretically means better performance, the experience of the pattern recognition community recognizes that an overwhelming amount of information will actually overfit the classifier, making it unreasonably sensitive to minute changes in the input.

During the course of our testing, it was found that the SVM code in the PRTools package for MATLAB would not run out of memory if less than around 600 samples

	Nominal	Fault
Nominal	Correct Detection	Detection Error Type I
Fault	Detection Error Type II	Correct Detection

	Fault 1	Fault 2
Fault 1	Correct Diagnosis	Diagnosis Error Type I
Fault 2	Diagnosis Error Type II	Correct Diagnosis

Figure 3.10: **Confusion Matrices, Tiered Classifier** - this table shows a set of typical confusion matrices in a 3-class problem, with detection and diagnosis errors labeled, when the classification decisions for detection and diagnosis are performed separately. For a corresponding non-tiered confusion matrix, see Figure 3.5.

were used for testing. In effect, this means there were 1800 data points with 11-32 dimensions (600 each from the three engine operation cases). We tested the performance of the classifier by selecting 100-600 consecutive samples from a particular record of the data and testing the remaining samples based on the classifier developed with this trained data.

Classification performance improves with larger sample sizes, as expected, but the time it takes to perform each set of training/testing roughly doubles with each increase of 100 for the training set. Thus, a sample size of 200 or 400 should be a pretty sufficient tradeoff between the runtime and performance. Further improvements can be achieved by two means - simply taking the lowest points (best performing training data) as the general training sets for the entire dataset, or finding specific "good" samples within the dataset that are representative of each class.

Another modification to the classification process that was made during the course of development was to include a two-phase classifier: one for detection of engine faults and one for their identification. Both stages of classification use SVMs and the same training/testing process, but it was discovered that breaking this process down into two steps achieved several important things:

1. **Improved overall classification performance** - the improvements vary from sensor to sensor, but a sample comparison can be seen in Figure 3.11

2. **Improved fault detection** - the number of false positives and false negatives when differentiating the nominal state from either of the two faulty states is much lower than in the 3 class problem

3. **Improved training runtime** - the time necessary to train each binary classifier is less than half the time it takes to train the original 3 class equivalent

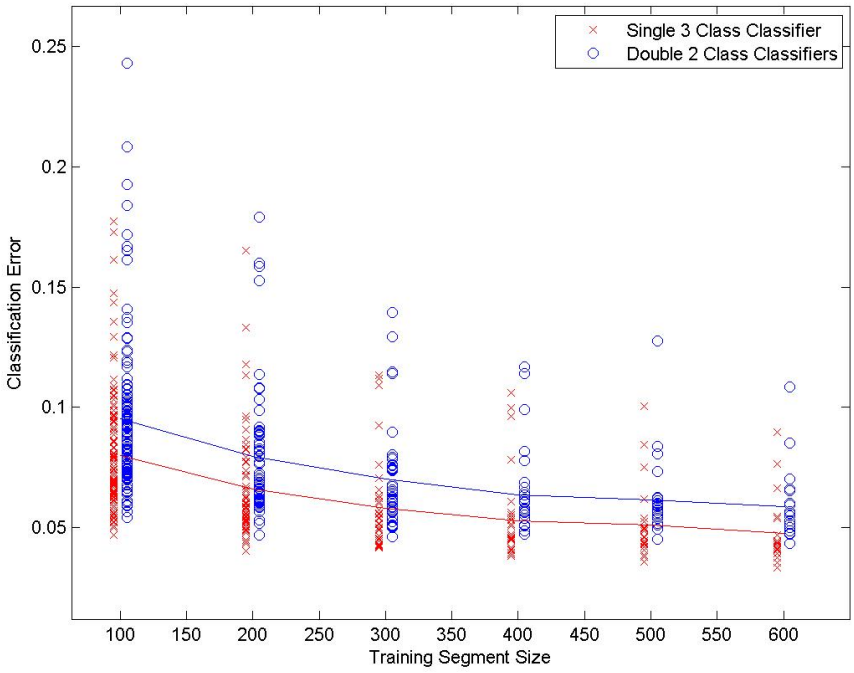


Figure 3.11: **Tiered Classifier Performance** - a comparison of a 2-stage binary classifier with a 1-stage 3-class classifier. The "x" and "o" markings indicated results for individual tests, while average performance is shown with the red and blue lines.

3.4 Results and Performance Tradeoffs

There are many perspectives through which the collected data and subsequent classification results may be viewed - these points of view yield a variety of interesting conclusions that are discussed in this section.

3.4.1 SR-30 Data

As mentioned in Section 3.2.1, the performance of the acoustic sensors when compared to the vibration sensors, was erratic and inconsistent at worst, and much less convincing at best. The performance of nearly half of the 11 acoustic sensors significantly resembled random guessing in either the Idle or Cruise stage, suggesting that there was something wrong with the setup of the sensor during that trial. The remaining acoustic sensors exhibited classification performance in the 90-95% range for Idle and Cruise stages and about 50% for the other stages (which was consistent for all sensors, as will be discussed in Section 3.4.4). The vibration sensors consistently had detection performance in the 95-99% range for both Idle and Cruise stages, even though there were only 4 of these sensors.

	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Average
Idle	95.5%	99.8%	100%	100%	**99.8%**
Acceleration	54.3%	83.1%	74.5%	79.1%	**72.6%**
Fast Acceleration	73.8%	81.9%	80.4%	82.0%	**79.5%**
Cruise	99.5%	99.8%	99.7%	99.7%	**99.7%**
Deceleration	69.5%	78.3%	82.6%	83.1%	**78.4%**
Fast Deceleration	72.7%	79.6%	86.1%	81.7%	**80.0%**

Figure 3.12: **SR30 Classification Results** - results for vibration sensors for the SR30 engine data. Average performance for idle and cruise stages is excellent, while the classification for "transient" stages is relatively mediocre. Handling of transients is addressed further in Section 3.6.

Both types of sensors, when operating consistently, had similar detection and diagnosis characteristics - the discrimination between the nominal and either one of the failure stages usually occured with very low error, but there was a lot of confusion between which type of failure state was actually present in the engine.

3.4.2 PW4000 Data

Overall, as seen from Figure 3.14, the performance of the algorithms on these vibration sensors are less efficient than expected, particularly in the case of the Idle Stage, where the average performance is around 33%, or in the realm of random guessing. One other thing of note in these figures is the lack of a trend of improvement with larger training sample sizes, which has been the norm for all cases when the algorithm did work (especially Idle and Cruise stages). This seems to reinforce the conclusion that our method was not effective in identifying these phases of operation.

	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Average
Acceleration	48.1%	48.7%	50.2%	50.8%	46.9%	51.1%	**49.3%**
Deceleration	51.7%	52.6%	49.5%	50.1%	53.0%	52.0%	**51.5%**

Figure 3.13: **PW4000 Classification Results** - detection and diagnosis results for six of the nine vibration sensors on the PW4000 engine (three sensors not shown produced inconsistent data, suggesting they were damaged). The idle stage shown in Figure 3.14 is not indicated here, as not all runs included enough data to yield comprehensive results.

A look at the slightly more detailed results gives a clue as to the reason for this. Figure 3.15 displays some interesting trends in the way the results are distributed with respect to the training segment selection. At each segment size, the actual performance of each selected segment is plotted with respect to its position in the training set, so each figure will contain 6 clusters (one for each training set size), which are ordered in their relative positions with respect to where they appear within the set of data.

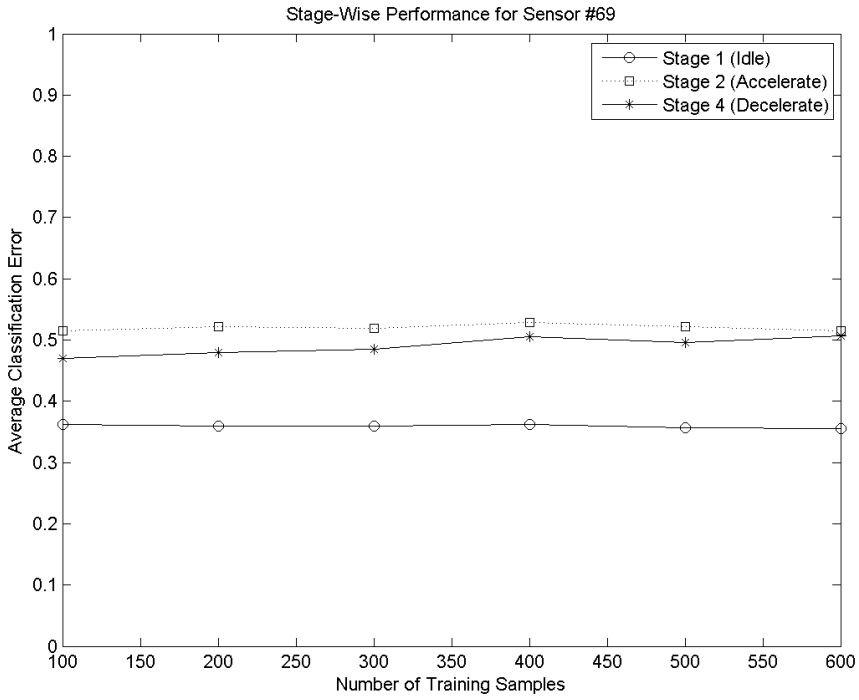


Figure 3.14: **Sample PW4000 Classification Results** - a comparison of classification results for a sample PW4000 vibration sensor. The figure indicates behavior typical of results for all sensors on the PW4000: the ineffectiveness of correct classification in the presence of complex speed patterns (transients). While results in Figure 3.13 only show classification accuracy for acceleration and deceleration stages, this plot also shows some sample results for the limited idle stage that was extracted from the start and end of each dataset (as can be seen in Figure 3.3).

There is an obvious "upward" trend in performance - earlier samples generate better classifiers. This would seem to imply that the features earlier in the training set were more characteristic of an idle phase, whereas later samples were less so. The most plausible explanation would seem to be that despite the constant engine rotation speed, the underlying statistics of the data were already changing before the shaft speed increased into the acceleration phase. Alternatively, if the statistics over the entire sample space do change significantly from idle, the earlier samples are changing at a slower pace than those later in the dataset.

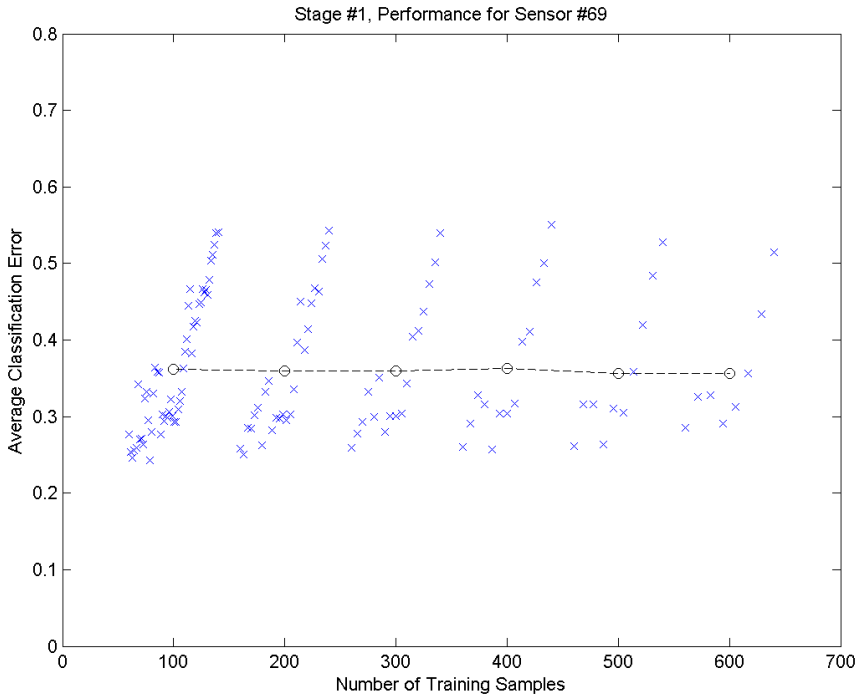


Figure 3.15: **PW4000 Results, Detailed View** - a detailed view of the results for the Idle stage from Figure 3.14 showing the classification performance of individual test iterations. It appears as though performance decreases over time (higher classification error), but as is discussed below, this is likely an artifact of non-stationarity.

Overall, these tests are largely inconclusive - the duration of the test was not long enough to give a representative operational run of the engine or our algorithm needs a radical improvement in the area of dealing with non-stationary signals. In latter sections, notably in Section 3.6, the prevailing view will be that engine transients (non-stationarities) tend to interfere with the simplicity of the model presented in this section.

3.4.3 Feature Sets

While investigating the feasability of feature space reduction, a comparison of how well MFCC and CELP features perform when used individually and jointly was conducted. The tables below summarize the findings.

Ground Truth	Classifier Decision: Nominal	Blade Fault	Bearing Fault	
Nominal	10689	1857	124	84.4%
Blade F.	778	11890	2	93.8%
Bearing F.	238	29	12403	97.9%
	91.3%	86.3%	99.0%	

Figure 3.16: **Performance for MFCC Features** - confusion matrix of results for classification using 13 MFCC features and a linear polynomial kernel SVM, based on SR-30 vibration sensor data. Using the above, the correct classification rate achieved by MFCC features alone is 92.0 %.

Ground Truth	Classifier Decision: Nominal	Blade Fault	Bearing Fault	
Nominal	10913	1744	13	86.1%
Blade F.	547	12121	2	96.7%
Bearing F.	19	4	12647	99.8%
	95.1%	87.4%	99.9%	

Figure 3.17: **Performance for CELP Features** - confusion matrix of results for classification using 11 CELP features discussed in Section 2.2.2 and a linear polynomial kernel SVM, based on SR-30 vibration sensor data. Using the above, the correct classification rate achieved by CELP features is 93.9 %.

Ground Truth	Classifier Decision: Nominal	Blade Fault	Bearing Fault	
Nominal	11222	1435	13	88.6%
Blade F.	648	12022	0	94.9%
Bearing F.	169	39	12462	98.4%
	93.2%	89.1%	99.9%	

Figure 3.18: **Performance for MFCC and CELP Features** - confusion matrix of results for classification using 32 joint MFCC and CELP features (as discussed in the above two tables) and a linear polynomial kernel SVM, based on SR-30 vibration sensor data. Using the above, the correct classification rate achieved by MFCC and CELP features is 94.0 %, a small improvement of the correct detection and diagnosis rate, but a significant decrease in Type I and II errors.

From the above results, it can be seen that the inclusion of CELP features improves the overall performance of the system. The data shown in Figure 3.18 is based on 32 features (11 CELP and 21 MFCC), but even this number of features can be slightly reduced to yield an acceptable detection rate, using an appropriate selection from Principal Components Analysis.

3.4.4 Flight Stages

In this section, results comparing stages of flight are presented for the vibration and acoustic sensors from the SR-30 Dataset. When operating consistently, both sets of sensors had similar characteristic performances demonstrated in Figure 3.19

As seen in the Figure, Idle and Cruise stage performance tends to gravitate towards the 90-95% detection region, while the performance for the other stages of flight is still far from reliable. What is more, there does not seem to be a particular effect of the number of training samples on the non-stationary stages of flight, indicating that there is something much deeper and more complext going on. Further discussion of this issue is presented in Section 3.6.

3.4.5 Computational Complexity

Analyzing the complexity (whether it is of the space or time) for this implementation of a classifier is difficult, primarily because of two factors related to our setup:

- MATLAB Environment - using MATLAB code speeds up development of an algorithm, but serves to hide some of the costs incurred during runtime. Overall, MATLAB requires a good deal of memory to deal with its own runtime environment, GUI, and internal functions. Generally, this means that code in MATLAB

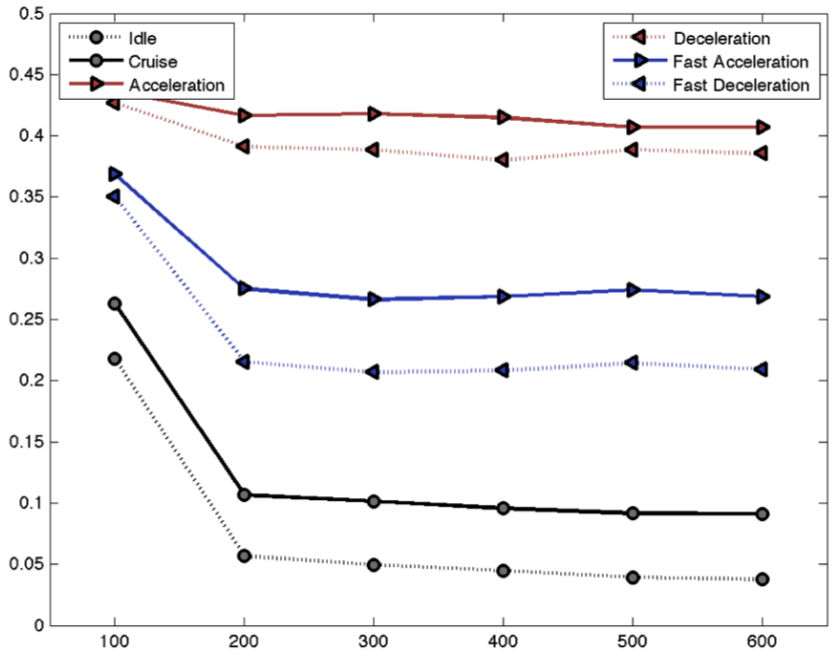


Figure 3.19: **Sample Results for SR-30 Data** - classification results for variable training set sizes for 6 phases of flight from SR-30 acoustic sensor #5. The plot displays characteristic behavior for all SR-30 sensors (acoustic and vibration), though the error rates of acoustic sensors in the transient phases of flight are generally worse than those of vibration sensors. Note that the Idle and Cruise stages generally exhibit much better performance than other stages, but that performance is only slightly affected by the size of the training set after including at least 200 samples.

will run slower and less efficiently than code that is developed for a specific platform in C/C++. Although it is possible to determine the runtime of code (see below), it is difficult to judge how much memory and system resources are being consumed to process the necessary information. As such, only rough estimates based on operating system monitoring software can be given, although these will be over-estimates of the actual necessary resources.

- PRTools Classification Package - because a partially open-source package is being used to perform the actual SVM classifier training and testing, there are inherent

inefficiencies introduced by the generality of the base code. The PRTools functions do a lot of things internally to calculate statistics and additional meta-data that are probably not necessary in a commercial implementation. There are also a few "workarounds" that have been implemented in our code that could be better implemented in dedicated code to speed up the process. Again, this implies that the resource requirements will be an over-estimate of the true system load.

We now describe some initial performance results for classifier training and testing, respectively. The two need to be separated because the classifier training is extremely memory intensive (using hundreds of samples of data at once to build a statistical model) while the testing is relatively simple (evaluating the features of a particular sample based on the classifier).

3.4.5.1 Classifier Training

This portion is the most data and processor intensive of the entire classification process, but needs only be performed once and could theoretically be complete offline, rather than in-flight.

- Code Segment Size - The code we have for the classification is a couple hundred lines long, and the inclusion of the PRTools code would most likely leave the entirety of the classifier in under 1000 lines. While running the code, Windows Task Manager showed an increase in memory usage between 20 and 40 MB. As mentioned before, this is MATLAB code, so its length and memory requirements could vary depending on the platform. In particular, since the MATLAB environment includes a lot of built in functionality it is difficult to truly estimate what the resource requirements of the code itself are.

- Data Segment Size - a typical training sample set is between 300 and 1200 samples (100 to 400 samples per class), since additional samples do not seem to improve performance considerably. Each sample currently consists of 21 MFCC features and 11 CELP features, each of which are stored as 8-Byte doubles. This results in a training data requirement of 75kB to 300kB for the data itself. The classifier is a set of mapping coefficients for each of the feature and classes into a decision space. The class provided by PRTools for this purpose currently requires 75kB and 120kB. Thus, the total required space in memory for the main data structures is between 150kB and 420kB.

- Instruction Throughput Requirement - MATLAB makes it difficult to calculate this, but an estimate can be given based on the physical runtime on our testing machine. The computer we are using has the following parameters:

  1. Operating System: 64-bit Windows 7
  2. Installed RAM: 6 GB
  3. Processor Type: Intel Core i7 @ 2.67 GHz

  The table below shows the typical runtimes for the classifier training for different training set sizes:

  Experiments have shown that the runtime is roughly quadratic with respect to the size of the training set, although this depends somewhat on the actual data, since the SVM classifier's stopping condition is based on an iterative optimality algorithm.

Training Set (samples)	**Runtime** (seconds)
300	2.7
600	20.3
900	72.1
1200	187.8

Figure 3.20: **Time Complexity, w.r.t. Training Set Size** - the above shows runtimes for the training portion of a linear SVM for SR-30 data from all 6 stages of flight for varying training set sizes, showing a roughly quadratic relationship: $O(n^2)$. This performance to complexity tradeoff leads us to suggest that 600 training samples is more than enough for constructing a suitable classifier. Experiments were performed on a dual-core 2.0 GHz PC, in MATLAB.

3.4.5.2 Classifier Testing

This portion of the program is relatively processor and memory un-intensive, because only one sample is being processed at a time and all other parameters have already been calculated.

- Code Segment Size - The code for this portion of the algorithm is relatively short, since the classifier is only being applied to samples of test data. The bulk of this work is being done by PRTools, but the classification is a very simple process - the discriminant functions (one per class) are applied to the test sample and map it into the decision space.

- Data Segment Size - a single testing sample will again consist of 21 MFCC features and 11 CELP features, each of which are stored as 8-Byte doubles. This means a memory requirement of 256 Bytes for each data sample, although that will be lower after the dimensions of the feature space are reduced. It is important to note that the classifier itself must be kept in memory during this part of the algorithm, so the 70kB to 420kB will also be necessary.

- Instruction Throughput Requirement - Again, we estimate the runtime for processing a single sample by referencing our machine's specifications in the previous section and noting additionally that we currently process all the testing samples in a batch process. This probably improves the runtime slightly (since less time is used on additional function calls and intermediate steps), but rewriting this to work on a per-sample basis will not increase the processing needs significantly. Regardless of how many samples the classifier was trained from, the algorithm required around 3 seconds to process 35,000 samples, for a runtime of around 11.6 ms per sample.

3.4.5.3 Effect of Down-Sampling

A critical down-sampling ratio, which keeps a fair classification performance but maximally reduces the number of samples, is also investigated. From extensive experiments on the sample data, there appears to be a good trade-off between the error rate and computational complexity if the samples are decimated by a ratio of 32 (25). As shown in Table **??**, the classification significantly declines at the down-sampling ratio of 64, which corresponds to an approximate sampling frequency of 1.5kH in the system, even though the computational complexity significantly decreases.

3.4.6 Decision Fusion

Depending on the memory considerations of an on-board system, it may be necessary to combine some of the classification results throughout the operation of the engine during the flight. Although a mechanic or maintenance crew would most likely prefer to see all of the information from each sensor throughout the duration of the flight, it

Down-Sampling (rate, N:1)	**Runtime** (seconds)	**Performance** (Correct %)
1	85.5	85%
8	9.3	81%
16	8.3	79%
32	8.2	80%
64	7.9	60%

Figure 3.21: **Time Complexity and Performance, w.r.t. Down-Sampling** - this table shows the rough tradeoff between detection accuracy and computational complexity for a linear SVM classifier with SR-30 data from all 6 stages of flight. While average (for all 3 classes, 6 flight stages) classification rate hovers around 80% until downsampling by a factor of 32, there is a significant dropoff in performance for higher factors (more discussion of this can be found in Section 3.5.2). Experiments were performed on a dual-core 2.0 GHz PC, in MATLAB.

is conceivable that a concise summary of the engine's performance might be required, without the needs of performing complicated data analysis.

The first method of aggregating the data would be to simply average the decisions for each sensor on the basis of a window, taking the most frequently occurring decision within that window as a representative decision for that window. This would significantly reduce the amount of data that needs to be stored, but it might also artificially remove behavior that might lead to suspicion of an immanent failure. The decisions for each window are determined based on the majority of the sensor's individual decisions within the confines of the window. Although the threshold of this majority-rule decision could be lowered to permit a more sensitive analysis of the underlying signal, there will still be a possibility of erasing important information that might give maintenance crews an early warning with respect to some impending problem. Because windowing is involved in this kind of signal synthesis, an appropriate investigation of proper techniques

would also need to be conducted, taking into account especially the effects of window overlap and window shapes/weights.

The information generated by the detection/diagnosis system can also be collected on a stage basis; different classification techniques would be necessary to analyze the engine's acoustics during different stages of flight, so this data would have different duration and even different importance. During the acceleration and deceleration stages of flight, the engine is put under much more stress, which might reveal structural integrity problems detectable with acoustic sensors. Therefore, although it would be possible to consider all flight-time data as equal and average everything together, it is important to keep in mind that different engine states mean different stresses put on the entire plane, which might give contradictory output signals in this system. A higher resolution output (lack of stage-based synthesis) would then be useful in analyzing such discrepancies.

Although the details have not been thoroughly investigated, it appears that synthesis of information on strictly a time scale would be recommended in this system, if memory restrictions allow it. Any other synthesis might introduce much more uncertainty and problems than it would alleviate, and even time-based synthesis would need a careful examination in order to retain the essential parts of the system output.

3.5 Discussion of Down-Sampling Effects

During the course of the project, the focus has been moving from theoretical development to practical implementations of the fault detection and diagnosis methods. A primary concern in this matter is the applicability of our algorithms to data that has been collected with a much lower sampling rate than that used in the experimental phase of the project, which has been 102.4 kHz. Current engines collect data at a rate of 1 Hz,

but a practical system would need to function at a sampling frequency no larger than 3 kHz.

The data sets we have worked with are 30 second segments of audio and vibration sensor recordings under different engine operation conditions. At 100 kHz, this translated to roughly 3 million data points. Our feature extraction procedures compute a series of 32 features from 180 consecutive overlapping samples and we have determined that we need at least 300 training samples in order to ensure good classification performance. Therefore, with direct sub-sampling with a factor of 32-64 times (the largest numbers that meet the above criteria), the highest sampling rates we are able to investigate are approximately 1.5-3 kHz. The table in Figure 3.22 shows the correspondence between the down-sampling rate and the sampling rate.

	Raw							**QAR**
DS Factor (N:1)	1	32	128	1024	5120	10240	40960	102.4k
Sampling Rate (Hz)	102.4k	3200	800	100	20	10	2.5	1
Performance* (%)	99%	84%	88%	82%	90%	85%	82%	83%

Figure 3.22: **Effects of Down-Sampling** - this table shows the tradeoff between the down-sampling factor and resulting data sampling rate. Note that the Nyquist frequency (the highest analyzable component) of the resulting data will be half the sampling rate. Performance* here is based on synthetic cruise stage SR-30 data that only pertains to the detection (not diagnosis) part of the classification, averaged over results shown in Figure 3.24. As seen from this table, a significant decrease in performance results from any down-sampling. Experiments were performed on a dual-core 2.0 GHz PC, in MATLAB.

Longer data sets were unavailable so in order to verify the feasibility of our system, it was necessary to synthesize data at a lower sampling rate that would still be characteristic of the behavior of a real engine.

3.5.1 Generation of Synthetic Data

In the data synthesis process, the length of the data is first extended to be multiples of the original size, and then the totality of the extended data is down-sampled in order to produce a representation of engine data at a lower sampling frequency. This is accomplished by multiple iterations of data stitching, where one round of data stitching increases the length of the data set by its original size. The stitching process consists of the following steps:

1. **Chopping** - the original (organic) data set is chopped into certain number of blocks

2. **Shuffling** - a random reordering is generated and the blocks are permuted given the new ordering.

3. **Concatenation** - given the reordered segments, the blocks from the original data set are concatenated together.

The primary challenge to this method occurs during the concatenation step, because the properties of the organic data may be altered if the blocks are simply joined together without considering the transitions of the signal from one block to another.

The sinusoidal nature of the data implies that there will be periodic correlations between portions of the signal, so for each concatenation, the correlation of the two adjacent blocks is examined and blocks are matched to overlap at the highest peaks.

The offset corresponding to the maximum correlation indicates the highest "similarity" between the two blocks, extraneous data is removed from the boundary of each block. Once the chopping-shuffling-concatenation steps have been repeated sufficiently often, the result is either stored or down-sampled, depending on the requirements.

In order to determine whether our method of synthesis generated appropriately indistinguishable data, we compared the classification performance between synthetic and "organic" data. A sample for acoustic data is shown below:

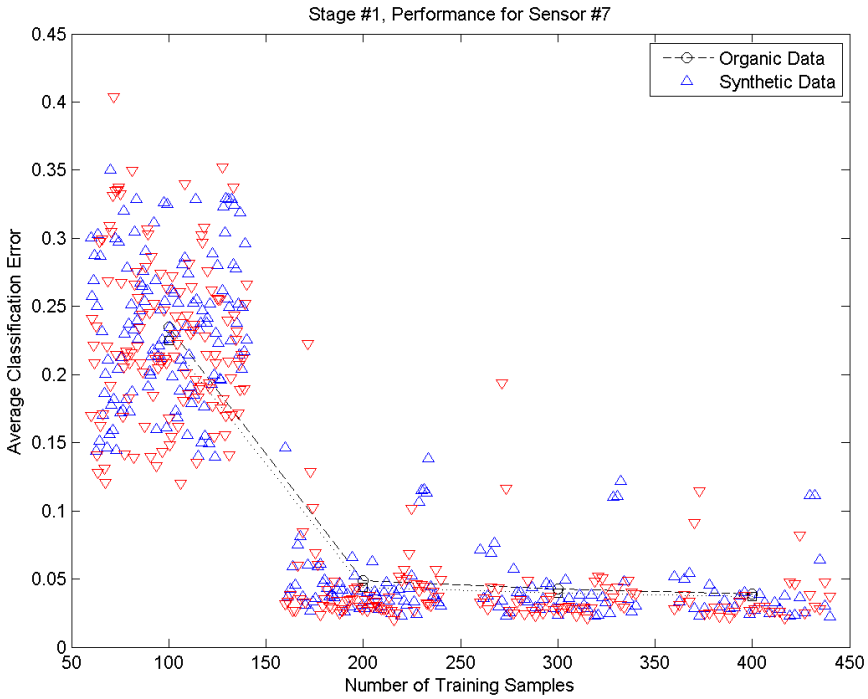


Figure 3.23: **Synthetic Data Validation** - classification performance for SR-30 Sensor 7 where synthetic data was generated for all 3 classes, all 32 joint MFCC/CELP features were extracted, and then a full-feature linear SVM classifier was applied. Note that the dashed lines (indicating mean performance) are nearly identical, even for smaller training sample sizes, implying that synthetic data is indistinguishable from real data.

Our results for vibration and acoustic data conclusively show that the synthesis method proposed here has performance that is within 1% of organic data performance. For our practical purposes, therefore, we consider data generated through this synthesis method and original data to be equivalent.

3.5.2 Down-Sampling Results

In the following figures, we summarize our results for selected acoustic and vibration sensors given down-sampled versions of the original data. Each dataset was generated using the methods described in Section II and was created to have roughly the same number of samples as the base data set (Down-Sampling Rate 1), for more appropriate comparisons. A segment size of 300 was chosen for all datasets as it provides a satisfactory tradeoff between performance and computational complexity.

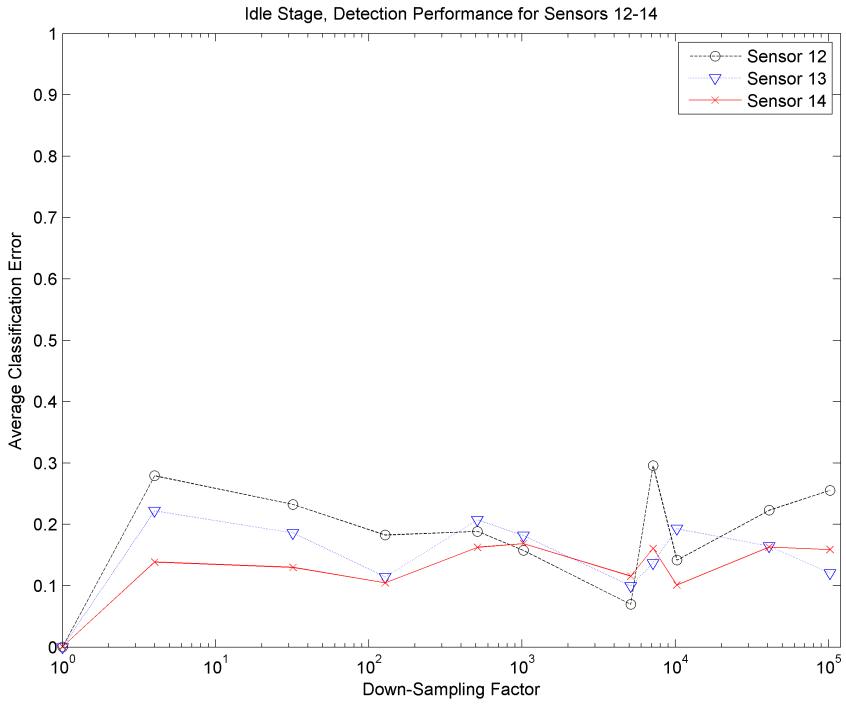


Figure 3.24: **Synthetic Down-Sampling Performance, Detection** - downsampling performance results for three vibration sensors from the SR-30 dataset, for the Cruise stage. These results are for the detection component of the classifier, and demonstrate a relatively high rate of correct discrimination between normal and abnormal engines.

As is visible from the results presented in the figures above, the performance under down-sampling is somewhat erratic. In order to confirm our results, we analyzed the error rate at the higher down-sampling rates with finer resolution, but came up with the

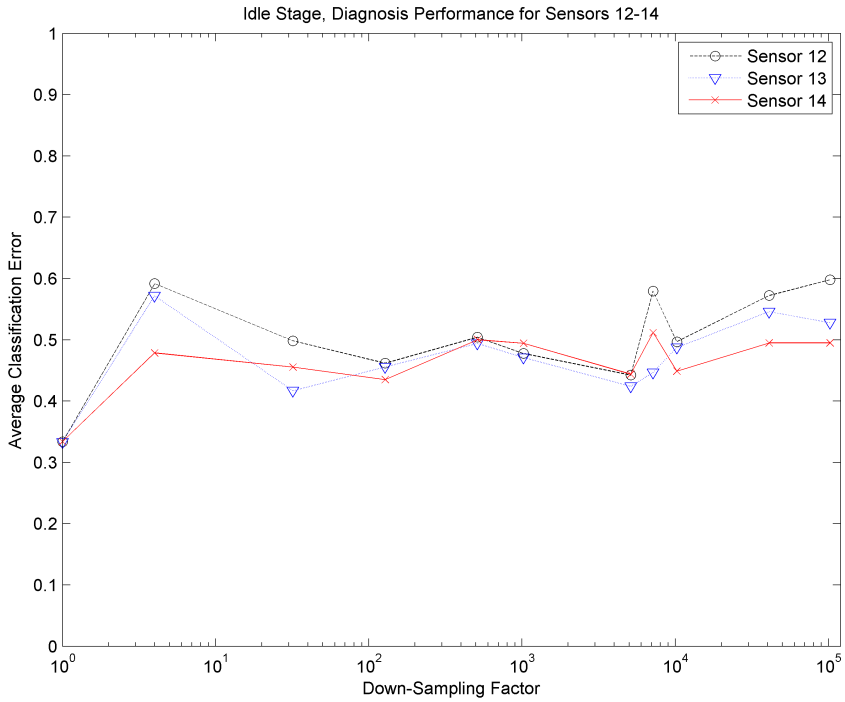


Figure 3.25: **Synthetic Down-Sampling Performance, Diagnosis** - downsampling performance results for three vibration sensors from the SR-30 dataset, for the Cruise stage. These results are for the diagnosis component of the classifier, and show that this type of analysis becomes harder as lower resolution data is used.

same patterns. All figures show a similar trend, however, in that there is a noticeable dip around the 10^2 order of down-sampling. This is helpful because a factor of 128 corresponds with an 800 Hz sampling rate, which is well within the target sampling rate of a real system. While we do not know what phenomena cause the performance to vary in this manner, there does seem to be a consistency to the lower (less than 10^3) down-sampling rate regions.

In Figure 3.26, we show a spectrogram of idle stage operation from the SR-30 dataset, along with labeled portions of the frequency spectrum that remain after several down-sampling operations. As is seen in the figure, after only 5 rounds of decimation ($32 = 2^5$), most of the information contained in the harmonics is no longer available for

DS Rate (N:1)	**Sampling Rate** (Hz)
1	102.4k
4	25.6k
32	3.2k
128	800
512	200
1024	100
5120	20
10240	10
40960	2.5
102400	1

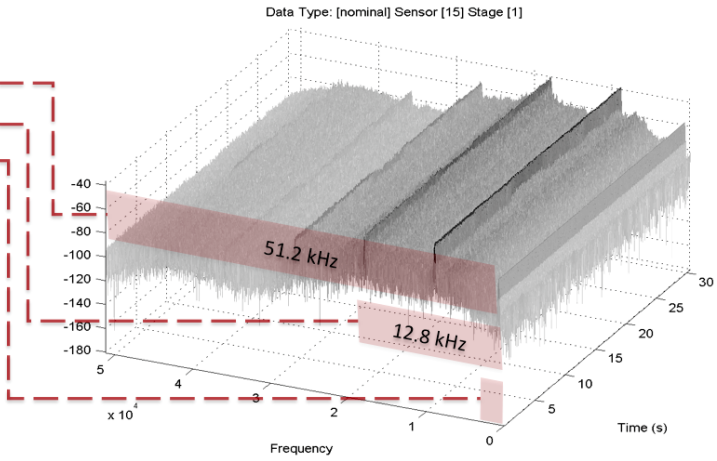


Figure 3.26: **Visualization of Down-Sampling, Idle Stage** - this figure shows a nominal, idle stage spectrogram (from Vibration Sensor 4 of the SR-30 data). The corresponding table references the portions of the frequency spectrum that are included in successively down-sampled versions of the data. After merely a factor of 32, brining down the sampling rate to 3.2 kHz, the majority of the harmonic information present in the data is no longer visible.

analysis by the MFCC and CELP features. We believe this is the reason why performance drops so drastically, as seen in Figures 3.24 and 3.25.

From this, we can concluded that use of such high fidelity features (with long analysis windows and many frequency bins) are ineffective when fundamental information is no longer present in the signal. To address this new and more complicated problem, several immediate solutions are discussed in Chapter 5 and a more comprehensive approach to low-frequency data is developed in Chapter 6. We still maintain that, given high enough sampling rates, these features and classifiers are extremely effective at handling faults detected with vibration sensors (especially given stationary stage of flight), but the complexities of non-stationarity (see Section 3.6) and low sampling rate pose significantly more complex challenges.

3.6 Non-Stationary (Transient) Segments of Flight

As was already mentioned, the non-stationary phases of flight are extremely difficult to deal with, due to factors that are sometimes out of our control. Non-stationarities may be caused by:

- Environmental Conditions - turbulance or adverse weather effects are largely unpredictable, so any data collected during extended periods of operation when an engine is subjected to such forces of nature offer little new information
- Operational Adjustments - during operation, the engine itself may vary it's behavior slightly, depending on
- Operational Transitions - an engine must transition from an "off" to an "on" state, but also frequently changes speed. The resulting vibration signatures show some patterns, which are usually highly linked to the real-time decisions of the operator.

The first two of these non-stationarities are unpredictable and cannot be simply modeled. The last, dealing with transitions between operating states, are more amenable to analysis although there is still a large degree of uncertainty in them. Although the shape (in the frequency domain) is smooth and with a predictable trajectory (for instance, when accelerating to a certain speed), the individual habits of operators prevents a blanket treatment of these trajectories as stationary accross even a large set of transitions. A similar argument can be made for the speed (in the time domain) of the transitions, since varying conditions and habits influence the rate at which the transition happens.

Nevertheless, we did attempt to approach this problem through several basic approaches. In this section, we discuss some approaches and insights into how to utilize this information to aid in the detection and diagnosis of engine problems.

3.6.1 Differential MFCC and CELP Features

In general, the problem of detection within the acceleration/deceleration modes of operation depends on the way transitions between states are treated. Figure 3.27 below shows sample delineations between idle, acceleration, and cruise, which was used by our research team in order to try to limit the types of irregularities seen in the datasets.

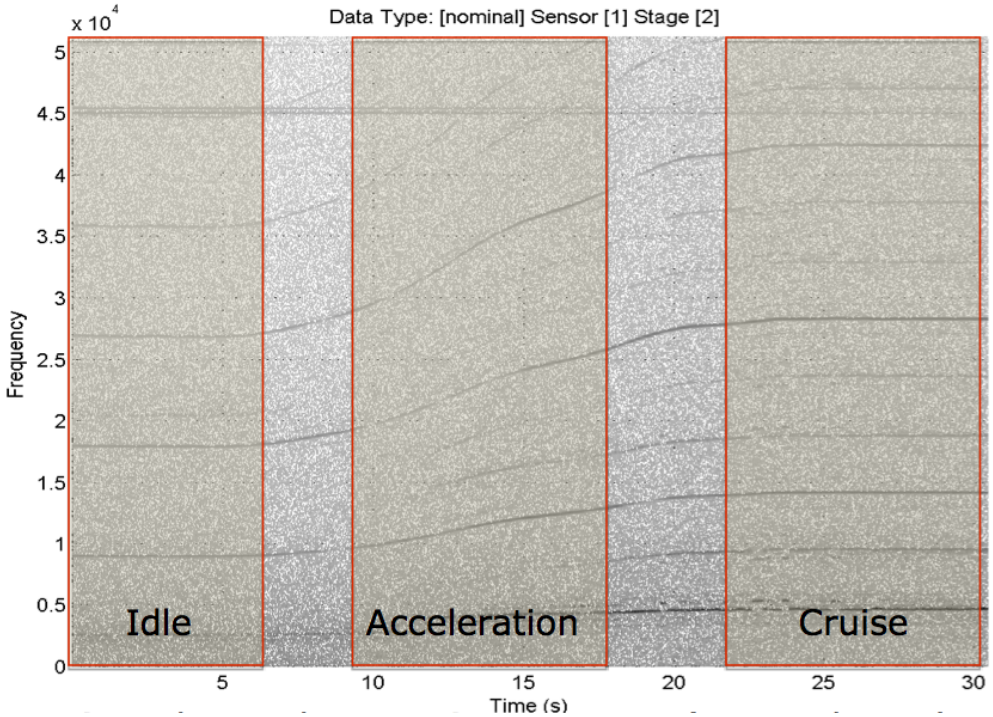


Figure 3.27: **Spectrogram with Different Phases of Flight** - a sample spectrogram showing the delineations between various stages of flight, as they were visually determined for SR-30 data.

One approach to handling non-stationarity, especially when it is known that the values of the features have a general trajectory (either increasing or decreasing, with varying speeds), is to use a first or second order derivative to track the speed at which these changes are occuring and determine whether or not the drift pattern belongs to a particular class. The differential features we used are defined as follows:

- **Delta MFCC** features are first order differences, in time, between the current and previous MFCC features. These features will detect the upward or downward trend for each frequency bin:

$$\delta_{MFCC}(i,t) = MFCC(i,t) - MFCC(i,t-1) \tag{3.3}$$

- **Delta-Delta MFCC** features are the second order differences, in time, between the previous and current Delta MFCC features. This is another way to detect trends that looks at a somewhat longer range of time data:

$$\delta^2_{MFCC}(i,t) = \delta_{MFCC}(i,t) - \delta_{MFCC}(i,t-1) \tag{3.4}$$

In our experiments, we looked at 13-dimensional and 21-dimensional base MFCC feature sets, which would then extend to 39 and 63 dimensional feature sets with the addition of both differential variants. The performance on some test data, which is representative of the performance changes we saw in all vibration and acoustic sensors, is shown below:

MFCC Dimensionality	**Stage 2** (Accelerate)	**Stage 5** (Fast Accel)
13	62.1%	53.4%
26	62.2%	53.5%
39	62.2%	53.6%

Figure 3.28: **Differential MFCC Performance, Base-13** - sample classification performance for Stage 2 (Acceleration) and Stage 5 (Fast Acceleration), assuming 13 base MFCCs. Note that there is no significant improvement.

MFCC Dimensionality	**Stage 2** (Accelerate)	**Stage 5** (Fast Accel)
21	62.9%	53.9%
42	62.9%	54.0%
63	62.9%	54.0%

Figure 3.29: **Differential MFCC Performance, Base-21** sample classification performance for Stage 2 (Acceleration) and Stage 5 (Fast Acceleration), assuming 21 base MFCCs. Note that there is no significant improvement.

As seen from the two tables, Differential MFCCs provide only a minimal improvement to the already difficult problem of detection (ignoring diagnosis) in this non-stationarity scenario. We do not believe that direct use of these features is the appropriate solution to the problem, but such features may be useful in a different capacity.

The failure of this approach can be narrowed down to the fact that it still over-emphasizes the stationarity of samples. Each sample is agnostic as to the effects its past and future neighbors have on the progression of the system. What is more, each frequency bin does not know of the relationships between it and its adjacent neighbors. An appropriate way to analyze Figure 3.27 in the context of this approach is shown in Figure 3.30 below:

Time-based trajectories would be able to detect whether a high amount of energy was continuously transitioning into a particular bin, while the majority of the high energy harmonics tend to move in and out of frequency bins. This means that the approach is ineffective because it fails to capture movement across bins - each bin is oblivious as to where the energy came from and where it is going, even though there is a clear pattern to its movement.

An approach that we did consider as an evolution of this one, with the above lesson in mind, was the definition of differential frequencies across frequency bins - either in

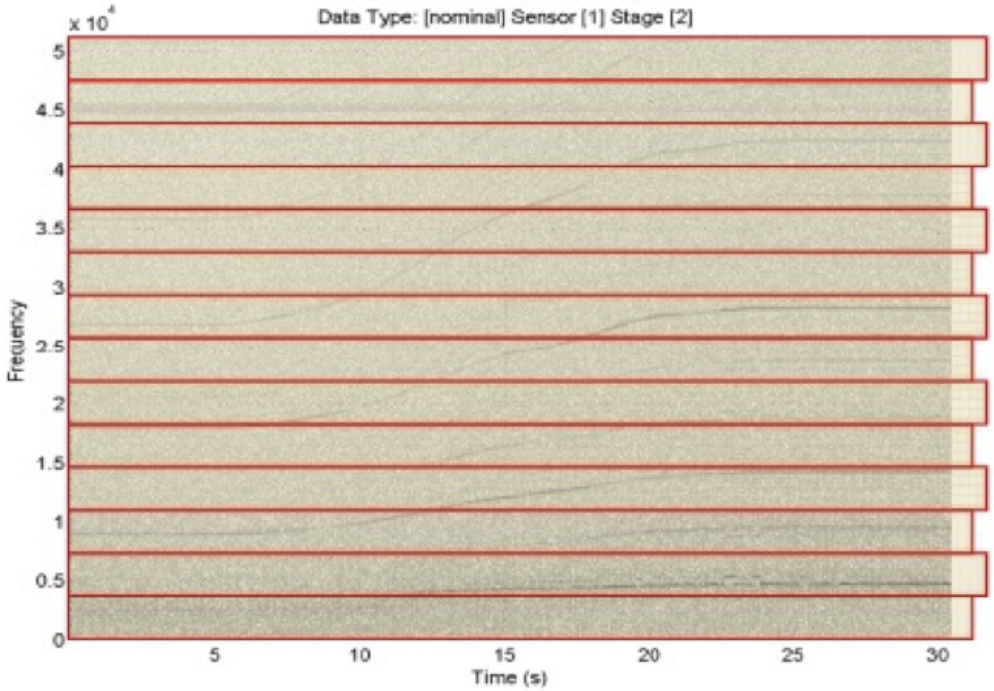


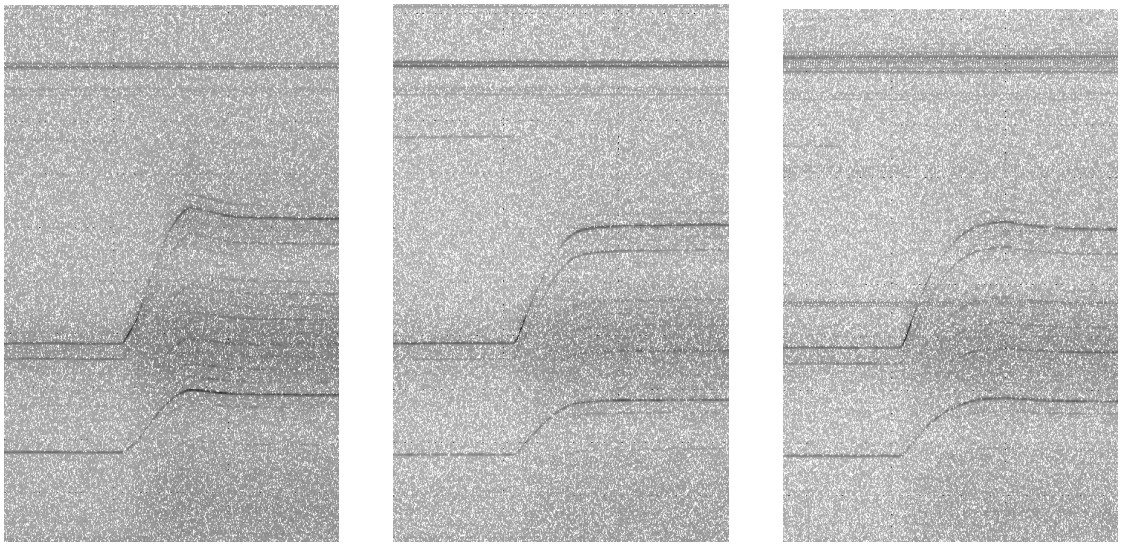
Figure 3.30: **Frequency Bins** - frequency "bin" delineations for a sample transition from idle through acceleration to cruise. Note that most high energy harmonics transition through bins, rather than moving into them permanently.

addition to or in place of the time differences. The hope would be that trajectories could be detected across bins, but here the time importance would be under-represented. An ideal view would look at two-dimensional trajectories in time-frequency, which would mean redefining the problem to a different set of features and assumptions. An analogous evolution of this idea is presented in Section 3.6.2.

We also did consider the use of differential CELP features, but because they correspond primarily to a curve-fitting polynomial approximation, there is no guarantee of their gradual or monotonic change as the underlying frequencies evolve in time.

3.6.2 Peak Tracking Through Spectrogram Analysis

As already shown in several figures, spectrograms are an intuitive and natural way of examining the time and frequency properties of a set of features. Indeed, because of the well-behaved harmonic structure of engine vibrations, a simple top-down view that largely ignores vibration magnitude resolution yields a relatively clear picture of what the differences are between acceleration in the nominal and failure states. The figure below shows a comparison of these tendencies.



(a) Bearing Failure (b) Nominal (c) Blade Damage

Figure 3.31: **Comparison of Fast Acceleration Spectrograms** - sample spectrograms for Sensor 7 during Stage 5 (fast acceleration), representing (a) Bearing Failure, (b) Nominal, and (c) Blade Damage behavior. Note that some differences between nominal and fault behavior are discernable visually, but difficult to characterize within a fixed time or frequency range.

As can be seen from the figure, each transition has differences in the trajectories of prominent harmonics in the 5 - 35 kHz range. A different view of these peaks (shown in Figure 3.31 as darker lines) is shown below in Figure 3.32:

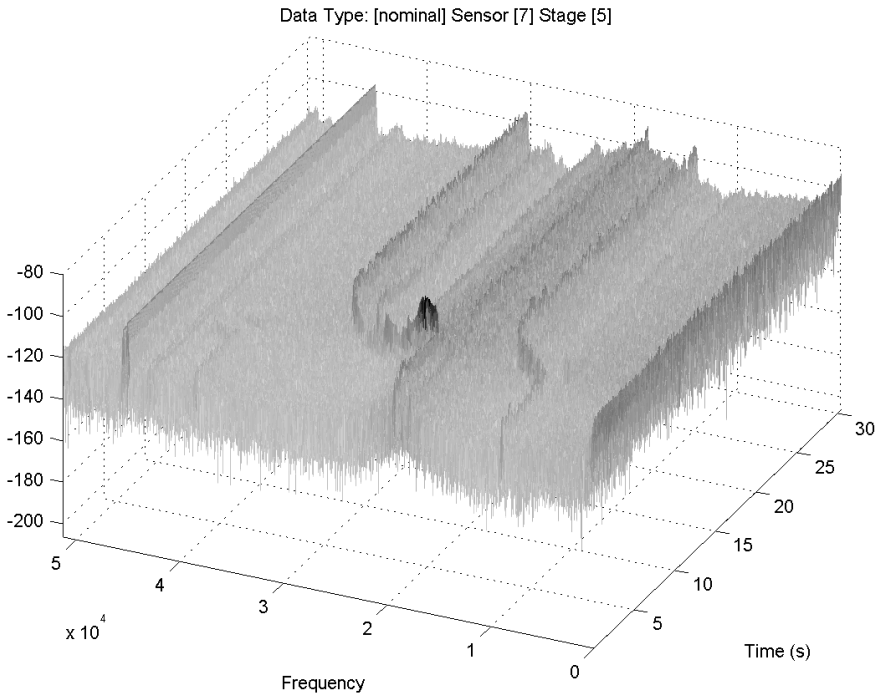


Figure 3.32: **Spectrogram, Side View** - sample spectrogram for SR-30 data from Sensor 7 during Stage 5 (fast acceleration) representing nominal behavior. Various harmonics can be seen transitioning from idle to cruise.

Although more peaks are visible, the higher frequency harmonics tend to blend with the noise floor and are therefore less visible. The goal of this approach is to somehow compare the shapes of the curves in 2-dimensional the time-frequency space for different scenarios and be able to determine what state the engine is operating in.

This approach was only developed theoretically because it departs greatly from the overall framework present in the rest of our work. Examining curves in time-frequency implies that different resolutions of frequency are needed than the MFCC or CELP features used up to this point. While MFCC features were especially well suited to frequency-based analysis, the energy bins they use are nonlinear and too large for the sort of peak tracking that this approach would require. It is therefore necessary to switch

to a standard FFT approach, where many more frequency coefficients are included in the overall analysis.

A key difficulty, and one visible within Figure 3.31, is the assumption that the input acceleration impulse provided by the pilot is identical in every scenario, and therefore each of the samples in the figure is representative of its class. Although in this case acceleration inputs were provided electronically so as to control the experiment as precisely as possible, this will not be the case in reality. It is therefore unlikely that any such peak-tracking system will be developed well enough to distinguish between real engine problems and a pilot's slip of the hand during takeoff.

The main practical issue in further developing this method is that there is really only a single set of data - one set of curves for nominal, blade damage, and bearing failure behavior. Although it is likely that given more data with high enough quality control on impulse inputs a method could be developed to detect these changes, a better approach might be to simply focus on general impulse-response behavior of an engine to precise and pre-determined inputs, rather than attempting to abstract this away into general acceleration and deceleration scenarios. Unfortunately, most such methods are most suited for maintenance settings and not applicable to live and real-time detection and diagnosis.

Although there are not enough data reproductions to confirm this, there should be similar patterns in many failure scenarios: given a similar shape and speed of the transition, an engine with an inherent problem will have noticably different vibration patterns.

3.7 Summary

From the performance analysis presented throughout this chapter, much can be concluded about the practical implementations of an engine failure detection and diagnosis

system. Above all, such a system is feasible and provides useful information when sufficient amounts of training data are available. The most reasonable approach would be to monitor the behavior of the engine while it is idling and cruising with vibration sensors, and first attempt to detect a problem before diagnosing it. While the impracticalities of operating on extremely low sampling rate data are only hinted at, there is still a reasonable chance of catching some serious mechanical failure without the need for detailed maintenance.

The approaches to the non-stationary phases of flight present an intersting problem - no doubt it is intuitive to think that the true test of any vehicle's integrity is when it is operating in rapid acceleration conditions, yet the basic pattern classification approach does not seem to yield useful results in this scenario. Alternative approaches have been suggested and partially tried, but an elegant, intuitive, and effective solution remains elusive.

Chapter 4

Real Engine Data Analysis via Change-Point Detection

4.1 Introduction

In addition to the more comprehensive and offline approach to engine fault detection and diagnosis discussed in Chapter 3, it is worthwhile considering a more immediate form of sensor data analysis. Whether breakdowns occur gradually or abruptly, we presume that the values registered by vibration or acoustic sensors would be considerably different when comparing a healthy and broken engine. Under this assumption, the analysis presented in this chapter will deal with the more sudden failures that may occur during the course of a flight. The discussion will further assume that when a change-point (transition from normal to failure operating state) occurs, some statistical property of the sensor signal will reflect this change. Our approach will attempt to detect these change-points as accurately and quicky as possible after their point of occurance in time, with the lowest computational complexity possible.

The main concern of this approach is a scenario where a component of the engine (such as a fan or bearing) fails during flight and while this may not directly or immediately lead to a comprehensive system failure, it is a cause for grave concern that pilots must be made aware of as quickly as possible. The situations where transitions from normal to failure operating states occur gradually (over the course of several flights)

and more as a consequence of wear-and-tear than severe problems will be discussed in Chapter 5, as these types of occurances require a different approach and set of assumptions.

The desirability for a real-time system that can detect an in-flight engine failure is clear when taking into account the issues mentioned in Section 1.1. That any asuch system would be able to predict sudden breakdowns is unlikely, but it is realistic to expect a method of detecting occuring mechanical failures that could lead to a breakdown. A full engine breakdown is a rare event generally caused by a confluence of internal and external events, only some of which are a result of engine component failures. By immediately detecting and identifying these individual failures, however, circumstances that put the plane in danger of a fullin-flight breakdown may be avoided.

In order to provide the pilots with the necessary information related to these potential component failures, such a detection and identification system should work by distinguishing when the behavior of the engine significantly changes from known norms. In light of the detailed results presented in Chapter 3, it would seem prudent to design such a system based on vibration and/or acoustic properties extracted from raw sensor data via MFCC and CELP features. While initially a classification system similar to that of Section 3.3.4 seems prudent, the relatively high failure detection characteristics discussed throughout Section 3.4 suggest that a computationally expensive operation such as classification may not be necessary to detect a deviation from the norm. In the majority of the confusion matrices in those results, it is clear that the detection is frequently done with little error while identification is where a larger percentage of actual confusion comes from. This indicates that a simpler analysis of raw vibration data or extracted features should be sufficient in detecting a problem, although a classifier-based

approach will almost certainly be necessary to identify the problem among a collection of possibilities.

4.2 Live Jet Engine Data

The data utilized for this analysis stems from the data used in the full offline detection/diagnosis work from Chapter 3. The key difference is that there is no practical way of generating live data that simulates a component failure - all of the failure data collected in the SR-30 and PW4000 experiments was generated by introducing a well-controlled man-made defect into the engine and collecting full take-off, flight, and landing information with the defect in place. At no point was the defect introduced during the flight itself, so analyzing failure change-points seems initially problematic. This difficulty was overcome using an approach similar to that from the down-sampling experiments discussed in Section 3.5 - the updated appropriately approach is described fully in Section 4.3.1

As discussed in Section 3.2, the vast majority of acoustic data collected and analyzed during the experiments did not have good discrimination properties between the normal and failure classes, so only vibration sensors were utilized in the synthetic generation of this change-point simulation dat.

Similarly, because of the limitations of PW4000 with respect to sufficient durations of cruise data mentioned in Section 3.2.3, only SR-30 data was used in this preliminarly analysis of change-point detection approaches.

4.2.1 Challenges and Goals

The primary goal of this part of research is to develop a method that is able to detect relatively abrupt changes (on the scale of several seconds to several minutes) in jet engine behavior in relatively real processing time (again, on a scale of several seconds), so that pilots can be notified of potential engine problems stemming from component failures. The main goal is rapid detection of abnormal behavior - differentiating between the nominal and all other forms of engine states - while operating with as little processing and memory complexity as possible.

The main challenge will be low computational complexity with a high detection rate; diagnosis or identification of the problem will be assumed to be handled by a secondary algorithm similar to that discussed in Chapter 3. While dealing with individual sensors initially, it is also desirable to verify change-points via a system of decision fusion similar to that described in Section 3.4.6

4.3 Experimental Setup and Methodology

As mentioned above, the first problem to be overcome in simulating this scenario was an appropriate dataset, which was not easily found in the data provided for the offline detection and diagnosis scenario discussed in Chapter 3. To this end, a synthetic data generation system is overviewed in the next section, following which the actual method of detection is introduced.

4.3.1 Synthetic Data Generation

Using the techniques of data synthesis for down-sampling mentioned in Section 3.5.1, it is possible to create a similarly synthetic set of abrupt failure data. In this these experiments, idle and cruise data for different failure modes was spliced together with nominal data from corresponding phases of flight in order to generate the desired transitions from a normally functioning engine to one with a specific defect. In each case, only a transition from a nominal to one of the two failure states was simulated.

Because it is not known how live failures of this type occur in flight (in particular with what speed they happen), two types of transitions were tested:

- **Abrupt Transition** - this is the fastest expected transition, which demonstrates an instantaneous change from the normal state to a failure state. Sample synthetic data for this type of transition were generated almost identically to the way synthetic data was created for the purposes of downsampling in Section 3.5.1, but the two pieces at the moment of change were taken from different classes.

- **Gradual Transition** - this is a slower change from one type of operation to the other, and includes an "intermediate" mode of operation. Several seconds after the designated change-point, there is a weighted mixing of the nominal and failure data, which simulates a less drastic change between the two states.

A total of 16 datasets were generated in this way, so as to include all of the possible transition types and failure modes, as well as four different change-points in the sequence of a 30-second segment of sensor data. The change-points were located at 30%, 45%, 60%, and 75% of the data length.

4.3.2 Change-Point Detection Features

As discussed above, one of the main difficulties of most change-point detection scenarios lies with the correct identification of a metric that exhibits different behaviors for the normal and abnormal types of data. In this case, results from the offline approaches in Chapter 3 suggest that some MFCC and CELP features are good candidates for this. Certain features have better discriminatory power than others and while the offline scenario may take care to use all the discriminatory and characteristic information contained in all of the features, an on-line system geared towards immediate detection may not have need of features that are redundant or ineffective.

To this end, a comparison of the histograms of feature values between each of the different classes was performed for all of the MFCC and CELP features. An example of an ineffective feautre is shown in Figure 4.1, where there is reasonably good identification of a bearing failure, but poor discrimination between normal and general failure states. Such features are less useful for change-point detection.

An example of a particularly well-behaved features is shown in Figure 4.2, which shows an excellent separation between representative samples from the normal and abnormal states of operation (even though it is hard to distinguish between the failure types).

4.3.3 Algorithms

As introduced in Section 2, the primary method of efficient change-point detection investigated in this research was the Cumulative Summation (CUSUM) algorithm, which computes a running statistic on the basis of Equations 4.3.3

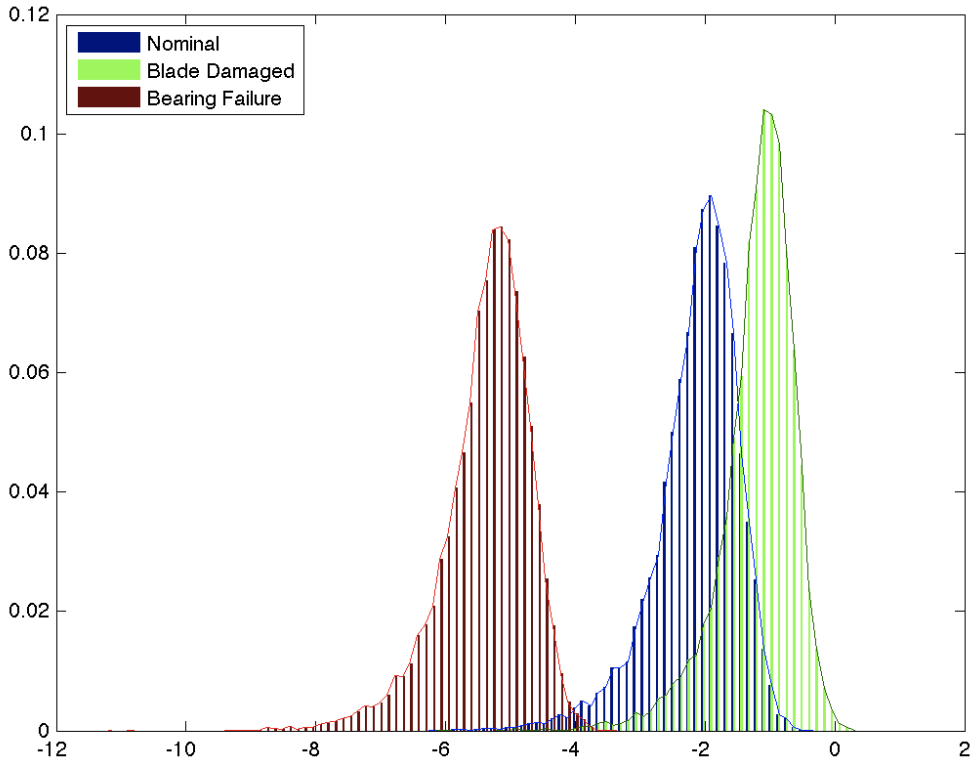


Figure 4.1: **Histogram of MFCC Feature 1 Values** - collected by vibration sensor 4 of the SR-30 dataset, displaying the distributions of each of the three operational modes. Note that it is hard to distinguish between Nominal (blue) and Blade Damage (green) samples, making this feature ineffective at detection. However, the high level of separation between Blade Damage (green) and Bearing Failure (red) means it is a great feature for fault identification.

$$\begin{aligned} &\text{Initialization:} && S_0 = 0 \\ &\text{Update:} && S_n = S_{n-1} + x_n - \mu \\ &\text{Detection Condition:} && \max(0, S_n) \geq T \end{aligned} \tag{4.1}$$

The function of these equations is to accumulate deviations of current parameter values from the mean, until a certain threshold T is reached. The main difficulty with this approach is the selection of an appropriate threshold - something usually done experimentally. As will be seen in the results in Section 4.4, an appropriate threshold is

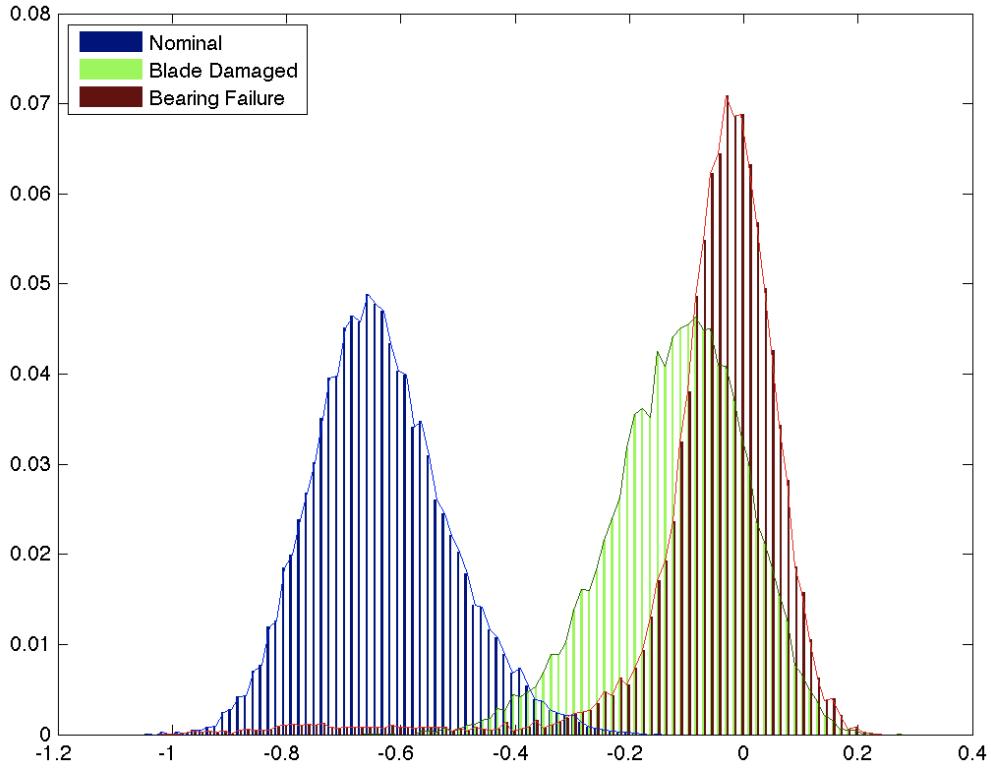


Figure 4.2: **Histogram of CELP Feature 4 Values**, collected by vibration sensor 4 of the SR-30 dataset, displaying the distributions of each of the three operational modes. Note that there is a reasonably clear distinction between Nominal (blue) and both failure feature values (green and red), even though the distinction between failures is nearly impossible from this sensor's data. This particular feature would be useful for detection, but not diagnosis.

frequently quite easy to choose due to the behavior of the CUSUM statistic under these experimental conditions.

4.4 Analysis of Results

Figures 4.3 and 4.4 demonstrate sample results from change-detection simulations on the SR-30 dataset. As seen from the CELP feature data, some changes are very distinct and easily detectable, making threshold selection for these features particularly easy,

even with large safety margins. From the MFCC feature data, however, it appears that this is not always the case. While the MFCC features seem to discriminate between normal and abnormal data sligly less than CELP features, it is surprising that this could yield such a drastic difference in detection performance.

It should be mentioned that the type of transition in the CELP dataset of Figure 4.3 was of the abrupt kind, while the other figure demonstrates a gradual transition. This intuitively makes sense in the context of how the CUSUM statistic works and partially explains the longer lag between change-point event and change-point detection, but does not explain the high variations in the MFCC features before the change-point event.

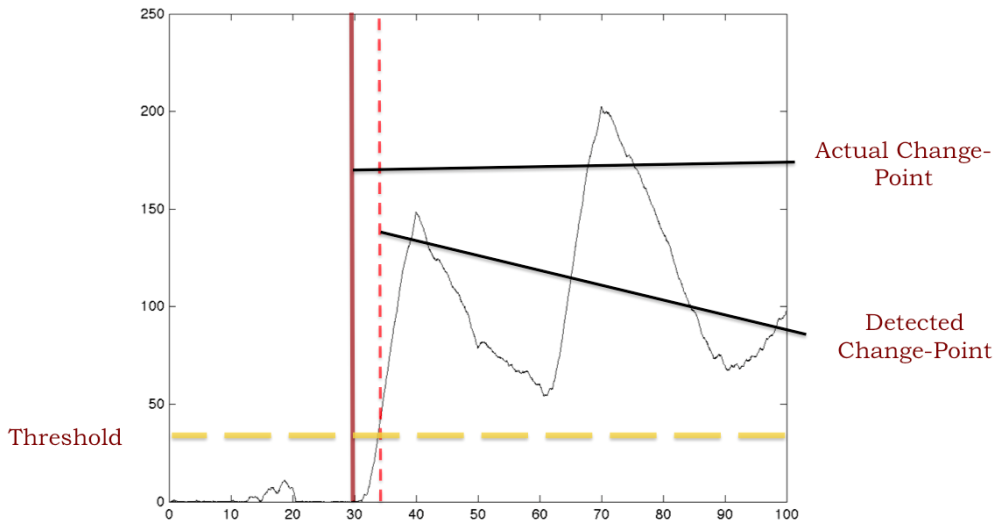


Figure 4.3: **CUSUM Chart for CELP Feature 4** - in this plot, a synthetic abrupt change occurs at the 30% mark. Note that the statistic climbs to a high value (relative to values before 30) very quickly, and it can be conservatively said that at 35%, the change has been detected. The entire dataset is 30 seconds long, so the detection takes approximately 1.5 seconds.

A total of 4 features (2 CELP and 2 MFCC) were tested under the myriad of scenarios described at the end of Section 4.3.1, and yielded an average detection rate of 75% using conservative experimentally determined thresholds within 5 seconds of the change-point event. Typical lag times varied between 1 and 5 seconds.

Figure 4.4: **CUSUM Chart for MFCC Feature 5** - in this plot a synthetic abrupt change occurs at the 75% mark. In this scenario, the system performs less convincingly, since setting the threshold to an adequately high value (around 70) means that detection occurs at the 90% mark, indicating a detection lag of about 4.5 seconds.

4.5 Summary

Overall, despite the lack of real-world data for this type of analysis, the proposed system works relatively well when considering different types of change-point events and specifically chosen features. The CUSUM algorithm requires $O(1)$ time for each step, so it is extremely time and space efficient. Having to deal with only a selected subset of features aditionally simplifies the problem, despite the fact that each feature set must still be extracted from a windowed set of raw data. A comprehensive decision fusion system should additionally increase overall detection without much additional overhead.

Because even the moderately good cases still have a significant level of noisiness and tuning required, use of single features is not likely to be successful at effectively detecting problems with the current setup. Future work extending this concept will need to focus on:

- **Detailed complexity analysis of change-point detection algorithms**. Though the current algorithm is mostly dependent on how fast the features are extracted, which is discussed in the implementation of the original method in Section 3.4.5, a more precise analysis of the the computational complexity would be necessary for any on-board implementation.

- **Change-point detection based on raw sensor data**. Pure detection of failures is done with relative ease in the original method discussed in Chapter 3, suggesting that something as complicated as MFCC and/or CELP features may not even be necessary. Extracting simpler audio-type features without the use of complicated extraction computations may yield enough data to discriminate between normal and abnormal data, at least at the full sampling rate.

- **Analysis of down-sampled data**. It is not known how badly or quickly the differences between normal and abnormal data deteriorate as the sampling rate decreases - an analysis of change-point detection performance on data synthesized similarly to what was mentioned in Sectionr̃eft1:ds warrants further investigation.

- **Analysis of PW4000 idle data**. As seen from the rotor speed plots in Figure 3.15, the dataset is dominated by accelerations and decelerations so that there is not enough data for synthesis of cruise stage data, but there may be a way to generate relatively acceptable synthetic idle-stage data to test this method.

- **Implementation of decision fusion**. Rather than generating random splicing points for each sensor, an investigation of joint datasets that have common failure events will be generated to test the performance of an individual detection and joint fusion system.

Chapter 5

Low Frequnecy Vibration Sensor Analysis

In Chapter 4, the possibility of anticipating engine break-down events was discussed as an extension of the feature extraction and classification method developed in Chapter 3. This was in the context of a real-time system that would require little computational power and be able to sense component failures that would be indicative of an engine break-down. In this chapter, a similar extension is considered, but with fewer theoretical and more practical assumptions. The approach will be solely used to detect arbitrary failures in low-frequency vibration data and attempt to do so on two time-scales: the duration of one flight and the lifetime of the engine.

5.1 Introduction

The majority of the work discussed so far has been in the context of the availability of high resolution data, both in time and frequency - a scenario which is technically and economically impractical in most circumstances. In order to make the most use of real engine vibration data, which is currently collected at a sampling rate of 1 Hz, a different method than the one based on high-frequency feature extraction and analysis introduced in Chapter 3 is needed. Although it should be apparent that such low-resolution data would either suffer from irreconcilable aliasing effects or lack any higher frequency

characteristics (because of a suitable low-pass filter within the vibration sensor), a justification of the lack of high-frequency information in these datasets is presented in Section 5.2.3.

In addition to dealing with data that is not adequate for processing via the previously developed methods, a long-term analysis of vibration sensor data should provide additional value :

1. **Performance Trends** - each engine has certain operational characteristics that are partially represented by the vibrations it makes. Capturing as much of the overall behavior from the incomplete information recorded by each sensor should allow for a by-flight analysis of how the engine performed throughout the flight. Abnormalities caused by component failures should likewise register as vibration anomalies even in low frequency data, according to the previously discussed assumptions.

2. **Break-Down Prediction** - the real value of a long-term perspective of engine vibration data is the ability to perform a comprehensive risk assessment of the engine's health in the context of previous flight history and performance of similar engines, flight patterns, or environmental conditions. With a sufficient collection of past performance data, there is a possibility of estimating when an engine's wear-and-tear influences the integrity of the system enough to cause a breakdown, irrespective of which components within the engine may be prone to failure.

5.2 Quick Access Recorder (QAR) Data

The Quick Access Recorder (QAR) is a module that allows for the recording of full-flight data from selected sensors throughout the plane. Not all planes have it or use it,

and there is no standard configuration universally accepted by the airlines, but the usage of this module is increasing in frequency, which in turn makes analysis of data from it extremely valuable.

The key challenge is that this type of data necessarily does not contain any faults or failures. Whereas test-cell data (like that collected in the SR-30 and PW4000 datasets) can be manufactured to be faulty at the mechanical level, it is unreasonable for any airline to risk permanent or catastrophic damage to a real plane by intentionally introducing problems into the engine. Even when, however rarely, problems do arise in-flight, airlines are careful not to make that type of data public for fear of legal reprecussions.

For the purposes of our experiments, several such real flight datasets were provided by Korean Airlines (KAL) from an unspecified single model of engine. Each of the 5 sets of recordings contain data for 2 engines, each of which were monitored with 5 unique vibration sensors operating at a sampling rate of 1 Hz. This raises a secondary challenge that will continue to be a theme throughout this dissertation - modern signal processing methods are designed to work well with high fidelity data, usually sampled at least at the kHz level. Most currently-used data collection equipment on commercial jets, however, uses a sampling rate of at most about 20 Hz [Vol13].

5.2.1 Description of Sensors

The configuration is summarized below:

1. **Low Pressure Turbine** - three of the sensors were placed around the low-pressure turbine (LPT, also designated N1), along with a rotational speed sensor which provided a %RPM reading indicating how rapidly the turbine was spinning.

2. **High Pressure Turbine** - one sensor monitored the high-pressure turbine (HPT, also designated N2), which also had a corresponding rotational speed sensor.

3. **Broadband** - the remaining sensor was a "broadband" vibration sensor, recording the overall viration of the entire engine

The reason for this particular concentration of sensors is that the LPT is most prone to component problems, while the HPT is less likely to fail. Of the entire engine, the N1 and N2 regions are the most critical, while all other components in the engine are monitored in aggregate via the broadband sensor. The layout of the engine and a summary of sensor locations can be seen in Figure 5.1.

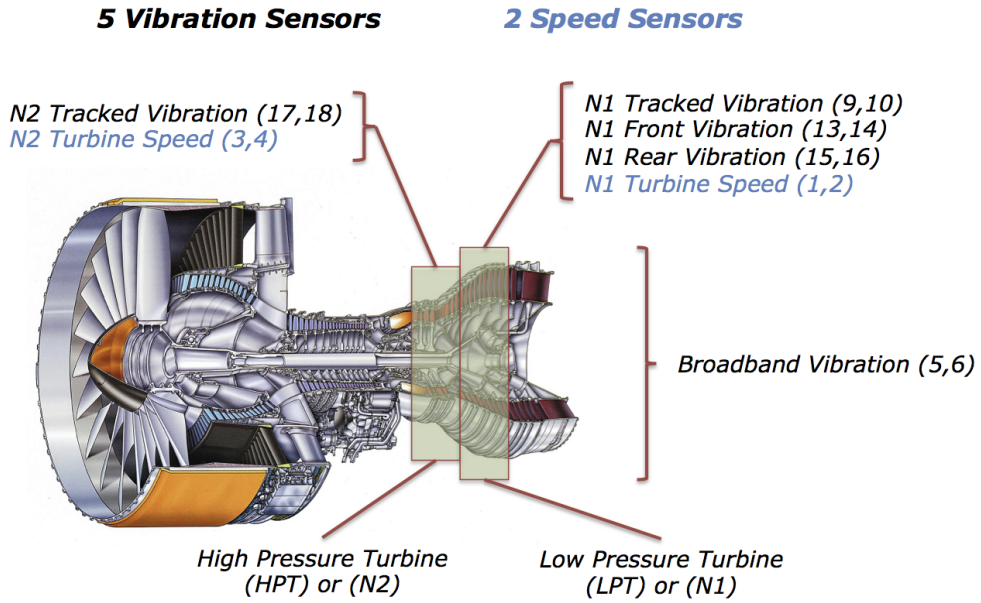


Figure 5.1: **QAR Vibration Sensor Layout** - in the figure, the High Pressure Turbine (HPT, also designated as N2) and the Low Pressure Turbine (LPT, also designated as N1) are labeled. The numbers in parentheses after each sensor name indicate the left (odd) and right (even) sensor readings. Turbine speed sensors are designated in blue, while vibration sensors are labeled in black. Note that the broadband vibration sensor monitors the vibrations of the entire engine, as opposed to the other vibration sensors, which specifically monitor N1 and N2 components.

In addition to the speed and vibration sensor readings, there is one additional designation present in the data - that of a phase of flight. In contrast to the PW4000 and SR-30 data referenced in the previous chapters, this set has only three phases of flight:

- **Climb (CL)** - this corresponds to acceleration or fast acceleration. As is seen in Figure 5.2 and in Table 5.3, this phase can last up to 30 minutes, depending on how long it takes the plane to achieve cruising altitude.
- **En Route (ER)** - this corresponds to the cruise phase and is presumed to be the most stable type of data (an assumption discussed in Section 5.1)
- **Descend (DC)** - this corresponds to the deceleration or fast deceleration phases, which (similarly to the Climb phase) may last up to about 20 minutes and may contain a series of maneuvers that cause the plane to accelerate or decelerate, depending on what path of the plane takes during final approach.

For reasons similar to those discussed in Section 3.6, the methods introduced in this chapter will primarily focus on the En Route phase of flight in order to provide the most consistent results. The CL and DC phases contain high level of nonstationarities, but are also hard to approach because of their extremely short durations in the context of the entire flight, as seen in Figure 5.2.

5.2.2 Overview of QAR Flights

Each of the five flights has unique duration and speed characteristics, and some of the details are summarized in Table 5.3 below. Again, note that the durations of the climb (CL) and descend (DC) phases of flight are considerably shorter than the en route (ER) phase, and that the durations of these are not standard in shape or length, making it

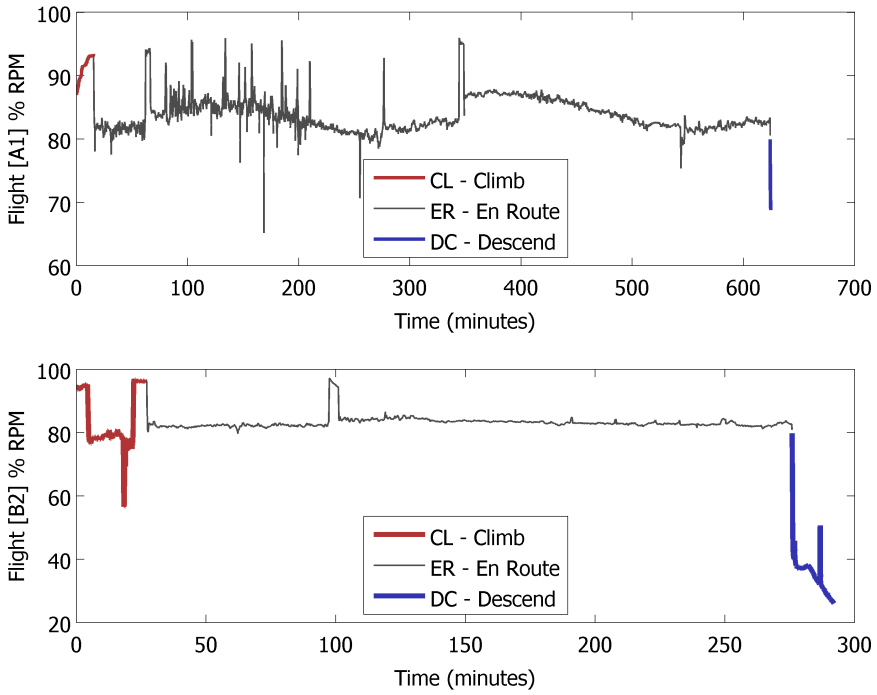


Figure 5.2: **Sample Speed Profiles** - the plot shows sample Low Pressure Turbine (LPT) speeds for Flight 1 (above) and Flight 3 (below). Note that Flight 1 has significant amounts of erratic behavior throughout the early portion of the flight, probably due to the plane's response to turbulance or maneuvers. Although much more stable, the speeds shown in Flight 3 also exhibit some irregular behavior.

additionally difficult to try to compare each ascent or descent, even in the context of the sort of spectrogram analysis discussed in Section 3.6.2.

It is further assumed that each of the two engines for each flight, while receiving similar thrust stimuli, will act independently of each other. Reasons for this assumption include the fact that each engine was manufactured and maintained separately, so while the stresses of usage may be similar, there is no reason to expect a causal relationship between two particular engines. Each pair is therefore treated separately.

The A1, B1, B2, C1, and C2 designations in Table 5.3 refer to the dataset names given to us by Korean Airlines (KAL), and specify another characteristic of the set:

Flight	Climb (min)	En-Route (min)	Descent (min)	Total (min)
1 (A1)	15	609	1	625
2 (B1)	1	49	2	52
3 (B2)	27	249	16	292
4 (C1)	10	610	22	642
5 (C2)	19	558	19	596

Figure 5.3: **List of Flights** - this table lists the flights from which QAR data was collected, including durations of each of the phases of flight (in minutes). Note that flights 1, 4, and 5 are similarly long, flight 3 is a mid-length flight, and flight 2 is considerably shorter than all the rest. The A1-C2 designations refer to the planes from which data was acquired, so that there are two datasets from two flights each (B and C), and a fifth set from a third flight (A). It is also important to note that the flight phase designations were recorded by the QAR system, though they sometimes do not accurately reflect the descent/landing phase of flight (see Figure 6.4 for a more realistic view of the lengths of typical phases of flight). Due to this inconsistency, and a focus on cruise behavior, the En-Route stage was used in most of this chapter.

Flights 2 and 3 were performed using the same plane, as were Flights 4 and 5, although none of these three plance (A,B, and C) is the same. Despite this, each flight is treated as an individual set of data.

***Note**: the only additional item of importance in the five data sets is that the KAL engineers indicated to us that Flights 4 and 5 (from Plance C) contain an overall vibration reading that is higher than normal, although still within safe operation margins.

5.2.3 Challenges of Full-Flight Profiles

Real-world data brings with it a set of unique challenges, most stemming from the inherent noisiness of non-controlled environments. The primary difference in this scenario is the lack of labeled "failure" data, which makes this a semi-supervised classification problem. The approaches to this type of problem generally rely on finding a way of

concisely describing the normal data and establishing boundaries for "failure" data. The key challenges of these approachs in the context of this dataset are as follows:

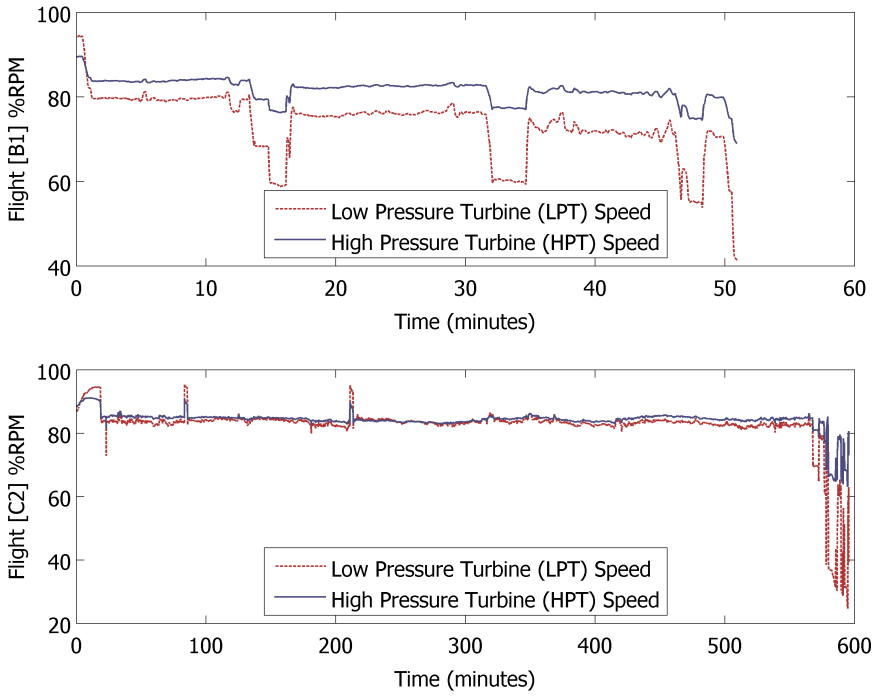


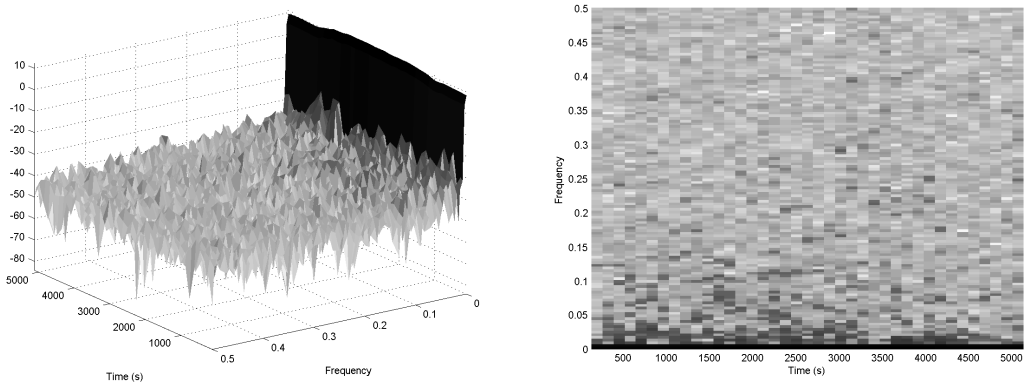
Figure 5.4: **LPT and HPT Speed Comparison** - the figure demonstrates a characteristic of all the flights: the LPT and HPT speeds are similar, but frequently have variations throughout the entire flight. Flight 2 (top) exhibits a relatively large disparity between the two speeds, while Flight 5 (bottom) has nearly identical speeds throughout the En Route phase of flight.

1. **Disparities Between LPT and HPT Speeds** - the plots in Figure 5.4 compares the LPT turbine speeds for Flights 2 and 5, demonstrating the variability of their behavior, even during the En Route (or cruise) phase of flight. Although it is relatively obivous that that the HPT speed influences the LPT speed, for the purpose of this analysis we treat these two components are independent (see Figure 5.7).

2. **Irregular En Route Speed Data** - as was seen in Figure 5.2, there is a large degree of variability each the phases of flight, but even the En Route phase (considered to be the most stable) exhibits quite extreme irregularities. All of these are assumed to be caused by the environmental effects on the plane and/or the subsequent responses to these stimuli from the pilots. They have been loosely classified to three kinds:

    - *"Salt and Pepper"* irregularities are best seen in the Flight 1 speed plots, which show a multitude of point discontinuities above and below the general trend of the speed. These are most likely caused by brief (on the order of several seconds) changes in thrust that the pilot is inducing during point maneuvers or turbulance.
    - *"Plateaus"* are portions in the flight when the speed of the plane significantly falls/rises to a different stable speed, stays at that speed for a few minutes, and then rises/falls back to the original speed. This behavior is characteristic of a maneuver to avoid a weather pattern or enact a gradual course change.
    - *"Global Trends"* are irregularities in the flight speeds that manifest themselves as very slowly varying changes throughout the duration of the flight that significantly deviate from the overall mean. Flight 1 is the most extreme example of this, but Flight 3 also exhibits a continual drop in speed throughout the course of the flight. Presumably, this behavior is a result of non-automatic pilots why try to maintain a constant speed, but invariably drift, or possibly due to a fuel or pressure leak in the system.

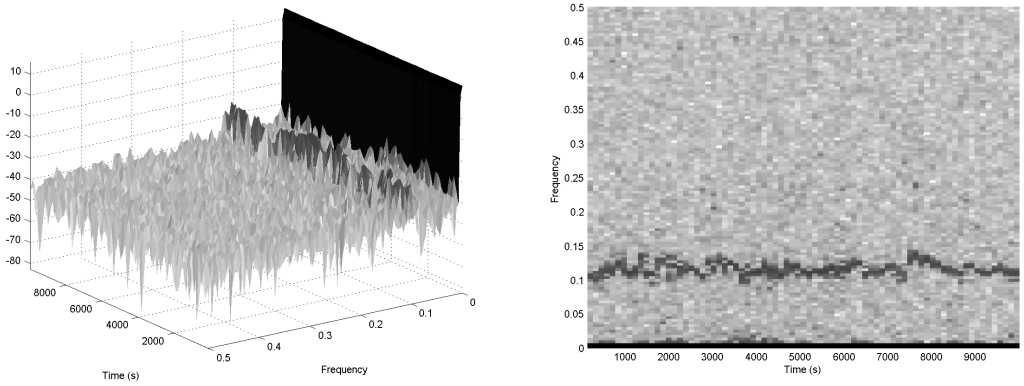
3. **Lack of Frequency Domain Information** - because of the extremely low time resolution of the samples, there is no reason to expect a wealth of frequency domain information. In the offline approaches discussed in Chapter 3, the spectrograms showed a clear representation of various harmonics representing vibrations of different engine components. Looking at the spectrograms of this dataset confirms the notion that little useful frequency domain information has been preserved after the sampling process. Figure 5.5 is typical of all the vibration sensor spectrograms for the En Route portion of flight, with the exception of Flights 4 and 5. Their typical spectrograms are shown in Figure 5.6, which has one relatively prominent harmonic component that can be explained as the unusually high vibration mentioned by the KAL engineers who provided the dataset (see the note at the end of Section 5.1.



(a) Flight 1, HPT Sensor, Side View

(b) Flight 1, HPT Sensor, Top View

Figure 5.5: **Flight 1, HPT Spectrogram** - this figure shows the side (left) and top (right) view of the HPT sensor's spectrogram for the En Route portion of Flight 1. Note that there is no non-noise information outside of the DC component.



(a) Flight 1, HPT Sensor, Side View

(b) Flight 1, HPT Sensor, Top View

Figure 5.6: **Flight 4, LPT Spectrogram** - this figure shows the side (left) and top (right) view of the LPT sensor's spectrogram for the En Route portion of Flight 4. There is a slight periodicity in this figure, but this can be attributed to the unusual vibration mentioned at the end of Section 5.2.2 since it shows up in all Flight 4 and 5 spectrograms, but in none of the others.

4. **Limited Scope of Data** - as is clear from the description of the available datasets, there is not as large a collection of sample flights as would be condusive to a thorough analysis. There is a relatively small diversity of flights (leaving other typical engine behaviors unknown), no duplicate flights (that may confirm general trends for certain routes), and no additional annotations explaining the irregularities of the LPT/HPT speed data. The general approach will be to treat all of the data as "normal," but some adjustments to this definition will be made in ordr to try to make the analysis less complicated.

5.2.4 Goals of the Unsupervised Learning Problem

One assumption that was mentioned in earlier sections is the relationship between the unknown thrust, the measured speeds, and the collected vibration sensor data. The implied and presumed interdependencies are shown in the diagram of Figure 5.7. While

there is likely some correlation between the LPT and HPT speeds, it is assumed that these speeds are independent, although the vibrations readings collected by sensors mounted to these respective components are highly correlated.

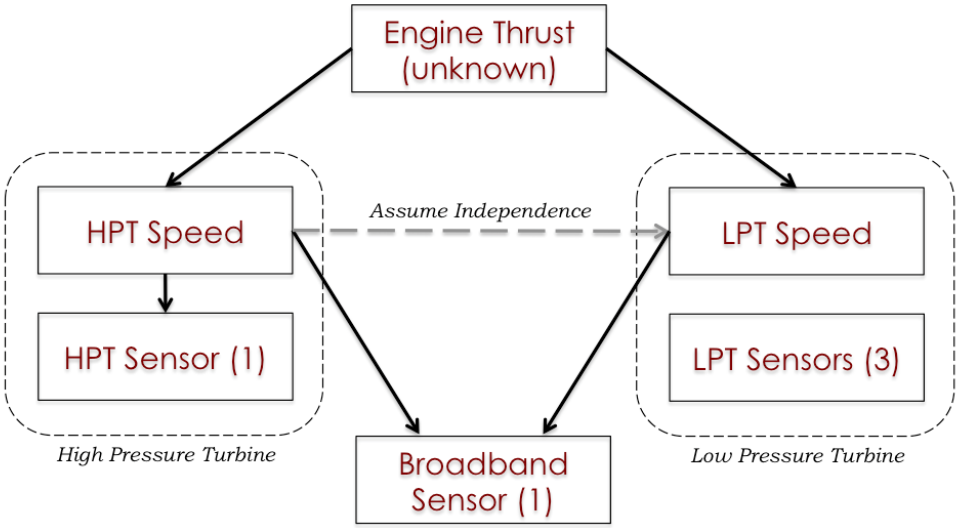
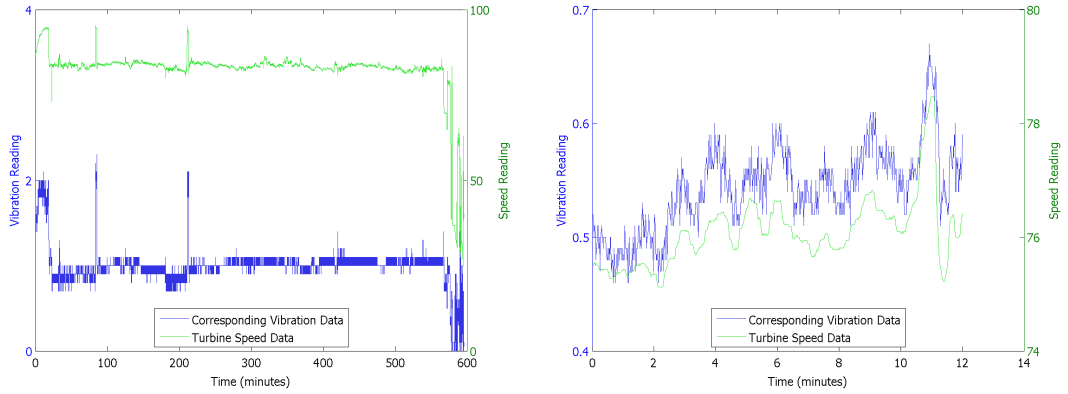


Figure 5.7: **Variable Dependences** - the above diagram details the relationship between independent and dependent variables, for a given engine (5 vibration sensors, 2 speed sensors). An unknown thrust induces certain HPT and LPT speeds, which in turn influence the recordings in corresponding vibration sensors. HPT speed has some effect on the LPT speed, but for the sake of speed/vibration and vibration/vibration analysis, these are treated as being independent (though in reality they are highly correlated, see Figure 5.4)

This correlation is confirmed by plotting the speeds and vibrations side-by-side, as seen in Figure 5.8. As a result, for a given LPT or HPT speed, there is a high expectation that a healthy engine will have vibration readings within a certain range of values - any large deviations outside of this range may indicate failure or, at the very least, some abnormal behavior warranting investigation. Thus, the heart of the problem is the selection of a suitable descriptor for the data, given the speed, and description of an appropriate set of boundaries that delineate between normal and abnormal data.



(a) Flight 5, LPT Sensor, Full Flight

(b) Flight 2, LPT Sensor, Detail

Figure 5.8: **Temporal Speed/Vibration Dependencies** - the figure demonstrates a correlation between the speed and vibration readings for selected sensors. The left plot shows the full flight speed (green) and vibration (blue) sensor readings for a given sensor while the right plot shows a detailed view, where it is visible just how closely these two signals may track each other.

Several methods in this field have been extensively developed, and can be loosely classified into boundary, reconstruction, and density methods. The first of these generally ignore time dependencies and seek to cluster data primarily based on distance information relative to other related data points. The class of reconstruction methods focuses largely on time dependent signals and seeks to predict where the next data point in a series should fall. The last class of methods assumes that data is drawn from a particular distribution and makes probabilistic estimates about where the normal/abnormal boundary should be, as well as with what certainty new data belongs to one class or the other.

While this data seems more suitable to a reconstruction approach, density methods are much more mathematically rigorous and provide clear risk assessments of how likely abnormalities (component failures) are to occur. Although density approaches demand a more precise set of conditions to be met before they are applied, the discussion of data

analysis in Section 5.3 will justify usage of these method in the case of engine vibration and speed data. Inclusion of features from other classes of methods (especially the reconstruction approach) will be discussed in Section 5.5.

In light of these general limitations, challenges, and motivations, the goals of this avenue of research are as follows:

1. **Assess Current Engine Health** through the comparison of per-fligh vibration sensor readings to general norms and boundaries learned from past data.

2. **Estimate Engine Lifetime** using a comprehensive view of how the data from successive flights changes over longer periods of time.

5.3 Tail Estimation Techniques

Density techniques in the context of semi-supervised classification rely on tail-estimation methods to make predictions about the behavior of future data based on a learned model. In general, a high volume of normal data will be used to generate a distribution from which that data is assumed to be drawn. This model can then be used to generate confidence boundaries that delineate normal/abnormal behavior and predict the likelihood of future samples occuring a certain distance away from the normal clusters.

In order to apply these methods, two main criteria must be met - the data being analyzed must be relatively stationary and it must conform to a known distribution. Both of these aspects are addressed in the following sections

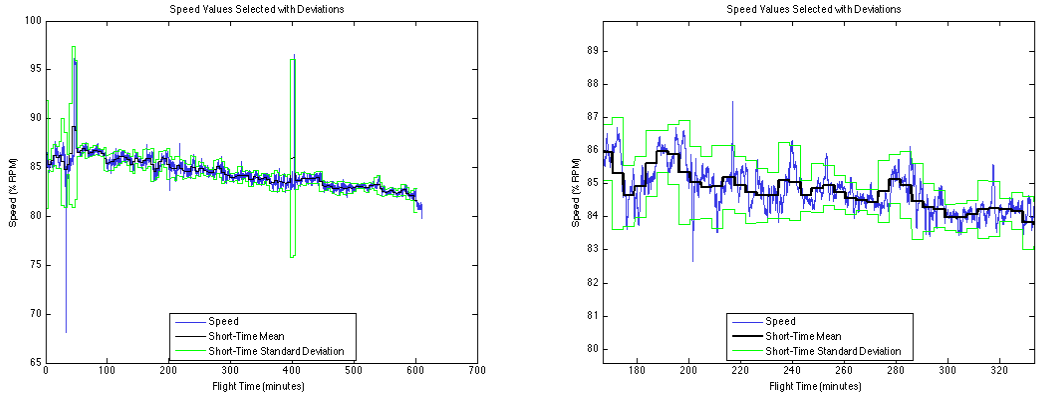
5.3.1 Stationarity Considerations

Wide Sense Stationarity dictates that a given stream of data has a constant mean and autocorrelation function with respect to shifts in time. In practice, this is an exremely

difficult criterion to check, much less fully satisfy. As was seen in Section 5.1, most of the characteristics of the given data (especially the three types of irregularities) are in direct contradiction to these notions, but the stationarity requirement will be verified visually and made more strict with the pre-processing of data. In order to satisfy the requirements for density fitting, however, the entire dataset need not be considered at once. Stationarity on a local time scale will be necessary so that a collection of adjacent samples corresponding to a fixed speed can be considered together, but this does not need to be longer then about 1 minute (64 samples). In the following, a window of 4 minutes (256 samples) is considered when verifying short-time stationarity, while the more rigorous 1 minute window will be later used for the Goodness of Fit tests.

In general, stationarity at a short-time scale can be viewed as data not deviating from the mean to within a standard deviation - all within some local time window. This kind of behavior is generally observed in the KAL data, as seen in Figure 5.9, which depics this kind of analysis for Flight 4. The left figure shows that the general outline throughought most of the flight duration includes the randomly varying speed data at the local level. The image on the right includes a detailed view of one of these areas - there are still relatively few points that stray far from these boundaries.

In order to improve on the short-time stationarity constraints the datasets are to satisfy, two pre-processing approaches were adopted in order to try to remove the more extreme regions of the data. Because of the choice of a density-based approach to this problem, there is no problem with breaking the time continuity of the data. Although it is somewhat irregular to remove "problematic" portions of the data, this can be justified as a de-noising step that aims at removing transients that intuitively should not be present in a clean data set. This removal will be addressed appropriately in the results.



(a) Flight 4 Speed, With Short-Time Bounds

(b) Flight 4 Speed and Bounds, Detail

Figure 5.9: **Flight 4 Speed and Short Time Boundaries** - the figure shows speed sensor readings (blue) bounded by local means plus/minus a standard deviation. This is done for the duration of the flight (left) and for a local window (right). Windows of approximately 4 minutes (256 samples) were used, under the assumption that, in general, flight conditions will be short-time stationary within these time frames, during cruising conditions.

5.3.2 Removal of Extrema

The first approach to improving the stationarity characteristics of the dataset is to remove portions of the data that have high first derivatives (or first-order differences). This is justified because En Route data should not contain sharp spikes that indicate rapid changes in speed rather than a gradual flight at cruising altitude. The explanations for these transient readings range from turbulence to rapid maneuvers the pilot needed to make - in either case, the behavior is not representative of the general breakdown trends under investigation.

The removal was done by simply computing a first-order difference and selecting samples whose absolute value derivative exceeded a pre-specified threshold (which was chosen by inspection, with respect to the mean and standard deviation). the 5 samples to the left and right of any such point were also removed as a means of eliminating the

entire impulse event, not just its center. Samples removed using this method can be viewed in the left portion of Figure 5.10.

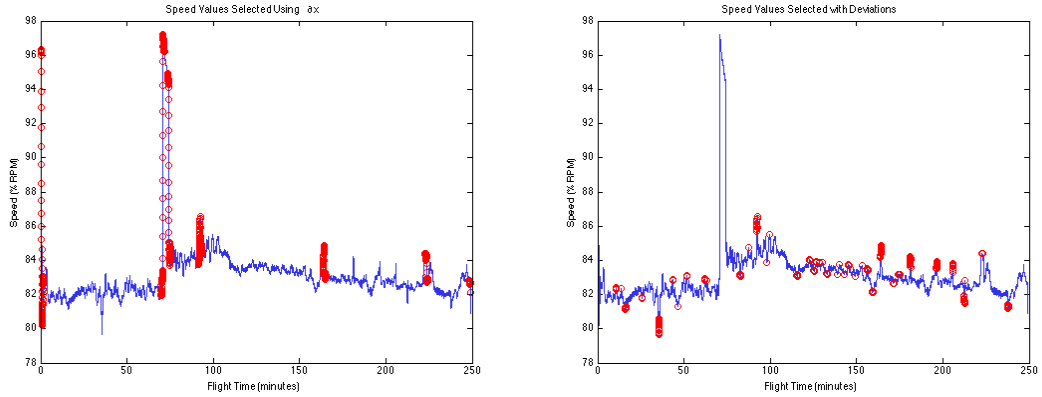
The second approach is to remove portions of the data that have relatively high deviations (above two standard deviations) from the local means. This method aims at removing the "plateau" irregularities mentioned in Section 5.1. The assumption is that the pilot may increase or decrease the speed of the plane for a short duration of the flight in order to enact a course change, avoid a weather pattern, or speed up a leg of the journey - these are not behaviors typical of long-term En Route behaviors being analyzed, and can be ignored in the context of the entire flight. Each individual pleatau is actually a separate "En Route" realization, but their treatment is something not currently addressed (see Section 5.5).

This type of data is removed as simply as it is detected - by finding a sample which deviates from the local mean by at least two standard deviations and removing all subsequent samples until they fall to within one standard deviation. There is a similar 10 sample window to allow for transitional phases. Samples removed using this method can be viewed in the right portion of Figure 5.10.

5.3.3 Single-Dimension Goodness of Fit (GoF) Tests

A goodness of fit (GoF) test is a statistical test used to check for the conformance of a specific realization of data to a particular distribution - a concept perfectly suited for verifying that the stationary data is derived from a probabilistic density. There are several popular tests for this, most common of which is Pearson's Chi-Square GoF Test.

The hypothesis being tested is conformity of the vibration sensor data to a Gaussian distriution. Model parameters (mean and variance) are estimated using maximum likelihood estimators:



(a) Flight 4 Speed, With Short-Time Bounds

(b) Flight 4 Speed and Bounds, Detail

Figure 5.10: **Flight 4 , Removal of Derivatives/Deviations** - in this figure data points exhibiting high derivatives (left) and/or high deviations (right) are labeled. In order to more closely conform to the definition of short-time stationarity, these points were removed (by replacement with local 5-7 sample means) from the full speed data shown in Figure 5.9.

$$\begin{aligned} \hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \hat{\sigma^2} &= \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{\mu})^2 \end{aligned} \tag{5.1}$$

Using the before-mentioned 1 minute (64 sample) windows, the short-time stationary samples (selected using the methods mentioned in the previous section) for each of the vibration sensors were checked for a distribition fit to a Gaussian. Figure 5.12 shows sample results for two of the five flights, while Table 5.11 details the results for all flights and applicable sensors.

From the results in Table 5.11 and Figure 5.12, it can be concluded that 4 of the sensors definitely pass the test and 2 partially pass the test. Sensors 13-16, although having abysmal results in Figure 5.12, are actually inapplicable for this (or any) Goodness of Fit test. Upon examining the kind of data recorded by these sensors, it was discovered that these four sensors had particularly coarse quantization resolution, which resulted in

Flight	Vibration Sensor Number					
	5	6	9	10	17	18
1 (A1)	80%	86%	85%	86%	60%	54%
2 (B1)	77%	72%	75%	71%	58%	57%
3 (B2)	89%	84%	80%	84%	19%	40%
4 (C1)	86%	84%	85%	91%	30%	47%
5 (C2)	90%	81%	90%	89%	30%	37%

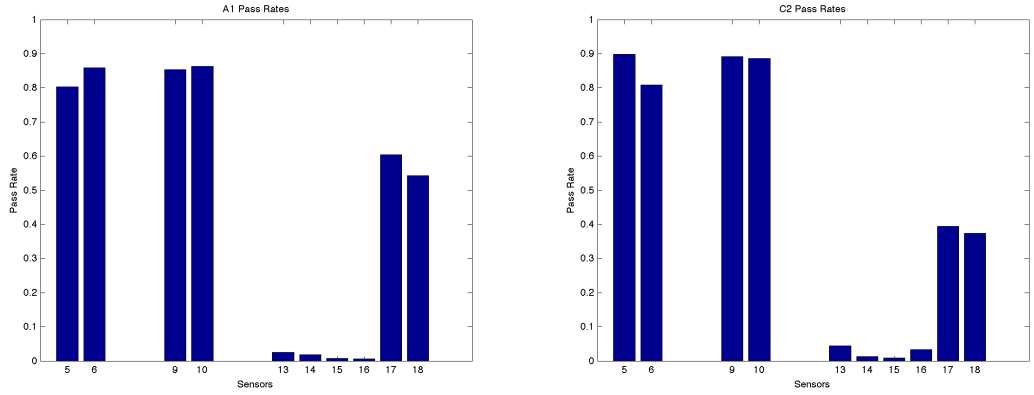
Figure 5.11: **Goodness of Fit Summary** - summary of the distribution fitting results, indicating relatively good fits for sensors 5, 6, 9, 10, 17, and 18. Each cell contains the percentage of the entire dataset which passed Pearson's Chi Square Goodness of Fit test, with more colored cells indicating a higher proportion of the dataset passing the test. The remaining 4 sensors had data that did not apply to the test (for details, see the end of Figure 5.12).

the majority of their recorded values being one of three or four distinct quantities. This low number of bins violates the constaints required by all popular GoF tests, making it impossible to tell which distribution these values are truly from.

Despite these mixed results, the rest of the chapter will proceed under the assumption that all sensors passed the GoF test and can be assumed to come from Gaussian random variables.

5.4 Discussion of Results

Under the assumptions and framework outline above, the method of leveraging the data to detect problems is most readily applied to the single-dimensional case: given a particular speed, the most probably distribution of sensor readings is computed and compared to live data. The Gaussian Quantile Function (Q-Function, Equation 5.4 can be used to estimate boundaries outside of which the occurance of sample readings is less and less probable, allowing for a rigorous method of calculating risk.



(a) Flight 1, GoF Results

(b) Flight 5 GoF Results

Figure 5.12: **Goodness of Fit Results** - the figure provides sample Goodness of Fit results for Flights 1 (left) and 5 (right), with respect to a Gaussian distribution. Note that for each flight, 4 of the sensors have extremely low pass rates as a result of the inapplicability of the test to the majority of the samples. See notes at end of Section 5.3.3 for a full explanation.

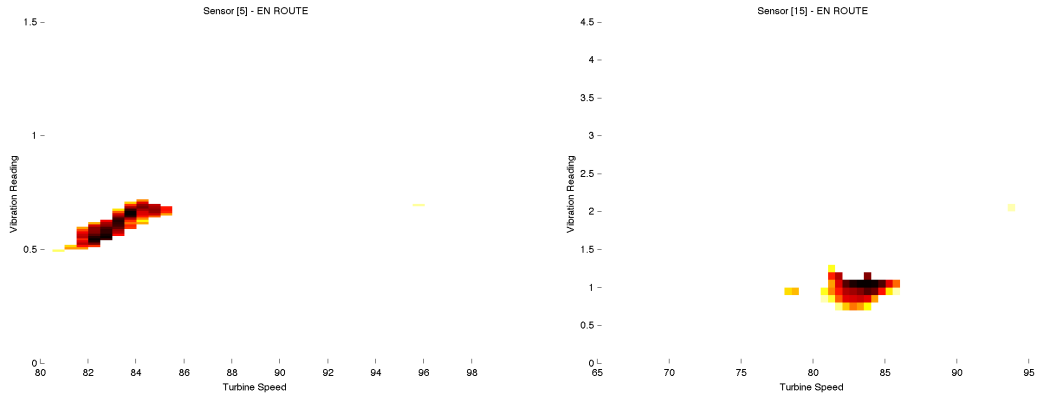
$$Q(x) \quad = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du \tag{5.2}$$

The main problem with this approach is the variability in the speed of live data; while a few of the datasets may be amenable to this one-dimensional approach, the general case will not be as simple. To attack this difficulty, two separate two-dimensional views of the data are being developed, each of which is described below.

5.4.1 Speed-Vibration Comparisons

The first method being developed plots speed and vibration data simultaneously for the entire flight. There is generally a relationship between the two, but there is enough variability and unpredictability in just the flights investigated here to warrant further investigation.

Figure 5.13 shows a pair of such plots, which demonstrates how well clustered this data generally is in the speed/vibration space. It should be noted that for each vibration sensor, the appropriate turbine speed is chosen for the plot and that there are still "transients" present in the clusters, despite efforts to remove them via the two-step process detailed above.



(a) Flight 3, Sensor 5

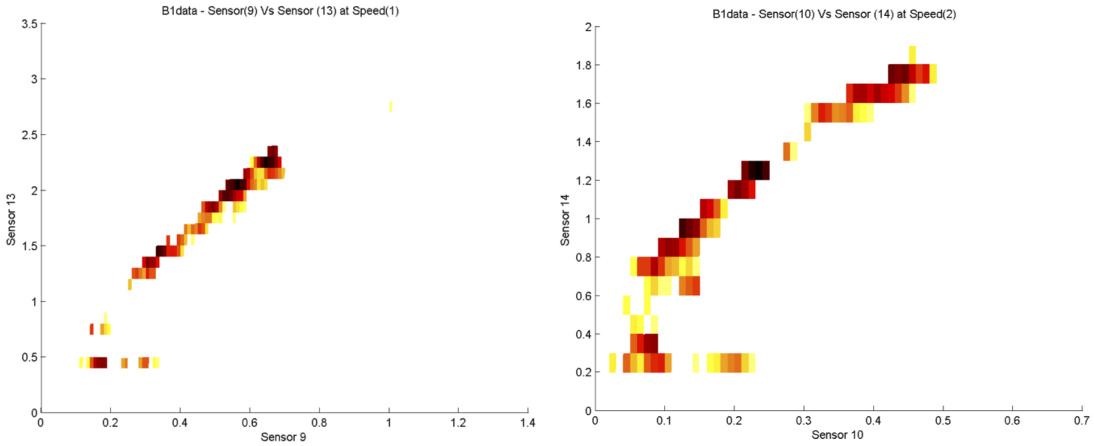
(b) Flight 5, Sensor 15

Figure 5.13: **Sample Speed/Vibration Plots** - this figure shows a pair of sample speed/vibration plots. Most of the flights, including Flight 3 (left) and 5 (right) exhibit data that has well-defined clusters like the ones above, allowing for a high degree of description of the data with a fitted Gaussian distribution.

5.4.2 Vibration-Vibration Comparisons

A second method of analysis is to compare pairs of vibration sensors in two dimensional space. This naturally encounters the same problem as a one-dimensional analysis of vibration data - the inability of incorporating variable speeds - but for flights with relatively constant speeds (or large regions of constant and similar speeds), these comparisons are possible. The most useful application may indeed be to compare the characteristics of these plots for the same flight and sensors, but different engines.

Figure 5.14 shows a pair of sample plots, with just this kind of comparison. Even though both sensors are placed in symmetric locations on the plane and their recordings correspond to the same time series subset, there are noticable differences. This confirms the assumption in Section 5.1, since at least the stresses exerted on each engine during turns are different enough to yield such discrepencies.



(a) Flight 2, Left Engine

(b) Flight 2,Right Engine

Figure 5.14: **Sample Vibration/Vibration Plots** - the figure shows two sample vibration/vibration plots, each taken from Flight 2 and both comparing the same sensor pair on either of the two engines. The amount of dissimilarity suggests a high degree of independence between the two engines, despite their relatively similar inputs.

5.4.3 A Method for Failure Detection via Outlier Tests

Considering the (relatively well confirmed) assumption that vibratio data has a Gaussian distribution, there are several ways of detecting abnormalities or problems that occurred during the course of a recent flight. Each of these depends on establishing a predetermined boundary of healthy operation and subsequent levels of lower confidence that correspond to less and less probably sensor readings. A 2-dimensional example of this type of approach is shown in Figure 5.15.

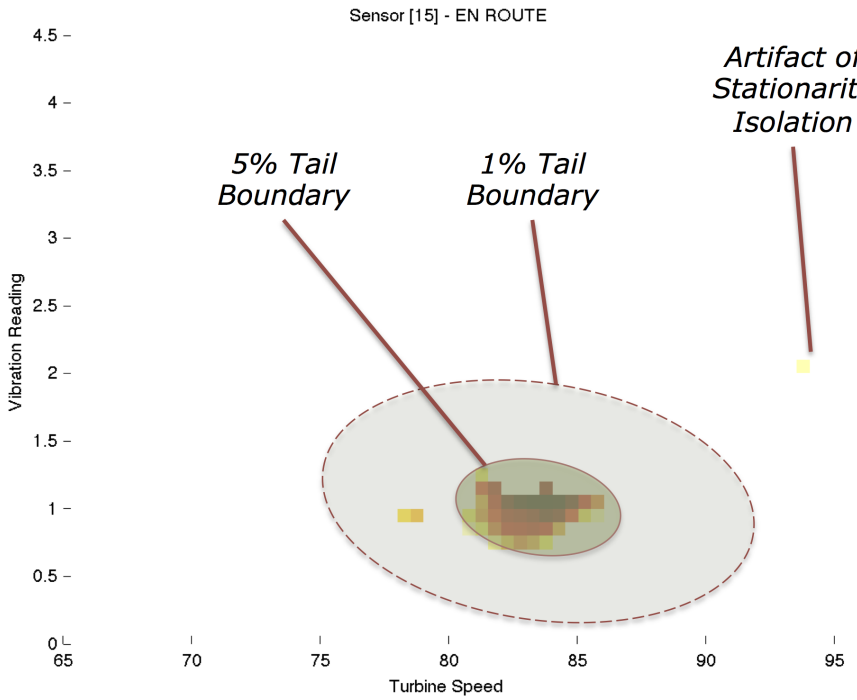


Figure 5.15: **Failure Detection via Outlier Testing Concept** - this figure demonstrates a sample procedure for the detection of component failures via outlier detection on a 2-dimensional plot. Here, it is the Speed/Vibration plot for Flight 5, Sensor 15, which indicates rough 5% and 1% tail boundaries for the 2-dimensional Gaussian distribution. Values within the 1% boundary but outside of the 5% boundary are likely outliers that can pose no significant problems, but values outside of the 1% boundary should be carefully examined as they may potentially be significant faults or anomalies.

5.4.4 A Method for Detection of Performance Degradation

An alternate use for the tools developed in the previous sections is the long-term health monitoring of engines. As wear-and-tear take their toll on the integrity of the components, it is highly beneficial to perform maintenance on those components that need it most - recording and comparing the way the sample distributions of vibration readings change over time is likely a cost-effective and efficient way to do this.

A proposed sample is shown in Figure 5.16, which shows how a potential distribution decays over time. As parts of the engine degrade and wear out, there is less

coherence in the way vibrations correspond to particular speeds and there are likely to be significant changes to the way these distributions look - whether they are more flat (as demonstrated in the figure), split into more clusters, or translated throughout the sample space.

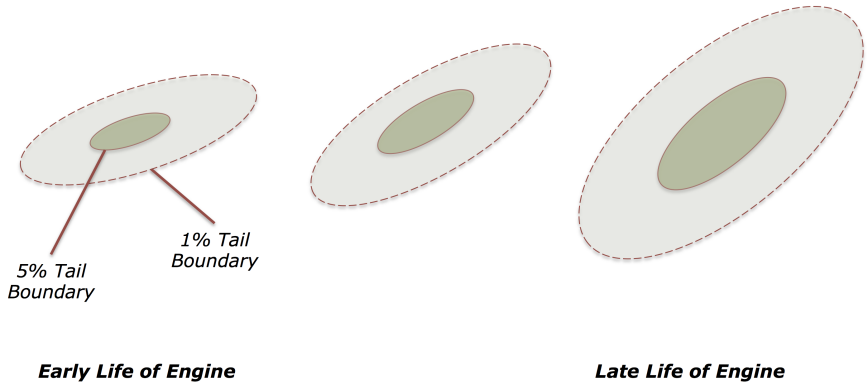


Figure 5.16: **Detection of Performance Degradation Concept** - this figure shows the concept for a system that anticipates wear-and-tear failures by analyzing how various sensor value distributions change over time. Here, a 2-dimensional vibration/vibration space decays from a well defined region to a spread out cloud of points as the engine slowly degrades.

5.5 Summary

In this chapter, a method for the analysis of low-frequency vibration sensor data was proposed and introduced. Despite many practical issues related to the data and challenging assumptions, the proposed system is seen to concisely represent the health of an engine, being able to distinguish normal and stationary behavior from several types of irregularities. Once more data is available for broader testing, it is likely that the long-term analysis of vibration data will yield similarly fruitful results.

Further development of the long-term applications of the methods proposed in Chapter 5 require a larger set of sample data, but the currently proposed approach would benefit from additional work in the following areas:

- **Inclusion of time-domain information**. While the initial approach disgards the time-domain relationships between samples, it may benefit from the inclusion of time-domain information in the strictly density-based approach currently being utilized. There is no doubt that time series data contain some correlations and attempting to leverage these to make better decisions will be key in improving this algorithm

- **Improvement of stationarity analysis**. A more precise treatment of stationarity would not only benefit this work theoretically, but allow for better selection of the parts of the flight that should be treated as having En Route characteristics. Extracting additional information from the platau irregularities mentioned in Section 5.1 and Section 5.3.1 could also benefit from this.

- **Finding a relationship between the HPT and LPT speeds**. It stands to reason that there should be some way to relate these two, rather than treating them as simply independent of each other in the context of what stimuli they enact on the vibration sensors.

Chapter 6

Advanced Gas Path Analysis Methods

6.1 Introduction

As discussed in Chapter 1, the traditional Gas Path Analysis (GPA) approach that has been utilized with reasonable success for the last several decades generally uses small amounts of infrequently collected data to trend the health of an engine over time. GPA is a mature technology (started in the 1970s) and is widely accepted both in the military and commercial domains of aerospace.

The fundamentals of the method are that it is a way for assessing the magnitude of engine performance *changes* at the component level, based on observations of gas path parameters. The important thing to note is that it is a relative analysis (always in reference to something, usually average fleet performance). All of the current methods of performing GPA are based on the same basic premise, which is described at a high level in Figure 6.1 [Vol13].

In practice, the intuition is that, at steady-state cruise, averaging over several minutes of samples yields a single reference vector for a particular flight that is "good enough" for detection of component degradation.

Improvements to this traditional approach from the perspective of mechincal engineering have taken the newly available full-flight data from QAR modules and applied it to a physical engine model. The GPA component of this evolved system is computing an estimation of degraded performance based on the data (observations). While (with

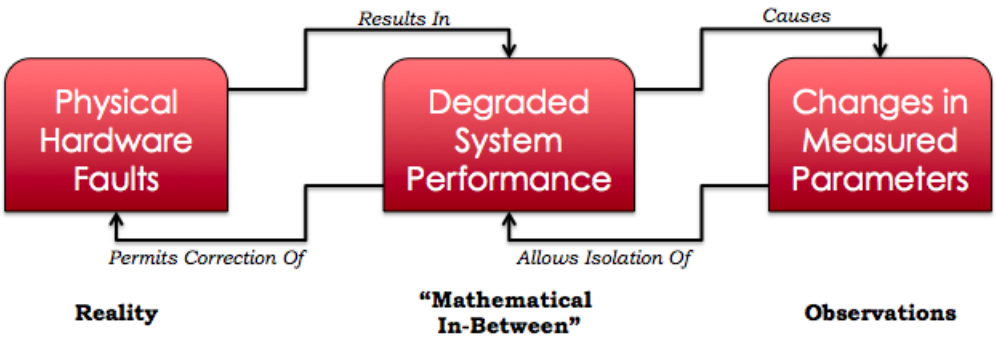


Figure 6.1: **Philosophical Overview** - this is the relationship between the state of components and observed parameters. Imperfect observations of degraded system performance can hint at fundamental physical hardware flaws, but reality is rarely this direct.

some additional learning components) this advanced version of the traditional model improves detection performance significantly, it does not account for instrumentation bias or differences, model inadequacies, or parameter normalization errors. The chief drawbacks, however, are the high computational cost and limitation to steady-state portions of flight for which the adaptive filters can sufficiently stabilize.

Our challenge is to, first, look at this problem from the transient point of view and try to reduce computational complexity where possible. This perfectly complements the lack of physical model understanding, so that the data is approached from a very fundamental signal processing perspective.

Initially, we are not only interested in the extremely difficult problem of performing detection of faults during transient portions of the flight *with the faults occuring within those transients*. It is much simpler, and more generalizable, to consider faults that happen outside of the transient regions (usually in an adjacent steady state area), and to see whether our approach can detect these within the steady state and the transient regions overall.

6.2 Simulation with CMAPSS

The fundamental datasets for these experiments were to be QAR flight records (recorded at 1Hz, for selected GPA parameters), which we had access to through Korean Airlines (KAL). The main conundrum was how to acquire data with faults; no airline would be willing or able to introduce problems into their own engines for data generation purposes, and the repository of historical flights with *recorded* full-flight data is limited to handful of instances.

A better path would be to somehow simulate faulty data along with nominal flights, develop a process that would be able to distinguish between the two, and then find a way to relate the simulated data to real-world QAR data. Unfortunately, nearly all commercial and military engine simulators are closely guarded secrets and it is unlikely that even the closest of partners would have the resources to generate a significant corpus of such data for general experimentation. Fortunately, there are a few experimental (and semi-open-source) engine simulators that can be used as very close substitutes.

During the course of work on jet technology, NASA developed a series of such simulators for fictional, theoretical engines that have recently gained enough stability to be reliable and realistic. The Commercial Modular Aero-Propulsion System Simulator (CMAPSS) [LFD+12] is the primary foundation upon which these projects are built, and includes a few variants (such as the 40K version [Lin12]). This software package is a series of MATLAB scripts and Simulink models emulating the physical behavior of a 90,000lb thrust N1-controlled twin bypass turbofan jet engine, and comes with the added benefit of a rudimentary Transient Test Case Generator (TTCG) that facilitates generation of batch data [Arm10].

The list of available simulators is shown in Table 6.2. Along with the two main types of software, the control and thrust parameters are listed, as these are the two variables

Simulator	Control Type	Thrust
C-MAPSS v2	N1	90,000
C-MAPSS 40k	EPR or N1	70,000

Figure 6.2: **List of Available Simulators** - table shows the general purpose simulators available at NASA, along with corresponding control and thrust characteristics. The CMAPSS version 2 was selected because of the availability of the Transient Test Case Generator software addon that simplifies generation of large datasets. The control type refers to the category of controller that is commonly used in turbofan jet engines, where the feedback can either be from the low-pressure (N1) subsystem or based on the Exit Pressure Ratio (EPR).

which mostly define the performance signature to which we'd like to match a real engine . There will never be an ideal match, because the NASA models are extremely generic simulators that were not designed to correspond to any existing machine.

Table 6.3 shows the corresponding parameters of real engines that KAL was able to retrieve QAR data from. While the PW4090 was a closer match to the thrust rating in CMAPSS version 2, it turned out that many of the parameters needed later for standard day correction and normalization for this engine would be impossible to retrieve, as that proprietary data was owned by another manufacturer. We decided that the GP7270 was the closest remaining fit, and matched the N1-control mechanism native to CMAPSS v2.

The earliest versions of these models were released in 2009-2010, and have been slowly evolving since, so there is not a large corpus of existing research that utilized them. Previous work in this area of general commercial simulation has largely been limited to internal projects [CMLG11], [CMGL12].

Engine	Control Type	Thrust
PW 4056 v2	EPR	56,000
PW 4062	EPR	62,000
GEnx 2B67	N1	67,000
PW 4090	EPR	90,000
GE90-115	N1	115,000
CFM56-7B24	N1	115,000
GP 7270	N1	70,000
PW 4168	EPR	68,000
PW 4170	EPR	70,000

Figure 6.3: **List of Available Engines** - this table details the engine types from which QAR data is available (courtesy of Korean Airlines, KAL), along with the corresponding control and thrust characteristics. The implicit best match, with respect to the simulators shown in Figure 6.2 is shown in bold: the GP7270. Control Type is more important to match perfectly, but the differences resulting from a thrust rating mismatch can be more easily mitigated (as will be discussed in Section 6.3.3).

6.2.1 Relevant Parameters and Nomenclature

Having a simulator is half the battle - we also needed input data to drive it, independently of the QAR records, in order to verify pure simulation results. CMAPSS has a few very rudimentary scenarios (basic takeoff, cruise), but nothing resembling the complexity of real-world flights, an example of which is depicted in Figure 6.4.

Figure 6.5 depicts a standard twin-bypass turbofan jet engine, with the prominent components labelled. From left to right, air passes through the system, while fuel is injected into the combustor (flow is regulated indirectly by the pilot, through appropriate controller commands). The compressor and turbine are each additionally separated into "low-pressure" and "high-pressure" subsystems, and it will be common to discuss the Low Pressure Compressor (LPC), High Pressure Compressor (HPC), High Pressure

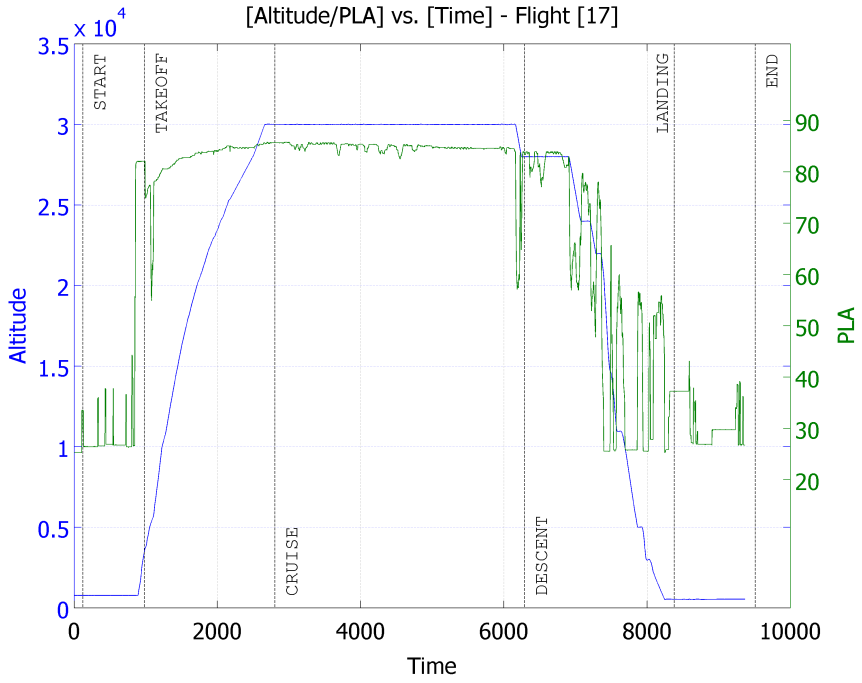


Figure 6.4: **Sample Flight Parameters** - Annotated plot of altitude and (adjusted) PLA versus time (in seconds) of Flight 17 from the NASA FOQA dataset. While altitude broadly tracks PLA, it is obvious that PLA is a much more complicated inputs signal that changes radically, especially during the descent andlanding phases of flight.

Turbine (HPT), and Low Pressure Turbine (LPT) in this parlance (notice that the highest pressure is, logically, located near the combustor, while the lower pressure subsystems are closer to the inlet and outlets). A somewhat more detailed version of this can be seen in Figure 6.6, which depicts a cross section diagram with labeled parameters of interest.

Based on this, it is important to organize the flow of time within the turbine, so as to determine which variables will lead or lag the others in time. The causal relationships between the variables are to group them as follows:

1. *Environmental* - Ambient Pressure and Temperature

2. *Input* - Altitude, TRA, and Mach Number

3. *Controller* - fuel flow, VSV, VBV

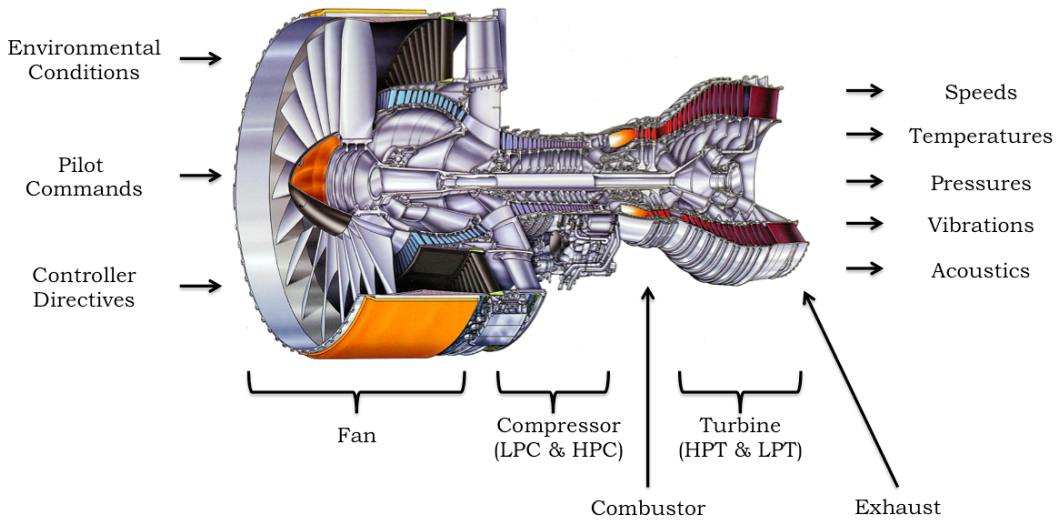


Figure 6.5: **Jet Engine Diagram** - Annotated diagram showing major turbofan jet engine components, as well as input and output variables of interest. Pilot commands (PLA/TRA) are generally transferred to the engine via controller directives (see Figure 6.6), but altitude and speed have an impact on performance. Fuel is injected into the combustor, which drives the high pressure (N2) and low pressure (N1) subsystems in the compressor, turbine, and fan.

4. *Compressor/Combustor* - core speed

5. *High Pressure (N2) System* - T30, Ps30, T48

6. *Low Pressure (N1) System* - T24, P24

7. *Fan System* - fan speed, T2, P2

Figure 6.6 provides a graphical representation of the descritpion above, which provides some additional clarity. The reason that the High Pressure (N2) system is affected jointly after the combustor stage is because the HPC and HPT are physically linked, and rotate at the same speed. Similarly, the LPT and LPC rotate at the same speed, being linked to the same internal shaft. Thus, outside of the environmental and input variables, the effects of input parameters are causally flowing from the combustor *outward* towards the fan and exhaust.

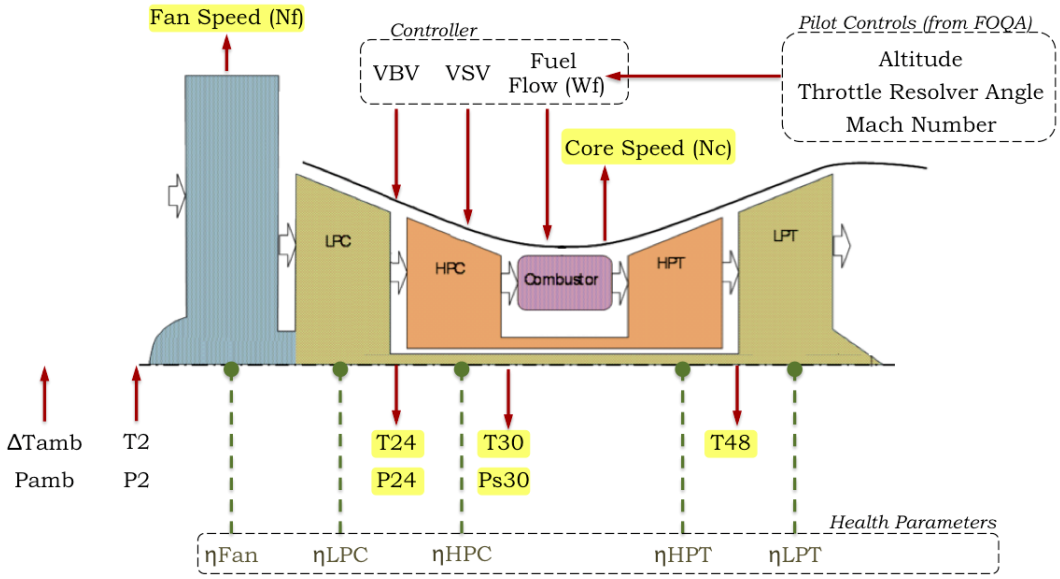


Figure 6.6: **CMAPSS Simulated Engine Diagram** - Annotated diagram showing a turbofan jet engine cross-section, with labelled components, as well as input, intermediate, and output variables of interest. Component health parameters, which are used to define the level of fault or degradation, are also shown at the bottom. The seven parameters highlighted in yellow are the output parameters for the system, while the three controller commands are considered to be "intermediate" parameters. All other variables, excluding component health parameters, and considered to be inputs.

The pressure and temperature variables, denoted with capital T and P letters, have station numbers that originally ranged from 1 to 5 in old propeller planes. As the engine grow outwards and became more complex internally, intermediate stations such as T2.4 were abbreviated as T24 (because of this, T30 and T3 are equivalent notations). Lastly, T48 or T50 is sometimes also referred to as EGT (Exit/Exhaust Gas Temperature), and is actually measured right before the final LPT stage; the temperature at the actual exhaust is so great that no sensors survive there for long, so it is hardly ever measured.

The primary controls over the level of degradation within a CMAPSS simulation will be through the health parameters depicted in Figure 6.6. There are other points of failure that could be considered (sensor and actuator faults), but we have limited outselves to a reasonable level of complexity. Each health parameter also has an associated flow

capacity that is linked in a 1:1 correspondence to health; we do not change this ratio in our simulations (see Appendix C for more information).

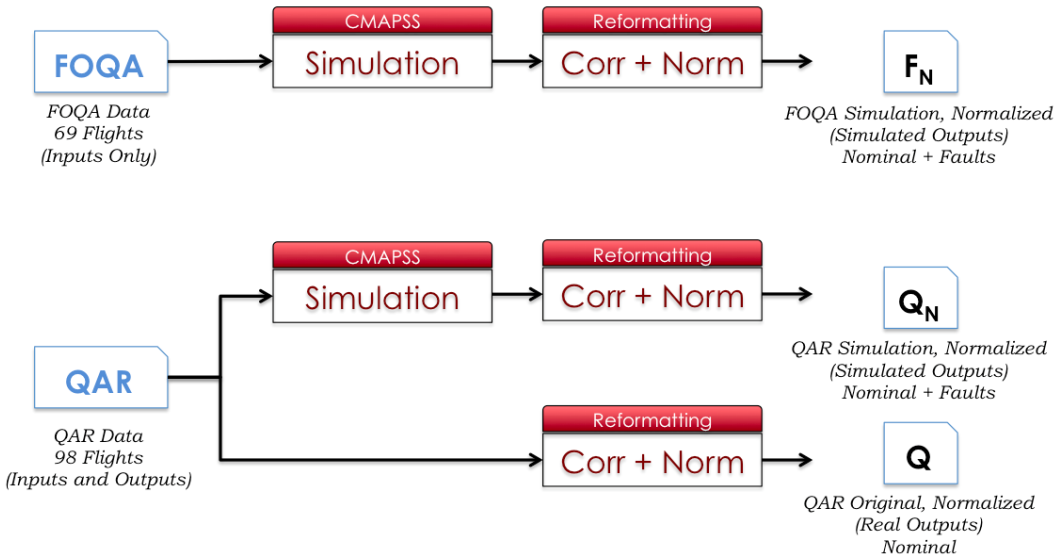


Figure 6.7: **Dataset Nomenclature** - this diagram shows an overview of the datasets and primary operations that will be performed on them, along with corresponding outputs. Input variables are used to drive CMAPSS simulations, which can produce a variety of data with synthetic faults. All input and output data is standard day corrected (Section 6.3.2) and normalized (Section 6.3.3), which allows for comparison of simulated and real world data for the purposes of system verification*.

As with all simulations and experiments, data is necessary to validate the pertinence and applicability of any new methods. We were fortunate to be provided with two such sources of information: Flight Operational Quality Assurance (FOQA) data from NASA and Quick Access Recorder (QAR) data from Korean Airlines (KAL).

FOQA data consists of 69 anonymized records that contained the 4 primary parameters needed to drive CMAPSS: Altitude, Mach Number (MN), Total Air Temperature (TAT), and Power Lever Angle (PLA). Note, that PLA is equivalent to Throttle Resolver Angle (TRA) and as both of these terms are used in relevant literature, they will also be used interchangeably here; both essentialy measure how far on the throttle (0-100%) the

pilot is pushing. Also, it's important to note that some modification to these four parameters is needed, as detailed in Section 6.2.3). As seen in the diagram in Figure 6.7, FOQA input data is used to drive CMAPSS and produce a collection of datasets labeled F_N (the N corresponds to different experimental conditions, see Section 6.2.4).

QAR data from KAL consisted of input and output parameters, with the goal of system verification in mind. These 98 flights were taken from 6 different planes, all of which were equipped with GP 7270 engines that would closely match with the CMAPSS simulator parameters (see Figure 6.3). The details of each flight were anonymized, so that only relevant input/output information was available - though the data consisted of pairs of flights, with odd numbered flights corresponding to even numbered return flights. As seen in Figure 6.7, QAR input data was used to drive the CMAPSS simulation and yield simulated nominal and fault data in the form of Q_N data (here, N will indicate the index of simulated datasets, as each dataset's experimental conditions are slightly different, see Section 6.2.4). The corresponding Q_N output data will then be compared to appropriately corrected and normalized real-world Q output data, in order to establish what level of correspondence exists between the two.

Generation of synthetic F_N and Q_N data is necessary for the purpose of simulating faults within nominal real-world data and using this to design a system that can detect and identify these faults. Initial system design was performed using NASA FOQA F_N data, because that is the first dataset that was made available to us, though similar system design was also performed with QAR Q_N data.

*** NOTE**: One final aspect of this experimental setup that needs addressing is the sampling rate. As was discussed at the introduction of QAR vibration data (Section 5.2), the standard sampling rate for these devices is 1 Hz. While CMAPSS can take input at any sampling rate (even non-periodic rates, since all it requires is a set of time:parameter

pairs that correspond to new input values at different points in the flight), its native output is a datastream sampled at 66Hz. For the purposes of compatability in this experiment, we down-sampled this 66Hz signal to *approximately* 1Hz using a factor-of-64 decimation filter. Using power of 2 decimation filters is much faster and easier (especially in hardware), but it does introduce a slight misalignment between the data we simulate and what is available in the real world QAR data. When designing detection and diagnosis algorithms, we ignored this difference (deeming it relatively insignificant at this scale), but when performing validation with respect to real-world data, we interpolated the 1.03125 Hz signal to match the 1 Hz signal from QAR data (see Figure 6.21). More on this operation can be found at the end of Section 6.2.4.

6.2.2 Definition of Engine Transients

When considering the task of transient fault detection, it is useful to discuss what a transient is, even after the brief introduction of stationary and transient behavior in Section 3.6. Speaking practically, a transient is any significant non-stationarity, or area when the main engine parameters change *quickly*. This definition of speed will necessarily have to be empirical, because the data will only be generated once every second, but an abrupt change over the course of several seconds is enough to qualify as a transient that might affect engine operation.

An appropriate starting point is to examine the PLA/TRA data stream and label a region as transient if the angle changes by at least 5-10 degress. This is a simple view, because what we really want to look at is the transient operation of the engine, but it takes some finite time before the requested angle propagates from the pilot, through the controller that changes fuel flow, and then into the compressor which first affects the HPT (N2, spinning the core or Nc) and then the LPT (N1, spinning the fan, or Nf). Thus

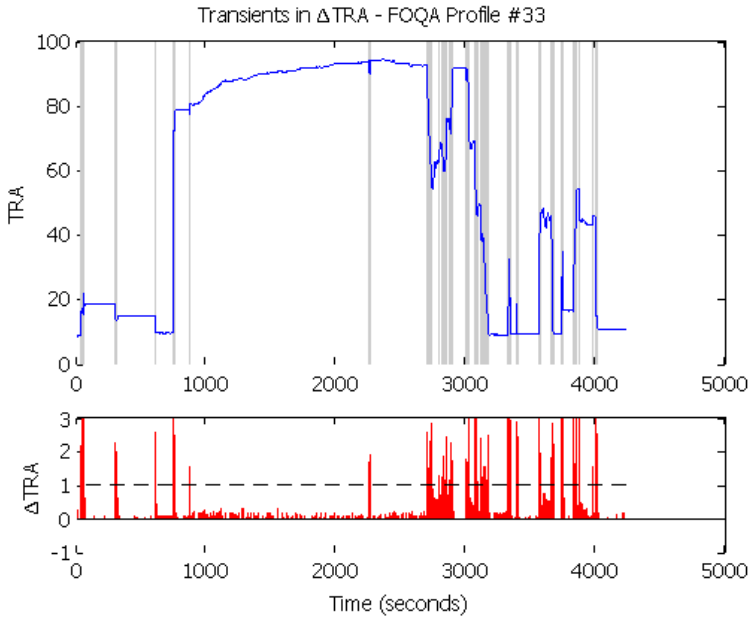


Figure 6.8: **Sample Transient Detection Plot**, for FOQA Flight 33. This figure shows an overview of the entire flight, with transient regions labeled in light gray in the top graph. The bottom plot demonstrates the technique used for automated transient region labeling, via a pre-set threshold on the changes in TRA (red bars)

a more appropriate process will take into effect the lag between PLA/TRA and N1 (the fan is much larger and takes a longer time to move into and out of steady state), and use derivatives of N1 and N2.

Based on initial experiments, the threshold will more likely be around 1 or 2 degrees, because a 1 degree resolution, even at the 1 second time scale, yielded very sparse transient areas. All plots here use a threshold of 1, as show in Figure 6.9, in order to make the effect of this change even more apparent.

In order to account for the time lags to different components, the approach was to add a padding "factor" that would allow the transient effect to propagate for a few seconds after it first occurred within PLA/TRA. At any moment when there was a transition from transient back to stationarity, we extended the transient region by 10 seconds (empirically obtained).

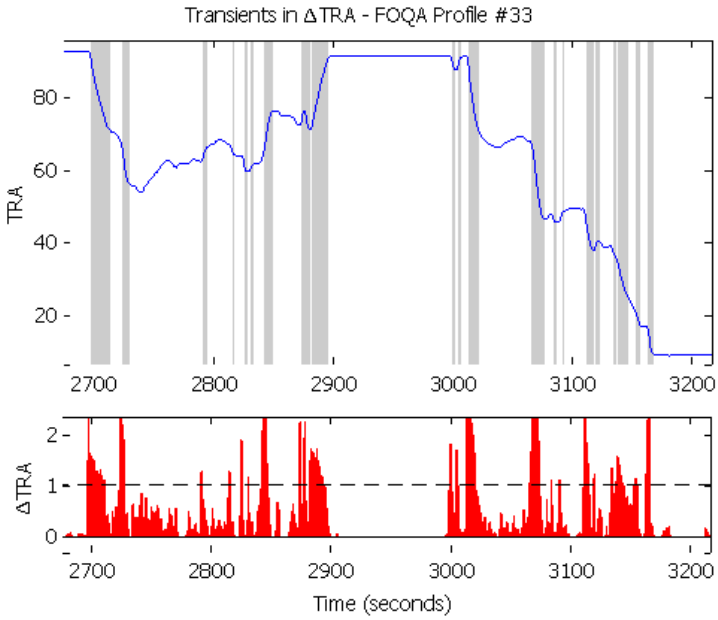


Figure 6.9: **Sample Transient Detection Plot Detail**, for FOQA Flight 33. This figure presents a detailed view of the time between 2700 and 3200 seconds into the flight, showing actual TRA (in blue) with the proposed labeling of transients (gray regions) in the top plot. The bottom plot shows the first order difference of TRA at corresponding times, with the threshold to which these differences are compared (dashed line).

Another problem visible in Figure 6.8 is the presence of non-contiguous regions that are in close proximity to each other. In reality, most human observers would combine these areas if they were "close enough", we we included additional pre-padding logic that checked for how close two adjacent regions were before combining them. A small adjustment of about 3-4 seconds can account for these discrepencies, with sample result of this modification visible in Figure 6.10.

6.2.3 Pre-Processing of Input FOQA Data

We must convert from base the FOQA variables into parameters that are expected by CMAPSS. This entails the transformation of Total Air Temperature (TAT) into delta ambient temperature (dTamb) as well as an approximate linear scaling from PLA to

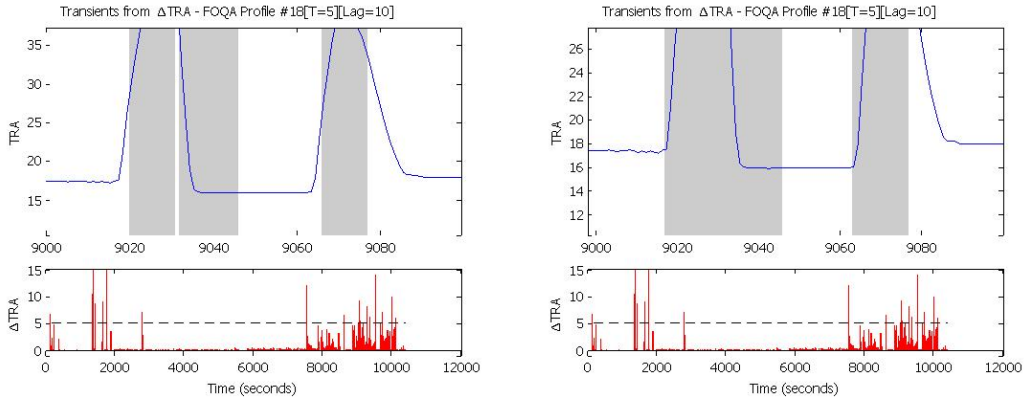


Figure 6.10: **Demonstration of "Prepending"** - the figure on the left shows a sample segment of the TRA plot with the prepend functionality turned off. Notice that for two of the transient events, there's a lag between when they really start (blue line goes up) and when they are labeled as transients (gray background label). In the figure on the right, the prepend functionality is enabled with a lag of 4 seconds, which makes the resulting labels much truer to reality.

TRA (thrust). All temperatures used in CMAPSS should be in Rankine, altitudes should be in feet, pressures in pounds per square inch (psi), and speeds in rotations per minute (rpm).

$$dTamb(degC) = Tamb(degK) - Tamb_{ISA}(degK) \tag{6.1}$$

Where:

$$Tamb(degK) = \frac{(TAT(degC) + 273.16)}{[1 + 0.2Mach^2]} \tag{6.2}$$

$$Tamb_{ISA}(degK) = 288.16[1 - 0.0000068793Alt] \tag{6.3}$$

For the PLA to TRA conversion, we will assume a roughly linear conversion that will only depend on what the minimum and maximum values of the CMAPSS TRA values are:

$$TRA = TRA_{to,C} - \frac{TRA_{to,C} - TRA_{gi,C}}{PLA_{to,F} - PLA_{gi,F}}(PLA_{to,F} - PLA_{Input}) \tag{6.4}$$

Where in the above, the "to" stands for "takeoff" (indicating a relationship to the maximum value that usually occurs during that phase of flight) and "gi" stands for "ground idle" (indicating the idling phase that is associated with minimum power). Similarly, the "F" subscript corresponds to the referenced FOQA parameter, while the "C" subscript corresponds to the referenced CMAPSS parameter.

FOQA data must also be truncated at the start and end of the flight to exclude some data below ground idle. This is necessary because CMAPSS approximations can break down when the PLA/TRA is lower than ground idle (in our case this is empirically found to be about 10-20 %).

In order to find the ground idle and takeoff TRA values, an examination of CMAPSS sample files includes the an explicit listing of these values. To find the corresponding takeoff FOQA PLA values, the maximum value over the entire dataset was computed. Calculating a ground-idle PLA setting was a bit more invovled because of the problem listed above; the only surefire way was to manually inspect all of the FOQA files and determine the minima at the start of an appropriately high PLA segment.

6.2.4 Test Scenarios

Simulations were performed with a variety of parameter settings, while varying the location of the faults, their intensity, and the subsequent noise that could potentially be corrupting the data. Explicit details about the generated datasets can be found in Appendix C, but list of fault conditions is as follows:

1. Addition of uniformly random fault magnitudes between 0.020 and 0.040. Magnitudes with more than 3 significant digits seem to cause CMAPSS to crash, and these values hover around the general "middle intensity" of faults at 3% degradation.

2. Fault insertion could occur randomly at any point during the, flight, during the descent/landing phase (randomly picked from last 30 minutes of flight), or from the takeoff (see below for details).

3. Modification of noise addition to have more noise sprinkled throughout the data, corresponding to a 1-3% variance of the base parameter variance (using additive white Gaussian noise).

In order to more closely address the task of transient fault detection, however, we elected to insert faults directly into a region that was known to be highly transient and whose location could automatically be determined for every flight: the takeoff. Takeoff occurs during an interval of about 5-20 second between ground idle and climb, during which the PLA is increased to its maximum value as the plane accelerates along the runway and lifts off the ground.

Based on rough calculations of known engine behavior [Vol13], takeoff *starts* when the following conditions are met:

$$N1c2_{Low} - e_1 \leq N1c2 \leq N1c2_{High} + e_1 \tag{6.5}$$

$$PLA_{Idle} - e_2 \leq PLA \leq PLA_{Takeoff} + e_2 \tag{6.6}$$

$$\dot{N}2_{Low} - e_3 \leq \dot{N}2 \leq \dot{N}2_{High} + e_3 \tag{6.7}$$

Takeoff ends roughly when the plane is about 20 feet above the altitude at which taxiing and ground idle occurred. Here, it is useful to approximate $N1c2_{low}$ as corrected core speed at Idle, $N1c2_{High}$ as corrected core speed at Takeoff PLA, and to use $e_1 = 20$. We used $e_2 = 1.5$ and for the final equation, $\dot{N}2_{Low} = 100$, $\dot{N}2_{High} = 1000$, and $e_3 = 25$.

The plot below in Figure 6.11 shows an "operating region" (typically used by industry experts) composed of an altitude vs. mach number plot for takeoff regions, with solid lines tracing time from sample to sample. What is interesting is that the takeoff phase (differentiated from cruise when the plane reaches an altitude of about 20 feet above the ground) really is particularly short, containing between 20 and 50 samples (at a 1Hz sampling rate) and still looks relatively smooth.

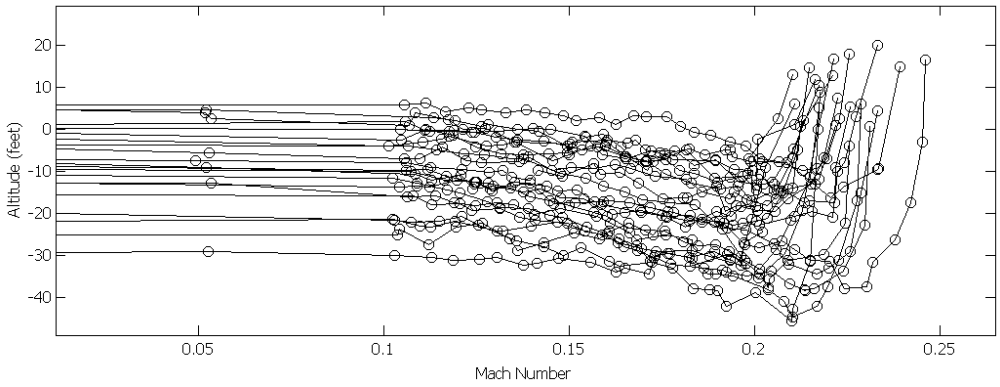


Figure 6.11: **FOQA Takeoff Characteristics** - Altitude-Normalized operating regions for FOQA flights. The graph demonstrates definite similarities in the trajectories of all flights (though these are not necessarily time-correlated). Each flight exhibits a characteristic drop in altitude associated with the speed just prior to takeoff (at about 0.2 MN).

The final post-processing step for CMAPSS simulated data is the need to downsample to 1 Hz. CMAPSS natively produces data at 66Hz, and the most expedient way to rectify this is to use a factor-of-64 decimation; which inadvertently introduces a slight timing mismatch between CMAPSS and FOQA input data. This is not relevant for the purposes of initial CMAPSS/FOQA experiments because simulation outputs will not need to align with FOQA flight profiles; in this scenario, we simply re-labeled the data to appear as if it was sampled at 1Hz rather than 66/64 Hz.

In the case of a CMAPSS/QAR comparison, this mismatch will prove problematic when looking at verification of simulated outputs to real-world outputs. This mismatch is addressed in Section 6.3.4, and can be seen more clearly in Figure 6.21.

6.3 Verification with Respect to Real-World QAR Data

Korean Airlines (KAL) provided us with a total of 98 real-world QAR flight records, all from the GP 7270 series engine, with flights of duration varying within two ranges: short (1-4 hours) and long (5-12 hours). Generation of simulation data from QAR data proceeded by extracting corresponding input parameters (the same 4 primary FOQA input parameters used in the original simulations discussed in previous sections).

The diagram in Figure 6.12 outlines the process that will be required for computing a correspondence between real-world QAR outputs and simulated CMAPSS outputs, in an effort to verify that the two datasets are interchangeable.

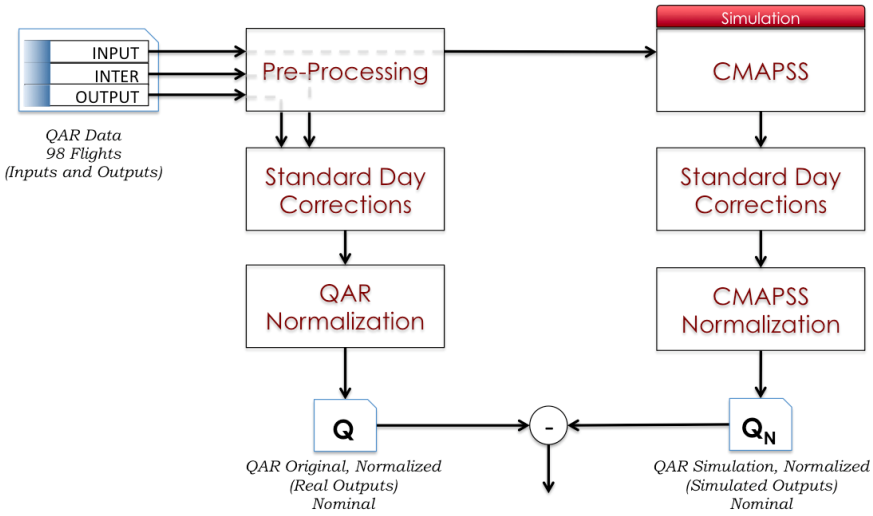


Figure 6.12: **Diagram of Verification Process** - the raw QAR data is split into input and output/intermediate parameters. QAR inputs are processed through CMAPSS to generate simulated outputs/intermediate parameters. Each set then goes through a uniquely tailored standard day correction, normalization, and adjustment process in order to perform a final comparison (see Sections 6.3.2 and 6.3.3).

6.3.1 QAR Pre-Processing Modifications

Every device and entity has a favorite data storage format, and QAR data is no different. Some modifications and pre-processing operations were necessary to transform input parameters into those that could correspond to what is discussed in Section 6.2.3.

First, there is a convention mismatch between T24/P24 and T25/P25, but as was discussed in Section 6.2.1 they are close enough to be interchangeable. A similar equivalency exists between T48 and EGT (Exhaust Gas Temperature).

Secondly, Nf and Nc will need to be suitably adjusted to match the CMAPSS data for comparison purposes. Nf corresponds to fan speed, but it is identical to N1 (since the LPT is connected to the fan and LPC), while Nc corresonds to the core speed and is identical to N2 (since the HPT is connected to the HPT, and they spin at the same higher speed). The main issue with the QAR data seems to be that the convention for maximum speeds is different from that in CMAPSS. In simulation, maximum speed is designated

by 100%, while many manufacturers rate maximum speed at around 120% since this high burst is used only briefly during takeoff. Naturally, appropriate readjustments for normalization need to be made.

Lastly, VSV are actually be inverted between QAR and CMAPSS, because some systems record the angles from a different baseline (either 0 or 90/180 degrees for open/closed), so they need to be appropriately readjusted (this is discussed in Appendix B).

6.3.2 Standard Day Corrections

Real-world and simulated parameters can vary over a wide range of values, and may never align for two identical flights because of differences in ambient enivornmental effects. In fact, variations in ambient conditions make comparisons of performance for a single engine difficult, because the inlet temperature and pressure influences the thermodynamics of the entire system. Even at a constant operating point, significant differences in measured parameters may be observed.

Understandably, this makes classification of engine state more difficult and may mean that verification is impossible if close matches are corrupted by differing atmospheric effects. Standard Day Corrections are a series of methods derived for the purpose of removing the influence of these ambient factors on other primary variables. Use of these corrections dates back to the 1940's, though their formalization and derivation are more recent. Standard Day Corrections (from here on, referred to simply as "corrections") all follow a pattern equation of the form: [Vol99]:

$$p^* = \frac{p}{\theta^a \delta^b} \tag{6.8}$$

The theory, derived from a long history of thermodynamic and conservation laws, is that the input conditions measured at the inlet (primarily T2 and P2) can be used to adjust the measurements at all other parts of the engine with a reference to standard day values. Using this international standard of $T2 = 518.67$ Kelvin and $P2 = 14.696$ pounds per square inch (psi), the two θ and δ parameters are dimensionless variables whose value is deteremed with a direct reference to standard day T2 and P2, as follows:

$$\theta = \frac{T2}{518.67} \tag{6.9}$$

$$\delta = \frac{P2}{14.696} \tag{6.10}$$

In Equation 6.8, a and b are parameter dependent exponents with approximate values related to the physics of a general turbofan jet engine, but precise values should be retreived from the manufacturer's literature (this is a standardized enough practice that most engines have these parameters empirically calculated). CMAPSS includes specific exponents in the documentation, allowing for accurate correction of simulated data with respect to ambient temperature and pressure conditions. The GP7270, however, has these exponent values as part of a proprietary set of engine controller parameters, so the approximate values were used instead (there is usually not much difference between the approximated and exact exponents).

For the 7 output parameters that we will need to correct, the corresponding equations (using exponent values for CMAPSS), are as follows:

$$T24^{*} = \frac{T24}{\theta^{1.0}} = \frac{T24}{(\frac{T2}{518.67})^{1.0}} \tag{6.11}$$

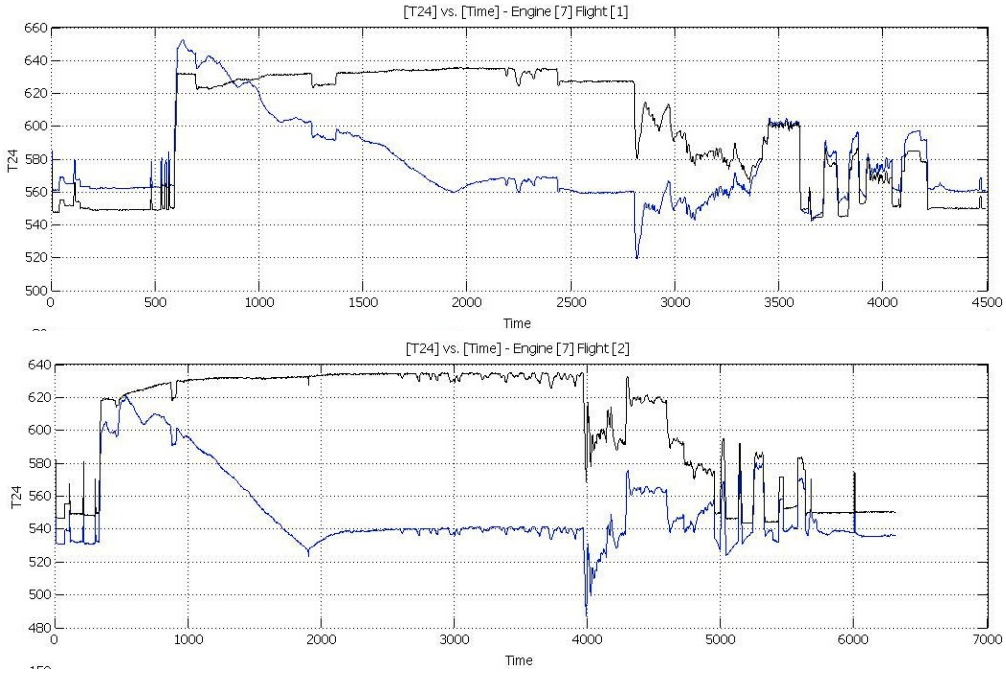


Figure 6.13: **T24 Plots, Before and After Standard Day Correction** - Sample readings from sensor T24 for Engine 7, Flight 1 (top) and Flight 2 (bottom). The value of the parameter before correction is shown in blue in both cases, and is characterized by a decline after the plane reaches cruising altitude. The value of the parameter after correction is shown in black, and exhibits a pattern much similar to that of the input indicators.

$$T48^* = \frac{T48}{\theta^{0.98}} = \frac{T48}{(\frac{T2}{518.67})^{0.98}} \tag{6.12}$$

$$P24^* = \frac{P24}{\delta^{1.0}} = \frac{P24}{(\frac{P2}{14.696})^{1.0}} \tag{6.13}$$

$$Ps30^* = \frac{Ps30}{\delta^{1.0}} = \frac{Ps30}{(\frac{P2}{14.696})^{1.0}} \tag{6.14}$$

$$Wf^* = \frac{Wf}{\delta^{1.0}\theta^{0.66}} = \frac{Wf}{(\frac{P2}{14.696})^{1.0}(\frac{T2}{518.67})^{0.66}} \tag{6.15}$$

$$Nc^* = \frac{Nc}{\theta^{0.5}} = \frac{Nc}{(\frac{T2}{518.67})^{0.5}} \tag{6.16}$$

$$Nc^* = \frac{Nf}{\theta^{0.5}} = \frac{Nf}{(\frac{T2}{518.67})^{0.5}} \tag{6.17}$$

These corrections in and of themselves seem to be improve the correspondence between even two different flights (as they are meant to) to the point that at least the takeoff and cruise portions of the flight generally overlap quite a bit. Figure 6.13 shows samples of T24 values before and after correction, highlighting the nature of these changes.

Figures 6.14 and 6.15 demonstrate the same plots, but compare different flights to each other. In this sense, the original data is much different from each other - despite the presence of the same strange declining behavior, the absolute values of the temperatures are drastically different. After corrections, the temeprature values themselves track each other relatively well for corresponding phases of flight, which will definitely make the data much more consistent.

6.3.3 Normalizations and Adjustments

At this point, we have corrected QAR output parameters and simulated CMAPSS output parameters independently of each other - but these two datasets still do not correspond well with each other. A further step of normalization each dataset to common unitless boundaries is necessary. Normalization for QAR data will need to be done with respect to the statistics of the data, rather than the baselines we worked with in generating our simulation data, because no such basalines were available for the PW7270 engine.

The proposed approach would be to "normalize" the data by effectively defining a 0% and 100% level for each input and output gas path parameter and then convert all the

data to percentage levels. There is some precedent for this, as analysis with normalized percentage levels is routinely carried out with respect to spool speeds (i.e. the PLA setting, which does not directly correspond to precise thrust values, but is simply a % thrust value on a 0-100 scale). This is definitely not a precise solution, but it would get a bit closer to allowing a 90K simulation to be used for a 70K real engine.

In practice, a further justification for this approach is that any CMAPSS v2 simulation will *not* represent a real existing engine in the first place. What we are hoping for is to detect and identify strongly faulted behavior which would be visible despite these fundamental model-engine mismatches.

A summary of the finalized normalization extrema for CMAPSS can be found in Table B.1 in Appendix B, along with some practical discussion of their computation. Normalization for QAR paramters must be performed with respect to the data itself, and yeilds slightly different parameter boundaries than the idealized simulation environment of CMAPSS. Most noticable is a bias of QAR data with respect to CMAPSS, and most puzzling is the frequent presence of two separate clusters of QAR data (Figures 6.16 and 6.17 show some samples).

As can be seen from both figures, there is a definite lack of overlap for a majority of samples, which not only frustrates verification, but will make classification impossible, provided that faulted classes scatter less than this discrepency (unfortunately, they do). For several of the plots, this seems to be a problem of outliers (like in Figure 6.17), where a few samples in the QAR data are skewing the majority upwards. This may be partially corrected by removing the relevant outliers from decisionmaking in the QAR normalization. The main problem, however, may be a matter of scaling, as seen in Figure 6.16.

We confirmed that the bimodal clusters we are seeing in the Wf vs. parameter plots do not occur during any particular flights; nor do they occur only during isolated phases of flight. They primarily seem to occur sporradically during cruise, but appear on return flights from identical routes on the same plane. The most telling example of this problem is shown in the figure below:

The best remedy for this is a series of intelligent, yet ad-hoc adjustements that are made to the normalization boundaries: primarily to remove outlier issues, but also in an attempt to reorient the distributions of the parameters so that they are more closerly aligned. The intuition behind this is that each single paramter should have a roughly similar distribution, with a focus on matching up the peaks in each plot. Outliers may occur to distort the width of the entire distribution, but as long as the majority of the points are well-matched, the plot should be more aligned.

The minimal set of adjustments fitting to this process is as follows:

1. Adjust Nf Min Cutoff to 15%
2. Adjust Nc Min Cutoff to 60%
3. Adjust T24 Min Cutoff to 15%
4. Adjust T48 Min Cutoff to 40%

Figure 6.19 shows a sample pair of distributions, before this adjustement process. Figure 6.20 shows the same comparison, after the final adjustment has been made.

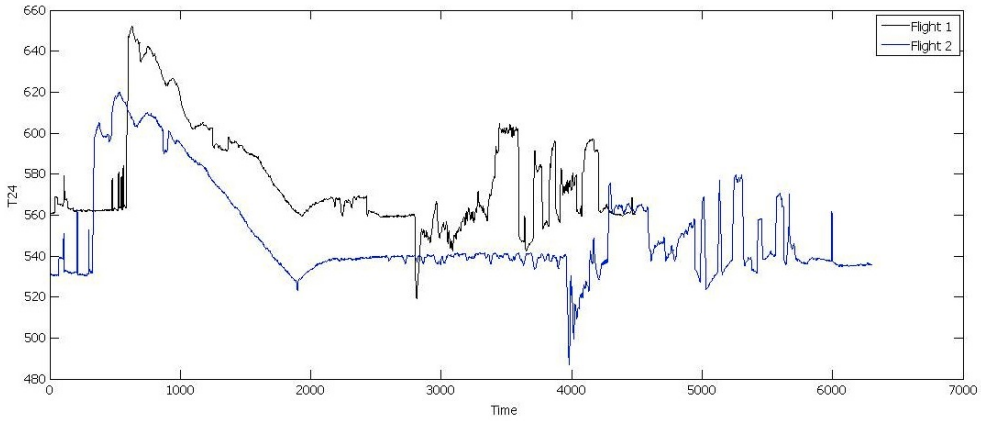


Figure 6.14: **T24 Plots, Uncorrected** - comparison of FOQA Flight 1 and Flight 2 sensor T24 readings without corrections. Both flights exhibit a gradual dropoff after reaching cruising altitude (though to different degrees), despite the fact that at cruising altitude the plane is operating at steady state maximum thrust. This is a result of ambient temperature influencing the cooling of the entire system over the course of the flight, until the heat generated by the engine and the temperature at cruising altitude reach equilibrium (at about 2000 minutes into the flight).

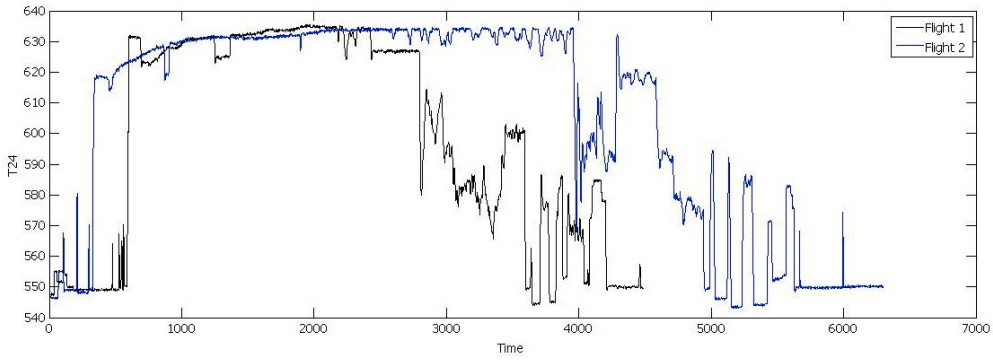


Figure 6.15: **T24 Plots, Standard Day Corrected** - comparison of FOQA Flight 1 and Flight 2 sensor T24 readings after corrections. Despite the differences in the original flight characteristics (Figure 6.14), the resulting readings look relatively similar for various stages of flight, improving upon what is seen in raw data. Note that the graduate decline in temperature no longer exists in either flight, and that despite the two being fundamentally different flights, their takeoff, cruise, and descent characteristics are generally similar.

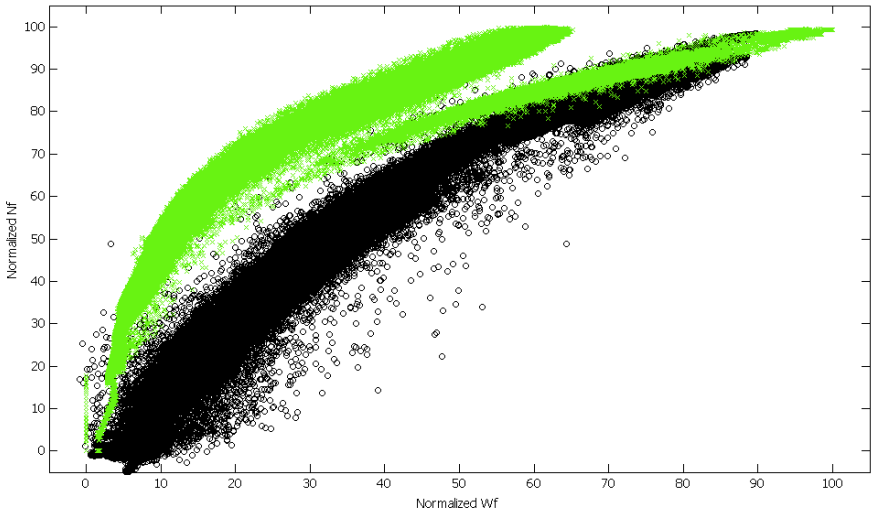


Figure 6.16: **Sample Comparison of Normalized Wf/Nf** - Distribution of normalized nominal CMAPSS flight points (black circles) and normalized QAR flight points (green x marks), comparing normalized fuel flow (Wf) to fan speed (Nf/N1). There is a distinct difference in the curve shapes owing to the scaling of fuel flow as a result of the differences between simulated thrust rating (90k for CMAPSS) and real engine thrust (70k for the GP 7270), but also because of a normalization mismatch.

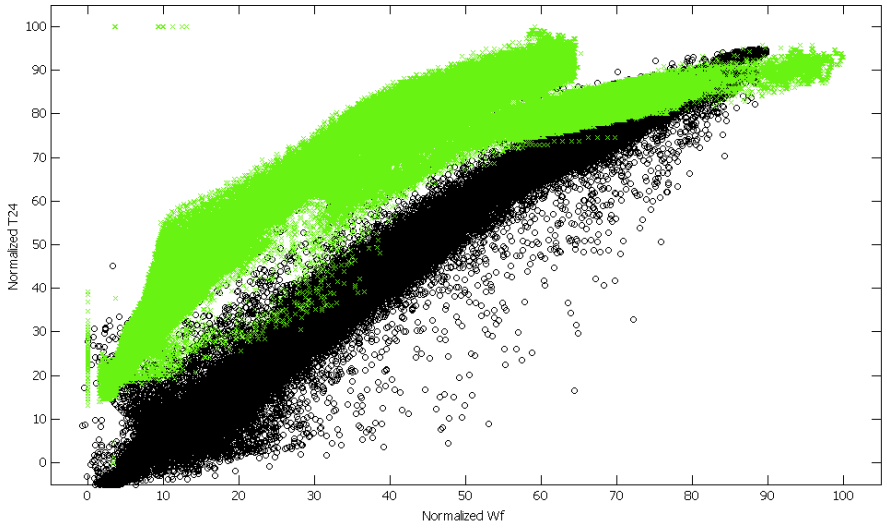


Figure 6.17: **Sample Comparison of Normalized T24/Wf** - Distribution of normalized nominal CMAPSS flight points (black circles) and normalized raw QAR flight points (green x marks), for the relationship between fuel flow (Wf) and the temperature at the HPC inlet (T24). Again, there is a distinct difference in the clusters, though less pronounced than in the Nf plot in Figure 6.16. Note, however, that the QAR points seem to really start clustering around a T24 value of 15 rather than 0, indicating that there may be a few outliers in the normalization process.

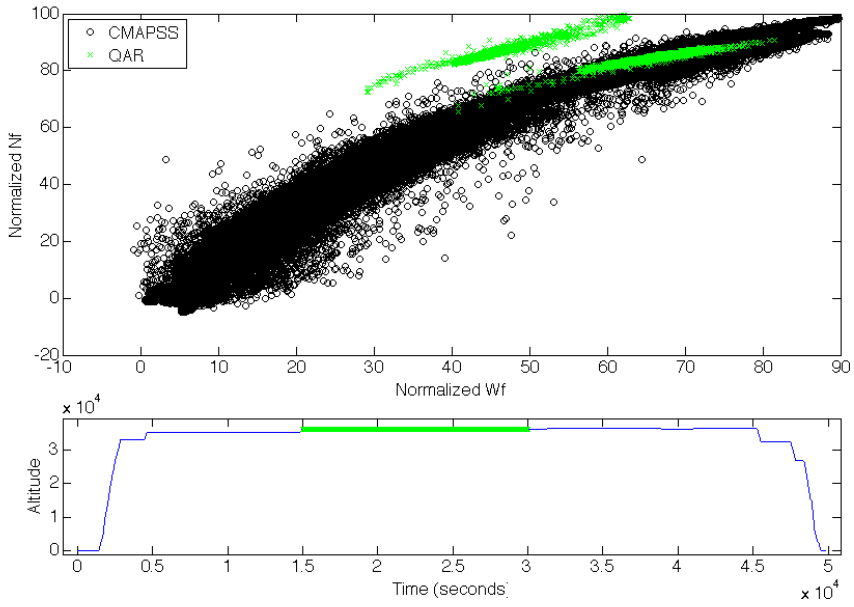


Figure 6.18: **Bi-Modal Clusters in "Stationary" Cruise** - Example of Flight 4, Normalized QAR Nf (green) overlayed on aggregate normalized Nf CMAPSS simulation values (black), during a portion of steady-state cruise, all with respect to normalized fuel flow (Wf). The top plot shows values from a portion of steady-state cruise that exhibit the bi-modal structure seen in previous figures, while the bottom plot shows the corresponding portion of the flight (altitude with respect to time), from which this data was taken.

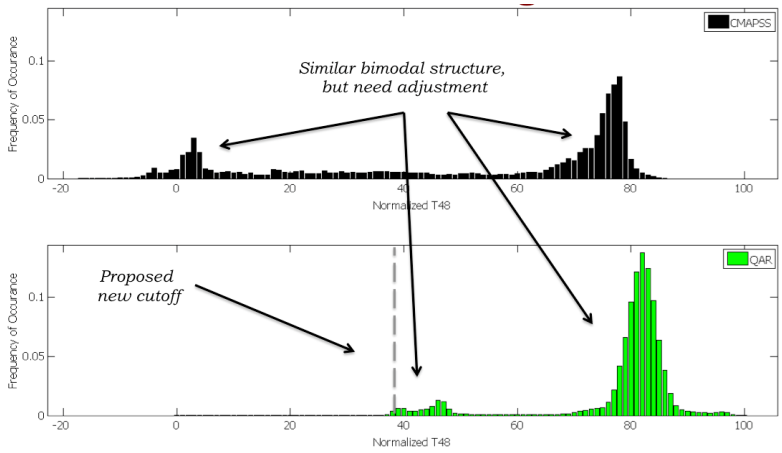


Figure 6.19: **Sample T48 Distributions, Pre-Adjustment** - histograms of T48 samples from corrected/normalized CMAPSS simulated data (black, top) and corrected/normalized real-world QAR data (green, bottom). Even after applying standard day corrections and normalizations, there can be this much misalignment between the CMAPSS and QAR data, motivating a proposed histogram matching procedure.

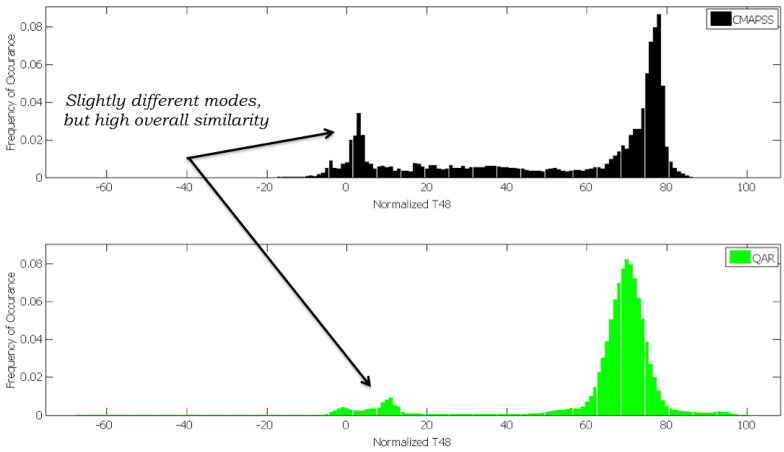


Figure 6.20: **Sample T48 Distributions, Post-Adjustment** - histograms corresponding to Figure 6.19, after matching adjustment (alignment of distribution peaks). After the adjustments, these two distributions are much more similar, having accounted for the discrepencies in thrust rating between CMAPSS and the GP 7270.

6.3.4 Pearson Correlation Coefficient Results

In order to compute the level of similarity between the original (corrected and normalized) QAR data and the simulated (corrected and normalized) CMAPSS data, we need a tool that can check the point-by-point and relativel correspondence between these pairs of time series. The Pearson Correlation Coefficient computes this measure of similarity as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{6.18}$$

As was mentioned earlier, there is a timing mismatch between the QAR and CMAPSS simulation data. QAR is sampled at 1Hz by definition, so its time indexes are integers. CMAPSS yields 66Hz data that we then decimate by 64 to *approximately* 1Hz, which yields fractional time indexes that are out of sync by 32/33 of a second (once every 33 seconds, there is an extra sample (see Figure 6.21 below).

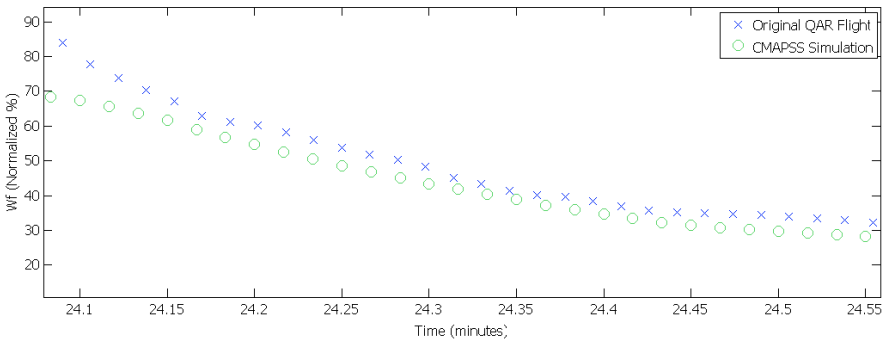


Figure 6.21: **Down-Sampling Mismatch in Verification** - There is a periodic misalignment between the original QAR (blue x) and simulated CMAPSS (green o) data, visible in the fact that at times the two markers line up, and at times they do not. This is an artifact of the factor-of-64 downsampling of 66Hz data to *approximately* 1 Hz, which was discussed at the end of Sections 6.2.1 and 6.2.4.

To solve this problem, we perform interpolation of the CMAPSS data into corresponding QAR time indexes, since that is the baseline ground truth in this case. This

may potentially introduce some minor errors because the interpolation is linear, but will likely not be a major factor in the end. With this re-adjusted data, we compute Preason Correlation Coefficients for all variables of interest; sample and averaged results are shown in Figure 6.22.

Plane HL7611, Engine 1 – Flight KE607

	Wf	VSV	Nf	Nc	T24	P24	T48
Original	0.775	0.922	0.956	0.499	0.562	0.901	0.890
Corrected	0.969	0.922	0.975	0.824	0.949	0.976	0.965
Normalized	0.969	0.924	0.977	0.880	0.961	0.981	0.965

Plane HL7611, Engine 1 – Flight KE81

	Wf	VSV	Nf	Nc	T24	P24	T48
Original	0.779	0.942	0.962	0.607	0.656	0.917	0.921
Corrected	0.974	0.943	0.978	0.883	0.946	0.967	0.871
Normalized	0.974	0.943	0.979	0.908	0.954	0.969	0.971

Averages, All Flights

	Wf	VSV	Nf	Nc	T24	P24	T48
Normalized	0.958	0.912	0.977	0.885	0.954	0.862	0.957

Figure 6.22: **Pearson Correlation Coefficient Results** - The top two charts show coefficients computed for sample flights using original, corrected, and normalized data. The bottom chart shows averaged results over all 98 QAR flight records.

As can be seen from the tables, virtually all variables of interest for our investigation exhibit a very strong correlation. The only real exception is Nc, whose coefficient can be in the 85% range, but all the other paramters show at least 95% and generally much higher correlations. Based on these results, it is safe to say that we are able to establish a provable correspondence between the two datasets. The lower Nc value may be due to the fact that this part of the engine is highly affected by the VSV, VBV, and Wf values, as well as any other changes that the controller on the engine makes - since the controllers in real data are bound to be different than those in CMAPSS, it is not unreasonable to see a lower correlation value.

6.4 Fault Detection and Diagnosis Algorithm

Having established a correspondence between real-world and simulated data, the next step is to develop a detection and diagnosis algorithm that works on the CMAPSS simulated data (which includes nominal and faulty data). We approach this problem from the standard machine learning approach - by selecting and suitable feature set and classification tool that maximizes correct classification rate with respect to a reasonably small time and space complexity.

6.4.1 Feature Set Selection

Initally, we were eager to try MFCC and CELP features, seeing their great performance in the context of vibration sensors. But both of these may not be very well suited to this scenario due to a problem of scale. Namely, MFCCs and CELP features are designe to analyze signals sampled at 8kHz, with windows of 256 samples, indicating a resolution of about 0.05 seconds. Our data is already sampled at 1 second, nearly two orders of magnitude difference. Preliminary experimental results confirm this supposition - MFCC and CELP feautures did not achieve a good classification rate.

Because of the low fidelity fo the signal, DCT and (Haar) wavelet features were tried next and ended up performing very well. There are two main parameters that can be set for feature extraction: window length and overlap percentage. In an attempt to tailor these to the dynamics present in the data, variable window lengths and overlap sizes were tested up to 32 samples (half a minute). The results of these tests are in Figure 6.23

On average, we have produced good results for classification of CMAPSS simulation flights with DCT (86.6%) and Wavelet (87.2%) features. All experiments are 10-fold cross-validated with a 50/50 split between the training and testing samples, where each

	Time (seconds)			Performance	
Features	**Extract***	**Train****	**Test***	**Segment****	**Flight****
DCT (8/0)	0.31	30.3	0.14	92.6%	93.5%
DCT (16/0)	0.17	15.9	0.13	96.9%	99.8%
DCT (16/8)	0.32	33.4	0.15	98.0%	100%
DCT (16/15)	2.50	259.8	0.35	93.3%	93.5%
DCT (32/16)	0.18	19.4	0.16	84.8%	85.5%
DCT (32/24)	0.34	31.2	0.13	89.9%	89.1%
DWT (16/8)	4.92	27.6	0.17	96.9%	98.4%
DWT (32/16)	2.31	15.7	0.16	93.2%	95.5%

* *Per flight, averaged over F_N datasets*
** *Per training run, averaged over 10-Fold Cross Validation, with 50/50 split*

Figure 6.23: **Sample Results for Different Feature Sets** - The table shows the detection accuracy and time complexity of several different combinations of DCT and DWT featres. The best performance was achieved with DWT features of length 16 and a 8 sample overlap, while discarding all but the first 8 coefficients.

subset is chosen randomly from the entire 69 flight dataset for each iteration. This guarantees results are not tied to a particular selection of samples, though it doesn't yet show how few flights are actually needed to get a reasonable result. The performance of the two features sets is relatively similar at the per-window level (and identically perfect at the per-flight level), but DCT features are extracted a bit faster.

The next item to address was how to reconcile the correspondance between the 4 input features that were being used in their raw state (Altitude, Mach Number, TRA, and delta Ambient Temperature) with the parameters that were being windowed. The most obvious choice is to have the sample at the center of the window represent the windowed region, which potentially means a corresponding windowing and averaging of the raw input data.

6.4.2 Classifier Construction

Following the experience from the classifier selection in Chapter 3, we elected to once again use the Support Vector Machine. The extremely fast testing times are important in developing this procedure as a potential real-time application that is able to provide immediate feedback to the pilots. As such, intial tests included a simple linear SVM, with all features extracted from the main output parameters (Wf, Nf, Nc, T24, P24, Ps30, and T48). Results of these initial tests are available in Section 6.5.3.

Because there are now 6 classes of data (1 nominal and 5 different faults), the classification is not as simple as the tiered or non-tiered discussed earlier. To avoid the added time complexity of training on all 6 classes at once, we elected to develop a set of binary classifiers that would differentiate between any two classes (this yields 15 classifiers total). After each classifier is applied to an unknown data sample, a majority voting mechanism is used to determine what scenario that particular data point belongs to. While this does provide room for ambiguous ties in some cases, in practice we found that such events occurred much less than 1% of the time.

Another important outcome of the discussion in Chapter 3 was the separation of detection and diagnosis, particularly in the effect it had computational complexity, since the detriment to performance was minimal. As was seen in the feature reduction analysis in Section 3.4.5, SVMs scale poorly with large feature sets, but can still perform well when smaller partial decisions are combined. We decided rather quickly to try to adapt this philosophy to the treatment of features as separate subgroups in order to utilize the parallelizing power of fusion. Rather than throwing all of the features derived from each sensor into a single classifier, it is more computationally efficient to treat each sensor/parameter as a separate entity based on which a decision can be made.

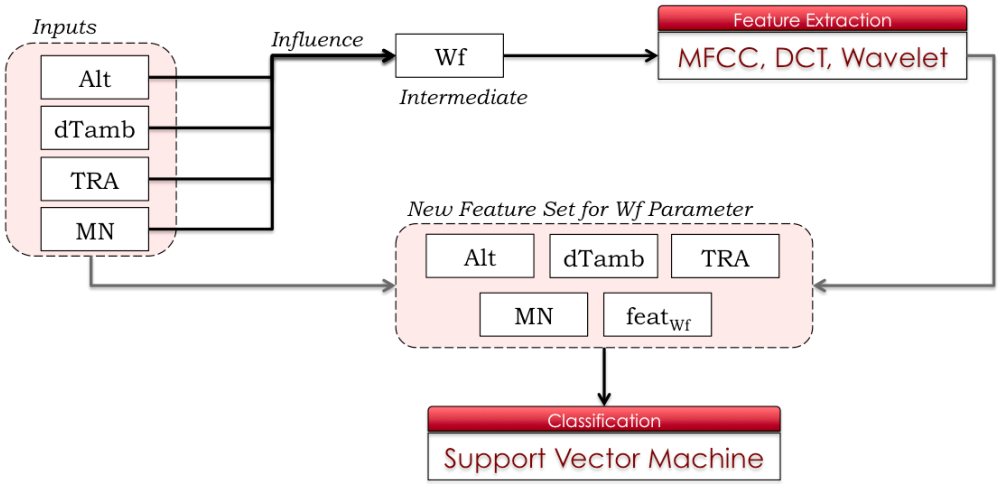


Figure 6.24: **Classifier Fusion, Single Output Variable** - this figure shows the influence of input variables on an intermediate (or output) variable, from which features are subsequently extracted. The combination of features and input variables is used as a new feature set for an individual parameter's Support Vector Machine. Classification results are combined as shown in Figure 6.25. Note that each "SVM" in this schematic is actually 15 binary SVMs with a majority voting setup over the 6 classes.

This requires a redefinition of the features along the lines of isolating particular output parameters along with related input/intermediate variables - and subsequent training of classifiers for these intermediate steps. An outline of this approach for a single sensor is shown in Figure 6.24, while the entire system is shown in Figure 6.25. Results of the training and testing with these modified fused feature sets can be seen in Section 6.5.2.

In the above setup, each "Support Vector Machine" classifier (including both the intermediate, paramter-specific SVMs and the comprehensive fusing SVM), actually consists of 15 binary SVMs that decide between all of the possible combinations (6 choose 2) of the 6 classes (1 nominal and 5 faults). As mentioned above, majority voting is used to determine the actual classifcation results, which are discussed in Section 6.5.

In many cases, we also found that using a logistic linear classifier was suitable in achieving a reasonable classification result, especially in the earlier design phases.

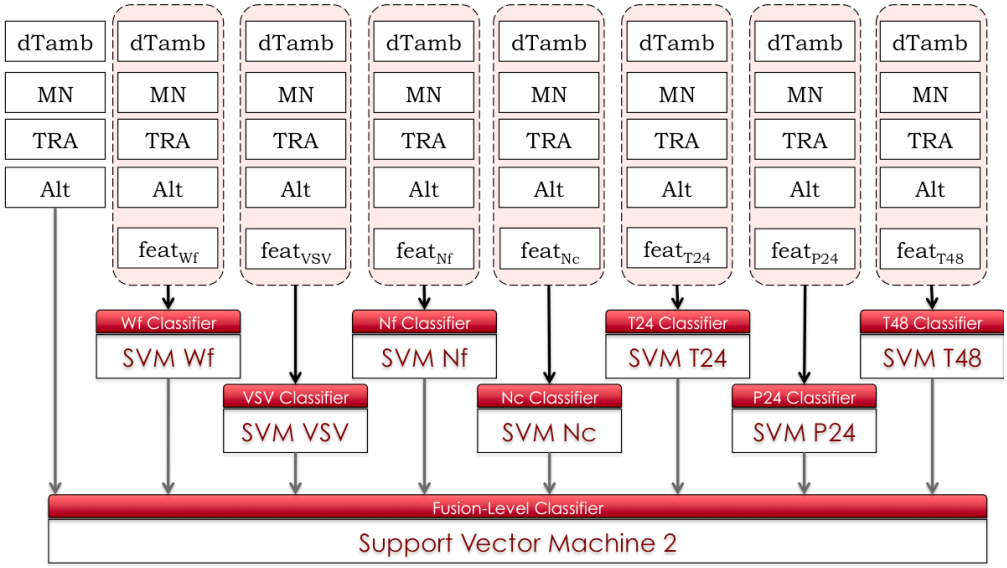


Figure 6.25: **Classifier Fusion, Full Setup** - diagram of the fusion classification approach. Individual SVMs are constructed for each intermediate or output parameter of interest and classification results are combined at the second stage with a comprehensive SVM that also includes input parameters.

6.5 Simulation Results

Figure 6.26 demonstrates the a basic "proof of concept" result. It is a plot for a single flight of the T48 values from the nominal and fault scenarios considered in our experiments, when the fault magnitudes were all the same (3%) and the fault was inserted at the start of the flight (Dataset F_1). As we would hope, the data is easily (and visually) separable within the steady state portions of the flight under these ideal conditions, and when examining the parameter that is most sensitive to changes in engine health. Yet even here, distinguishing between these faults during transient regions is difficult.

Subsequent results in the following sections are performed on datasets that are much more complex than this one from several perspectives: variable fault times, variable fault magnitudes, addition of noise, etc. Each set of results will show a confusion matrix for the windows (all feature extraction segments in all flights) as well as an aggregate result

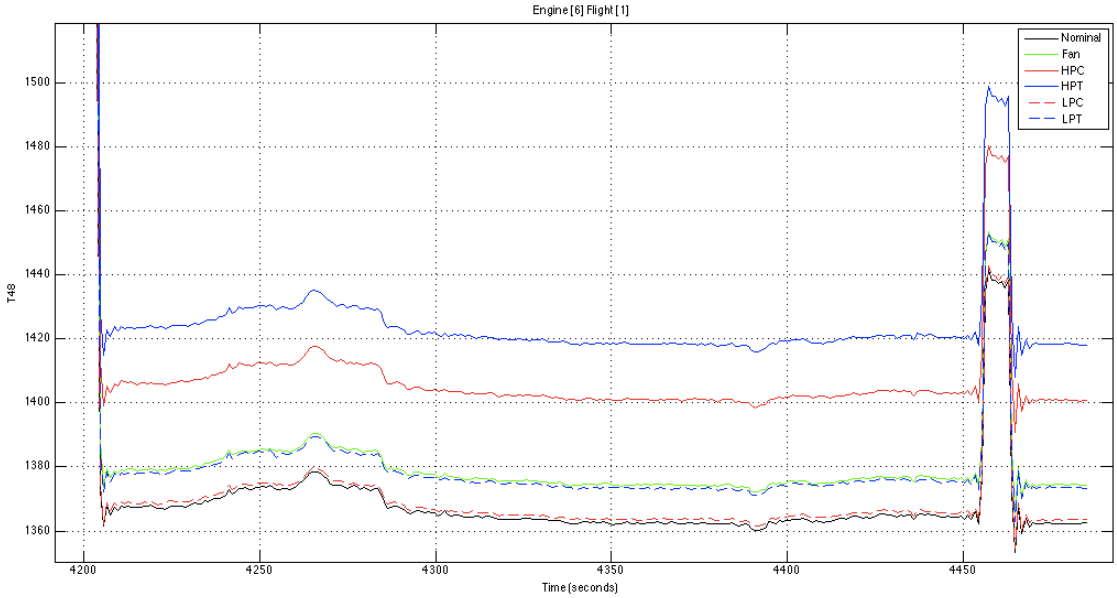


Figure 6.26: **Sample Simulated T48 Values, Nominal and Faults** - Comparison of T48 values at end of FOQA Flight 6 with all faults and nominal data. Note that the sensor value is a relatively consistent "scaling" of the original nominal plot (black solid line, near the bottom of the plot), and that each type of fault is identifiable by its magnitude deviation, but that this is possible here because of identical magnitudes for each fault. The plots that are closer to nominal result from faults inserted into turbomachinery components further away from the exhaust, where T48 is measured.

for an entire flight computed using a majority vote over the window decisions for that flight. While the former is of more academic interest, industry expectations are primarily focused on the latter, which is why we present both.

In each case, we ran experiments with a 50/50 split between training and testing data (no parameter tuning was necessary, so no validation set was needed), and cross-validated by selecting individual points for training randomly 10 times. A more standard cross-validation using a 20/80 split between training/testing (respectively) was also performed for initial tests, and showed a marginally lower classification result.

In all experiments, unless specified otherwise, there was a uniform prior distribution - there were equal amounts of nominal and fault data for each class.

6.5.1 Simulation Results

As Figure 6.27 shows, the overall classification results are consistent with the high percentages seen in the comparison experiments, yielding about 86.6% overall classification accuracy. Naturally, when these results are aggregated over the course of the flight (taking windows as samples in a majority vote), the per-flight results in Figure 6.28 are 100% for all the flights tested.

	Nom	**Fan**	**HPC**	**HPT**	**LPC**	**LPT**	
Nom	11277	645	639	648	781	823	76.1%
Fan	68	11324	14	19	751	381	90.2%
HPC	199	152	10768	282	158	154	92.9%
HPT	7	7	1092	11574	7	6	91.2%
LPC	990	561	362	358	11025	394	80.5%
LPT	374	225	40	34	194	11159	92.8%
	87.3%	87.7%	83.4%	89.6%	85.4%	86.4%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from Batch $\boldsymbol{F_1}$ (CMAPSS FOQA simulation only)*

Figure 6.27: **DCT 16-8 Results, Per Window, No Fusion** - confusion matrix showing the average correct classification rates for the experiment without fusion. The overall correct classification rate for the entire dataset is 86.6%.

Perhaps surprisingly, Figure 6.27 shows that the detection portion of the system does not perform as well as the diagnosis component (notice higher correct classification along the non-nominal rows/columns in the confusion matrix). Furthermore, HPT faults seem particularly easy to identify, noting the low numbers in the corresponding row within the confusion matrix.

	Nom	Fan	HPC	HPT	LPC	LPT	
Nom	34	0	0	0	0	0	100%
Fan	0	34	0	0	0	0	100%
HPC	0	0	34	0	0	0	100%
HPT	0	0	0	34	0	0	100%
LPC	0	0	0	0	34	0	100%
LPT	0	0	0	0	0	34	100%
	100%	100%	100%	100%	100%	100%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from Batch F_1 (CMAPSS FOQA simulation only)*

Figure 6.28: **DCT 16-8 Results, Per Flight, No Fusion** - confusion matrix showing the average correct classification rates for the experiment without fusion. The overall correct classification rate for the entire dataset is 100%, using majority voting over window segments in each flight.

6.5.2 Simulation Results with Fusion

When a fusion paradigm (Section 6.4.2) combines individual sensor results to learn from intermediate decisions, the window classification rate increases dramatically to 97%, as seen in Figure 6.29. When more opportunities for correct classification are available, as in the feature stage of fused classification, the comprehensive classifier can learn when the individual feature classifiers make mistakes and act to correct them.

Here, we also see that the detection performance is nearly as good as for diagnosis; in contrast to what was seen in the case without fusion. It is likely that these situations had a higher percentage of borderline decisions that were made in favor of incorrect choices in the system with no fusion.Computationally, this approach requires more memory, but actually achieves a speedup when compared to the non-fusion version where all the features from all 7 output/intermediate parameters are used to train a single set of 15 SVMs.

	Nom	Fan	HPC	HPT	LPC	LPT	
Nom	37410	92	88	87	468	1652	94.0%
Fan	6	23125	1	0	237	322	97.6%
HPC	15	3	23297	14	5	3	99.8%
HPT	2	0	38	23362	0	0	99.8%
LPC	710	79	0	0	22564	143	96,0%
LPT	295	168	42	3	192	21346	96.8%
	97.2%	98.5%	99.3%	99.6%	96.1%	90.9%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from Batch $\boldsymbol{F_1}$ (CMAPSS FOQA simulation only)*

Figure 6.29: **DCT 16-8 Results, Per Window, With Fusion** - confusion matrix showing the average correct classification rates for the experiment with fusion. The overall correct classification rate for the entire dataset is 97.0%.

	Nom	Fan	HPC	HPT	LPC	LPT	
Nom	34	0	0	0	0	0	100%
Fan	0	34	0	0	0	0	100%
HPC	0	0	34	0	0	0	100%
HPT	0	0	0	34	0	0	100%
LPC	0	0	0	0	34	0	100%
LPT	0	0	0	0	0	34	100%
	100%	100%	100%	100%	100%	100%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from Batch $\boldsymbol{F_1}$ (CMAPSS FOQA simulation only)*

Figure 6.30: **DCT 16-8 Results, per Flight, with Fusion** - confusion matrix showing the average correct classification rates for the experiment with fusion. The overall correct classification rate for the entire dataset is 100%, using majority voting over window segments in each flight.

6.5.3 QAR Verification Results

In the final set of experiments, we conduct classifier training on the simulated CMAPSS data derived from FOQA inputs (which includes all the different fault types), and subsequently test this classifier on the the corrected and normalized QAR data (all of which is presumably nominal). Figure 6.31 shows the results of this experiment.

	Nom	Fan	HPC	HPT	LPC	LPT	
Segments	25173	362054	859	15030	40048	16	5.6%
Flights	3	92	0	0	3	0	3.1%

Training on $\mathbf{F_1}$ data (FOQA CMAPSS Simulation), Testing on raw $\mathbf{Q}$ (QAR) data

	Nom	Fan	HPC	HPT	LPC	LPT	
Segments	32389	230979	831	8878	9919	9559	11.1%
Flights	5	61	0	0	0	0	7.6%

Training on $\mathbf{Q_1}$ data (QAR CMAPSS Simulation), Testing on raw $\mathbf{Q}$ (QAR) data

Figure 6.31: **DCT 16-8 Results, for QAR Data** - confusion matrices for a series of experiments involving testing of the verified QAR dataset. The top table correspond to training on FOQA-derived simulation data, yielding a 3-5% correct classification rate for segments and flights. The bottom table correspond to training on a part of QAR-derived simulation data and subsequently desting on a different part of the corrected and normalized QAR outputs, yielding a correct classification rate of 7-11% for flights and segments.

As seen from the top two rows, the end result is seemingly abysmal failure - almost none of the flights and segments were correctly classified. Even random guessing should have yielded better results.

Yet maybe the fault is in the choice of underlying data; training on CMAPSS simulated data that is based on QAR inputs (rather than FOQA inputs) may yield better classifiers for subsequently corrected and normalized QAR output data. This was the experiment performed and the bottom part of Figure 6.31, where the performance improves slightly, but is still disappointing.

Both of these, however, have one curiousity in common - the vast majority of samples were classified into the Fan Fault category. Overwhelmingly, the classifiers believed there was a problem with the Fan module. Upon further investigation into this issue by our contacts at Pratt-Whitney and NASA, it was discovered that the CMAPSS simulation software has a flaw in a component of the model that deals with simulating the fan's performance.

Further experiments (see Appendix D) yield similarly confusing results, despite our best efforts to rectify the contradiction. At the time of publication, that issue has still not been resolved, so it is uncertain whether the underwhelming final results were indeed caused by this coincidentally malfunctioning component, or perhaps a result of imperfect normalization procedures. It is our sincere hope that it is the former, though it is currently impossible to say for certain.

6.6 Summary

In this chapter, we introduced an improved approach to Gas Path Analysis in the context of jet engine fault detection and diagnosis. The great strenght of this work is the adaptation of signal processing techniques to very low frequency data, with successful simulation results. One of the key challenges of the processes developed in Chapters 3, 4, and 5 was the inability to deal with industrial fidelity data (primarily sampled at 1 Hz). The approach presented in this chapter effectively works on fully simulated data (Section 6.5.2) and *likely* also works on real world data.

The high volume of simulated data itself is an achievement, because to date CMAPSS has not been so extensively used to generate high quality, full-flight records. To the best of our knowledge, this is the first work that has taken input flight records from several sources and produced a wide-ranging corpus of benchmark gas path data.

We successfully established and verified a correspondence between simulated and real-world data; the statistical test that was performed in Section 6.3.4 is reasonable proof that these two datasets do highly correspond to each other, even if experimental results are tentatively inconclusive. If, when the fan problem in CMAPSS is corrected, this approach proves effective in accurately classifying real-world data as nominal flights when trained on purely simulated nominal and fault information, the main contribution of this work will be this combined process that allows for the development of a system of fault detection and identification without the need for expensive, real-world faulty training data.

Chapter 7

Conclusion

The effectiveness of any mechanical system is inexorably tied to how well it is designed and maintained, with the latter of the two usually being of greater practical interest. In an increasingly competitive global economy, there is a constant need to revisit the tradeoffs between the costs of maintenance and the benefits of safe, reliable operation. One way of decreasing maintenance costs in the world of cheap and miniaturized processing power is through automatic monitoring, which not only makes tracking of engine health faster but also more easily quantifiable.

From the work presented in the above chapters, it is clear that an analysis of vibration sensor data, suitably performed, can be a great help in discoverying and identifying problems with jet engine operation. The discussion in Chapter 3 describes a compelling approach that works extremely well on the most abundant of in-flight data (idle and cruise), albeit the performance degrades somewhat in an unexpected manner when the sampling frequency is reduced. Results in the initial analysis of currently-used vibration sensors (shown in Section **??** demonstrates a complete lack of harmonic information at the 1 Hz resolution, indicating a need of higher-frequency sampling rates for a useful system, but the down-sampling results of synthetically generated data in Section **??** suggest that this does not need to higher than several hunderd Hertz. Treatment of so-called "non-stationary" phases of flight is still a difficult and open problem, but one that probably warrants a more model-based approach to the dynamics of engine operations.

The first of the proposed extensions - the design of a highly responsive real-time system of engine abnormality detection introduced in Chapter 4 - still requires much development. The effectiveness of the results is highly dependent on the trustworthiness of failure modeling assumptions (how the transitions from normal to abnormal behavior are realized in synthetic data), but a stricter analysis of change-point alternatives to the CUSUM algorithm will also serve to better understand how such a system can be best realized. The preliminary results are, however, generally optimistic and

The second outgrowth of the original research is the potential for a more comprehensive view of engine behavior based on the statistical distribution of vibration sensor readings on a by-flight and historical basis. The method discussed in Chapter 5 is still in its nascent stage of development, but demonstrates a potential for cost-effective management of engine maintenance. While much of its usefulness depends on training with a sufficiently large dataset tailored to variables such as flight routes and maintenance history (both of which are relatively difficult to quantify), the potiential benefits justify further investigation into the feasability and practicality of this approach.

The last topic, a signal processing take on traditional Gas Path Analysis approaches to engine fault detection and diagnosis, demonstrated the effectiveness of appropriate feature selection and classification methods, at least for strictly simulated data. Even if the verification and correspondence experiments at the end of Chapter 6 do not prove sufficiently robust to withstand the challenge, we the excellent results for CMAPSS simulated data provide an enticing glimmer of hope that an approach that does not rely on intimate knowledge of mechanics is feasible, and at a great computational discount.

The engine of the future will become even more integrated and responsive to the diagnostic systems that are being developed. The development of a new generation of

sensors will necessarily lag behind the appropriate data storage or transmission capabilities, and likely be slowed by the reasonable observation that current equipment performs sufficiently well. The analysis of collected information will continue to develop in stride with advances in pattern recognition and machine learning, especially in the context of big data.

The main technological drive will likely be towards developing fuller cooperation between the control and monitoring systems, which are currently independently functioning modules. This new paradigm of reactive contextual control will require years of careful collaboration between various engineering disciplines, to develop proper monitoring algorithms, design appropriate interface protocols, and consider the possible responses that a modern control system could have to changing engine conditions. Certification of such components, especially those dealing with automated direct control over an engine's internal workings, itself requires many years of testing by OEMs and appropriate federal or international institutions. The path ahead is long, but it is a rewarding one.

References

[Arm10] Jefferey Armstrong. Transient test case generator, user's manual. In *NASA Research Document, Glenn Research Center / ASRC Aerospace Corporation*, January 2010. 6.2

[BOS09] A. Babbar, E.M. Ortiz, and V.L. Syrmos. Fuzzy clustering based fault diagnosis for aircraft engine health management. In *Control and Automation, 2009. MED 2009. 17th Mediterranean Conference on*, pages 199 –204, June 2009. 1

[BSOA09] A. Babbar, V.L. Syrmos, E.M. Ortiz, and M.M. Arita. Advanced diagnostics and prognostics for engine health monitoring. In *Aerospace conference, 2009 IEEE*, pages 1 –10, March 2009. 1

[CB01] George Casella and Roger L Berger. *Statistical Inference*. Duxbury Press, 2 edition, 2001. 2.3.1

[CMGL12] J.T. Csank, R.D. May, T.H. Gou, and J.S. Litt. The effect of modified control limits on the performance of a generic commercial aircraft engine. 2012. 6.2

[CMLG11] J.T. Csank, R.D. May, J.S. Litt, and T.H. Guo. A sensitivity study of commercial aircraft engine response for emergency situations. 2011. 6.2

[CNK09] Selina Chu, Shrikanth Narayanan, and C.-C. Jay Kuo. Environmental sound recognition with time-frequency audio features. In *IEEE Transactions on Audio, Speech, and Language Processing*, volume 17, August 2009. 2.2

[CSR11] Subhadeep Chakraborty, Soumik Sarkar, and Asok Ray. Symbolic identification for fault detection in aircraft gas turbine engines. In *Proceedings of the IMechE, Part G: Journal of Aerospace Engineering*, 2011. 2.1

[HSL+98] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society*

of London. Series A: Mathematical, Physical and Engineering Sciences, 454(1971):903–995, 1998. 2.2

[Joh11] Stephen B. Johnson. *System Health Management: with Aerospace Applications*. Wiley, 1 edition, July 2011. 1.1

[LFD+12] Yuan Liu, Dean K. Frederick, Jonathan A. DeCastro, Jonathan S. Litt, and William W. Chan. User's guide for the commercial modular aero-propulsion system simulation (c-mapss), version 2. In *NASA Research Document TM 2012-217432*, 2012. 6.2

[Lin12] James (Yuan) Lin. Commercial modular aero-propulsion system simulation 40k. In *3rd NASA GRC Propulsion Control and Diagnostics Workshop*, 2012. 6.2

[Mal93] Stephanie Mallat. Matching pursuit with time-frequency dictionaries. In *IEEE Transactions on Signal Processing*, pages 3397–3415, December 1993. 2.2

[RI05] Rolf and Isermann. Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, 29(1):71–85, 2005. 1

[RK00] M.J. Roemer and G.J. Kacprzynski. Advanced diagnostics and prognostics for gas turbine engine risk assessment. In *Aerospace Conference Proceedings, 2000 IEEE*, volume 6, pages 345–353 vol.6, 2000. 1

[Saf13] Boeing Aviation Safety. Statistical summary of commercial jet airplane accidents 1951-2010 worldwide. http://www.boeing.com/news/techissues/pdf/statsum.pdf, 2013. 1.1

[SMSR12] Soulmayla Sarkar, Kushal Mukherjee, Soumik Sarkar, and Asok Ray. Symbolic dynamic analysis of transient time series for fault detection in gas turbine engines. *Journal of Dynamic Systems Measurement and Control*, 2012. 2.1

[TB99] I Y Tumer and A Bajwa. A survey of aircraft engine health monitoring systems. *Proceedings of AIAA*, (June), 1999. 1, 1.1

[TZD11] Liang Tang, Xiaodong Zhang, and Jonathan DeCastro. Diagnosis of engine sensor, actuator and component faults using a bank of adaptive nonlinear estimators. In *IEEE Aerospace Conference*, 2011. 2.1

[VBL08] Al Volponi, Tom Brotherton, and Rob Luppold. Empirical tuning of an on board gas turbine engine model for real-time performance estimation. *ASME Journal of Engineering for Gas Turbines and Power*, March 2008. 1.1

[Vol99] Al Volponi. Gas turbine parameter corrections. *ASME Journal of Engineering for Gas Turbines and Power*, October 1999. 6.3.2

[Vol13] Al Volponi. Gas turbine engine health management: Past, present, and future trends. Presi Presentation, June 2013. 1, 1.2, 2.1, 2.1, 5.2, 6.1, 6.2.4

[ZGW11] Dave Zhao, Ramona Georgescu, and Peter Willett. Comparison of data reduction techniques based on svm classifier and svr performance. In *SPIE Signal and Data Processing of Small Targets*, 2011. 2.1

[ZTD12] Xiaodong Zhang, Liang Tang, and Jonathan Decastro. Robust fault diagnosis of aircraft engines: A nonlinear adaptive estimation-based approach. In *IEEE Transactions on Control Systems Technology*, 2012. 2.1

Appendix A

Faults in Test Cell Data

In this appendix, we provide a few additional details regarding the acquisition of data from test cell experiments for the SR-30 and PW400 engines, introduced in Section 3.2. Data collection was performed as a courtesy by UTC Pratt-Whitney for both datasets, who had the necessary equipment and resources to generate this fidelity of data. Normally, introducing damage to a complicated piece of equipment in order to characterise such behavior is cost-prohibitive and dangerous in real scenarios, but doing so in a laboratory test-cell environment can be extremely valuable. We had the ability to request and receive such data in order to test the viability of vibration and acoustic sensors in detecting a few typical problems that may occur in jet engine operation.

A.1 SR-30 Engine

The Turbine Technologies SR-30 turbojet engine in an experimental laborator-type engine assembly designed for testing and benchmarking **??**. Four vibration sensors were placed around the exterior of the assembly seen in Figure A.1, spaced roughly 90 degrees around the main cylinder. 11 acoustic sensors were also placed at various positions around the engine.

Figure A.2 shows the faults introduced into the engine during the experiments - simulated fan blade damage and bearing failures. As mentioned in Section 3.2.2, a sampling rate of 102.4 kHz was utilized to record 30 seconds of vibration data during 6 different

speed profiles: idle (43 kRPM), cruise (70 kRPM), acceleration, fast acceleration, deceleration, and fast deceleration. All changes in speed were occurred strictly between 43 and 70 kRPM, with varying speeds for the normal and fast versions. More details of these flight stages can be found in Figure 3.2.

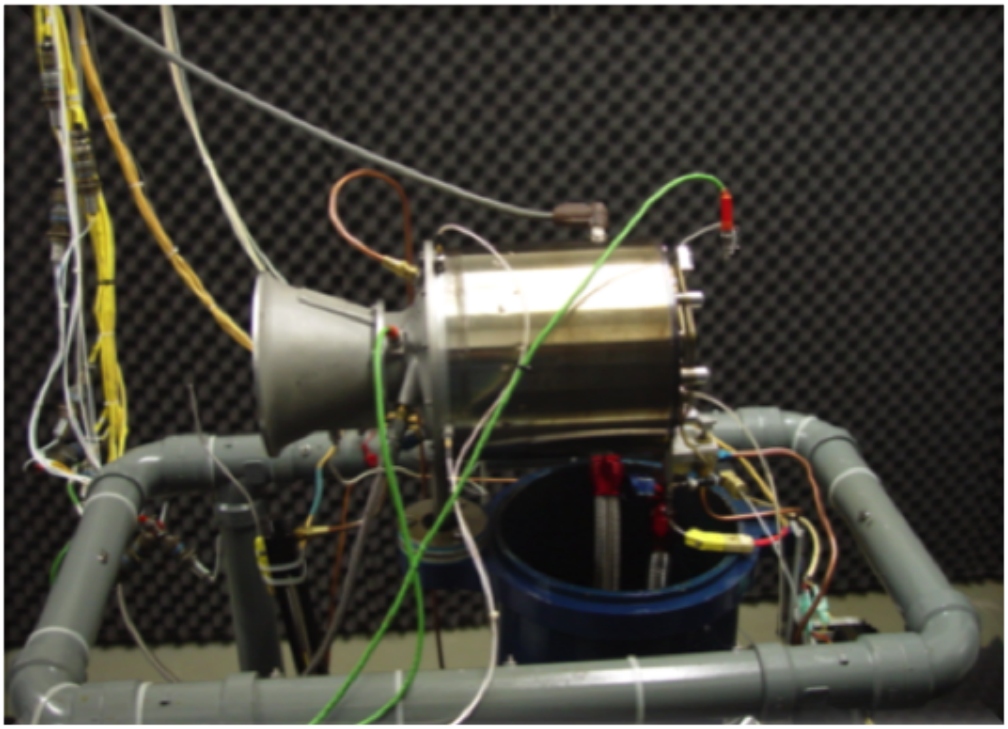
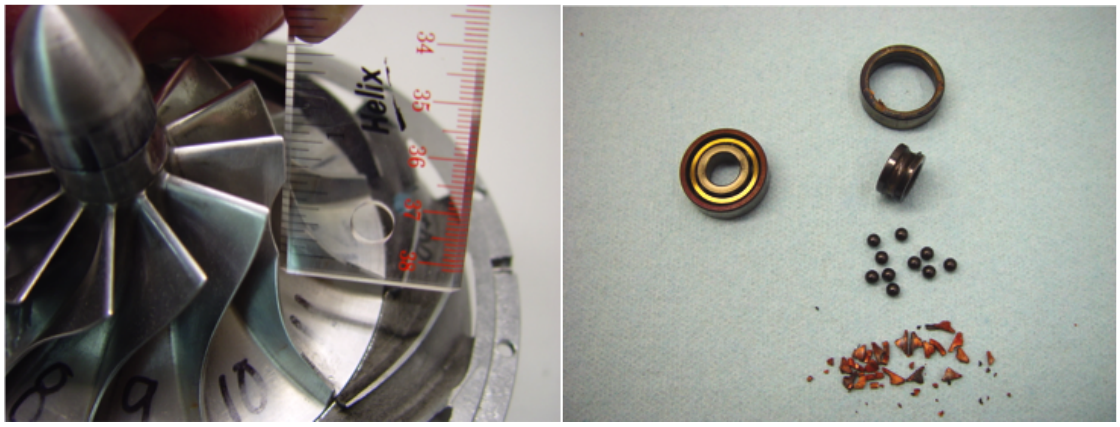


Figure A.1: **SR-30 Experimental Setup** - this image shows the Turbine Technologies SR-30 engine in the test cell, with added vibration and acoustic sensors. (photo courtesy of UTC Pratt-Whitney)

A.2 PW4000 Engine

The particular engine type (as there are many slight variations on the main model) is the PW4062, which is an axial-flow engine with two spools and a 94" fan. The compressor frequency (N1) and the turbine frequency (N2) differ because they are separate spools,



(a) Fan Blade Damage (b) LPT Bearing Failure

Figure A.2: **SR-30 Experimental Fault Details** - this figure shows the faults introduced into the SR-30 dataset. In (a), the fan blades were shaved off, resulting in simulated damage that can be characteristic of a chipping or object strike against the inlet of the engine. In (b), the bearings were damaged to simulate a fault characteristic of extreme wear-and-tear conditions. (photo courtesy of UTC Pratt-Whitney)

as typical of a high-bypass turbofan jet engine **??**. An image of the engine itself can be seen in Figure A.3.

There were three runs, corresponding to three various engine conditions: nominal, fan fault, and Low Pressure Turbine (LPT) fault. All 3 runs consisted of slow accelerations/deceleration test patterns. Imbalances were achieved by placing counterweights on blades in the Fan and LPT sections, as indicated in the diagrams shown in Figure A.4.

Vibration data from 9 sensors was collected on a DSPCon data acquisition unit and then converted into MATLAB readable files. A 25 kHz sampling rate was utilized to record 150-180 second test runs with a characteristic acceleration from approximately 6 kRPM to 10 kRPM and a subsequent deceleration back to 6 kRPM.

Figure A.3: **PW4000 Engine** - this picture shows a PW4062 engine as it appears on the test cell. The large add-on assembly seen in front of the fan includes sensors and instrumentation used for the collection of high fidelity data streams.The engine can be seen mounted at a test cell facility. Some of these facilities allow for the simulation of high altitude temperature and pressure conditions, though most measure engine operation within local ambient conditions that are referenced to earlier baseline recordings. (photo courtesy of UTC Pratt-Whitney)

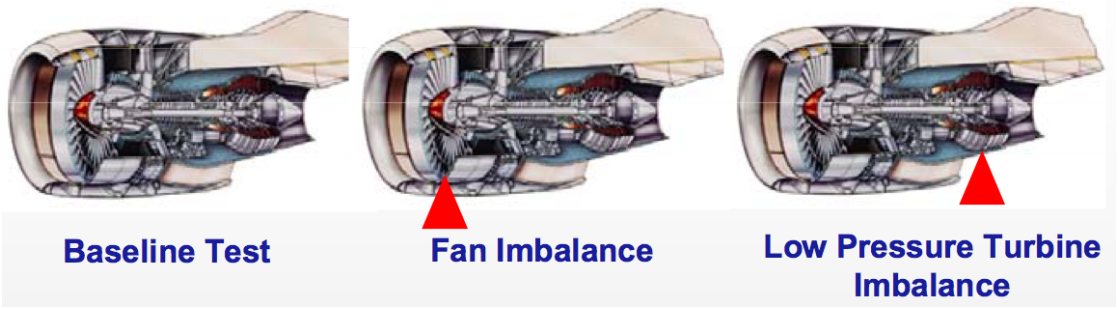


Figure A.4: **PW4000 Fault Locations** - this diagram shows cross-sections of the PW4000 where the faults were introduced into the test cell runs for the PW4000. Only vibration data was collected.

Appendix B

Table of CMAPSS Normalization Minima and Maxima

In this appendix, we briefly describe how the values used for CMAPSS parameter normalization were computed and list their final form.

As a brief recap of what is discussd in Section 6.3, normalization to a [0,100] interval is necessary for comparisons between real and simulated data from live engines. In the context of purely simulated data algorithm development, it makes no difference whether raw or normalized values are used, since no knowledge of the meaning of these values is utilized. However, when input data from a real engine is fed into CMAPSS, the resulting simulated outputs will be constrained to the thrust rating and control characteristics of the simulator - relating them back to the original data requires a re-scaling of both to a common reference point; hence, this normalization.

B.1 Computation

We decided that certain intermediate and output parameters crucial to the detection and diagnosis algorithms should be normalized to the interval [0,100] and were looking for suitable normalization extrema. Initially, we performed an investigation into the extrema present in the datasets themselves, both with and without the standard day corrections. We decided that the corrected values were more representative of the kind of data we

would see, and those values are shown in Table B.1 along the left side under the "Flight Simulation" heading.

This, however, may not be the best way to perform normalization since it only takes into account what exists in a particular dataset, and not all possible ranges of behavior. A more uniform method of normalization would use two types of baseline simulation to find the corresponding extrema that would be expected to provide better boundaries:

- Minima - to find the minima, we set up an "idle" type profile, which contained standard day behavior (sea level altitude and no change in DTamb) while the plane was at minimum TRA and zero mach speed. The choice of minimum TRA is a bit problematic, because at some early in the run, the simulation enforces a hard lower-bound of 10-20%, but later allows this to drop below this.

- Maxima - to find this, a "cruise" profile was constructed, with the plane in a fast linear climb from 0 to 30,000 feet above sea level, also at standard day conditions. The mach number steadily increased from 0 to 0.8 (which is the observed maximum from simulation data and not expected to be exceeded) while the TRA was constant at 100% engine thrust.

One thing to note is that these extrema may be reversed for the Variable Bleed Valve (VBV) and Variable Strator Vane (VSV), depending on the nomenclature for the simulator or engine. Manufacturers sometimes indicate these with respect to different reference points, and for CMAPSS the minimum/maximum for VSV was revresed.

Both of these simulations can be easily reproduced, and can hopefully be referenced to similar test-cell exepriments for real engines. The results of these two experiments are also shown in Table B.1, under the heading "Boundary Simulation".

B.2 Minima and Maxima

Shown below is the table listing normalization extrema for CMAPSS simulated data:

	Flight Simulation		Boundary Simulation	
Parameters	Min	Max	Idle Min	Takeoff Max
Wf	4347.3	22371	3979.0	25456
VBV*	0.0202	0.586	0.0	0.558
VSV*	6.6	19.8	4.77	11.53
Nf	1268.0	2349.5	1322.5	2405.3
Nc	7615.6	8934.7	7736,7	9127.5
T24	542.8	635.7	548.8	644.9
P24	16.7	27.7	17.0	29.0
T30	1144.4	1546.5	1185.7	1600.7
Ps30	162.6	485.5	169.6	536.8
T48	1,259.5	2004.1	1332.4	2156.3

Table B.1: List of finalized minima and maxima for relevant CMAPSS input and output parameters, with values for corrected simulation data (left) and idealized boundary simulations (right). The baseline simulations use a TRA of 10 for the idle and 100 for the takeoff, while flight simulation statistics are computed over the entire range of FOQA samples.

As seen from the table, the maxima from the boundary simulations generally exceed those from the flight simulation analysis, except in the case of VBV and VSV. The minima values from the constructed "idle" case, however, do not come close to the minima found from the examination of simulated data. This leads us to believe that there may be two processes that are distorting the results. First is the loose enforcement of the lower bound by CMAPSS - this means that real flights have a higher chance of entering extremely low TRA stages at the end of the flight when the plane lands or even glides towards landing.

This likely accounts for a large part of the disparity, but it is possible that for some parameter values, the minima and maxima do not usually occur at times of steady state but at times of extreme transience. This is especially true for VSV and VBV which are

both part of the control system that stabilizes engine operation. Regardless of whether the plane is operating at maximum or minimum TRA, if that operation is steady-state, these control mechanisms will not activate as fully as they would during rapid changes in engine operation mid-flight. As a result, additional distortions in the parameters of all boundary simulations may be occuring, but these are likely much smaller in magnitude than those caused by the first issue.

The method described in Section B.1 for determining the "Flight Simulated" values is a *semi-standardized* way of empirically determining operational extrema for engines in a laboratory test cell. Whether or not it translates directly to what can be done in simulation is uncertain. The partial mismatch between these extrema in the table and the differences between live and simulated QAR samples in Section 6.3 seem to suggest that this approach is overly optimistic, but at the very least it is easily defined and reproducible, independent of particular input data.

Appendix C

CMAPSS Simulated Datasets

In this appendix, we briefly review the datasets generated throughout the process of developing the fault detection and diagnosis system for Gas Path Analysis (GPA) described in Chapter 6. As discussed in Section 6.2, the Commercial Modular Aero-Propulsion System Simulator was used to create artificial flight data with simulated faults based on real-world input data from two sources: 68 anonymized Flight Operations Quality Assurance (FOQA) data files provided by NASA and 98 Quick Access Recorder (QAR) records from Korean Airlines GP 7270 engines (see Section 6.2.4).

In Table C.1 the possible faults that can be introduced into a CMAPSS simulation is listed. As visible from the highlighted portions of the table, our focus in this investigation was on primary turbomachinery component efficiency faults, which leaves a great many other potential faults for exploration. Such faults are potentially the most dangerous to the health of the aircraft if left unidentified and of greatest interest when it comes to a detection and diagnosis system; hence our interest in their generation within simulated flight data.

In CMAPSS, and by extension, the Transient Test Case Generator (TTCG), faults are introduced by distorting a nominal flight data file with a pre-determined fault setting. This distortion is performed by re-running the simulation with a specified fault component and fault value, and by using a nominal data file to base the simulation on. Thus, all datasets will consist of a set of base nominal files generated by running the simulation using input QAR or FOQA data (see Section 6.2.1 for details) along with

Class	ID	Comment	Range
Sensor Fault	Nf	Fan Speed	± 1 σ to ± 10 σ
	Nc	Fan Core	± 1 σ to ± 10 σ
	P2	Inlet Total Pressure	± 1 σ to ± 10 σ
	P24	LPC Exit Pressure	± 1 σ to ± 10 σ
	Ps30	HPC Exit Static Pressure	± 1 σ to ± 10 σ
	P50	LPT Exit Pressure	± 1 σ to ± 10 σ
	T2	Inlet Total Temperature	± 1 σ to ± 10 σ
	T24	LPC Exit Total Temperature	± 1 σ to ± 10 σ
	T48	Inter-Turbine Total Temperature	± 1 σ to ± 10 σ
Actuator Lockup Fault	Wf	Fuel Flow	± 1% to ± 7%
	VBV	Variable Bleed Valve	± 1% to ± 19%
	VSV	Variable Stator Vanes	± 1% to ± 7%
Turbomachinery Component Fault	Fan Eff	Fan Efficiency	± 1% to ± 7%
	Fan Flow	Fan Flow Capacity	1x to 2x Factor
	LPC Eff	LPC Efficiency	± 1% to ± 7%
	LPC Flow	LPC Flow Capacity	1x to 2x Factor
	HPC Eff	HPC Efficiency	± 1% to ± 7%
	HPC Flow	HPC Flow Capacity	1x to 2x Factor
	LPT Eff	LPT Efficiency	± 1% to ± 7%
	LPT Flow	LPT Flow Capacity	-1x to -2x Factor
	HPT Eff	HPT Efficiency	± 1% to ± 7%
	HPT Flow	HPT Flow Capacity	-1x to -2x Factor

Figure C.1: **Summary of Possible CMAPSS Faults** - this table details the CMAPSS faults that can be introduced into nominal flight data. The highlighted rows correspond to those faults that were introduced in this work: turbomachinery component efficiency faults*.

corresponding fault data files for each flight. Each number in the list below corresponds to the N parameter in the F_N and Q_N nomenclature discussed in Figure 6.7.

1. 3% Faults at takeoff, Random 1% Errors
2. 3% Faults just after takeoff, Random 1% Errors
3. 4% Faults just after takeoff, Random 1% Errors
4. 5% Faults just after takeoff, Random 1% Errors
5. 3% Faults at end of flight (descent/landing), Random 1% Errors
6. 3-5% Faults randomly throughout flight, Random 1% Errors

The "Random 1% Errors" added to each dataset constitute an attempt at increasing the difficulty with which different classes could be distinguished from each other. A similar effect could be achieved by adding noise to different components within CMAPSS, but as we did not have access to all the details of their implementation, we deemed it appropriate to add our own, well controlled, noise. Each of the datasets was tested without noise, and then with a gaussian white noise with a power of 1% of the local signal standard deviation (over a 1 minute window).

*** NOTE**: While fault intensities were focused on the efficiency degradation of turbomachinery components listed in Table C.1, it became apparent that there is an internal mechanism within CMAPSS for linking the flow and efficiency, even when only an efficiency fault is specified. Thus, it should be understood that ALL faults introduced in the data were a combination of an efficiency and a flow capacity degradation.

Appendix D

Q and Q' Results

In this appendix, we briefly list the results from the experiments performed using partial QAR original Q data and partial QAR simulated Q_N data, using a specialized mixture of the two datasets:

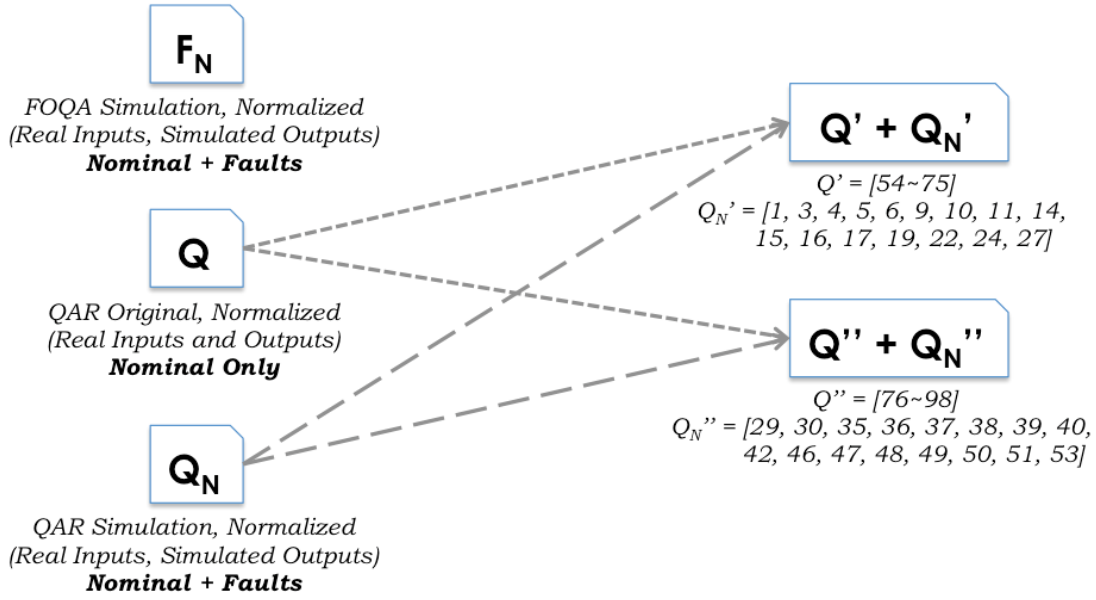


Figure D.1: **Outline of QQ' Setup:** The composition of the training and testing data consists of a mixture of QAR-driven simulated data and original QAR data (both corrected and normalized appropriately).

The hope of this experiment was to shed more light onto the results discussed in Section 6.5.3, and interpret the failures from those experiments in the context of QAR data only. The results of these experiments are shown in Figures D.2 and D.3 below.

Ground Truth \ Decision	Nom	Fan	HPC	HPT	LPC	LPT	
Nom	351035	2402	74	760	126	914	98.8%
Fan	21382	12947	21	1823	411	4633	31.4%
HPC	17237	1135	6050	8869	26	211	18.0%
HPT	27284	313	7441	7910	13	64	18.4%
LPC	36246	363	5	401	66	989	0.1%
LPT	8788	26513	7	2576	148	1988	5.0%
	76.0%	29.7%	44.5%	35.4%	8.4%	22.5%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from the **Q' and Q_N'** split*

Figure D.2: **QQ' Results, Per Window** - as seen in this table, the perception is that nominal conditions are correctly identified, while fault scenarios are completely inconsistent. In reality, the classifiers will all highly favor nominal (as seen in the left-most column) because of the high preponderance of nominal data in this dataset.

Ground Truth \ Decision	Nom	Fan	HPC	HPT	LPC	LPT	
Nom	39	0	0	0	0	0	**100%**
Fan	15	1	0	0	0	0	**6.3%**
HPC	16	0	0	0	0	0	**0%**
HPT	15	0	0	1	0	0	**6.3%**
LPC	16	0	0	0	0	0	**0%**
LPT	11	5	0	0	0	0	**0%**
	34.8%	16.7%	0%	100%	0%	0%	

** Results generated using 50/50 training/testing split with 10-fold random cross-validation, from the **Q' and Q_N'** split*

Figure D.3: **QQ' Results, Per Flight** - aggregating the results from Figure D.2, it is unsurprising to see that everything is again "forced" into the nominal class, resulting in highly skewed results and low overall performance.

In the above, initially one might be led to think that nominal classification, at least, is being performed well. The truth is less optimistic - the "Nom" column (indicating the classification decisions made in favor of the nominal condition) shows that the vast majority of test samples are classified as nominal regardless of the ground truth. The classifier, becuase of the deluge of nominal training samples, has become biased in favor of the nominal class.

The following tables show a set of binary scenarios - when nominal data is compared directly to only a single fault class. Some information can be gleaned from this as well: there is a high percentage of correct classification (without confusion) when nominal is compared to HPC and HPT faults, while all other fault types are highly confused with nominal. This would suggest the problem exists in the LPC and fan part of the engine, again pointing to a potential fan problem.

	Nom	**Fan**	
Segments	42322	250233	14.4%
Flights	4	62	6.1%

Figure D.4: **QQ' Binary Results, Nominal vs. Fan Fault** - a relatively low percentage of correct classification, with a high bias for the Fan Fault rather than Nominal.

	Nom	**HPC**	
Segments	282699	9856	**96.6%**
Flights	66	0	**100%**

Figure D.5: **QQ' Binary Results, Nominal vs. HPC Fault** - a high percentage of correct classification, with a small amount of overall confusion.

	Nom	HPT	
Segments	282728	9827	**96.6%**
Flights	66	0	**100%**

Figure D.6: **QQ' Binary Results, Nominal vs. HPT Fault** - a high percentage of correct classification, with a small amount of overall confusion.

	Nom	LPC	
Segments	41862	250693	14.3%
Flights	5	61	7.9%

Figure D.7: **QQ' Binary Results, Nominal vs. LPC Fault** - again, a low percentage of correct classification, confusing with LPC Faults.

	Nom	LPT	
Segments	43171	249384	14.7%
Flights	4	62	6.1%

Figure D.8: **QQ' Binary Results, Nominal vs. LPT Fault** - a low percentage of correct classification, confusing with LPT Faults.