Signal Processing and Fault Detection with Application to CH-46 Helicopter Data [1]

Fang Wen, Peter Willett
U-157, University of Connecticut
Storrs, CT 06269
860-486-2195, willett@engr.uconn.edu

Somnath Deb
Qualtech Systems, Inc.
6 Storrs Rd, Suite 6
(860) 423-2099, deb@teamqsi.com

Abstract

Central to Qualtech Systems' mission is its testability and maintenance software (TEAMS) and derivatives. Many systems comprise components equipped with self-testing capability; but, if the system is complex (and involves feedback and if the self-testing itself may occasionally be faulty) tracing faults to a single or multiple causes is difficult. However, even for systems involving many thousands of components the PC-based TEAMS provides essentially real-time system-state diagnosis.

Until recently TEAMS operation was passive: its diagnoses were based on whatever data sensors could provide. Now, however, a signal-processing (SP) "front-end" matched to inference needs is available. Many standard signal processing primitives, such as filtering, spectrum analysis and multi-resolution decomposition are available; the SP toolbox is also equipped with a (supervised) classification capability based on a number of decision-making paradigms. This paper is about the SP toolbox. We show its capabilities, and demonstrate its performance on the CH-46 "Westland" data set.

Table of contents

[1]© 2000 IEEE

1. Introduction

Diagnosis of abnormal operation condition is of interest in many or most systems, and is of singular importance to those systems which are airborne. This is not only due to the obvious consideration that a failed component may cause loss of life (or at least of a valuable asset), but also because systematic avoidance of failures means scheduled – rather than as-needed – part replacement and overhaul. Consequently in many such systems the cost of maintenance is considerably higher than the operating expenses, and indeed is comparable to the original system price.

Qualtech Systems has developed a suite of fault-isolation tools (TEAMS) which can, in real time and based on binary (good/no good) sensor data, isolate single and even multiple faults in complex systems. However, many sensors (for example, of vibration) are incapable of reliable decision-making on their own, and hence it has become necessary to develop a (real-time) signal processing "front-end" to the TEAMS inference engine whose goal is to render decisions as intelligent as possible. The signal processing system includes a wide menu of spectral and statistical manipulation primitives such as filters, harmonic analyzers, transient detectors, and multi-resolution decomposition; these and their interaction with the TEAMS inference engine are described in the paper.

If the mapping from fault to signal were in all cases well known (for example, that a defect on a gear tooth causes a specified harmonic power to increase beyond an acceptable level) the above toolbox would probably be sufficient. However, this is often not the case: faults appear gradually and can be difficult to discern even when the most informative "features" are specified (and noisy features ignored). Thus, it has become necessary to augment the signal processing kit with pattern classification software, including techniques based on restricted Coulomb energy (RCE), decision trees, learning vector quantization (LVQ), fuzzy logic, Bayesian data reduction (BDRA), multi-layer perceptrons (MLPs), and Gaussian mixtures. At present the former three are implemented.

Recognition of faults can hence be automated provided there is sufficient training data. This paper thus includes analysis of no-fault and seeded-fault vibration data from a CH-46 ("SeaKnight") helicopter aft gearbox as collected from a test-stand. Results show promising fault detection accuracy, particularly when learning is based on auto-regressive (AR) coefficient features.

In section 2 we discuss the signal processing toolbox, its capabilities and operation, and its fit with the TEAMS engine. In the following section 3 we go into detail about the toolbox classification techniques: LVQ, DT, RCE classifiers. And in section 4 we apply the signal processing and classifiers to the Westland helicopter dataset. Similar to results reported elsewhere, we find near-perfect fault-recognition accuracy, in our case with relatively small feature sets involving autoregressive coefficients.

2. The System

2.1. TEAMS

An embedded real-time fault detection and isolation solution will reduce the likelihood of loss of mission due to sudden failures, and improve system availability. Existing embedded systems are constrained by available memory and processing power, and are mostly limited to custom simplistic if-then-else rules embedded in the control logic. In an effort conducted with NASA-ARC in 1996, Qualtech demonstrated a real-time model-based reasoning engine, TEAMS-RT, for the detection and isolation of multiple faults. with key features:

1. Separation of the system-specific knowledge, captured in terms of models, from the fault-isolation methods. This allows for the same tool to be used on multiple systems using different models.
2. Ultra-compact memory requirements: only about 1MB for a subsystem with 1000 failure sources and 1000 test points.
3. Excellent performance using low-end microprocessors (e.g. less than 200ms for the abovementioned system on a 75MHz Pentium processor).
4. Ability to perform diagnosis in fault-tolerant systems with multiple modes of operation.

In this (original) version, the processing and memory requirements were found to scale super-linearly. Consequently, it was deemed infeasible to monitor large complex systems such as helicopters. More recently a distributed diagnosis architecture that allows scalability of this solution to multiple systems covering – say, an entire aircraft – has been developed. Representative results are given in table 1.

faults	good	bad	susp.	time
1	997	1	2	50
2	996	2	2	50
5	991	5	4	50
10	983	10	7	75
20	973	20	7	87

Table 1: Performance results of TEAMS-RT for simulated system with 1000 failure sources and tests. The fifth column represents the number of "suspicious" parts, and the time is in milliseconds.

2.2. Signal Processing

As first envisioned and implemented, the TEAMS-RT inference engine relied on binary "good/no-good" data as supplied by stand-alone sensors. For enhanced diagnostic – and particularly prognostic – capability, a signal processing functionality has been added. Most sensors (e.g. of vibration) come either with an inherent preset testing capability or with an add-on interface which supplies *fault* data, often with an operator-panel destination in mind. However, many sensors also yield access to their raw data. From the point of view of TEAMS this may be preferable, since in addition to the truism that unquantized data contains more information:

- TEAMS may operate more effectively using different, or even adaptive, set points (P_{fa}, P_d)
- sensor outputs may be correlated to observe faults not readily recognizable independently
- incipient faults may be predicted from trending

In view of the above, a signal processing (SP) toolbox has been developed.

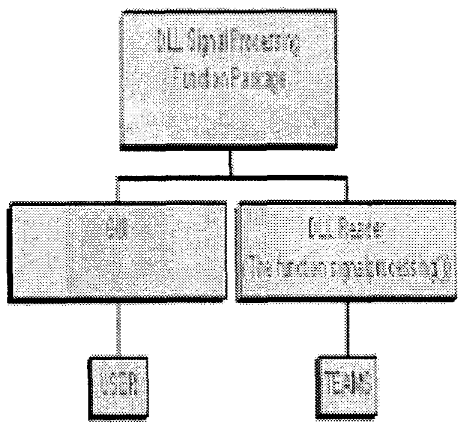


Figure 1: The SP toolbox exists in PC-based platforms as a DLL, callable either from TEAMS or from a user-friendly GUI via a C-structure wrapper.

As basic requirements it is clear that the SP toolbox must be flexible, portable, and fast. It is thus available for PC-based implementation as a dynamically-linked library (DLL), and since it is written in ANSI-C its migration to a UNIX-based platform is straightforward. With reference to figure 1 a C-structure "wrapper" provides a link to TEAMS, and an alternative graphical user-interface (GUI) affords the system designer direct access to the capabilities.

The SP toolbox is not intended for exploratory data analysis, simulation, or text manipulation: the sole goal is digestion of raw data to features, and to draw conclusions based on those features. The following functions have been implemented:

Filtering on block of data:

- design lowpass, highpass, bandpass, and bandstop filters, both windowed FIR and any of Butterworth, Chebychev, or elliptic IIR filters
- run a filter efficiently on a block of data

Spectrum analysis on block of data:

- FFT, Bartlett or Welch periodogram
- find dominant frequency
- find relative power in harmonics of dominant frequency
- find frequency sweep parameters
- find autoregressive (AR) parameters and spectrum

Empirical statistics:

- moments: mean, variance, skewness, and kurtosis
- order statistics

Sample-by-sample nonlinearities: logarithmic, polynomial (arbitrary order), soft-limiter

Binary-output tests:

- threshold test on sum
- Page (or CUSUM) test for quickest detection of statistical change

Multiresolution decomposition: representation via Daubechies "wavelets"

Classification on feature vectors:

- restricted Coulomb energy (RCE) neural network
- learning vector quantization (LVQ) classifier
- decision tree (DT) classifier
- fuzzy ARTMAP

Generally the above operations are performed on an entire file of data. For example, a filter based on 10000 input data yields an output file also of length 10000. A magnitude-square FFT output length would be 16384 (the next highest power of two), and a *skewness* interrogation would have output of unity length. In certain cases it is also appropriate that operations be performable on sub-blocks of data; for instance it may be requested to find the tenth-order AR coefficients on sub-blocks of length 200, in which case the output file would be concatenated from each block, a total of length $10 \times (10000/200) = 500$.

All the above functions are easily callable via a windows *dialog box*-based GUI, as pictured in figure 11; recall from figure 1 that the GUI calls the DLL workhorse, the same as is called directly by TEAMS when in run-mode. Communication with the DLL is in either case controlled by a special-purpose C-structure:

```
struct sig_proc_stages
{
    int input_signal_length;
    double *input_signal;
    int *output_signal_length;
    double **output_signal;
    int number_of_stages;
    char **stage_parameter_list;
}
```

in which *float* or *int* can replace *double* if speed is a particular consideration. The last element of the structure executes the control, and "tells" the DLL what sort of operation is to be performed. If the penultimate element is greater than unity, this indicates that multiple processing passes are needed. For example, to test the kurtosis of bandpass-filtered data which has been preprocessed to remove outlying observations, we might have:

```
stage_parameter_list[0]:
    TYPE=NONLINEARITY;
    NAME=SOFT_LIMITER;

stage_parameter_list[1]:
    TYPE=FILTER; NAME=BPIIR;
    SAMPLING_FREQUENCY=20[kHz];
    (filter coefficients)

stage_parameter_list[2]:
    TYPE=STATISTIC;
    NAME=Get_Moments;
    MOMENT_NUMBER=4;

stage_parameter_list[3]:
    TYPE=TEST;
    NAME=THRESHOLD_SUM;
    THRESHOLD=10;
```

The final output is the binary fault/no-fault signal necessary for input to TEAMS.

The functions of the signal processing toolbox are not unique: many systems, MATLAB naturally included, have capabilities beyond that of this set. The key to the SP toolkit is that is specifically targeted for tandem use with TEAMS in real-time systems health monitoring.

3. Classification Techniques

The previous section has discussed the workings of the combined TEAMS/SP-toolbox system. This powerful combination can be applied in many problems, but in this paper the primary target is the condition monitoring of a helicopter. Now, it would be appealing if the signs of impending or occurred malfunction were predictable based on physics or experience: for example, it may be expected that appearance of a seventh harmonic of the primary rotational frequency is symptomatic of a particular kind of damage. If this is so, then the SP toolbox is well-equipped to render the appropriate fault signal. However, there does not seem to be such a well-defined body of knowledge available; what *does* exist is a database of fault and no-fault signals.

As such, what is needed is a means of supervised (learning) classification based upon these training data sets. We offer in the following a brief discussion of the SP toolbox's classification capabilities.

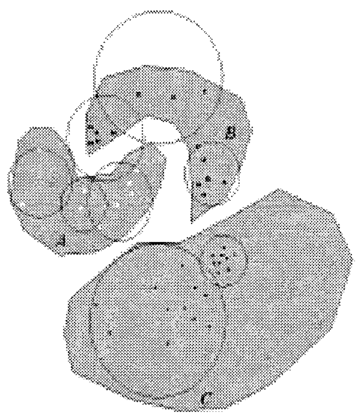


Figure 2: Illustration of classifier approximation via the RCE approach. The classes labeled A, B, and C, can generate observations within the indicated shaded regions. The actual training observations are given by dots or small squares. The approximating circles are shown also.

3.1. Restricted Coulomb Energy Classification

At fundament, the RCE classifier [3, 6] relies on the approximation of a decision region via a union of hypersphere "cells", as illustrated in two dimensions in figure 2. Cells may overlap if they do not belong to the same class, and this may produce ambiguous outputs. Note that partition of the observation space into decision regions is not exhaustive in the RCE approach.

Training of an RCE classifier is of course iterative, with the training data sets cycled repeatedly as training "epochs". Training is as follows:

1. Randomly shuffle training data.
2. Consider a training data point, and find those hyperspheres which contain it.
   (a) If there are none, then initialize a new cell centered at that data point and having a (pre-specified) maximum radius.
   (b) If a containing hypersphere is associated with the correct class, then do nothing.
   (c) If a containing hypersphere is of an incorrect class, then decrease its radius to correct this.
3. Repeat the above for all data in training set.
4. If the training epoch has resulted in changes neither in hypersphere membership nor in modification of hypersphere number or radius, stop. Otherwise return to 1.

We are not aware of a proof of convergence of RCE training, but we have observed no lack of convergence.

After the network has become fixed classification is accomplished by interrogation of membership of the various cells: each cell is assigned a class, and the output corresponds to that class. For the cases that data is either a member of *no* cell, or of several which are of different classes, the RCE classifier gives an indeterminate output: such cases may be decided randomly or by heuristic.

The RCE classifier appears to be a good choice when the classes are separable (i.e. an ideal classifier would operate without error) and when there is sufficient training data that separation is possible. This is similar to simpler setups such as linear and quadratic discriminant analysis [7]; but whereas those techniques must have decision boundaries either hyperplanar or hyperellipsoidal shells, the RCE decision regions can be quite weirdly-shaped and non-convex.

3.2. Learning Vector Quantization Classification

The LVQ classifier [4] is a variation on the traditional cluster-classifier based on K-means training [7]. In essence, each class is assigned sub-clusters defined by their centroids (see figure 3), and data are classified based on the membership of the centroid to which they are nearest.

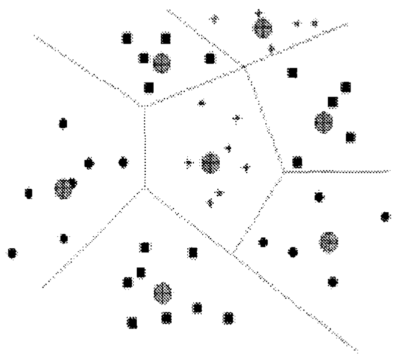


Figure 3: Illustration of classifier approximation via LVQ approach. Training observations are denoted as stars, boxes, or circles, depending on class. Subclusters are formed for each class, and the centroids for each are represented as $\oplus$.

Training of an LVQ classifier must proceed from an initial "guess" at cluster centroids; this may be from K-means. Thereupon consider training datum x_i, which is of class $C(x_i)$ and is closest to centroid μ_j: if the membership of μ_j is also $C(x_i)$, then we update in the direction of the new data

$$\mu_j^{new} = \mu_j^{old} + \eta(x_i - \mu_j^{old}) \qquad (1)$$

and otherwise in the opposite direction

$$\mu_j^{new} = \mu_j^{old} - \eta(x_i - \mu_j^{old}) \qquad (2)$$

Generally η is decreased from epoch to epoch. Eventual convergence is assured here provided η is sufficiently small, but in practice training is ceased when changes become small.

In our implementation clusters are never created, but may be merged. After cessation of training (as above), successive pairs of common-class clusters are testing for the appeal of their combination: for the proposed super-cluster a new mean μ^* and "radius" R^* are calculated, the former being the usual centroid and the latter the greatest Euclidean distance from the centroid to a member point. If this radius is less than a distance β to the nearest centroid *not* in the current class, then the merge is accepted, and otherwise it is not. Typically we have $.33 \leq \beta \leq 1$.

An LVQ classifier may be considered a development on the earlier K-means based cluster classifiers in that non-separable classes cause no intrinsic, and in that there is an intelligent means of "pruning" clusters.

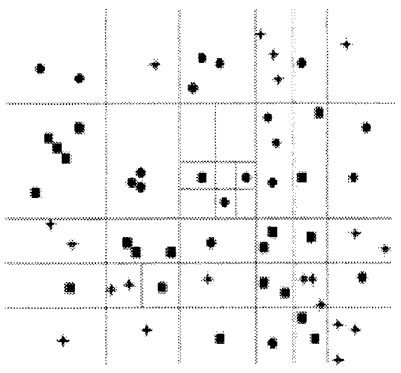


Figure 4: Illustration of classifier approximation via DT approach. Training observations are denoted as stars, boxes, or circles, depending on class. Subclusters are based on answering yes/no questions, and hence must be hyper-rectangular – in two dimensions, they must be rectangles.

3.3. Decision Tree Classifier

At core the DT classifier [7] produces its output by asking a series of questions which must have binary answers. These queries, for instance "Is feature 2 greater than 7.53?" may be based on previous answers, and each must interrogate one feature alone. The "path" taken may be thought of as traversal of a logical tree; but the form of the resultant decision regions must be as hyper-rectangles, as illustrated in figure 4.

In principle it is possible and easy to separate the training data precisely via a sufficiently-rich question set. In practice there are too many "questions" (*parameters*), and the DT classifier is found not to have a particularly good generalization ability. There are a number of means to limit the number of questions, and these generally amount to the choice of a cost to be placed on a question's posing. In our implementation we use an information-theoretic cost function, although admittedly its basis is empirical rather than true prior statistics.

4. Results on CH-46 Data

4.1. The Data

In the early 1990's the US Navy contracted with Westland, a British helicopter manufacturer, to develop and study vibration signatures for the CH-46 (SeaKnight) aft gearbox. Essentially this is "test-stand" (not in-flight) data; this is a disadvantage from the perspective of result reliability, but offers a distinct advantage in that the vibration signatures are *labeled.* Testing was at a variety of torque levels (since this is a rotorcraft angular velocities are relatively constant), and there are a number of faults and sensors used. Faults were "seeded" (by electronic discharge milling) in the sense that parts with known defects were installed and de-installed. Fault levels used here are either no-fault, level-1 spall, and level-2 spall.

test	defect	torque	acc.
9	0	light	4,7
9	0	medium	4,7
9	0	medium	3,4,7
9	0	heavy	4,7
9	2	medium	4,7
4.1	1	light	4,7
4.1	1	medium	4,7
4.1	1	medium	3,4,7
4.1	1	heavy	4,7
4.2	2	light	4,7
4.2	2	medium	4,7
4.2	2	medium	3,4,7
4.2	2	heavy	4,7
4.2	0	medium	4,7

Table 2: The Westland data files used for testing. The first column refers to the test number, the second to the kind of defect – either no-fault (0), or level 1 or 2 spalling, the third to the torque level (low is 100-108 ft. lbs., medium is 264-305 ft. lbs., heavy is 374-378 ft. lbs.), and the fourth the accelerometers (vibration sensors) from which data is available.

The data set presently available to us is as in table 2, and refer to the aft spiral bevel input pinion location. There are thus 31 data files, each of approximately 500000 samples and representing about 200 revolutions of the slowest gear.

The data has been analyzed previously (e.g. [1, 5, 8]) using a variety of classification techniques such as multi-layer perceptrons and fuzzy reasoning. Indeed, this is apparently an "easy" (or separable) dataset for classification, as the reported accuracies approach 100%. Thus, our goal here is not really to *beat* previous (unbeatable!) approaches, but to attempt to match them using the SP toolbox classifiers. Further, it appears that past approaches have often used a rather dense feature set (several hundred features, such as FFT outputs), and we attempt here to use a much sparser arsenal.

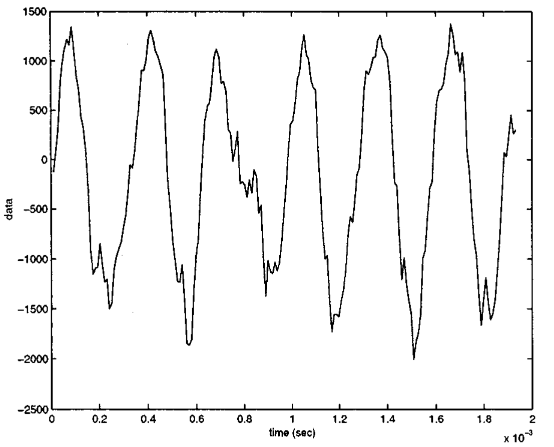


Figure 5: Example no-fault data from accelerometer 4, under light torque. Faulty data traces show no qualitative differences.

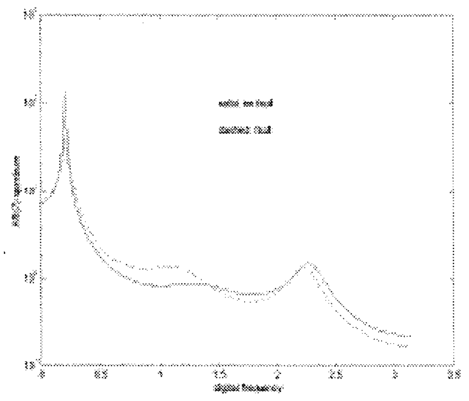


Figure 6: Spectrum via AR model from no-fault and level 2 fault.

An example of no-fault data from accelerometer 4 under light torque is given in figure 5. It is easily seen that the data is strongly tonal; unfortunately (or fortunately from the point of view of someone interested in classification), data from level 1 and level 2 faults "looks" similar. Figure 6 confirms the tonality, but shows that there is indeed a difference in spectral signatures between the conditions.

It is possible to use periodogram outputs explicitly as features for classification; however, in general this implies a great many features, and the usual "curse of dimensionality" may ensue. Since it is clear that spectral features do indeed yield much relevant information, we propose to use a concise way of representing the spectrum: the autoregressive (AR) parameters. An autoregressive modeling of a random signal implies that the output observations process $\{x_n\}$ is formed as

$$x_n = w_n - \sum_{k=1}^{p} a_k x_{n-k} \tag{3}$$

which is easily recognizable as the response of an all-pole IIR digital filter to a white noise input $\{w_n\}$. The power spectral density of a random signal so modeled is

$$S(\omega) = \frac{\sigma_w^2}{|1 + \sum_{k=1}^{p} a_k e^{j\omega k}|^2} \tag{4}$$

in which σ_w^2 is the variance of the (assumed) white noise input process $\{w_n\}$. It is straightforward to calculate the AR parameters using the *normal* or *Yule-Walker* equations (e.g. [2]) as

$$\begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix} = - \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{pmatrix} \tag{5}$$

in which r_k is the estimated k^{th} *autocorrelation lag* $\mathcal{E}\{x_n x_{n+k}\}$, best estimated directly for a block of data $\{x_n\}_{n=0}^{M-1}$ and small p as

$$r_k = \frac{1}{M-k} \sum_{n=0}^{M-k-1} x_n x_{n+k} \tag{6}$$

The form of solution implied by (5) takes on the order of p^3 operations; the Levinson-Durbin algorithm [2] affords a more efficient $\mathcal{O}(p^2)$ calculations, and this is used by the SP toolbox. We use the AR coefficients $\{a_k\}_{k=1}^{p}$ directly as features for classification; the driving noise level σ_w^2 can also be calculated rather easily, but we favor not using it since it (and not the AR coefficients) is amplitude (power) dependent.

Examples of AR coefficients are given in figures 7 and 8 for accelerometer 7 and for light and heavy torque levels, respectively. It is clear that in the AR coefficient domain the differences between the regimes become quite apparent. Note that these are only AR coefficients a_1 and a_2 plotted against one another – the model is AR(4), so there may be even further classification power in a_3 and a_4. However, note that the AR coefficients estimated under the different torque levels fluctuate considerably.

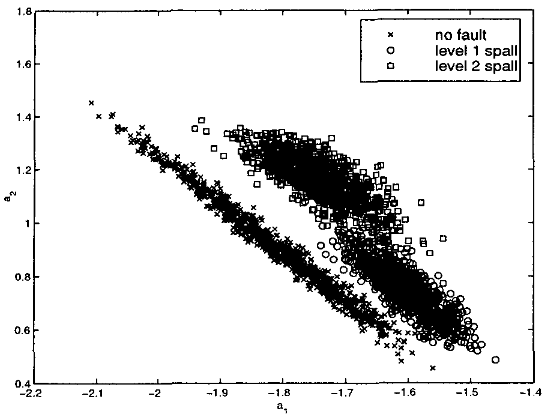


Figure 7: Scatter plot of AR coefficients a_1 versus a_2 (out of $p = 4$ AR coefficients) for the three fault conditions, under light torque.

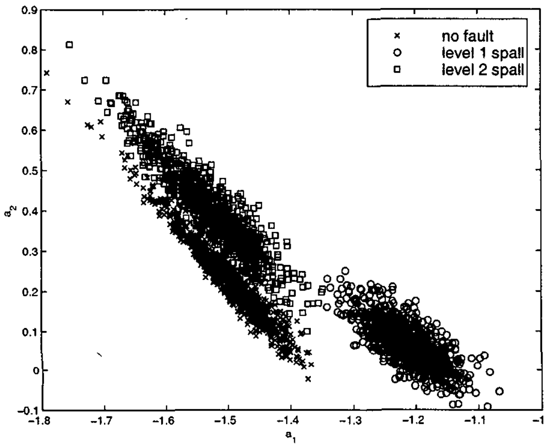


Figure 8: Scatter plot of AR coefficients a_1 versus a_2 (out of $p = 4$ AR coefficients) for the three fault conditions, under heavy torque.

4.2. Results Using AR Features

Separated Data: We have tested the classifiers described in section 3 on our data; the results are given in table 3. To derive this table, each dataset of table 2 is split into blocks of length 256 samples, and the AR coefficients of order p calculated for each. Then this list is split (randomly) into two subfiles: one for training and one for testing. The percentage of correct classification is shown in table 3, and it should be noted that this is an *average* value, meaning for example that the decision tree (DT) classifier working using 8^{th}-order AR coefficients achieved 91.4% accuracy on testing datasets as averaged across *all* 31 datasets in table 2.

p	RCE	LVQ	DT
2	66.3	68.5	78.5
3	83.3	54.0	80.4
4	89.6	70.8	87.2
6	90.7	67.7	89.9
8	91.4	72.5	91.4
12	91.5	65.8	90.6
16	92.3	67.1	92.1
20	87.5	67.3	87.3
24	87.0	67.3	87.3

Table 3: Percentage of correct classification for three classifiers based on a feature set given by AR parameters of order p, estimated on blocks of length 256.

We draw three conclusions. First, as is clear that our LVQ classifier is markedly inferior to the other two; and indeed the RCE classifier seems slightly better than DT. Second, the best AR order is in the range 8-16; when $p = 2, 3$ performance is poor, and likewise when the order is too large classification ability begins to drop off. In fact, $p = 4$ seems to achieve quite good accuracy. The third conclusion is perhaps the most important: using AR features as above we do *not* appear able to achieve the near-perfect discrimination reported by other authors.

p	acc.	RCE	LVQ	DT
3	4,7	99.1	90.1	98.3
4	4,7	99.1	94.7	96.2
5	4,7	98.3	96.2	98.4
2	3,4,7	99.8	98.2	99.6
3	3,4,7	99.3	97.7	99.6
4	3,4,7	98.2	96.7	98.7

Table 4: Percentage of correct classification for three classifiers based on combined feature set of AR parameters of order p from indicated accelerometers.

Combined Accelerometers: Part of the problem with the disappointing accuracy in table 3 is that, as mentioned, all datasets were used. It turns out that accelerometer 4 is a relatively poor predictor of health, whereas accelerometer 7 is much better. In table 4 we show the classification performance when datasets are combined – that is, AR coefficients from either two or three accelerometers (from table 2 we see that we have accelerometer 3 data only for some cases). For example, if third-order AR coefficients from accelerometers 3, 4, and 7 are combined to a feature of dimension 9, the RCE classifier is empirically 99.3% accurate. It is clear that this is a significant improvement over table 3, even with low-order AR coefficients.

M	acc.	RCE	LVQ	DT
64	4	26.7	33.9	31.2
256	4	89.6	70.8	87.2
1024	4	96.7	91.0	97.2
4096	4	100	100	98.9
16384	4	100	100	95.6
256	4,7	99.1	94.7	98.8
4096	4,7	100	100	99.4

Table 5: Percentage of correct classification for three classifiers based on combined feature set of AR parameters of order $p = 4$, as estimated on blocks of length M, from indicated accelerometers.

Block Length: The previous two tables 3 and 4 have used the block length 256 – that is, with reference to (6) we have $M = 256$. From table 5 it is apparent that increasing the block length on which the autocorrelations are estimated improves the classification accuracy dramatically. The choice of block length $M = 4096$ may be appropriate. It is interesting, and perhaps puzzling, that the accuracy of the decision tree (DT) classifier degrades with increasing M.

training	testing	RCE	LVQ	DT
light	medium	16.2	33.3	45.6
light	heavy	0.3	33.3	33.8
heavy	light	33.3	78.1	33.3
heavy	medium	33.6	56.6	66.7
combined	combined	99.8	78.4	92.6

Table 6: Percentage of correct classification for three classifiers based on combined data for accelerometers 4 and 7, AR order $p = 4$, and block length $M = 1024$. The classifiers are trained on a dataset with the indicated torque level, and tested on a dataset with a different torque level. The final row refers to the case of combined-torque-level training.

Torque Level: In our previous results torque level is not a parameter. That is, a classifier trained at a low torque level is also tested at a low torque level, etc. (see table 2). This may be reasonable, since the engine (or diagnostic system) will "know" the torque level and can be assumed to use an appropriate classifier for that torque level. However, since torque levels do not necessarily easily quantize themselves into "light", "medium", and "heavy", it would be useful to have a classifier which was independent of torque level.

It may be that the classifiers already designed possess that kind of robustness. In table 6 we use classifiers trained on light and on heavy torque datasets – even with combined accelerometer data – to test on all: the results (ignoring the final row) are not very impressive. Particularly the sensitive RCE classifier is poor.

We therefore train classifiers based on a *combined* training dataset at all torque levels. The results are shown in the last row of figure 6, and except for the LVQ classifier, are very good. Apparently by such "cross-training" there is no need to have a different classifier for each torque level.

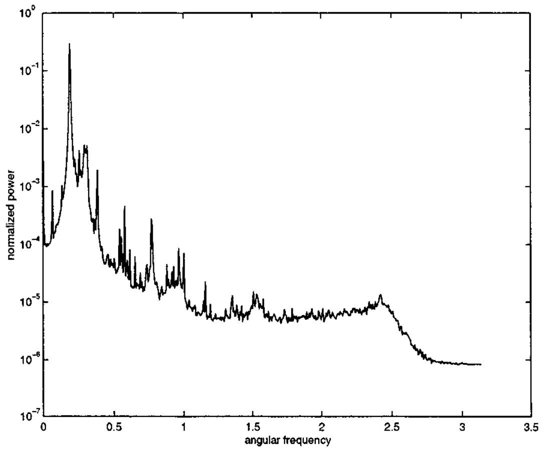


Figure 9: Example normalized (to total power) periodogram of accelerometer 7 under no-fault conditions and heavy torque.

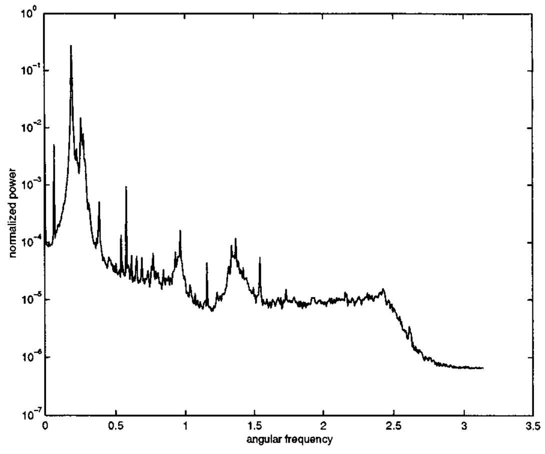


Figure 10: Example normalized (to total power) periodogram of accelerometer 7 under level 2 fault conditions and heavy torque.

4.3. Augmentation by FFT Features

The prior analysis used AR coefficients as features for classification. In this section we investigate the use of FFT features. Example periodograms are shown in figures 9 and 10 for no-fault and serious-fault conditions, respectively. It is clear that the spectra are different, and in fact that there is a greater distribution of power to harmonic peaks in the latter. The autoregressive spectrum provides broad (smooth) frequency behavior; but direct access to the level of harmonic power is often lacking.

As such, we take as features the level of power in the highest 4 periodogram peaks – relative to the total power so that the features are amplitude independent. The results are shown in the first row of table 7, and are not particularly encouraging. Apparently, while there is some diagnostic capability in the new features, performance is not as good as for the AR coefficients alone. However, combining this feature with the 4^{th}-order AR coefficients (i.e. a feature vector now of length 16), as shown in the second row, yields a considerable improvement: there are essentially no errors.

features	RCE	LVQ	DT
FFT	85.3	80.3	86.3
FFT+AR	99.8	99.8	99.8

Table 7: Percentage of correct classification for three classifiers based on combined data for accelerometers 4 and 7 with block length $M = 1024$. Data is averaged over all torque levels. The upper row uses the FFT (relative power in the highest four peaks) alone; lower row combine this with 4^{th}-order AR coefficients.

5. Summary

Qualtech Systems has developed TEAMS, an inference mechanism for automatic condition monitoring and diagnosis of complex systems. TEAMS relies on binary good/bad sensor data; refinement is expected if the data is "raw", but there must be a signal processing component to extract useful testing information for TEAMS.

Here we have reported on a signal processing toolbox specially matched to TEAMS. Its capabilities are mainly in the digestion of useful information from time series, but it has inherent a classification capability. We have used both the signal processing and classification on an abbreviated version of the "Westland" helicopter data. We have found that near-perfect classification is possible via use of autoregressive coefficients as features, particularly when these are combined from multiple accelerometers, and especially when these are combined with spectral peak relative power information. Classification is possible independent of torque level. The restricted Coulomb energy (RCE) classifier is particularly promising.

6. Acknowledgments

This research has been supported by US Army STTR contract DAAG55-98-C-0057, with prime contractor Qualtech Systems, Inc., and a subcontract to the University of Connecticut. The authors also wish to acknowledge the help of Vivek Rajan for his work on the classification package, the support of NASA, and to express their thanks to Jack Tinkel, James Rozak, and Nathan Merritt of Sikorsky Aircraft for providing the datasets and advice.

References

[1] B. Cameron, "Final Report on CH-46 Aft Transmission Seeded Fault Testing", *Westland Research Paper RP907*, Westland Helicopters, 1993.

[2] S. Haykin, *Adaptive Filter Theory*, 3^{rd} ed., Prentice Hall, 1996.

[3] M. Hudak, "RCE Classifiers: Theory and Practice", *Cybernetics and Systems: An International Journal*, vol.23, pp483-515, 1992.

[4] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ-PAK: A Program Package for the Correct Application of Learning Vector Quantization Algorithms", *Proceedings of the IJCNN*, pp725-730, 1992.

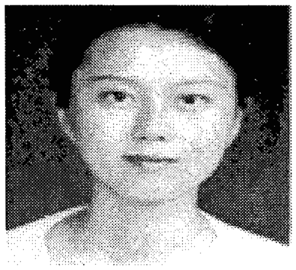
[5] P. Monson, M. Dwonczyk, and E. Manolakos, "Analog Neural Networks Based Helicopter Gearbox Health Monitoring System", *Journal of the Acoustical Society of America*, vol.96(6), pp3235-3249, 1995.

[6] D. Reilly, "The RCE Neural Network", *CRC Press Industrial Handbook*, pp.1025-1037, 1997.

[7] B. Ripley, *Pattern Recognition and Neural Networks*, Cambidge University Press, 1997.

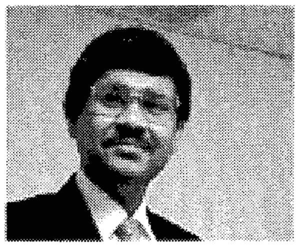
[8] G. Yen, "Health Monitoring of Vibration Signatures in Rotorcraft Wings", *Neural Processing Letters*, vol.4(3), pp127-137, 1996.

Biographies



Fang (Flora) Wen was born on February 4, 1971. She received her B.S. degree in opto-electronic engineering from Huazhong University of Science and Technology in 1992, and her M.S. in electrical engineering from Beijing University of Posts and Telecommunications in 1995. She was a telecommunication system engineer in Wuhan Research Institute of Posts and Telecommunications from 1995 to 1998. She is currently enrolled as a graduate student/research assistant at University of Connecticut, pursuing her PhD degree. Her primary research interest is in digital signal processing.

Peter Willett is a professor at the University of Connecticut, where he has worked since 1986. Previously he was at the University of Toronto, from which he received his BASc in 1982, and at Princeton University from which he received his PhD in 1986. His interests are generally in the areas of detection theory and signal processing. He is a senior member of the IEEE, and is an associate editor both for IEEE Transactions on Aerospace and Electronic Systems and for IEEE Transactions on Systems, Man, and Cybernetics.



Somnath Deb, Chief Scientist at Qualtech Systems, Inc., received the B.Tech. degree in Electrical and Electrical Communications Engineering from IIT-KGP, India (1987) and M.S. and Ph.D. degrees in Control and Communication Systems from the University of Connecticut in 1990 and 1994, respectively. Dr. Deb's research has included the development of advanced optimization algorithms for systems testability analysis and improvement and multisensor multitarget tracking and data association. He has published over 30 journal and conference papers. He received the Best Technical Paper Awards at the 1990 and 1994 AUTOTEST Conferences for his work on tools for system testability analysis and multi-signal modeling. He is a senior member of IEEE.

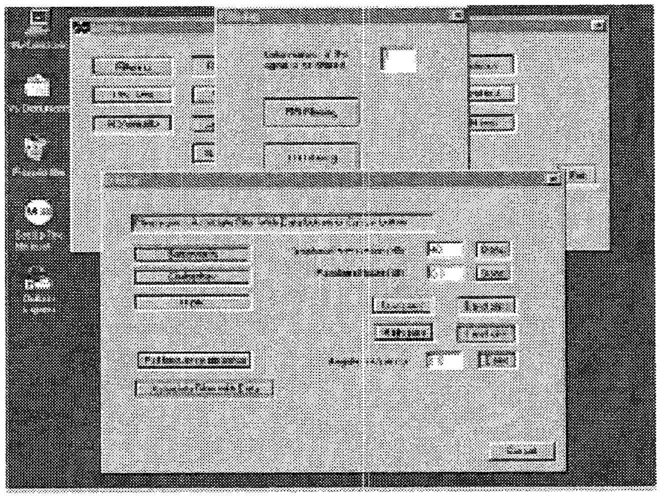


Figure 11: GUI dialog box for design of elliptic IIR filter. Preceding dialog boxes are seen in background.

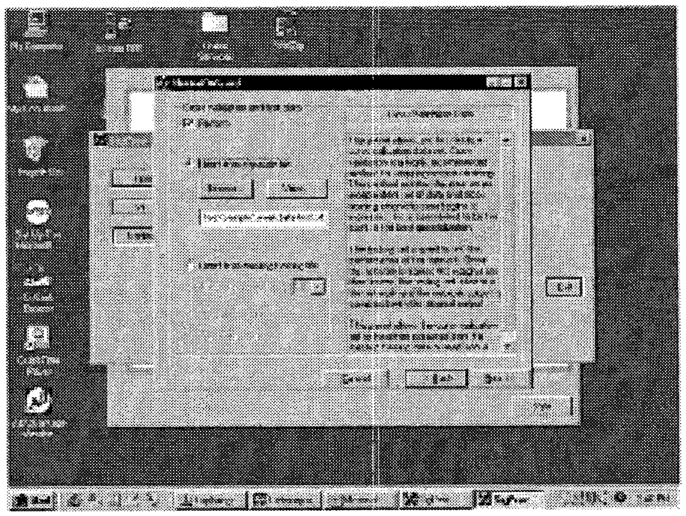


Figure 12: GUI dialog box for specification stage of RCE neural network. Preceding dialog boxes are seen in background.