UC San Diego

UC San Diego Previously Published Works

Title

Structural health monitoring feature design by genetic programming

Permalink

https://escholarship.org/uc/item/86q725h3

Journal

Smart Materials and Structures, 23(9)

ISSN

0964-1726

Authors

Harvey, Dustin Y

Todd, Michael D

Publication Date

2014-09-01

DOI

10.1088/0964-1726/23/9/095002

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Structural health monitoring feature design by genetic programming

Dustin Y. Harvey[1] and Michael D. Todd[1]

[1]Department of Structural Engineering, University of California, San Diego, 9500 Gilman Drive, San Diego, CA 92093 USA

Abstract

Structural health monitoring (SHM) systems provide real-time damage and performance information for civil, aerospace, and other high-capital or life-safety critical structures. Conventional data processing involves pre-processing and extraction of low-dimensional features from in-situ time series measurements. The features are then input to a statistical pattern recognition algorithm to perform the relevant classification or regression task necessary to facilitate decisions by the SHM system. Traditional design of signal processing and feature extraction algorithms can be an expensive and time-consuming process requiring extensive system knowledge and domain expertise.

Genetic programming, a heuristic program search method from evolutionary computation, was recently adapted by the authors to perform automated, data-driven design of signal processing and feature extraction algorithms for statistical pattern recognition applications. The proposed method, called Autofead, is particularly suitable to handle the challenges inherent in algorithm design for structural health monitoring problems where the manifestation of damage in structural response measurements is often unclear or unknown. Autofead mines a training database of response measurements to discover information-rich features specific to the problem at hand. This study provides experimental validation on three structural health monitoring applications including ultrasonic damage detection, bearing damage classification for rotating machinery, and vibration-based structural health monitoring. Performance comparisons with common feature choices for each problem area are provided demonstrating the versatility of Autofead to produce significant algorithm improvements on a wide range of problems.

Keywords: feature design, structural health monitoring, genetic programming, algorithm design, signal processing, pattern recognition

1. Introduction

Structural health monitoring (SHM) systems provide real-time damage and performance information for civil, aerospace, and other high-capital value or safety critical structures. Summary reviews on the field are provided in Doebling, et al. [1] and Farrar, et al. [2]. For the interested reader, Farrar and Worden provide a more comprehensive treatment of the field of SHM in their recent work [3]. The conventional data flow for SHM systems begins with acquisition of time series structural response measurements. After pre-processing, a set of one or more damage-sensitive features is calculated in a feature extraction process, which, for data-driven SHM applications, is then fed to a pattern recognition algorithm to perform the desired task. The classification or regression analysis forms the basis for decision-making under the uncertainty and noise that typically affect the SHM process.

Development of effective features for a specific SHM system presents two key difficulties. First, the features must be sensitive to the damage states of interest in the system such that statistically detectable changes occur when the system undergoes damage. Traditionally, designers of feature extraction algorithms have relied either on engineering judgment of the expected behavior of the system under damage or on the parameters and predictive errors of models, which can be data-based or physics-based [3]. Second, the features must be insensitive to the varying operational or environmental conditions the system may experience in use. SHM axiom IVb states that "Without intelligent feature extraction, the more sensitive a measurement is to damage, the more sensitive it is to changing operational and environmental conditions" [4]. The design of robust, damage-sensitive features for a specific application can be an expensive and time-consuming task requiring extensive system knowledge and domain expertise, or absent that, substantial trial-and-error.

The objectives of a fully data-driven approach to feature design are thus two-fold. First, the question is posed "How should we process response measurements to extract damage information when minimal system and domain knowledge is available?" An automated, data-driven approach to feature extraction algorithm design directly infers from a training database of response measurements a feature set specific to the problem at hand. Such an automated approach provides the additional benefit of significantly reduced time and effort

for the overall SHM system design process. Second, we propose that the additional goal of designing robust features may be met directly by including examples of the expected operational and environmental variability within the training database. Thus, the fundamental assumption in this approach is that data are available that are known to span the desired classification or regression spaces.

The adopted approach to automated feature extraction algorithm design, called Autofead, uses a custom genetic programming variant to search for the optimal processing path from measured responses to features relevant to the SHM task [5]. Genetic programming is an evolutionary, heuristic search method in the program space which has been successfully adapted to a wide range of data mining and engineering design problems [6,7]. In Autofead, candidate feature extraction algorithms are evolved using a wrapper approach where fitness is computed as the error of a standard pattern recognition algorithm using the measured features.

This paper overviews the Autofead method and its applications in the field of SHM as well as providing three experimental studies on a wide range of SHM problems. The rest of this paper is organized as follows. Section 2 details the current Autofead method. Sections 3, 4, and 5 demonstrate the application of Autofead to three SHM problem domains with comparisons to conventional feature choices. Section 3 describes a simple ultrasonic damage detection problem. An experiment on damage type classification for a bearing in a rotating machine is presented in section 4 followed by a vibration-based damage extent estimation experiment in section 5. Section 6 draws conclusions and suggests directions for future work.

2. Autofead

Autofead is an SHM system development tool for autonomously designing feature extraction algorithms from given data sets. Autofead uses a genetic programming variant to solve the feature extraction algorithm design problem as depicted in Figure 1. Algorithms are induced directly from a database of labeled dynamic measurements. Any numeric sequence (one-dimensional signal) can be used as input, such as time series, spectral measurements, frequency response functions, or mode shapes. Solutions represent feature vectors evolved in the program space from a pre-defined function library containing common mathematical and

digital signal processing operations. Solutions are evolved based on error-based fitness measures from a pre-selected pattern recognition algorithm.

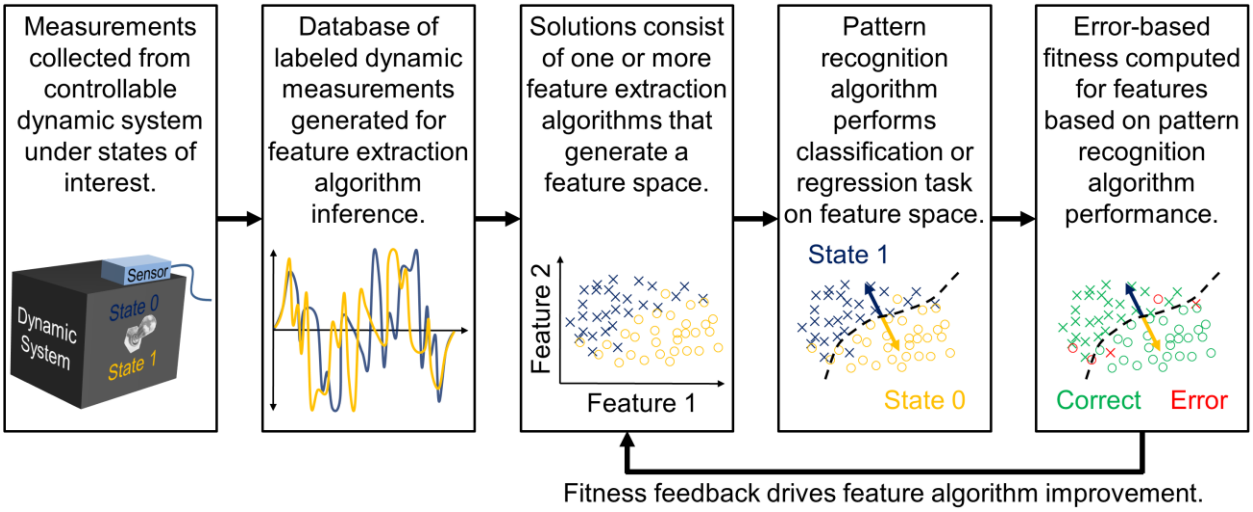

Figure 1. Autofead search data flow diagram.

The solution structure and function library designed for Autofead are introduced in sections 2.1 and 2.2, respectively. Section 2.3 presents the overall search process followed by sections detailing the four primary Autofead modules: parameter optimization, fitness cross-validation, evolution strategy, and breeding in sections 2.4 through 2.8. Finally, section 2.8 summarizes the configuration of Autofead for the experiments to follow.

2.1. Solution structure

The Autofead solution structure is highly-constrained and simplified compared to other genetic programming systems developed for general purpose feature design such as PADO [8], Zeus [9], and FIFTH [10]. Each solution consists of a set of features where each feature is a sequence of functions operating on one or two inputs. Features must terminate with a dimension-reduction function such as a summation to output a scalar feature from its sequence inputs. This highly restricted structure was chosen primarily for two reasons: first, most conventional algorithms are linear in structure and operate on one or two signals; second, inclusion of high-level operations within a large function library necessitates constraints on the solution structure to restrict the size of the solution space.

Figure 2 diagrams an example individual containing three features. An individual represents a single candidate solution within the evolving Autofead population. The example shown includes 3 features of increasing complexity through the inclusion of merging functions and the *sliding windows* functions in features 2 and 3, respectively.. Merging functions allow a feature to operate on two input sequences of the same length. Each sequence is processed identically through functions prior to the merging function. The *sliding windows* function transforms the input sequence to a matrix of subsequences. Subsequent functions operate on each subsequence individually until the first dimension-reduction function returns the matrix to a single sequence. A second dimension-reduction function is then required to compute a scalar feature. *Sliding windows* is especially useful to identify transients and non-stationary behavior in the input sequence.

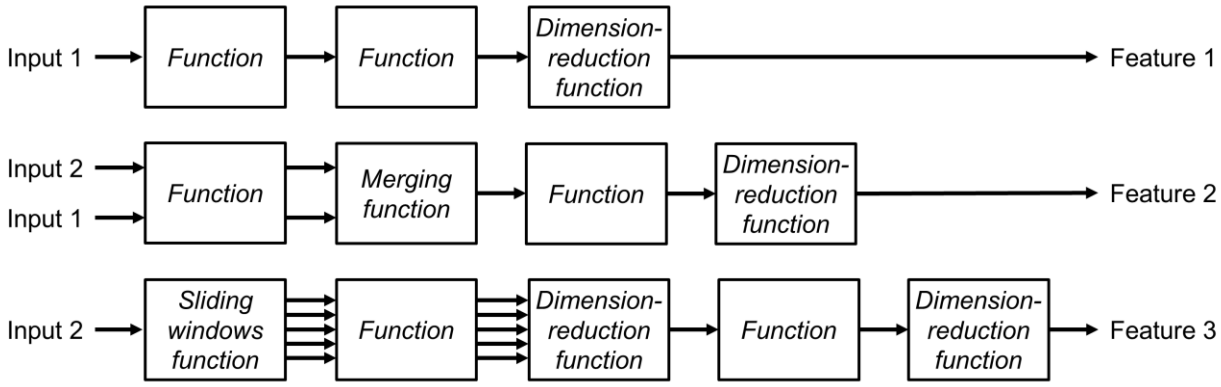

Figure 2. Example individual solution structure including a simple Feature 1, Feature 2 with a merging function, and Feature 3 using the *sliding windows* function.

2.2. Function library

The Autofead function library is based on a decomposition of common signal processing and feature extraction algorithms into their basic operations. Table 1 lists each function and its output. The behavior of some functions is controlled by an additional scalar parameter. For example, *low-pass filter* requires specification of a cutoff frequency parameter. These parameters are optimized prior to fitness evaluation through the numerical optimization scheme presented in section 2.4.

Table 1. Autofead function library

Function	Output	Parameter (*integer type)
Element operations		
Cube	Cube of input values	-
Square root	Square root of magnitude of input values, sign retained	-
Exponential	Exponential function of input values	-
Sigmoid	x/(1+|x|) for input values, x	-
Absolute value	Absolute value of input values	-
Square	Square of input values	-
Inverse	Reciprocal of input values	-
Sign	Sign of input values, -1 or 1	-
Log10	Base-10 logarithm of magnitude of input values	-
Distribution-altering functions (sample order independent)		
Demean	Input sequence with mean offset removed	-
Normalize	Standard deviations from mean of values in input sequence	-
Center	Input sequence with mean of entire dataset removed	-
Center and scale	Standard deviations from mean of entire dataset for values in input sequence	-
Set minimum value	Input with values below threshold raised to threshold	Data range threshold (0-1)
Set maximum value	Input with values above threshold lowered to threshold	Data range threshold (0-1)
Control chart	Input with central values of data range set to center of range	Percentage of range kept (0-1)
Sort order	Indices of values in sorted input sequence, ascending order	-
Order-dependent functions		
Difference	First differences of input sequence	-
Cumulative summation	Cumulative summation of input sequence	-
Hanning window	Input sequence with Hanning window applied	-
Low-pass filter	Low-pass filtered input sequence, zero-phase digital filtering by 3rd order Butterworth filter	Normalized cutoff frequency (0-1, upper bound = π rad/s)
High-pass filter	High-pass filtered input sequence, zero-phase digital filtering by 3rd order Butterworth filter	Normalized cutoff frequency (0-1, upper bound = π rad/s)
Auto-correlation function	Biased estimate of input sequence auto-correlation function for positive lags	-
FFT magnitude	Magnitude of FFT of input sequence, [0,π] rad/s bin range	-
FFT phase	Phase of FFT of input sequence, [0,π] rad/s bin range	-
FFT real	Real part of FFT of input sequence, [0,π] rad/s bin range	-
FFT imaginary	Imaginary part of FFT of input sequence, [0,π] rad/s bin range	-
Hilbert magnitude	Magnitude of Hilbert transform of input sequence	-
Hilbert phase	Phase of Hilbert transform of input sequence	-
Hilbert imaginary	Imaginary part of Hilbert transform of input sequence	-
Convolve	Convolution of input sequence with pre-specified sequence	-
Wavelet	Detail coefficients of discrete-wavelet transform using pre-specified wavelet family, parameter value of 0 produces approximation coefficients at the maximum useful level of decomposition	*Wavelet detail level
Index-altering functions		
Keep beginning	Input sequence with samples removed from end	*Samples to remove
Keep end	Input sequence with samples removed from beginning	*Samples to remove
Sliding windows	For non-windowed input*:* Subsequences extracted from sliding a window along input sequence, number of windows and overlapping samples between adjacent windows determined internally	*Window length
Transpose windows	Swaps window indices and sample indices	-
Merging functions		
Element sum	Element-wise sum of two input sequences	-
Element difference	Element-wise difference of two input sequences	-
Element product	Element-wise product of two input sequences	-
Element quotient	Element-wise quotient of two input sequences	-
Cross-correlate	Cross-correlation of two input sequences, output is same length as inputs	-
Dimension-reduction functions		
Sum (implicit)	Sum of all input values, occurs at end of every feature	-
Slope fit	Slope of best linear fit to each input sequence	-
Sliding windows	For windowed input*:* Sequence of sums of subsequences	-
Select	Sample at index of input sequence; internal brute force index search to optimize performance of output as single feature solution; for windowed input, each subsequence is treated as a separate fitness case index selection	*Internal index selection
Sorted bisection select	Same as select function but with input sequence sorted and using bisection index search	*Internal index selection

2.3. Search process

To begin an Autofead search run, the user defines the fitness cases and selects any custom run parameters. A fitness case consists of one or two input sequences and the target output value or class. Then, an initial population is randomly generated. The size of feature algorithms and feature vectors within the initial population is varied, and the use of the functions from the library in the initial population is uniform. Next, the search process begins as shown in Figure 3 with optimization of the parameters within the feature algorithms. After parameter optimization, *N*-fold cross-validation is carried out to determine the fitness of

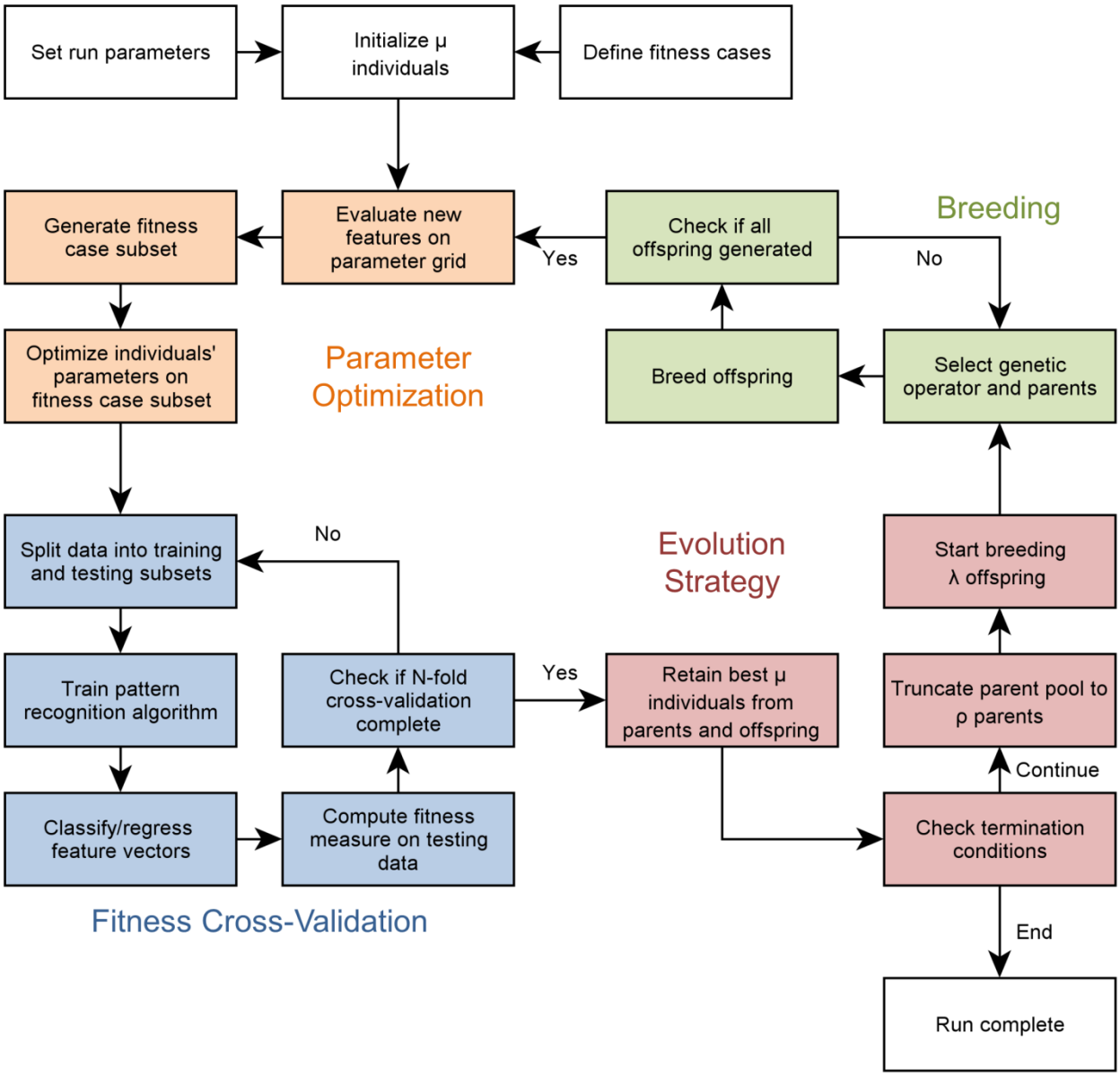

Figure 3. Autofead search process diagram.

each individual. New solutions are generated preferentially from higher fitness individuals in the breeding process. The run continues until termination conditions are met such as a desired fitness level or a maximum number of individuals.

2.4. Parameter optimization scheme

Before feature vectors can be evaluated for fitness, the parameter values within the algorithms must be selected. A single individual can contain multiple features each containing multiple parameters. Due to the nature of the parameter surfaces, a global optimization strategy is required; however, global optimization of all parameters within an individual at each iteration of the search process is prohibitively computationally expensive. Therefore, a sequential global optimization strategy is employed. First, an adaptive parameter grid is built and evaluated for each new feature in the population. Grid evaluations are stored for subsequent optimizations to reduce repeated computations. Next, parameters are locally optimized starting from the best grid point on a randomly selected subset of the fitness cases.

Additionally, within a single individual, each feature is optimized independently to avoid the requirement of a multivariate objective function. Independent optimizations, however, may lead to high correlation between features if no method is used to promote orthogonality in the optimization process. For example, in an individual containing two identical algorithms that estimate a natural frequency in a parameterized frequency band, performance will improve if the two features operate in separate frequency bands. Therefore, a fitness case weighting scheme is employed. After the first feature's parameters are optimized, error-based weights are assigned to each fitness case such that difficult fitness cases receive higher weights. Subsequent features' parameters are optimized according to a weighted objective function with the weights adjusted after each optimization.

2.5. Fitness cross-validation

Fitness of candidate individuals is based on performance of a pre-selected standard pattern recognition algorithm. For example, Gaussian Naïve-Bayes, decision trees, and *k*-nearest neighbors may be used for classification tasks. Linear regression is a common choice for regression problems. Error-based fitness

measures are computed based on the performance of the pattern recognition algorithm using each candidate feature vector through *N*-fold cross-validation.

2.6. Evolution strategy

The evolution strategy determines how the population is controlled and allowed to evolve. A generational approach is the simplest strategy where the entire population is replaced with an equal number of offspring during each iteration of the search process. An issue with the generational approach arises when the search moves in a detrimental random direction such that the offspring generation has lower fitness than the parents. Introducing elitism is one method to alleviate this problem. Autofead adopts a more advanced ($\mu/\rho+\lambda$) strategy based on methods in Back [11] which maintains a large diverse population of μ individuals. During each iteration of the search, the population is truncated to a parent pool of the best ρ individuals from which λ offspring are generated. Finally, an updated population is selected from the best offspring and parents such that the search only moves in the direction of the offspring if they outperform the parents. Figure 3 depicts the ($\mu/\rho+\lambda$) strategy in the context of the overall Autofead search process.

2.7. Breeding

The breeding process generates new features and individuals from the best individuals in the current population. Parents are selected by tournament selection from the parent pool. Then, a genetic operator is randomly chosen based on pre-determined operator probabilities. Autofead includes five operators. Crossover is the most common operator where a segment of code from one feature in the second parent replaces a code segment in one feature of the first parent. The reproduce operator clones a single parent. Mutate modifies a single feature in one parent by inserting, removing, or swapping one function. Lastly, the add feature and remove feature operators control the number of features in an individual. Remove feature eliminates the worst feature in the individual based on use of each feature in individual as a single feature solution. Similarly, add feature determines which feature from the second parent is most beneficial to transfer to the first parent.

2.8. Experimental configuration

The Koza tableau in Table 2 summarizes the run parameters and Autofead configuration used in this study. The configuration is identical for all problems other than the selection of appropriate classification and regression algorithms and fitness measures. Current run parameters are derived from common practice with other GP systems and the authors' engineering judgment. Future parametric studies will be valuable for optimizing population size, genetic operator probabilities, individual size limits, and termination conditions. For each experiment, a training database of fitness cases is used for the Autofead runs and a separate test set is used for reporting final fitness values. The authors presented early results for these experiments using a previous version of Autofead [12] which lacked critical components presented in the previous sections; merging functions, the weighted parameter optimization scheme, cross-validation for fitness estimates, and the current evolution strategy. These additions and changes resulted in consistently higher-fitness solutions for each of the three experiments, and they represent a substantial improvement to the Autofead architecture.

Table 2. Koza tableau of Autofead parameters

Parameter	Setting
Objective	Design optimal feature set from numeric sequence data
Solution structure	Set of features each composed of a sequence of functions which together compute a feature vector
Function library	46 functions from Table 1, convolve only used for CBM problem (with actuation signal)
Pattern recognition	Kernel-based Naïve-Bayes for classification, linear regression for regression
Fitness	Classification accuracy for classification, root mean square (RMS) error for regression
Population initialization	Ramped to initial maximum size of 3 features and up to 5 functions and 3 parameters per feature
Population size	1,000 individuals, no parent pool truncation, 50 offspring per search iteration
Selection	Tournament (tournament size 2)
Genetic operators	Crossover, 80%; mutate, 5%; reproduce, 5%; add feature, 5%; remove feature, 5%
Individual size limits	5 features each limited to 15 functions and 8 parameters
Termination	50,000 individuals

3. Ultrasonic damage detection

For the first experiment, the goal is to determine the presence of damage simulated by a magnet on a steel beam using a single pitch-catch measurement from a pair of piezoelectric elements. The experimental structure is shown in Figure 4. The training database contains 500 undamaged measurements with no magnet and 500 damaged measurements with the magnet located at 40 different locations on the same side of the beam as the actuator and sensor. The undamaged measurements are averaged to provide an additional baseline input sequence. By including a baseline as an additional input channel, it is possible for Autofead to construct features including a baseline subtraction component. Recorded waveforms include 750 samples

collected at 2 MHz. The actuation signal was a 200 kHz Gaussian-modulated sine wave with 0.6 normalized bandwidth.

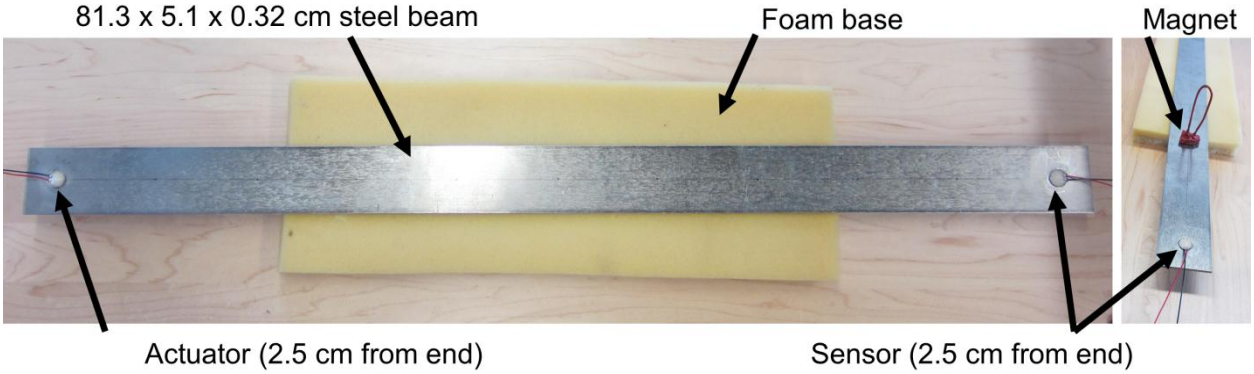

Figure 4. Ultrasonic damage detection experiment structure and sensing system.

3.1. Conventional solution

Conventional pre-processing operations for this task include matched filtering, envelope analysis, and baseline subtraction. The *convolve* function is configured here to perform convolution with the actuation signal for matched filtering. An envelope analysis can be carried out by the *Hilbert magnitude* function. Finally, because the averaged baseline is included as a second input channel, *element difference* performs baseline subtraction. The use of the actuation signal for convolution and inclusion of an average baseline channel represent examples of how domain knowledge can be incorporated into Autofead to improve the search. For comparison solutions, waveforms are pre-processed by matched filtering and envelope analysis; then total energy and peak amplitude features with and without baseline subtraction are computed. Interested readers are directed to Farrar and Worden [3], Raghavan and Cesnik [13], Flynn, et al. [14], and Hu, et al. [15] for an introduction to the field of guided-wave SHM.

3.2. Autofead solution

The Autofead solution contains a single feature based on a cross-correlation between the measured waveform and average baseline. Figure 5 characterizes the original input data and how the data progresses through each processing step in the Autofead feature. In each image, the black and white lines show the median and quartiles, respectively, of sequences from all classes at each sample. These lines show the overall waveform shapes at the current processing stage. In the case of small variance at a sample relative to the entire data

range, the white quartile lines may be hidden under the black median line as in the input signals image. The background reveals how the individual classes are distributed over the value range at each sample. The far right column of each image shows the sums of the sequences at the current processing step. Lastly, vertical and horizontal yellow lines indicate threshold values used by functions in the following step. For example, the horizontal line in the bottom-left image after step 2 indicates the thresholds used by the *control chart* function in step 3.

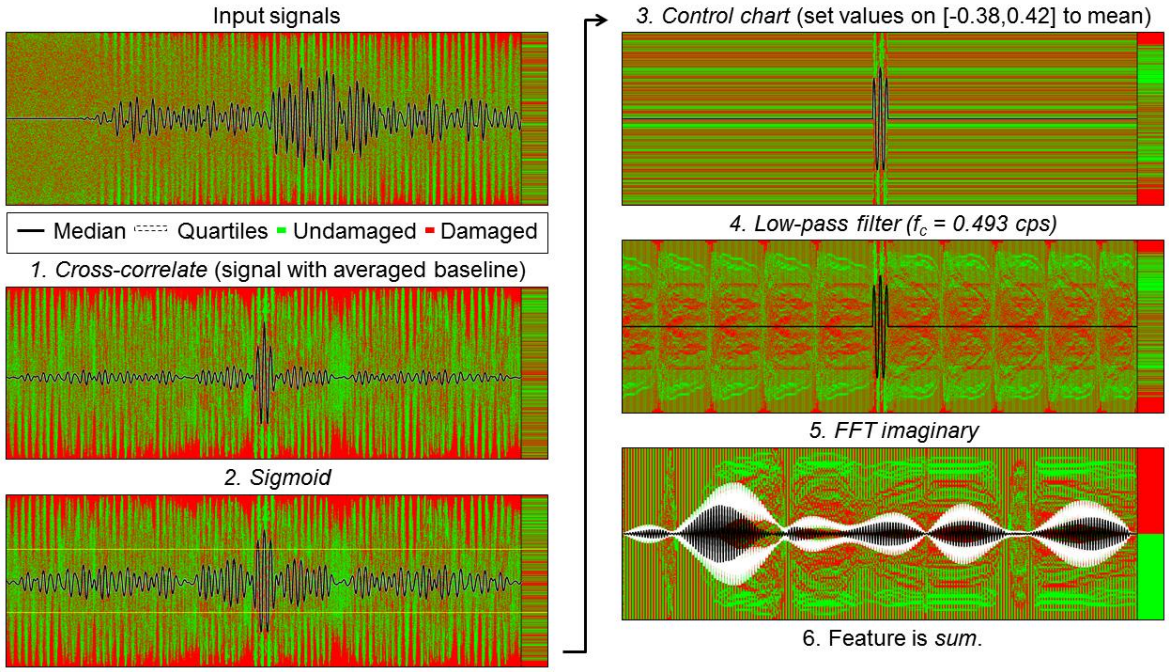

Figure 5. Autofead feature signal processing flow for ultrasonic damage detection experiment.

These visualizations allow for easier understanding and interpretation of feature behaviors and in some cases indicate opportunities for further improvement. In the input signals image, the homogeneous background prior to first arrival indicates identical class distributions. The larger magnitude arrivals show bands where a single class contains most of the extreme values. After cross-correlation, the extreme values belong primarily to the damaged class, and the sum in the last image shows near perfect class separation.

3.3. Feature distributions

Figure 6 shows class-conditioned histograms and probability density function (PDF) estimates for the Autofead feature and four conventional features. From the distributions, the Autofead feature clearly provides greater separation of the undamaged and damaged classes. The unusual distributions for the damaged class

are primarily due to the inclusion of 40 different magnet locations with varying levels of detection difficulty. For example, one would expect peak amplitude without baseline subtraction to either be smaller, larger, or identical with damage to the undamaged class if only a single damage location was included.

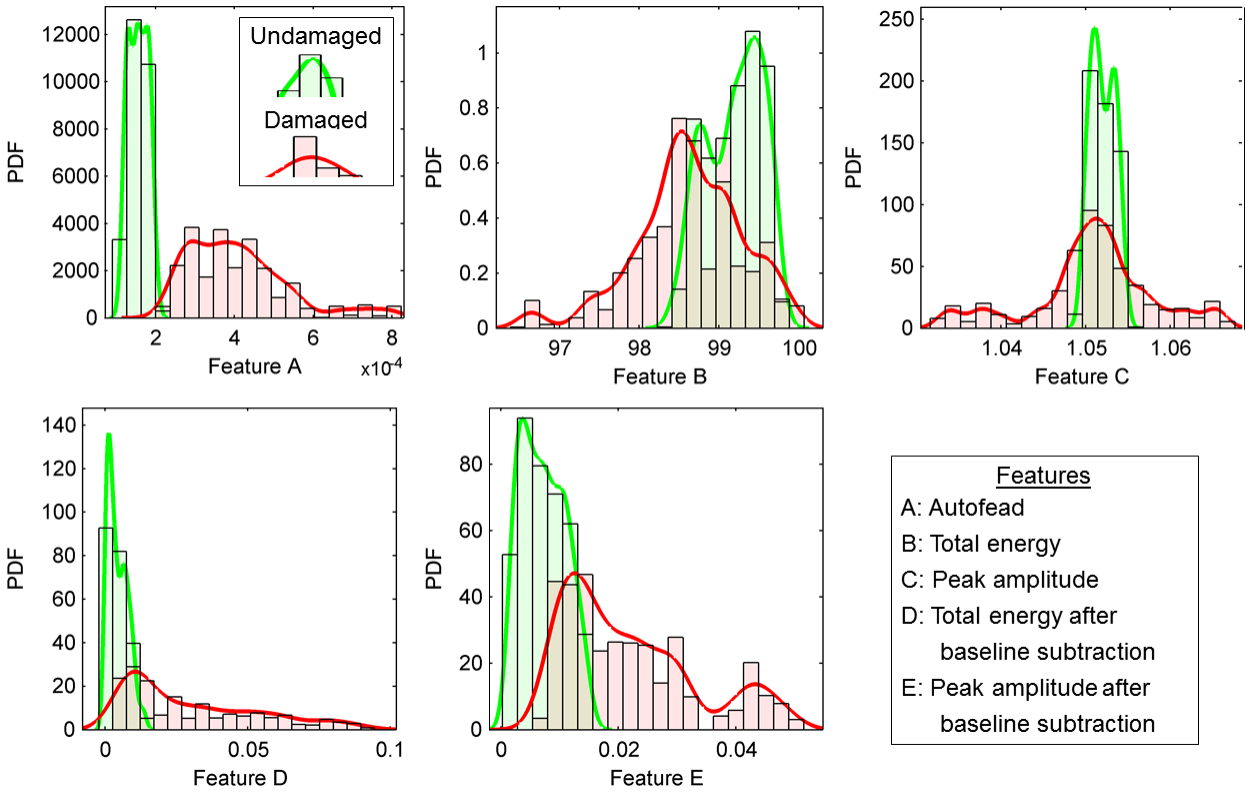

Figure 6. Class-conditioned feature probability density estimates for ultrasonic damage detection experiment.

3.4. Solution fitness

The receiver operating characteristics (ROC) curves in Figure 7 show the trade-off between false positives and correct detections using each feature from Figure 6. The ROC curves confirm that the Autofead feature outperforms the conventional solutions with nearly perfect classification accuracy at 99.9%. However, it is important to note that the Autofead solution is designed to be specific to the training database, which only contains a single beam, single sensor pair, and single damage type in a controlled laboratory environment in this case. The comparison solutions may generalize better to variation of the conditions of the experiment as well as other ultrasonic structural interrogation applications. To design a general feature for these applications, any expected structural, operational, and environmental variability would need to be included in Autofead's training database.

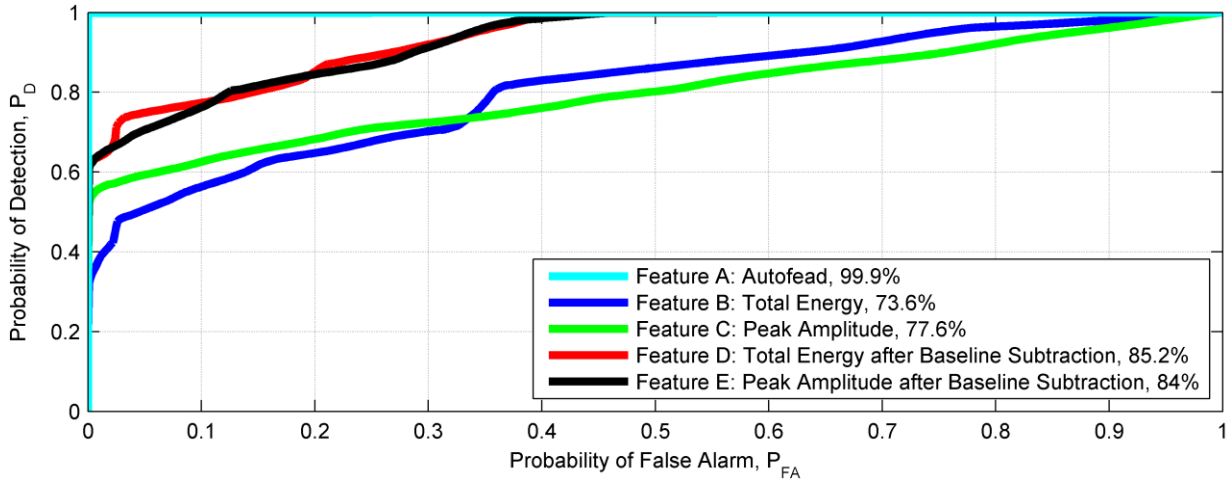

Figure 7. ROC curves and fitness (classification accuracy) for five ultrasonic damage detection features.

4. Damage type identification for rotating machinery

The second experiment involves identifying three bearing health states in the Machinery Fault Simulator from Spectraquest, Inc. depicted in Figure 8. The system consists of a motor and driveshaft supported by three bearings connected to a belt-driven gearbox. Damage is introduced by replacing a healthy bearing with an artificially damaged unit. The damaged units include a bearing with an outer race defect and a bearing with ball spalling damage. The goal is to identify which of the three bearings is installed in the bearing housing nearest the gearbox drive belts.

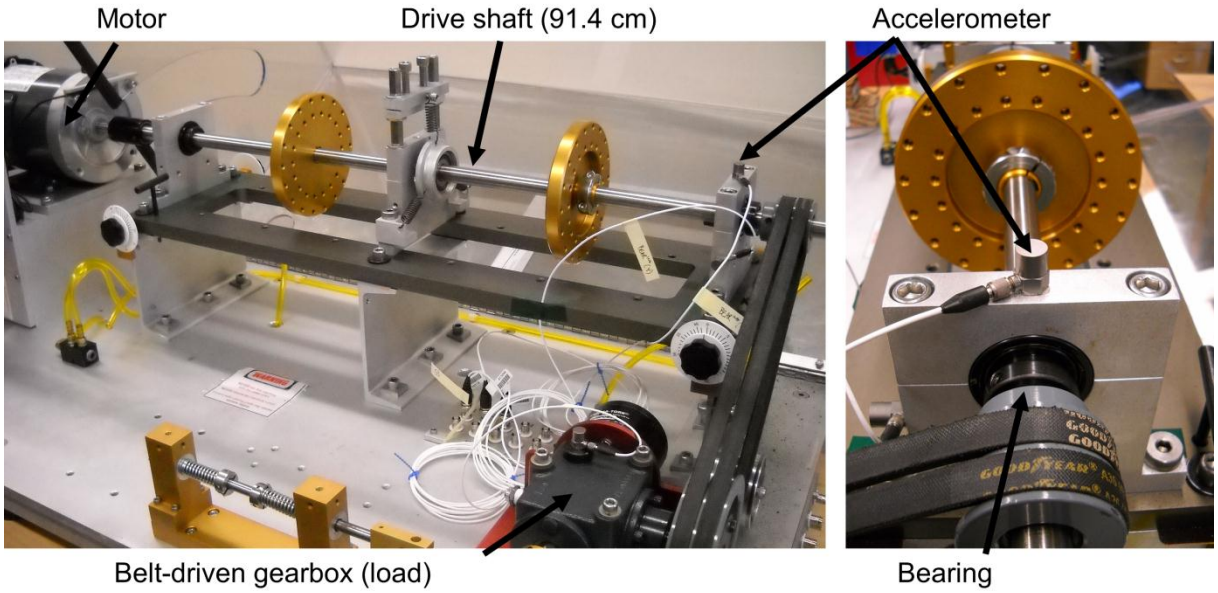

Figure 8. Rotating machinery experiment structure and sensing system.

In this experiment, the motor is driven at a single nominal shaft speed of 1,000 rpm (16.7 Hz). 1,280 sample measurements are recorded form an accelerometer on top of the bearing housing at 2.56 KHz. The training database includes 1,280 fitness cases for each of the three bearings. The system is disassembled and rebuilt to change bearings 24 times to avoid experimental errors introduced by any inconsistency in assembly.

4.1. Conventional solution

Conventional solutions for comparison are selected from ten metrics presented in Lebold's review of vibration analysis methods for rotating machinery [16]. Calculation of many of the metrics requires knowledge of the internal geometry of components such as bearings and gearboxes to identify fundamental frequencies for various filtering operations. Of the ten metrics, FM4 and RMS provide the best performing pair of features for use with a kernel-based Naïve-Bayes classifier.

4.2. Autofead solution

Figure 9 shows the signal processing flow for the two features in the Autofead solution. The first feature is approximately the median value after a non-linear scaling operation from the first two steps, then high-pass filtering, and lastly an envelope analysis. This feature is difficult to interpret due to the untraditional pre-processing sequence. Feature 2 is simply the energy within a narrow frequency band after the *sigmoid* operation, which primarily reduces the effect of outliers. Both features contain parameters that are optimized to remove lower-frequency components of the response.

Figure 10 shows the parameter spaces for the two parameters in feature 1 and three parameters in feature 2. In each image, fitness levels are computed with all other parameters in the individual held to their final optimized values indicated by the black squares. These surfaces show some of the complexity and variety of parameter optimization problems encountered within a single Autofead run. The parameter space for feature 1 is fairly smooth and unimodal, while the space for the integer parameters in feature 2 is dominated by a large low-fitness region. The small region of higher fitness is fairly rough, containing many local extrema. Finally, the surface for the minimum value parameter in feature 2 includes a significant discontinuity around 0.75. In each case, Autofead's parameter optimization scheme provided results near the global optimum.

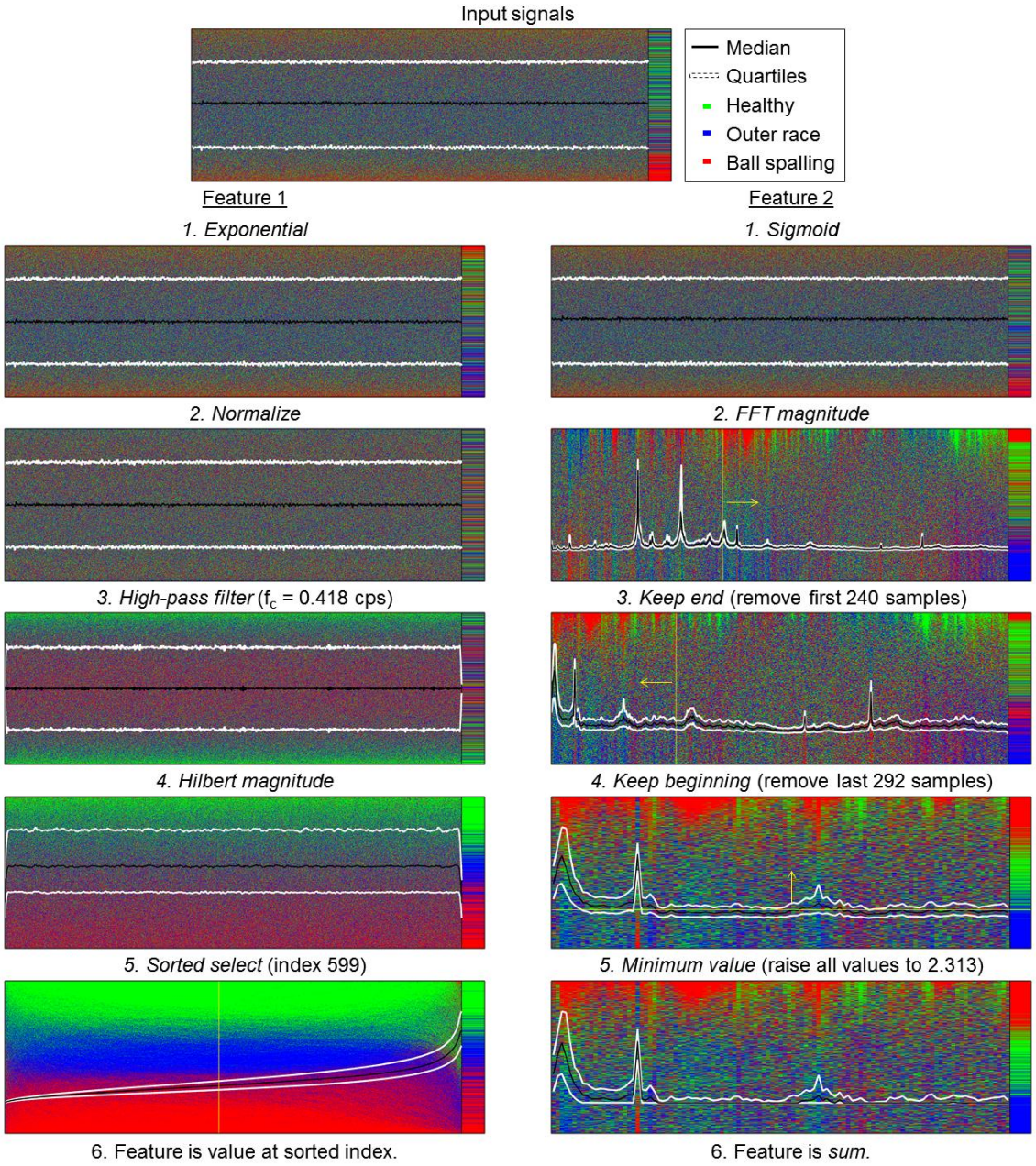

Figure 9. Signal processing flow for two Autofead features in rotating machinery experiment solution.

4.3. Feature spaces and fitness comparison

The two-dimensional feature spaces for the Autofead solution and conventional metrics are given in Figure 11 including decision boundaries from the Naïve-Bayes classifier. The conventional solution relies primarily on the RMS metric to separate the classes and achieve classification accuracy of 78%. In comparison, the two features in the Autofead solution complement each other to provide excellent separation between all three classes resulting in 99% classification accuracy.

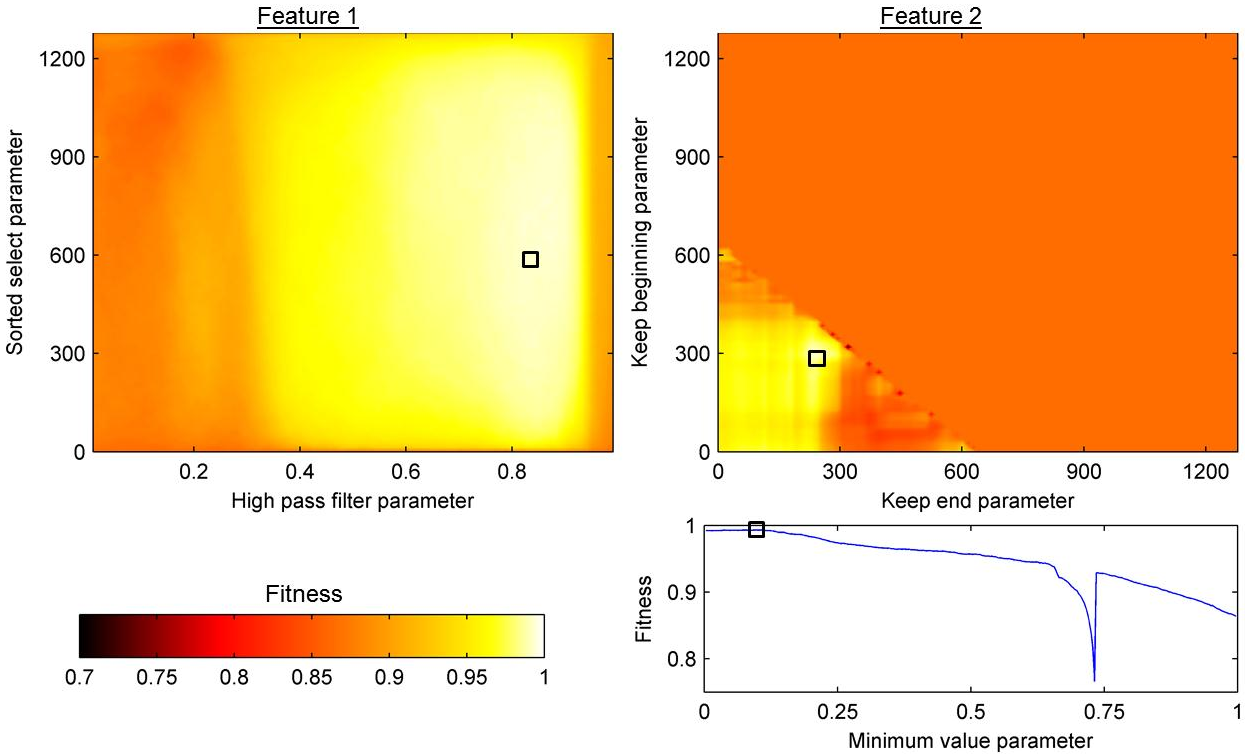

Figure 10. Parameter surfaces for Autofead solution features to rotating machinery experiment.

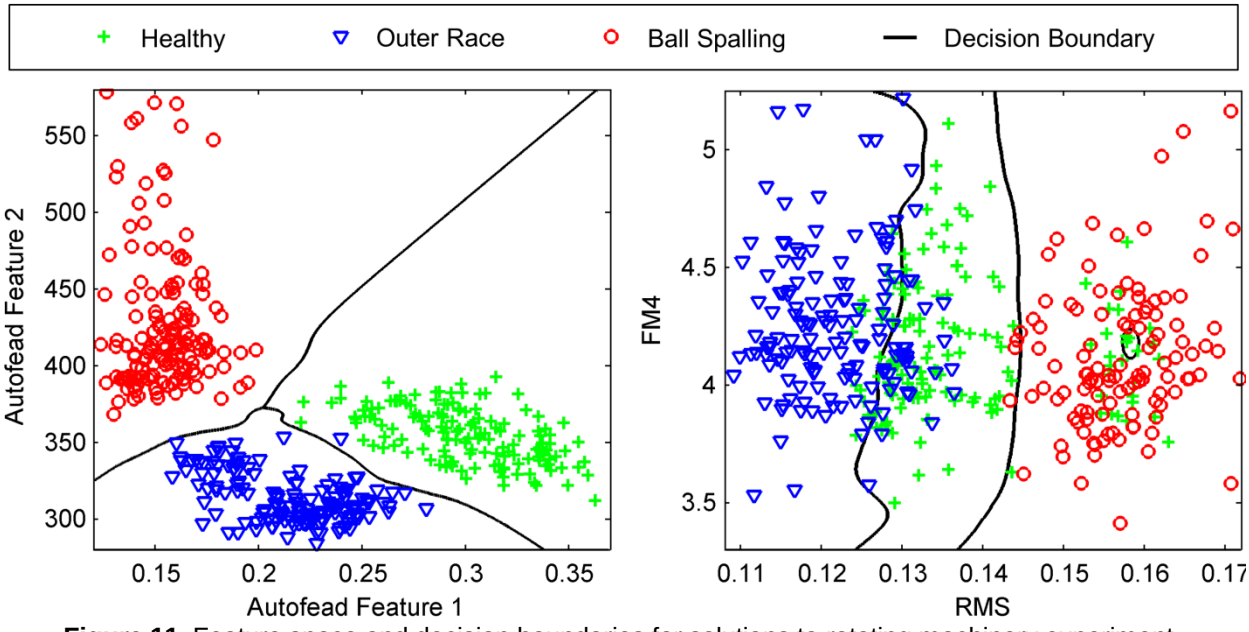

Figure 11. Feature space and decision boundaries for solutions to rotating machinery experiment.

5. Vibration-based damage extent estimation

The final experiment uses the bolted, aluminum bookshelf structure in Figure 12 with the goal of estimating damage extent from vibration-based response measurements. The structure is composed of four 2.5 cm thick aluminum plates measuring 30.5 cm wide by 30.5 cm deep. The plates are supported by rectangular columns at each corner for a total structure height of 53.1 cm. The entire structure is mounted on a rail system to constrain the motion to a single primary direction. An electrodynamic shaker provides excitation along the midline of the bottom floor though a stinger and load cell. Responses are measured by an accelerometer mounted to the midline of each floor in the primary direction of motion.

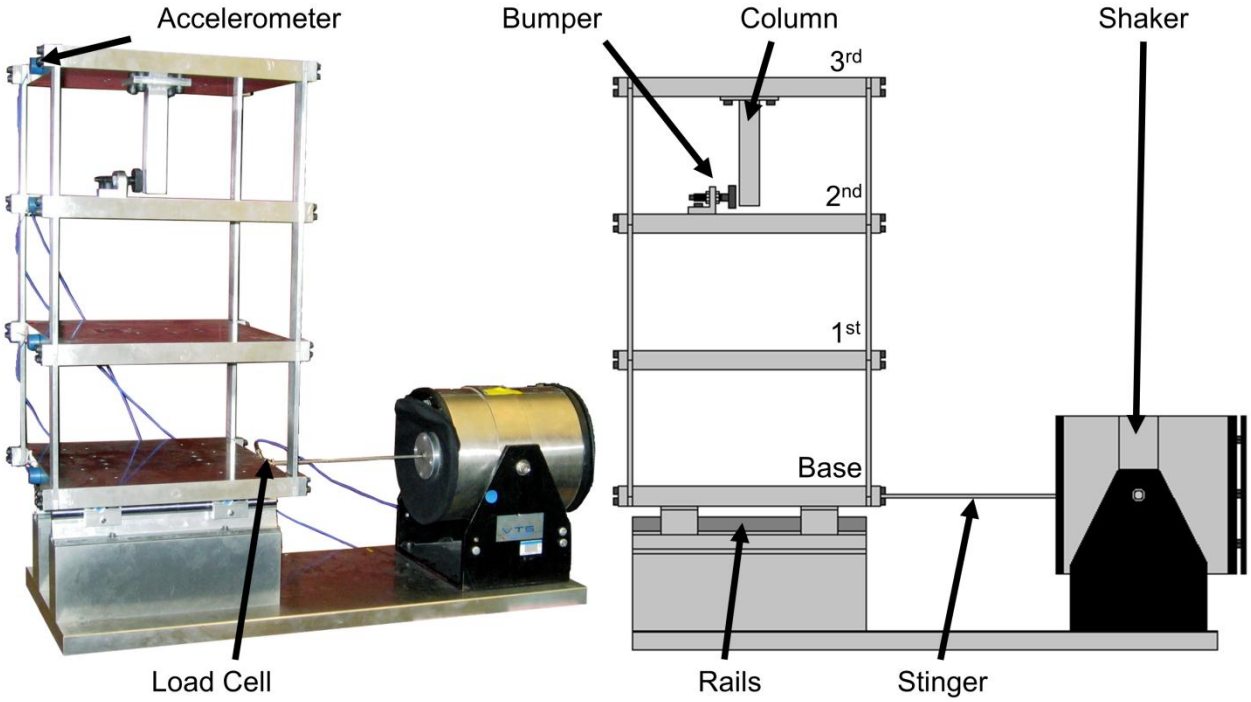

Figure 12. Bookshelf experiment structure and sensing system.

Damage is introduced by changing the width of the gap in a bumper and column system mounted on the second and third floors, respectively. Relative motion of the two floors causes impacts between the bumper and column. When the width of the gap is small, more impacts occur representing a higher level of damage. Six damage levels are included in the training database with the task of estimating the damage level using linear regression from the four response measurements and the input measurement from the load cell.

Damage level 0 represents a wide enough gap such that no impacts occur. Damage levels 1, 1.33, 1.54, 2, and 4 are proportional to their respective gap widths of 0.20, 0.15, 0.13, 0.10, and 0.05 mm.

The structure was excited by band-limited white noise from 20-150 Hz to include the first three natural frequencies. Response measurements were collected at 320 Hz for 3.2 seconds (1,024 samples); 200 fitness cases are included in the training database from each of the six damage levels. Full details of the structure, data acquisition system, and test plan are found in Figueiredo, et al. [17].

5.1. Conventional solution

A multitude of possible solutions exist in the literature for vibration-based damage detection and localization such as those in Farrar and Worden [3] and Hu, et al. [18]. For this example, conventional solutions were selected from the methods compared in Figueiredo, et al. [17]. The statistical moments solution includes the mean, variance, skewness, and kurtosis for each of the four response channels for a total of 16 features. Skewness is a particularly useful feature for this problem as the simulated damage case is highly asymmetric. The natural frequencies solution includes estimates of the first three natural frequencies of the structure estimated from the complex mode indicator function. Two solutions are based on autoregressive (AR) time series modeling. First, AR(5) parameters uses the parameters from a fifth-order model as features. A separate model is fit for each response channel to generate 20 features. Lastly, AR(20) RMS error includes the root mean square error level for each response channel using a 20^{th} order model fit to responses from damage level 0. The RMS error is expected to increase as the damage level increases.

5.2. Autofead solution

The Autofead solution uses three features as depicted in Figure 13. Features 1 and 2 use the response from floor 2 where the bumper is mounted. Feature 3 uses a cross-correlation between the top two floors. While these features are difficult to interpret due to highly non-linear behaviors of some of the element operations, it is clear that higher damage levels contain more high-frequency content.

Investigation of the three-dimensional feature space formed by the Autofead solution reveals the complementary nature of the three features. Figure 14 depicts a three-dimensional scatter plot with PDF estimates along each feature axis conditioned to the six damage levels. Features 1 and 3 both increase

monotonically with the damage level but cannot separate damage levels 0 and 1; however, feature 2 provides the additional information needed to separate the lowest two damage levels.

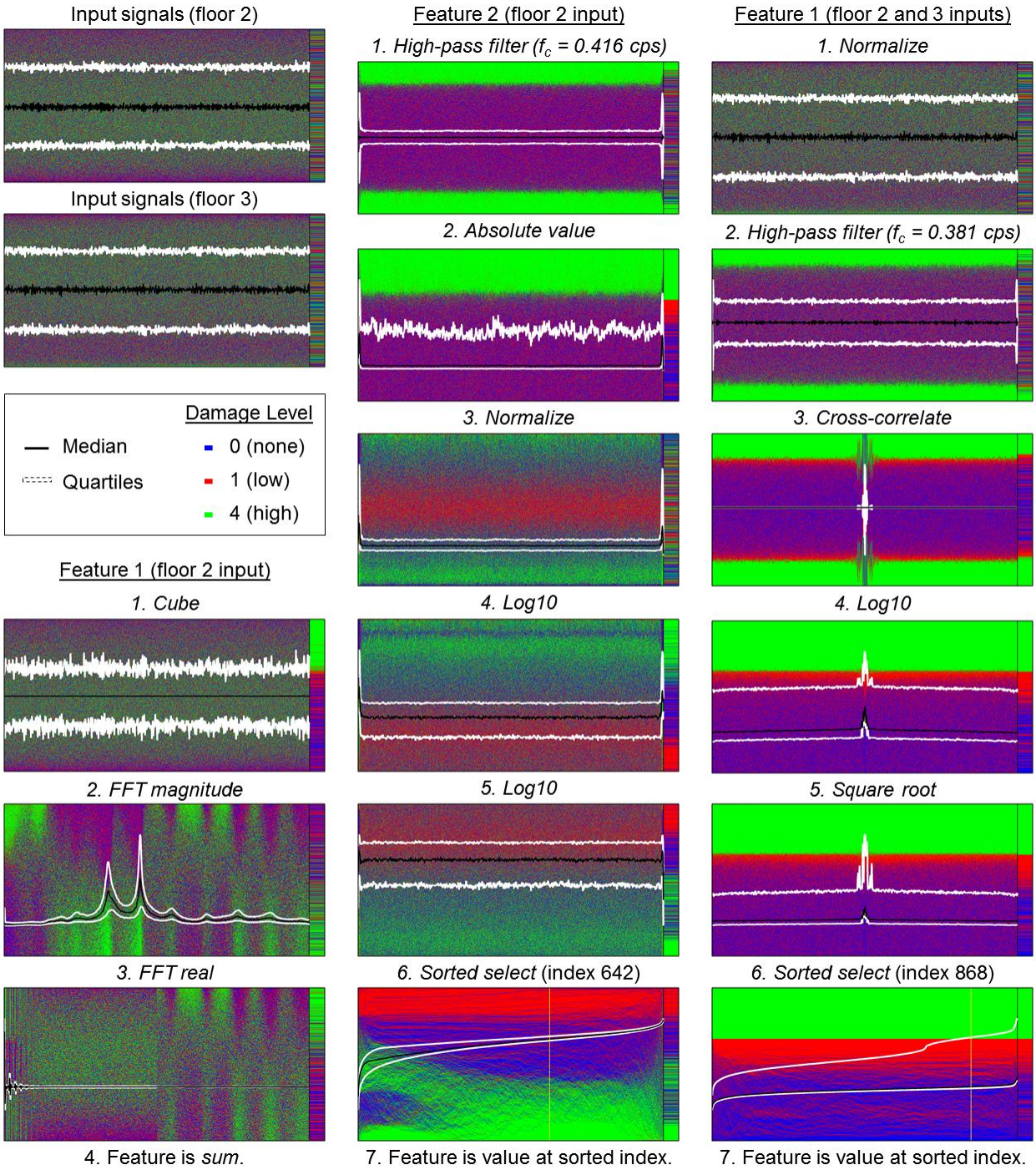

Figure 13. Signal processing flow for three Autofead features in bookshelf experiment solution.

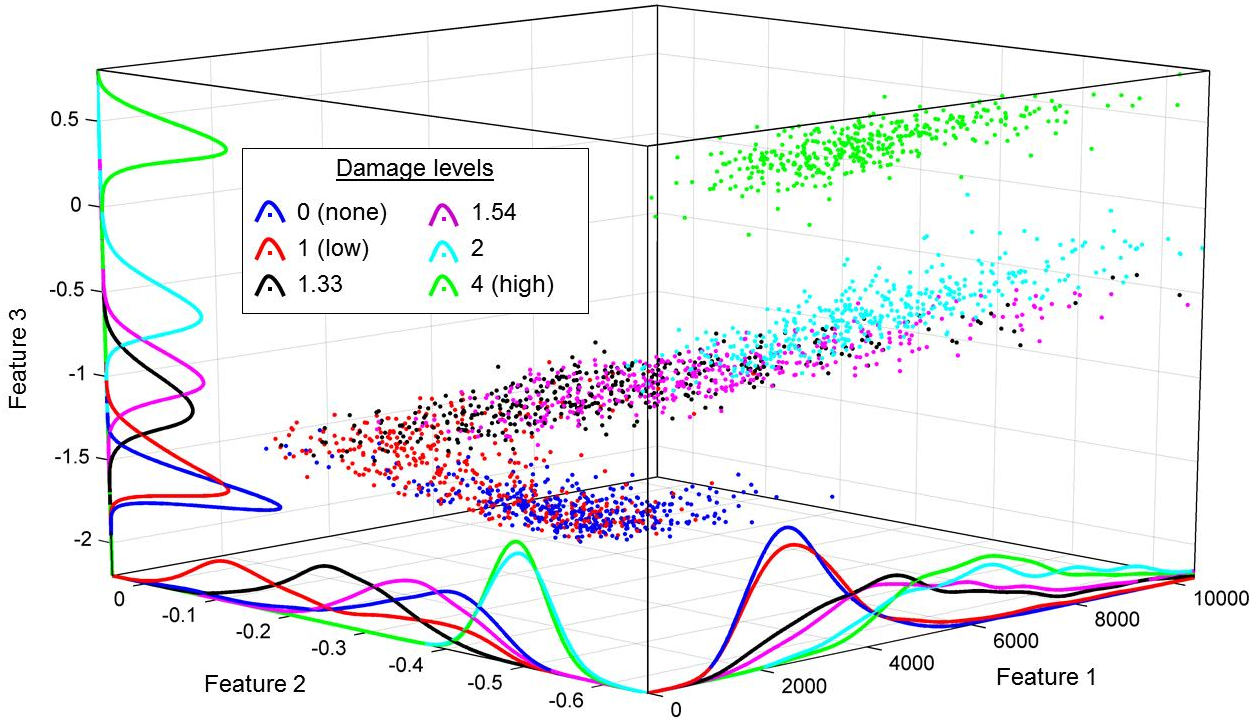

Figure 14. Feature space for Autofead solution to bookshelf experiment.

5.3. Damage extent estimates

To evaluate the relative performance of the Autofead solution and four comparison solutions, distributions of the damage level estimates from each solution are shown in Figure 15 along with the true damage levels shown by vertical, dashed lines. Clearly, the Autofead solution and AR(5) parameters provide the most accurate and consistent estimates across the damage levels. Damage level 1 is an interesting case where bimodal behavior is observed for some of the solutions with one mode overlapping damage level 0. This result is most likely due to a mislabeling of measurements in the training database in cases where the gap width was set correctly to 0.20 mm but no actual impacts occurred within the measurement. However, without an independent measure of impact counts, this hypothesis could not be confirmed. If true, the Autofead solution significantly outperforms the other solutions in separating the measurements from damage level 1 where impacts do and do not occur.

Table 3 provides the fitness levels (RMS error) for each of the five solutions at each individual damage level. The two best solutions at each damage level are shown in bold. The results for damage level 1 do not follow the trends of the other levels due to the previously discussed bimodal behavior issue. For all other

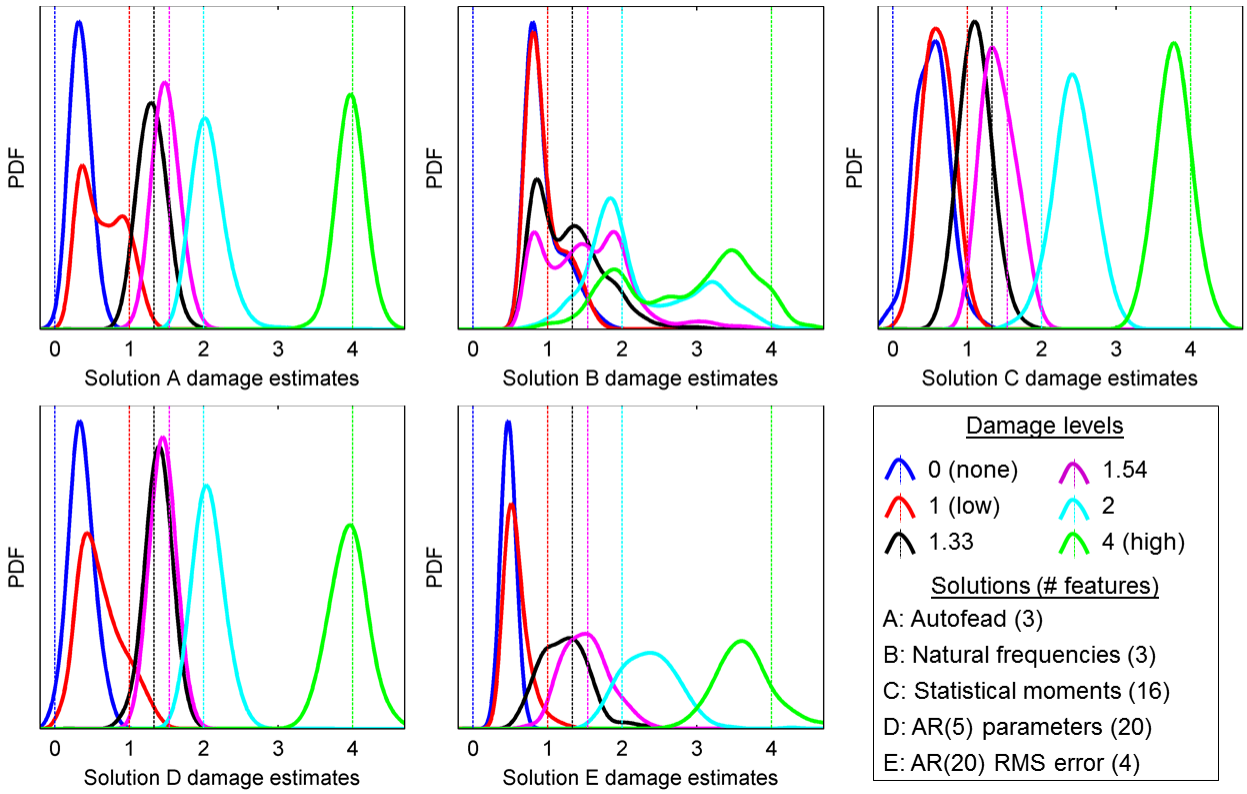

Figure 15. Damage level estimate distributions for five solution to bookshelf experiment.

levels, as well as on average, the Autofead solution provides the lowest or second-lowest error level. While the error is similar for the next best solution of AR(5) parameters, the dimension reduction provided by Autofead is significantly better using only 3 features compared to 20.

Table 3. Damage extent estimate RMS errors

Damage level	Autofead	Natural frequencies	Statistical moments	AR(5) parameters	AR(20) RMS error
0	**0.36**	0.98	0.56	**0.39**	0.47
1	0.45	**0.24**	**0.44**	0.48	0.45
1.33	**0.16**	0.44	0.30	**0.16**	0.34
1.54	**0.15**	0.58	0.24	**0.16**	0.30
2	**0.21**	0.78	0.50	**0.19**	0.53
4	**0.14**	1.41	0.30	**0.23**	0.52
Average	**0.25**	0.74	0.39	**0.27**	0.44

6. Conclusions

This work presents and experimentally validates the Autofead approach to automated feature extraction algorithm design, particularly suited for cases where domain knowledge is minimal. Significant performance improvements over conventional feature extraction methods are presented for damage detection, damage

classification, and damage extent estimation experiments. Autofead can develop complete measurement to feature space algorithms for signal processing and feature extraction or serve as useful tool for feature designers in SHM and other fields to provide significant insight into databases of numeric sequence measurements. With Autofead, algorithms can be inferred from data alone or in conjunction with domain knowledge provided through transformation of the input space, augmentation of the function library, selection of appropriate pattern recognition algorithms, or fitness function customization. Future development of the method will evaluate and improve on the generalization capability of evolved solutions when the available training measurements are limited to a subset of possible conditions the system would operate under in actual usage. Additionally, the choice of classification or regression algorithm to use may be evolved within the Autofead search process as no single algorithm is suitable for all problems.

Acknowledgements

This work was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. The authors would like to thank Luke Robinson for his contributions to the rotating machinery experiment.

References

[1] Doebling S W, Farrar C R and Prime M B 1998 A summary review of vibration-based damage identification methods *Shock and Vibration Digest* **30** 91–105

[2] Farrar C R, Doebling S W and Nix D A 2001 Vibration–based structural damage identification *Philos. T. Roy. Soc. A* **359** 131–49

[3] Farrar C R and Worden K 2012 *Structural Health Monitoring: A Machine Learning Perspective* (Wiley)

[4] Worden K, Farrar C R, Manson G and Park G 2007 The fundamental axioms of structural health monitoring *Proc. Roy. Soc. A-Math. Phy.* **463** 1639–64

[5] Harvey D Y and Todd M D 2013 Automated extraction of damage features through genetic programming *Proc. SPIE 86950J*

[6] Koza J R 1992 *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (A Bradford Book)

[7] Poli R, Langdon W B and McPhee N F 2008 *A field guide to genetic programming* (Lulu Enterprises Uk Limited)

[8] Teller A and Veloso M 1995 Program evolution for data mining *Int. J. Expert Syst. Res. Appl.* **8** 213–36

[9] Eads D R, Williams S J, Theiler J, Porter R, Harvey N R, Perkins S J, Brumby S P and David N A 2004 A multimodal approach to feature extraction for image and signal learning problems *Optical Science and Technology, SPIE's 48th Annual Meeting (San Diego, CA)* 79–90

[10] Holladay K L and Robbins K A 2007 Evolution of Signal Processing Algorithms using Vector Based Genetic Programming *15th International Conference on Digital Signal Processing (Cardiff, UK* 503–6

[11] Back T 1996 *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms* (Oxford University Press, USA)

[12] Harvey D Y and Todd M D 2013 Automated Near-Optimal Feature Extraction Using Genetic Programming with Application to Structural Health Monitoring Problems *Proc. 9th International Workshop on Structural Health Monitoring (Palo Alto, CA)*

[13] Raghavan A and Cesnik C E 2007 Review of guided-wave structural health monitoring *Shock and Vibration Digest* **39** 91–116

[14] Flynn E B, Todd M D, Wilcox P D, Drinkwater B W and Croxford A J 2011 Maximum-likelihood estimation of damage location in guided-wave structural health monitoring *Proc. R. Soc. A* **467** 2575–96

[15] Hu N, Shimomukai T, Fukunaga H and Su Z 2008 Damage identification of metallic structures using A0 mode of Lamb waves *Structural Health Monitoring* 7 271–85

[16] Lebold M, McClintic K, Campbell R, Byington C and Maynard K 2000 Review of vibration analysis methods for gearbox diagnostics and prognostics *Proc. 54th Meeting of the Society for Machinery Failure Prevention Technology (Virginia Beach, VA)* 623–34

[17] Figueiredo E, Park G, Figueiras J, Farrar C and Worden K 2009 *Structural health monitoring algorithm comparisons using standard data sets* Los Alamos National Laboratory, LA-14393

[18] Hu N, Wang X, Fukunaga H, Yao Z H, Zhang H X and Wu Z S 2001 Damage assessment of structures using modal test data *Int. J. Solids Struct.* **38** 3111–26