An Investigation into Classification of High Dimensional Frequency Data

John M. McGraw

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master's of Science
in
Statistics

Eric P. Smith, Chair
John A. Burns
William H. Woodall

23 October 2001
Blacksburg, Virginia

Keywords: Confidence Interval, Correlation, Data Analysis, Frequency Response, Maximum Likelihood, Multivariate Normal

An Investigation into Classification of High Dimensional Frequency Data

John M. McGraw

(ABSTRACT)

We desire an algorithm to classify a physical object in "real-time" using an easily portable probing device. The probe excites a given object at frequencies from 100 MHz up to 800 MHz at intervals of 0.5 MHz. Thus the data used for classification is the 1400-component vector of these frequency responses.

The Interdisciplinary Center for Applied Mathematics (ICAM) was asked to help develop an algorithm and executable computer code for the probing device to use in its classification analysis. Due to these and other requirements, all work had to be done in Matlab. Hence a significant portion of the effort was spent in writing and testing applicable Matlab code which incorporated the various statistical techniques implemented.

We offer three approaches to classification: maximum log-likelihood estimates, correlation coefficients, and confidence bands. Related work included considering ways to recover and exploit certain symmetry characteristics of the objects (using the response data). Present investigations are not entirely conclusive, but the correlation coefficient classifier seems to produce reasonable and consistent results. All three methods currently require the evaluation of the full 1400-component vector. It has been suggested that unknown portions of the vectors may include extraneous and misleading information, or information common to all classes. Identifying and removing the respective components may be beneficial to classification regardless of method. Another advantage of dimension reduction should be a strengthening of mean and covariance estimates.

Acknowledgments

This work is an outgrowth of a project administered by the Interdisciplinary Center for Applied Mathematics (ICAM), Virginia Tech, Blacksburg, VA. As such, much of it represents a collaboration between individuals from government, private industry, and academia. Those portions of what follows involving overall data representation, the MLLE method, and the symmetry metric are largely a minor repackaging of work done by Drs. B. G. Fitzpatrick (Tempest Technologies), J. A. Burns (ICAM), E. M. Cliff (ICAM), as well as Mr. C. K. Frick (ICAM). The correlation approach evolved from an exploration begun by the company who designed the probe device. The confidence band method stemmed from a purely geometric consideration suggested by confidence bands used in statistical linear regression.

My thanks to my advisor, Dr. E. P. Smith, for his willingness to direct me under uncommon circumstances. Although not directly involved in the project, he provided insightful suggestions and guidance. I am indebted to Dr. W. H. Woodall, who introduced me to the book *Discrimination and Classification* by D. J. Hand. He further helped me with some key concepts and provided some much appreciated proofreading.

My appreciation to those at ICAM for the opportunity to work with them on this project. They also provided support and understanding in full measure – especially Drs. J. A. Burns, E. M. Cliff, and B. B. King.

A special thanks goes to Dr. J. A. C. Weideman, University of Stellenbosch, South Africa. He provided an opportunity to present preliminary work on the confidence band approach at the 25th Annual South African Symposium on Numerical and Applied Mathematics (SANUM). As one always hopes, this was also a chance to reap insights from the many fertile minds there. However my gratitude to Dr. Weideman extends to an earlier time, when he first made me aware of several important differences between theoretical results and computational reality – especially that one never computes the inverse of a matrix, one solves the associated system instead.

This work was supported in part by the Air Force Office of Scientific Research under Grant F49620-96-1-0329.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work is part of a project having the goal of identifying the class of a physical object out in the field in "real-time". Each object under consideration belongs to one of 22 classifications/categories. A portable probing device has been developed for this classification procedure. The Interdisciplinary Center for Applied Mathematics (ICAM) was consulted to help develop an algorithm and executable computer code for the probing device to use in its classification analysis.

The probe excites a given object at frequencies from 100 MHz up to 800 MHz at intervals of 0.5 MHz. Thus the data used for classification is the 1400-component vector of these frequency responses. An example of such a response vector is given in Figure B.1 on page 50. Based on this response, we wish to classify the object as to its category. One may think of this as trying to identify the location of a hole drilled in a plate. Certain excitation frequencies will have modes that intersect with the hole. This will disrupt those frequencies in a measurable way. Other frequencies may not intersect the hole, and their responses should remain unchanged[1].

Work progressed in two stages.

1. Initially, 10 sample response vectors were provided from each category (220 total) to construct classifiers, along with 9 additional vectors for testing. At this time, two deterministic approaches were investigated as well as a maximum log-likelihood estimator (MLLE) and a confidence band estimator. Initial results showed promise, but it was recognized that many more training samples were necessary to use any statistical methods.

2. A second set of 660 sample response vectors (30 from each category) was provided. This is referred to as the "A-data" and was used as the training set. In addition, the

[1] A potential difficulty may stem from other geometric features unaccounted for. These might interact with the frequencies and produce ambiguous or unreliable responses at certain levels.

probing point on objects for this new set of data was changed to reflect the physical symmetry among certain pairs of classifications. (Each of the original 22 classes had a corresponding symmetry class, leaving 11 symmetry classes of uniquely paired original classes.) Hence results from the earlier work could not be used to test new response vectors (nor vice versa). However the earlier code was adaptable to the new situation. At this point, the company had recognized the difficulty of full classification and was willing to accept a classification into one of the 11 paired symmetry classes, rather than to expect full classfication into one of the 22 original classes. In fact, their position was now that the new probing method should not be able to distinguish between two classes within the same symmetry class. Hence the new major focus was to develop a classifier which only discriminated in terms of the 11 symmetry pairs. Efforts were concentrated in developing the MLLE, a related log-likelihood ratio as a symmetry metric, and a correlation coefficient approach (first suggested by the manufacturing company). The confidence band classifier was also used but largely as originally written.

About the same time, we were given an additional 110 response vectors (5 from each class) to use in evaluating our procedures. This is referred to as the "B-data" and was used as the test set.

Using the second set of data, work was grouped into four focus areas.

1. Provide a basic way to represent the data in Matlab that could be used by all parties in all investigations.

2. Determine procedures that classified a vector into its proper symmetry class.

3. Investigate whether the developed procedures would correctly classify a vector into its "opposite" symmetry class. For example, the A class and S class share certain symmetry characteristics that are supposed to make them indistiguishable to the probing device[2]. Suppose the A class was removed as a classification option. Would an A vector then be properly classified into its S symmetry class? An extension of this was to develop a new symmetry metric used to measure how close the MLLE for a vector's true class was to the log-likelihood value of its associated symmetry class.

4. Develop methods to reduce the number of components within each vector necessary to make a correct (symmetry) classification, and determine which components those were.

Assumptions included:

[2]Recall that the two data *sets* were designated "A" and "B". In addition, the 22 data *classes* themselves were also designated with the letters A, B, C, ..., V.

- The data followed a set of 22 multivariate normal distributions; i.e., we had 22 random vectors $\mathbf{x}_j$ such that

$$\mathbf{x}_j \sim N_p(\boldsymbol{\mu}_j, \Sigma_j), \quad j = 1, 2, \ldots, 22, \quad p = 1400$$

- Because of computational complexities associated with the high dimension of the data, we assumed each vector component (element) was independent of others.
- Differences between classifications/symmetries should be adequately discernable via different mean vectors and/or variance matrices for each of the 22 classes/11 symmetries. (Figure B.2 on page 51, is a plot of the 22 class averages.)
- For the correlation coefficient, vectors from like classes/symmetries should be correlated in some way – either along components or across components.

Generally our results were in accordance with the work of D. J. Hand in his text *Discrimination and Classification* [2]. MLLE approaches tend to work well on the data used to train them, but not necessarily on new test data. This could be a result of Bellman's *curse of dimensionality*, ([2], pp 17) – as the dimension of the multivariate problem increases, a commensurate number of samples is necessary to estimate the desired statistics[3]. With only 30 samples from each class, one can likely gain only rough estimates for the class mean and variance in $\mathbb{R}^{1400}$.

The correlation coefficient approach was only slightly less successful than MLLE on the A-data (training data). However it did quite well on the B-data (test data). Hence it appears to be a robust method and have the most potential for classification accuracy.

Since the confidence band method relies on the same estimates for means and covariances as MLLE, it is susceptible to similar performance difficulties. It did reasonably well on the training data (A-data) but poorly on the test data (B-data).

Little has been done with the symmetry metric at this time. It is designed to use MLLE and give a relative measure of how close an already classified vector is to also being placed within its associated symmetry class. At this point, it has been used to compare which classes, on average, the MLLE best matched to their respective symmetry classes. In the future, it may be a helpful tool in evaluating which window of components gives a closer symmetry match over other windows. Knowing such a window may have two immediate benefits: identifying those components which truly reflect the physical symmetry inherent in the two classes, and reducing the necessary dimension of the data required for classification.

[3]An additional difficulty in higher dimensions is that the resulting probability density function values tend toward zero for all observations, including the mean. Thus possible Bayesian approaches may not be effective when used to estimate the probability of being in a particular class.

Chapter 2

Background and Theory

2.1 Assumptions

In almost any statistical analysis, certain assumptions must be made. Which to make is not always obvious, however a good practice is to try to keep them within the bounds of necessity and what is reasonable for the given setting.

Our first assumption must be that the frequency responses provide adequate information about the objects tested – or if they do not, that the course of our investigations with the responses will enable us to discover any errors. Accepting this, we state what follows in terms of the observations, keeping in mind that they just as readily apply to the objects themselves.

- Within classes, observations are distributed symmetrically about a common mean.
- Within classes, observations tend to be more like the class mean than not.
- The differences between classes are significant enough to be detectable using statistical methods.

Statistically, these translate into properties we look for in determining the associated probability density functions (pdfs).

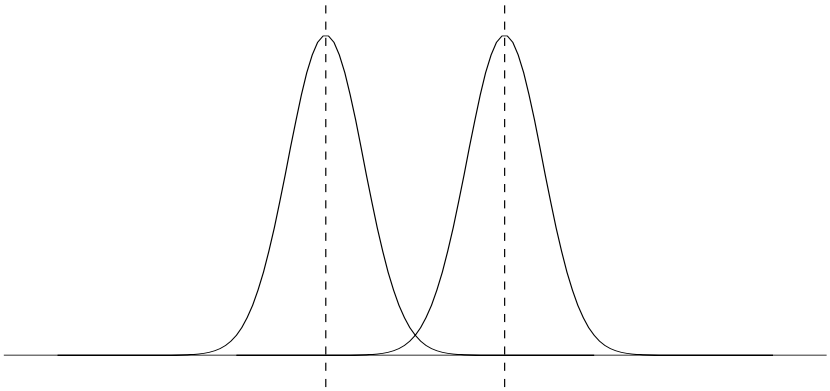
- The pdf for each of the 22 classes is symmetric and unimodal.
- The tails of each pdf drop fairly quickly as distance from the mean increases.
- Given any two classes, their repective means are well separated, and their respective variances are not too large.

Because they satisfy the above properties and carry others that are statistically appealing, we look to multivariate normal (Gaussian) pdfs. As such, we assume the data represents 22 random vectors $\mathbf{x}_j$, such that

$$\mathbf{x}_j \sim N_p(\boldsymbol{\mu}_j, \Sigma_j), \quad j = 1, 2, \ldots, 22, \quad p = 1400$$

As a conceptual aid to what follows, it may be useful to consider the case of a univariate normal random variable. Its geometry can be used to help visualize the above features, and is illustrated in Figure 2.1 below.

Figure 2.1: Two Well-spaced Univariate Normal PDFs



2.2 Maximum Log-Likelihood Estimates

2.2.1 Maximum Likelihood Estimates

The technique of *maximum likelihood estimates* (MLE) is a standard statistical tool to try and match an observed random variable $\mathbf{x}_0$ with its probability density function (pdf). The approach is to evaluate candidate pdfs at the observed $\mathbf{x}_0$, then identify $\mathbf{x}_0$ with that pdf which returns the maximum value. Since larger pdf values indicate a higher likelihood of occurrence, we are also associating the observation with that population in which it is most likely to occur; i.e., we are classifying the observation via its maximum pdf-value[1].

[1]Here we must acknowledge a key assumption – that our given observation is in some sense representative of its population. This may not always be the case. The observation may be very unusual in its true population, but common in another. However, we cannot generally determine which is the case, so we appeal to the idea that we are more apt to observe the more likely occurrences.

We will represent the MLE with the letter L and define it as

$$L(\mathbf{x}_0) \;=\; \max_{\theta\in\Theta}\{f(\mathbf{x}_0,\boldsymbol{\theta})\}$$

where

$$\begin{aligned}
\mathbf{x}_0 &= \text{The sample under consideration.}\\
\boldsymbol{\theta} &= \text{The set of parameters that uniquely define each pdf.}\\
\Theta &= \text{The family of possible pdf parameter sets (which corresponds directly to the family of pdfs).}\\
f(\mathbf{x}_0,\boldsymbol{\theta}) &= \text{The value of the pdf defined by the parameter set } \boldsymbol{\theta} \text{ and evaluated at the sample } \mathbf{x}_0.
\end{aligned}$$

Although f represents a pdf, in this context it is sometimes referred to as the *likelihood function.*

Since we are assuming a multivariate normal pdf with samples from $\mathbb{R}^{1400}$, our likelihood function has the specific form

$$f\left(\mathbf{x}_0,(\boldsymbol{\mu},\Sigma)\right) \;=\; \frac{1}{(2\pi)^{1400/2}\,|\Sigma|^{1/2}}\,\exp\left\{-\frac{1}{2}\,(\mathbf{x}_0-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}_0-\boldsymbol{\mu})\right\}$$

where

$$\begin{aligned}
\boldsymbol{\theta} &= (\boldsymbol{\mu},\Sigma)\\
\boldsymbol{\mu} &= \text{The mean vector of the population}\\
\Sigma &= \text{The covariance matrix of the population}\\
|\Sigma| &= \text{The determinant of } \Sigma
\end{aligned}$$

Let the subscript M denote maximum. Then for each observation $\mathbf{x}_0$, ideally we seek the pair of parameters $(\boldsymbol{\mu}_M,\Sigma_M)$ which satisfy

$$f(\mathbf{x}_0,(\boldsymbol{\mu}_M,\Sigma_M)) \;\geq\; f(\mathbf{x}_0,(\boldsymbol{\mu},\Sigma)) \quad \text{over all allowable parameters } (\boldsymbol{\mu},\Sigma)$$

and classify $\mathbf{x}_0$ as being in the class represented by the pdf with parameters $(\boldsymbol{\mu}_M,\Sigma_M)$. Unfortunately in high dimensional settings, this is where theory and practice collide. In such cases, pdf values may all be numerically evaluated as zero. The reason for this is in the next section and is what prompted the change from likelihood to log-likelihood functions.

2.2.2 Maximum Log-Likelihood Estimates

Since the natural logarithm function is monotone, composition with it preserves most of the desirable features of the normal pdf, including locations of maxima and minima. For

this reason, it is common to consider the maximum value of the log of a pdf instead of the maximum of the pdf itself. Again an appeal to the geometry of the univariate case, given in Figures 2.2 and 2.3, may help to illustrate this.

Figure 2.2: Univariate Normal

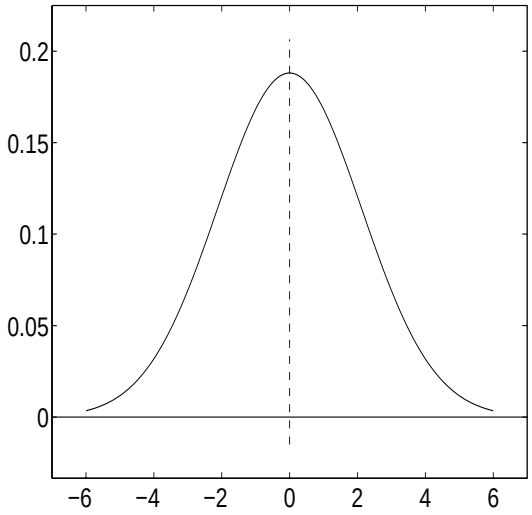
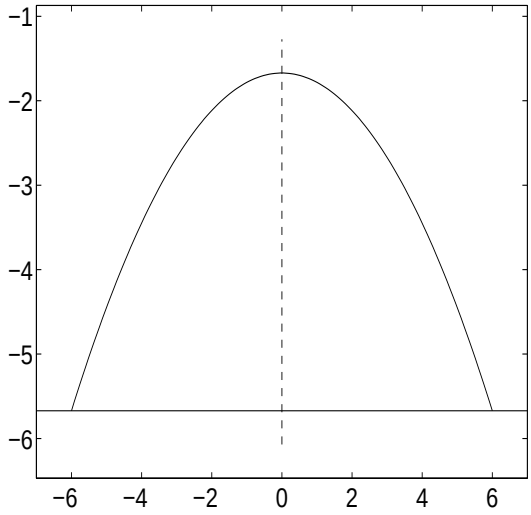


Figure 2.3: Log-Normal



This preservation property is especially useful when the pdf is a multivariate normal of high dimension, as is our particular case. One reason for this is that as dimension increases, pdf height decreases. For samples from $\mathbb{R}^{1400}$, this results in pdf-values satisfying

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{(2\pi)^{1400/2}\,|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}\,(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \qquad (2.1) \\ &\leq \left(\frac{1}{(2\pi)^{1400}\,|\Sigma|}\right)^{1/2} * 1 \end{aligned}$$

It should be clear that the right side of the inequality is virtually zero for any realistic Σ. In fact, most current computers will likely return only a zero value for the pdf calculation. However the natural logarithm of the above pdf has the form

$$\ln(f(\mathbf{x})) = -\frac{1}{2}\left(1400\,\ln(2\pi) + \ln|\Sigma|\right) - \frac{1}{2}\,(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

and is well within the realm of any scientific calculator. So we will use this log-pdf and work with *maximum log-likelihood estimates* (MLLE) denoted by $\mathcal{L}$ and defined via

$$\begin{aligned} \mathcal{L}(\mathbf{x}_0) &= \max_{\theta\in\Theta}\left\{\ln\left(f(\mathbf{x}_0,\boldsymbol{\theta})\right)\right\} \\ &= \max_{(\boldsymbol{\mu},\Sigma)}\left\{-\frac{1}{2}\left(1400\,\ln(2\pi) + \ln|\Sigma|\right) - \frac{1}{2}\,(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \end{aligned}$$

2.2.3 Construction of the MLLE Classifier

Each observation is a 1400-component vector of frequency responses and belongs in one of 22 classes. Thus we have a family of 22 associated multivariate normal densities, each with a unique parameter pair $(\boldsymbol{\mu}_j, \Sigma_j)$, $j = 1, \ldots, 22$.

Since the true mean $\boldsymbol{\mu}_j$ and covariance matrix Σ_j cannot be known, we attempt to estimate them in the standard ways using some sample from the population. The current classifier hase been constructed from the A-data, which contains 30 training vectors for each of the 22 classes (660 total vectors).

The mean for each jth classification is estimated by the *sample average* $\overline{\mathbf{x}}_j$ taken over the $n_j = 30$ training vectors known to be from the jth class.

$$\overline{\mathbf{x}}_j \;=\; \frac{1}{n_j}\sum_{k=1}^{n_j}\mathbf{x}_{jk} \;=\; \frac{1}{30}\sum_{k=1}^{30}\mathbf{x}_{jk}, \quad j = 1, \ldots, 22$$

The appropriate *sample covariance matrix* $\widehat{\Sigma}_j$ should be used to estimate each classification covariance matrix. Its construction is via

$$\begin{aligned}\widehat{\Sigma}_j &= \frac{1}{n_j - 1}\sum_{k=1}^{n_j}(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)^T \\ &= \frac{1}{29}\sum_{k=1}^{30}(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)^T \qquad (2.2)\end{aligned}$$

Unfortunately with vectors in $\mathbb{R}^{1400}$ and only 30 samples, $\widehat{\Sigma}_j$ constructed in this manner is a singular matrix. This is because

$$\text{rank}\left((\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)^T\right) \;\leq\; \text{rank}\left((\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)\right) \;\leq\; 1$$

which leads to

$$\begin{aligned}\text{rank}\left(\widehat{\Sigma}_j\right) &= \text{rank}\left(\sum_{k=1}^{30}(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)^T\right) \\ &\leq \sum_{k=1}^{30}\text{rank}\left((\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j)^T\right) \;\leq\; 30\end{aligned}$$

But $\widehat{\Sigma}_j$ must be 1400×1400, thus $\widehat{\Sigma}_j^{-1}$ cannot exist, and substituting for Σ^{-1} in Equation 2.2 is not possible. To address this, we make the simplifying assumption that all components are independent of each other. We then have a strictly diagonal matrix of component variances

with diagonal given by

$$\begin{aligned}
\operatorname{diag}\left(\widehat{\Sigma}_j\right) &= \begin{bmatrix} \dfrac{1}{n_j-1}\sum_{k=1}^{n_j}(x_{jk(1)}-\overline{x}_{j(1)})^2 \\ \dfrac{1}{n_j-1}\sum_{k=1}^{n_j}(x_{jk(2)}-\overline{x}_{j(2)})^2 \\ \vdots \\ \dfrac{1}{n_j-1}\sum_{k=1}^{n_j}(x_{jk(1400)}-\overline{x}_{j(1400)})^2 \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{1}{29}\sum_{k=1}^{30}(x_{jk(1)}-\overline{x}_{j(1)})^2 \\ \dfrac{1}{29}\sum_{k=1}^{30}(x_{jk(2)}-\overline{x}_{j(2)})^2 \\ \vdots \\ \dfrac{1}{29}\sum_{k=1}^{30}(x_{jk(1400)}-\overline{x}_{j(1400)})^2 \end{bmatrix}
\end{aligned}$$

(Subscripts in parantheses indicate component positions in the vectors.)

Using these statistics, we form a log-pdf for each class symbolically given by

$$\ln\left(f_j(\mathbf{x})\right) = -\frac{1}{2}\left(1400\,\ln(2\pi)+\ln|\widehat{\Sigma}_j|\right) - \frac{1}{2}\left(\mathbf{x}-\overline{\mathbf{x}}_j\right)^T\widehat{\Sigma}_j^{-1}(\mathbf{x}-\overline{\mathbf{x}}_j) \tag{2.3}$$

Since $\widehat{\Sigma}_j$ is a diagonal matrix, we simplify the above by recognizing that

$$|\widehat{\Sigma}_j| = \text{The product of the diagonal elements of } \widehat{\Sigma}_j.$$

$$\widehat{\Sigma}_j^{-1} = \text{A diagonal matrix of reciprocals of the diagonal}^2\text{of } \widehat{\Sigma}_j.$$

We then classify a sample vector $\mathbf{x}_0$ as being in the class that returns the maximum log-likelihood; i.e.,

$$\mathbf{x}_o \in \text{class j} \iff \mathcal{L}(\mathbf{x}_0) = \ln\left(f_j(\mathbf{x})\right)$$

[2]It is well known among numerical analysts that a matrix inverse should not be computed numerically. Not only is it computationally expensive, the possibility of corrupting, undetected error is significant. Even in those matrices that are not diagonal, better and more reliable numerical techniques exist (*cf.* Atkinson [1]).

2.3 Correlation Coefficient

One way to view classification is via a method to group vectors with like characteristics. As an example, given a vector $\mathbf{x}$, suppose we consider all other vectors $\mathbf{y}$ that are linear functions of $\mathbf{x}$ satifying

$$\mathbf{y} = a\,\mathbf{x} + b\,\mathbf{1}, \qquad a, b \in \mathbb{R} \tag{2.4}$$

where $\mathbf{1}$ is a vector of all ones. Then any such $\mathbf{y}$ is just a scaled and/or component-constant shift of $\mathbf{x}$. The defining characteristic is that vectors in this group are either parallel or orthogonal. There is empirical evidence to suggest that our classes of frequency reponses may share parallelism as a means for classification.

One way to statistically examine the above linear property is through a form of correlation. Extending the concept of univariate correlation, we consider a vector correlation ρ defined by

$$\rho \;=\; \frac{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{y})\Big|}{\sqrt{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|\;\Big|\mathrm{Cov}\,(\mathbf{y},\mathbf{y})\Big|}}$$

where $|\cdot|$ indicates matrix determinant. For a vector $\mathbf{y}$ defined as in Equation 2.4, this becomes

$$\begin{aligned}
\rho \;&=\; \frac{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{y})\Big|}{\sqrt{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|\;\Big|\mathrm{Cov}\,(\mathbf{y},\mathbf{y})\Big|}} \;=\; \frac{\Big|\mathrm{Cov}\,(\mathbf{x},a\mathbf{x}+b\mathbf{1})\Big|}{\sqrt{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|\;\Big|\mathrm{Cov}\,(a\mathbf{x}+b\mathbf{1},a\mathbf{x}+b\mathbf{1})\Big|}} \\
&=\; \frac{\Big|a\,\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|}{\sqrt{\Big|\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|\;\Big|a^2\,\mathrm{Cov}\,(\mathbf{x},\mathbf{x})\Big|}} \;=\; \frac{a}{\sqrt{a^2}} \;=\; \begin{cases} -1, & \text{if } a < 0 \\ \;\;1, & \text{if } a > 0 \end{cases}
\end{aligned}$$

Thus two random vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy Equation 2.4 if and only if their vector correlation coefficient ρ is -1 or 1. A value of 1 means the vectors are parallel, and -1 indicates they are perpendicular. So the above correlation coefficient provides an indication of how close two vectors are to being parallel. Accordingly, one would like to use this as some sort of measure. Unfortunately since this vector correlation involves estimating vector covariances, we are faced with the same sample-size vs vector-size issues of the previous MLLE classification method. Again, we can turn to a component-level analysis. For any two vectors satisfying Equation 2.4, the same property applies at the component level.

$$y_i \;=\; ax_i + b, \quad i = 1, 2, \ldots, 1400$$

Since the same a and b are used for each component, we expect that all components of the two vectors should be similarly correlated – across indices. This also suggests that we consider

treating each vector as a single 1400-sized sample of a univariate random variable. We can then examine the associated correlation coefficient for the two scalar random variables as a measure of the distance between the two vectors. This is a variation on the *nearest neighbor* approach to distance found in Hand [2] among others.

Note that this changes our viewpoint from a 1400-dimensional space to single-dimensional, scalar "space". Thus the associated variance-covariance estimates should be quite good due to the 1400-to-1 relation between sample and "vector" size.

2.3.1 Construction of the Correlation Coefficient Classifier

Following Hand [2], we treat the components in any two vectors $\mathbf{x}$ and $\mathbf{y}$ as single samples of size 1400 of respective univariate random variables X and Y. For each jth class, we define the distance between a test vector $\mathbf{x}$ and that class in terms of its distance from the class average $\mathbf{y}_j = \overline{\mathbf{x}}_j$, as represented by the correlation coefficient for X and $Y_j = \overline{X}_j$. We denote this distance as $d(\mathbf{x}, \overline{\mathbf{x}}_j)$, which is given by

$$\begin{aligned} d(\mathbf{x}, \overline{\mathbf{x}}_j) &= \frac{\mathrm{Cov}\,(X, \overline{X}_j)}{\sqrt{\mathrm{Var}\,(X)\mathrm{Var}\,(\overline{X}_j)}} \\ &= \frac{\sum_{i=1}^{1400}(x_{(i)} - \overline{x})(\overline{x}_{j(i)} - \overline{x}_j)}{\sqrt{\sum_{i=1}^{1400}(x_{(i)} - \overline{x})^2 \sum_{i=1}^{1400}(\overline{x}_{j(i)} - \overline{x}_j)^2}} \end{aligned} \tag{2.5}$$

$$\left.\begin{aligned} \overline{x} &= \frac{1}{1400}\sum_{i=1}^{1400} x_{(i)} \\ \overline{x}_j &= \frac{1}{1400}\sum_{i=1}^{1400} \overline{x}_{j(i)} \end{aligned}\right\} \text{ sample averages over all components.}$$

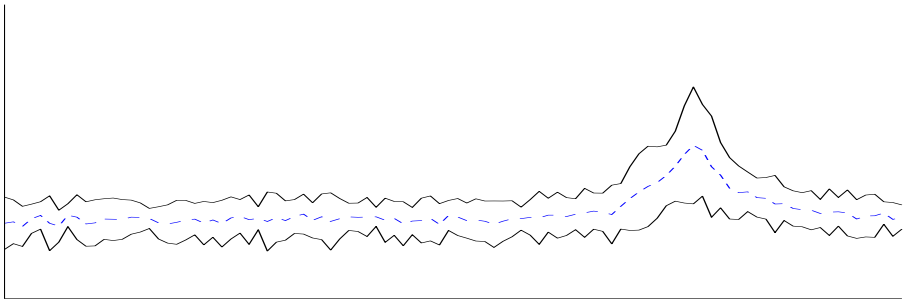
It is reasonable to expect vectors in the same class to have a higher correlation coefficient than vectors in different classes. Given our normal distribution assumptions, it is also reasonable to expect that within a class, a typical vector will be most like the mean of the class. So our classification scheme is to calculate the above distance correlation between an observed vector and each of the sample average vectors for the 22 classes, then place the observed vector in that class represented by the highest correlation value[3].

[3]Intuitively, one expects minimum distance to be desired, not maximum. Since correlation gives values in the interval $[-1, 1]$, an equivalent distance with this minimization property can be constructed by defining $d^*(\mathbf{x}, \mathbf{y}) = 1 - d(\mathbf{x}, \mathbf{y})$.

2.4 Confidence Bands

The confidence band idea was inspired by a graphical/geometric viewpoint. If we can find some reasonably accurate representative vector for a class, then all other observation vectors belonging to this same class should closely "parallel" this representative and be within some minimum distance away from it. In other words, one would like to construct an upper and lower band for each representative vector that also contains a high percentage of the components of other vectors from the same class. This is illustrated in Figure 2.4.

Figure 2.4: Confidence Band



Classification of an observed vector becomes a simple matter of counting how many vector components fall within the confidence band for each class representative. The observed vector is classified with that class representative that has the largest number of the observed vector's components within its confidence band.

An appeal of this approach is its simplicity and small computational requirements. Since classification needs to be done in the field in only a few seconds, reductions in computational complexity are desired. As a comparison, Table 2.1 shows the flop count of the three methods we considered[4]: MLLE, confidence bands, and correlation.

Table 2.1: FLOPS Comparison of the Three Methods

MLLE	Confidence Bands	Correlation
184,844	61,622	370,150

We see that the MLLE approach has almost three times the computational cost over confidence bands, and correlation is about six times more costly. The major reason for this

[4]Computations were done using Matlab 5.3 on both UNIX and PC based systems. "flops" stands for "floating point operations" and is a common measurement of computational cost.

difference is that with the confidence band, the only "on-line" computations involve two vector subtractions and a counting of the number of negative component results. The MLLE has the same number of vector subtractions, but also includes a component-wise multiplication of two vectors[5] and an inner product.

$$\ln\left(f_j(\mathbf{x})\right) \;=\; -\frac{1}{2}\left(1400\,\ln(2\pi) + \ln|\widehat{\Sigma}_j|\right) \;-\; \frac{1}{2}\,(\mathbf{x}-\overline{\mathbf{x}}_j)^T\widehat{\Sigma}_j^{-1}(\mathbf{x}-\overline{\mathbf{x}}_j) \tag{2.6}$$

As can be seen in Equation 2.5 on page 11, correlation requires considerably more arithmetic operations.

2.4.1 Construction of the Confidence Band Classifier

A natural candidate for the representative vector for each category is the mean of that category. As with the MLLE, we use sample averages to estimate the various means.

The upper and lower *confidence bands* about each average are formed from a series of *confidence intervals* (or *widths*), one for each vector component. As with the covariance matrix Σ, we make an important simplification by treating all components as independent. Hence we treat each ith component of each class as having its own univariate normal (Gaussian) pdf. This results in 1400 pdfs for each of the 22 classes. This is similar to what we did when constructing the MLLE.

For simplicity, we consider only one class and drop the class subscript j. So $\overline{\mathbf{x}}$ will refer to a generic class average with $i = 1, \ldots, 1400$ components. Each ith component will have an associated average $\overline{x}_i$, sample variance s_i^2, and sample size n_i.

Although each underlying component population may have a normal pdf, standard techniques require using the Student's t distribution to construct the confidence interval. This is because the mean and variance must be estimated from a sample. Each t_i random variable is defined using the population mean μ_i, sample average $\overline{x}_i$, sample variance s_i^2, and sample size n_i via

$$t_i \;=\; \frac{\overline{x}_i - \mu_i}{\sqrt{s_i^2/n_i}} \;=\; \sqrt{n_i}\,\frac{\overline{x}_i - \mu_i}{s_i}$$

The pdf for the Student's t distribution is given by

$$g(t_i) \;=\; \frac{\Gamma\left(\dfrac{\nu_i+1}{2}\right)}{\sqrt{\pi\nu_i}\;\Gamma\left(\dfrac{\nu_i}{2}\right)}\left(1+\frac{t_i^2}{\nu_i}\right)^{-(\nu_i+1)/2}$$

[5] This comes from using the diagonal structure of Σ. The inverse is never calculated, nor is the matrix-vector product. Further, even if Σ is not diagonal, the inverse would not be calculated numerically. See the footnote on page 9.

where

$$\Gamma(\alpha) = \int_0^\infty y^{\alpha-1}\, e^{-\alpha}\, dy = \text{the } \textit{gamma function.}$$

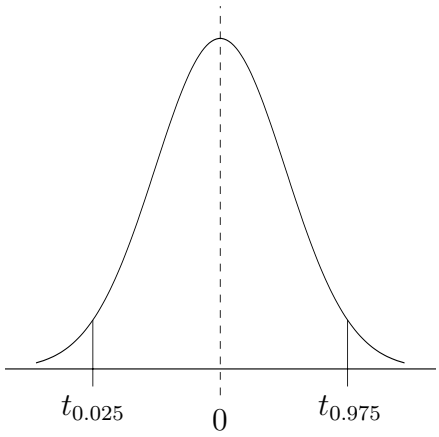
$$\nu_i = n_i - 1 = \text{the degrees of freedom remaining after estimating the mean } \overline{x}_i \text{ with } n_i \text{ samples.}$$

Note that $E[t_i] = 0$, because $E[x_i] = \mu_i$. Thus the Student's t pdf is centered at $t = 0$. The shape of the Student's t pdf is basically like the standard normal, but has fatter tails.

Since all the confidence intervals are constructed in the same manner, we will illustrate one such construction by example (suppressing the subscript i).

Suppose we want to be 95% confident that the interval we construct will contain the true mean μ. (This 95% value is referred to as the *confidence level*.) Suppose further that we want the interval to be located symmetrically about μ. This is equivalent to determining two symmetrically placed values on the horizontal axis of the Student's t pdf plot so that the area under the density curve between the two will be 0.95. (See Figure 2.5.) Since the

Figure 2.5: Confidence Interval Location



error will now fall symmetrically in the tails of the density plot, we will have 1/2 of the probability error in each tail. Thus the probability of falling below (to the left of) the lower bound will be 0.025 as will the probability of falling above (to the right of) the upper bound. We denote these bounds by $t_{0.025}$ and $t_{0.975}$ and observe that they represent upper and lower limits to the following three definite integrals of the Student t pdf.

$$0.025 = \int_{-\infty}^{t_{0.025}} g(t)\, dt, \qquad \text{the lower-tail probability.}$$

$$0.95 = \int_{t_{0.025}}^{t_{0.975}} g(t)\, dt, \qquad \text{the middle interval probability.}$$

$$0.025 \;=\; \int_{t_{0.975}}^{\infty} g(t)\,dt, \qquad \text{the upper-tail probability.}$$

The *confidence interval* is just the middle interval $(t_{0.025}, t_{0.975})$. Since this example has a confidence level of 95%, we generally refer to it as a *95% confidence interval*.

If we let $P(t_{0.025} \le t \le t_{0.975})$ represent the probability that t lies in the interval $(t_{0.025}, t_{0.975})$ and observe that $t_{0.025} = -t_{0.975}$ due to symmetry, then the analytical details are

$$\begin{aligned}
0.95 \;&=\; P(t_{0.025} \le t \le t_{0.975}) \;=\; P\left(t_{0.025} \le \sqrt{n}\,\frac{\overline{x}-\mu}{s} \le t_{0.975}\right) \\
&=\; P\left(\frac{s}{\sqrt{n}}t_{0.025} - \overline{x} \le -\mu \le \frac{s}{\sqrt{n}}t_{0.975} - \overline{x}\right) \\
&=\; P\left(\overline{x} - \frac{s}{\sqrt{n}}t_{0.975} \le \mu \le \overline{x} - \frac{s}{\sqrt{n}}t_{0.025}\right) \\
&=\; P\left(\overline{x} - \frac{s}{\sqrt{n}}t_{0.975} \le \mu \le \overline{x} + \frac{s}{\sqrt{n}}t_{0.975}\right) \quad \text{(symmetry)}
\end{aligned}$$

Thus this *particular sample* of the ith component random variables that was used to compute $\overline{x}$ indicates that the component mean μ satisfies

$$\mu \quad \in \quad \left(\overline{x} - \frac{s}{\sqrt{n}}\,t_{0.975}\,,\; \overline{x} + \frac{s}{\sqrt{n}}\,t_{0.975}\right)$$

Finally, we construct the full confidence band for each class average by forming the sequence of component confidence intervals. We test an observed vector against each of the 22 classification confidence bands by counting how many components lie within a respective confidence band. The class with the maximum number of contained components is where the observed vector is classified.

Some important points to note:

- Although we use the term 95% confidence level, this does *not* mean that there is a 95% probability that the component mean μ takes on a value somewhere within the indicated interval. Such a statement implies that μ is a random variable in its own right. However μ is a parameter with a unique value. Instead, what we have determined is a value for a *random interval* – using estimates from a sample of size n of random variables x from the population associated with a particular ith component. What we are 95% confident about is that 95% of the intervals we create using this method (each time using a different sample from the ith component population) will contain the true component mean μ. If we want to investigate the *probability* that μ takes on one of a particular set of of values, we need to treat it as a random variable, which

requires Bayesian techniques. We did not pursue this further, because working in R^{1400} makes obtaining any *useful* non-zero probability for the associated mean vector almost impossible – unless all intervals are constructed so that the probability of the mean being in the interval is around 99% or greater. This dimensionality problem is similar to that described below for confidence levels of bands vs intervals. However, performing the associated computations are not necessarily the issue. We must also decide what distribution each ith component mean has and estimate the relevant parameters. At some point, we lose faith in how accurately our working model reflects reality.

- Recall that in the construction of a component confidence interval, the Student's t pdf is only used to obtain the value of $t_{0.975}$. This value is based on a standardized distribution and never changes – regardless of which component or class is being considered. Hence the pdf is used only once in this whole process, and its only use is to determine the confidence level for a given $t_{0.975}$-value or vice versa. Further, this computation is done before actually constructing the classifier and is not necessary during testing.

- This is the only method that allows a way for the designer to influence method accuracy (apart from specifying training sample size). As will be seen in the chapter on results, there is no definitive way to determine an optimal $t_{0.975}$-value. This introduces a certain amount of subjectivity, which could also be interpreted as flexibility.

Confidence Levels for Bands vs Intervals

A word needs to be said here about the confidence level for the *complete band* vs that for *individual intervals*. In the above example, we specified a 95% confidence level for each interval. Each band contains 1400 such intervals. So under our assumption of component independence, our overall probability that a band will contain the true mean is given by $0.95^{1400} \approx 6.50 \times 10^{-32}$ or virtually zero. To achieve any useful confidence level for the whole band, we need to increase our individual interval level above 95%. Table 2.2 contains a selection of comparisons using the Student's t distribution with 29 degrees of freedom. The

Table 2.2: Confidence Levels: Bands vs Widths

t-value	2.045	2.500	5.000	8.500
Interval	0.950	0.982	0.9999756	0.9999999977
Band	6.50×10^{-32}	5.68×10^{-12}	0.9651099	0.9999967828

purpose of this information is to demonstrate how large the confidence level for intervals must be to obtain useful levels for bands. The particular value of $t = 8.5$ is shown because

it is the value eventually used with the classifier. This choice of t-value was based on an examination of results over a wide range of values. This is further discussed in Chapter 3 on page 26.

An additional point is that although $t = 8.5$ is associated with fairly high probabilities, working with these is well within the capabilities of current technology.

Confidence Bands vs Prediction Bands

It has recently be suggested that a *prediction band* would be more appropriate. In this setting, for each jth class, such a band would be constructed of *prediction intervals* in a manner similar to using confidence intervals for confidence bands. Then a test vector would be placed in that class with the largest number of its components lying within the appropriate jth class prediction band.

To construct a prediction interval, consider any ith component of the jth class vector mean, denoted $\overline{x}_{j(i)}$, and let $x_{0(i)}$ brespectively, we form the associated t-statistic

$$t \;=\; \frac{\overline{x}_{j(i)} - x_{0(i)}}{\sqrt{s^2_{j(i)}/n \,+\, s^2_{0(i)}}} \quad\sim\quad t_{n-1}$$

which leads to the ith prediction interval of

$$x_{0(i)} \;=\; \overline{x}_{j(i)} \,\pm\, t_{n-1,1-\alpha/2}\,\sqrt{s^2_{j(i)}/n \,+\, s^2_{0(i)}}$$

The advantage to this interval is that its width has a nonzero lower bound as $n \to \infty$. On the other hand, a paradox occurs in determining $s^2_{0(i)}$, because we don't know *a priori* which class it is from. One plausible solution may be to use $s^2_{0(i)} = s^2_{j(i)}$ with each jth class. This leads to prediction intervals of the form

$$x_{0(i)} \;=\; \overline{x}_{j(i)} \,\pm\, t_{n-1,1-\alpha/2}\,\sqrt{s^2_{j(i)}\,(1/n+1)}$$

2.5 Bayesian Classifiers

A preliminary investigation was made into using Bayesian statistics to build a classifier. The aesthetic value in this approach is that one might then be able to assign a probability to the correctness of a classification. However as is shown below, computational difficulties became an issue, and necessary modification of the method returns something equivalent to the MLLE.

Although nothing is known *a priori* about which class of objects one might encounter in the field, it seems reasonable to start with the assumption that each of the 22 classes is

equally likely. Hence our set of parameter pairs $\{\boldsymbol{\theta}_j\} = \{(\boldsymbol{\mu}_j, \Sigma_j)\}$ has a discrete uniform distribution (called a uniform *prior* distribution), each with probability 1/22. Since it is discrete, the prior also represents a pdf, and we use the notation $\pi(\boldsymbol{\theta})$ for both. Using Bayes' Rule, we can (theoretically) find the conditional probability distribution $\pi(\boldsymbol{\theta}_j \,|\, \mathbf{x}_0)$ for the parameters $\boldsymbol{\theta}_j$, which represents the probability that $\mathbf{x}_0$ came from the jth distribution. (This is called the *posterior* distribution). Symbolically, this is derived via

$$\begin{aligned}
\pi(\boldsymbol{\theta}_j \,|\, \mathbf{x}_0) &= \frac{f_{\theta,x}(\boldsymbol{\theta}_j\,,\, \mathbf{x}_0)}{f_x(\mathbf{x}_0)} = \frac{f_{x|\theta}(\mathbf{x}_0 \,|\, \boldsymbol{\theta}_j)\, \pi(\boldsymbol{\theta}_j)}{\sum_{j=1}^{22} \Big[f_{x|\theta}(\mathbf{x}_0 \,|\, \boldsymbol{\theta}_j)\, \pi(\boldsymbol{\theta}_j) \Big]} \\
&= \frac{f_j(\mathbf{x}_0)\, \frac{1}{22}}{\sum_{j=1}^{22} \Big[f_j(\mathbf{x}_0)\, \frac{1}{22} \Big]} = \frac{f_j(\mathbf{x}_0)}{\sum_{j=1}^{22} f_j(\mathbf{x}_0)} \qquad (2.7)
\end{aligned}$$

Here we follow the convention that $f_{\theta,x}$ is the *joint* pdf for $\mathbf{x}$ and $\boldsymbol{\theta}$, that $f_{x|\theta}$ is the *conditional* pdf for $\mathbf{x}$ given θ, and recognize that $f_{x|\theta}(\mathbf{x}_0 \,|\, \theta_j) = f_j(\mathbf{x}_0)$ (as in Equation 2.3, pp 9).

As with the MLLE, the high dimensionality of our observations causes a numerically computed value of zero to be returned for the value of $f_j(\mathbf{x}_0)$ for any jth class (Equation 2.1 on page 7). This affects the denominator as well as the numerator of the above expression, and we end up with $\pi(\boldsymbol{\theta}_j \,|\, \mathbf{x}_0) = 0/0$. One might hope to take natural logs as before, evaluate the arithmetic operations, then compose the result with the exponential function to finally obtain the desired probability. However we cannot do this, because we have

$$\begin{aligned}
\ln\Big(\pi(\boldsymbol{\theta}_j \,|\, \mathbf{x}_0)\Big) &= \ln\left(\frac{f_j(\mathbf{x}_0)}{\sum_{j=1}^{22} f_j(\mathbf{x}_0)} \right) \\
&= \ln\Big(f_j(\mathbf{x}_0)\Big) - \ln\left(\sum_{j=1}^{22} f_j(\mathbf{x}_0) \right) \\
&\neq \ln\Big(f_j(\mathbf{x}_0)\Big) - \sum_{j=1}^{22} \ln\Big(f_j(\mathbf{x}_0)\Big)
\end{aligned}$$

Unfortunately the last expression is the only one we can numerically evaluate with meaningful accuracy. This means we cannot obtain a useful numerical value for the denominator in Equation 2.7, nor for the desired probability. Since the denominator is constant for each $\pi(\boldsymbol{\theta}_j \,|\, \mathbf{x}_0)$, we could ignore it and use the log of the numerator as a relative measure for comparison between pdfs. But we already have this with the MLLE, so nothing useful is gained with a Bayesian approach.

2.6 Symmetry Metric

Physically, an object from any one class has a counterpart in another class that is symmetric in its construction. Using the hole-in-plate analogy, one can characterize this via a "left-hand" class and a "right-hand" class. This results in a coupling of the 22 actual classes into 11 symmetry pairs of classes. An issue underlying classification is to what extent the measured frequency response vectors reflect this physical symmetry of classes. In other words, do the frequency responses of two vectors from respective symmetry pairs differ in any appreciable way?

From a design standpoint, actual probing should not be able to detect any meaningful difference between an object from a left-hand class and another object from the respective right-hand class. As an example, A and S represent two classes that are symmetry pairs. An object from the A class should produce a highly similar frequency response when compared to a response from an object in the S class. Looked at another way, the A-class response should be like the mean of the S class in some quantifiable way. Along these lines, we split the classes into their respective left-hand and right-hand families. We then sought to use each classification method to test left-hand *vectors* against only right-hand *classes*. Using the previous example, if a classification method is truly "symmetry blind", then an A-class vector should be classified in its symmetrical S class. Unfortunately, the percentage of correct pairings was quite low for all cases. A clue to why this occurred may be given in a plot of the two class averages for the A-S symmetry pair (Figure B.3, pp 52). Clearly there are differences. Determining how important these are requires some way to quantify either the differences or the similarities. At this stage, we have developed a measure related to the degree of symmetry similarity for the MLLE method. We call this a *symmetry metric*[6].

2.6.1 Construction of the Symmetry Metric

The MLLE symmetry metric that was created is similar to the log-likelihood ratio that underlies much of hypothesis testing. First we find the MLLE value of a given vector $\mathbf{x}$, then determine the log-likelihood value of the respective symmetry class (which we will denote by SLLE). We then form the ratio of the MLLE divided by the SLLE. If we use $M(\mathbf{x})$ to represent the symmetry metric, then mathematically we can write

$$M(\mathbf{x}) \;=\; \frac{\mathrm{MLLE}(\mathbf{x})}{\mathrm{SLLE}(\mathbf{x})}$$

One appealing feature of this definition is that

$$0 \;\leq\; M(\mathbf{x}) \;\leq\; 1$$

[6]Mathematically this is not a true metric. However it is a consistent indicator of distance between a vector's true class and its symmetry class.

So the symmetry metric is similar in this respect to probability or the absolute value of correlation[7]. The reasons for these inequalities are as follows.

- Since we are considering logarithmic values, both the MLLE and SLLE will never be positive, which ensures that the symmetry metric is always nonnegative.
- By definition the MLLE is always greater than or equal to the SLLE. But both being negative, the MLLE is smaller than the SLLE in absolute value. Hence the symmetry metric is bounded above by one.

As further discussed below, we would like to eventually reduce the testing dimension of the frequency response vector. We hope to create a minimal size window of vector components by identifying those components that are key to classification and discarding the others. Since classification will be made with respect to symmetry, we can use the symmetry metric to evaluate how well various windows maintain the connection between symmetry pairs. To this end, we augmented our Matlab code to allow for changing windows. This is reflected with the new notation $M(I, \mathbf{x})$, where I indicates the windowing interval being used.

2.7 Dimension Reduction

Working in $\mathbb{R}^{1400}$ and facing Bellman's *curse of dimensionality* makes dimension reduction very desirable. Future work will investigate ways to reduce the number of components necessary from a vector to make a classification. Possible methods under consideration are:

- One-way ANOVA performed on each component across the 22 classes (as levels). If this shows promise, we may extend this to multi-way ANOVA performed on several components across the 22 classes. Hopefully this will indicate those components that statistically give the same average results regardless of class. These can then be dropped from consideration in any classification method.
- Hypothesis testing for equality between two means. This would be performed on each respective component pair across two classes deemed symmetric. The goal here is to identify those components that are statistically the same within a symmetry class pair. This may indicate where the probing device is registering desired data about the symmetry class. This may also indicate where symmetry data in general is contained in the frequency response vector. Components outside of these symmetry groupings may not contain relevant information specific to classification.

[7]On the other hand, $1/M(\mathbf{x})$ is conceptually similar to a F statistic. This reciprocal is actually the symmetry metric first developed for the project.

- Clustering techniques. With the increase in genetic identification and data mining, several techniques are available to investigate whether data has some sort of natural grouping. We hope to use some of these to see if certain components tend to cluster together. We may then be able to eliminate certain components or use a single representative of a resulting cluster for classification.

Chapter 3

Results, Conclusions, Future Work

3.1 Results

3.1.1 Notation

Since several people were involved in this project, we needed a common, consistent way to represent and transfer data. Brief descriptions of data form and the relevant Matlab data structures are given to aid in interpreting results. More complete descriptions appear in the appendices.

Class identification

Each classification method was trained from the same set of 660 vectors in $\mathbb{R}^{1400}$. This set is commonly refered to as the "A-data". The set was subdivided into 22 classes of 30 vectors each. Each subdivision contained only vectors known to be from the respective class. Each class had three identification designations: (1) one of the integers 1-22, (2) a capital letter A-V (refered to as the "GS" notation), and (3) a matrix designation M_{ij} (refered to as the "JAB" notation). These are related through the following schemes:

GS Notation

A	B	C	D	E	F	G	H	I	J	K
1	2	3	4	5	6	7	8	9	10	11
L	M	N	O	P	Q	R	S	T	U	V
12	13	14	15	16	17	18	19	20	21	22

JAB Notation

1		8	12		19
2	5	9	13	16	20
3	6	10	14	17	21
4	7	11	15	18	22

Typically a class is refered to by its capital letter (GS representation) or integer equivalent. The JAB representation is used to indicate the symmetry pairs among the classes. For

example, those vectors from Class A (or 1) are symmetric to those from Class S (or 19). Similarly, Class E (or 5) and Class P (or 16) are symmetric, as are Class K (or 11) and Class O (or 15). If one inserts a vertical line between columns three and four of the JAB matrix, this becomes a line of symmetry for the matrix entries. Note that two matrix positions are empty, because there are only 22 classes in the 4×6 matrix.

Data Organization for Testing

Each of the testing methods was used on the 660 vectors in the A-data for a consistency check. We then used another independent set of data called the "B-data" for a "real-world" simulation. It consists of five vectors from each of the 22 classes for a total of 110. Each of the two data sets was organized by class into a Matlab *cell array*. Each cell array contains class identification information and the associated set of raw data vectors.

- `CA_basic`

  This contains information and the raw data vectors from the A-data set organized into the 22 original classes. When `CA_basic` is referred to in this chapter, one can think in terms of the A-data as the training set for the method being considered. In these cases, the method is used to perform a consistency check to see if it correctly classifies the data it was trained with.

- `CB_basic`

  This is the B-data analog to `CA_basic`, except that it contains testing data, not training data. When `CB_basic` is referred to in this chapter, one can think in terms of the B-data being used for "real-world" testing of the method being considered.

In addition to the three testing methods, there are three classification schemes available for each method. These schemes are variations on classifying vectors into their proper symmetry class; i.e., the combined class of its true class and respective class of symmetry as indicated by the JAB matrix. The three schemes are

- *Full classification* of vectors into one of the 22 original classes, then grouping any symmetry "missclassifications" in with proper classifications. As an example, suppose an A-class vector was classified in the S-class. It would be considered correctly classified, because A and S are symmetry pairs (as represented by 1 and 19 in the JAB matrix).

- *Combined classification* of vectors into one of the 11 combined symmetry classes formed by unioning the data from symmetry pair classes in accordance with the JAB matrix.

- *Left-to-right classification* of vectors. In this case, vectors from those classes on the "left" symmetry side of the JAB matrix were classified only to those classes on the "right" symmetry side of the JAB matrix. Of interest was how well a method put a vector into its associated symmetry class.

All method/scheme combinations relied on comparisons with a set of average vectors – either a set representing the 22 true classes or one representing the 11 combined symmetry classes. MLLE additionally required diagonal variance matrices, and multivariate log-normal pdf coefficients

$$-\frac{1}{2}\left(1400\,\ln(2\pi)\,+\,\ln|\Sigma|\right)$$

to evaluate log-normal pdfs. Two sets of the above three parameters were created and stored in `CA22mcov` and `CA11mcov`. The A-data served as the source for both of these structures, i.e., as the training data.

- `CA22mcov` contains the sample average vector, diagonal variance matrix, and multivariate log-normal pdf coefficient for each of the original 22 classes. Tests involving this structure are geared to full classification to actual class. However, as noted above, these results are subsequently adjusted for symmetry.
- `CA11mcov` contains the average vector, diagonal variance matrix, and multivariate normal adjustment constant for each of the 11 combined symmetry classes. These symmetry classes were constructed by first unioning the two 30-vector sets from each class in a symmetry pair into one 60 vector sample (from the A-data), then finding the associated average, sample variance, and multivariate log-normal pdf coefficient. Tests involving this structure were compared with the adjusted symmetry results from `CA22mcov`.

3.1.2 Method Evaluation

General Evaluation Procedure

The following set of evaluation tests was done on each of the three methods.

1. Using the A-data in `CA_basic`
   - Perform a consistency check by using the *full classification* scheme, i.e., with `CA22mcov` as the target.
   - Perform a consistency check by using the *combined classification* scheme, i.e., with `CA11mcov` as the target.
   - Perform a consistency check by using the *left-to-right classification* scheme, i.e., using the appropriate "left-hand" vectors from `CA_basic` and "right-hand" classes from `CA22mcov`.
2. Perform the same tests as above using the B-data contained in `CB_basic`.

The basic results of these evaluations are summarized in Table 3.1.

Table 3.1: Method Success Rates

Method		Scheme		
	Cell Array	Full	Combined	Left-to-Right
MLLE	`CA_basic`	99.7%	99.5%	20.6%
	`CB_basic`	15.5%	24.5%	4.5%
Confidence Band	`CA_basic`[a]	95.5%	90.0%	[b]
	`CB_basic`[a]	18.2%	13.6%	
Correlation	`CA_basic`	98.8%	97.7%	29.1%
	`CB_basic`	97.3%	95.5%	32.7%

[a]These values were obtained with a t-value of 8.5.

[b]MLLE and Correlation did so poorly that it was decided not to proceed with Confidence Bands.

3.1.3 Overall Results

The MLLE clearly does best on the consistency check with the A-data (`CA_basic`) when using the full and combined schemes. However it falls short on the "real-world" test with the B-data (`CB_basic`). In fact, every observation in the B-data was placed in the S class. To investigate this further, the A-class and S-class observations were removed from the B-data (because they are symmetry pairs), and another test with the full scheme was made with the S-class option removed. In this case, the success rate increased to 21.1% relative to the reduced set. This translates into 35.5% over the complete set. This is still a disappointment, but at least the misclassifications were not all to the same class. They were quite varied (even within classes). This may suggest that the S-class frequency responses contain some sort of central information, perhaps pertaining to physical characteristics of all objects[1]. On the other hand, the above difficulties with the B-data may be due to trying to estimate statistics on a 1400 dimensional system with only 30 samples per class (from the A-data). It should be noted that similar types of results have been reported by others at least as early as Hand [2] in 1981.

The confidence band method parallels MLLE. This seems to agree with intuition. As with MLLE, confidence bands rely on estimates of class vector means and covariances – hence they are susceptible to the same sensitivities. The above results were obtained with a t-value

[1]It is possible that these characteristics may only be due to a set of common manufacturing anomalies.

of 8.5. From Table 2.2 on page 16, we recall that this gives an overall band confidence level of 99.99967828%. This t-value was chosen, because it is the first t-value where the success rates for both the full and combined schemes began to stabilize (although at different rates for the A-data and for the B-data). One might interpret this t-value as where the method first becomes robust. One suggestion was made to choose the t-value that produces optimal classification. However, defining "optimal" became difficult. As Figure 3.1 shows on page 27, this changed depending on the data set and scheme.

The correlation approach does the best overall. It performs roughly the same for both sets of data when used with either the full or combined scheme. The success rates are all above 95% with only a 3.3 percentage point spread. This seems to indicate it is quite robust.

A look at the MLLE and correlation results for the left-to-right scheme shows that this is not a promising approach. Because of these initial results, it was decided not to pursue writing code for the confidence band method. The poor performance of the left-to-right scheme may be another example of attempting to estimate a high-dimensional mean vector with too few samples. However, the design of this scheme directly tests a set of vectors against their symmetry classes only. Hence, it raises the question that the frequency responses themselves may not be truly "symmetry blind" – or perhaps that a significant window of the components may be unrelated to classification and mask the presence of symmetry in the other components.

3.1.4 Related Observations

Confidence Bands and the t-value

As mentioned in the previous chapter, the confidence band method is dependent on the choice of t-value used to construct the classifier – and that this is determined by the researcher. Figure 3.1 shows a plot of confidence band method success rates vs t-values. "$*$" represents A-data, "`o`" represents B-data, solid lines represent the full scheme (`CA22mcov`), and dashed lines represent the combined scheme (`CA11mcov`).

t-values that were examined ranged from 0.5 to 15 at intervals of 0.5. As the plot illustrates, both schemes showed little change for t-values above 8.5 (although at different levels for the A-data and B-data, respectively). This may be interpreted as the point where the confidence band method becomes robust. No choices for t-values lower than 8.5 seem clearly appropriate.

MLLE Symmetry Metric

Since the left-to-right scheme was not as successful as hoped, we turn to the MLLE symmetry metric to help give some relative measure of how far the MLLE method with the full scheme

Figure 3.1: Confidence Band Success Rates vs t-value

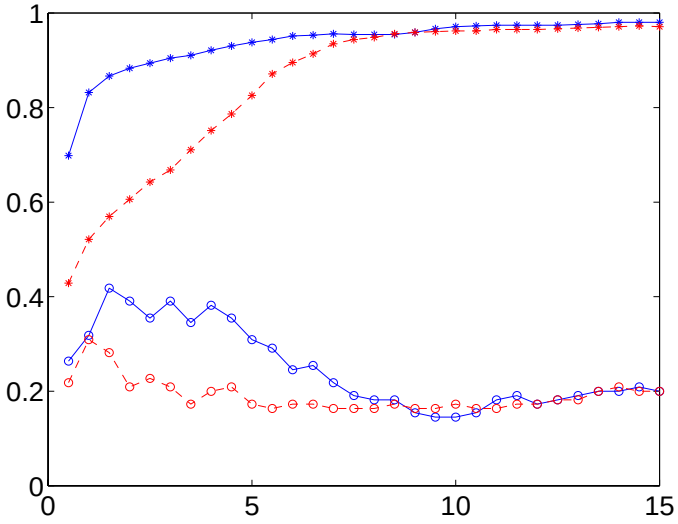


Table 3.2: Average MLLE Symmetry Metric Results

Left-to-Right			Right-to-Left		
True Class	Ratio	Symmetry Class	True Class	Ratio	Symmetry Class
A	0.7999	S	S	0.6287	A
B	0.6837	T	T	0.5674	B
C	0.7065	U	U	0.5790	C
D	0.6898	V	V	0.6651	D
E	0.5528	P	P	0.4765	E
F	0.7324	Q	Q	0.7445	F
G	0.6608	R	R	0.4845	G
H	0.8110	L	L	0.6622	H
I	0.7192	M	M	0.6697	I
J	0.8567	N	N	0.7451	J
K	0.6425	O	O	0.5057	K

is from complete symmetry identification. We looked at each of the 22 classes individually using the A-data[2]. Note that this allowed us to consider ratios of right-to-left symmetry as well as left-to-right. Since the symmetry metric gives a ratio for each vector submitted, we used the average results from the set of vectors from each class as an overall measure. These average ratios are listed in Table 3.2.

Curiously, inspection shows that except for the FQ symmetry pair, left-to-right ratios are larger than right-to-left ratios. We can also observe that the JN pair has the highest ratio, starting from either left or right. However there does not seem to be this same sort of shared ordering as we continue. For example, the HL pair has the second highest left-to-right ratio, but has the fifth highest right-to-left ratio. Perhaps at this point, we can only conjecture that the JN pair may be a good starting point to look for components related to symmetry. Recalling the MLLE performance on the B-data (classifying everything in the S class), we observe that the AS pair has the third highest ratio (left-to-right).

If we take the overall average of the above ratios, we get 0.6643. This can serve as a benchmark, since it evaluates symmetry over the complete frequency response dimension. Thus we can compare this with ratios from other windowed portions of the vector space, with metric values closer to unity being more desirable.

3.2 Conclusions and Future Work

At this time, we have three classifiers that rely on information from all 1400 components in a submitted frequency response vector: MLLE, confidence band, and correlation coefficient. Table 3.1 above indicates that the correlation coefficient method is a fairly accurate and robust method. Although it is the most computationally intensive, this is likely not an issue with current technology.

All the classifiers were constructed from a 660 vector set, 30 for each of the 22 classes, called the A-data. They were all tested with a 110 vector set, five for each of the 22 classes, called the B-data. From statistical theory, we know that 30 samples per class are not enough to obtain unbiased estimators for all the statistics used in the MLLE and confidence band classifiers, i.e., for the vector mean and vector variance/covariance matrix. In high-dimensional settings such as ours, this difficulty is sometimes refered to as Bellman's *curse of dimensionality*. On the other hand, this may provide insight as to why the correlation coefficient classifier does so much better. It operates by treating the submitted vector and each class average vector as two 1400-sized samples taken from populations of univariate random variables. Thus estimating the correlation distance between the vectors is only degraded by the original estimate of the vector mean (used as the second set of random variable observations). The vector variance/covariance estimate from MLLE does not influence correlation results, so its additional variability is not present.

[2]Since the MLLE classifier did so poorly with the B-data, we did not use it with the symmetry metric.

With the above in mind, if the empirical success rates given in Table 3.1 for the correlation coefficient are acceptable, this researcher recommends adopting that method. However, the experience and intuition of some involved with the project have led us to believe that further investigation into component identification (as to associated physical characteristics) and subsequent dimension reduction are worthwhile. This is because certain portions of the vectors (currently unknown) may include extraneous and misleading information. There may also be components that carry information common to all classes. Thus we believe that future work identifying component relevance and related dimension reduction may be fruitful in all of the methods. In this regard, three particular avenues to pursue have been identified.

1. Some components may not relate to physical characteristics of interest. Using the hole-in-plate analogy, hole depth is irrelevant and unreliable as a classification indicator. We seek to identify and remove the irrelevant components.

2. Certain component values may reflect distinctions specific to a particular manufacturing site and/or production run within a site. Again, hole depth is an example. When changing from manufacturing one class of device to another, care is taken in hole placement, but a wide tolerance for depth is acceptable. So devices made during the same batch process will have the same hole depth, but may easily differ from those of the same class but manufactured during a different batch process. We seek to identify and remove these confounding or masking components.

3. From a probe design standpoint, if two objects are taken from two separate yet symmetric classes, actual probing should not be able to detect much of an important difference between them. With the hole-in-plate analogy, this is equivalent to having a hole on the left side of the plate mirrored by one on the right side, with probing being done along the line of mirror symmetry. As far as the probe is concerned, the whole is y units away from the probe along the symmetry line and x units away perpendicular to the symmetry line. The perpendicular direction cannot be detected by the probe. Consequently, there should be associated frequency response components that return roughly the same values no matter which of the two symmetry classes the sample comes from. We seek to identify and retain these components.

An initial investigation into the symmetry issue led to development of the symmetry metric. As we progress, it can both benefit from dimension reduction and help measure its effectiveness. Effective dimension reduction should improve the estimates for class mean and the variance/covariance matrix. In turn, this should improve the quality of the results from the symmetry metric. At the same time, the symmetry metric may indicate when removal of certain components represents a loss of symmetry information.

As we seek further insights into symmetry and the other issues, some suggested approaches to component identification include

- One-way ANOVA performed on each component across the 22 classes (as levels), using the 30 samples for each class in the A-data. If this shows promise, we may extend to a multi-factor ANOVA performed on several components across the 22 classes. Hopefully this will indicate those components that statistically give the same average results regardless of class.

- Hypothesis testing for equality between two means. This would be performed on each respective component pair across two classes deemed symmetric. The goal is to identify those components that are statistically the same within a symmetry class pair. This may indicate where the probing device is registering desired data about the symmetry class. This may also indicate where symmetry data in general is contained across all classes.

- Clustering techniques which seek to determine various ways in which data seems to be related or can be grouped. The relatively recent increase in work on genetic identification and data mining has provided useful techniques to investigate whether data has some sort of natural grouping. We hope to use some of these to see if certain components tend to cluster together – either across all classes or within significant families of classes. We may then be able to eliminate undesired or redundant components and/or use a single representative of a component cluster for classification.

Because ANOVA and hypothesis testing typically rely on pdf assumptions similar to the MLLE, these approaches may be susceptible to associated estimation limitations such as sample size. Visually this may be illustrated by the plot of class averages in Figure B.2 on page 51. Given the approximately 70 unit range of response values, the variation among the averages exhibited in the plot exceeds what is expected from the actual physical differences between classes. However, judicious grouping of components and/or classes may provide valuable insights.

Clustering employs a wide range of evaluation techniques, mainly based on some definition of "closeness" or distance. One can cluster via almost any numerical measure that can be defined for the data. However this diversity of options can also have associated imprecision – in actual implementation of the clustering as well as related to interpretation of results. For example, judicious choices for distance measures are not always apparent and may depend on the viewpoint of the researcher. Another aspect directly influenced by user input is how to define "close". As we begin our investigation into these areas, we acknowledge that ANOVA and hypothesis testing can be viewed as types of clustering. We have listed them separately, because of their well-known and well-established roles in many fields of research. In addition to these, we will likely first look toward some use of correlation, because of its better performance with small sample sizes relative to sample dimension. If these are not satisfactory, we can begin the search for other distance measures.

Finally, any good researcher should accept the possibility that a key premise may be incorrect. In our case, we need to consider that the probing device does not perform as expected

– whether by design, manufacture, operation, or accident. At this point, any evidence to suggest this may, instead, be suitably attributed to the *curse of dimensionality* that underlies this whole enterprise. Perhaps our investigations into clustering and component identification will also help us with this conundrum.

Appendix A

Matlab Files

A.1 File Listings

Two types of listings are provided – alphabetical and by subject. The alphabetical list gives a general description of the type of file. The subject listing does not follow the alphabetical scheme. It is grouped more by what the file is used for. It is also sublisted to indicate various file dependencies.

A.1.1 Alphabetical Listing

Filename	File Type
`CA_b11.m`	data organization file
`CA_b22.m`	data organization file
`assem2.m`	data organization file
`citest.m`	confidence band method file
`conwidth.m`	confidence band method file
`corrtest.m`	correlation method file
`eval_mle.m`	MLLE method file
`gs2jab.m`	class identification file
`jab2gs.m`	class identification file
`makeCAs.m`	data organization file
`mleparam.m`	MLLE method file
`mletest.m`	MLLE method file
`read_ds.m`	data organization file
`symallav.m`	MLLE symmetry metric file
`sympair.m`	MLLE symmetry metric file
`pltnvecs.m`	vector graphing file

A.1.2 Subject/Hierarchical Listing

- Training/Assembling Files

```
makeCAs.m
  - read_ds.m
  - CA_b22.m
      - assem2.m
  - mleparam.m
  - CA_b11.m
      - assem2.m
conwidth.m
```

- Symmetry Metric Files

```
symallav.m
  - sympair.m
      - eval_mle.m
      - gs2jab.m
      - jab2gs.m
```

- Field Testing Files

```
mletest.m
  - eval_mle.m
  - gs2jab.m
  - jab2gs.m
citest.m
  - gs2jab.m
  - jab2gs.m
corrtest.m
  - gs2jab.m
  - jab2gs.m
```

- Plotting Files

```
pltnvecs.m
```

A.2 Data Representations and File Descriptions

A.2.1 Data Structures and Cell Arrays

All training/assembling files work with at least one of two types of Matlab data representations: data structures and cell arrays. These in turn are constructed from a family of 22 Matlab MAT-files, one for each class provided as training vectors for the various classifiers. These contain the A-data. Each of the MAT-files contains a 30×1400 matrix of row vectors, which is transposed into a 1400×30 matrix at the time of initial training/assembling. Data structure `DS_basic` is assembled from the MAT-files and is used to construct the cell array `CA_basic`, which is then used to assemble the data structures `CA22mcov` and `CA11mcov`. Data structures and cell arrays are constructs used to group different types of information into one unit. They are conceptually similar but differ in the ways that they specifically handle data. More on these types of data representations can be found in Matlab manuals or by typing "help struct" or "help cell" at the prompt while in a Matlab session[1].

`DS_basic` is a 1×22 data structure array assembled directly from the A-data. Each array cell has the two fields `data` and `array_label`.

[1] "help struct" gives one way to create a data structure. These can also be created "on-the-fly", as is the case in our `read_ds.m`

- `data` contains the matrix from the associated A-data MAT-file that has been transposed into column vectors.
- `array_label` contains the (i, j) class identification associated with the JAB matrix representation.

The ith `cell.field` is accessed via `DS_basic(i).data` or `DS_basic(i).array_label`.

`CA_basic` is a 22×4 cell array created from `DS_basic`. Each row represents a class, and each column contains, respectively

1. The capital letter identification of the class, i.e., A, B, ..., V.
2. The integer identification of the class, i.e., 1, 2, ..., 22.
3. A vector of component identifications common to each class column vector. This could be used to designate a window of components smaller than the whole vector. At this time it is the complete vector represented as $\begin{bmatrix} 1 & 2 & \cdots & 1400 \end{bmatrix}$.
4. The associated ith class matrix from `DS_basic(i).data`; i.e., the 1400×30 matrix from the A-data training set.

`CA22mcov` is a 1×22 data structure assembled from `CA_basic`. Each array cell has the three fields `mu`, `sigma`, and `con`.

- `mu` contains a 1400×22 matrix of the sample average for each class. Each average is obtained from the 30 column vectors in the cell array `CA_basic{i,4}`.
- `sigma` has the 1400×22 matrix of variance vectors. Each vector is taken from the assumed diagonal variance-covariance matrix.
- `con` is a single 1×22 vector of constants. Each element is associated with the log of the multivariate normal pdf for each class

$$\ln\left(f_j(\mathbf{x})\right) = \underbrace{-\frac{1}{2}\left(1400\,\ln(2\pi) + \ln|\widehat{\Sigma}_j|\right)}_{\texttt{con}} - \frac{1}{2}\left(\mathbf{x} - \overline{\mathbf{x}}_j\right)^T \widehat{\Sigma}_j^{-1}\left(\mathbf{x} - \overline{\mathbf{x}}_j\right) \quad \text{(A.1)}$$

`CA11mcov` is similar to `CA22mcov`, except that it is constructed from the 11 combined classes formed from symmetry pairs. Hence it is a 1×11 data structure with appropriately sized fields `mu`, `sigma`, `con`.

A.2.2 File Descriptions

CA_b11.m

M-file that creates the 11 × 4 cell array `CA_bas11`, similar to `CA_basic`, from the A-data. (See above for description.) Each row represents a symmetry pair class. Columns for each row contain: the two symmetry pair names, the two symmetry pair numbers, a vector of component indices (1 to 1400), and the actual 60 data vectors from the unioned symmetry pairs. As an example, typing `CA_bas11(1,:)` at a Matlab prompt would return

```
'AS', [1x2 double], [1x1400 double], [1400x60 double]
```

The values in the four cells would be

```
[AS], [1 19], [1 2...1400], [the 30 A-vectors and the 30 S-vectors]
```

This file is called by `makeCAs.m` and requires the file `assem2.m`.

CA_b22.m

M-file that creates the 22×4 cell array `CA_basic` from the A-data. (See above for description.) Each row represents a data class. Columns for each row contain: the class name, the class number, a vector of component indices (1 to 1400), and the actual 30 data vectors from the associated A-data. As an example, typing `CA_basic(1,:)` at a Matlab prompt would return

```
'A', [1], [1x1400 double], [1400x30 double]
```

The values in the four cells would be

```
[A], [1], [1 2...1400], [the 30 A-vectors]
```

This file is called by `makeCAs.m` and requires the file `assem2.m`.

assem2.m

Function that assembles the set of desired matrices (or submatrices) of vectors for all rows of the fourth cell of a cell array like `CA_basic`. For example, `CA_basic(:,4)` would contain 22 matrices, one for each of its 22 rows. The call looks like

```
CA_basic(:,4)=assem2(CA_basic(:,2),CA_basic(:,3),DS_basic);
```

`CA_basic(:,2)` contains the list of class numbers, 1-22. `CA_basic(:,3)` contains the list of component indices desired for each class (e.g., [1, 2, . . . , 1400] or [20:300,750:1000]). `DS_basic` is the data structure containing the set of full matrices for each class. The function is called by `CA_b11.m` and `CA_b22.m`.

citest.m

Function that classifies a submitted column vector (or matrix of column vectors) using the confidence band method. It requires the prior creation of a matrix of confidence widths – one column vector for each class – via the function `conwidth.m`. The call looks like

```
classid=citest(testvecs,CA22mcov,ciwidth,[1:1400]);
```

where `ciwidth` is the matrix of confidence widths. The output is a row vector for each test vector listing the vector number, number of components within the confidence band, actual classification, and associated symmetry pair class.

conwidth.m

Function that creates a 1400×22 matrix of confidence interval widths, one for each component average of each class. This is a training/assembling file used with `citest`. The call looks like

```
ciwidth=conwidth(CA22mcov,30,t);
```

where `CA22mcov` is a data structure of averages, sample variances, and pdf constants (see above); `30` is the number of training vectors (for 29 degrees of freedom); and `t` is the upper-cutoff t-value for the confidence level. The function requires the Matlab statistics function `tcdf.m` to determine the confidence levels for each interval and the band. However this is not necessary to construct the actual confidence widths, so those respective lines can be commented out if `tcdf.m` is not available.

corrtest.m

Function that classifies a submitted column vector (or matrix of column vectors) using the correlation coefficient method. It also allows for an option to plot the correlation with all classes using Matlab's `pcolor.m`. The default option is not to plot. The correlation coefficients are calculated using Matlab's function `corrcoef`. The call has two basic forms.The first includes the option to plot, the second does not.

```
classid=corrtest(testmtrx,[CA22mcov.mu],[1:1400],'y');
classid=corrtest(testmtrx,[CA22mcov.mu],[1:1400]);
```

The output is a row vector for each test vector listing the vector number, the max correlation coefficient value, actual classification, and associated symmetry pair class.

Note the square brackets around `CA22mcov.mu`. This is because the function is written to use a standard matrix of sample averages. The square brackets return the actual contents of the data structure cell `mu`. The vector `[1:1400]` is the full window of the vectors and applies to both `testmtrx` and `[CA22mcov.mu]`. This feature allows for future window reduction, if that becomes advantageous.

eval_mle.m

Function that finds the set of 22 log-likelihood values for a submitted vector. It is called by `mletest.m`, which finds the max value of the vector returned by `eval_mle.m`. The call looks like

```
llhds=eval_mle(CA22mcov,testvec);
```

Note that `eval_mle.m` admits only one vector at a time.

gs2jab.m

Function that converts the integer class identification 1-22 into the JAB matrix identification. The inverse mapping function is `jab2gs.m`. The call looks like

```
[i,j] = gs2jab(k);
```

This is used in the field testing files to identify an associated symmetry class via two calls of the form

```
[iirow,iicol]=gs2jab(ii);
iisym=jab2gs(iirow,7-iicol);
```

where `ii` is an integer identification of a class.

jab2gs.m

Function that converts the JAB matrix identification into the integer class identification 1-22. The inverse mapping function is `gs2jab.m`. The call looks like

```
k = jab2gs(i,j);
```

This is used in the field testing files to identify an associated symmetry class via two calls of the form

```
[iirow,iicol]=gs2jab(ii);
iisym=jab2gs(iirow,7-iicol);
```

where `ii` is an integer identification of a class.

makeCAs.m

M-file that loads the A-data contained in `A11.mat` through `A46.mat` into the data structure `DS_basic`. This is done via `read_ds.m`. Creates the cell array `CA_basic` for the 22 classes from `DS_basic`. Calls the function `mleparam.m` to create the `CA22mcov` and `CA11mcov` data structures from DS_basic. Clears `DS_basic` to save memory.

mleparam.m

Function that creates the data structures `CA22mcov` and `CA11mcov` from `DS_basic`. Called by the M-file `makeCAs.m`. The call has two forms, which look like

```
C22mcov=mleparam(CA_basic,vecs);
C22mcov=mleparam(C_basic);
```

where `vecs` represents a subset of vectors to be used from `CA_basic`. This optional argument may be used if one wishes to hold out a selection of vectors to use for testing instead of training. We used all A-data to train our classifiers, hence `makeCAs.m` uses the second call.

mletest.m

Function that classifies a submitted column vector (or matrix of column vectors) using the MLLE method. The call looks like

```
classid=mletest(testvecs,CA22mcov);
```

The output is a row vector for each test vector listing the vector number, actual classification, and associated symmetry pair class.

pltnvecs.m

Function to plot any integer number of vectors on the same set of axes. The set of vectors must be presented as a matrix of column vectors. The function breaks the plot window into four consecutive plots over component windows of size 350 each. It will plot a subvector given as the argument `window`, but only if it is a single consecutive set of components. The function allows for entering a plot title as the argument `plttitle`. (Note: the text of `plttitle` must be entered within single quotes.) The function can be used to plot a single vector, all 22

average vectors, and a pair of vector averages from respective symmetry classes. The calls for each of these examples look like

```
pltnvecs(CA_basic1,4(:,1),[1,1400],'First Vector in A class')
pltnvecs([CA22mcov.mj],[1,1400],'Set of Class Averages')
pltnvecs([[CA22mcov(:,1).mu],[CA22mcov(:,19)]],[1,1400],...
          'A-S Symmetry Averages')
```

(Note the use of the ellipsis to continue a command onto another line. This is a standard Matlab convention.)

read_ds.m

M-file to recursively load the A-data contained in `A11.mat` through `A46.mat` into the data structure `DS_basic`. File is called by `makeCAs.m`.

symallav.m

Function to find the average symmetry metric values for each of the 22 classes. Calls the function `sympair.m`. The call looks like

```
results=symallav(CA_basic,[1:1400],CA22mcov);
```

The second argument allows for eventually reducing the necessary number of components for evaluation. However this feature has not been included in the function presented here. That will require constructing new parameters for the lower-dimension log-likelihood.

sympair.m

Function that tests all vectors in the indicated `ith` class against the set of classes in the "opposite symmetry half", using log-likelihood ratios. "opposite" is determined by the symmetry in the JAB matrix. The function tests both "left-to-right" and "right-to-left". It returns a vector of ratios of log-MLE values for the `ith` class vs log-MLE values for the symmetry class. Each test vector is given two ratios:

(`i` class)/(`i` symmetry class) and (`i` symmetry class)/(`i` class)

The function is called by `symallav.m` and has been used only for the A-data. The call looks like

```
llr=sympair(CA_basic,8,CA22mcov);
```

where 8 represents the integer identification for the H class.

A.3 Matlab Code

CA_b11.m

```
% Used by makeCAs.m in creating the data structure CA11mcov.
% Requires the file assem2.m and creation of DS_basic.
vecpairs=[1,19,2,20,3,21,4,22,5,16,6,17,7,...
          18,8,12,9,13,10,14,11,15];
CA_bas11=cell(11,4);
for j=1:length(vecpairs)/2
 CA_bas11(j,1)={[char(vecpairs(2*j-1)+64),char(vecpairs(2*j)+64)]};
 CA_bas11(j,2)={[vecpairs(2*j-1),vecpairs(2*j)]};
 CA_bas11(j,3)={[1:1400]};
end;
CA_bas11(:,4)=assem2(CA_bas11(:,2),CA_bas11(:,3),DS_basic);
clear j vecpairs
```

CA_b22.m

```
% Used by makeCAs.m to create CA_basic and CA22mcov.
% Requires the file assem2.m and creation of DS_basic.
CA_basic=cell(22,4);
for jj=1:22
  CA_basic(jj,1)={char(64+jj)};
  CA_basic(jj,2)={[jj]};
  CA_basic(jj,3)={[1:1400]};
end;
CA_basic(:,4)=assem2(CA_basic(:,2),CA_basic(:,3),DS_basic);
clear jj
```

assem2.m

```
function [cell_data] = assem2(cell_list,row_list,DS)
% Assembles the set of desired matrices (or submatrices) of vectors
% for all rows of the fourth cell of a cell array like CA_basic.
% Called by CAb_11.m and CAb_22.m.
for jj = 1:length(cell_list)
    cell_data{jj} = assemble(cell_list{jj},row_list{jj},DS);
end
% internal function
 function [accum_data] = assemble(ds_list,row_list,DS)
 accum_data = [];
 if isempty(row_list)
```

```
   for jj = 1:length(ds_list)
     accum_data= [accum_data  DS(ds_list(jj)).data];
   end
 else
   for jj=1:length(ds_list)
     accum_data=[accum_data DS(ds_list(jj)).data(row_list,:)];
   end;
 end;
```

citest.m

```
function results=citest(tstmtrx,Cmcov,ciwidth,window);
% Tests tstmtrx against averages in Cmcov using confidence width
% approach.
% Example:
%    results=citest(CA_basic{1,4},CA22mcov,ciwidth,[1:1400]);
[dum,nbrtest]=size(tstmtrx);
[dum,nbrcats]=size(Cmcov);
[obssize,dum]=size(Cmcov(1).mu);
results=[[1:nbrtest]',zeros(nbrtest,3)];
for m=1:nbrtest
   finalcnt=0;
   indicate=1;
   for j=1:nbrcats
      current=0;
      check=abs(tstmtrx(:,m)-Cmcov(j).mu);
      sgncheck=sign(check);
      current=obssize-length(find(sgncheck<0));
      if current>finalcnt
         finalcnt=current;
         indicate=j;
      end;
   end;
    results(m,2:3)=[finalcnt, indicate];
   [iirow,iicol]=gs2jab(indicate);
   results(m,4)=jab2gs(iirow,7-iicol);
end;
disp('Vector #, # hits, Class ID, Sym ID')
```

conwidth.m

```
function ciwidth=conwidth(Cmcov,nbrobs,t);
```

```
% Creates matrix of confidence interval widths, one for each class
% component average. Requires the Matlab stat function tcdf.m to
% determine confidence levels.
% Example:
%    ciwidth=conwidth(CA22mcov,30,t);
[rows,nclass]=size(Cmcov);
[ncomps,cols]=size(Cmcov(1).sigma);
t
format long
disp('Individual interval confidence level = ')
cilevel=tcdf(t,nbrobs-1)-tcdf(-t,nbrobs-1)
disp('Joint vector band confidence level = ');
cblevel=cilevel^(1400)
format short
ciwidth=zeros(ncomps,nclass);
for j=1:nclass
   varvec=Cmcov(j).sigma;
   ciwidth(:,j)=t*sqrt(varvec/nbrobs);
end
```

corrtest.m

```
function maxcorr=corrtest(testmat,avgmat,window,pltchar);
% Tests each column of testmat matrix against all columns of
% avgmat matrix for correlation and can plot using Matlab's
% pcolor.m.
% Example:
%   results=corrtest(CA_basic{1:3,4},[CA22mcov.mu],[1:1400],'y');
% Requires Matlab's corrcoef.m and pcolor.m.
if nargin<4
   pltchar='n'
end
[ncomps,navgs]=size(avgmat);
[dum,nobs]=size(testmat);
if pltchar=='y'
   cormat=zeros(nobs,navgs);
end
maxcorr=zeros(nobs,3);
for j=1:nobs
 corvec=zeros(1,navgs);
 for k=1:navgs
    Temp=corrcoef([testmat(window,j), avgmat(window,k)]);
```

```
    corvec(k)=Temp(2,1);
    if pltchar=='y'
       cormat(j,k)=corvec(k);
    end
 end;
 [maxval,maxind]=max(corvec);
 maxcorr(j,:)=[j,maxval,maxind];
end;
disp('Vector #, Max corr, Class ID, Sym ID')
% Plot routine
if pltchar=='y'
   cormat=[cormat,zeros(nobs,1);zeros(1,navgs+1)];
    % add row and column of zeros to get pcolor to plot everything.
    gridbase=1:nobs+1;
    [x,y]=meshgrid(gridbase,gridbase);
    subplot(1,1,1);
    pcolor(cormat);
    shading faceted;
    view([0,270]);
    title('Correlation Map');
    xlabel('min = blue < green < yellow < orange < red = max')
 end
```

eval_mle.m

```
function llhds=eval_mle(C,t);
% Computes the log-likelihoods for test vector t over all classes
% represented in data structure C.
% Example:
%   llhds=eval_mle(CA22mcov,CA_basic{1,4}(:,1));
for jj=1:length(C)
   llhds(jj)=mleval(C(jj).mu,C(jj).sigma,C(jj).con,t);
end
% internal function
function y=mleval(mu,sigma,con,testvec);
 y=-0.5*((testvec-mu)'*((1./sigma).*(testvec-mu))) + con;
```

gs2jab.m

```
function [i,j] = gs2jab(kk)
% Maps GS integer identification (1,..,22) to JAB (4 x 6) array
% identification. Inverse map is jab2gs.m
```

```
% Example: (i,j)=gs2jab(14)
%   Maps GS 14 (i.e., N class) to (3,4) element in JAB array.
if  kk > 15
   kk = kk + 2;
elseif kk > 4
   kk = kk + 1;
end
if (kk > 24 | kk < 1)
   i = []; j = []; % illegal k values
else
   i = 1 + mod(kk-1,4);
   j = (kk-i)/4 + 1;
end
```

jab2gs.m

```
function kk = jab2gs(i,j)
% Maps JAB [4 x 6] symmetry matrix identification to
% GS single integer identification. Inverse map is gs2jab.m
% Example: gsnumb=jab2gs(3,4)
%   Maps JAB 3rd row, 4th column to GS 14 (i.e., N class).
if  i==1 & (j==2 |  j==5)

   kk = [];
elseif  i > 4 | i < 1
   kk = [];
elseif  j > 6 | j < 1
   kk = [];
else
   kt = (j-1)*4 + i;
   if kt > 16
      kk = kt - 2;
   elseif kt > 4
      kk = kt -1;
   else
      kk = kt;
   end
end
```

makeCAs.m

```
% Creates cell arrays and data structure for A-data.
% Requires files read_ds.m, CA_b11.m, CA_b22.m, mleparam.m.
disp('Loading A-data into DS_basic')
 read_ds
[obssize,nbrobs]=size(DS_basic(1).data);
disp('Creating cell array CA_basic for the 22 classes');
 CA_b22;
disp('CA_basic created');
disp('Creating data structure CA22mcov');
 CA22mcov=mleparam(CA_basic);
 disp('CA22mcov created');
disp('Creating CA11mcov');
 CA_b11
 CA11mcov=mleparam(CA_bas11);
 disp('CA11mcov created');
clear DS_basic
disp('DS_basic cleared to save memory')
```

mleparam.m

```
function C=mleparam(Carray,J);
% Creates data structures of the form CA22mcov. Called
% by makeCAs.m.
[nc,dum]=size(Carray); % nc is number of classes
[nr,np]=size(Carray{1,4});
for jj=1:nc
  tmp=B{jj,4}(:,J);
  Cmcov(jj)=mcov(tmp);
end;
% internal function
function Cmcov=mcov(V)
% Construct average & covariance matrices, and vector of constants.
[u,v]=size(V);
Cmcov.mu=mean(V,2);
temp=V-Cmcov.mu*ones(1,v);
Cmcov.sigma=sum(temp.^2,2)/(v-1);
Cmcov.con=-0.5*(u*log(2*pi)+sum(log(C.sigma)));
```

mletest.m

```
function classid=mletest(testmtrx,Cmcov);
```

```
% Tests each column of testmtrx using MLLE.
% Example:
%   classid=mletest(testvecs,CA22mcov);
% Requires the function file eval_mle.
[nr,nc]=size(testmtrx);
classid=[[1:nc]',zeros(nc,2)];
for jj=1:nc
   llhds=eval_mle(Cmcov,testmtrx(:,jj));
   [maxval,maxind]=max(llhds);
   classid(jj,2)=maxind;
   [iirow,iicol]=gs2jab(classid(jj,2));
   classid(jj,3)=jab2gs(iirow,7-iicol);
end
disp('Vector #, Class ID, Sym ID')
```

pltnvecs.m

```
function y=pltnvecs(vecmtrx,window,plttitle)
% Plots n vectors on same axes over indicated window of independent
% values. Plotting format is subplot(4,1,x), with each plot window
% covering 350 vector components.
% Example:
%    pltnvecs(CA_basic{1,4},[1,1400],'Class A');
%       plots the 30 vectors from A-data, class A.
nbrplots=4;
hlength=ceil((window(2)-window(1))/350);
lows=[1,351,701,1051];
highs=[350,700,1050,1400];
c=min(min(vecmtrx));
d=max(max(vecmtrx));
border=.1*(d-c);
 c=c-border;
 d=d+border;
for h=1:hlength-1
 a=lows(h);
 b=highs(h);
 v=[a,b,c,d];
 subplot(nbrplots,1,h);
 x=a:b;
 plot(x,vecmtrx(a:b,:));
 axis(v);
 if h==1
```

```
  title(plttitle);
 end;
end;
subplot(nbrplots,1,hlength);
a=lows(hlength);
b=highs(hlength);
v=[a,b,c,d];
x=a:window(2);
plot(x,vecmtrx(a:window(2),:));
axis(v);
```

read_ds.m

```
% Reads in the A-data (.mat files) and assembles the data structure
% DS_basic. A-data files have form Aij.mat, where i, j are integers.
for kk = 1:22
   [i, j] = gs2jab(kk);
   eval(['load A',int2str(i),int2str(j),'.mat']);
   DS_basic(kk).data = eval(['a',int2str(i),int2str(j),'''']);
   DS_basic(kk).array_label = [i, j];
     eval(['clear a',int2str(i),int2str(j)]);
end
clear i j kk
```

symallav.m

```
function allratios=symallav(CA,window,C_mle);
% Finds average log-ratio for all classes.
% Example:
%    allratios=symallav(CA_basic,[1:1400],CA22mcov);
% Requires the file sympair.m.
[nc,dum]=size(CA);
allratios=zeros(nc,2);
for ii=1:nc
   allratios(ii,:)=mean(sympair(CA,ii,C_mle));
end
```

sympair.m

```
function [llr]=sympair(CA,ii,C_mle);
% Tests vectors in class ii using "left-to-right" scheme and ratios
% from MLLE symmetry metric.
```

```
% Example:
%    llr=sympair(CA_basic,9,CA22mcov);
% Requires the function files eval_mle, gs2jab.m, jab2gs.m, mleparam.m.
[iirow,iicol]=gs2jab(ii);
iisym=jab2gs(iirow,7-iicol);
[nr,nv]=size(CA{ii,4});
llr=zeros(nv,2);
for jj=1:nv
   llhds=eval_mle(C_mle,CA{ii,4}(:,jj));
   llr(jj,1)=llhds(ii)/llhds(iisym);
end
llr(:,2)=1./llr(:,1);
disp(['Average log ratios = [',num2str(mean(llr)),']'])
```

Appendix B

Assorted Plots

The following three plots may help to visualize the difficulties faced in this project. Each plot is composed of a consecutive series of four plots. These four are necessary to cover the full range of a single vector and still allow reasonable differentiation among component results.

- Figure B.1 is an example of the frequency response vectors returned by the probing device. In particular, this is the first vector in the A class from the A-data. It is the first column of the matrix contained in the first row, fourth column cell of `CA_basic`. The plot was constructed with the command

  ```
  pltnvecs(CA_basic{1,4}(:,1),[1,1400],'')
  ```

- Figure B.2 shows all 22 class averages which were constructed using the A-data and the file `makeCAs.m`. The brackets `[ ]` change the `mu` field of `CA22mcov` into a matrix. The plot was constructed with the command

  ```
  pltnvecs([CA22mcov.mu],[1,1400],'')
  ```

- Figure B.3 is an example of two class averages that are symmetry pairs. These two are from the A-S pair. The vectors are the first and nineteenth columns of the matrix contained in the `mu` field of the data structure `CA22mcov`. Putting brackets `[ ]` around `CA22mcov(:,1).mu` creates the one-dimensional matrix (a vector) for the A-class average. The double brackets create a 1400×2 matrix of the two vectors. The plot was constructed with the command

  ```
  pltnvecs([[CA22mcov(:,1).mu],[CA22mcov(:,19).mu]],[1,1400],'')
  ```

Figure B.1: Sample Frequency Response Vector

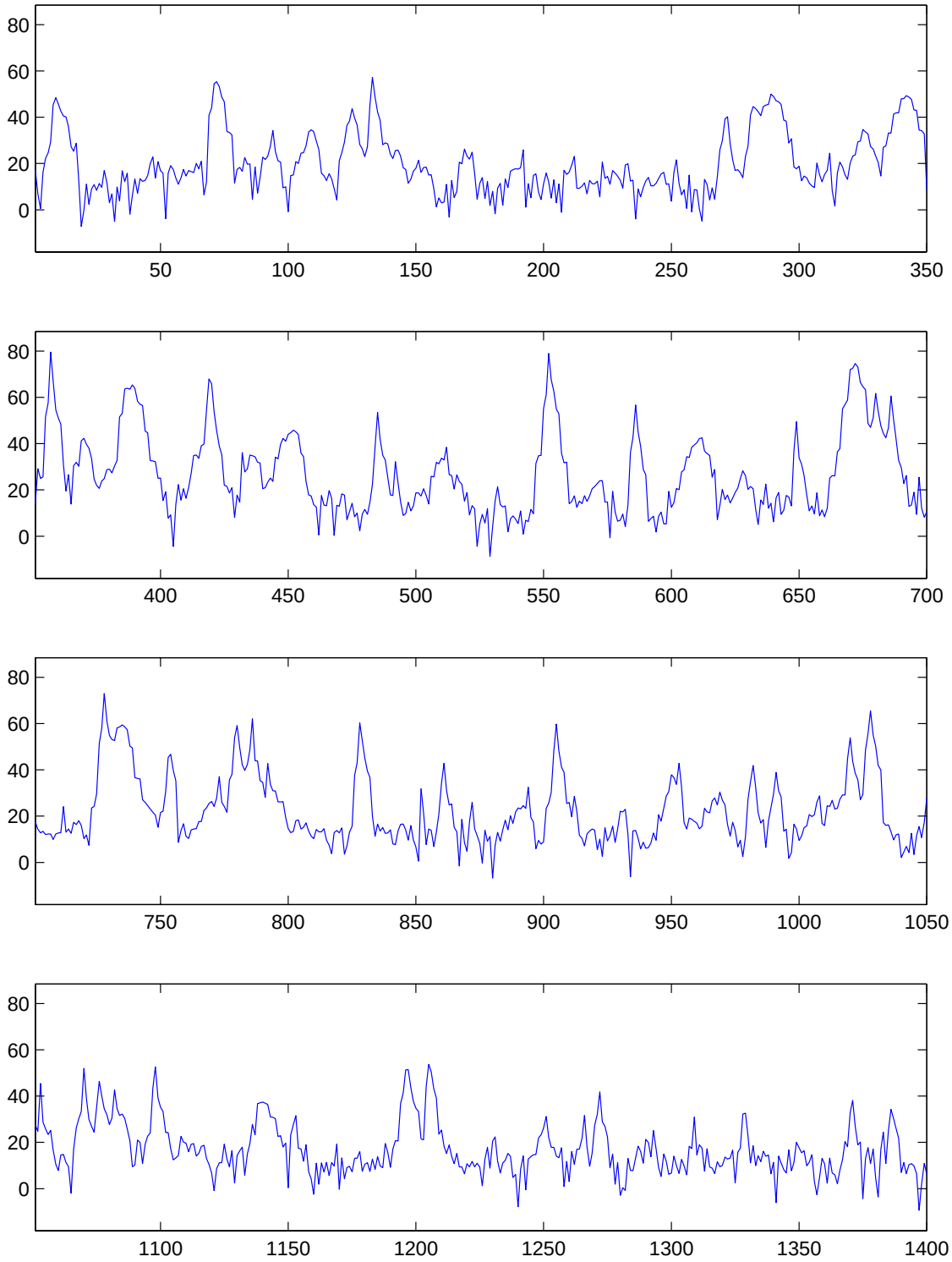


Figure B.2: The 22 Class Average Vectors

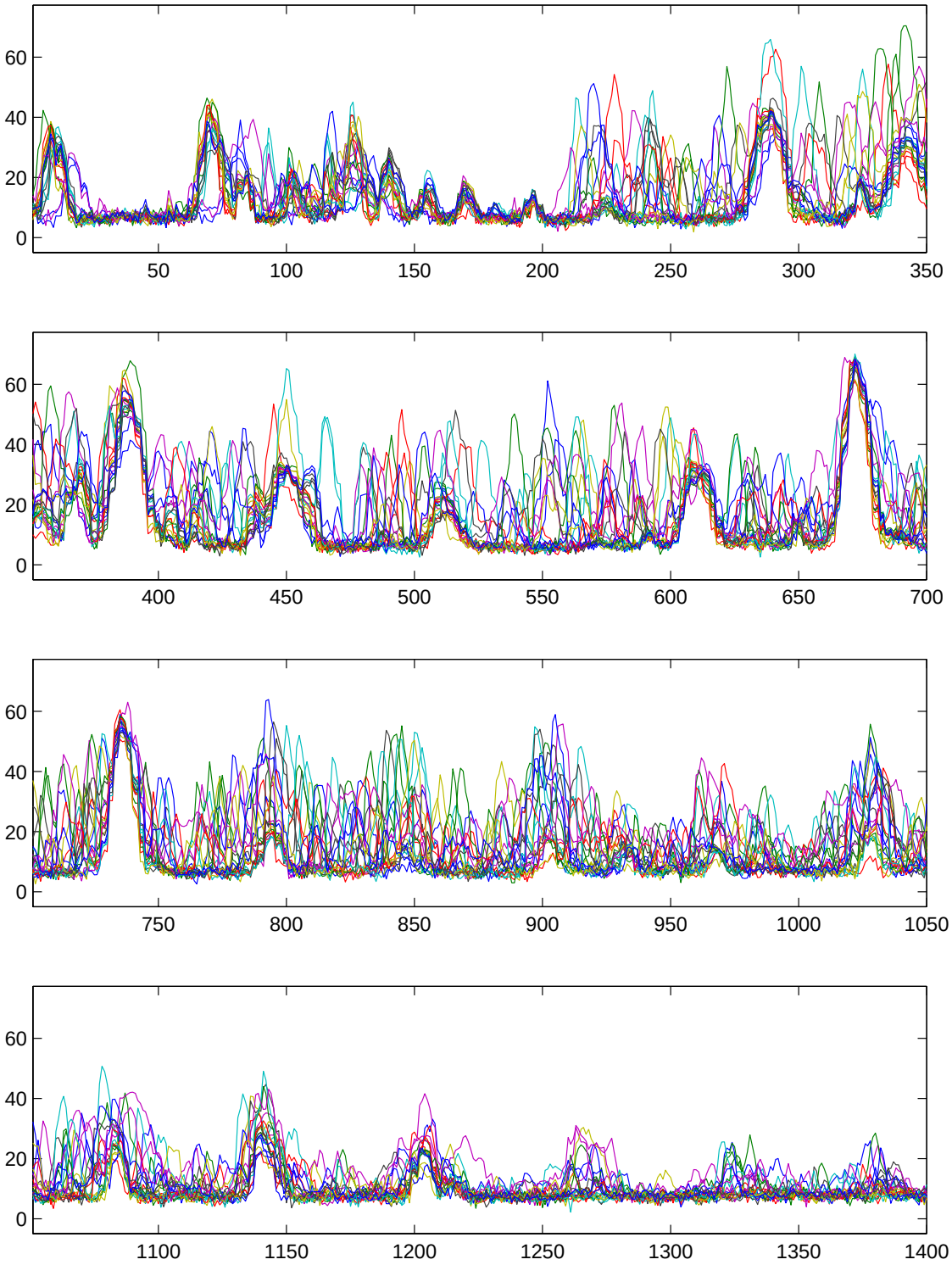
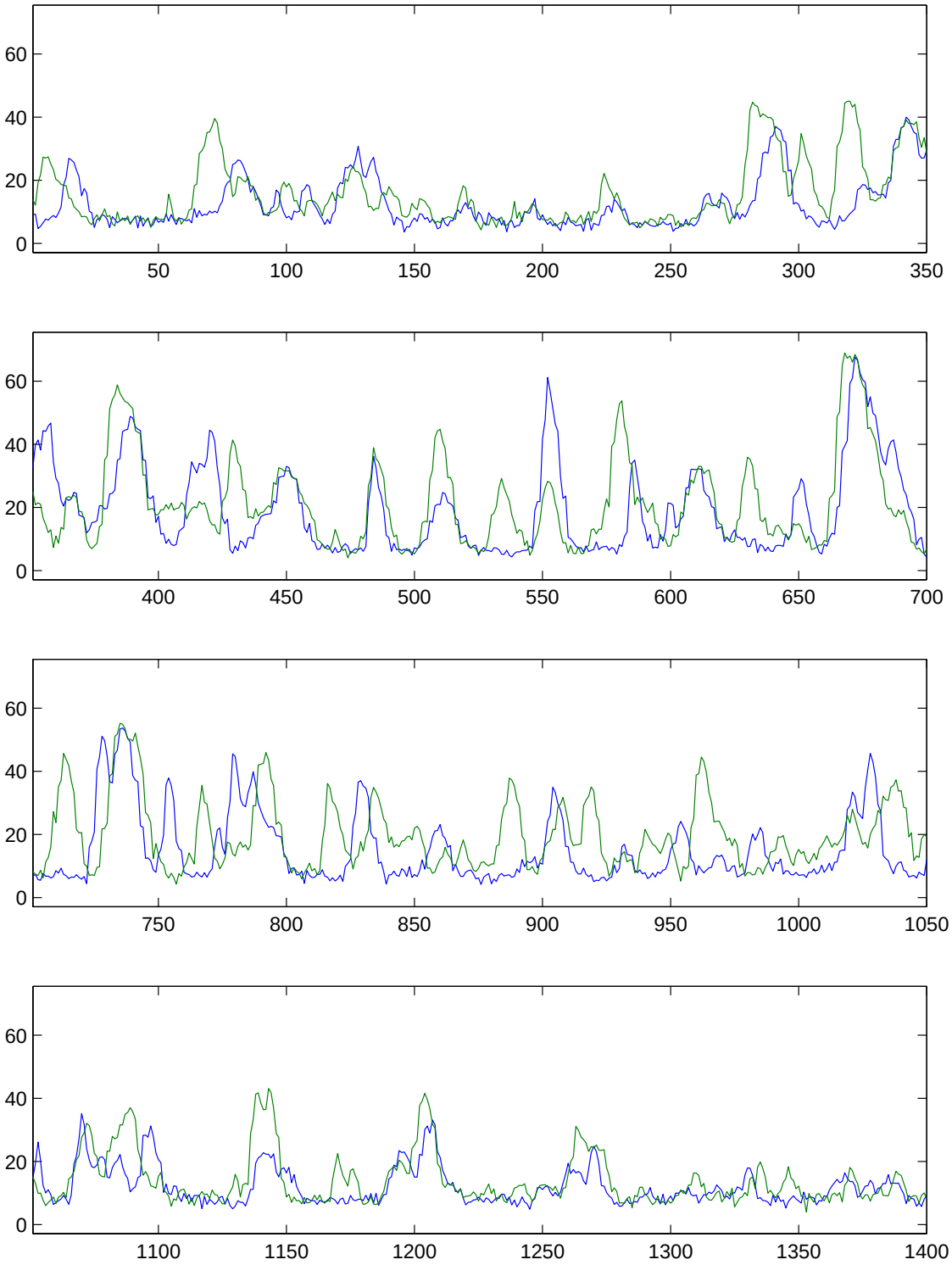


Figure B.3: Sample of Symmetry Pair Average Vectors



Bibliography

[1] K. E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley & Sons, New York, NY, 1978.

[2] D. J. Hand. *Discrimination and Classification.* John Wiley & Sons, New York, NY, 1981.

JOHN M. McGRAW
Department of Statistics
Virginia Tech
Blacksburg, VA 24061-0439
(540) 231-7667 Fax: (540) 231-7079
email: jmcgraw@icam.vt.edu
http://www.icam.vt.edu/Students/jmcgraw

Education

M.S., Statistics, Virginia Tech, Blacksburg, VA, December 2001
Thesis: An Investigation into Classification of High Dimensional Frequency Data
Major Professor: Dr. Eric P. Smith
M.S., Mathematics, Oregon State University, Corvallis, OR, June 1995.
B.S., Mathematics, Cum Laude, Albertson College of Idaho, Caldwell, ID, June 1991.

Professional Experience

Graduate Research Assistant/Graduate Teaching Assistant
Department of Statistics, Virginia Tech: August 2000-Present
Consultant on various projects. (Research Assistantship funded via Air Force Office of Scientific Research grant administered through the Interdisciplinary Center for Applied Mathematics, Virginia Tech.) Undergraduate teaching.

Instructor
Department of Mathematics, Virginia Tech: August 1998-July 2000
Undergraduate teaching and curriculum development. Creation and supervision of course web pages/online materials. (See Publications/Materials, below.)

Instructor/Graduate Teaching Assistant
Department of Mathematical Sciences, Clemson University: August 1997-May 1998
Undergraduate teaching and curriculum development.

Graduate Teaching Assistant
Department of Mathematics, Oregon State University: September 1992-June 1996
Undergraduate teaching and curriculum development. Creation and publication of calculus study guide used by all first-year calculus students. (See Publications/Materials, below.)

Records Clerk
Department of Health and Welfare, State of Idaho: April 1992-August 1992
Recording and accounting duties for Child Support Enforcement Division.

Graduate Teaching Assistant
Department of Mathematics, Oregon State University: September 1991-December 1991.
Undergraduate teaching.

Technical Experience

Microsoft Office Suite, HTML, LaTeX, Matlab, SAS, Mathematica, UNIX, Linux, Windows, MAC/OS, C, Pascal

Academic Honors and Awards

William F. Burger Teaching Award, Department of Mathematics, Oregon State University, 1994-95

Willis Fry Science Scholarship, Albertson College of Idaho, 1990-91

Publications/Materials

MTH 251 – Differential Calculus. A study guide to accompany *Calculus of a Single Variable*, Dick and Patton (1994). Used by all first-year engineering calculus students at Oregon State University.

MTH 252 – Integral Calculus. A study guide to accompany *Calculus of a Single Variable*, Dick and Patton (1994). Used by all first-year engineering calculus students at Oregon State University.

McGraw's MATH 2214 Resource Page. A web page of materials for Virginia Tech's Ordinary Differential Equations course. Items included links to my M-files, a 28 page manual to use MATLAB with ODEs, a complete set of lecture notes and associated presentation materials. All were designed for on-line use inside and outside the classroom.

Academic Contributions

Invited Talks

25th Annual South African Symposium on Numerical and Applied Mathematics (SANUM), University of Stellenbosch, Stellenbosch, South Africa, April 2001

Book Reviews

Thomas Branson, *Differential Equations for Engineers*: Addison Wesley Longman.
W.H. Chung and J.L. Speyer, *Stochastic Processes, Estimation, and Control*: Society of Industrial and Applied Mathematics.

Academic Service

Mathematics Learning Center tutor, Department of Mathematics, Oregon State University, 1992-1996

Panel of Judges, State Finals, MathCounts Middle School Mathematics Competition, Oregon State University, July 1994

Subcommittee to organize 15th Oregon Invitational Mathematics Tournament for High School Math Students, Oregon State University, May 1995

Math Emporium Supervisor, Virginia Tech, August-December 1998

Ad-Hoc Committee on Math 2214 Revision, Department of Mathematics, Virginia Tech, August 1998-July 2000

Statistical Consulting Center, Department of Statistics, Virginia Tech, August 2000-Present

Professional Affiliations

Society for Industrial and Applied Mathematics

Interdisciplinary Center for Applied Mathematics, Virginia Tech