EXPLORATION OF MULTIVARIATE CHEMICAL DATA IN NOISY ENVIRONMENTS: NEW ALGORITHMS AND SIMULATION METHODS

by

Stephen P. Driscoll

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
December 2019

"Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise" – John W. Tukey.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With high-dimensional measurements becoming increasingly common in chemistry, the efficient extraction of meaningful information from chemical data has never been more important. Chemometrics, a sub-discipline of analytical chemistry, emerged from the need for more advanced multivariate data analysis methods capable of solving more complex chemical problems. The goal of chemometrics can simply be stated as the differentiation between chemical variance and the variance due to measurement error. All analytical measurements are subject to errors, sometimes called noise, that contribute uncertainty to any type of analysis. The current state of the literature lacks both realistic noise simulation in the evaluation of new algorithms, as well as approachable methods to perform such noise simulation. Chapters 2, 3, and 4 of this thesis address these shortcomings. Chapters 2 and 3 describe a simple method for simulating realistic analytical measurement errors while Chapter 4 describes a method for accommodating different error structures in the analysis of fused multivariate data, an advance that circumvents the need for complicated preprocessing of these increasingly common data structures.

Although many advances have been made in developing new algorithms that provide meaningful results when exploring modern chemical data sets, variance-based methods, such as principal component analysis (PCA), still dominate the field. A promising alternative algorithm that is not based on variance is projection pursuit analysis (PPA). However, due to the nature of the ordinary PPA algorithm, it requires the use of PCA when there are many response variables with respect to samples, which is the case in most multivariate chemical data sets. Chapter 5 and 6 address this issue by proposing a sparse PPA algorithm that is independent of PCA and is shown to reveal meaningful results where PCA and ordinary PPA cannot. Another issue with ordinary PPA is that it performs poorly when applied to unbalanced data sets or data sets with a number of classes not equal to a power of 2. Chapter 7 addresses this issue by implementing an augmentation strategy that allows for the analysis of unbalanced data and the sequential extraction of clusters with projection pursuit.

LIST OF ABBREVIATIONS USED

Abbreviation	Description
ALS	Alternating Least Squares
AR	AutoRegressive
AsLS	Asymmetric Least Squares
augPPA	augmented Projection Pursuit Analysis
DFT	Discrete Fourier Transform
ECM	Error Covariance Matrix
EDA	Exploratory Data Analysis
FA	Fatty Acid
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FT	Fourier Transform
FTIR	Fourier-Transform Infrared
GA	Genetic Algorithm
HCA	Hiearchial Cluster Analysis
HPLC	High-Performance Liquid Chromatography
iid	Independent and Identically Distributed
IIR	Infinite Impulse Response
IR	Infrared
kPPA	Kurtosis-based Projection Pursuit Analysis
LDA	Linear Discriminate Analysis
MAD	Mean Absolute Deviation
MCR	Multivariate Curve Resolution
MCR-ALS	Multivariate Curve Resolution-Alternating Least Squares
MCR-WALS	Multivariate Curve Resolution-Weighted Alternating Least Squares
MIR	Mid-Infrared
MLPCA	Maximum Likelihood Principal Component Analysis
MS	Mass Spectrometry
NIR	Near Infrared

Abbreviation	Description
NMR	Nuclear Magnetic Resonance
PAF	Principal Axis Factoring
PC	Principal Component
PI	Projection Index
PLS	Partial Least Squares
PP	Projection Pursuit
PPA	Projection Pursuit Analysis
PSD	Power Spectral Density
RSD	Relative Standard Deviation
SD	Standard Deviation
seqPPA	Sequential Projection Pursuit Analysis
SPPA	Sparse Projection Pursuit Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machines
vis-NIR	Visible-Near Infrared

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Peter Wentzell for his supervision and guidance on our projects. The work in this thesis could not of been completed without his guidance and sharp insight. I would like to thank my immediate family, Gail Driscoll, Robert Driscoll, and Paul Driscoll for their love, support, and understanding over the past 3 and a half years. I would like to thank Dr. Roderick Chisholm, Dr. Jennifer MacDonald and Dr. Peter Wentzell for allowing me to teach over the required number of hours for a PhD student, as this was essential in my development as a scientist.

I would like to thank the graduate students that were here during my PhD, many of whom are now great friends. I would like to specifically thank Dr. Fabien Lindeperg and Dr. Colin Kelly for many nights of CCR and games while we lived together. This decompression time was invaluable over the course of this degree.

I would like to thank Dr. Jean Burnell, Dr. Peng Zhang, Dr. Erin Johnson, Dr. Fabrice Berrue, and Dr. Peter Wentzell for allowing me to research topics that are not contained in this thesis, but have led to published work.

I would like to thank my committee members Dr. Alan Doucette, Dr. Michael Dowd and Dr. Erin Johnson for providing helpful guidance and insight on my preliminary exam and various research projects.

CHAPTER 1

INTRODUCTION

With an increase in data complexity, mostly driven by advancements in instrumentation, analytical chemistry has evolved over the past few decades into much more of an information science. Intelligent instruments that provide high-dimensional data in the form of matrices and vectors are now the norm.[1] This evolution in the field, and the parallel increase in computational power, spawned a new sub-field of analytical chemistry called chemometrics.[2] Chemometrics, a term coined by Svante Wold in a grant application in the 1970s, seeks to improve the efficient extraction of chemical information from data.[3] Tools developed to address multivariate problems in other fields, such as principal component analysis (PCA),[4] hierarchical cluster analysis (HCA),[5] and partial least squares (PLS) regression,[6] are now at the forefront of data analysis in chemistry. However, there is a demand for new analysis tools that are useful for investigating the wide array of evolving problems in many analytical applications and techniques (*e.g.* metabolomics,[7] hyperspectral imaging,[8,9] environmental analysis[10]).

Most often, the roots of the data analysis tools used in chemometrics come from techniques founded in other scientific disciplines (*e.g.* deep learning,[11] support vector machines (SVM)[12]), or the adaption of a previous method that provides improved analysis results. Regardless of a method's origin, it is of the utmost importance to understand how the method operates and compares to others in the field. A key component in understanding the fundamentals of an algorithm is the simulation of realistic data and their errors. This not only benefits the researcher employing the method, but also the scientific community, as new insights can be made which can lead to improved methods or the development of superior alternatives.

This chapter serves to present the reader with the background required to appreciate the results of the remaining chapters. We will begin with a discussion of multivariate chemical data and the relevant tools used for its exploration. Following this, a discussion of algorithm development and the important role of simulation is presented. The chapter concludes with an illustrative example to unify the central themes of the thesis and a discussion of the key topics that serve to motivate the problems addressed in the work presented in the remaining chapters.

1.1 Multivariate Chemical Data

The measurement tools used by early researchers in the field of analytical chemistry produced predominantly univariate data, acquiring a single scalar measurement on a system. For example, techniques based on weighing, titrating, and single channel spectroscopy are all univariate measurements which were used to investigate various problems of interest during that time.[13] In contrast to univariate measurements, modern analytical techniques are mainly multivariate in nature, such as high-performance liquid chromatography (HPLC), mass spectrometry (MS), nuclear magnetic resonance (NMR) spectroscopy, and infrared (IR) spectroscopy.[14] Each of these multivariate measurement techniques measure a response as a function of multiple variables (retention time for HPLC, wavelength for IR, chemical shift for NMR, and mass-to-charge ratio for MS). In turn, this produces multivariate data in the form of vectors, matrices, and in some cases, such as hyphenated methods,[15–18] higher-rank tensors (multiway data). Typically, when more than one sample is analyzed using a multivariate technique, the data are organized such that one dimension corresponds to the various samples and the other corresponds to the response variables common for all samples. For example, if multiple spectra were obtained for different mixtures, one could construct a matrix where each row corresponds to a different sample and each column corresponds to a specific wavelength channel. For multiway data that involve additional variables (*e.g.* time), additional dimensions are typically added to form higher-order tensors (cubes or hyper-cubes).

Although the structure of multivariate data is more complex than univariate data, it undoubtedly has the opportunity to contain more information about the measured system.[19] Not only does multivariate data contain information from each channel individually, but it also contains information about the relationships among the channels. This allows for the

extraction of information that cannot be obtained through any of the individual variables, but is expressed in a combination of the individual variables. These combinations of observable variables cannot be measured directly and, therefore, are referred to as latent variables. However, since modern data sets contain thousands or even millions of response variables, finding the relationships among variables is practically impossible without implementing a set of rules that are applied to the data in attempt of uncovering useful information. These rules, or collection of rules, define the data processing methods or algorithms. These methods usually rely on principles of multivariate statistics and the act of applying such methods is called multivariate analysis.[20]

1.2 Exploratory Data Analysis

One of the most popular approaches for analyzing multivariate data is "exploratory data analysis", a phrase originally coined by John W. Tukey. The idea of exploratory data analysis is to approach a data set with an open mind and an adaptability, where the goal is to find information in the data set that may reveal a hypothesis worthy of further investigation. This is the opposite of confirmatory data analysis, where one uses traditional statistical concepts such as confidence, inference and significance to accept or reject an already defined hypothesis. Tukey describes this idea by stating "finding the question is often more important than finding the answer".[21] However, he was not proposing that exploratory data analysis replace confirmatory data analysis, merely that exploratory data analysis is an additional approach an analyst can use in attempt to reveal information in a complex data set.

A major element of exploratory data analysis is data visualization.[22] Representing high-dimensional data in one, two, or three dimensions allows the analyst to visualize relationships among objects (*e.g.* samples from an experiment) in a lower-dimensional space. This can be interpreted as mapping the high-dimensional data to a lower-dimensional space. Of course, there are many ways to define this mapping function. One method that is ubiquitous across multiple scientific disciplines is principal component analysis (PCA).[4,23,24] Within the fields of chemistry, a wide variety of visualization methods have been employed to examine the relationships among objects (*e.g.* hierarchical clustering, self-organizing maps), but PCA is unequivocally the most widely applied, with applications including chemotaxonomy, medical diagnostics, forensics, metabolomics and proteomics.

Exploratory data analysis is usually performed with *unsupervised* data analysis methods, which is in direct contrast with another class of analysis algorithms called *supervised* data analysis methods.[25] Unsupervised methods, sometimes referred to as unsupervised learning methods or algorithms, attempt to extract useful information from data without utilizing any input about the information they are attempting to discover (*e.g.* sample class, chemical concentration). Because of this, unsupervised methods are less vulnerable to bias introduced by prior knowledge within the algorithm and reveal natural structures in data. For example, suppose there is a set of analyzed tissue samples that were obtained from both normal and diseased patients. This is a two-class data set with respect to the normal and diseased states. Although this class information would not be used by an unsupervised algorithm, the results may reveal that the data naturally separate into two clusters in the low-dimensional space obtained by the algorithm. By labeling the classes in this space using the class information it is possible to reveal that this analysis method is able to discriminate between the diseased and normal patients using only the information from the measurements. For this reason, unsupervised exploratory methods have become *de facto* methods for evaluating hypotheses for multivariate data sets.

In contrast, supervised methods, which include supervised classification (*e.g.* linear discriminate analysis (LDA)) and multivariate calibration (*e.g.* partial least squares regression (PLS)), provide the algorithm with the target information (*e.g.* object class, chemical concentration). The algorithm builds a model to predict this information, but because the number of variables often greatly exceeds the numbers of objects (samples), such models can be prone to bias, referred to as "overfitting". This means that, while the model may be effective for data in the data set used to train it (the training set), it will fail when applied to new samples. From a classification perspective, this means that supervised methods may give a satisfactory visual representation of classes for the training set, but fail to correctly classify future samples because the model is based on spurious correlations. Although supervised methods are potentially more powerful than unsupervised methods, they require careful validation (external data, cross-validation, etc.) and experimental design to ensure the validity of the results.[26–31] Inadequate validation is often the source of poor reproducibility in reported results.

The clear advantage of using unsupervised methods is twofold. First, the requirement for validation is less strict since no class information is used to obtain the low-dimensional

representation of the data and, second, it is possible to find relationships in the data that were unknown to the analyst at the beginning of the experiment. The principal disadvantage is that, without any additional information, they are more restricted in their ability to model the data, which highlights the need for more powerful exploratory tools to be developed.

1.3 Algorithm Development and Evaluation through Simulation

Developing new data analysis algorithms, or modifying existing algorithms, is a major research area for many in the chemometrics community. Computer simulations are a key element in many scientific disciplines and algorithm development/evaluation in chemometrics is no exception. The two main components in simulating any multivariate data set are: 1) the pure data in the absence of any measurement artifacts or noise, which is defined by some parameterized system model, and 2) measurement artifacts (nonlinearities, channel shifts, etc.) and random noise (offset, drift, measurement noise, etc.) that obscure the target information in the data. Both of these components contribute to the total variance in the data set. Simulation studies allow researchers to evaluate the performance of an algorithm as a function of the parameters used to simulate the data set. For new data analysis algorithms, simulations applied during the development process help to confirm the expected behaviour of an algorithm in specific, well-controlled environments and guide improvement of the algorithm.

Since in any simulation the "true" values describing the data are known, it is trivial to evaluate the performance of an algorithm in terms of obtaining the "true" result. However, for such an evaluation to be useful, it is crucial that the simulation is representative of the experiment. If not, the evaluation provides no new practical insight into the behaviour of the algorithm. This is especially true for noise simulation, where the structure of the errors being added to a simulated data set are critical. This has been an area overlooked by the chemometrics community, as most simulated error in the literature is represented by statistical characteristics that are rarely observed in experiments, such as white noise (see for example references 32–34). There are several possible reasons for this. The most obvious is that white noise, consisting of independent random variations with a normal distribution and uniform variance, is easy to parameterize and simulate. A second reason is that many algorithms are designed with the assumption of white noise, so extending

this to the simulation seems natural, even if it does not reflect reality. A third factor is that measurement errors in most measurement systems are really well-characterized. Even when such information is available, methods to simulate complex error structures can be challenging. Regardless of the reason, it is clear that there is a need for an approachable method for realistic analytical measurement noise simulation for use in the evaluation and development of multivariate data analysis algorithms.

1.4 Thesis Goals

It is the goal of this thesis to explore and expand on two main facets concerning the differentiation between chemical variance and noise variance in multivariate chemical measurements: 1) the simulation of realistic measurement noise to aid the development and evaluation of multivariate chemical data analysis algorithms and 2) the development of alternative projection methods for the exploratory analysis of high-dimensional chemical data. The remaining sections of this introductory chapter define the mathematical structures and techniques that are the foundation of the work presented later in this thesis. We begin with the fundamental concept of measurement error.

1.5 Measurement Errors

Measurement error, e, can be defined as the difference between a measured value y and the "true" value μ,

$$e = y - \mu. \tag{1.1}$$

This definition is central to any measurement science and is the foundation of all data processing algorithms. In practice, μ is not generally known, so the exact value of e for an individual measurement cannot be determined. However, the statistical behaviour of the measurement error can be characterized by performing replicate measurements. This is commonly achieved in practice by taking the mean of the replicate measurements as an approximation of μ and then using the difference from the mean to estimate the dispersion of e. An important consideration in this calculation, however, is how a replicate is defined, since this can greatly influence the distribution of measurement errors.

In discussing the different types of replication it is helpful to consider a hypothetical experiment. Suppose a technician is to prepare a solution and measure its absorbance at a certain wavelength to determine the concentration of an analyte. In addition, the technician is told to perform replicate measurements. Without being told any additional information, there are several possibilities as to what defines a replicate in this scenario. At the lowest level, simply scanning the same sample many times would be defined as an *instrumental replicate*, which would give insight to the variability of the measurement introduced by the instrument itself (*e.g.* detector noise). Another possibility would be to perform replicate scans, but with removal and replacement of the sample cell between measurements. This is what is referred to as a *replacement replicate*. To investigate the error associated with the certain steps of the sample preparation, replicates of specific procedural steps could be performed. This is an example of a *technical replicate*. Other types of replicates are also possible, but are not discussed here. The purpose of this example is to introduce the importance of the definition of a replicate and the sources of variation included, as this has a direct consequence on the characterization of what is being defined as error. Once this definition has been established, the statistical characteristics of the measurement can be calculated.

The population variance, σ^2, of the measurement errors is the expectation value, denoted as E($\bullet$), of the squared error, or the average squared deviation from the mean taken over N replicates as N goes to infinity,

$$\sigma^2 = \mathrm{E}(e^2) = \mathrm{E}[(y-\mu)^2] = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2. \tag{1.2}$$

In practice, the sample variance s^2 is calculated from a limited number of replicate measurements using the sample mean $\bar{y}$ as an approximation to the true mean μ,

$$\sigma^2 \approx s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i-\bar{y})^2. \tag{1.3}$$

The sample variance and standard deviation are the most widely used measures of error dispersion, although the mean absolute deviation (MAD) is sometimes used as a more robust measure. The variance and standard deviation are routinely used in calculations involving confidence intervals, test statistics, and figures of merit.

While the sample variance is useful for univariate measurements, modern analytical

instruments generate vectors of data (*e.g.* spectra, chromatograms) for which the sample variance is insufficient to describe the measurement errors. For this reason, it is more important for chemists to discuss multivariate measurement errors and their statistical properties.

1.5.1 Multivariate Measurement Errors

Multivariate measurement error is often synonymous with measurement "noise", implying a sequence of errors that are potentially related. In analytical chemistry, it is easy to understand the idea of measurement noise by considering an experiment in which replicate spectra are obtained for a sample. For each individual wavelength channel, one can attribute univariate statistical properties associated with the error. To organize this information, it is useful to define the multivariate measurement error vector, $\mathbf{e}$, that has length equal to the number of measurement channels, n, and describes the individual error at each channel of the system,

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix}. \tag{1.4}$$

However, unlike the univariate case, the multivariate system has an additional statistical property, called the error covariance, that defines the relationship between the measurement error at one channel with that at the other channels. For any two variables, x and y, the error covariance σ_{xy} describes the statistical relationship between the errors in x and y,

$$\sigma_x^2 = \mathrm{E}(e_x^2), \quad \sigma_y^2 = \mathrm{E}(e_y^2), \quad \sigma_{xy} = \mathrm{E}(e_x \cdot e_y). \tag{1.5}$$

Similar to the sample error variance, the sample error covariance can be defined by including the estimations on e_x and e_y with a limited number of replicates N,

$$\sigma_{xy} \approx s_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}). \tag{1.6}$$

The error variance and covariance are important in characterizing multivariate errors in chemical systems. For example, if the system of interest is a function of wavelength

channels, then it is important to compare the error variances of each channel to determine if the variance is uniform. If the variances are the same, the errors are said to be *homoscedastic*. If the variances are non-uniform, then the errors are *heteroscedastic*. Although this information is important, it gives no indication of the correlation among the errors. The error covariance determines whether the errors are independent or related to one another. If the error covariances within a group of variables, or channels, is zero, then errors are said to be *independent*. Otherwise, if the covariances are non-zero, then the errors are said to be *correlated*. This type of error characterization becomes important when applying current data processing procedures as well as developing new data processing methods.

It is in the transition from univariate to multivariate measurements in chemistry that the field of chemometrics was spawned. With vectors and matrices being the norm, the goal of chemometrics can simply be stated as differentiating the meaningful chemical variance from other sources of variance (including measurement noise and chemical noise) to provide the maximum extraction of chemical information from data. This can be done most effectively when the characteristics of the measurement variance are well known. Methods for characterizing and describing the properties of multivariate measurement errors are presented in the following section.

1.5.2 Characterization of Multivariate Measurement Errors

The term "measurement noise" is often taken to be synonymous with "measurement error", but carries with it a suggestion of an ordered sequence, or vector, of errors, as well as an association with purely instrumental sources of analytical error, as opposed to, for instance, errors in sample preparation. The concept of a noise vector is particularly important when the analytical measurement is multivariate in nature, such as a vector representing a spectrum or chromatogram. In such cases, the descriptions of the error must go beyond a point description of the variance; a full characterization of the error must describe how the variance changes from one channel to the next (*i.e.* its heteroscedasticity), as well as any correlations that exist among the errors at different channels. Two approaches are commonly used to characterize measurement error vectors: the power spectral density (PSD) function (based on the Fourier transform)[35] and the error covariance matrix (ECM),[36] calculated on the basis of replicated experiments. The approaches are complementary, although the ECM generally provides a more complete description of the errors.

1.5.2.1 The Error Covariance Matrix

To obtain all combinations of covariance terms for a system of n variables, the expectation of the outer product of the measurement error vector $\mathbf{e}$ with itself gives rise to the ECM $\mathbf{\Sigma}$,

$$\mathbf{\Sigma} = \mathrm{E}(\mathbf{e} \cdot \mathbf{e}^{\mathrm{T}}) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \dots & \sigma_{1n} \\ \sigma_{12} & \sigma_2^2 & \sigma_{23} & \dots & \sigma_{2n} \\ \sigma_{13} & \sigma_{23} & \sigma_3^2 & \dots & \sigma_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{1n} & \sigma_{2n} & \sigma_{3n} & \dots & \sigma_n^2 \end{bmatrix}. \tag{1.7}$$

This formulation generates a $n \times n$ square symmetric matrix with diagonal elements, $\sigma_{j=k}$, containing the variance at each channel and the off-diagonal elements, $\sigma_{j \neq k}$, containing the covariance of the measurement errors at channels j and k. For example, if a system contains only uncorrelated errors, then $\mathbf{\Sigma}$ is a diagonal matrix. The diagonal of the ECM provides direct information on the scedasticity of the errors; if the values of $\sigma_{j=k}$ are significantly different, then the errors are considered to be heteroscedastic. Otherwise, if the values along the diagonal are uniform, then the error structure is said to be homoscedastic. The ECM is sensitive to many properties of noise (*e.g* independence, correlation, scedasticity, power law dependence) which makes it a good tool for identifying and characterizing noise in analytical measurements.

An alternative way of representing the error structure is the error correlation matrix. With the same dimensions as the ECM, the error correlation matrix has elements ρ_{jk} given by

$$\rho_{jk} = \frac{\sigma_{jk}}{\sigma_j \sigma_k}. \tag{1.8}$$

This produces a matrix where all of the diagonal elements are equal to 1 and all of the off-diagonal elements have values that can range from -1 to 1. The error correlation matrix provides information that may be qualitatively hidden in the ECM due to scale. In practice, it is recommended to investigate both the error covariance matrix, which gives information regarding the magnitude of the errors and their relationships, as well as the error correlation matrix, which shows these relationships independent of scale.

1.5.3 The Fourier Transform

Another common method for characterizing multivariate measurement errors is Fourier analysis. Most modern day instruments use discretely sampled values of ordered variables as part of their measurement procedure, which makes it possible to discuss the frequency profile of the measurement error vector. Although traditional Fourier analysis is usually applied to systems for which the ordinal variable is time, it is more common in chemical data to have other ordinal variables, such as wavelength or chemical shift. In this case, the other domain will be referred to as the Fourier domain as it is not technically frequency. In practice, the discrete Fourier transform (DFT) of a vector $\mathbf{x}$ that has length n can be calculated as follows:

$$y(k) = \sum_{j=1}^{n} x(j) W_n^{(j-1)(k-1)}. \tag{1.9}$$

From Equation 1.9, the inverse Fourier transform can be calculated as

$$x(j) = \frac{1}{n} \sum_{k=1}^{n} y(k) W_n^{-(j-1)(k-1)}, \tag{1.10}$$

where

$$W_n = e^{(-2\pi i)/n} \tag{1.11}$$

for both equations.

To obtain the two-sided amplitude spectrum from $\mathbf{y}$, one must take the magnitude of $\mathbf{y}$ (square root of the sum of the squared real and imaginary parts) divided by n. Since this contains both the positive and negative side of the spectrum, which will be symmetric for real-valued measurements, it is common to only look at the one-sided spectrum; in this case half of the two-sided spectrum must be multiplied by 2 to obtain the correct amplitude. This analysis procedure is shown visually in Figure 1.1. In modern software, the DFT is calculated using fast Fourier transform (FFT) algorithms. The most popular of these algorithms is the Cooley-Tukey algorithm.[37]

It is not uncommon for measurement errors in chemistry to have various types of noise that can be identified using Fourier analysis, although research in this area with respect to analytical measurements is sparse at best. Perhaps the most well-known type of noise is white noise. White noise gets its name from its behaviour in the frequency domain.

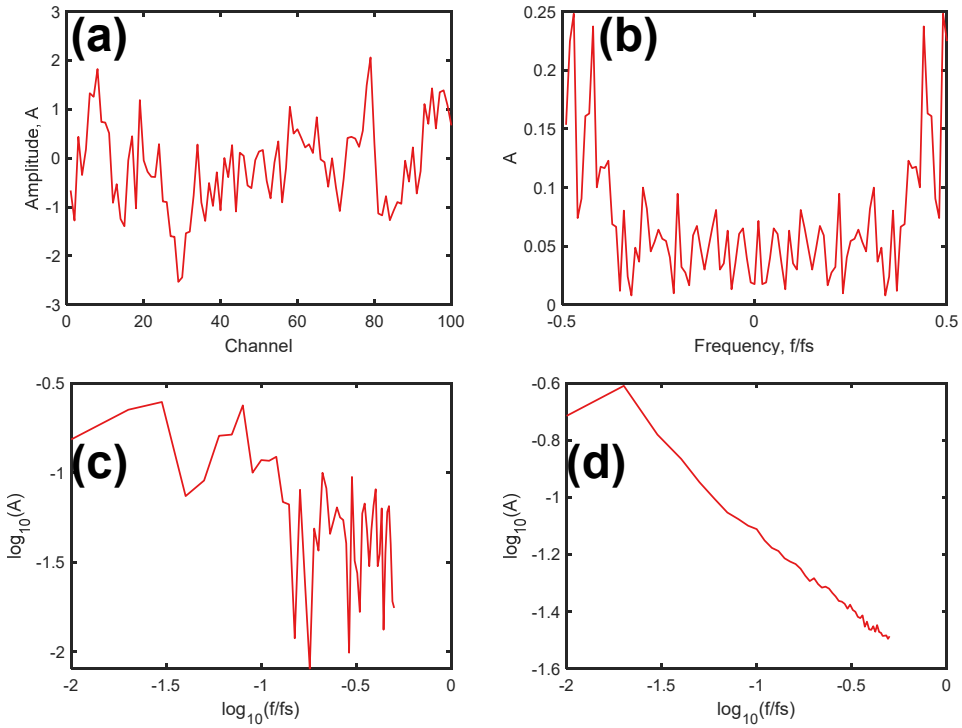

Figure 1.1: (a) A pink noise signal 100 channels in length. (b) The two-sided amplitude spectrum resulting from the Fourier transform of the signal in (a). (c) One-sided amplitude shown on a log-log plot. (d) The same plot as (c) except from the average of 1000 realizations of the pink noise signal amplitude spectrum.

With equal amplitude at all frequencies it is analogous to white light. Other types of noise, such as interference noise and power law noise, are often referred to as "coloured noise" to differentiate them from white noise. Power law noise has an amplitude that depends on the frequency in the Fourier domain as $1/f^{\alpha}$ where α is related to the noise type. Power law dependent noise such as pink noise and brown noise are also observed in analytical measurements. When investigating possible power law dependent noise, it is often convention to use a log-log plot of the the amplitude squared, or power, as a function of frequency. The slope of the line from the $\log$(power) vs $\log$(frequency), or power spectral density (PSD), plot is a way to characterize the noise. White noise, as mentioned earlier, has a slope of zero and can be referred to as $1/f^0$ noise. Pink noise, or $1/f$ noise, has a slope of -1 and brown noise has a slope of -2 (commonly called $1/f^2$ noise or random-walk noise).

It is important to use many replicates when using Fourier analysis as a diagnostic tool for measurement errors. For example, trying to determine if a signal is white (flat amplitude

vs frequency plot), or pink (sloped amplitude vs frequency plot), can be difficult with only a few replicates. As shown in Figure 1.1(c), looking at only one amplitude spectrum of pink noise, for example, it is hard to determine if there is a non-zero slope, but if many amplitude spectra are averaged, the resulting spectrum will approach a profile consistent with a pink noise signal, as shown in Figure 1.1(d). For this reason, replicate samples of a noise sequence are collected if one wants to classify it using Fourier analysis. These types of noise, along with other common noise structures observed in analytical chemistry are discussed in the following section.

1.6 Common Multivariate Measurement Error Structures

Although complex, commonly observed measurement error structures can be classified in various ways. Two of the most important terms for classifying these commonly observed structures are *stationarity* and *independence*. In the context of analytical measurements, a noise signal is considered stationary if it only depends on the measurement channels, as opposed to, for example, the signal amplitude. A noise signal is considered independent if the values for all measurement channels are uncorrelated with one another. Within each of these classes there exists multiple examples of noise structures that are commonly observed in analytical chemistry. This section details these noise structures in terms of the ECM.

1.6.1 Stationary Independent Noise

Stationary independent noise is uncorrelated and only a function of the measurement channels. This is the most basic type of noise observed experimentally and can be identified by specific structural features of the ECM. Due to independence, this type of noise only impacts the diagonal of the ECM. Therefore, there are two possibilities, either all diagonal elements are equal (homoscedastic), or not (heteroscedastic).

1.6.1.1 Homoscedastic White Noise

Homoscedastic white noise is the simplest type of noise. It is observed as simple detector noise where all variables have the same measurement error variance drawn from the same distribution. If this distribution is Gaussian, then this type of noise is called *iid* normal

(independent and identically distributed with a normal distribution). Most data processing algorithms assume that the measurement error structure is *iid* normal because it is simple to model. Figure 1.2 shows an *iid* normal noise signal of length 100 and the corresponding average ECM from 1000 replicate signals.

1.6.1.2 Heteroscedastic White Noise

Heteroscedastic white noise is similar to *iid* noise in that they both are independent, but it differs due to the fact that it has a non-uniform variance across channels. In other words, this type of noise has an ECM that is diagonal, but contains different values. Although, as the variance is only dependent on each individual channel, this type of noise is still stationary and uncorrelated. Figure 1.3 shows a heteroscedastic white noise signal of length 100 and the corresponding average ECM of 1000 replicate signals.

As seen in Figure 1.2 and 1.3, the ECM from 1000 replicate signals still has a slightly noisy ECM. The reliability of experimental ECMs is dependent on how many replicate samples are analyzed. Only a few realizations of the noise vector, which is common in most experimental designs, makes it difficult to classify the ECM.

1.6.2 Stationary Correlated Noise

Stationary correlated noise is the simplest type of correlated noise. In this case, the ECM exhibits non-zero off-diagonal elements, but the error structure still only depends on the measurement channels. The most common types of stationary correlated noise in analytical chemistry are offset noise and power law, or $1/f^{\alpha}$, noise.

1.6.2.1 Offset Noise

Sometimes called additive noise, offset noise arises from the signal shifting up or down. Common sources of this type of error include changes in cell position between measurements or a slowly drifting detector baseline. This results in a random offset in the signal which corresponds to a flat non-zero ECM. To illustrate this, Figure 1.4 shows 10 Gaussian signals with a random baseline offset and the corresponding ECM.

1.6.2.2 $1/f^{\alpha}$ Noise

$1/f^{\alpha}$ noise, or source flicker noise, is a type of noise observed due to baseline variation introduced from the detector. The most common type of $1/f^{\alpha}$ in analytical measurements

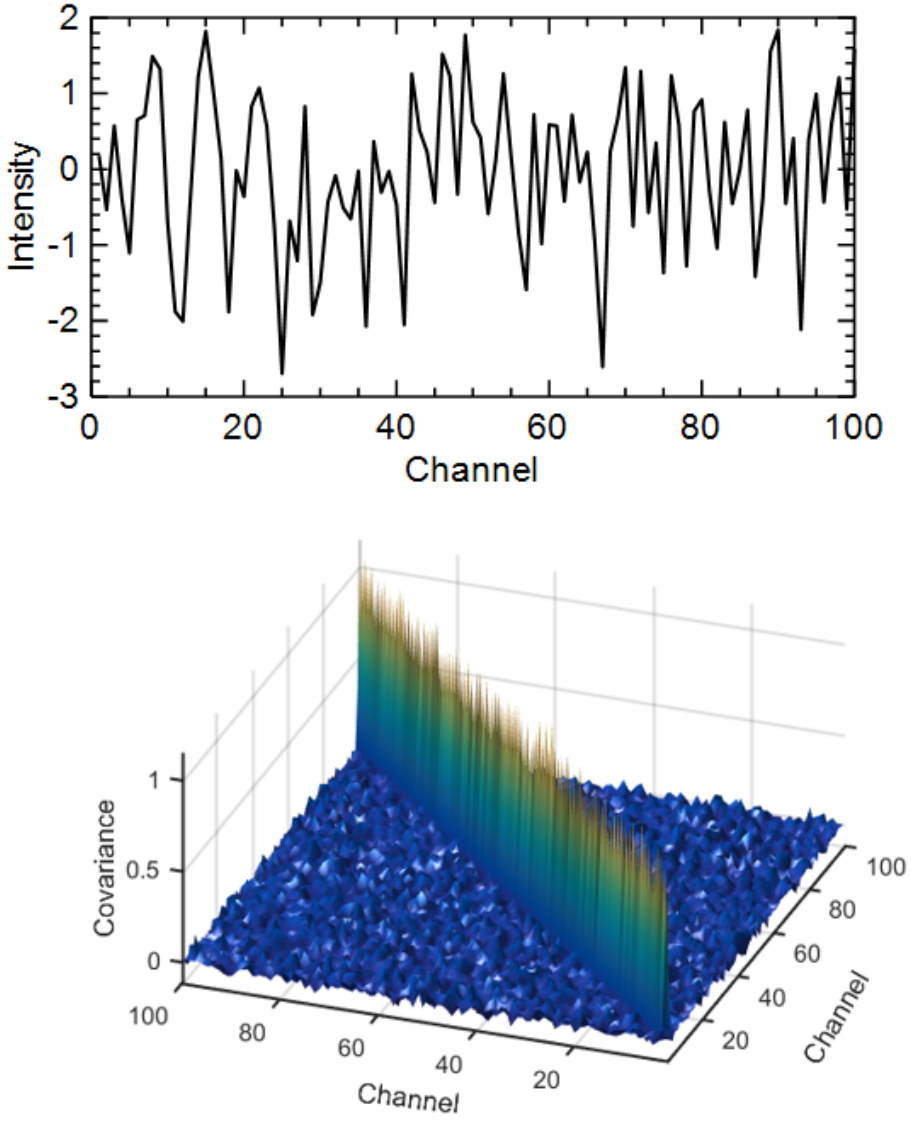

Figure 1.2: A homoscedastic white noise signal over 100 channels (top) and its ECM averaged over 1000 realizations (bottom).

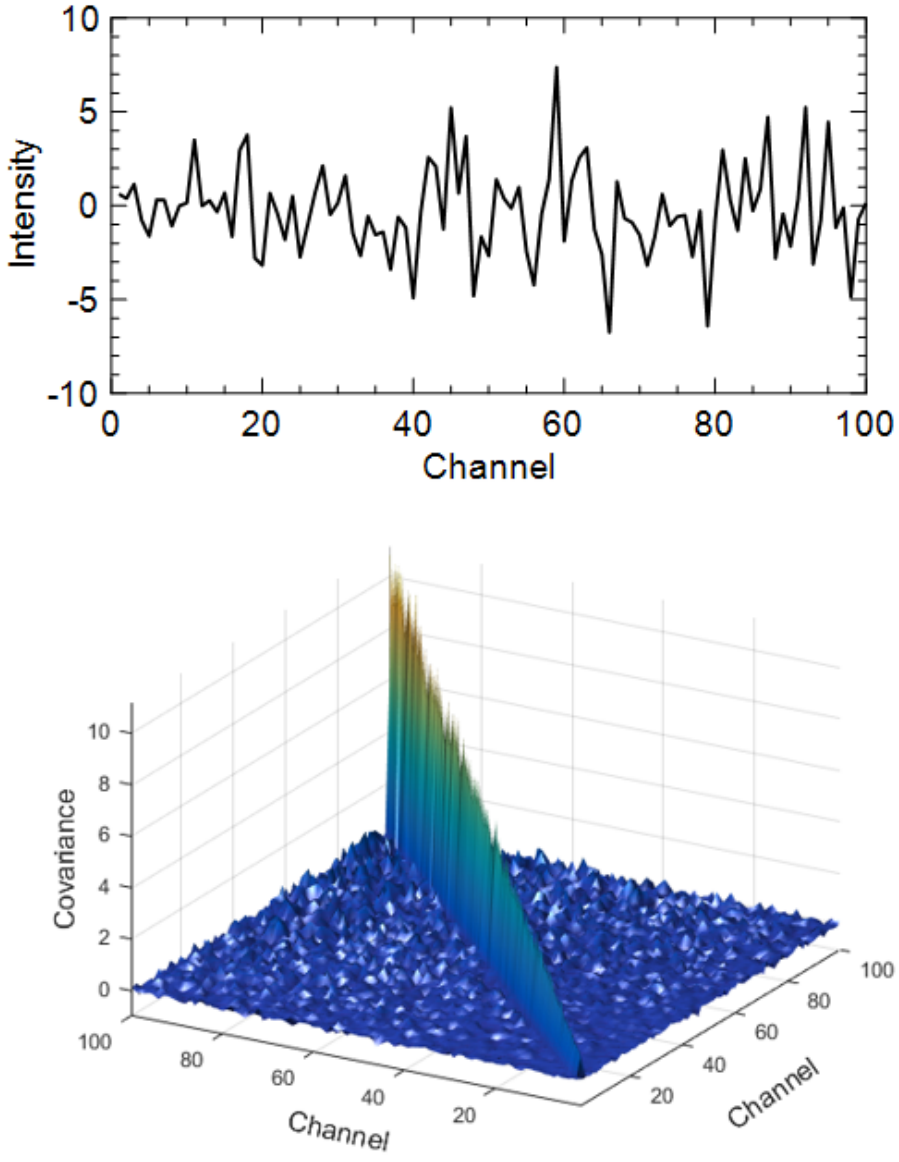

Figure 1.3: A heteroscedastic white noise signal over 100 channels (top) and its ECM averaged over 1000 realizations (bottom).

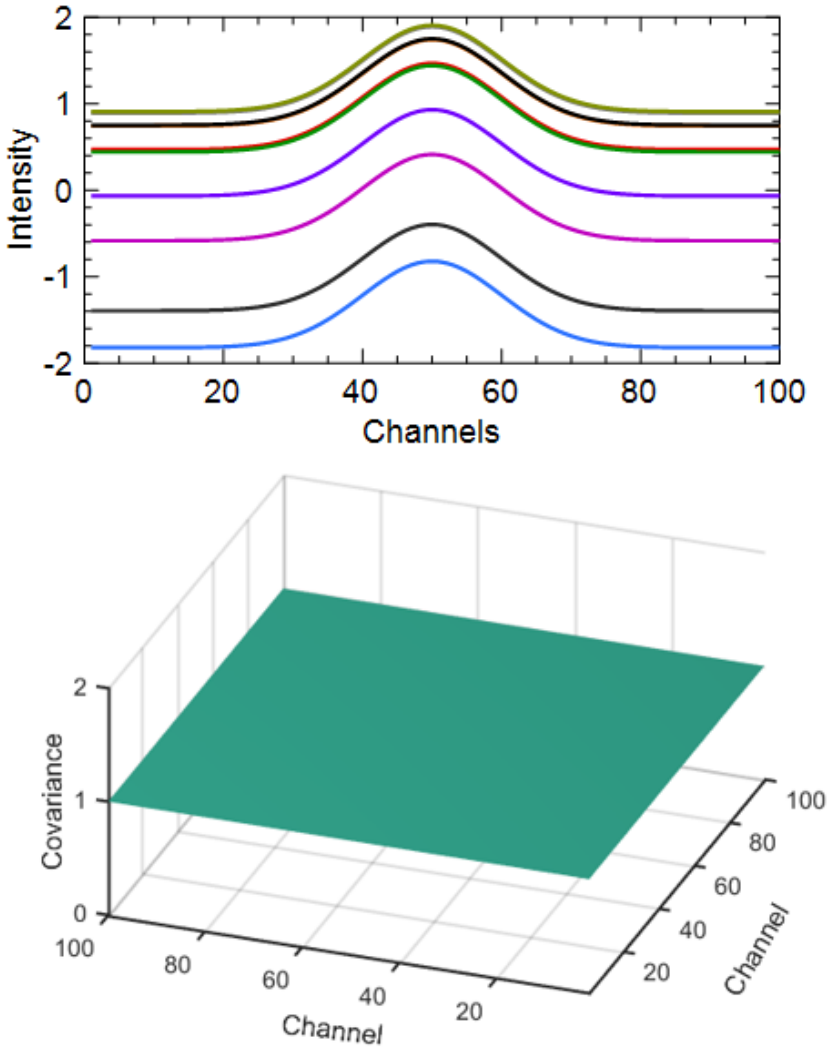

Figure 1.4: 10 random additive offsets of a Gaussian like signal over 100 channels (top) and the corresponding ECM (bottom).

is pink noise ($\alpha = 1$). For example, researchers have reported observing source flicker noise from vacuum tubes as early the 1920s.[38,39] This is the last of the common types of observed measurement noises to be studied in terms of its ECM and noise simulation for use in analytical chemistry. A long standing problem in analytical measurement noise simulation and modeling is the lack of accessible methods for simulating and modeling $1/f^{\alpha}$ noise that is relevant to analytical chemists. Chapter 2 of this thesis presents a newly proposed and accessible method for simulating $1/f^{\alpha}$ noise in analytical measurements.

1.6.3 Non-stationary Independent Noise

Non-stationary independent noise is a type of noise that only impacts the diagonal of the ECM, but in a way that the the profile of this diagonal is dependent on the signal. These types of errors are typically associated with the source (*e.g.* flame, ion source) and can be divided further into two sub-classes, proportional noise and shot noise.

1.6.3.1 Proportional Noise/Multiplicative Noise

Proportional noise is such that the standard deviation of the error is proportional to the signal intensity. Figure 1.5 shows a Gaussian signal with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) and 0.15 RSD proportional noise over 100 channels, and the corresponding ECM averaged over 1000 realizations.

1.6.3.2 Shot Noise

Shot noise is a type of noise where the variance of the error is proportional to the square root of the signal intensity. It arises from counting statistics at detectors that measure quantized signals in time (*e.g.* electron multipliers). These randomly arriving signals follow the Poisson distribution and result in the square root proportionality between the standard deviation of the error and the signal. Figure 1.6 shows a Gaussian signal with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) and 0.05 proportionality parameter (not technically RSD, as will be explained in later sections) shot noise, and the corresponding ECM averaged over 1000 realizations.

1.6.4 Non-stationary Correlated Noise

Perhaps the most problematic type of noise in terms of analysis, non-stationary correlated noise exhibits non-uniform correlated components that are dependent on the signal. This

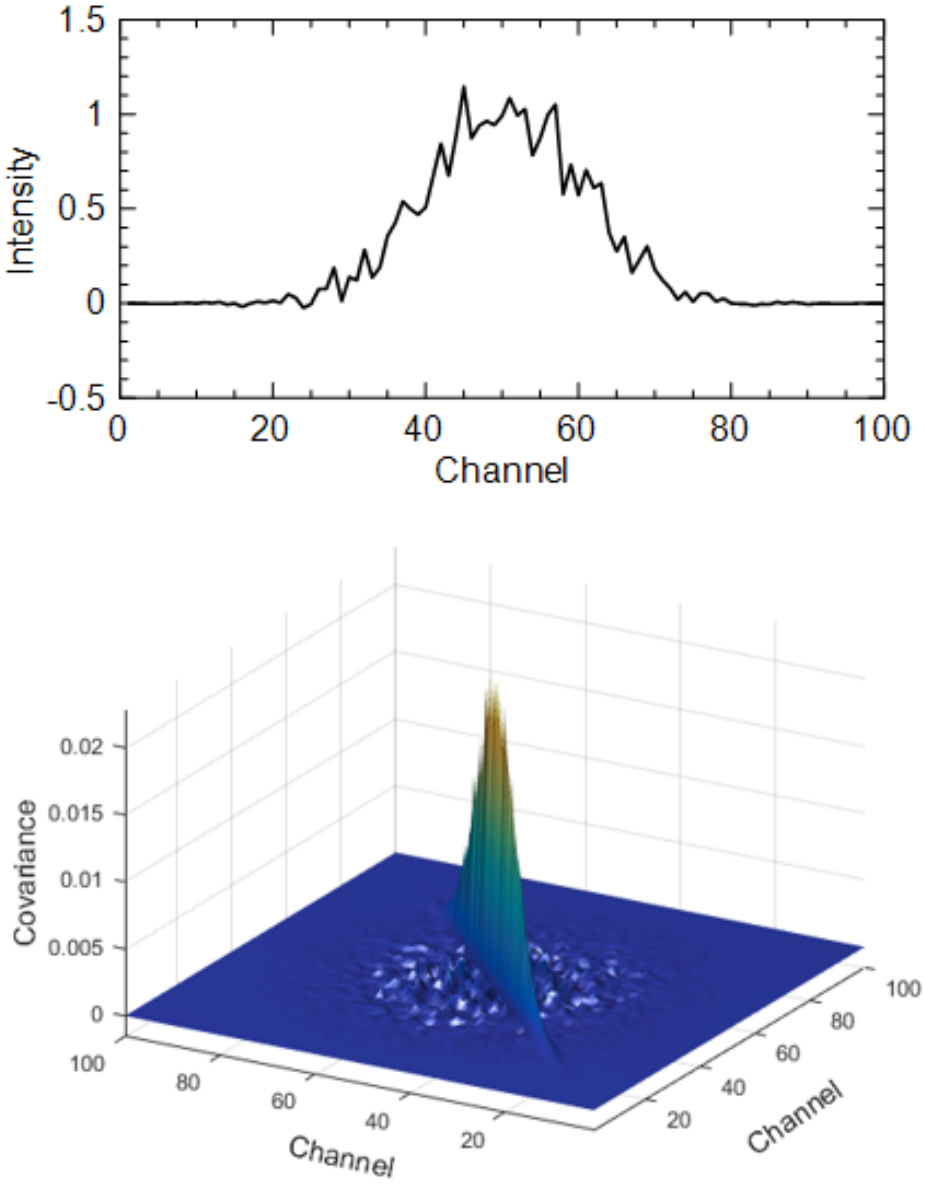

Figure 1.5: A Gaussian signal with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) and 0.15 RSD proportional noise over 100 channels (top), and the corresponding ECM averaged over 1000 realizations (bottom).

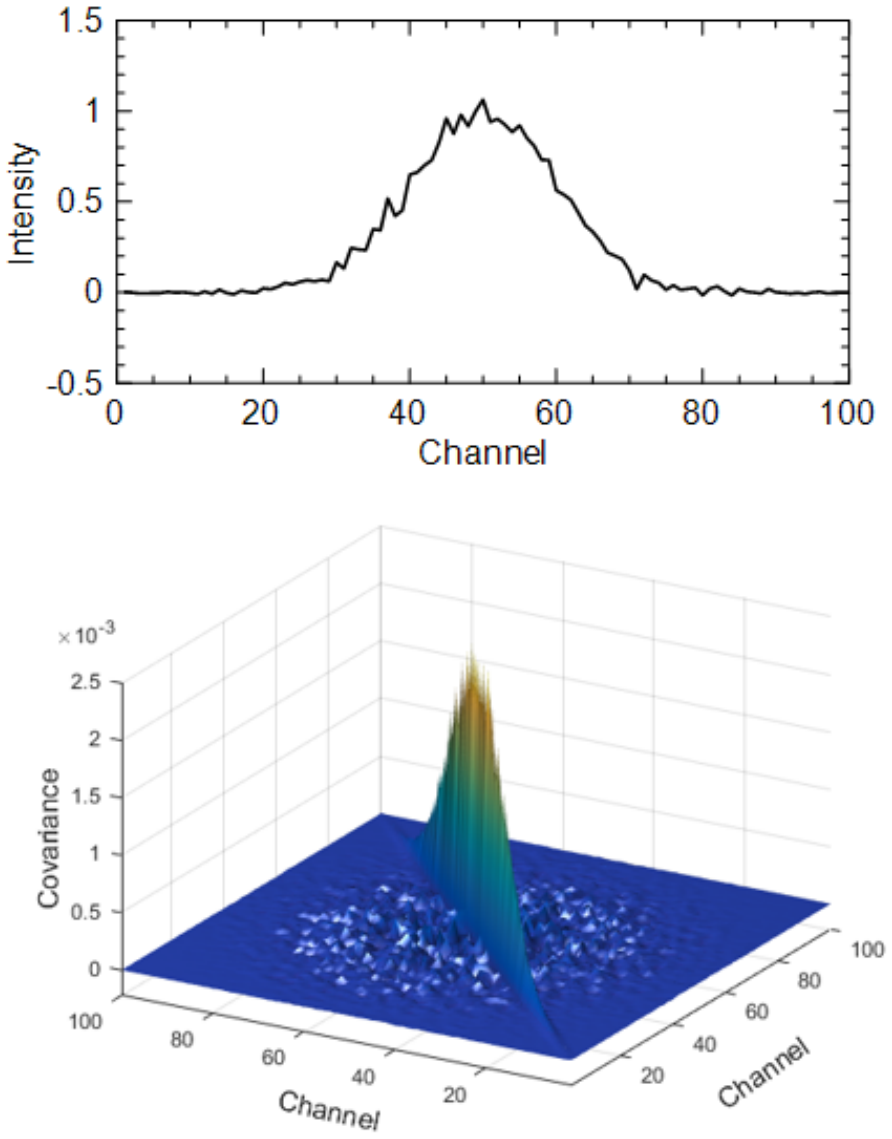

Figure 1.6: A Gaussian signal with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) and 0.05 RSD shot noise over 100 channels (top), and the corresponding ECM averaged over 1000 realizations (bottom).

type of noise is the largest deviation from the *iid* normal noise assumed by most data analysis algorithms. Unfortunately, it is somewhat common in analytical measurements. The two most dominant cases of non-stationary correlated noise in analytical chemistry are multiplicative offset noise and proportional $1/f^{\alpha}$ noise.

1.6.4.1 Multiplicative Offset Noise

Multiplicative offset noise is similar to the offset noise mentioned previously in that the impact of this noise arises from a shift of the signal up or down, but in this case the effect is multiplicative and not constant over all channels. This type of noise introduces an offset that is proportional to the signal. This is commonly seen in near-infrared (NIR) measurements from path length variation. Figure 1.7 shows 10 Gaussian signals originally with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) subjected to 0.5 RSD multiplicative offset noise, and the corresponding ECM.

1.6.4.2 Proportional $1/f^{\alpha}$

Proportional $1/f^{\alpha}$ noise is $1/f^{\alpha}$ noise that is only present when there is a signal. This can arise from variations in source intensity in spectrometers or ionization efficiency in mass spectrometers. Although this type of noise is speculated to be present in many analytical measurements, little research has been done on its modeling and simulation. More details on $1/f^{\alpha}$ noise, both stationary and non-stationary, will be presented in Chapter 2.

1.7 Projection Methods for Exploratory Data Analysis

The proceeding discussion of measurement noise characteristics is relevant to exploratory data analysis because the fundamental goal of chemometrics is to isolate the relevant variance due to chemical difference (*i.e.* the chemical information) from irrelevant sources of variance, including measurement noise. The development of data analysis tools for this purpose naturally imposes some assumptions regarding the noise to facilitate this task, and when these assumptions are violated, the methods can perform sub-optimally to a greater or lesser degree. Therefore, it is important to understand how different error structures might affect data analysis.

A main goal of exploratory analysis is to obtain a subspace of the original data where there is discrimination between classes of samples due to the chemical variance. There are many methods that can be used to perform exploratory data analysis. A common

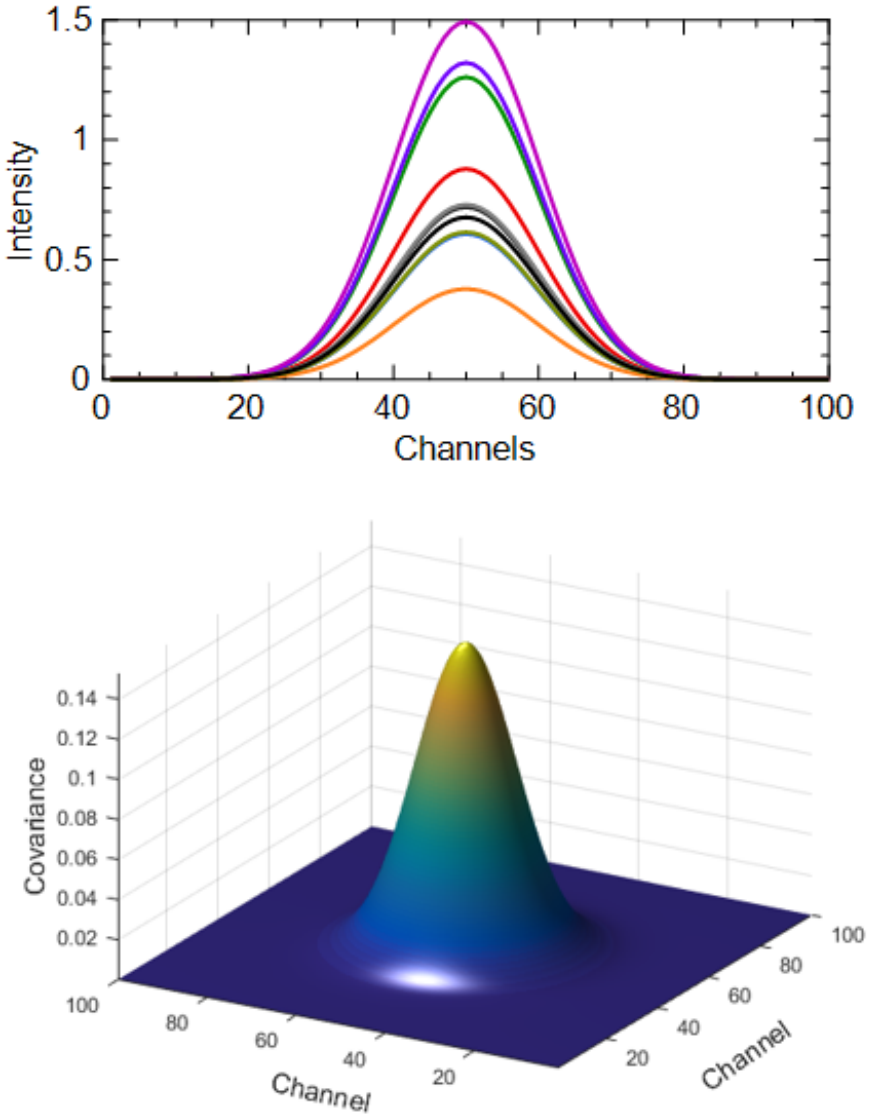

Figure 1.7: 10 Gaussian signals with a maximum intensity of 1 ($\mu = 50$, $\sigma = 10$) and multiplicative offset noise of 0.5 RSD, and the corresponding average ECM).

sub-class of these methods is linear projection methods. Projection methods attempt to project high-dimensional data into a low-dimensional space that can be visualized (1, 2, or 3 dimensions). These projections are subspaces of the original data and the idea is to capture the most relevant information in this subspace. The purpose of this section is not to give the reader a complete overview of projection methods, but to present the relevant projection methods commonly used in chemistry and related to the work presented in later chapters of this thesis. Two methods in particular are the focus of the background material presented here. The first is principal component analysis (PCA), which has already been mentioned as the workhorse of exploratory data analysis. The second method, projection pursuit analysis (PPA), has seen more limited application in chemistry due to algorithmic limitations, but these have been recently overcome with the introduction of kurtosis-based PPA using a quasi-power algorithm.[40] Extensions of this algorithm are a central theme of this thesis and are discussed in more detail in Chapter 5-7.

1.7.1 Principal Component Analysis

PCA is applied to multivariate chemical data for a variety of purposes.[4] Rather than changing information in the original data, its purpose is to perform an orthogonal rotation on the original variables to obtain a new set of measurements (the "scores") which are linear combinations of the original variables. Mathematically, PCA is usually implemented through singular value decomposition (SVD) to produce a scores matrix, $\mathbf{T}$, and a loadings matrix, $\mathbf{L}$, as shown in Equation 1.12,

$$\mathbf{X} = \mathbf{TL}. \tag{1.12}$$

If $\mathbf{X}$ is $m \times n$ (where $m \leq n$), $\mathbf{T}$ is $m \times m$ and $\mathbf{L}$ is $m \times n$. The rows of $\mathbf{L}$ are also called the eigenvectors, which describe the rotation of the original space. Row i of $\mathbf{T}$ contains the scores, which are the coordinates of the row i of $\mathbf{X}$ on the new space defined by $\mathbf{L}$.

The advantage of PCA comes from the truncation of the scores and loadings to represent the data in a space of lower dimensionality, p, where $p < m$. This is possible because the rotation of the original space is such that the first eigenvector accounts for the largest amount of variance in the data, the second eigenvector accounts for the largest proportion of the remaining variance, and so on. In principle, then, retaining the first p columns of $\mathbf{T}$ and the first p rows of $\mathbf{L}$ retains the maximum amount of information in a p-dimensional

subspace. This is represented as

$$\hat{\mathbf{X}}_p = \mathbf{T}_p\mathbf{L}_p, \tag{1.13}$$

where $\mathbf{T}_p$ ($m \times p$) and $\mathbf{L}_p$ ($p \times n$) are the truncated scores and loadings matrices.

The dimension reduction of PCA is exploited in several ways, but there are three main applications. In exploratory data analysis, the goal is to visualize the data in a space of lower dimensionality. In other cases, especially multivariate calibration and curve resolution, the goal is subspace estimation, where it is anticipated that the subspace contains all (or most) of the chemical variance. For example, if $\mathbf{X}$ contains spectra for mixtures of three chemical components (*i.e.* it has a so-called pseudorank or chemical rank of 3), it is anticipated that the eigenvectors for $p = 3$ will define the space containing the spectral vectors of the three components. These eigenvectors can then be used as a basis for developing a calibration model (multivariate calibration) or extracting the pure component spectra (curve resolution).

When measurement errors in $\mathbf{X}$ are *iid* normal and p is equal to the number of observable components present, PCA is the maximum likelihood estimation of the subspace (unbiased solution with minimum variance). However, when heteroscedastic and/or correlated errors are present, this is no longer true and the estimation of the subspace is sub-optimal. The extent to which results are affected depends on the characteristics of the measurement error. Erroneous estimation of the subspace affects all three applications (exploratory data analysis, multivariate calibration and curve resolution).

1.7.2 Maximum Likelihood Principal Component Analysis

As noted above, PCA assumes homoscedastic and uncorrelated errors for maximum likelihood subspace estimation. In chemical data, this error structure is rarely the case, resulting in a sub-optimal estimation of the chemical subspace by PCA. Maximum likelihood principal component analysis (MLPCA) was proposed to address this issue.[36,41,42] MLPCA incorporates knowledge of the error structure in the data when estimating the chemical subspace. The specific objective function used to perform the optimization, which is analogous to weighted least squares in univariate regression where a weighted sum of squared residuals is minimized, depends on the structure of the measurement errors. This results in improved subspace estimation when an accurate estimate of the error structure

and the rank of the data are known. However, accurate estimates of the ECM are rarely available unless the experiment was well designed and sufficient replicates were obtained with the idea of leveraging error information in the data analysis.

Commonly, MLPCA is applied assuming that error correlation exists only in the rows of the data (*i.e.* in the direction of the response variables). In this case, the objective function that is minimized is

$$S^2_{\text{obj}} = \sum_{i=1}^{m} (\mathbf{x}_i - \hat{\mathbf{x}}_i) \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_i - \hat{\mathbf{x}}_i), \tag{1.14}$$

where $\mathbf{x}_i$ represents the ith row of the $m \times n$ data matrix $\mathbf{X}$ (m samples and n measurement vectors), $\hat{\mathbf{x}}_i$ is the estimate of the ith measurement vector obtained from the MLPCA decomposition, and $\boldsymbol{\Sigma}_i$ is the ECM associated with the ith measurement vector. In the special case where the ECM for each row is the same, minimization of the objective function can be solved in one step. If the ECM is different, then the function is minimized using alternating least squares (ALS). Instead of performing an orthogonal projection into the subspace like what is done in PCA, MLPCA performs an oblique projection. This projection is weighted by the measurement error information. In theory, MLPCA should provide the optimal subspace estimation if the rank of the data and the ECM are known. However, in practice, the rank of a data set is sometimes not clearly defined and, more often than not, there is not a good estimate of the ECM, although there have been efforts to model ECMs with a limited number of replicates using a variety of methods.[43,44] Despite these two requirements, MLPCA has been shown to provide superior results compared to PCA in many applications.[45–48]

A critical component to both the modelling of ECMs and assessing the utility of MLPCA is the simulation of measurement errors for analytical measurements, as discussed in Chapters 2 and 3. Additional applications of MLPCA are discussed in Chapter 4.

1.7.3 Kurtosis-Based Projection Pursuit Analysis

Revealing clusters of similar samples (classes) in data is a common goal in exploratory data analysis. Although PCA is the most widely applied projection method in this regard, another method that can be employed when PCA fails is projection pursuit analysis (PPA).[49] PPA is an exploratory data analysis method that searches for "interesting" projections of the data by optimizing a projection index. For a data matrix $\mathbf{X}(m \times n)$ with m samples

and n response variables, PPA seeks the projection matrix $\mathbf{P}_p(n \times p)$ to project $\mathbf{X}$ into a p-dimensional space with coordinates, or scores, $\mathbf{T}_p$:

$$\mathbf{T}_p = \mathbf{X}\mathbf{P}_p. \tag{1.15}$$

Here, the columns of $\mathbf{P}_p$ contain the projection vectors for each of the p dimensions. These vectors are typically orthogonal, but this is not a requirement. In most implementations of PPA, the projection vectors are found sequentially. After finding the first projection vector, the variance associated with that vector is removed from the data set and then the next projection vector is sought. This process is referred to as deflation. Many different projection indices have been investigated for targeting specific structures in multivariate data.[50–54] For clustering, it is usually of interest to search for non-Gaussian distributions as these tend to reveal useful information. PCA can be thought of as a projection pursuit method where the projection index is the variance and the goal is to find projection vectors that maximize this index. Finding projection vectors that maximize the variance is equivalent to obtaining the PCA decomposition of a data set.

Although different projection indices have been explored, one index that has been shown to reveal meaningful clusters in multivariate chemical data is the minimization of kurtosis.[40,55,56] Kurtosis is the fourth statistical moment and is a measure of the density in the tails, or "tailedness", of a distribution.[57] The univariate sample kurtosis, κ, is

$$\kappa = \frac{1/N \sum (x_i - \bar{x})^4}{(1/N \sum (x_i - \bar{x})^2)^2}, \tag{1.16}$$

where x_i is the ith measurement out of a total of N measurements. The kurtosis is a useful and simple metric for determining the non-Gausianity of distributions as a Gaussian distribution has a value of 3 and deviations from this either increase or decrease the kurtosis. Therefore, minimizing or maximizing the kurtosis can be used to target non-Gaussian distributions in data. Maximization of the kurtosis tends to reveal outliers in data whereas minimization tends to reveal binary clusters of data.

Although the idea of kurtosis as a projection index makes intuitive sense for finding "interesting" distributions in data, there was major problem limiting its practicality: how does one optimize the kurtosis as a projection index to find projection vectors in multivariate

data? In 2011, Hou and Wentzell proposed an efficient algorithm, termed the quasi-power algorithm, that enabled kurtosis-based PPA to be used as an exploratory method.[40] Providing solutions for optimizing both the univariate and multivariate kurtosis as the projection index, this contribution allows one to explore multivariate data using something other than variance as the projection index.

In kurtosis-based PPA (kPPA), the univariate kurtosis of the data matrix $\mathbf{X}$ projected onto a projection vector $\mathbf{p}$ can be defined as

$$\kappa = \frac{1/m \sum_{i=1}^{m} (\mathbf{x}_i \mathbf{p})^4}{(\mathbf{p}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \mathbf{p})^2}, \tag{1.17}$$

where $\mathbf{x}_i$ is a row from the data matrix $\mathbf{X}$. Instead of extracting individual projection vectors sequentially, which is required when optimizing univariate kurtosis, the projection matrix $\mathbf{P}$ can also be found by optimizing the multivariate kurtosis. The multivariate kurtosis K can be defined as

$$K = m \sum_{i=1}^{n} \{\mathrm{tr}[(\mathbf{P}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \mathbf{P})^{-1} (\mathbf{P}^{\mathrm{T}} \mathbf{x}_i \mathbf{x}_i^{\mathrm{T}} \mathbf{P})]\}. \tag{1.18}$$

To minimize or maximize the univariate or multivariate kurtosis, kPPA implements the quasi-power algorithm, which is similar to the power method used for eigenvalue estimation. The algorithm uses an iterative learning method for finding the projection vectors. The structure of this learning algorithm changes depending on whether the univariate or multivariate kurtosis is being used as the projection index, as well as whether or not they are being minimized or maximized. For example, below is the learning algorithm for the minimization of univariate kurtosis,

$$\mathbf{p}_j(\text{new}) \leftarrow \left[\sum_{i=1}^{m} (\mathbf{x}_i \mathbf{p}_j)^2 \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_i \right]^{-1} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right) \mathbf{p}_j, \tag{1.19}$$

where j is the iteration number. After convergence, which can be defined in a number of different ways, Equation 1.19 provides a single projection vector associated with a minimum of the kurtosis of the corresponding scores. If more dimensions are of interest, the data are then deflated by the variance associated with the first projection vector and the same steps are applied to obtain a second projection vector and so on.

kPPA has proved useful in many exploratory analysis applications. For example, it has

been used as a tool to assess the similarity and clustering of chiral polysaccharide-based systems,[58] unsupervised analysis for the forensic investigation of fraudulent documents using IR hyperspectral images,[59] exploratory analysis of natural health products by NMR,[60] and the unsupervised clustering of four different species of Brazilian trees by NIR reflectance.[61] In the rest of this work kurtosis-based PPA will be referred simply as PPA.

1.7.3.1 Drawbacks of PPA

Despite its many advantages the original stepwise-univariate PPA algorithm, which is the most effective for classification problems, suffers from several deficiencies that limit its widespread application to analytical data: (1) it requires "skinny" data sets, where the number of samples is larger than the number of variables, (2) it is optimally designed for balanced data sets, in which there are equal number of samples in each class, and (3) it is most effective for binary data sets (2, 4, or 8 classes). The solutions to these problems are a main focus of this thesis and elaborated on below.

All PPA algorithms (univariate, multivariate, minimization, maximization) suffer from a data dimensionality problem. For kPPA to produce meaningful results the sample to variable ratio ideally should be $\sim$5:1. Practically, to overcome this, it is common practice is to first compress the data with PCA to an appropriate number of principal components (effectively achieving the $\sim$5:1 ratio) before applying PPA. This is done to avoid giving the algorithm too many variables as spurious combinations of variables can result in an artificially low kurtosis value and separations that are not informative. This is referred to as "overfitting" or "overmodeling", although this is not in the usual sense of supervised methods. This compression is far from ideal as useful information might be stripped from the data due to low variance. Since, PPA does not seek projection vectors based on their variance, but their kurtosis, it is possible that the best projection vector is being lost in this compression step. Chapter 5 of this thesis seeks to solve this problem by implementing a variable selection genetic algorithm to make the proposed method truly independent of PCA and improve variable interpretability.

Another major problem with the ordinary univariate PPA algorithm is that it favors data sets with a number classes equal to a power of 2. This is a consequence of the univariate kurtosis as in each dimension the PPA algorithm is looking to segregate the data into two clusters (the reason for this will be discussed in Chapter 5 and 7). This results in a geometric restriction on the separation space where, ideally, a maximum of 8

classes can be separated in a 3-dimensional space (each class clustered around a vertex of a cube). Additionally, when classes are unbalanced (*i.e.* they have a different number of samples), the ordinary univariate PPA algorithm will fail to separate even two classes from one another as the minimization of univariate kurtosis favors equal bimodal distributions. This results in projections that have heavily overlapped classes when one class has fewer samples than the other class. In Chapter 7 of this thesis an augmented method is proposed that addresses both of these problems.

1.7.4 Variance and Chemical Information

Earlier in this chapter, it was stated that the goal of exploratory data analysis is to separate meaningful chemical variance from variance that is not relevant using only the measured data (no external information). Since the advancement of methods to achieve this goal is the central theme of this thesis, this section provides an illustrative example to unite the ideas of measurement noise, chemical variance and exploratory data analysis.

The illustrative data sets consist of two-dimensional data (two variables, x_1 and x_2) that are to be projected into one dimension. The two dimensions are a surrogate for high-dimensional data. Although the relationships of the objects are easily visualized in two dimensions, this would not be possible for high-dimensional data. The first set of data, shown in Figure 1.8(a) consists of two classes of 100 samples each. Each class of samples is subject to two sources of variance, represented by the circular and elliptical boundaries. Both boundaries represent independent (uncorrelated) variance sources. Variance source 1 (solid circles) is characterized by equal variance in x_1 and x_2, while variance source 2 (dotted ellipses) has a higher variance in x_2 (heteroscedastic).

For interpretation purposes the variance sources can be assigned to two origins: (1) natural chemical variation arising from heterogeneity of samples around the population mean (*i.e.* sampling variance), and (2) measurement error variance, arising from noise in x_1 and x_2. The assignment of these origins to the sources is irrelevant to the data obtained. In other words, the same observed data would be obtained whether the elliptical (heteroscedastic) variance were assigned to chemical variance or measurement noise. This is critical, since PCA makes no distinction about sources of variance.

The objective of exploratory data analysis is to separate the two classes of samples when projected into a one-dimensional subspace. PCA models the total variance of the data, so the first loading vector (eigenvector) shown in Figure 1.8(a) aligns predominantly

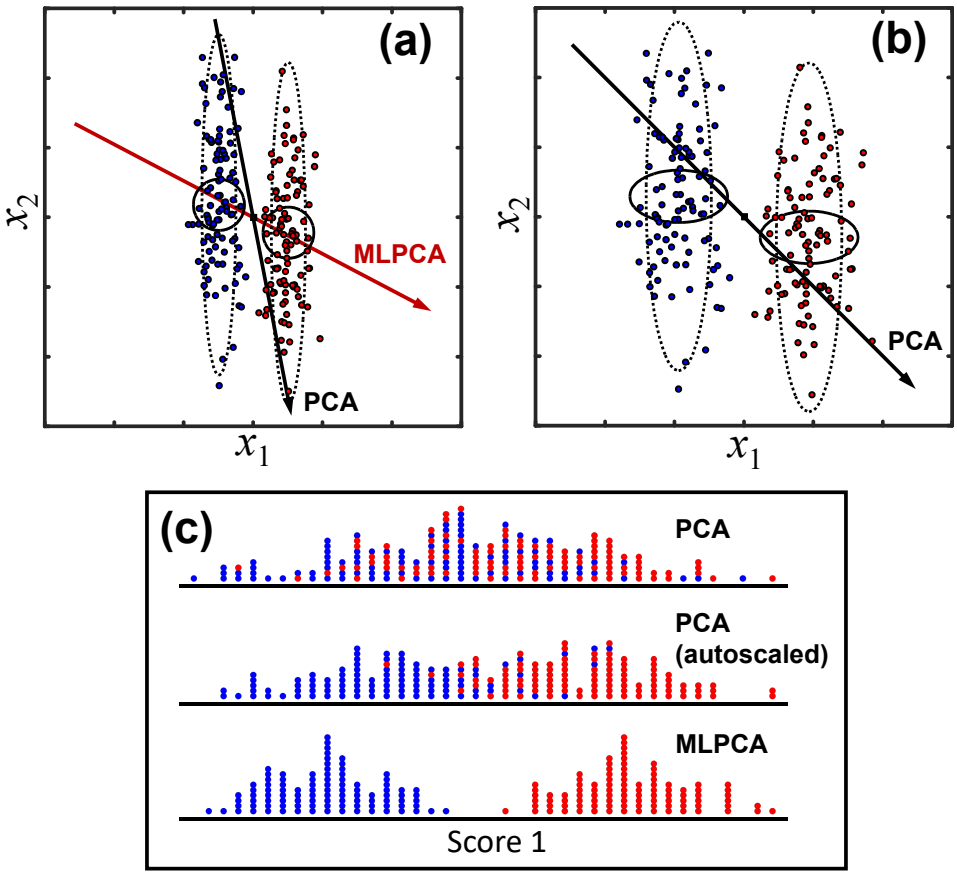

Figure 1.8: (a) A two-class (100 samples in each class), two-variable, uncorrelated data set where the variance in x_2 is larger than that in x_1 and the corresponding projection vectors obtained by PCA and MLPCA. (b) The same data, but autoscaled. (c) The corresponding scores on the projection vectors for (a) and (b).

with x_2 since that is the largest source of variance in the data. Consequently projecting onto this vector, shown at the top of Figure 1.8(c) does not separate the classes. This illustrates how PCA, which is based on total variance, can fail to reveal the important chemical information. Unlike PCA, MLPCA uses information about the measurement error variance (in the ECM) to attempt to isolate the chemical variance. If the elliptical variance (source 2) is associated with measurement noise and its corresponding ECM is provided to MLPCA, the resulting projection vector is as shown in Figure 1.8(a) and is much more closely aligned with the x_1 axis. Consequently, the maximum likelihood (oblique) projection of data onto this axis leads to a complete separation of the classes, as shown the bottom part of Figure 1.8(c). This illustrates how the application of MLPCA when measurement error information is available can (sometimes) improve the outcome

of exploratory data analysis. However, it is important to stress that, if the measurement variances were associated with the chemical variance (source 1), MLPCA would fail and produce the same result as PCA. This is because, although it would model the chemical variance, the important chemical variance (for our purposes) is not aligned with x_2.

In cases where the measurement variance is unknown or is not the dominant source of variance, another widely used option is scaling of the data. Autoscaling is a routinely used process where each variance is normalized to a total variance of unity. The effect of this on the data and the resulting PCA eigenvector is shown in Figure 1.8(b). The eigenvector is rotated closer to x_1 because of the compression of the x_2 axis and results in an improved separation of classes as shown in Figure 1.8(c) (middle). The separation is better than PCA, but not as good as MLPCA.

To expand this example, a second data set is introduced which has the same characteristics as the first, except that covariance has been introduced into variance source 2 (ellipses) as shown in Figure 1.9(a). This is equivalent to rotating the ellipses so that they are no longer aligned with the axis as in Figure 1.8(a). The loadings (PCA, MLPCA, autoscaled PCA) and scores were generated as before (Figure 1.9) and results in the following observations: (1) PCA still fails to separate the classes since the eigenvector aligns with the direction of greatest variance, (2) MLPCA is still able to separate the classes as long as the measurement variance is associated with the ellipses (source 2) and is known *a priori*, (3) MLPCA will still fail if the measurement error variance is associated with source 1 (circles), and (4) Autoscaled PCA is ineffective in this case. So what can be done if the measurement error ECM is unknown, or the chemical variance is in a direction that does not align with the direction of class separation? One approach is to use PPA, which does not rely on variance to extract information. In this example, the PPA projection vector will be aligned to minimize kurtosis, generating a bimodal distribution. The PPA vector shown in Figure 1.9(a) results in the sample scores shown at the bottom of Figure 1.9(c), demonstrating a complete separation of sample classes. Thus, it represents a powerful option when other methods of exploratory analysis fail.

1.8 Overview of Thesis

The preceding sections have attempted to illuminate the main premises of this thesis that unify its central chapters.

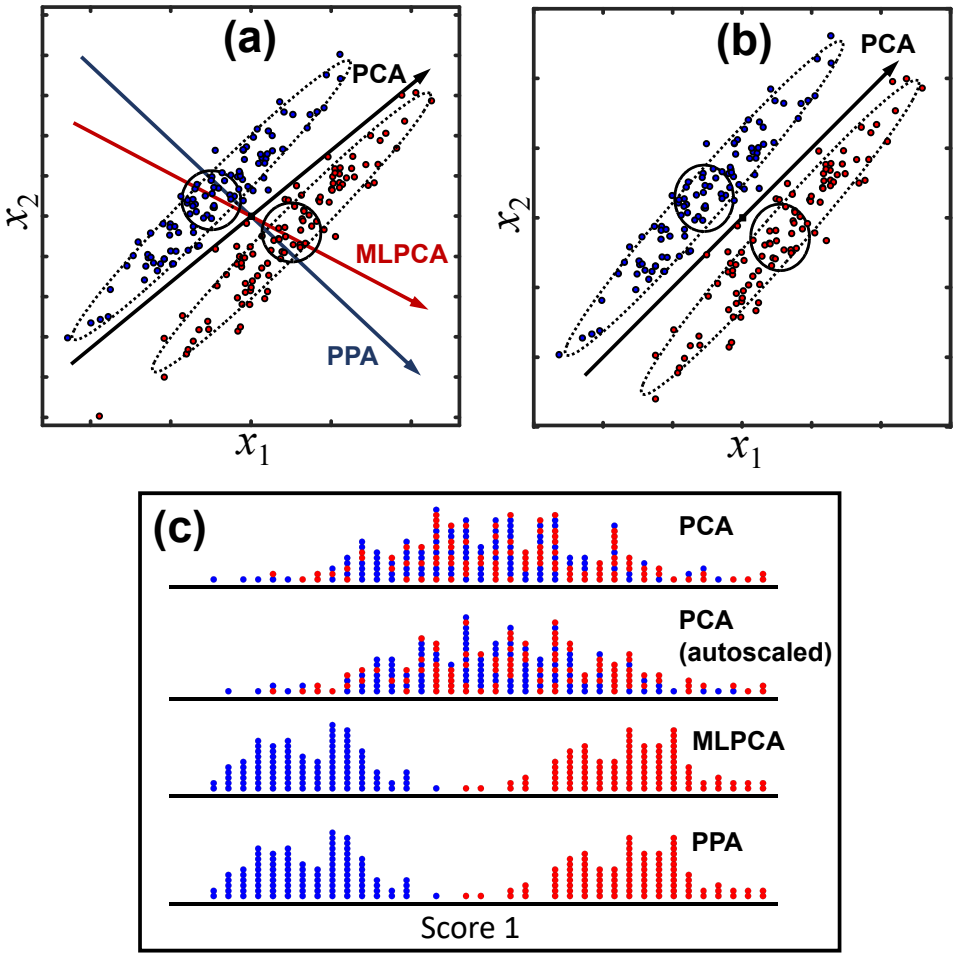

Figure 1.9: (a) A two-class (100 samples in each class), two-variable, correlated data set where the variance in x_2 is larger than that in x_1 and the corresponding projection vectors obtained by PCA and MLPCA. (b) The same data, but autoscaled. (c) The corresponding scores on the projection vectors for (a) and (b).

1. Exploratory data analysis is a fundamental application of chemometrics, but a primary tool in this application, PCA, often fails because it models total variance.

2. MLPCA can improve classification by incorporating measurement error variance information if the ECM is known.

3. With respect to 2, the ability to simulate measurement errors accurately is critical to assess there impact on data analysis strategies, including exploratory data analysis.

4. Where measurement error information is unknown or unhelpful, PPA is a powerful alternative to variance based methods, but suffers from a number of deficiencies that

limit its utility.

Chapter 2 of this thesis addresses a persistent limitation of noise simulation methods, namely the development of methods to simulate power law noise based on the ECM. Despite its prevalence in experimental systems, this approach to simulating power law noise has not been previously attempted. In Chapter 3, a publicly available software package (in MATLAB) is described that incorporates power law noise simulation with other types of noise generation to provide a versatile approach to the simulation of noise for any type of data. The software is demonstrated using MLPCA and baseline fitting as examples.

The application of noise generation software is further demonstrated in Chapter 4, which proposes the application of MLPCA to data fusion, where different types of data are combined to enhance information content. Simulated data are used to support the hypothesis that MLPCA is an optimal way to compensate for differences in error structures among data sets. Experimental data are then used to illustrate this point.

Where PCA and MLPCA fail, PPA is a promising alternative and Chapters 5-7 address the current deficiencies in PPA algorithms. Chapter 5 introduces a sparse form of PPA that addresses the problem of "fat" data and improves interpretability. Chapter 6 introduces a "banded" form of this algorithm that leads to more stable solutions. Finally, Chapter 7 solves the persistent problems of non-binary and unbalanced data sets by a modification referred to as augmented PPA.

CHAPTER 2

SIMULATION OF $1/f^{\alpha}$ NOISE FOR ANALYTICAL MEASUREMENTS *

This chapter presents a simple procedure that can be used to generate $1/f^{\alpha}$ noise, also known as power law noise, in simulated analytical measurement vectors. Certain types of power law noise, such as pink noise ($\alpha = 1$), dominate many types of analytical signals, so its simulation is important in optimizing and evaluating data processing strategies. In this work, simulated $1/f^{\alpha}$ error sequences are created directly from white noise via the theoretical measurement error covariance matrix (ECM) by rotation and scaling. The $1/f^{\alpha}$ ECM is obtained from the coefficients of a finite impulse response filter and is easily adapted to generate multiplicative $1/f^{\alpha}$ noise, which is probably more common for analytical systems exhibiting proportional or shot noise characteristics. Simulating $1/f^{\alpha}$ noise directly from the ECM offers two main advantages. First, $1/f^{\alpha}$ noise can be easily simulated in the presence of other common analytical measurement errors by additive combination of the ECMs. Second, the theoretical ECM can be used to model real experimental measurement noise. It is shown that the power spectral density function of measurement error sequences generated by the proposed method closely approximates the theoretical behaviour of $1/f^{\alpha}$ noise. To demonstrate the utility of this method in evaluating data processing methods, simulated data exhibiting $1/f$ (pink) noise is analyzed by maximum likelihood principal component analysis (MLPCA), which takes measurement error structure into account, and baseline noise is simulated using brown noise to test

*This chapter is based on the published article: Driscoll, S, Dowd, M, Wentzell, P.D. Simulation of $1/f^{\alpha}$ noise for analytical measurements. *Journal of Chemometrics*. 2019;e3137. https://doi.org/10.1002/cem.3137. **Contribution to Manuscript** SD performed all calculations, and wrote all drafts of the manuscript with edits by MD and PW. PW supervised the project.

baseline fitting by asymmetric least squares (AsLS).

2.1 Introduction

All analytical measurements are subject to errors that lead to uncertainty in derived results. For this reason, a great deal of research in analytical chemistry has been directed towards understanding these measurement errors, including various aspects of their origins, their mitigation and their consequences for analytical results. For example, in their classical textbook on spectrochemical analysis, Ingle and Crouch discuss the origins of measurement uncertainty for a variety of spectroscopic techniques.[62] Measurement errors are also an essential element in any discussion of analytical figures of merit, such as the limit of detection, for both univariate and multivariate methods.[1,63]

From the perspective of method development, especially as it relates to any kind of data processing (ranging from simple calibration to computationally intensive machine learning), the simulation of analytical measurement errors is an important part of any evaluation. Unfortunately, most approaches to the simulation of analytical signals in the literature rely on the addition of *iid* normal (independent and identical draws from a normal distribution) errors (see for example references 32–34). This limits the generality of the simulation, since most commonly encountered analytical measurement errors exhibit heteroscedasticity (non-uniform variance) and/or correlation.[36,44,64] The accurate representation of noise is important for the assessment of both simple signal processing procedures (*e.g.* smoothing, differentiation, integration, baseline correction, peak detection, normalization, scaling) and more complex operations, such as PCA,[4] curve resolution,[65] and multivariate calibration.[66] By evaluating data analysis methods under a variety of noise structures, it is possible to assess the impact of the noise characteristics on the results. More recently, noise simulation was used as a key element in the evaluation of multivariate figures of merit[67] and the bootstrap evaluation of measurement error models for analytical measurement systems.[43]

The most basic approach for simulating noise is generating a random sequence of numbers, each drawn from the same normal distribution with the identical variance (independent and homoscedastic). This is referred to as white noise or *iid* normal noise. For independent (uncorrelated) noise, it is straightforward to adapt noise simulation to account for heteroscedasticity (e.g. proportional noise, shot noise) simply by scaling each element

of the noise sequence according to the standard deviation at that channel. However, real analytical measurement errors are often comprised of many types of errors, both independent and correlated, so a more effective way to approach noise simulation is through the ECM. The ECM describes the variance and covariance terms for all combinations of measurement channels in a system. It has been shown that noise simulation can be performed by scaling and rotating white noise via the ECM of the desired noise.[43] Correlated noise components, such as baseline offset noise and multiplicative offset noise, are easily simulated when formulated in terms of the ECM, the latter typically being represented as the outer product of two fixed or signal-shaped vectors.[36] However, one type of correlated noise that is widely observed in analytical systems, $1/f^{\alpha}$ noise or power law noise, is more challenging to simulate.

A common example of power law noise is so-called "pink noise", also known as source flicker noise or $1/f$ noise. The name $1/f$ noise arises from its characteristics in the Fourier domain, where the power decays proportionally to the inverse of the frequency. Although the behaviour of $1/f^{\alpha}$ noise is referenced to frequency in the Fourier domain, implying the original signal was in the time domain, the behaviour can be observed even if the original signal was recorded in a different domain (e.g. wavelength), since ordinal variables may be correlated with time. While the Fourier transform of white noise exhibits a flat power spectrum, the power spectrum of pink noise decays to have lower magnitudes at high frequencies. In practical terms, this means that error components are concentrated at lower frequencies which results in long-range correlated errors. This is especially troublesome for chemical signals, as these signals are typically in the same frequency domain as the noise. Visually, a $1/f$ noise sequence varies more slowly than white noise, as shown in Figure 2.1 where white noise ($\alpha = 0$) and pink noise ($\alpha = 1$) are separately added to a Gaussian signal. The addition of pink noise introduces correlation in the idealized (*iid* normal) errors, easily seen in Figure 2.1 near the baseline where another peak appears to emerge in the noise, even though this is an artifact.

The origin of $1/f$ noise is not entirely understood, but it has been observed to apply to a wide variety of behaviours that include kinetics in condensed matter,[68] radiation from black hole objects,[69] heart rate variability,[70] respiratory intervals,[71] traffic flow/jams,[72] and is consistent over large frequency ranges.[73] Specific to analytical chemistry, $1/f$ noise has been explored in measurement systems such as inductively coupled plasma mass

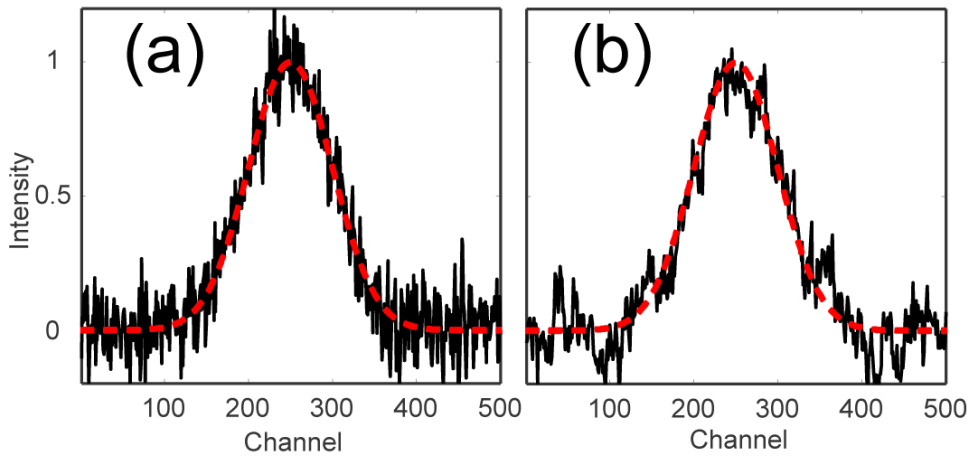

Figure 2.1: (a) The addition of white noise and (b) pink noise to a Gaussian peak (red dashed line).

spectrometry,[74] liquid chromatography,[75] fluorometry,[76] and polarimetry,[77] and methods have been proposed for its detection in a wide variety of analytical signals.[78] In the present case, instead of focusing on the simulation of a specific type of power law noise, such as pink noise ($\alpha = 1$), a better and more general approach is to develop a description for simulating a range of power law noise types that are observed in analytical measurements. This restriction will limit our discussion to α values in the range $0 < \alpha \leq 2$. The purpose of this work is to develop a method for simulating $1/f^{\alpha}$ noise in the presence of other common types of analytical measurement errors for use in the evaluation data analysis procedures. The approach for power noise simulation developed here differs from other strategies in that it is based on the generation of an ECM. Using the additive properties of covariance matrices, power noise can be easily combined with other types of noise to efficiently simulate complex noise structures in analytical data.

2.2 Background

2.2.1 The Power Spectrum of $1/f^{\alpha}$ noise

The relationship between the power, P, and frequency, f, of noise is referred to as the power spectral density (PSD). The spectral density gives information on how the variance of a sequence or process is distributed over a frequency range. The notation $1/f^{\alpha}$ is derived from this relationship as noise of this type has power inversely proportional to its frequency and so the absolute value of the slope of the line formed when plotting $\log P$ vs

$\log f$ corresponds to the value of α in $1/f^{\alpha}$. When $\alpha = 0$, the noise has equal power at all frequencies and is said to be white. This type of noise is assumed in most data analysis methods, such as PCA. However, real analytical signals typically exhibit more complex noise structures, including power law noise, with $\alpha = 1$ (pink noise) and $\alpha = 2$ (brown noise) being the most common.

In this work, the PSD is used as a tool to determine the frequency characteristics of simulated noise signals. The standard approach for calculating the PSD is the periodogram spectrum method, where the squared amplitude components of the discrete Fourier transform (DFT) estimate the power spectrum.[79] However, the power spectrum of one signal, or even the average obtained from a few signals, does not give an accurate representation of the true power spectrum due to large stochastic variations. To overcome this, large sets of simulated signals can be ensemble averaged to give a better estimate of the true power spectrum.

2.2.2 Methods for Generating $1/f^{\alpha}$ noise

A thorough review on the simulation of $1/f^{\alpha}$ noise was published by Kasdin in 1995, in which the advantages and drawbacks of many techniques for power law noise simulation are discussed.[80] Generally, the approaches to discrete simulation can be placed into two categories: 1) simulation of power law noise by filtering white noise in the time domain and, 2) simulation of power law noise by filtering white noise in the frequency domain. A brief and very general overview of these methods will be given here.

In the time domain, the convolution of a white noise sequence with a set of filter coefficients can produce noise with specific frequency characteristics. In general, either non-recursive (finite impulse response, FIR) or recursive (infinite impulse response, IIR) digital filters can be used, but the latter are more efficient in terms of the number of coefficients. For $1/f^{\alpha}$ noise, this method can be represented as an autoregressive process of order n (AR(n)) with coefficients defined by α. This is consistent with an IIR filter of sufficient length to represent the process. The MATLAB DSP System Toolbox contains the object "*dsp.ColoredNoise*" that uses this method to generate power law noise,[81] originally proposed by Kasdin in the review. Specifically, it uses an AR(63) model, designated in standard form by Equation 2.1, where the filter coefficients, a_k, are defined by Equation

2.2, α is the noise power, and $\mathbf{w}$ represents a zero mean white noise process.

$$\sum_{k=0}^{63} a_k y(n-k) = w_n. \tag{2.1}$$

$$a_k = (k - 1 - \frac{\alpha}{2})\frac{a_{k-1}}{k}. \tag{2.2}$$

In practical terms, each element of the coloured noise sequence y_n, is obtained by applying the recursive filter to a white noise sequence, as shown in Equation 2.3,

$$y_n = w_n - \sum_{k=0}^{63} a_k y(n-k). \tag{2.3}$$

Filtering of white noise by FIR filters to generate power law noise is also possible. The design of FIR filters from the desired power spectrum, as discussed in Section 2.3.2, is more straightforward than for IIR filters. However, more FIR coefficients will be needed for results of similar quality, increasing the computational time substantially.

It is also possible to generate power law noise in the Fourier domain.[82] This can be accomplished by first generating a white noise sequence in the time domain and then applying a Fourier transform (FT). The real and imaginary parts of the transformed data are then multiplied by the desired (normalized) noise power spectrum and then transformed back to the time domain through an inverse FT.

All of the above methods are effective and have trade-offs with respect to design complexity, accuracy, and computational burden. However, for applications to analytical measurements, simulation of noise via the ECM is often most convenient because it more easily allows for mixed error models often observed. This approach has not previously been used for power law noise, but can be implemented via the FIR filter approach described above. More details are given in Section 2.3.2.

2.2.3 The Error Covariance Matrix

The ECM is introduced here since it is a critical part of the current implementation. Defining $\mathbf{x}^\circ$ as the $n \times 1$ true measurement vector and $\mathbf{x}$ as a $n \times 1$ vector of experimental measurements, we can define the difference between them as the measurement error vector, $\mathbf{e}$,

$$\mathbf{e} = \mathbf{x} - \mathbf{x}^\circ. \tag{2.4}$$

To obtain all combinations of covariance terms for a system of n variables, the expectation, E(*), of the outer product of the measurement error vector $\mathbf{e}$ with itself gives rise to the ECM, $\mathbf{\Sigma}$,

$$\mathbf{\Sigma} = \mathrm{E}(\mathbf{e}\mathbf{e}^{\mathrm{T}}). \tag{2.5}$$

Obtaining an expression for the theoretical ECM of $1/f^{\alpha}$ noise offers two main advantages. First, simulation of noise consistent with the ECM is simple if the ECM is known. This is detailed in the following section. Second, previous measurement error models and data processing methods have been developed using the ECM at their core. For example, methods that include measurement error information, such as MLPCA[41,42] and multivariate curve resolution-weighted alternating least squares (MCR-WALS),[48] use the ECM to incorporate the measurement error information in the analysis. The goal of this chapter is not strictly to simulate $1/f^{\alpha}$ noise, but to express the characteristics of this unique type of noise in the form of the ECM. As shown in a previous section, there are many satisfactory methods for simulating $1/f^{\alpha}$ noise. However, these do not provide the ability to easily simulate $1/f^{\alpha}$ noise in the presence of other types of noise commonly encountered in analytical signals (e.g. shot noise, baseline offset). Obtaining an expression for the $1/f^{\alpha}$ ECM as a function of α allows for the simulation of any type of power law noise in the presence of other common noise via rotation and scaling of *iid* normal noise (similarity transform).

2.3 Theory

2.3.1 Noise Simulation

As shown in Equation 2.6, by applying a transformation (rotation), $\mathbf{P}$ $(n \times n)$, derived from singular value decomposition (SVD) of an ECM, $\mathbf{\Sigma}$ $(n \times n)$, to a vector of *iid* normal errors $\mathbf{e}_{iid}$, an error vector $\mathbf{e}_{\Sigma}$ consistent with the ECM is obtained.[43]

$$\mathbf{e}_{\Sigma} = \mathbf{L}\mathbf{\Lambda}^{\frac{1}{4}}\mathbf{e}_{iid} = \mathbf{P}\mathbf{e}_{iid}. \tag{2.6}$$

In this equation, $\mathbf{\Lambda}$ $(n \times n)$ is a diagonal matrix containing the eigenvalues and the columns of $\mathbf{L}$ $(n \times n)$ contain the eigenvectors, both obtained from the SVD of $\mathbf{\Sigma}$, while $\mathbf{e}_{iid}$ $(n \times 1)$ is a vector of *iid* normal errors. Using this method, one is able to obtain correlated and

heteroscedastic noise simply by rotating and scaling *iid* normal noise with the proper information obtained from the SVD.

More commonly, in chemical applications, signals are represented as rows of a matrix, $\mathbf{X}$ $(m \times n)$, so Equation 2.6 can be generalized to:

$$\mathbf{X} = \mathbf{X}^{\circ} + \mathbf{E}_{iid}\mathbf{P}^{\mathrm{T}}, \tag{2.7}$$

where $\mathbf{X}^{\circ}$ is the noise-free simulated data $(m \times n)$, $\mathbf{X}$ contains the noisy data with the desired structure, and $\mathbf{E}_{iid}$ is a matrix of *iid* normal errors. Of course, if $\mathbf{\Sigma}$ is signal dependent (*e.g.* proportional noise) noise vectors need to be generated individually. This method is powerful and makes it clear that, if we can obtain the theoretical ECM for power law noise, then simulation of $1/f^{\alpha}$ noise can be performed simply by scaling and rotating *iid* normal noise. Therefore, the first step in this investigation is finding an expression for the $1/f^{\alpha}$ ECM as a function of α. Formulating the problem from the perspective of a FIR filter provides direct access to the $1/f^{\alpha}$ ECM in an intuitive way.

2.3.2 FIR Filter Design

An FIR filter is a digital filter that produces an output by the convolution of a fixed set of filter coefficients, represented by a vector $\mathbf{c}$ of length m, with a vector $\mathbf{x}$ of length n, as shown in Equation 2.8, where $\mathbf{y}$ is the output vector and "$*$" represents the convolution operator.

$$\mathbf{y} = \mathbf{c} * \mathbf{x}. \tag{2.8}$$

In practice, both $\mathbf{c}$ and $\mathbf{x}$ are finite in length and so will define the length of $\mathbf{y}$. If we define $\mathbf{c} = \begin{bmatrix} c_{-w} & c_{-w+1} & \dots & c_0 & \dots & c_w \end{bmatrix}$, where $m = 2w + 1$, $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_p \end{bmatrix}$, we can write Equation 2.9 as:

$$y_i = \sum_{j=-w}^{w} c_j x_{i+j+w}. \tag{2.9}$$

This convolution defines $p = n - m + 1 = n - 2w$, requiring that n (the length of the signal vector) is greater than or equal to m (the length of the filter). The filter coefficients, $\mathbf{c}$, define the frequency response of the filter, *i.e.* the Fourier spectrum of $\mathbf{y}$ when $\mathbf{x}$ is a white noise sequence. The most common example of an FIR filter is a least squares

polynomial (Savitzky–Golay) filter,[83,84] which attenuates the high frequency components of a signal, $\mathbf{x}$.

If the frequency response of a FIR filter is known, the filter coefficients can be obtained from the Fourier domain. In the present case, simulation of $1/f^{\alpha}$ noise is carried out by filtering white noise in the time domain. Since the desired frequency response is known ($1/f^{\alpha}$, by definition), we work backwards to determine the filter coefficients.

To determine the filter coefficients, we first define a vector of frequency values, $\mathbf{f}$ of length $m(= 2w + 1)$, where w is a positive integer, that extends from -0.5 to 0.5 in steps of $1/2w$: $\mathbf{f} = [\, f_{-w} \cdots f_0 \cdots f_w \,]$. These frequencies are given relative to the sampling frequency, f_s, and the limit of 0.5 represents the Nyquist frequency, where $f_N = 1/(2\Delta t)$ and Δt is the sampling interval. The filter mask, $\mathbf{H}$, for the white noise is then defined in terms of the power law noise desired for both positive and negative frequencies. This vector is symmetric about its centre, H_0, such that $H_{-w} = H_w$.

$$\mathbf{H} = \left[\frac{1}{f_{-w}^{\alpha/2}} \quad \cdots \quad \frac{1}{f_0^{\alpha/2}} \quad \cdots \quad \frac{1}{f_w^{\alpha/2}} \right]. \tag{2.10}$$

This definition has some practical implications that need to be addressed. The most obvious of these is that $1/f_0$ is undefined since f_0 is zero, but there are also implications associated with the filter width, m. These will be set aside for the moment and discussed in detail in Sections 2.3.2.1 and 2.3.2.2.

Once $\mathbf{H}$ is defined, the coefficients of the filter are obtained by first taking the inverse Fourier transform of $\mathbf{H}$ and then calculating the amplitude. This two step operation is shown in Equations 2.11 and 2.12, where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform operator (implemented as the fast Fourier transform (FFT)).

$$\mathbf{Z} = \mathcal{F}^{-1}(\mathbf{H}). \tag{2.11}$$

$$\mathbf{c}_i = \sqrt{Z_i Z_i^*}. \tag{2.12}$$

Here, Z_i^* indicates the complex conjugate of z_i. A Hamming window[85] is then applied to the filter coefficients. This is a common apodization function used in signal processing to reduce artifacts in the filter coefficients and was found to improve the quality of power law noise in this study. The coefficients for a Hamming window of length m can be generated

using Equation 2.13, where the multiplication of Hamming coefficients, $w(i)$, with the filter coefficients, $c(i)$, produce the windowed coefficients. In this work, the number of Hamming coefficients was set equal to the length of the FIR filter coefficients.

$$w(i) = 0.54 - 0.46\cos\left(2\pi\frac{i-1}{m-1}\right), \quad 1 \leq i \leq m. \tag{2.13}$$

After applying the Hamming window, the coefficients are then normalized to produce an output whose variance matches the input, as shown in Equation 2.14.

$$\mathbf{c}' = \mathbf{c}/||\mathbf{c}||. \tag{2.14}$$

It should be noted that the actual implementation of the procedure described above in the MATLAB programming environment is somewhat different due to the format of the Fourier coefficients. This requires a reordering of the vectors consistent with the specifications of the FFT.

2.3.2.1 Practical Considerations: DC Noise

As previously noted, Equation 2.10 is problematic since $f_0 = 0$ and therefore H_0 is undefined. This relates to the fact that true $1/f^\alpha$ noise has an undefined variance. To apply the proposed method, it is necessary to define a finite filter response value for f_0. Although this assignment might seem arbitrary and of little consequence since it relates to the DC noise level, it is actually important to the practical application of the filter and the calculation of the ECM. Suppose, for example, that H_0 were simply set to zero, implying no DC offset but $1/f^\alpha$ behaviour everywhere else. This seems reasonable, but discontinuities in $\mathbf{H}$ lead to artifacts at low frequencies that cause deviations from $1/f^\alpha$ behaviour. Moreover, the assignment of a value to H_0 implies, by interpolation, the response of the filter at the low frequencies between f_0 and f_1. These low frequencies have the greatest effect on long-range correlation, and therefore the magnitude of H_0 will affect the appearance of the ECM.

The solution to the problem of discontinuities in $\mathbf{H}$ is a smooth extrapolation of the frequencies approaching f_0, using a simple function such as a polynomial to estimate a value for H_0. Although the exact value of the extrapolated point will depend on several factors (order of polynomial, number of points, etc.) it has been found in this work that good results are obtained if H_0 is assigned a value of $(6w)^{\alpha/2}$ (or $(3m)^{\alpha/2}$) as shown in

Figure 2.2(a). Alternatively, f_0 can be set to $1/(6w)$ prior to applying Equation 2.10. These values produced reliable behaviour in the frequency domain for all reasonable values of filter width (m) and power factors (α).

2.3.2.2 Practical Considerations: Filter Width

Another important consideration is the length of the filter, m, selected in a given application. In general, larger filter widths (relative to the number of data points to be generated, n) will lead to a frequency response which more accurately reflects the desired behaviour, with shorter filters leading to a flatter response at low frequencies. Under conditions where, $m > n$ ($w \geq n/2$), the distortion in the frequency response is minimal, but the selection of the filter width will also affect the shape of the ECM generated. This can be understood by recognizing that extending the length of the filter does not change the Nyquist frequency, but it does decrease the interval between points on the frequency axis. In particular, because the interval between f_0 and f_1 is decreased and the value of H_0 is changed, the low frequency region (which affects long-range correlation) is modified. This will tend to increase the degree of correlation among measurements as the filter length is increased. Taking another perspective, long-range correlation can extend only to the length of the filter. Where $m < n$, the ECM will drop to zero whenever the separation of the channels is greater than m. Where $m > n$, correlations can extend beyond the window of the data, leading to an ECM which shows a higher degree of correlation.

The power noise simulated in this manner essentially depends on two parameters: α and the ratio of the filter width to the number of points generated in the signal vector, m/n, which we will define as ρ. When α is changed, the slope of the PSD plot ($\log P$ vs $\log f$) will change accordingly and the ECM will show higher correlations as α is increased, since the variance is shifted to lower frequencies. When $\rho > 1$, the slope of the PSD plot remains essentially unchanged as ρ increases (as long as α remains fixed) but the vertical position of the PSD will shift by a small amount, as shown in Figure 2.2(b). However, the value of ρ will have a significant effect on the ECM, and therefore the nature of the noise generated from it, with higher values of ρ leading to more correlated noise.

The foregoing description correctly implies that there is not a unique ECM for $1/f^{\alpha}$ noise, which may seem counterintuitive, but is again a consequence of undefined variance at $f = 0$. All analytical measurement systems are band-limited in some way, and so the variance in the DC signal must be constrained to reasonable values. Likewise, many

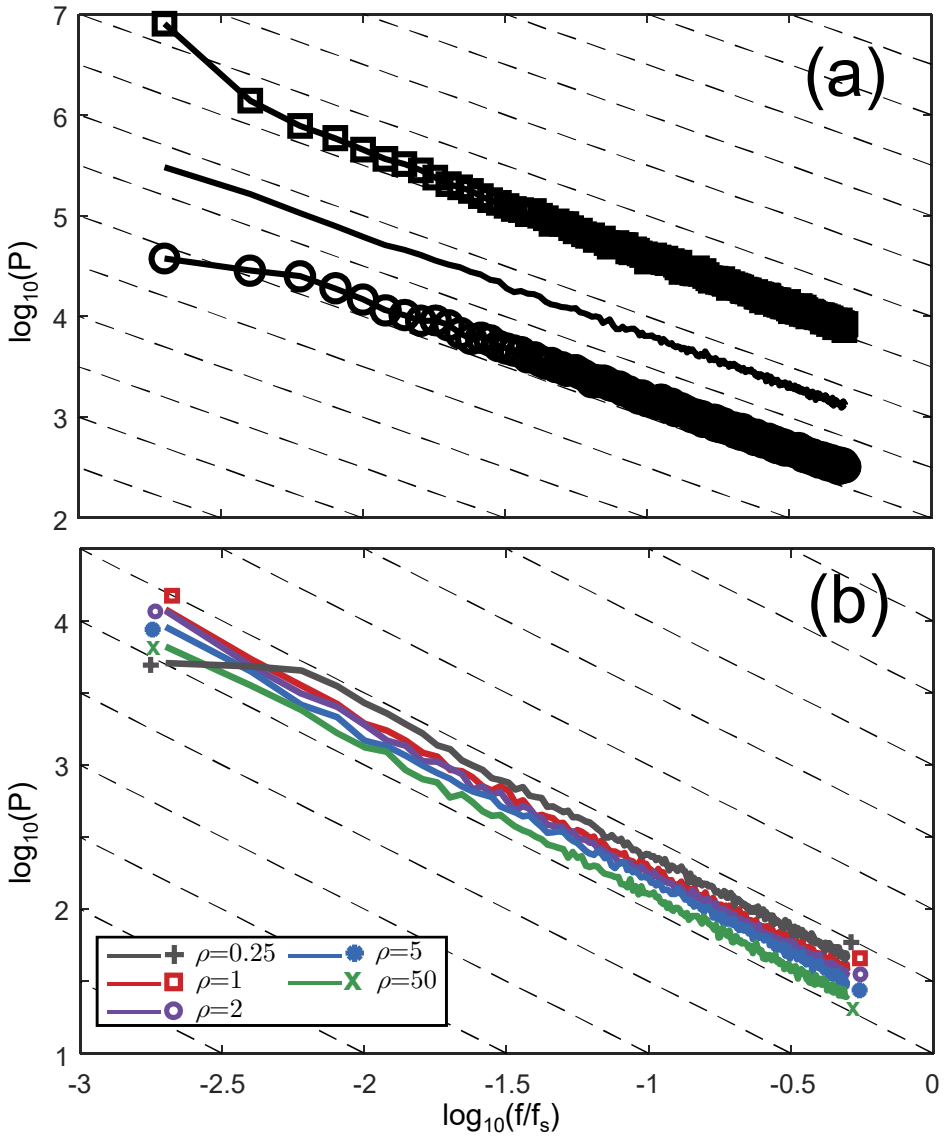

Figure 2.2: (a) PSD obtained from 1000 realizations of noise ($n = 500$, $\alpha = 1$, $\rho = 1$) with $H_0 = 6w^{\alpha/2}$ (solid black line), $H_0 = 0.6w^{\alpha/2}$ (solid black line with circle markers), and $H_0 = 60w^{\alpha/2}$ (solid black line with square markers). Note thesis plots are offset for clarity. (b) PSD obtained from 1000 realizations of noise ($n = 500$, $\alpha = 1$) with $\rho = 0.25$, 1, 2, 5, and 50. These plots were not offset to show the vertical shift observed between each of the cases.

instrument systems employ anti-aliasing filters at high frequencies. It is the low frequency behaviour in particular that influences the correlation in the measurements. Because extending the filter width decreases the interval between f_0 and f_1, the value of H_0 and the implied behaviour between DC and the next lowest frequency observable in the Fourier transform of the data are modified. This modification changes the distribution of the power at the low frequencies. Although this behaviour cannot be observed directly in the PSD of the noise signal, whose lowest frequency is defined by the number of measurement channels, it is observable in the ECM, where the redistribution at low frequencies leads to more correlated noise as the filter width (ρ) is increased. Since, in general, the low frequency behaviour of the analytical signal is not known, it is impossible to know the "correct" ECM. In principle, the low frequency behaviour could be determined empirically by taking a very long sample sequence, but this is generally impractical since many analytical measurement systems have a finite sequence length (e.g. a finite number of spectral channels in a spectrometer).

To simulate a $1/f^{\alpha}$ noise sequence of length n using the ECM, the recommendation of this work is to adjust the value of ρ (i.e. the filter length, m), to match the level of noise correlation anticipated, or to use a range appropriate for the simulation. For low values of ρ, the signal channels will be correlated over a short range, extending only the length of the filter. For larger values, low frequency (drift noise) components become more dominant, ultimately approaching simple DC offset (baseline shift) noise in the limit of large ρ. Some examples of this will be presented under Results and Discussion.

2.3.3 Generation of the $1/f^{\alpha}$ ECM

Once the filter coefficients have been generated using the above method in Section 3.2, they can be directly transformed into an ECM. Conceptually, this can be described in terms of the creation of a filter matrix using the following procedure. First, the filter coefficients are converted into a filter matrix that generates filtered data from the input data as shown in Equation 2.15.

$$\mathbf{y} = \mathbf{F}\mathbf{x}. \tag{2.15}$$

In this equation, $\mathbf{y}$ is the output sequence ($n \times 1$), $\mathbf{x}$ is the sequence to be filtered (($n + 2w) \times 1$) and $\mathbf{F}$ is the filter matrix ($n \times (n + 2w)$). The filter matrix contains the filter coefficients such that each row i contains the full set of filter coefficients beginning at

column i and ending at column $i + n$, with all other elements equal to zero, producing a band diagonal matrix. When the filtering procedure is presented in this way, the ECM can be described by Equation 2.16.

$$\mathbf{\Sigma} = \mathbf{F}\mathbf{F}^{\mathrm{T}}. \tag{2.16}$$

Although this procedure correctly reflects the process behind the generation of the ECM, the matrix manipulations can be computationally inefficient, especially when large filter widths are involved. For this reason, the actual procedure uses a more efficient convolution of the filter vector with itself to produce an equivalent result.

When the $1/f^{\alpha}$ ECM is formulated using the above definitions, all channels have a theoretical variance of unity. From here, the $1/f^{\alpha}$ ECM for power law noise, $\mathbf{\Sigma}_{\mathrm{PL}}$, can be multiplied by the desired variance and used with Equation 2.6 to generate $1/f^{\alpha}$ noise. To obtain the multiplicative $1/f^{\alpha}$ noise ECM, $\mathbf{\Sigma}_{\mathrm{PPL}}$ (proportional power law noise), Equation 2.17 can be used, where $\mathbf{x}^{\circ}$ is the signal vector ($n \times 1$), RSD is the relative standard deviation of the noise with respect to $\mathbf{X}^{\circ}$ and $\odot$ indicates the Hadamard product.

$$\mathbf{\Sigma}_{\mathrm{PPL}} = \mathbf{\Sigma}_{\mathrm{PL}} \odot (\mathbf{x}^{\circ}\mathbf{x}^{\circ\mathrm{T}})(\mathrm{RSD})^2. \tag{2.17}$$

Code for the procedures described in Section 2.3 can be found in Appendix A.

2.4 Experimental

2.4.1 Computational Details

All calculations were performed using programs written in-house with the MATLAB programming environment (MathWorks, Natick, MA).[81] A MATLAB function for performing the proposed method is included in Appendix A. The MATLAB function "*subspace.m*" was used to obtain subspace angles.[86,87]

2.4.2 Simulation Studies

To demonstrate the utility of this method, two simulations were used. In the first simulation, ^{1}H NMR data from 231 three-component mixtures (propanol, butanol, and pentanol) were used as a reference data set to evaluate the performance of PCA and MLPCA after the addition of $1/f^{\alpha}$ noise.[88] A specific region (0.8369 ppm - 0.9510 ppm) of the original data

set (231×14000) was chosen to produce a truncated data set (231×500). This data set was subjected to PCA and MLPCA after the addition of $1/f^{\alpha}$ noise ($\alpha = 1$ and 2, $\rho = 1$).

In a second study, to demonstrate the utility of this method with respect to data preprocessing, a simple chromatographic data set was simulated and its baseline was estimated using the asymmetric least squares (AsLS) smoothing algorithm[89] after the addition of $1/f^2$ noise. The simulated chromatographic vector (1×2000) was constructed by summing 6 Gaussian peaks centered at channel 100 ($\sigma = 5$, maximum height, $h = 1$), 300 ($\sigma = 10$, $h = 0.5$), 350 ($\sigma = 10$, $h = 0.5$), 500 ($\sigma = 15$, $h = 0.3$), 1100 ($\sigma = 25$, $h = 0.2$), and 1500 ($\sigma = 50$, $h = 0.1$).

2.5 Results and Discussion

2.5.1 Simulation of $1/f^{\alpha}$ Noise via the Theoretical ECM

Various noise sequences were simulated using the proposed method based on scaling and rotation via the generated $1/f^{\alpha}$ ECM. Sequences of 500 measurements with $\sigma^2 = 1$ were obtained for $\alpha = 0$ (white noise), 0.5, 1, 1.5, and 2, with a filter width of $m = 501$ $(\rho = 1)$. Typical noise sequences (offset for clarity) are shown in Figure 2.3. The general trend towards lower frequency variation is evident in these plots as α is increased. To further confirm the frequency behaviour of these noise sequences, 1000 realizations were generated under each set of conditions and individual power spectra were averaged to create a PSD plot. These plots are given in Figure 2.4, where the cases are offset from one another vertically for clarity (note that the markers used in the plot do not imply anything about the confidence levels of the simulation at certain frequencies). The highly linear behaviour and slopes of PSD plots confirm the $1/f^{\alpha}$ behaviour of the noise generated in this manner. Mesh plots of the ECMs corresponding to $\alpha = 1$ and $\alpha = 2$ used to generate the noise sequences in Figure 2.3 can be found in Figure 2.5.

A more efficient way to show the correlation observed in the ECM is to plot the first row (or column) of the ECM, since this indicates the correlation of the first channel with all of the other measurements. Figure 2.6 shows the correlation of channel 1 with other measurement channels for different values of α, again confirming the increase in correlation as low frequency noise components are more heavily weighted with increasing α.

The algorithm proposed here requires specification of the parameters H_0 and ρ, with the

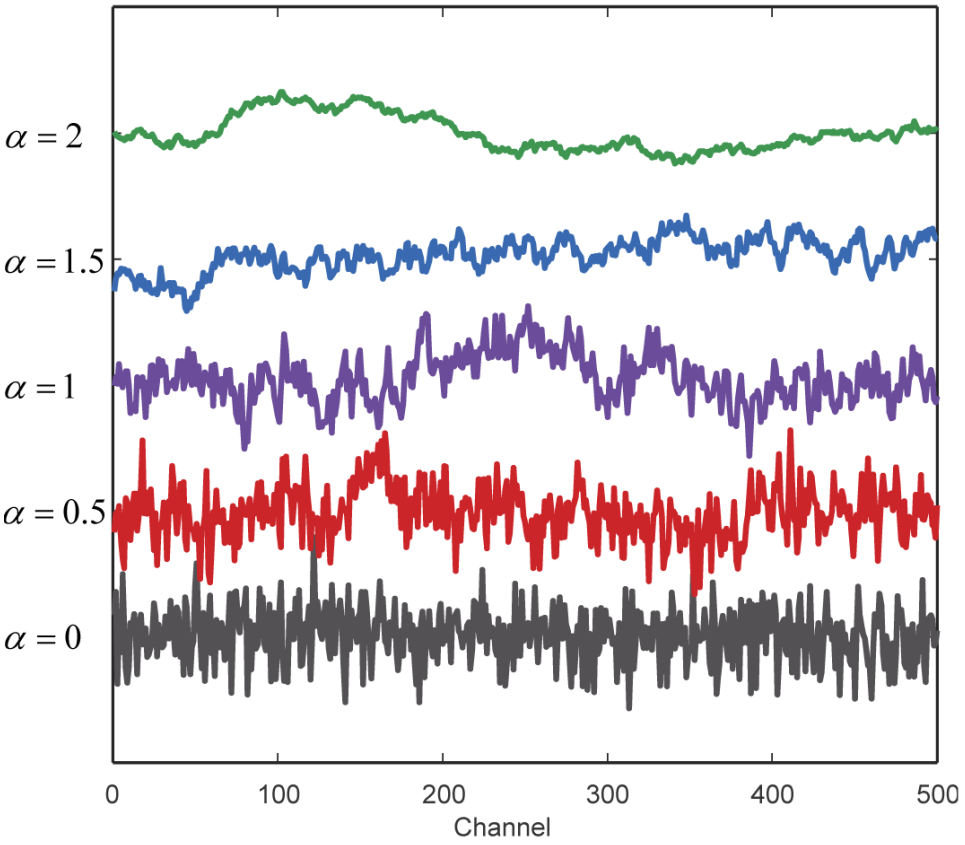

Figure 2.3: Sequences of 500 measurements ($n = 500$, $m = 501$, $\sigma^2 = 1$) simulated using the proposed method for $\alpha = 0$, 0.5, 1, 1.5, and 2. The sequences are offset vertically for clarity.

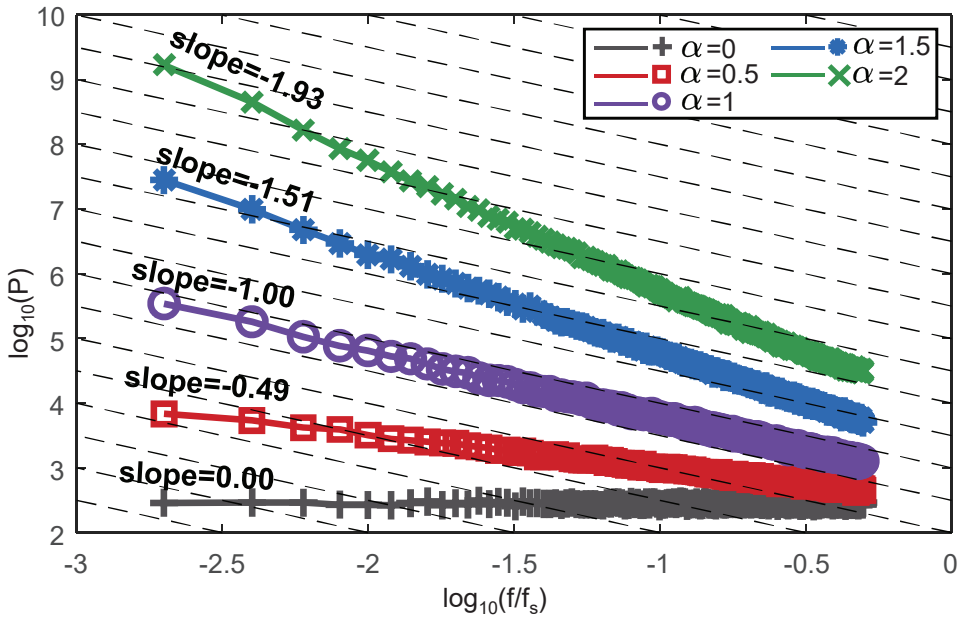

Figure 2.4: Average PSD plots obtained from 1000 realizations of simulated sequences ($n = 500$, $m = 501$) for $\alpha = 0$, 0.5, 1, 1.5, and 2. The dashed lines have a slope of -1 and the plots are arbitrarily shifted vertically for clarity.

former related to the finite DC variance and the latter defining the filter width. Simulations were performed to explore the effects of these parameters on the PSD plot using $n = 500$ and $\alpha = 1$, with 1000 replicates employed to evaluate the PSD. It was found that when H_0 changes significantly from the recommended value of $(6w)^{\alpha/2}$, negative or positive deviations in the PSD plot can result, although small deviations from the recommended value have limited impact, as shown in Figure 2.2(a). Similar behaviour was observed for different values of w and α, confirming our recommendation to assign the value of H_0 as described. On the other hand, the value of ρ is an adjustable parameter. When $\rho < 1$, distortion of the PSD plot away from the ideal is observed, with a flatter response at low frequencies. When $\rho > 1$, the slope remains true to the expected value regardless of the value of ρ, and only a small shift in the vertical position of the curve is evident (see Figure 2.2). However, the effect on the corresponding ECM is more significant, as illustrated in Figure 2.7(a), which shows the calculated channel 1 correlations for various values of ρ with $n = 500$ and $\alpha = 1$. As ρ is increased, the correlations in the measurement errors expand in magnitude across all of the channels. Figure 2.7(b) shows samples of the noise for each of these cases (offset for clarity). Despite the differences in the ECM there are no obvious features that distinguish the identity of the noise sequences because the

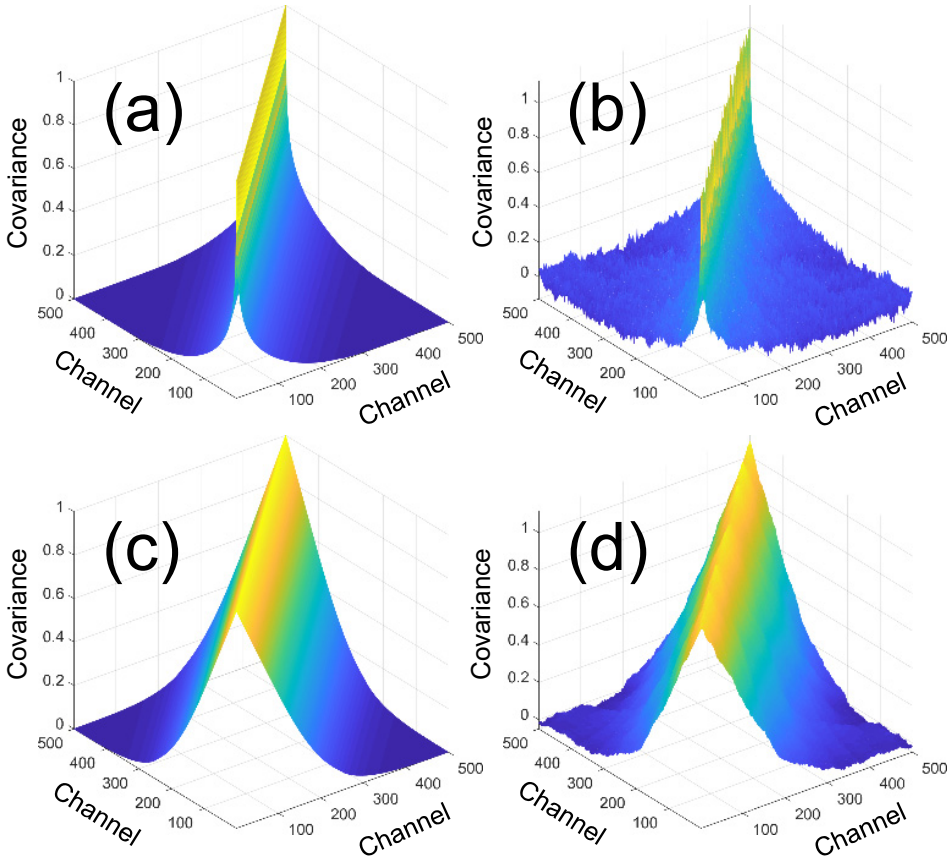

Figure 2.5: (a) Theoretical ECM and (b) experimental ECM for $\alpha = 1$ ($n = 500$, $m = 501$, 1000 replicates) as well as (c) the theoretical ECM and (d) the experimental ECM for $\alpha = 2$ ($n = 500$, $m = 501$, 1000 replicates).

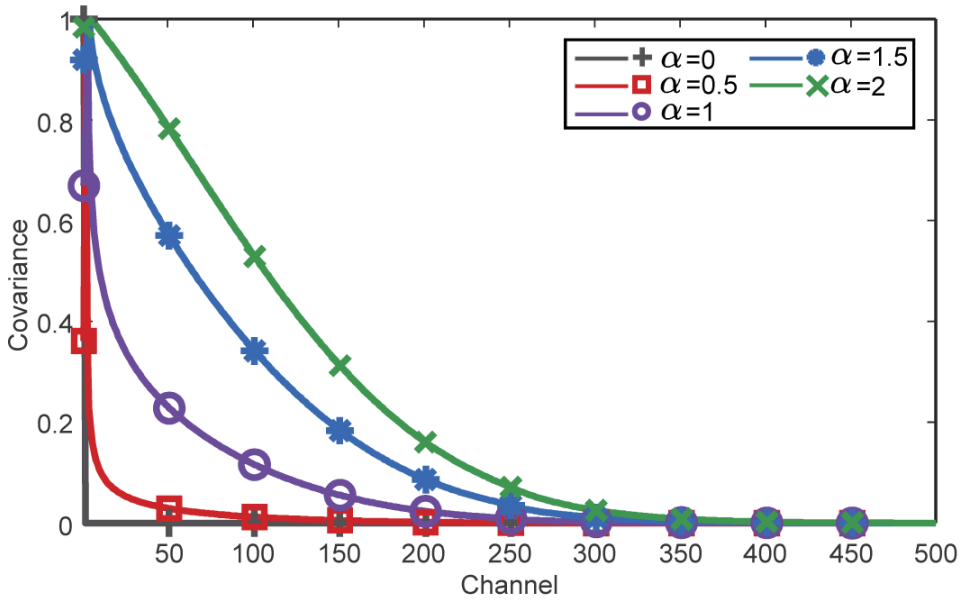

Figure 2.6: The first row of the theoretical ECM for $\alpha = 0$, 0.5, 1, 1.5, and 2 ($n = 500$, $m = 501$).

changes are at very low frequencies that can only be detected by a statistical examination of multiple sequences. The effect of increasing ρ becomes greater as α is increased, but the shape of the ECM remains largely unchanged with increasing n; that is, the same shape is observed but over a larger range of channels.

The reason for the change in ECM as a function of ρ can be understood by looking at the frequency response of the corresponding filter over a wide range. The amplitude response of any symmetric FIR filter with coefficients given by $\mathbf{c}$ at a frequency, f, is given by :[90–92]

$$H(f) = A\cos\phi + B\sin\phi. \tag{2.18}$$

where,

$$\phi = \tan^{-1}(B/A). \tag{2.19}$$

$$A = \sum_{k=-w}^{w} c_k \cos(2\pi k f). \tag{2.20}$$

$$B = \sum_{k=-w}^{w} c_k \sin(2\pi k f). \tag{2.21}$$

Figure 2.7(c) shows a log-log plot of the frequency response for the filters corresponding

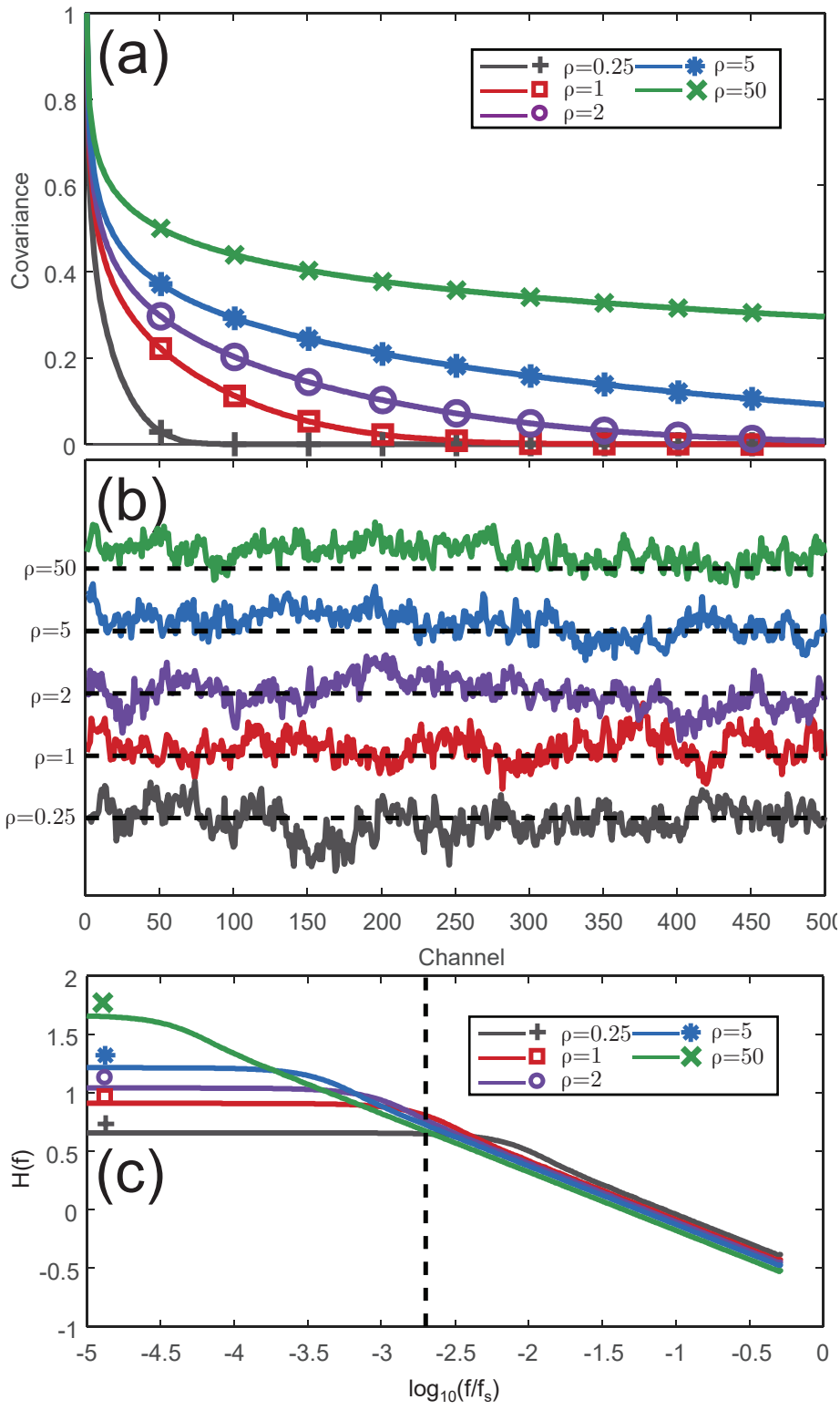

Figure 2.7: (a) The first row of the ECM ($\alpha = 1$, $n = 500$) for various values of ρ (0.25, 1, 2, 5, and 50), (b) the corresponding noise samples (offset for clarity) and (c) the frequency response for the filters constructed using various values of ρ (0.5, 1, 2, 5 , 50).

to Figures 2.7(a) and (b) extending to low frequency regions. The vertical line indicates the lowest non-zero frequency available in the PSD plots. The behaviour below this frequency shows how low frequencies are attenuated by the filter, with each one showing a plateau approaching zero frequency. As ρ increases, a greater fraction of the low frequency region is incorporated into the filter. Since the low frequencies are associated with long-range correlation, this explains why the ECM shows greater correlation as the filter width is increased, even though the slope of the PSD plot does not change in the observable region. Without knowing the behaviour of the analytical measurement system at low frequencies, it is impossible to anticipate which (if any) of the ECMs represents the "true" characteristics, but for simulation purposes, adjustment of ρ allows these features to be manipulated to explore the limits of the data analysis procedure.

Application of the power noise simulation in the examples that follow requires the specification of three parameters: the number of points (n), the power factor (α), and the filter width multiplier (ρ). For these examples, the parameters have been selected to demonstrate the utility of the noise simulation, but are not intended to represent recommended values.

2.5.2 NMR Spectral Simulation

The principal objective of this work is to create a method to generate power law noise through the ECM for the purpose of simulating analytical measurements and testing data analysis strategies. While other methods can be used to generate power law noise, the main advantage of using the ECM is that it can be easily added to ECMs representing other noise contributions (white noise, offset noise, multiplicative offset noise, etc.) to measure the overall impact under various conditions. To demonstrate this, NMR data are used here to test the hypothesis that maximum likelihood principal components analysis (MLPCA) should perform better than PCA under non-*iid* noise conditions. MLPCA methods take the measurement error structure into account in estimating the subspace of the data; in essence, they try to separate the noise variance from the chemical variance in an optimal way. A drawback with this method is that it is difficult to evaluate to what extent MLPCA improves results under different conditions since reliable ECMs (which MLPCA requires) are generally unavailable for experimental data sets due to limited replication. This is just one example of where simulation is a critical tool in assessing the efficacy of proposed new techniques and, by implication, the value of better understanding analytical measurement errors.

For the simulation here, $1/f^{\alpha}$ ($\alpha = 1$ and $\alpha = 2$) noise sequences were added to the reference NMR data. Pink and brown noise are fairly common in spectroscopy, where source components are often subject to these kind of variations. (It is not implied here that the NMR data used are subject to power noise, they were simply used as a model data set.) To make the simulation more interesting and demonstrate the additivity of noise sources, white noise was also added to the data.

To demonstrate the value of the noise simulation method, the standard deviation of the $1/f^{\alpha}$ noise was increased from 0 to 15% of the maximum signal in steps of 0.5% while the level of white noise remained fixed at 1% of the maximum signal. The performance of PCA and MLPCA was evaluated at each noise level used by examining the angle between the reference component subspace (based on reference NMR data) and the subspace estimated by the decomposition methods. With this approach, smaller subspace angles indicate a better fit to the reference, which is important in applications such as multivariate curve resolution and multivariate calibration. To obtain statistically meaningful results, these angles were averaged over 50 runs (same data structure, different noise realizations). The value of ρ was kept at 1 for all simulations. The results of this study for both $\alpha = 1$ and $\alpha = 2$ are shown in Figure 2.8.

As expected, increasing the amount of $1/f^{\alpha}$ noise added to the data hinders the ability of both methods to estimate the reference subspace, but as anticipated, MLPCA provides a more reliable subspace estimate than PCA under all conditions. This is observed for both $\alpha = 1$ and $\alpha = 2$, but is more drastic for $\alpha = 2$. This can be rationalized by considering the increase in correlation of the noise being added to the system for $\alpha = 2$, relative to $\alpha = 1$. PCA assumes that there is no correlation in the noise structure of the data and its ability to estimate the reference subspace is impacted more by noise with a more prominent correlation structure. Although the subspace angle profile obtained using MLPCA for both cases is similar, there are some noticeable differences. At the maximum addition of noise (15% RSD), MLPCA seems to obtain a lower subspace angle (~ 10 degrees) for the $\alpha = 2$ case compared to the $\alpha = 1$ case (~ 25 degrees). This most likely has to do with the correlation of the noise with respect to the pure component spectra of the reference data.

2.5.3 Baseline Correction Simulation

The simulation of power law noise is useful not only for testing complex multivariate analysis methods, but also for the evaluation of simpler data treatments such as baseline

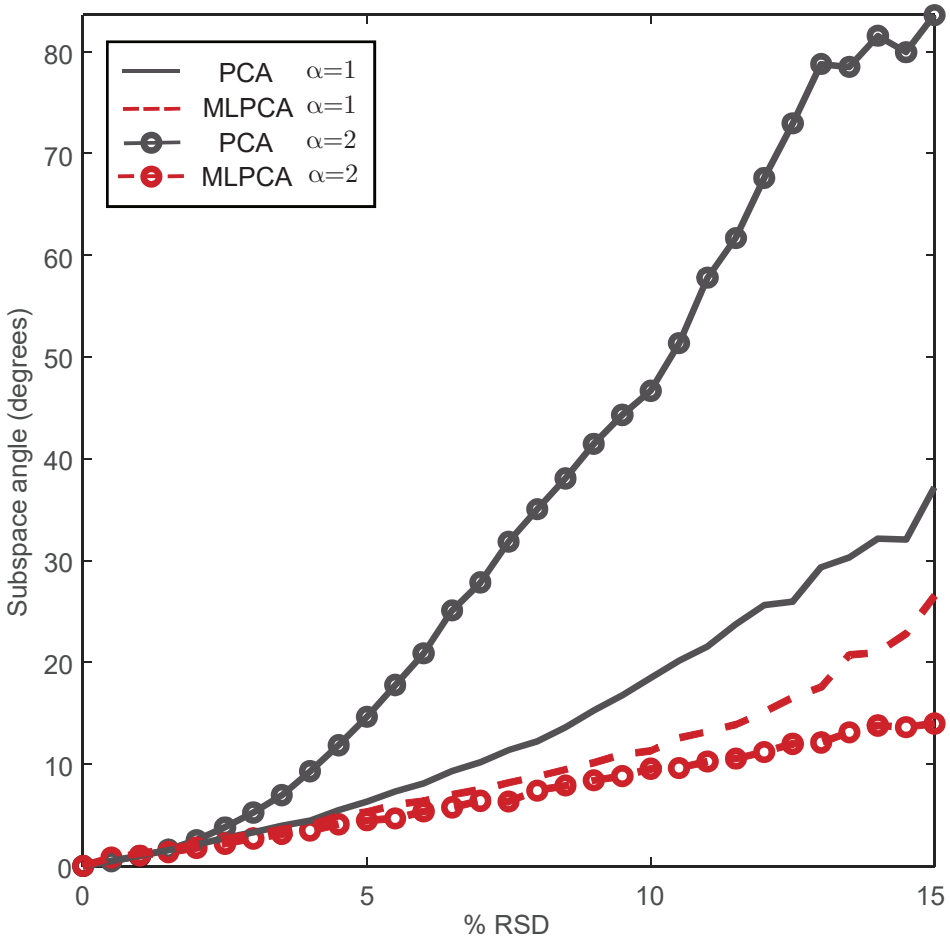

Figure 2.8: Average subspace angle (50 replicates) between the reference subspace and the subspace estimated by both PCA and MLPCA as a function of % RSD of $1/f^{\alpha}$ noise for $\alpha = 1$ and $\alpha = 2$.

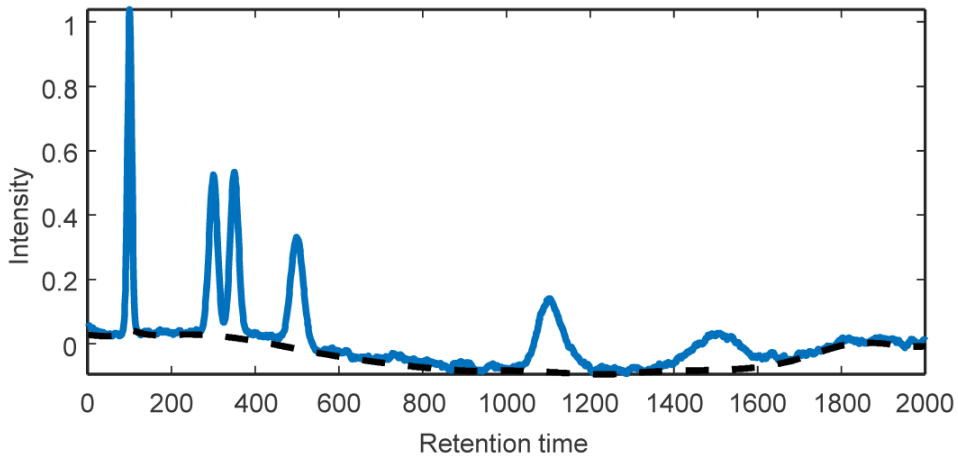

Figure 2.9: Simulated chromatographic vector vector with $1/f^2$ noise (5% RSD) and an estimate of its baseline obtained using AsLS ($\lambda = 9.5 \times 10^4$).

correction. Many baseline correction algorithms have been developed, but the generation of realistic stochastic baselines for testing that do not have a defined functional form is a challenge. Power law noise typically exhibits low frequency variations consistent with baseline drift and was used to test asymmetric least squares (AsLS),[89] a popular baseline correction method, in this simulation. The ability of the AsLS smoothing baseline correction algorithm to estimate the baseline of the simulated chromatographic data set with 5% RSD (relative to the max signal in the reference data) $1/f^2$ noise was investigated. Due to the adjustable parameter λ in the algorithm, which determines the smoothness of the baseline estimate, multiple simulations were performed to determine an optimal value by inspection ($\lambda = 9.5 \times 10^4$). Figure 2.9 shows typical results for the AsLS baseline estimation after brown noise was added to the chromatographic signal.

The AsLS algorithm is able to estimate the low frequency fluctuations in the baseline introduced by the $1/f^2$ noise. However, small deviations are observed in higher frequency fluctuations. This can be rationalized considering the main variation introduced by the $1/f^2$ noise is in the low frequency domain.

2.6 Conclusions

In this work, a method for simulating $1/f^\alpha$ noise, which can easily be extended to simulate multiplicative $1/f^\alpha$ noise has been described. The proposed method is based on scaling and rotation of *iid* normal noise by a projection obtained from the singular value decomposition

of the desired noise ECM which, in the case of $1/f^{\alpha}$ noise, is obtained from an FIR filter. To generate the ECM, the parameters specified are the number of measurement channels (n), the power factor (α) and the filter width parameter (ρ). The last parameter can be used to adjust the influence of low frequency noise on the ECM, but does not affect the slope of the PSD plot for the noise generated. A major advantage of simulating $1/f^{\alpha}$ noise via the ECM is that it can be simulated in the presence of other analytical measurement errors by assuming an additive ECM model. The quality of the $1/f^{\alpha}$ noise produced using this method was assessed by inspection of the PSD for various values of α and was found to closely match the theoretical values. To demonstrate the utility of this method, a simulation was performed where increasing amounts of $1/f^{\alpha}$ noise was added to a known three-component data set. The angle between the reference subspace and the subspace predicted by both PCA and MPLCA was calculated as a function of the increased $1/f$ noise. MLPCA, which includes information on the ECM in its analysis, was found to give better subspace estimations than that of PCA. To further demonstrate the utility of the proposed method, the ability of the AsLS smoothing baseline correction algorithm to estimate the baseline of simulated chromatographic data was assessed by using $1/f^2$ noise to simulate random low frequency baselines. It is anticipated that there are many other examples of analytical signal processing methods where the ability to simulate power law noise will greatly benefit the evaluation of procedures.

CHAPTER 3

NOISEGEN - ANALYTICAL MEASUREMENT ERROR SIMULATION SOFTWARE *

This chapter presents analytical measurement noise simulation software in the form of a MATLAB toolbox. Through the use of this easy-to-use software, multivariate noise structures that imitate those observed in real analytical measurements can be simulated for use in evaluating data processing procedures. In its current form, NoiseGen can simulate *iid* normal noise, multiplicative noise (shot noise, or any other proportionality), multiplicative offset noise, baseline offset noise, and independent and proportional power law noise. An overview of the simulation algorithm is given and some examples using both the command-line and graphical user interface are presented.

3.1 Introduction

As discussed in Chapter 1 and 2, the accurate simulation of measurement noise plays a crucial role in the development and evaluation of data analysis methods. The assumption of *iid* normal measurement errors has been shown to be a major drawback of popular data analysis methods such PCA[4] when applied to data sets that exhibit heteroscedastic and/or correlated noise.[45,93–95] Applying methods that assume a homoscedastic measurement error structures makes it difficult to differentiate between the meaningful variance in the data

*This chapter is based on the published article: Driscoll, S, Wentzell, P.D. NoiseGen - Noise Simulation Software for Analytical Measurements. *Chemometrics and Intelligent Laboratory Systems*. 189 (2019) 155-160 https://doi.org/10.1016/j.chemolab.2019.04.011. **Contribution to Manuscript** SD performed all calculations, and wrote all drafts of the manuscript with edits by PW. PW supervised the project.

and the variance due to noise. This leads to a sub-optimal estimation of the true chemical subspace. Despite this knowledge, there is still a tendency in the literature to represent measurement error as *iid* normal noise when evaluating data processing methods,[32–34] most likely due to the simplicity in its simulation. That being said, some recent articles have reported the utilization of simulated non-*iid* normal errors. For example, in the assessment of new data processing procedures[96] and in the evaluation of measurement error models.[43] The clear advantage in these studies is that the nature of the simulated measurement noise increases the generality of the evaluation and development process. In other words, the simulation of non-*iid* normal noise provides a more realistic and reliable evaluation/development environment. Unfortunately, these examples are rare in the current analytical chemistry literature. It is clear that there is a demand for comprehensive noise simulation software that is easy to implement and able to simulate the types of noise commonly observed in analytical measurement systems. Therefore, the purpose of this chapter is to introduce freely available software developed in-house as a MATLAB[81] toolbox that meets this demand.

To briefly refresh the reader with the basics of noise simulation, some strategies for simulating some error structures discussed in Chapter 1 are considered. In general, simple noise simulation can be performed using a random number generator, such as those available in most programming languages. For example, multivariate *iid* normal noise can be simulated by drawing numbers from a single normal distribution, $N(0, \sigma^2)$, for each measurement channel in the system. Similarly, heteroscedastic noise can be simulated by allowing the variance of the distribution to change as a function of measurement channel, *i.e.* $N(0, \sigma_i^2)$. For example, systems using photomultiplier or photodiode detectors usually exhibit shot noise, a common type of heteroscedastic noise where the standard deviation of the noise is proportional to the square root of the signal.[97] In this case, the simulation can be done by scaling the standard deviation of the distribution from which the random numbers are being drawn at each measurement channel by the square root of the corresponding value of the signal. Other common types of measurement noise, such as correlated noise, can be simulated by introducing dependent structures in the errors across the measurement channels. For example, this can be done by using an autoregressive process to generate a series of values that each depend on the previous value in the sequence plus a random

component.[79] This type of correlated noise is characteristic of systems that exhibit source-flicker noise, also known as $1/f^{\alpha}$ noise.[62] Although these individual methods provide a way to simulate more realistic individual noise components, they become cumbersome when the simulation of multiple sources of noise is required (*e.g. iid* normal noise in the presence of shot noise), which is characteristic of real measurement errors.

A more general method for noise simulation was reported by Wentzell and coworkers in a study that involved bootstrapping to validate measurement error models.[43] The method requires access to the theoretical error covariance matrix (ECM), a square-symmetric matrix that describes the variance and covariance between all channels in the system, of the noise to be simulated. This ECM is then used to rotate and scale *iid* normal noise via its eigenvalues and eigenvectors. Using this methodology, one only needs to know the theoretical ECM for the type of noise, or combinations of noise, they want to simulate. The ECM offers two main advantages in terms of noise simulation. First, it makes the simulation of multicomponent noise trivial by assuming a simple additive model for the ECM. Second, the simulation algorithm is general enough that any type of noise can be simulated as long as the ECM structure is known. Previous studies have reported how to represent common types of measurement noise using the ECM.[36,44,98] For example, multiplicative offset noise, a type of correlated heteroscedastic noise that randomly shifts the signal up or down proportional to the magnitude of the signal, can be represented as the outer product of the signal multiplied by the relative standard deviation of the noise. Using this information, the NoiseGen software allows for modular input of various types of noise sources, so the only information required from the user is the components of the noise to be simulated and their magnitudes. This is the core of the NoiseGen algorithm which makes the simulation of single and multicomponent noise easy to perform.

3.2 Algorithm

In general, any noise sequence, regardless of its complexity, can be generated if the theoretical ECM, $\mathbf{\Sigma}$, is known. By performing SVD on the ECM to rotate and scale *iid* normal noise, a realization of noise, $\mathbf{e}_{\Sigma}$, consistent with the theoretical ECM can be obtained. This relationship is shown in Equation 3.1, where $\mathbf{S}$ and $\mathbf{V}$ are the matrices that contain the singular values and loadings, respectively, obtained via SVD of $\mathbf{\Sigma}$ and $\mathbf{e}_{iid}$

$(1 \times n)$ is a vector of *iid* normal errors with $\sigma = 1$.

$$\mathbf{e}_{\Sigma} = \mathbf{e}_{iid}\mathbf{S}^{\frac{1}{2}}\mathbf{V}^{\mathrm{T}}. \tag{3.1}$$

This formulation can be generalized to create m realizations of noise simultaneously by redefining the dimensionality of $\mathbf{e}_{iid}$ as $m \times n$, where each row is a realization of *iid* normal noise.

From a user's perspective, it would be cumbersome to input the ECM of the noise sequence they want to generate. Instead, NoiseGen takes the parameters associated with different noise sources as input and constructs the theoretical ECM, $\mathbf{\Sigma}$, as shown in Equation 3.2.

$$\mathbf{\Sigma} = \mathbf{\Sigma}_{\text{IID}} + \mathbf{\Sigma}_{\text{MN}} + \mathbf{\Sigma}_{\text{BO}} + \mathbf{\Sigma}_{\text{MO}} + \mathbf{\Sigma}_{\text{PL}} + \mathbf{\Sigma}_{\text{PPL}}. \tag{3.2}$$

The following subsections will define and describe each of the terms in Equation 3.2.

3.2.1 Independent and Identically Distributed from a Normal Distribution Noise

Noise which is independent and identically distributed with a normal distribution (*iid* normal) noise, or white noise, is the most commonly assumed type of noise in chemical data. This noise is defined by having zero correlation among all measurement channels and uniform variance. In terms of the ECM, white noise shows up as a constant and uniform diagonal term which can be modeled using Equation 3.3.

$$\mathbf{\Sigma}_{\text{IID}} = \alpha_1^2 \text{diag}(\mathbf{a}_1). \tag{3.3}$$

In Equation 3.3, $\mathbf{a}_1$ is a vector of ones $(n \times 1)$ and α_1 represents the standard deviation of the noise. Figure 3.1 shows the theoretical ECM for some *iid* normal noise ($\alpha_1 = 0.1$) and the addition of a realization of this noise to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a maximum height of 1).

3.2.2 Multiplicative Noise

Multiplicative noise is a type of noise where the standard deviation of the error is proportional to the signal intensity. This type of noise is uncorrelated, but is heteroscedastic. This can be modeled in the ECM by creating a matrix where the diagonal elements are equal to

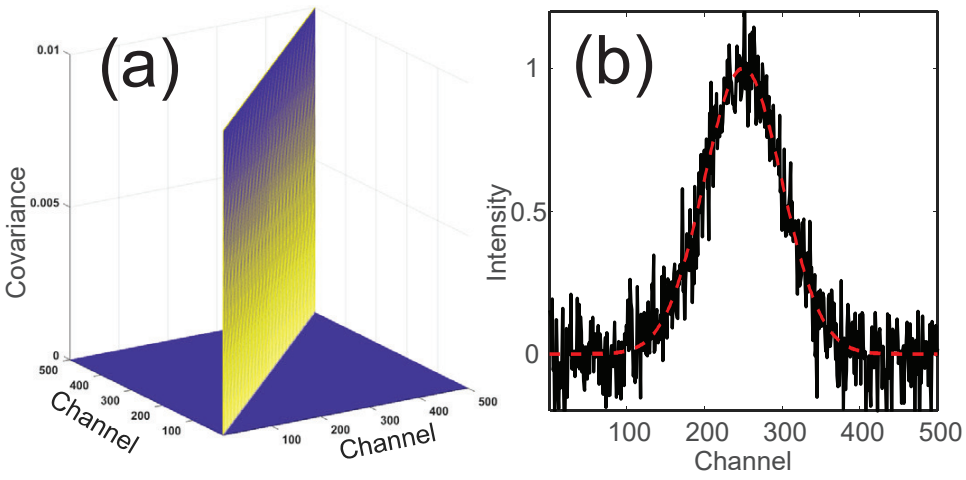

Figure 3.1: (a) The ECM model of some *iid* normal noise ($\alpha_1 = 0.1$) and (b) this noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

a representative signal vector raised to some power, as show in Equation 3.4.

$$\mathbf{\Sigma}_{\text{MN}} = \alpha_2^2 \text{diag}(\mathbf{a}_2). \tag{3.4}$$

In Equation 3.4, $\mathbf{a}_2$ is a representative signal vector with its elements raised to the power of $2\gamma_1$. If $\gamma_1 = 1$, this is true proportional noise and α_2 represents the noise RSD. However, γ_1 can generally take on other values, with $\gamma_1 = 0.5$ (shot noise) being the most common. Shot noise arises from counting statistics at detectors that measure quantized signals in time (*e.g.* electron multipliers). These randomly arriving signals follow the Poisson distribution and result in a proportionality between the variance of the error and the signal. Figure 3.2 shows the theoretical ECM for shot noise with $\alpha_2 = 0.1$ relative to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1) along with a realization of this noise added to the same Gaussian signal.

3.2.3 Baseline Offset Noise

Sometimes referred to as additive noise, baseline offset noise arises from the signal shifting up or down. A common example of this type of error is changes in cell position between measurements, but it can also result from instrument drift. This results in a random offset in the signal which corresponds to a flat non-zero ECM. This can be modeled using Equation

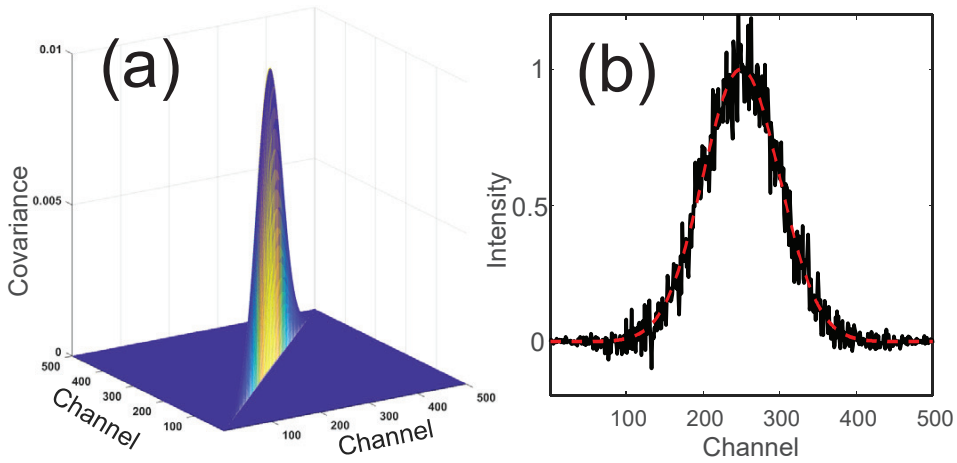

Figure 3.2: (a) The ECM model of shot noise ($\gamma_1 = 0.5$, $\alpha_2 = 0.1$) and (b) this noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

3.5.

$$\mathbf{\Sigma}_{\text{BO}} = \beta_1^2 \mathbf{b}_1 \mathbf{b}_1^{\text{T}}. \tag{3.5}$$

In Equation 3.5, $\mathbf{b}_1$ is a vector of ones ($n \times 1$) and β_1 represents the standard deviation from that source. To illustrate this, Figure 3.3 shows random offset noise ($\beta_1 = 0.1$) of a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1) and the corresponding ECM.

3.2.4 Multiplicative Offset Noise

Multiplicative offset noise is similar to baseline offset in that the impact of this noise arises from a shift of the signal up or down, but in this case the effect is multiplicative and not constant over all channels. This type of noise introduces an offset that is proportional to the signal and is commonly observed in near-infrared (NIR) reflectance measurements due to path length variation. Equation 3.6 can be used to model multiplicative offset noise.

$$\mathbf{\Sigma}_{\text{MO}} = \beta_2^2 \mathbf{b}_2 \mathbf{b}_2^{\text{T}}. \tag{3.6}$$

As with proportional noise $\mathbf{b}_2$ is a representative signal vector raised to the power of γ_2. Most often, $\gamma_2 = 1$ and β_2 will represent the RSD of the multiplicative offset, but other values can be used. Figure 3.4 shows the theoretical ECM for multiplicative noise ($\beta_2 = 0.1$, $\gamma_2 = 1$) relative to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1) and 10 realizations of this noise added to the same Gaussian signal.

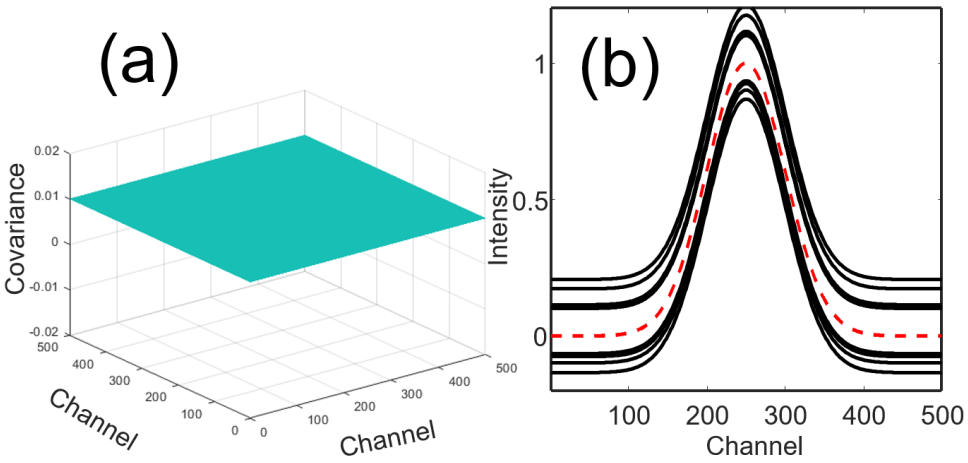

Figure 3.3: (a) The ECM model of baseline offset noise ($\beta_1 = 0.1$) and (b) 10 realizations of this noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

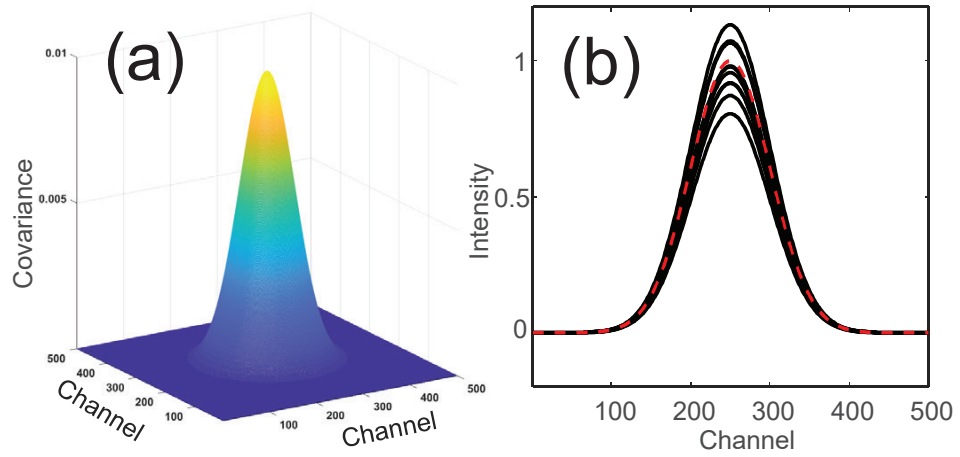

Figure 3.4: (a) The ECM model of multiplicative offset noise ($\gamma_2 = 1$, $\beta_2 = 0.1$) and (b) 10 realizations of this noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

3.2.5 Power Law Noise

Power law noise, also called $1/f^{\alpha}$ noise or source-flicker noise, is a type of low frequency noise observed in almost all fields of science. In analytical chemistry, this type of noise is thought to originate in instrumental components such as flames and ion sources. It is defined in terms of the power spectral density (PSD), which describes the power of a signal, P, as a function of frequency, f. The α value in $1/f^{\alpha}$ is equal to the negative slope of the line formed when plotting $\log P$ vs $\log f$. As α increases, more power is observed in the low frequency components of the noise (*i.e.* more correlation among channels). Values that are of interest and observed in analytical chemistry are $\alpha = 1$ and $\alpha = 2$, sometimes called pink and brown noise, respectively. These noise types are especially worrisome in analytical chemistry because measured chemical signals can often be found in the same frequency domain.

The work completed in Chapter 2 allows for a theoretical ECM of power law noise using a finite impulse response filter.[98] The results of this work allow for the simulation of $1/f^{\alpha}$ noise with α as an adjustable parameter. In addition, an important finding of this work is that it is possible to obtain $1/f^{\alpha}$ noise using a theoretical ECM that depends on not only α, but ρ as well, which is the ratio of the length of the filter to the number of channels. Practically, these are the only two value that need to be defined to simulate power law noise using this method, where an increase in α changes the type of power law noise and an increase in ρ increases the correlation within a type of power law noise. The model ECM can be constructed using Equation 3.7, where λ_1 is the standard deviation of the power law noise and $\mathbf{C}_1$ is the $1/f^{\alpha}$ theoretical ECM designed with unit variance. It should be noted that $\mathbf{C}_1$ is both a function of α (defined in NoiseGen as γ_3 for notational consistency) and ρ.

$$\mathbf{\Sigma}_{\text{PL}} = \lambda_1^2 \mathbf{C}_1. \tag{3.7}$$

Figure 3.5 shows an example of a theoretical $1/f^2$ noise ECM ($\alpha = 2$, $\rho = 1$, $\lambda = 0.1$) and the corresponding noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

In systems that exhibit shot-noise like characteristics, it is expected that the $1/f^{\alpha}$ noise is multiplicative in nature. To simulate this type of noise, Equation 3.8 can be used, where $\mathbf{C}_2$ defined as the Hadamard (element by element) product of the $1/f^{\alpha}$ noise matrix (specified

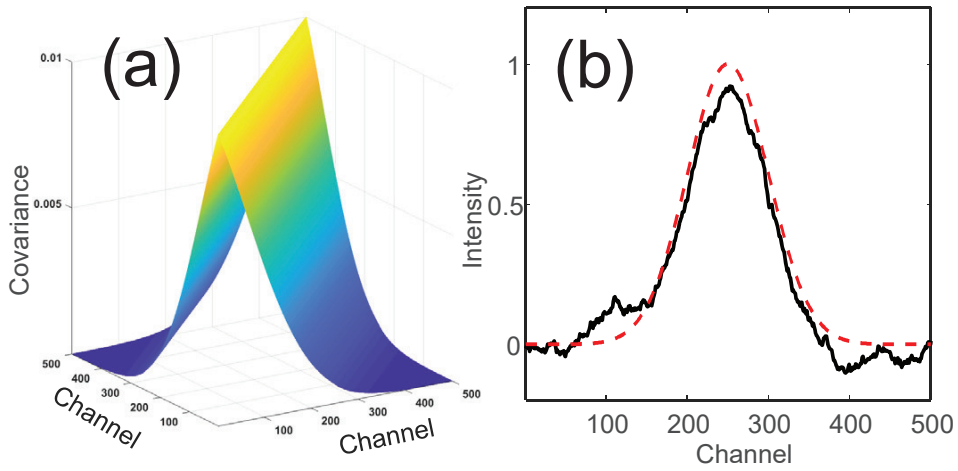

Figure 3.5: (a) The ECM model of $1/f^2$ noise ($\alpha = 2$, $\rho = 1$, $\lambda = 0.1$) and (b) the corresponding noise added to a Gaussian signal ($\mu = 250$, $\sigma = 50$, and a max height of 1).

by $\mathbf{C}_1$) with the outer product of the representative signal vector.

$$\mathbf{\Sigma}_{\text{PPL}} = \lambda_2^2 \mathbf{C}_2. \tag{3.8}$$

3.2.6 The Overall ECM Model

Using the model expressions for the individual types of noise we want to model, the individual terms in Equation 3.2 that describe the additive ECM model can expanded, as shown in Equation 3.9,

$$\mathbf{\Sigma} = \alpha_1^2 \text{diag}(\mathbf{a}_1) + \alpha_2^2 \text{diag}(\mathbf{a}_2) + \beta_1^2 \mathbf{b}_1 \mathbf{b}_1^{\text{T}} + \beta_2^2 \mathbf{b}_2 \mathbf{b}_2^{\text{T}} + \lambda_1^2 \mathbf{C}_1 + \lambda_2^2 \mathbf{C}_2. \tag{3.9}$$

With the model ECM formulated in this fashion, the only input from the user is the length of the noise sequence to be generated, how many sequences to generate, and the type of noise and the parameters (α, β, γ) associated with each type. Of course, for proportional components (*i.e.* $\mathbf{\Sigma}_{\text{MO}}$, $\mathbf{\Sigma}_{\text{MN}}$, and $\mathbf{\Sigma}_{\text{PPL}}$), a reference signal needs to be provided. Using this information, the model ECM is constructed and SVD is performed along with the generation of a white noise sequence. Equation 3.1 is then applied to generate the simulated noise sequence consistent with $\mathbf{\Sigma}$. A general flowchart for this methodology is shown in Figure 3.6.

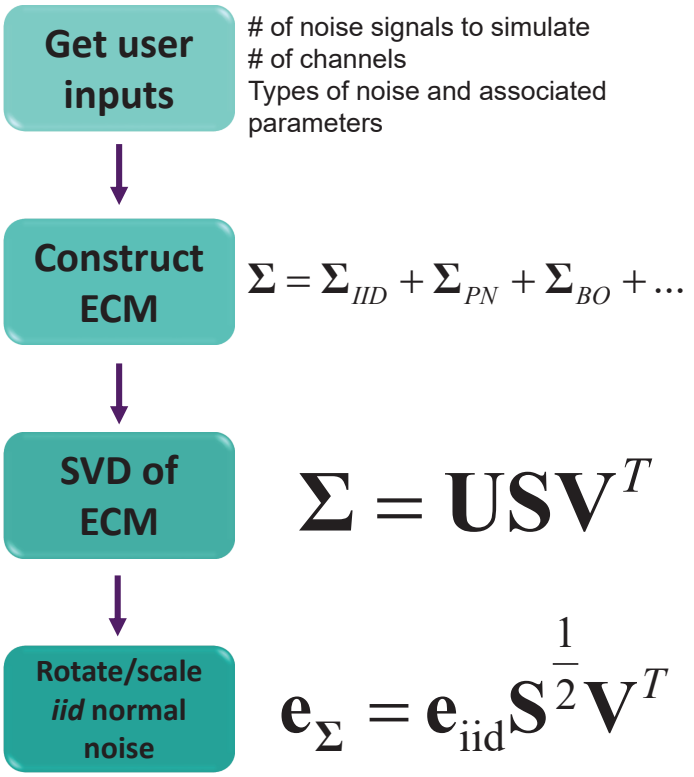

Figure 3.6: General flowchart for the NoiseGen algorithm.

3.3 Software Specifications and Dependencies

NoiseGen is developed as a MATLAB toolbox in MATLAB release 2018a (MathWorks, Natick, MA).[81] The toolbox contains two files, *NoiseGen.m* and *NoiseGenGUI.mlapp*. Both files can be found at the following public repository: https://github.com/S-Driscoll/NoiseGen. Appendix B contains code for the command line implementation of NoiseGen. The *NoiseGen.m* file is the main function called for performing the noise simulation. The explicit inputs to the function are the number of channels, *chan*, and the number of noise sequences to be generated, *reps*. Additional variable-length input arguments (*varargin*) are introduced to allow for modular input of noise components and parameters. Input to the NoiseGen function is structured such that the user only needs to specify the necessary parameters for the noise they wish to simulate. Some types of noise, such as power law noise, have additional parameters that can be changed, but defaults are used if they are not specified. All variables and their descriptions, types and defaults are listed in Table 3.1. The returned variables, written in bold italics here for clarity, are the matrix of the simulated noise, $\boldsymbol{X}$ $(reps \times chan)$, the theoretical ECM, $\boldsymbol{COVM}$ $(chan \times chan)$, and the

Table 3.1: NoiseGen variables and their descriptions and default values for the command-line interface.

Noise Type	Description	Parameters	Default Value
chan	Number of channels	-	Required
reps	Number of replicates	-	Required
'BO'	Baseline offset noise	SD	Required
'MO'	Multiplicative offset noise	RSD	Required
		Reference signal	Required
		Proportionality	1
'IID'	White noise	SD	Required
'MN'	Multiplicative noise	RSD	Required
		Reference signal	Required
		Proportionality	1
'PL'	Power law $(1/f^{\alpha})$ noise	SD	Required
		α	1
		ρ	1
'PPL'	Proportional power law noise	RSD	Required
		Reference signal	Required
		α	1
		ρ	1

ECM calculated based on the number of replicates, ***COV*** $(chan \times chan)$.

The call format of the NoiseGen function in MATLAB function syntax is as follows: [***X***, ***COV***, ***COVM***] = NoiseGen(chan, reps, varargin). Where *varargin* takes the variable input arguments, such as 'MO' or 'BO'. When specifying a noise type the user must make sure to specify the appropriate parameters immediately after or the default values will be used. Structuring the function in this way allows for modular input of noise types. For example, [***X***, ***COV***, ***COVM***] = NoiseGen(100, 30, 'MN', 0.05, y, 'BO', 0.1) would generate 30 noise sequences 100 channels in length that have (theoretically) a multiplicative (proportional) noise RSD of 0.05, and baseline offset noise with a standard deviation of 0.1. It is important to specify the reference signal vector for the multiplicative noise (in this case as y (1×100), which would hold the reference signal in the MATLAB workspace) because this type of noise depends on the reference signal. The order of the noise types requested in the function call does not matter, so calling the NoiseGen function as [***X***, ***COV***, ***COVM***] = NoiseGen(100, 30, 'BO', 0.1, 'MN', 0.05, y) is equivalent to the call in the previous example. Although the order of types of noise does not matter, the order of

the parameters for each type of noise does matter. Each noise type has a required number of parameters and optional parameters that have defaults. This information is also found in Table 3.1. For example, in the previous call if we wanted to change the proportionality of the multiplicative noise to the reference signal to be 0.5 (shot noise), then the structure of the function call would be: [***X***, ***COV***, ***COVM***] = NoiseGen(100, 30, 'BO', 0.1, 'MN', 0.05, y, 0.5). Since NoiseGen builds the theoretical ECM from the individual ECMs it is possible to simulate multiple types of the same noise type. For example, to generate 100 samples of noise 500 channels in length that exhibits both pink and brown power law noise ($\alpha = 1$ and $\alpha = 2$) each with a SD of 1, the structure of the input would be: [***X***, ***COV***, ***COVM***] = NoiseGen(500, 100, 'PL', 1, 1, 'PL', 1, 2).

For proportional components ('MO','MN', and 'PPL'), different reference signals can be used for each replicate (*reps* $\times$ *chan*) to mimic a constant proportional noise RSD, but varying spectral profiles in a data set. It is even possible to have individual proportional noise RSDs and individual reference signals for each replicate by defining a vector of RSDs ($1 \times$ *reps*) and a matrix of reference signals (*reps* $\times$ *chan*). In this case, the output ECMs (theoretical and experimental) have dimensionality of *chan* $\times$ *chan* $\times$ $reps$. This proves useful when adding proportional components to an already defined set of data.

As noted above, a version of NoiseGen with a graphical user interface (GUI) is also available in the toolbox (*NoiseGenGUI.mlapp*). It is developed as a MATLAB AppDesigner application and requires MATLAB release 2016b or later to run. A general overview of the user interface and an example of using the GUI in practice is given in the next section. The GUI version is algorithmically the same as the command-line version, but the input environment is different. An annotated picture of the GUI layout is shown in Figure 3.7, where two screen captures of NoiseGen with descriptions of each of the components are shown. The output from the GUI is written to the MATLAB workspace with the same variable names as the command line interface by default; the output simulated sequences are stored in the matrix ***X*** $(reps \times chan)$, and the theoretical ECM and the ECM based on the number of replicates are stored in ***COVM*** and ***COV***, respectively, by default in the workspace. Although, these variable assignments can be changed in the GUI. A limitation of the current version of the NoiseGen GUI is that multiple contributions of noise of the same type is not supported.

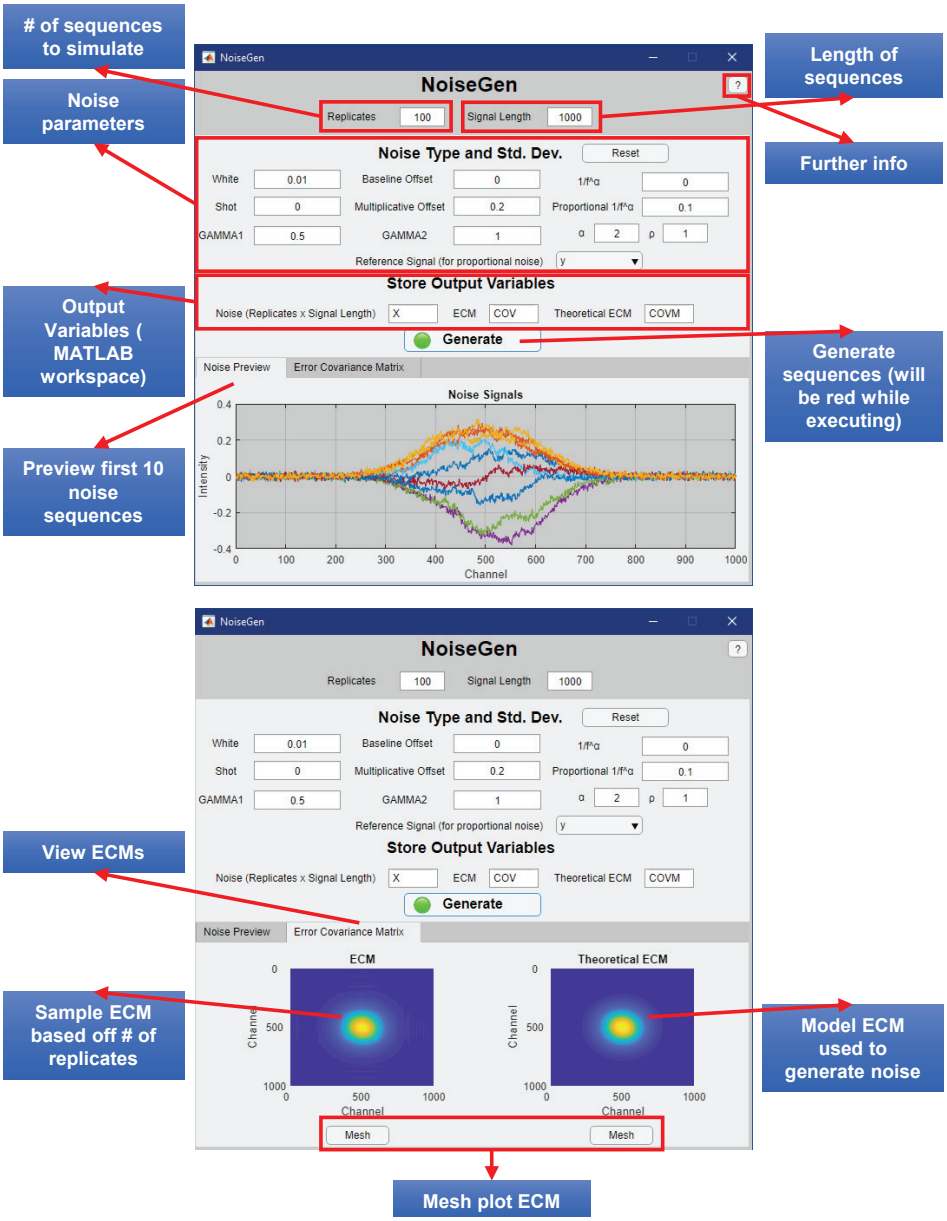

Figure 3.7: The NoiseGen GUI.

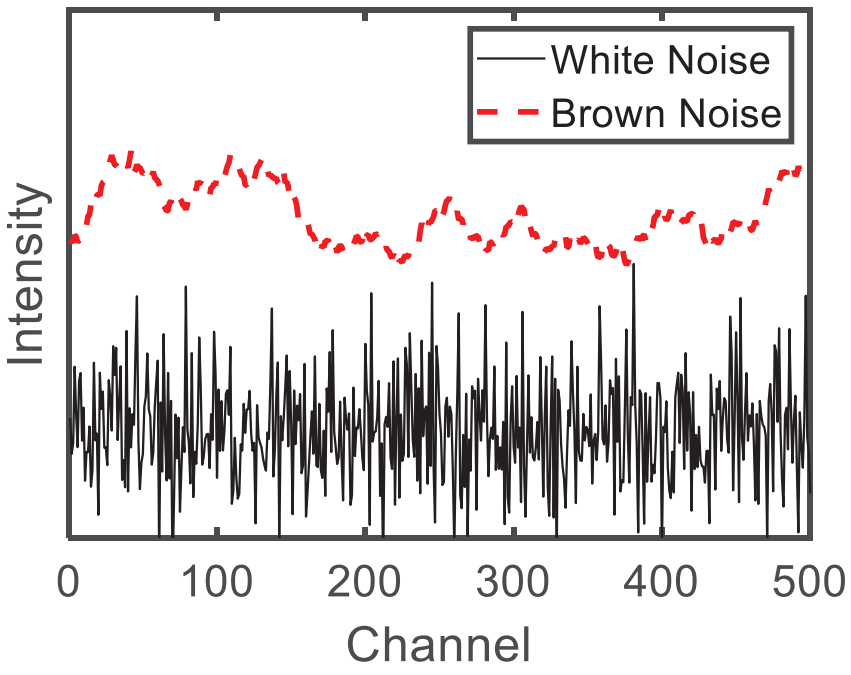

Figure 3.8: An *iid* normal (*i.e.* white noise) sequence 500 channels in length (black solid line) and a $1/f^2$ (*i.e.* brown noise) sequence 500 channels in length (red dashed line).

3.4 Example Simulations

3.4.1 Command-line Noise Simulation

In this example, the NoiseGen function is used to investigate the impact brown noise ($1/f^2$) has on the visualization of PCA scores for a simulated data set designed to cluster four different classes. Brown noise is a class of power law noise characterized in the frequency domain by having a decaying power spectral density that falls off with a slope of -2 on a log-log plot of power vs frequency. This results in lower frequencies having more power which corresponds to high correlation between adjacent measurement channels. The origin of the noise is not fully understood, as with all power law related noise, but is most likely introduced from source components in instruments (flames, plasmas, etc.). This is a concern for most analytical measurements as the signals containing chemical information are usually in the same frequency domain. A sample signal of brown noise is shown in Figure 3.8 where a *iid* normal noise sequence (white noise) is plotted for comparison. It is easy to see the non-zero correlation in the brown noise, made clear by the "slow-moving" features that almost look like informative peaks.

As PCA assumes *iid* normal errors for maximum likelihood estimation, the brown noise should cause the subspace estimated by PCA to be inaccurate. To test both the error-free case and the case of brown noise, a simulated data set was constructed as follows. The data set consisted of 400 objects equally divided into four classes. The center of these classes were designed such that in a two-dimensional space they were located on the vertices of a square centered at the origin with the length of each side equal to unity. The objects were randomly clustered around the centers according to a symmetric bivariate distribution with a standard deviation of 0.20. This data set is shown in Figure 3.9 (A). The data set was then rotated into a 500 dimensional space using a random rotation matrix, creating a 400×500 multivariate data matrix. PCA was then applied to the error-free data, shown in Figure 3.9 (B), and the data tainted by brown noise ($\sigma = 0.30$), shown in Figure 3.9 (C). To generate the brown noise, the NoiseGen function was called as follows: [***X***, ***COV***, $\sim$]=NoiseGen(500, 400, 'PL', 0.30, 2) in the MATLAB command window.

To demonstrate the benefit of including error information in the subspace estimate, maximum likelihood principal component analysis (MLPCA) was applied to the same data set using the estimated covariance matrix and the corresponding scores were plotted, as shown in Figure 3.9 (D). MLPCA, developed by Wentzell and coworkers, includes measurement error information in the subspace estimation procedure via the ECM and should give a better subspace estimate as long as the ECM is known. This ECM, which was assigned to ***COVM*** when NoiseGen was called to generate the brown noise, was fed into the MLPCA algorithm (case d - equal row ECMs) and the subspace was determined. These results show that brown noise can severely impact a simple clustering problem when applying PCA. The assumption that the ECM is simply the identity matrix multiplied by a constant (the variance) causes an inaccurate representation of the data and, in this case, the inability to correctly cluster all four classes.

3.4.2 GUI Noise Simulation

The NoiseGen GUI can be used to quickly visualize the behavior of different combinations of noise types. In this example, the NoiseGen GUI is used to simulate noise comprised of shot noise, proportional brown noise, also known as $1/f^{\alpha}$ noise, and multiplicative offset noise. It is often hard to distinguish between multiplicative offset noise and proportional $1/f^{\alpha}$ noise due to their similar impact on the ECM. The NoiseGen GUI can be used to quickly visualize this impact in both the theoretical ECM and the ECM calculated from

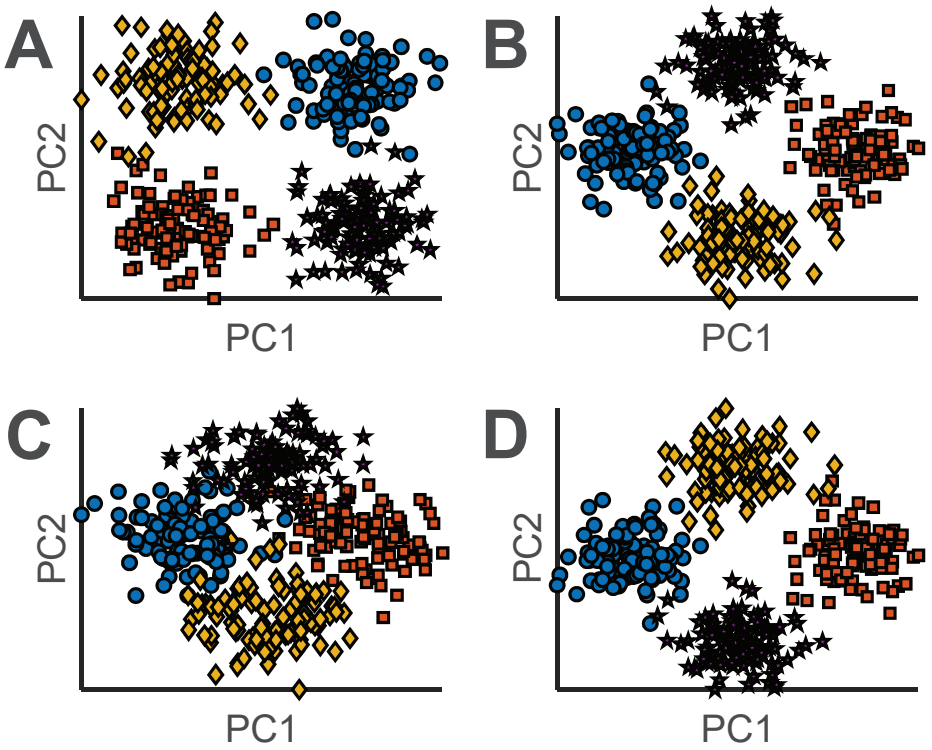

Figure 3.9: (A) Simulated data consisting of four classes, each clustered around the vertices of a square (symmetrical bivariate distribution with $\sigma = 0.20$). These data were then rotated into a 500 dimensional space to make them multivariate. (B) PCA scores plot of the error-free data (C) PCA scores plot of the data tainted with brown noise ($1/f^2$ noise, $\sigma = 0.3$). (D) MLPCA-D scores plot of the tainted data using the ECM model obtained from NoiseGen.

the the simulated noise sequences. Here, I explore this qualitative change as a function of the ratio between proportional pink noise and multiplicative offset noise in the presence of shot noise.

To start the NoiseGen GUI in MATLAB type *NoiseGenGUI* in the command window (note that this is case sensitive). Note that either the working directory needs to contain the *NoiseGenGUI.mlapp* file or it needs to be added to the search path. The window when the app is called is similar to that shown in Figure 3.2, but with the default values for all parameters. Here, the parameters of the noise to be simulated can be defined. In the current case, we first simulate 50 replicates, 1000 channels in length, of a noise sequence that exhibits a theoretical shot noise standard deviation of 0.5, proportional brown noise RSD of 0.5, and multiplicative offset noise RSD of 0.5. Of course, because all three of the noise components in this simulation depend on an input signal, a simulated Gaussian profile that is 1000 channels in length centered on channel 500 with a standard deviation of 100 and a max value of unity is created for demonstration purposes. This is stored as *refsignal* in the MATLAB workspace. This signal can be specified by selecting it from the drop-down menu. Note that this drop-down menu only becomes visible when a proportional noise type is given a value greater than zero. To update the drop-down menu the reset button can be pressed; this will also reset all noise type parameters. A screenshot of the GUI configured with the parameters used in this example is shown in Figure 3.10.

Clicking the button with the green lamp labeled "Generate" will generate noise, as well as the ECMs, with the specified parameters. While simulating, the button text will change to "Working!" and the lamp will turn from green to red, indicating that the program is still running. This feature becomes more noticeable when the size of the output matrix $\boldsymbol{X}$ increases. Of course, this is also dependent on the computer performance. Once complete, the button will reset back to its original state and sample noise sequences will be shown under the "Noise Preview" tab. Clicking the "Error Covariance Matrix" tab will show both the theoretical ECM used to simulate the noise (right) and the ECM created using the number of replicates requested (left). First, a closer visual inspection of the theoretical ECM can be done by clicking the "Mesh" button. This creates a new figure containing a mesh plot of the ECM. To explore the qualitative change of the ECM, five cases that vary the ratio of multiplicative offset noise to proportional brown noise were simulated. The parameters defining these simulations are shown in Table 3.2.

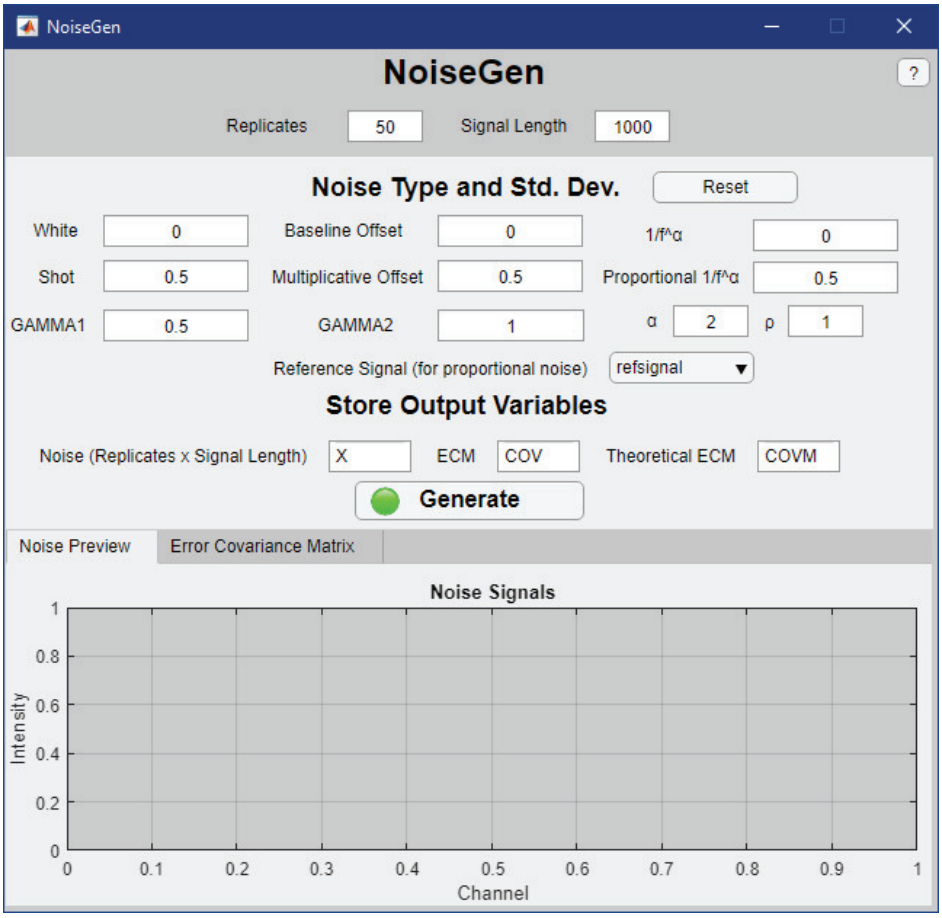

Figure 3.10: NoiseGen GUI example configuration for simulating 50 replicates of a noise sequence 1000 channels in length that has a theoretical shot noise RSD of 0.5, multiplicative offset noise RSD of 0.5, and a proportional brown noise ($1/f^2$ RSD of 0.5. Since all of these noise types depend on a reference signal, the reference signal *refsignal* is selected from the MATLAB workspace.

Table 3.2: RSD Parameters used in NoiseGen to inspect the structure of the model ECM.

Case	Shot Noise	Multiplicative Offset Noise	Proportional Brown Noise
Case (a)	0.50	0.50	0.50
Case (b)	0.50	0.25	0.75
Case (c)	0.50	0.00	1.00
Case (d)	0.50	0.75	0.25
Case (e)	0.50	1.00	0.00

For each case, the ECM variable was assigned in the workspace as a different variable so a quick inspection can be done after all cases were simulated. This can be done in the NoiseGen GUI by changing the name of the "Theoretical ECM" in the "Store Output Variables" panel. The results are shown in Figure 3.11 where all 5 theoretical ECMs defined by the parameters shown in Table 3.2 are shown along with the ECM profile (The $n = 500$ row of the ECM). The results show that while there is only a slight qualitative change in the ECM as the proportion of multiplicative offset noise to proportional brown noise is varied, there is a more noticeable change when the profile of the ECMs are plotted. Comparing Cases (c) and (e), it is clear that multiplicative offset noise has a slightly broad profile compared to proportional brown noise, meaning it introduces more correlation into the noise. The remaining Cases provide insight to how this changes as the ratio of the two types of noise varies. Of course, this analysis could of been performed using the command-line as well, but the GUI provides a more immediate qualitative results when surveying impacts of different noise types on the ECM and the resulting simulated noise sequences.

3.5 Conclusions

NoiseGen provides a structured approach to simulating realistic analytical measurement errors. The present implementation offers both command-line and GUI versions of NoiseGen in the form of a MATLAB toolbox. Multicomponent noise sequences are easily simulated using NoiseGen, where only the magnitude of the noise types and their associated parameters are required as input from the user. The algorithm used for simulating noise revolves around scaling and rotating *iid* normal noise using the scaling and rotation matrices obtained from SVD of the theoretical ECM constructed from user inputs. It is the authors' hope that this software will promote the use of more realistic measurement errors in the evaluation of current data analysis methods and the development of new data analysis methods.

3.6 Independent Testing

Dr. Ana de Juan. Associate Professor in the Department of Analytical Chemistry at the University of Barcelona, Diagonal, 645. 08028 Barcelona.

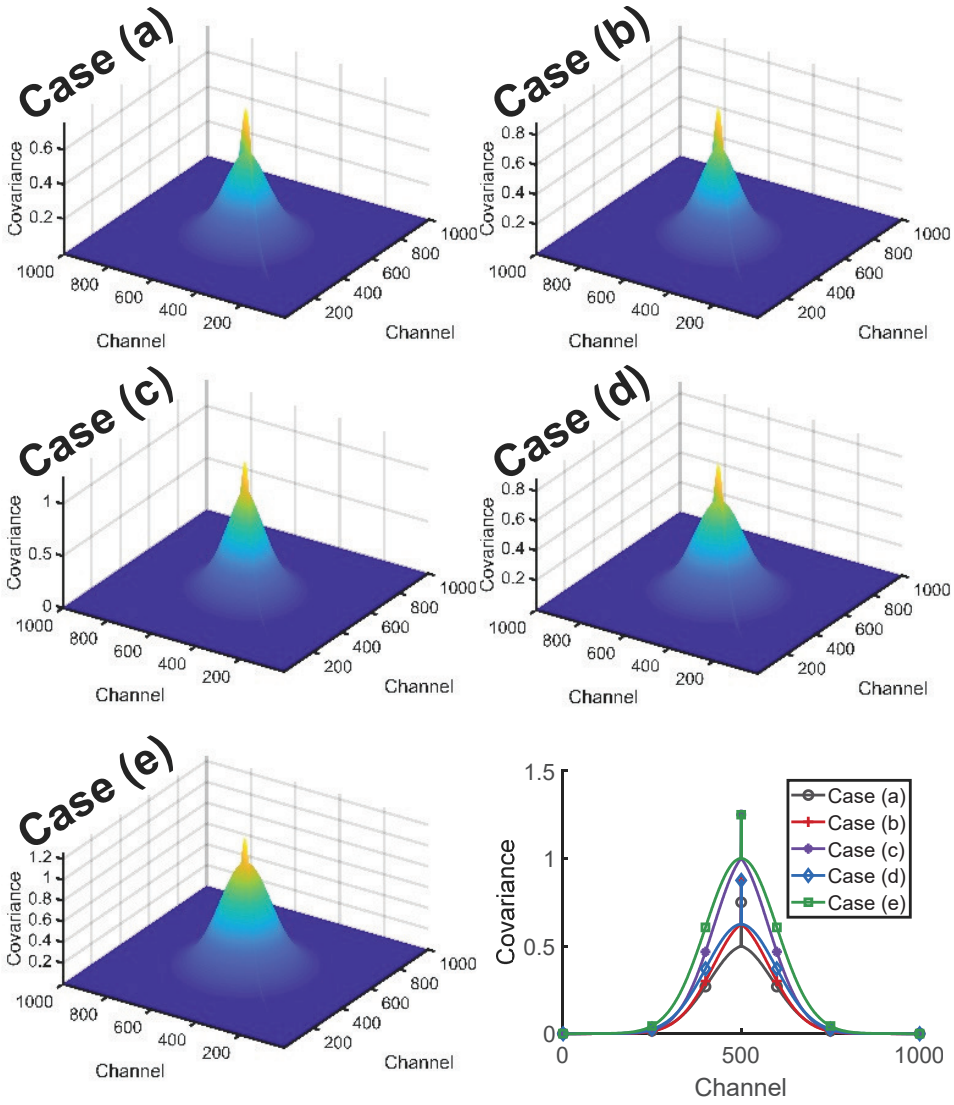

Figure 3.11: Theoretical ECMs obtained from the NoiseGen GUI output for the following cases: Case (a) shot noise (SN) RSD of 0.50, multiplicative offset noise (MON) RSD of 0.50, and proportional brown noise (PBN) RSD of 0.50; Case (b) SN RSD of 0.50, MON RSD of 0.25, and PBN RSD of 0.75; Case (c) SN RSD of 0.50, MON RSD of 0.00, and PBN RSD of 1.00; Case (d) SN RSD of 0.50, MON RSD of 0.75, and PBN RSD of 0.25; Case (e) SN RSD of 0.50, MON RSD of 1.00, and PBN RSD of 0.00.

The NoiseGenGUI is a useful and user-friendly toolbox to get acquainted with different kinds of noise and to clearly see how they can modify reference signals. It is an asset that the different noise contributions be very well defined and separated both in the GUI and the related publication, since analytical and, in general, scientific measurements, present noise patterns that can issue from very diverse combinations of these contributions. The use of NoiseGenGUI does not only have an educational side, oriented to help in understanding the nature of the different noise contributions, but it is also a very handy tool to generate easily and properly datasets affected by different noise levels and patterns to test how these variations may affect the performance of the data analysis algorithms that we design or that we commonly use in our data analysis tasks. Future wishes for updates could be having a standalone version of the GUI and, more ambitious, setting an auxiliary tool to diagnose noise patterns in real data sets provided that an estimate of the global noise is available.

CHAPTER 4

DATA FUSION WITH NOISY MEASUREMENTS

The preceding two chapters introduced methods to simulate complex error structures of various types for multivariate chemical measurements through the error covariance matrix (ECM). This capability is important for the evaluation of data analysis methods ranging from simple signal processing (smoothing, differentiation, baseline correction, multiplicative signal correction, transformation, etc.) to more complex applications (decomposition, multivariate calibration, curve resolution, classification, etc.). A natural consequence of this evaluation is to address the question: given a particular error structure (*i.e.* the ECM), what is the best way to analyse the data?

The connection between error structure and optimal data analysis has been recognized implicitly and explicitly since chemometric tools were first employed. The issue is most frequently addressed through preprocessing of the data, *i.e.* by modifying the data to fit the analysis method rather than the other way around. For example, if heteroscedastic errors are an issue, scaling of individual channels is often used to render the error variance more uniform. If multiplicative offset noise is the main problem, techniques such as derivative filtering or multiplicative signal correction (MSC) can be applied. However, these methods are generally *ad hoc* and limited in their ability to treat complex error structures arising from multiple sources. A better way would be to incorporate the error information contained in the ECM into the data analysis algorithm itself.

For methods that rely on decomposition through PCA, MLPCA[41] can be an optimal or near optimal way to model the data if the ECM is available, since subspace estimation and data projection are based on principles of maximum likelihood estimation. This has

been demonstrated for multivariate calibration,[99–102] multivariate curve resolution,[48,103] and exploratory data analysis.[93] In this chapter, the extension of MLPCA to the problem of data fusion is demonstrated using both simulated and experimental data.

4.1 Introduction

All chemists are aware that different analytical tools provide different kinds of information. For example, infrared (IR) spectroscopy and nuclear magnetic resonance (NMR) spectroscopy reveal different structural information about a molecule and this complementary information can lead to the elucidation of the complete structure in a way that could not be achieved by either method individually. Likewise, if two sets of multivariate measurements (*e.g.* UV-visible and IR spectra) are made on a common set of samples, combining these measurements in the case of multivariate data analysis may reveal information that could not be obtained through the analysis of each set individually. This process of combining data matrices with a common dimension of information (the rows or samples in this case) is known as data fusion, but is often referred to as multi-block analysis or data integration.[104]

The concept of data fusion has been around for some time, but has become more prominent in chemistry over the last two decades as instrumental capabilities have expanded and new kinds of information have been sought. Many different types of instrumental measurements have been combined through data fusion and these are often quite distinct from one another. Examples include X-ray fluorescence with Raman spectroscopy,[105] ^{1}H NMR and UV-vis spectroscopy,[106] HPLC and MS,[107] and IR, Raman and NIR spectroscopy.[108] Applications are also quite diverse, ranging from assessing the varietal origin of olive oil[109] and food and beverage authentication,[110,111] to biosynthesis monitoring,[112] and classification of pigments in works of art.[113]

In principle, data fusion is a simple matter of concatenating two or more data matrices along the rows, as shown in Figure 4.1, effectively pushing them together and then applying an appropriate data analysis tool such as PCA. This is known as "low-level" data fusion. In practice, this simple approach often presents problems. First, there is a question of scale, since measurements will have different units which may cause one or the other to dominate the variance. Although scaling may address this, differences in the error structures can complicate preprocessing options. While one block might have issues with heteroscedastic noise, another may require MSC to remove multiplicative offset noise.

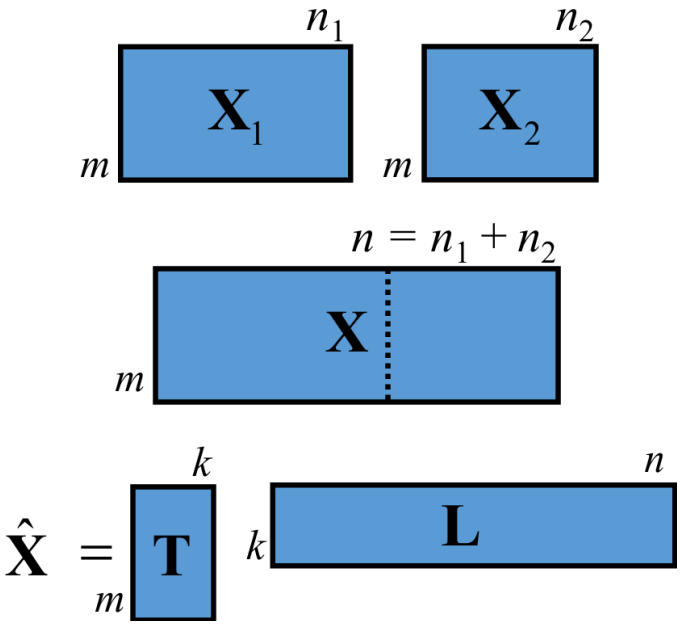

Figure 4.1: Concatenation, or fusion, of data set $\mathbf{X}_1$ ($m \times n_1$) and $\mathbf{X}_2$ ($m \times n_2$) to form the multi-block data set $\mathbf{X}$ ($m \times n$). Concatenated PCA produces a rank k estimation of $\mathbf{X}$ from the scores, $\mathbf{T}$ ($m \times k$), and the loadings, $\mathbf{L}$ ($k \times n$).

Optimizing preprocessing for multi-block data is more complex, since results may be sub-optimal if any of the blocks are not treated correctly. In addition, blocks contain both common and mutual information,[114] which increases the complexity of the process.

Beyond low-level fusion, mid-level fusion extracts features (*e.g.* scores) from each data set independently and these features are then combined to make the fused data set. In high-level data fusion, each data set is analysed individually and the final results (*e.g.* class, concentration) are combined.[115] This chapter focuses on low-level data fusion.

It has long been recognized that resolving the incompatibilities in the measurement error structure of multi-block data is a key challenge for data fusion. Most often, this is addressed through a variety of preprocessing strategies applied to each block. Since optimal preprocessing is a challenge even for a single block, it becomes even more difficult when multiple blocks are involved, since the nature of the data and errors can be very different. Recently, Waaijenborg et al.[116] discussed some of the challenges and solutions for metabolomics data sets. In this chapter, I consider a more direct and versatile approach through MLPCA.

PCA is the basis for a wide variety of common multivariate data analysis methods, and by extension multi-block data analysis methods as well. It is known that PCA provides

a sub-optimal subspace estimate in the presence of non-*iid* normal measurement errors, such as those observed in most chemical data.[93] A common strategy to overcome this is to preprocess the data such that the measurement errors become more similar to *iid* normal errors. However, a standing issue in multi-block analysis, and single-block analysis, is how to preprocess the individual data sets that are to be fused.[117–120] As with single-block data, preprocessing procedures are highly variable and data set dependent. Preprocessing of multi-block data is especially difficult as improper preprocessing of even one of the data sets would likely negatively influence the analysis and defeat the purpose of data fusion.

As mentioned in Chapter 1, MLPCA is a "single-block" subspace estimation method proposed by Wentzell and co-workers.[41,42] MLPCA takes the measurement error structure into account when estimating the subspace of the data, leading to more reliable results in the presence of non-*iid* normal noise and obtaining the PCA result when the errors are exactly *iid* normal. Of course, a major drawback of MLPCA is that the error structure must be known, but an estimate of the error structure can often be obtained if replicates are available. A desirable property of MLPCA is that preprocessing does not need to be done prior to applying the algorithm as this information is encoded in ECM. Therefore, applying MLPCA to concatenated blocks of data should lead to better subspace estimation without the need of complicated preprocessing if a proper estimate of the fused ECM is known. The purpose of this chapter is to explore the use of MLPCA in multi-block data analysis to improve information extraction from multi-block data.

4.2 Theory

4.2.1 Data Fusion

To introduce the general terms of data fusion, the theoretical framework for performing concatenated PCA on a two-block data matrix will be presented. Let $\mathbf{X}_1$ ($m \times n_1$) be the block 1 data matrix and $\mathbf{X}_2$ ($m \times n_2$) be the block 2 data matrix with m samples and n_1 and n_2 variables, respectively. The horizontal concatenation of these two blocks forms the fused data matrix $\mathbf{X}$ ($m \times n$), where $n = n_1 + n_2$. By this formulation it is required that the same samples are represented in each block. Performing PCA on the concatenated matrix can be done the same way as conventional PCA to produce a rank k solution from the scores, $\mathbf{T}$ ($m \times k$), and the loadings, $\mathbf{L}$ ($k \times n$). This procedure is shown in Figure 4.1. In practice, optimal preprocessing must be done on $\mathbf{X}_1$ and $\mathbf{X}_2$ before performing PCA

on $\mathbf{X}$ because the PCA model assumes that the measurement errors are *iid* normal across both blocks. This decomposition is often done using singular value decomposition (SVD), shown in Equation 4.1,

$$\hat{\mathbf{X}} = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^{\mathrm{T}} = \mathbf{TL}, \tag{4.1}$$

where $\mathbf{U}_k\mathbf{S}_k$ is equal to the scores ($\mathbf{T}$) and $\mathbf{V}_k^{\mathrm{T}}$ is equal to the loadings ($\mathbf{L}$) for the rank k decomposition.

4.2.2 Multi-block MLPCA

Conceptually, extension of MLPCA to the multi-block case is straightforward. The concatenated matrix, $\mathbf{X}$, is formed in the same way as described above, *i.e.* $\mathbf{X} = \mathbf{X}_1|\mathbf{X}_2$. All that remains is to define the ECM, $\mathbf{\Sigma}_X$, prior to carrying out MLPCA. However, this requires some further discussion of the practical implementation of MLPCA and the estimation of the ECM.

MLPCA is based on a single philosophy of subspace estimation, but is practically embodied by six separate algorithms (cases A–F) that reflect different error structures.[42] Cases A–C apply to independent errors and cases D–F to correlated errors. Case A, which is just PCA, applies to *iid* normal errors, and therefore can be excluded as a trivial case. Case F applies to instances where correlation may exist in both the row and column direction (*e.g.* in a fluorescence excitation–emission matrix). It is not generally applicable to cases where the rows represent individual samples and, additionally, is computationally demanding to the point of making it impractical except for small matrices. Therefore, case F can also be excluded. Of the remaining cases, B and D apply to stationary noise (*i.e.* the ECM is the same for all rows of $\mathbf{X}$) while C and E apply to non-stationary noise (*i.e.* each row of $\mathbf{X}$ has a different ECM). While the latter situation is certainly observed in chemical measurements and can be employed, stationary noise is usually assumed because (1) cases B and D are computationally more efficient, (2) it can be difficult to estimate individual row ECMs reliably, and (3) because signals are typically very similar, the differences in ECMs among rows is usually too small to have an effect. Finally, since case B (independent errors) is a special case of case D (correlated errors) we will limit our discussion to case D for simplicity. However, any of the error structures could be employed with some relatively minor adjustments.

Based on the preceding discussion, the error structures in $\mathbf{X}_1$ $(m \times n_1)$ and $\mathbf{X}_2$ $(m \times n_2)$ can be described separately by the ECMs $\mathbf{\Sigma}_{X1}$ $(n_1 \times n_1)$ and $\mathbf{\Sigma}_{X2}$ $(n_2 \times n_2)$. For the fused data matrix, $\mathbf{X}$, however, we also need to describe (in principle) the covariance between the errors in $\mathbf{X}_1$ and those in $\mathbf{X}_2$. This matrix is designated as $\mathbf{\Sigma}_{X12}$ $(n_1 \times n_2)$ or its transpose $\mathbf{\Sigma}_{X21}$ $(n_2 \times n_1)$. While in principle $\mathbf{\Sigma}_{X12}$ could be estimated using identical replicates on the two instruments, in practice it is normally assumed that $\mathbf{\Sigma}_{X12} = \mathbf{\Sigma}_{X21} = 0$. This is equivalent to saying that measurement errors on one instrument, say an IR spectrometer, should have no relationship with those on another instrument, such as an NMR spectrometer. This is a reasonable assumption unless dominant errors are associated with some non-instrumental characteristic, such as sample preparation, that affects both instruments in the same way. With simplification, the fused ECM is a block diagonal matrix $((n_1 + n_2) \times (n_1 + n_2))$ as shown in Figure 4.2

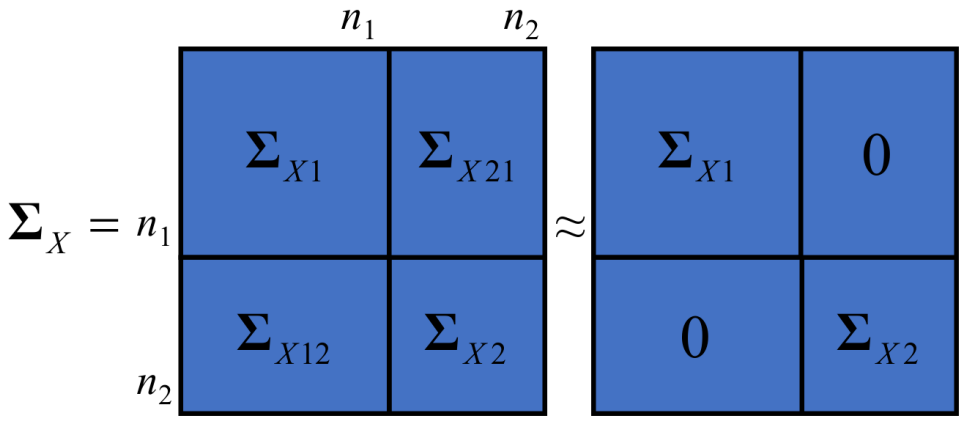

Figure 4.2: The general structure of the ECM for the multi-block data set $\mathbf{X}$, which is comprised of the ECM from block 1, $\mathbf{\Sigma}_{X1}$, and block 2, $\mathbf{\Sigma}_{X2}$, as well as their interactions, $\mathbf{\Sigma}_{X12}$ and $\mathbf{\Sigma}_{X21}$.

4.2.3 Estimation of the ECMs

The question remains regarding how to estimate the ECMs $\mathbf{\Sigma}_{X1}$ and $\mathbf{\Sigma}_{X2}$ for the practical implementation of MLPCA on the fused data. Experimentally the ECM can be estimated in the same way as the variance by measuring replicates for each sample. Suppose $\mathbf{X}$ represents s samples, each replicated r times (*i.e.* $m = r \cdot s$). We can construct an error matrix, $\mathbf{E}$, that contains the difference between each row in $\mathbf{X}$ and its corresponding sample

mean. In other words,

$$\mathbf{e}_i = \mathbf{x}_i - \bar{\mathbf{x}}_j. \tag{4.2}$$

Here $\mathbf{e}_i$ is a row vector of $\mathbf{E}$ ($i = 1 \ldots m$), $\mathbf{x}_i$ is the corresponding row of $\mathbf{X}$, and $\bar{\mathbf{x}}_j$ is the mean vector ($j = 1 \ldots s$) associated with $\mathbf{x}_i$. Based on this, the ECM can now be estimated from

$$\hat{\mathbf{\Sigma}} = \frac{\mathbf{E}^{\mathrm{T}}\mathbf{E}}{m - s}. \tag{4.3}$$

This is a pooled ECM based on the replicate measurements and can be applied to $\mathbf{X}_1$ and $\mathbf{X}_2$ individually. If the number of replicates is different for each sample, some adjustments to the degrees of freedom is necessary, but the same process applies.

In principle, the ECMs calculated in this way could be used for MLPCA, but in practice this is problematic unless the number of replicates is very large because: (a) the $\hat{\mathbf{\Sigma}}$ is often rank deficient (singular) leading to certain computational complications, and (b) the covariance estimates can have a large uncertainty. Consequently, it is much better to use an ECM obtained by one of the following means:

1. Theoretical prediction. Except in rare cases, this is only possible for simulations in which the true error structure is known.

2. Theoretical modelling. This is based on developing a mathematical description of the ECM using experimentally measured ECMs and a parameterized model consistent with the types of errors expected. By fitting the model, reasonable estimates can be obtained.[43,44] However, this requires a good knowledge of the system under study.

3. Empirical modelling. Like theoretical modeling, this approach generates a fit of the experimental ECM, but does so using principal axis factorization (PAF), which does require a causal model.[121] PAF is similar to PCA and also based on SVD. It provides a model that separates out correlated sources of error from independent sources of error.

In this work, method 1 is used for proof of principle for a simulated data set. For the experimental data, method 3 is used.

4.3 Experimental

4.3.1 Computational Details

All calculations were performed in MATLAB release 2018a (MathWorks, Natick, MA)[81] using scripts programmed in-house. As standard practice, all data was mean centered before applying PCA or MLPCA unless otherwise stated.

4.3.2 Simulated Data Set

To evaluate the performance of multi-block MLPCA, a four-class (50 samples each), two-block data set was simulated. The two blocks, $\mathbf{X}_1$ and $\mathbf{X}_2$, were simulated according to the following mixture model:

$$\mathbf{X}_1 = \mathbf{C}\mathbf{S}_1, \tag{4.4}$$

$$\mathbf{X}_2 = \mathbf{C}\mathbf{S}_2, \tag{4.5}$$

where row i of $\mathbf{C}$ (200×4) describes the concentration of the four species in sample i, and $\mathbf{S}_1$ (4×200) and $\mathbf{S}_2$ (4×200) describe the spectral response for each of the 4 species in block 1, and block 2, respectively. The concatenation of these two blocks forms the error-free fused data matrix, $\mathbf{X}$ (200×400) $= \mathbf{X}_1|\mathbf{X}_2$. The first two columns of $\mathbf{C}$ were constructed using a set of 50 samples randomly clustered (bivariate normal distribution with $\sigma^2 = 0.3$) around each of the following concentration coordinates: (2, 4), (4, 2), (4, 4), and (2, 2). This describes the first two columns of $\mathbf{C}$ and will be referred to as the principal species, as they contain the information that will form four distinct clusters (50 samples each) in a two-dimensional space. Additionally, two ancillary species were also defined, but drawn from the same concentration coordinate distribution for all samples with a much smaller concentration variance relative to that in the first two columns (concentration coordinate of (3, 3), bivariate normal distribution with $\sigma^2 = 0.02$). This describes the last two columns of $\mathbf{C}$. These ancillary components should not be the dominant source of variance in the error-free data. To encode the unique response information for each of these species, each block has its own $\mathbf{S}$ matrix, and in conjunction with $\mathbf{C}$, was designed to separate the four classes into only two groups when PCA is applied to the individual blocks. $\mathbf{S}_1$ was designed to only contain a response that separates group 1 and 4 from

groups 2 and 3 and $\mathbf{S}_2$ to separate groups 2 and 4 from groups 1 and 3. Specifically, each row of $\mathbf{S}_1$ was constructed with a Gaussian profile over the 200 channels: row 1, centered at 160 with a height of 0.05; row 2, centered at 120 with a height of 0.01; row 3, centered at 80 with a height of 0.8; row 4, centered at 40 with a height of 1. The standard deviation for all Gaussian profiles was 20. $\mathbf{S}_2$ was designed similarly, but with the following parameters for the Gaussian profiles: row 1, centered at 40 with a height of 0.01; row 2, centered at 80 with a height of 0.1; row 3, centered at 120 with a height of 1.2; row 4, centered at 160 with a height of 0.8. The row profiles for $\mathbf{S}_1$ and $\mathbf{S}_2$ are shown in Figure 4.3(a) and (b), respectively.

NoiseGen,[122] which was highlighted in Chapter 3, was employed to simulate noise for each block. Brown noise with a standard deviation equal to 1% of the maximum value of the mean signal in $\mathbf{X}_1$ was added to $\mathbf{X}_1$. Proportional white noise (RSD equal to 1% of the maximum value of the mean signal in $\mathbf{X_2}$ and proportional to the mean signal in $\mathbf{X_2}$) along with white noise (standard deviation equal to 1% of the maximum value of the mean signal in $\mathbf{X_2}$) was added to $\mathbf{X_2}$. The simulated data are shown in Figure 4.3, where the error-free data, $\mathbf{X}$, are shown in Figure 4.3(c) and the data with the added noise is shown in Figure 4.3(d).

4.3.3 UV, NIR, and MIR Olive Oil Data Set

This unpublished data set was obtained from the group of Dr. Patricia Valderrama from Maringá State University in Brazil. The data set consists of 24 samples of three different brands of flavoured olive oil (extra virgin, lemon, pepper, garlic, basil, bell pepper, and orange) from different regions of Italy (Puglia, Emilia-Romagna, and Umbria) acquired in Brazil and the Netherlands. The samples from Brazil have two different lots for two different brands and the samples from the Netherlands were produced manually and on a small scale from the same lot. For each olive oil sample a total of 10 replacement replicates were acquired on three different instruments. The measurements in the UV-Vis region were carried out using a quartz cuvette 1 mm optical path in the range of 200 to 800 nm, with a portable spectrometer (Ocean Optics Red Tide USB 650). NIR spectra were collected using a glass cuvette and a portable MicroNIR spectrometer (JDSU) in the range of 900 to 1700 nm. The MIR spectra for the olive oil samples were obtained using a Shimadzu spectrometer (IRAffnity-1) equipped with ZnSe attenuated total reflectance (ATR) sampling interface and the spectra were collected from 600 to 2200 nm. Using

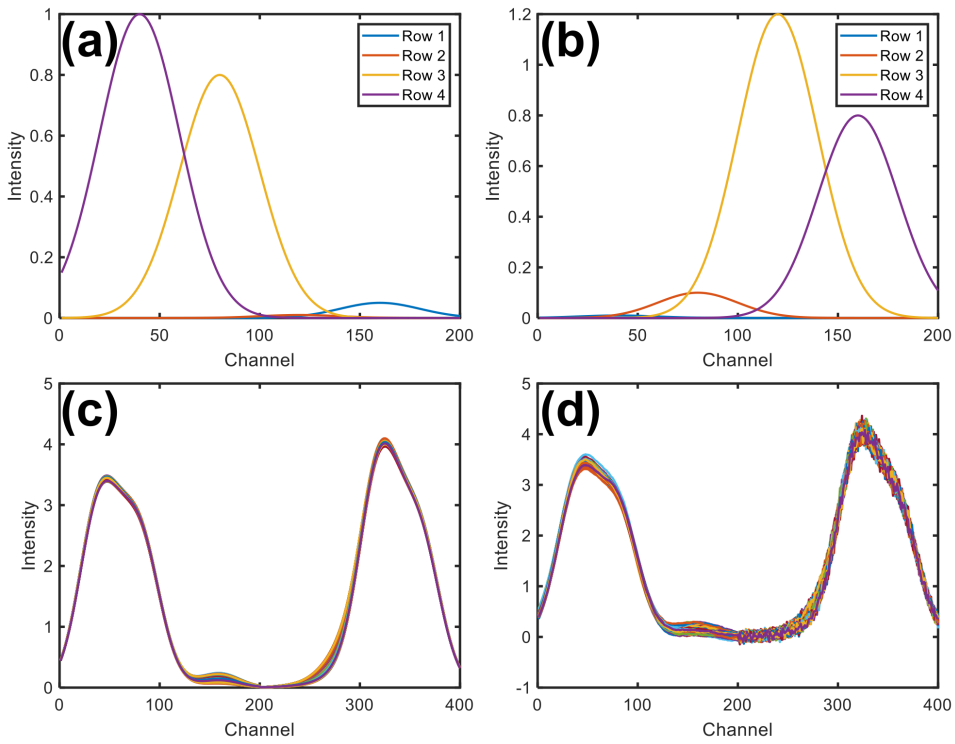

Figure 4.3: The simulated spectral response for (a) block 1, $\mathbf{S}_1$ (4×200) and (b) block 2, $\mathbf{S}_2$ (4×200). (c) The simulated error-free fused data $\mathbf{X}$ (200×400) and (d) the same data where each block is subjected to a different noise structure and then fused.

these measurements a three-block fused data set was built where block one (UV-vis) is 240×651, block two (NIR) is 240×124 and block three (MIR) is 240×934. Therefore, the fused data set has dimensions of 240×1709.

4.4 Results and Discussion

4.4.1 Simulated Data

Figure 4.4(a) shows the simulated multi-block data in the two-dimensional space that was designed to separate four clusters (two principal species concentrations, or the first two columns of $\mathbf{C}$), as well as the scores obtained after applying PCA to the block 1 data (Figure 4.4(b)) and block 2 data (Figure 4.4(c)), and the fused data matrix (Figure 4.4(d)). As expected, after applying PCA to the block 1 data, the corresponding scores were observed to separate classes 1 and 4 (red circles and purple upward-pointing triangles, respectively) from class 2 and 3 (blue squares and green downward-pointing triangles, respectively), while applying PCA to block 2 was found to separate classes 2 and 4 from

classes 1 and 3. The result of applying PCA to the fused error-free data is shown in Figure 4.4(d), where the scores plot reveals all four clusters. In the absence of error, the directions of maximum variance in the fused data correspond to the directions that discriminate among the four classes. In the case of applying PCA to only block 1, only the first PC corresponds to one of the discriminating vectors, as the spectral contribution associated with the other discriminating direction was designed to have small variance compared to the ancillary components. The same is true for block 2, but the discriminating vectors are reversed. This mimics the ideal case of real data fusion, where fused data can contain more chemical information than each of the blocks independently. However, in reality, the error structures for block 1 and 2 would be different and likely not *iid* normal. To probe the impact of the errors on fused data, PCA was also applied to the fused data in the presence of error (brown noise in block 1 and both proportional and non-proportional white noise in block 2).

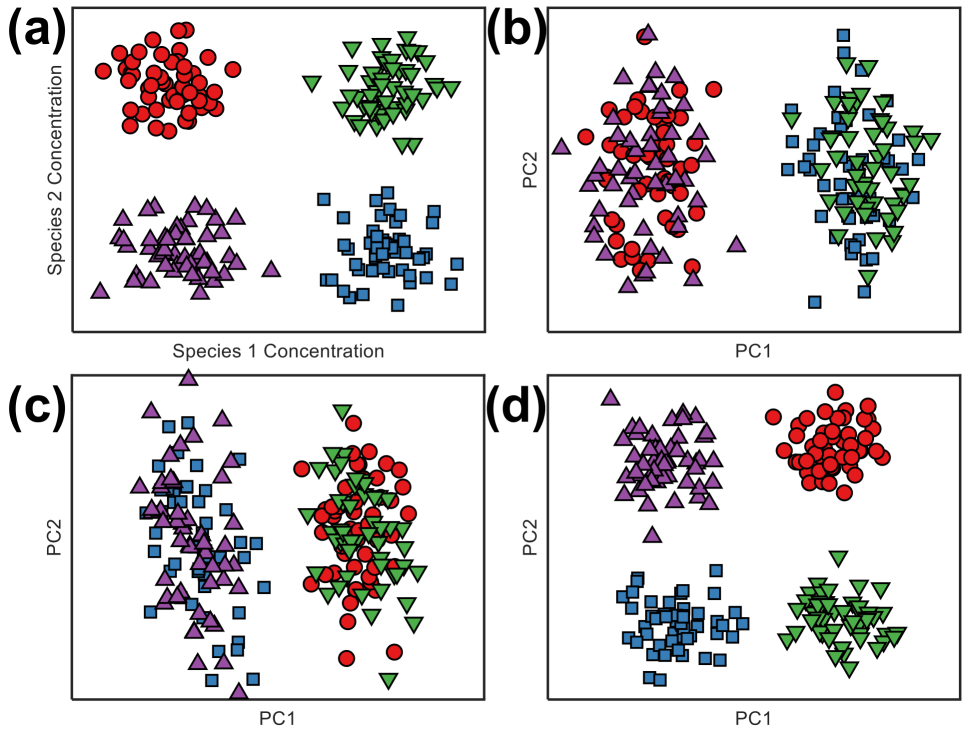

Figure 4.4: (a) Clustering observed when plotting the two principal species concentrations against one another. Scores plot obtained from applying PCA to the (b) error-free block 1 data, (c) the error-free block 2 data, and (d) the error-free fused data.

Figure 4.5(a) shows the result of applying PCA to the noisy fused data. As expected, due to the non-*iid* normal error structure in the fused data, PCA fails to reveal the target

clustering space. Neither PCs correspond to the chemical variance of interest due to the addition of noise to the data. As mentioned in the introduction of this chapter, when applying PCA to fused data, it is common practice to individually scale the independent blocks prior to the analysis to get their responses in the same range. Figure 4.5(b) shows the result of applying PCA to the fused data set after scaling the individual blocks by the standard deviation of the corresponding mean spectrum. The quality of the separation was observed to improve, with a projection similar to that observed when applying PCA to the error-free block 2 data, Figure 4.4(c). Autoscaling was also explored, but the resulting PCA scores were not as good as scaling the individual blocks by the standard deviation of the mean spectrum.

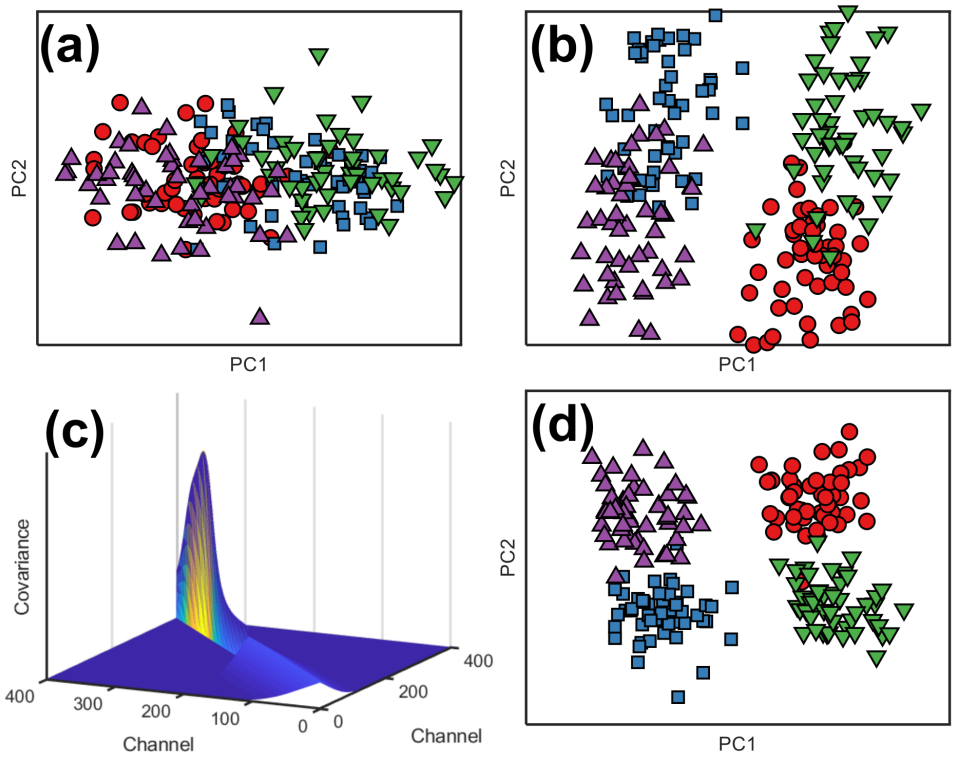

Figure 4.5: (a) PCA scores obtained from applying PCA to the fused noisy data. (b) PCA scores obtained from applying PCA to the data after scaling block 1 and 2 to the respective standard deviation of the mean signal of each block. (c) The fused ECM constructed from the block 1 and 2 model ECMs used to generate the noise added to the data set. (d) MLPCA scores obtained using the ECM in (c).

MLPCA was then applied to the fused noisy data using the fused ECM shown in Figure 4.5(c). The ECM was constructed from the individual model ECMs that were used to generate the noise. The results of the application of MLPCA are shown in Figure 4.5(d).

As anticipated, these results show a clear improvement over the PCA results on the noisy data, although the separation of the clusters is not as distinct as for the error-free data. The application of PCA to the scaled data (Figure 4.5(b)) shows mainly the separation consistent with the block 2 data, suggesting that the brown noise is more problematic than the proportional noise in block 2. The MLPCA results suggest that MLPCA is more effective at extracting the information in block 2.

This is a very simple simulation with only four chemical components and ECMs which are known in advance, but it serves to support the idea that MLPCA could be an effective tool to fuse data with different error structures.

4.4.2 UV, NIR, and MIR Olive Oil Data Set

The individual spectra for all 240 samples of olive oil measured on each of the three instruments are shown in Figure 4.6. The UV-Vis data, Figure 4.6(a), contains some notable areas. First, in the UV region, it appears that low light levels at the detector are contributing to noisy signals for all samples. It is anticipated that this will be a significant contribution to variance in the corresponding ECM. Peaks in the 400 to 550 nm may contain information for discriminating between some samples and some baseline offset is evident. The magnitude of the NIR data, Figure 4.6(b), is much greater than that of both the UV-Vis and MIR data, which will cause PCA to primarily model variance in this data set first if scaling is not performed. In terms of the ECM, the error structure of this block is likely to be dominated by offset noise and should be the main component of the fused ECM purely due to scale. The MIR data, Figure 4.6(c), is on the same scale as the UV-Vis data (both are absorbance) and is hard to tell by visual inspection if there is any discriminatory information in the spectra.

The ECMs for each of the individual blocks were modelled by Thays R. Gonçlaves as part of her research. The ECMs were each modelled using principal axis factoring (PAF), a method similar to PCA, but allows for heteroscedastic measurement error.[121] As this thesis is not concerned with the ability of PAF to model ECMs, the details of the modelling and quality will not be discussed, however, by visual comparison to the sampled ECMs, the PAF reconstructed ECMs seemed to adequately model the error structure in the data. Mesh plots of the PAF reconstructed ECMs for the UV-Vis, NIR, MIR data, and fused data are shown in Figure 4.7.

Figure 4.8 shows the scores obtained from various decomposition methods on the fused

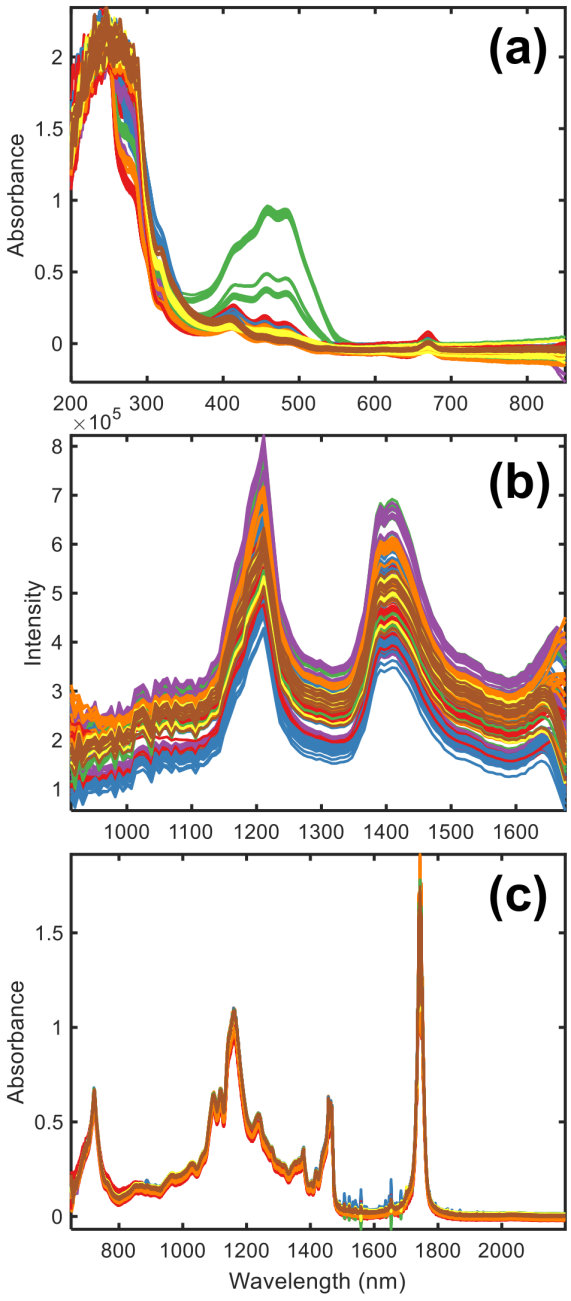

Figure 4.6: (a) UV-Vis, (b) NIR, and (c) MIR spectra of the olive oil data set.

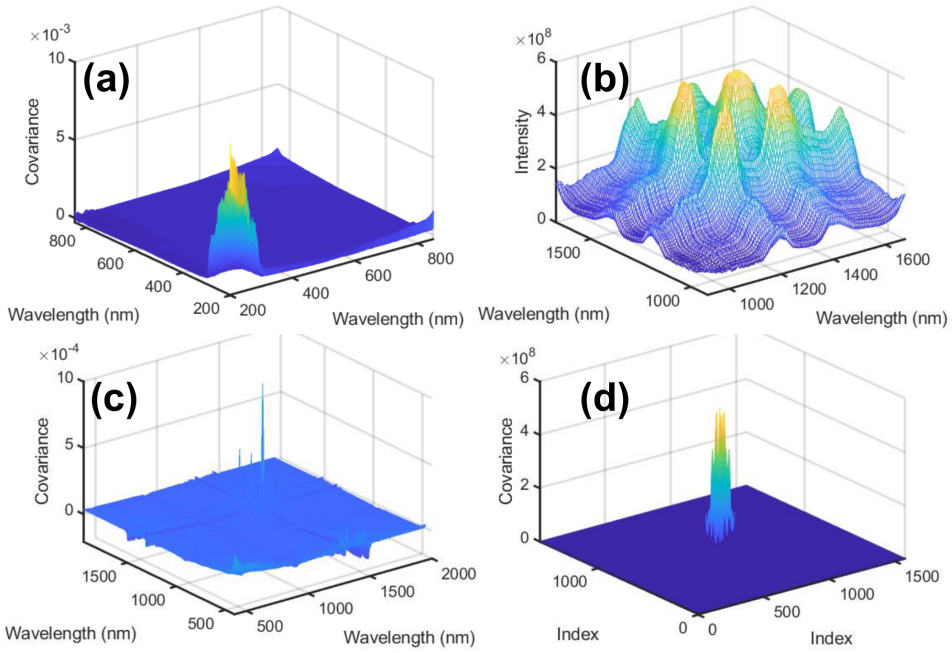

Figure 4.7: PAF reconstructed ECMs for (a) block one (UV-Vis) (b) block two (NIR), and (c) block three (MIR) of the olive oil data set. (d) is the result of constructing the fused ECM which was used for MLPCA of the fused data.

data including (a) PCA on the intensity normalised data, (b) PCA on the autoscaled data, and (c) MLPCA using the fused ECM. To aid interpretation of the results, the samples in all three plots have their brand encoded by shapes: brand 1 represented by circle markers, brand 2 represented by square markers and brand 3 represented by triangle markers, their lot number encoded by unfilled (lot 1) and filled (lot 2) markers, and their flavour encoded by colour: extra virgin olive oil flavour represented by red markers, lemon flavour represented by blue markers, pepper flavour represented by green markers, garlic flavour represented by purple markers, basil flavour by orange markers, bell pepper flavour by yellow markers, and orange flavour by brown markers.

The PCA scores in Figure 4.8(a) suggest some grouping by flavour, where a majority of the pepper flavoured oils (green markers) in the first two dimensions are clustering away from all other flavours. This is not surprising, since the pepper oils show distinctive colour. However, there does not seem to be much clustering of the other flavours, and it is difficult to extract other information from the patterns observed. Looking at results of applying PCA to the autoscaled data, Figure 4.8(b), it seems that the separation between the green class (pepper flavoured oils) is amplified, but all other samples are heavily overlapped,

resulting in an inferior projection compared to applying PCA to the scaled data alone. This result is expected, as the autoscaling transform is giving equal weight to the noise in the baseline as well as the informative chemical variance, effectively obscuring any useful information.

In contrast to the two results of PCA applied to the fused olive oil data set, the MLPCA scores shown in Figure 4.8(c) reveal more information about the relationships among the samples. First, all replicates are tightly clustered together, which was only somewhat observed for the PCA results, where some replicates were not showing up in the same groupings. Second, there seems to be a more distinctive (although not perfect) separation of brand and lot which is mostly independent of the flavouring. This can be seen with the unfilled square markers (lot 1, brand 2), the unfilled triangle markers (lot 1, brand 3), and, to a somewhat lesser extent, the unfilled circle markers (lot 1, brand 1). The same groupings can generally be seen for lot 2 for both brands (filled markers). This suggests something about the manufacturing process of the oils, where the flavourings may be added at the end of the process and do not have as much impact on the chemical information of the oil compared to the batch or lot to which the oil belongs. Another notable difference is that all Dutch samples are grouping together (triangle markers). This aligns with the experimental preparation where the Dutch samples were made up in small batches from the same lot and produced manually, which one would expect to have similar characteristics and spectral response. However, this data set is complex and drawing any hard conclusions based on the results, without more data, is difficult. Nonetheless, it is clear that by including the error information in the analysis of the fused matrix, information that was not expressed in the PCA results is obtained.

4.5 Conclusions

The advantage of applying MLPCA to multi-block data was demonstrated in this work. First, it was shown that applying MLPCA to fused data is conceptually the same as applying PCA, but requires some reformulation in terms of defining the fused ECM. Through simulation, it was found that applying MLPCA with a good representation of the fused ECM provides an improved representation of the true chemical subspace without the need of individually preprocessing the different data blocks. Applying the method to a three-block UV-Vis, NIR and MIR data set concerned with various lots, batches,

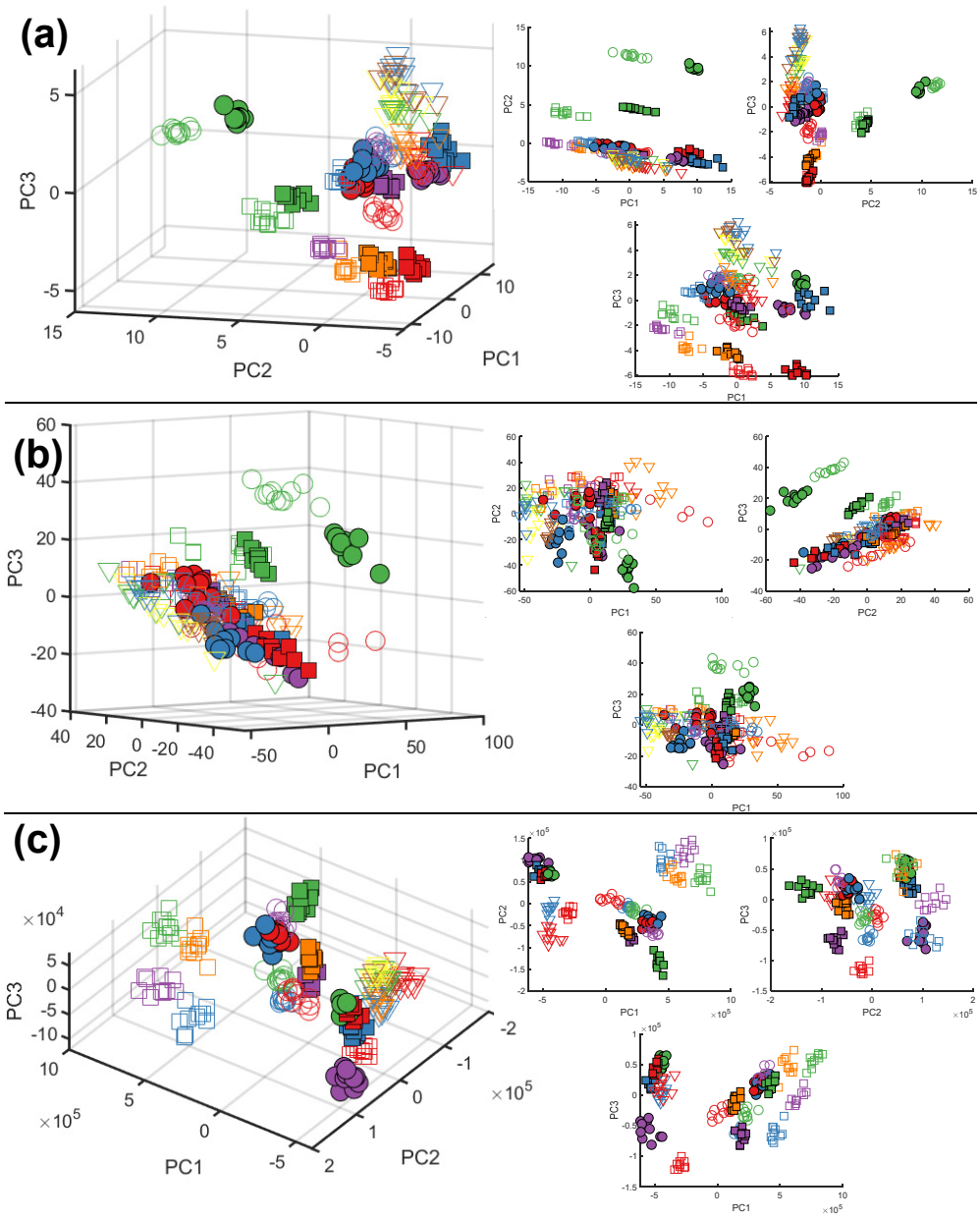

Figure 4.8: Scores plots from applying (a) PCA, (b) PCA on the autoscaled data, and (c) MLPCA using the PAF calculated fused ECM. The shapes denote the different brands, filled or unfilled marker denotes the lot, and colour denotes the flavour: brand 1 represented by circle markers, brand 2 represented by square markers and brand 3 represented by triangle markers, their lot number encoded by unfilled (lot 1) and filled (lot 2) markers, and their flavour encoded by colour: extra virgin olive oil flavour represented by red markers, lemon flavour represented by blue markers, pepper flavour represented by green markers, garlic flavour represented by purple markers, basil flavour by orange markers, bell pepper flavour by yellow markers, and orange flavour by brown markers.

and flavourings of olive oils proved useful in revealing information about the data that is obscured when traditional methods, such as PCA, are applied. It is my hope that this work motivates researchers to perform properly designed experiments with many replicates to get access to the fused ECM. The initial investment in time (*i.e.* performing replicates) should result in time saved by not needing to try various combinations of preprocessing strategies.

CHAPTER 5

SPARSE PROJECTION PURSUIT ANALYSIS: AN ALTERNATIVE FOR EXPLORING MULTIVARIATE CHEMICAL DATA *

In the preceding chapter, it was demonstrated how the addition of measurement error information can improve the outcome of chemometric methods, specifically exploratory data analysis, by allowing chemical variance to be distinguished from measurement variance. However, as demonstrated in Chapter 1, this approach has limitations when it is the chemical variance structure that impedes the extraction of the desired information. In this case, projection pursuit analysis (PPA) has been demonstrated as a viable alternative to variance-based methods. In this chapter, sparse projection pursuit analysis (SPPA), a new approach for the unsupervised exploration of high-dimensional chemical data, is proposed as an alternative to traditional exploratory methods such as PCA and hierarchical cluster analysis (HCA). Where traditional methods use variance and distance metrics for data compression and visualization, the proposed method incorporates the fourth statistical moment (kurtosis) to access interesting subspaces that can clarify relationships within complex data sets. The quasi-power algorithm used for projection pursuit is coupled with a genetic algorithm for variable selection to efficiently generate sparse projection vectors that

*This chapter is based on the article: Driscoll, S.P, MacMillan, Y.S, Wentzell, P.D. Sparse Projection Pursuit Analysis: An Alternative for Exploring Multivariate Chemical Data. *Analytical Chemistry*. 2019. 10.1021/acs.analchem.9b03166. **Contribution to Manuscript** SD and YM performed all calculations. YM is responsible for writing the first version of the SPPA algorithm and SD is responsible for writing the final version of the algorithm. SD wrote the first draft of the manuscript. PW contributed to the final version of the manuscript. PW supervised the project.

improve the chemical interpretability of the results while at the same time mitigating the problem of overfitting. Several multivariate chemical data sets are employed to demonstrate that SPPA can reveal meaningful clusters in the data where other unsupervised methods cannot.

5.1 Introduction

The essential role of multivariate statistical tools in modern analytical chemistry has been firmly established. For example, in 2018, roughly 6% of the articles in *Analytical Chemistry* made reference to PCA as a data analysis tool,[123] and this percentage is even higher (*ca.* 50%) in specialized fields such as metabolomics. PCA and HCA are particularly popular methods because they are unsupervised approaches that allow the compression and visualization of high dimensional data.[4,124–128] The application of PCA often involves the depiction of scores plots to visualize the relationships among objects (*e.g.* samples) in two or three dimensions, but it is also used as a data compression tool prior to the implementation of methods such as linear discriminant analysis, multivariate calibration, image analysis and multivariate curve resolution. The role of PCA and HCA in data exploration and visualization is not purely cosmetic, as the observed separation of classes (*e.g.* normal vs. diseased states) is widely used to establish proof-of-principle that a method can discriminate among samples of different types. Because they are unsupervised, these tools have become *de facto* standards to establish the validity of scientific hypotheses involving high-dimensional data. In this context, unsupervised methods are generally viewed as more reliable than supervised methods (*e.g.* discriminant analysis), especially where the number of samples is limited, since there is no training step that uses class information and therefore there is less chance of bias in the results arising from improper validation.[30,31,129–132]

Despite the importance of unsupervised methods in establishing the veracity of analytical studies involving multivariate data, PCA and HCA have remained essentially the only tools employed for many years. While the separation of objects by class using these methods can validate a hypothesis, their failure to do so does not invalidate it, since they represent only two ways of looking at the data. These approaches use metrics based on variance (PCA) and Euclidean distance (HCA) to establish relationships among the objects. Both are predicated on the assumption that the between-class separation of objects is larger

than within-class separation in the selected variable spaces, presenting a restricted view of complex relationships. Moreover, both methods are highly sensitive to preprocessing (*e.g.* scaling) of the data.[128,133] Clearly there is a need to expand the toolbox of unsupervised methods that can be used to validate chemical studies.

Here I describe a new method, sparse projection pursuit analysis (SPPA), as a powerful alternative for the unsupervised exploration of multivariate chemical data. Based on principles established five decades ago and exploiting more recent algorithmic advances, projection pursuit is a simple linear projection method that allows access to subspaces distinct from PCA and HCA to provide alternative views of multivariate data. Moreover, the proposed method incorporates variable selection to provide an avenue for chemically meaningful interpretation of results. Through the use of several multivariate data sets, SPPA is shown to provide class separation where traditional approaches such as PCA and HCA fail.

5.2 Theory

Projection Pursuit Analysis. The objective of projection pursuit analysis (PPA), first formulated by Kruskal[134] and expanded by Tukey and Friedman,[49] is simply to find interesting projections of the data. Like PCA, PPA is a linear mapping method that involves the projection of high-dimensional data into a low-dimensional subspace through the use of a projection matrix, $\mathbf{P}$, such that $\mathbf{T} = \mathbf{XP}$. For the original matrix $\mathbf{X}$ (m objects by n variables), $\mathbf{P}$ ($n \times p$) projects the m objects into the lower p-dimensional space of scores, $\mathbf{T}$ ($m \times p$, where $p < n$). The columns of $\mathbf{P}$ are known as the projection vectors and the dimension of the scores space is normally much less than that of the original data, with $p = 2$ or 3 for visualization.

To find the projection vectors, the "interestingness" of the projection is optimized using a projection index (PI). Strictly speaking, PCA can be considered a PPA method where the PI is the percentage of variance captured. More generally, the implementation of PPA has explored a variety of PIs, but the definition of useful PIs and their optimization has been a barrier to practical applications in chemistry and other fields. One PI that has been effectively used is kurtosis, the fourth statistical moment, since the minimization of this parameter emphasizes a separation of groups. However, practical implementation of kurtosis-based PPA was limited by the lack of an efficient optimization algorithm until the

recent development of a quasi-power method,[40] which has shown promising results for the analysis of chemical data sets.[40,55,56,59,61,135–138] Although there are several variants of this algorithm, the focus here is on the stepwise univariate approach, which extracts projection vectors successively by minimizing the kurtosis of the scores, K, as defined by Equation 5.1,

$$K = \frac{1}{ms_t^4}\sum_{i=1}^{m}(t_i - \bar{t})^4. \tag{5.1}$$

Here, the t_i are the elements of the scores vector ($\mathbf{t} = \mathbf{X}\mathbf{p}_j$, where $\mathbf{p}_j$ is the projection vector being optimized) and s_t is the standard deviation of the m projected scores around the mean value, $\bar{t}$. Following the optimization of each projection vector, $\mathbf{p}_j (j = 1 \text{ to } p)$, the matrix $\mathbf{X}$ is deflated to remove the variance associated with that dimension and the subsequent projection vector is found using the deflated matrix. The projection vectors are constrained to form an orthonormal set through this algorithm. The optimization uses a simple iterative learning rule given in Equation 5.2.

$$\mathbf{p}_j(\text{new}) \leftarrow \left[\sum_{i=1}^{m}(\mathbf{x}_i\mathbf{p}_j)^2\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i\right]^{-1}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)\mathbf{p}_j. \tag{5.2}$$

In this equation, $\mathbf{x}_i$ is a row vector of $\mathbf{X}$ (deflated $\mathbf{X}$ if $j > 1$) and $\mathbf{p}_j$ is the column vector of the projection matrix that is sought. This approach has been referred to as the quasi-power method because of its similarity to the power method used to solve eigenvalue problems. Although it is a nonlinear optimization, it generally converges quickly and provides reliable solutions, especially if multiple starting points are used.

The procedure described above is known as stepwise univariate PPA because it optimizes each dimension in succession. An alternative approach, called multivariate PPA, optimizes a multivariate form of kurtosis in several dimensions simultaneously and is better suited to certain kinds of data sets.[40] A similar iterative learning rule is employed.

Kurtosis is an effective projection index because it is a quantitative metric of normality, and non-normal projections are usually the most interesting. Moreover, it is independent of scale, which makes it well-suited for exploratory analysis, but it has a complex relationship with the shape of a distribution. Figure 5.1 shows kurtosis values for selected distributions. A normal distribution has a kurtosis value of 3, while flatter distributions tend to have lower values and more tailed distributions have higher values. In particular, minimization

of K along each dimension tends to partition the data into two groups, if such partitioning is possible. A limiting value of unity results when two balanced groups are separated into infinitely narrow distributions, as shown in Figure 5.1(i). Because it is not based on variance, previous studies have demonstrated that PPA methods can separate classes where PCA and HCA fail.[40,55,56,59,61,135,136]

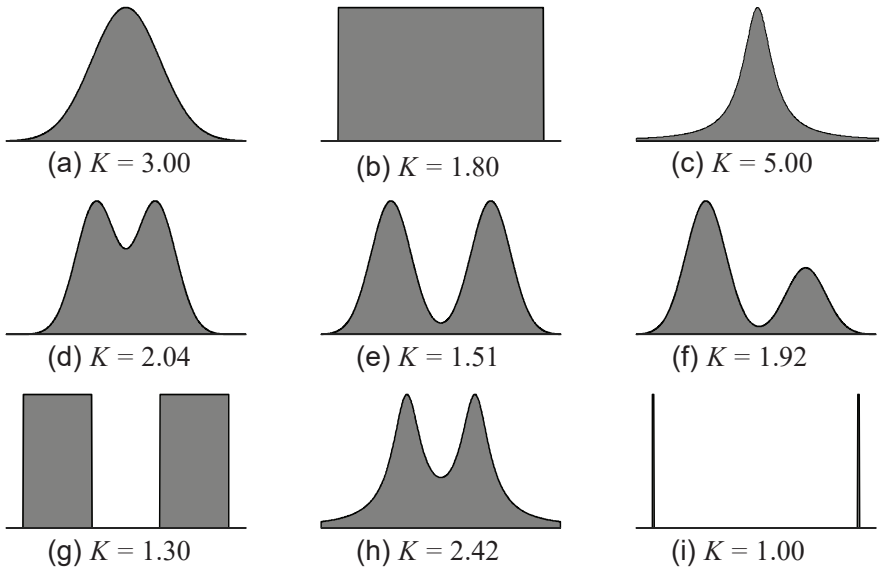

Figure 5.1: Kurtosis values for selected distributions: (a) normal, (b) uniform, (c), truncated Cauchy, (d) mixed normal, (e) mixed normal, (f) unbalanced mixed normal, (g) mixed uniform, (h) mixed truncated Cauchy, (i) mixed delta.

Fat Data. Despite their advantages, one weakness of PPA methods is a susceptibility to overfitting when the number of variables exceeds the number of objects (samples), *i.e.* so-called "fat" data matrices where $m < n$. This is not overfitting in the usual sense of supervised classification methods, since no class information is provided to the algorithm. Rather, the excess of data allows PPA to arbitrarily partition objects using spurious correlations in the variables. Ideally, the ratio of objects to variables should be greater than 5, but this condition is widely violated for chemical data. Effective solutions to this problem include matrix regularization[56] and compression of the original data using PCA,[59,136] although both approaches require some optimization of parameters and run the risk of losing information. Additionally, with PCA, interpretation is further complicated because the projection vectors are linear combinations of the loading vectors instead of the original variables. To address both the interpretability and overfitting problems associated with PPA, a sparse implementation is proposed here.

Sparse PPA. While PCA is not subject to the same object-to-variable issues as PPA, sparse implementations of PCA have been proposed to improve the interpretability of the loading vectors.[139–143] Such methods impose a penalty function in the PCA decomposition that favors a reduction in the number of variables represented in the loadings, making meaningful associations with chemical measurements easier. A similar sparse implementation of PPA was sought in this work, but because of difficulties in incorporating a penalty function into the quasi-power algorithm, an alternative strategy based on variable selection was employed. The approach is similar to other variable selection methods used for calibration and classification, but because it is based on kurtosis for selection rather than the quality of fit to a training set, the integrity of PPA as an unsupervised method is maintained. The objective is to choose a subset of the original variables that minimizes the kurtosis obtained through PPA. Although many variable selection methods have been employed for other methods,[144–147] it was felt that the use of a genetic algorithm (GA) was the most compatible with PPA.

Genetic Algorithm. While the use of GAs for variable selection is not new,[148] they are well-suited to PPA where the variables in the subset are considered collectively rather than one at a time. With this approach, an initial population of chromosomes is established containing genes representing the n variables in $\mathbf{X}$, as shown in Figure 5.2. If a gene is set to 1, the variable is selected, so each individual in the population represents a different subset of variables. The initial population of N individuals is created so that p of the n variables in each chromosome are randomly selected. These are adjustable parameters, but the default values for SPPA are $N = 100$ and $p = 5$. It was found that setting $p \approx m/25$ (with a minimum of 3) gave good performance over a wide range of data sets.

Once the initial population is set, the fitness of each individual is evaluated by performing PPA using the subset of p variables and returning the optimized kurtosis value, κ. The optimization is fast since the dimensionality is typically small and only one initial guess is used. Normally, multiple initial guess (*ca.* 100) would be used for PPA to ensure a global optimum, but this is not required here since the optimization is more reliable for smaller dimensions and individuals retained in subsequent generations are re-evaluated, with the minimum value of κ retained. In this way, only the best subsets are reassessed.

The individuals in the population are ranked at this stage and parents to be used for recombination (exchange of variables) are selected using a standard roulette wheel approach.[149]

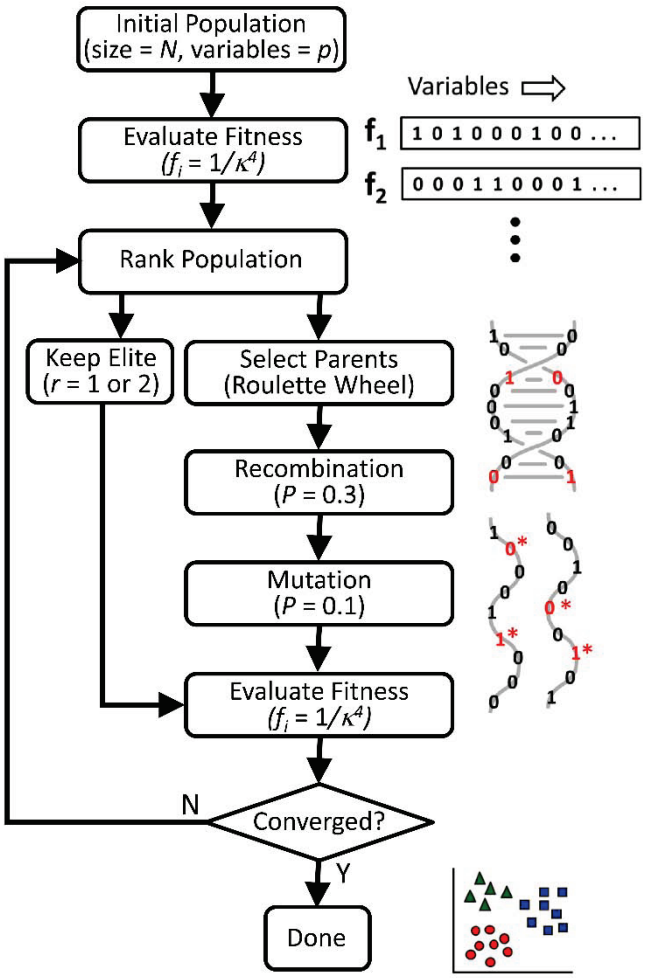

Figure 5.2: Genetic algorithm for variable selection in SPPA.

For this purpose, the fitness is defined as $1/\kappa^4$, with the inversion and transformation optimized for the ranking step. In this calculation, kurtosis values below 1.5 (or 4.5 for multivariate kurtosis) are set to this limit, since they are considered equally viable for class separation (see Figure 5.1) and this helps prevent premature elimination. For even N, $N-2$ parents are selected ($N-1$ for odd N) using the roulette wheel strategy, in which the probability of the selection favors individual with high fitness (small κ). Recombination (mating) occurs by random pairing of the selected parents followed by random exchange of selected variable with a probability of $P = 0.3$. The resulting children ($N-2$ or $N-1$) then undergo mutation where there is a 10% probability that each variable will be substituted with another variable not already in the subset. The probabilities for recombination and mutation are adjustable, but these values have been found to be effective for a range of data sets.

The children produced in this process are united with r "elite" individuals which had the highest fitness in the parent population. For even N, the original size is retained with $r = 2$, whereas $r = 1$ for odd N. These individuals are reintroduced to ensure that the best solutions are not eliminated through the recombination and mutation steps. The fitness of this new population of N individuals is evaluated in the same way as the original population and the process is repeated until the solution converges (no change in the best solution for 50 generations) or a time limit is exceeded. Upon convergence, the selected optimum variable subset is re-evaluated with multiple starting points to ensure the global optimum has been found.

Another parameter specified in the optimization is the number of dimensions (projection vectors) to be extracted, normally one to three. For stepwise univariate PPA, successive dimensions are extracted after deflation of $\mathbf{X}$. For multivariate PPA, the projection vectors are extracted simultaneously.

In addition to the projection vectors, the final population provides information about the variables that are most useful for separating the classes. Alternatively, SPPA can be applied multiple times and the optimal solutions can be used to extract this information.

5.3 Experimental

5.3.1 Data Sets

To demonstrate the utility of the SPPA algorithm for different types of multivariate chemical data, four experimental data sets were selected and are summarized in Table 1. The data sets are available in the public domain. Three of these are spectroscopic data sets (Vis/NIR, IR, NMR) and one contains compositional data. The data are characterized by different numbers of samples (48-480), variables (47-3351) and classes (3-8) to be as representative as possible. As well, one contains unbalanced classes. Further information on the data is provided in the Results and Discussion and in the original references, given in Table 5.1.

5.3.2 Computational Details

All calculations were performed using programs written in-house with the MATLAB programming environment (MathWorks, Natick, MA).[81] Code for implementing the proposed method is available in Appendix C. As standard practice, PCA and SPPA were applied to the data after column mean centering. In some cases, scaling of the columns

Table 5.1: Summary of Data Sets.

Data Set	No. of Classes	No. of Samples [a]	No. of Variables [b]	Ref
1. Wine Grape (vis-NIR)	3	250 (100,100,50)	256 (10)	[150]
2. Ink (FTIR)	8	480 (8×60)	3351 (15)	[151]
3. Salmon (NMR)	5	75 (5×15)	1117 (5)	[152]
4. Pacific Cod (Fatty Acids)	4	48 (4×12)	47 (3)	[153]

[a] The class partitions of the samples are shown in parentheses.
[b] The number of variables extracted for each dimension of SPPA is shown in parentheses.

by the column standard deviation (*i.e.* autoscaling) was also applied for PCA if superior results were obtained. Scaling has no effect on the SPPA results.

5.4 Results and Discussion

5.4.1 SPPA vs. PCA

In exploratory data analysis, which is by definition unsupervised, there is no defined optimal subspace projection. However, where class information is available, projections which establish the separation of classes are generally viewed as supporting the hypothesis that the classes are separable on the basis of the measurement employed. In many cases, techniques such as PCA and HCA are sufficient to support this hypothesis, but the examples presented here demonstrate that this is not always the case and PPA methods present a powerful alternative.

The Wine Grape Data Set consists of visible/near-infrared (Vis-NIR) spectra of 100 samples each of *carignan* (*ca)* and *grenache blanc* (*gb*), and 50 samples of *grenache noir* (*gn*) grapes. This represents a typical case where, as shown in Figure 5.3(a), PCA fails to produce a clear separation of the classes. The figure shows the results with only mean-centering of the data prior to PCA, but the use of auto-scaling did not improve class separation, even after the exclusion of baseline regions. In contrast, Figure 5.3(b) shows that the same data set produces a definitive separation of classes when subjected to SPPA (stepwise univariate PPA with 10 variables selected), although there is some contamination of the *gb* class with the *ca* class, which may be due in part to unbalanced classes. Nevertheless, the projection clearly indicates that the data contain information about the grape variety.

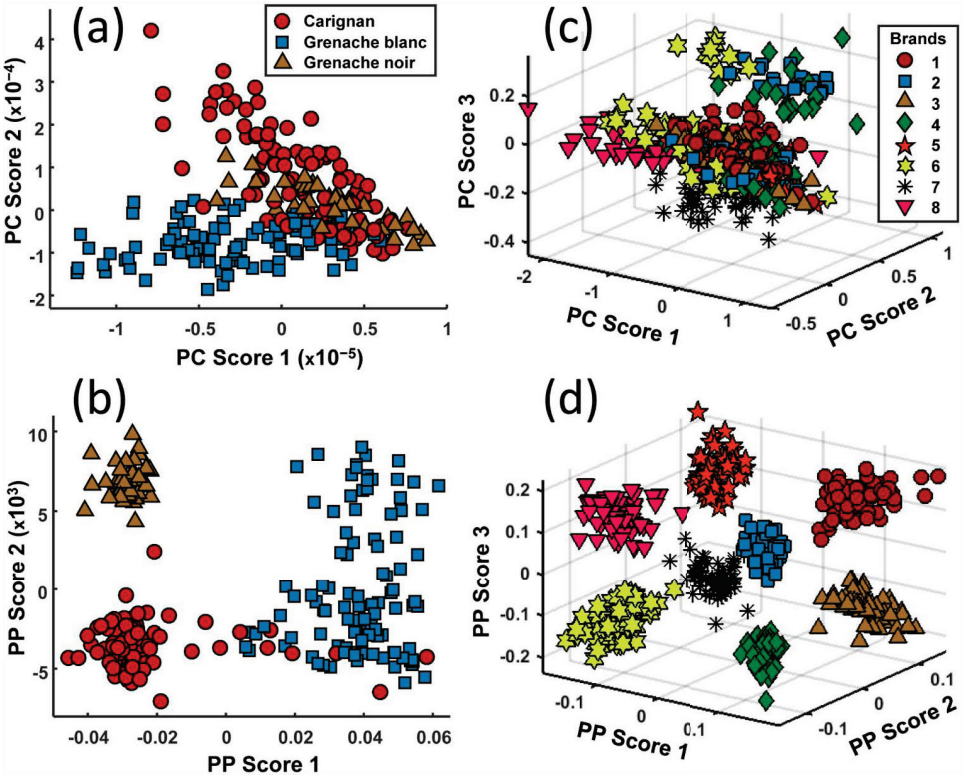

Figure 5.3: Projections for Wine Grape and Ink data sets: (a) PCA scores for Wine Grape data, (b) SPPA scores for Wine Grape data, (c) PCA scores for Ink data, (d) SPPA scores for Ink data.

The Ink Data Set provides perhaps the most compelling example of the capabilities of SPPA. These data, consisting of IR spectra of eight brands of pen inks, were collected in a carefully designed experiment intended to establish the potential of IR spectroscopy to detect forgery. The data set is both balanced (60 samples per group) and binary ($2^3 = 8$ classes) so it is well-suited to analysis by stepwise univariate SPPA. The best PCA results, shown in Figure 5.3(c), were obtained with mean-centered data. Although not obvious from the angle of the figure, which was oriented for maximum visibility of all of the classes, brand 7 is well separated from the rest of the brands and brand 8 is partially separated. Some other groupings are evident, but no clear separation of the eight brands is apparent. On the other hand, the SPPA projection (with 15 variables) shown in Figure 5.3(d) reveals a distinct separation of all eight pen brands. For an unsupervised method, it is remarkable to achieve such a clear partitioning of a large number of classes. Although such results are not universally achieved, they do indicate the potential when metrics other than variance are employed in exploratory analysis.

SPPA can also be applied to multivariate kurtosis, which is often better suited for unbalanced and/or non-binary separations. Although multivariate PPA seeks to achieve a radial separation of the objects rather than clusters, separation of classes is often a fortuitous consequence. The Salmon data set, consisting of NMR spectra of salmon blood plasma from a metabolomics study, serves to illustrate this point. The aim of the original study was to examine the sources of variation in the NMR data using 15 replicate blood samples from each of five individual fish that served as biological replicates. It was of interest to know if the analysis of the data would show unique characteristics of the individuals. The PCA results for autoscaled data (one outlier removed) presented in Figure 5.4(a), show some class structure, but PCA fails to separate three of the fish from each other. In contrast, the multivariate SPPA results presented in Figure 5.4(b) (using only five variables) show a clear separation of the replicate samples from each individual, indicating baseline differences in their metabolic profiles.

Another type of multivariate data commonly encountered consists of discrete variables, such as elemental concentrations, where optimal scaling for PCA or HCA can be challenging. Scaling is usually necessary in these cases to account for different variable ranges, but this can amplify the effects of noisy variables. Since PPA is not scale dependent, these issues are removed. The Pacific Cod Data Set, consisting of fatty acid (FA) profiles for 48 fish serves to illustrate this. As part of a larger study, the relative abundances of 47 fatty acids (normalized to 100%) were determined in Pacific cod sampled at two sites, Graves Harbor (Site A) and Islas Bay (Site B), in the Gulf of Alaska in 2011 and 2013. Mean FA proportions ranged from about 0.02% to 30%. The best PCA results, shown in Figure 5.4(c), were obtained with autoscaled data. The figure shows a reasonable grouping of the four classes (with the exception of one sample), although the boundaries between groups are not clearly defined. In contrast, the SPPA projection in Figure 5.4(d) shows a distinct division of all four groups by year (dimension 1) and location (dimension 2).

5.4.2 Variable Selection

Prior application of PPA suggests that the sample-to-variable ratio should be at least 5:1 to avoid overfitting. This condition is violated for all of the data sets examined here, so some type of variable reduction was necessary. Variable compression through PCA is one approach but requires some optimization to ensure that the necessary class information is included in the retained components.[136] Moreover, the chemical interpretation of the

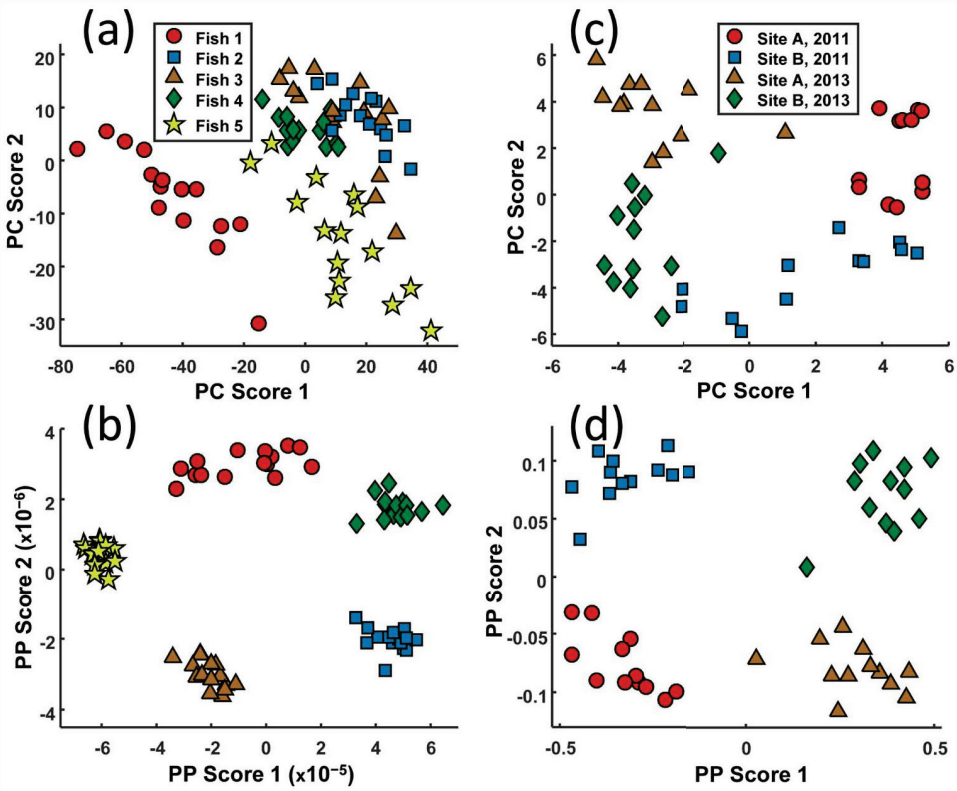

Figure 5.4: Projections for Salmon and Pacific Cod data sets: (a) PCA scores for Salmon data, (b) SPPA scores (multivariate PPA) for Salmon data, (c) PCA scores for Pacific Cod data, (d) SPPA scores for Pacific Cod data.

resultant loadings is confounded by the linear transformation and the important variables are hard to identify. SPPA is an alternative to reduce the number of variables and provide direct interpretability.

The analysis of variables selected by SPPA is complicated by the nonlinear nature of the algorithm and correlations among the variables. SPPA uses multiple starting points for each run to avoid local minima and, while multiple applications to the same data set produce consistent class separation, it is probable that no two will be identical or result in the extraction of exactly the same set of variables. Therefore, rather than identifying the variables from a given run, characterization typically involves the application of SPPA multiple times (usually 100) and examination of the aggregate of variables chosen. For spectra, these data can be represented in terms of the number of times variables were selected, revealing not only the most useful variables, but also bands where correlated variables may have substituted for each other in different runs.

Examples of this representation are presented in Figure 5.5, where the spectra (mean for

each class) are superimposed on a heat map showing the number of times variables were selected over 100 runs. Figures 5.5(a–c) show the variables selected for each dimension of the Ink data. Variables with counts below a set threshold (< 3 in this case, based on a binomial distribution) were excluded. Visual inspection of the results of the 100 trials confirmed similar separations of the classes, although orientations and exact sample positions varied from run to run.

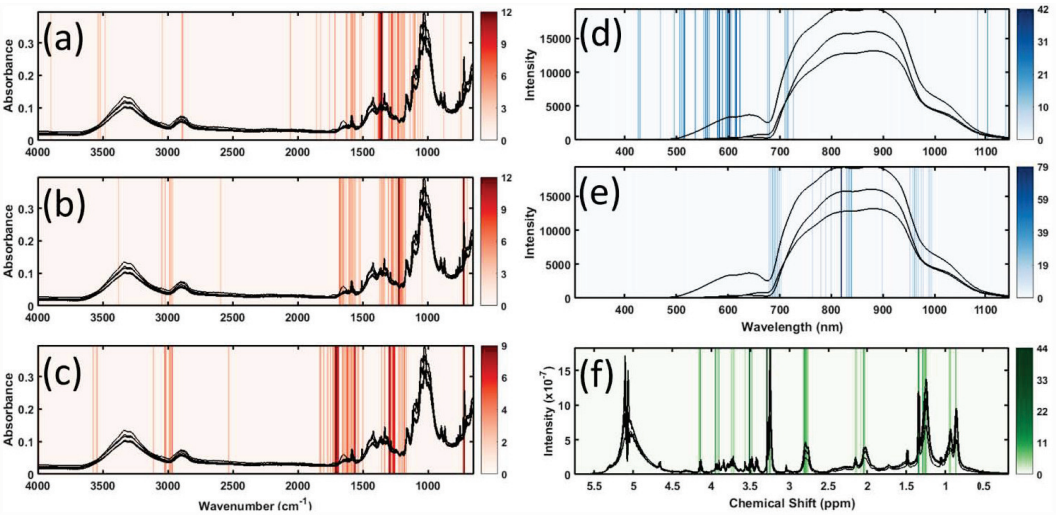

Figure 5.5: Variable selection counts for 100 SPPA runs. (a)-(c) Ink data set (dimensions 1-3), (d)-(e) Wine Grape data set (dimensions 1 and 2), (f) Salmon data set. Mean spectra for each class are superimposed on heat maps.

Several regions of interest can be identified for the Ink data and, not surprisingly, most are in the fingerprint region below 1700 cm^{-1}. Although distinct regions are more strongly associated with certain dimensions (*e.g.* 1360, 1220 and 1730 $\text{cm}^{-}1$), selected channels may appear across multiple dimensions because the dimension order can be permuted on successive runs, leading to "cross-talk" in the variables selected for each dimension. However, such a mapping indicates which variables or regions contain important information for interpretation or the development of supervised classification methods.

A similar mapping is shown for two dimensions of the Wine-Grape data in Figures 5.5(d–e). In this case, fewer variables were available (256), so the counts are higher and the threshold was set to 10. Variables for the separation of *gb* from the other two varieties (dimension 1) are mainly in the 500-650 nm region, which was anticipated based on spectral differences. The second dimension, separating *gn* from *ca*, is dominated by the signal at about 820 nm, but includes other channels in this region, as well as around 700

and 1000 nm.

The analysis of Salmon data was carried out using multivariate PPA, so only one set of variables is extracted in each run, even though the scores plot shows two dimensions. The variable map in Figure 5.5(f) shows, not surprisingly, that the selected variables typically align with peaks in the NMR spectrum, but some align with very specific features (*e.g.* 3.282 and 3.512 ppm) while others are part of a broader regions (*e.g.* 2.8 and 1.3 ppm). Since only five variables were required for each run, the NMR features appear to contain some redundancy.

For discrete/compositional data such as the Pacific Cod data, density maps are not as useful because there is not a local correlation of variables. Application of the SPPA method over 100 trials produced 8 unique sets of six variables (three in each dimension), with each solution having very similar quality of separation (K = 1.26 to 1.30). Of these, two sets represented 82% of the solutions and included the global optimum. The variables selected for this optimum are shown in Figure 5.6, along with the mean FA compositions for each class. The FAs have been ordered by mean concentration and a log scale is used due to the large data range. Separation by location (second dimension) employed variables 3 (FA 16:0), 35 (17:1) and 45 (16:3 n-4), with the first two variables present in 100% of solutions. The separation by year in the first dimension (K = 1.26) used variables 8 (22:5 n-3), 9 (16:1 n-7), and 23 (iso-16:0), but an equally frequent solution (K =1.28) employed an alternate set of variables (5, 20, 41). These results indicate that a small number of variables can be used to distinguish the classes, although direct interpretation may be complicated by the complex correlations introduced by the row normalization of the original data.

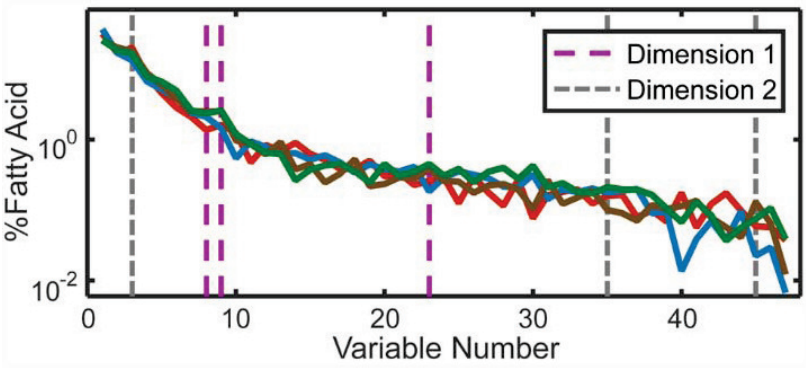

Figure 5.6: Variables selected for Pacific Cod data superimposed on mean FA concentrations for each class.

5.5 Conclusions

Data visualization methods are critical for discovery in modern chemical measurement science. As demonstrated here using diverse data sets, SPPA is a powerful method that extends the toolbox of exploratory methods beyond variance based techniques like PCA. It is not intended to replace supervised classification methods, but provides a much needed alternative to support hypotheses and guide variable selection and interpretation.

As demonstrated here, SPPA provides a solution to the two problems commonly encountered in the application of PPA methods. The first is the problem of variable reduction. Other methods used to address this problem, PCA and matrix regularization, risk losing information by excluding variables whose variance is too small. By considering all variables with equal weight, SPPA avoids this problem. A second advantage of SPPA is that it simplifies the interpretation of results. This is a problem with all dimensionality reduction methods in that it is difficult to provide a chemical rationale for the observed projection (*i.e.* which variables are important and why). While some claim that PCA loading vectors provide this ability, in practice this is dubious because of the large number of variables involved and the complex correlation represented in the loading vectors. SPPA indicates the relevant variables directly, which is more likely to facilitate rational interpretation. Moreover, selected variables provide an indication of which regions are most likely to be most useful for supervised classification methods.

Despite these gains, there are still some limitations to the PPA methodology. With regard to SPPA specifically, the variation in selected variables from one run to another hinders the interpretability. From a broader perspective, PPA algorithms also have limitations when unbalanced classes (unequal class membership) are used, or non-binary class separation is required (*i.e.*) other than 2, 4, or 8 classes). These issues are addressed in the chapters that follow.

CHAPTER 6

BANDED SPARSE PROJECTION PURSUIT ANALYSIS

This chapter presents a modification of the SPPA algorithm, called the banded sparse projection pursuit analysis algorithm, that is shown to provide improved results when applied to ordinal data (*e.g.* spectroscopic data). Although the SPPA algorithm was found to provide improved clustering results on a wide variety of multivariate chemical data sets, it lacks some interpretability and efficiency when applied to data sets with a large number of response variables because of the large search space. This can cause sub-optimal solutions to be obtained, which is indicated in some instances by some selected variables clearly not being important in revealing clusters (*e.g.* a variable in an area with zero signal for all samples in the data set). Here, instead of treating each response variable as a gene in the genetic variable selection algorithm, I propose reformulating the algorithm to treat a band of neighboring variables, with a pre-determined size, as genes that comprise the individuals. This strategy can be rationalized for multivariate ordinal chemical data by asserting that neighboring response channels are likely to contain similar information up to some correlation limit. The proposed algorithm is shown to be able to provide projections competitive with SPPA while also providing improved variable interpretation.

6.1 Introduction

As discussed in Chapter 5, PPA is a capable alternative to PCA[4] and HCA[5] for the exploratory analysis of chemical data. The sparse implementation of PPA proposed in Chapter 5 (SPPA) was shown to be an effective solution to the sample-to-variable ratio

problem that was previously overcome using PCA compression. The added benefits of sparse PPA (SPPA) are that (1) there is no longer a need for PCA compression prior to applying PPA, and (2) variable interpretation is possible through genetic variable selection. Although the SPPA algorithm was found to provide improved clustering results on a wide variety of multivariate chemical data sets, there are cases where some variables selected by the algorithm are clearly not relevant to the associated low-dimensional clustering. This is due to either the retention of an excess number of variables, or to premature convergence of the algorithm, both of which are inefficiencies in terms of algorithm performance and hinder the interpretability of the variable selection. To address this observation, I restricted the genetic variable selection algorithm to produce only solutions that contain bands of variables since, intuitively as chemists, we know that neighboring channels in ordinal data are expected to contain similar information.

All plots of the variables selected by SPPA when applied to ordinal data in Chapter 5 clearly show that selected variables in repeated runs form bands in the spectra. This is a reassuring result, as each variable is considered a "gene" in the SPPA variable selection algorithm and there is nothing in the algorithm forcing this behaviour. As mentioned in Chapter 5, this suggests that when we observe neighboring variables being selected and clusters being revealed in the resulting projection, that these variables contain similar information. However, in these same figures, there are variables selected that do not form bands and may not be important to the observed clustering. The purpose of this chapter is to describe a modification to the SPPA algorithm that restricts the SPPA algorithm to selecting bands of variables. This modification should result in more efficient convergence as well as improved variable interpretation for ordinal data.

6.2 Theory

A brief overview of sparse projection pursuit will be given as it forms the core of the banded sparse projection pursuit algorithm.

6.2.1 Sparse Projection Pursuit (SPPA)

In SPPA, the objective of the genetic variable selection algorithm is to find sets of variables that minimize the kurtosis of the optimized projection vector obtained by applying ordinary PPA to the sparse data (the data with all variables other than the selected variables cut out).

These variables are considered genes and the individuals are comprised of a set of these variables. To begin the genetic algorithm, individuals are created by randomly selecting subsets of the original variables (with a size appropriate for the number of samples). The fitness values of the individuals are then evaluated ($f = 1/K^4$) and the recombination procedure begins, where individuals with high fitness (low kurtosis) are more likely to pass on their genes (variables) to their children. The children are obtained by recombining two individuals from the parent population. This involves swapping, at random, a certain number of variables (genes) between the two parents to produce the children. The children produced by recombining the parent vectors comprise the new population. To help avoid local optima, each selected variable has a small chance of being changed to another variable from the original data set. Finally, elite individuals (the top 1 or 2 individuals with the lowest kurtosis) are passed onto the next generation without alteration to ensure that the best current solution is not lost through recombination or mutation. This procedure is repeated until there is no change in the population over a defined number of generations.

6.2.2 Banded Sparse Projection Pursuit (banded SPPA)

The benefit of implementing the proposed method in the framework of SPPA is that only the definition of the gene and individual need to be changed in the context of the genetic variable selection algorithm. Instead of treating each response variable as a possible gene, the goal of banded SPPA is to impose a constraint that treats bands of neighboring variables as single genes. To impose this constraint in the framework of SPPA, two parameters must be defined: 1) the width of the band, w, and 2) the number of bands, a.

The algorithm for banded SPPA is a simple adaptation of the SPPA algorithm. Based on prior knowledge of the system under study, the user decides on an appropriate number of measurement channels to include in a band, w. For systems with information condensed into relatively narrow regions (*e.g.* sharp peaks, such as in NMR spectroscopy), the size of the band may represent a smaller fraction of the spectral range than for cases where the information is more dispersed (*e.g.* UV spectroscopy). For ease of implementation, the banded SPPA algorithm has been designed with bands of uniform width and position, but it could be readily adapted to custom widths and positions for specific applications. This could specifically exclude spectral regions or even combine non-contiguous regions if appropriate for a given application.

A possible drawback of the proposed method is that, depending on the band size

and number of bands, the number of variables can increase quickly and could lead to overmodelling from PPA. However, carefully choosing a and w such that an acceptable sample-to-variable ratio is achieved should make this a non-issue for most data sets. Another alternative would be to optimize the band width and size during the GA process. This creates certain algorithmic complications, such as how to carry out the evolutionary operators, how to deal with one band impinging on another, and the limits on the actual band size. It is also likely to slow the optimization. In the end, it was decided that the simplicity and efficiency of the fixed band size implementation outweighed the potential advantages of more elaborate approaches, especially at the proof-of-principle stage. Nevertheless, future refinements may improve the overall approach.

Once the banded variables have been assigned, the GA proceeds in the same way as SPPA, with the banded variables set used to define the genes.

6.3 Experimental

6.3.1 Data Sets

To demonstrate the performance of the banded SPPA algorithm relative to the original SPPA algorithm, the Wine Grape Vis-NIR Data Set and the Salmon ^{1}H NMR Data Set from Chapter 5 are once again employed.

Wine Grape Vis-NIR Data Set. This data set contains three classes of wine grapes analyzed by Vis-NIR (*Grenache noir* (50 samples), *Carignan* (100 samples), and *Grenache blanc* (100 samples)).[150]

Salmon ^{1}H NMR Data Set. This data set contains five classes of individual atlantic salmon analyzed by ^{1}H NMR. Each class contains 15 samples except one class, which only contains 14 samples.[152]

6.3.2 Computational Details

All calculations were performed using programs written in-house with the MATLAB programming environment (MathWorks, Natick, MA).[81]

6.4 Results and Discussion

The results of applying banded SPPA to the Wine Grape Vis-NIR Data Set are shown in Figure 6.1. In this case, SPPA was limited to selecting $a = 2$ bands ($w = 5$, which

corresponds to 15 nm) in each of the two dimensions. Figure 6.1A shows the bands selected by the algorithm for dimension 1 (red) and dimension 2 (blue). In the first dimension (PP score 1) two distinct, non-overlapping, bands (535-550 nm and 730-745 nm) were found to be selected by the algorithm. In dimension 2 (PP score 2), two distinct non-overlapping bands were selected (816-831 nm and 965-980 nm). The corresponding projection is shown in Figure 6.1B, where a projection of similar quality to the SPPA results in Chapter 5 is observed. Although the discrimination between *Grenache noir* and *Carignan* is worse in the present implementation, this is not surprising as the banded implementation of SPPA has less of an opportunity to fine tune the kurtosis value as it is operating in a restricted variable space.

Applied to the Salmon ^{1}H NMR Data Set, banded SPPA optimizing the multivariate kurtosis in 2 dimensions is able to cluster all five classes in a similar projection obtained by ordinary SPPA, as shown in Figure 6.2B. However, the banded SPPA projection was found by limiting the algorithm to select a single band $w = 5$ (0.05 ppm) in size. The banded algorithm selects a band of variables around 3.3 ppm, as shown in Figure 6.2A. In this case, since only one band is being selected, the algorithm is equivalent to sliding a $w = 0.005$ ppm band across the variables and determining which region gives the smallest kurtosis, with no real need for applying the steps of the genetic algorithm.

A closer look at the band selected in Figure 6.2A reveals that the banded SPPA algorithm is picking a shoulder region of a peak at 3.3 ppm. One might expect the algorithm to position the band over the full peak somewhere in the 3.25–3.3 ppm region, however, shifting the band to include more of the actual peak must increase the kurtosis obtained by the algorithm since this was not the result observed in practice. This could be due to the fact that the real chemical information that is discriminating these classes is contained only in the shoulder of the peak, or that because the banded SPPA algorithm is operating in a reduced search space, including the variables to the left or right of the obtained band position results in a higher kurtosis value. This suggests that some fine-tuning of the band size might be needed to further interpret the results of banded SPPA. However, as proof-of-principle, it is clear that the banded algorithm is obtaining a projection of similar quality to SPPA without the additional selection of dubious variables.

To determine the important variables involved in the projection obtained via ordinary SPPA, the algorithm must be applied repeatedly and the resulting selected variables must

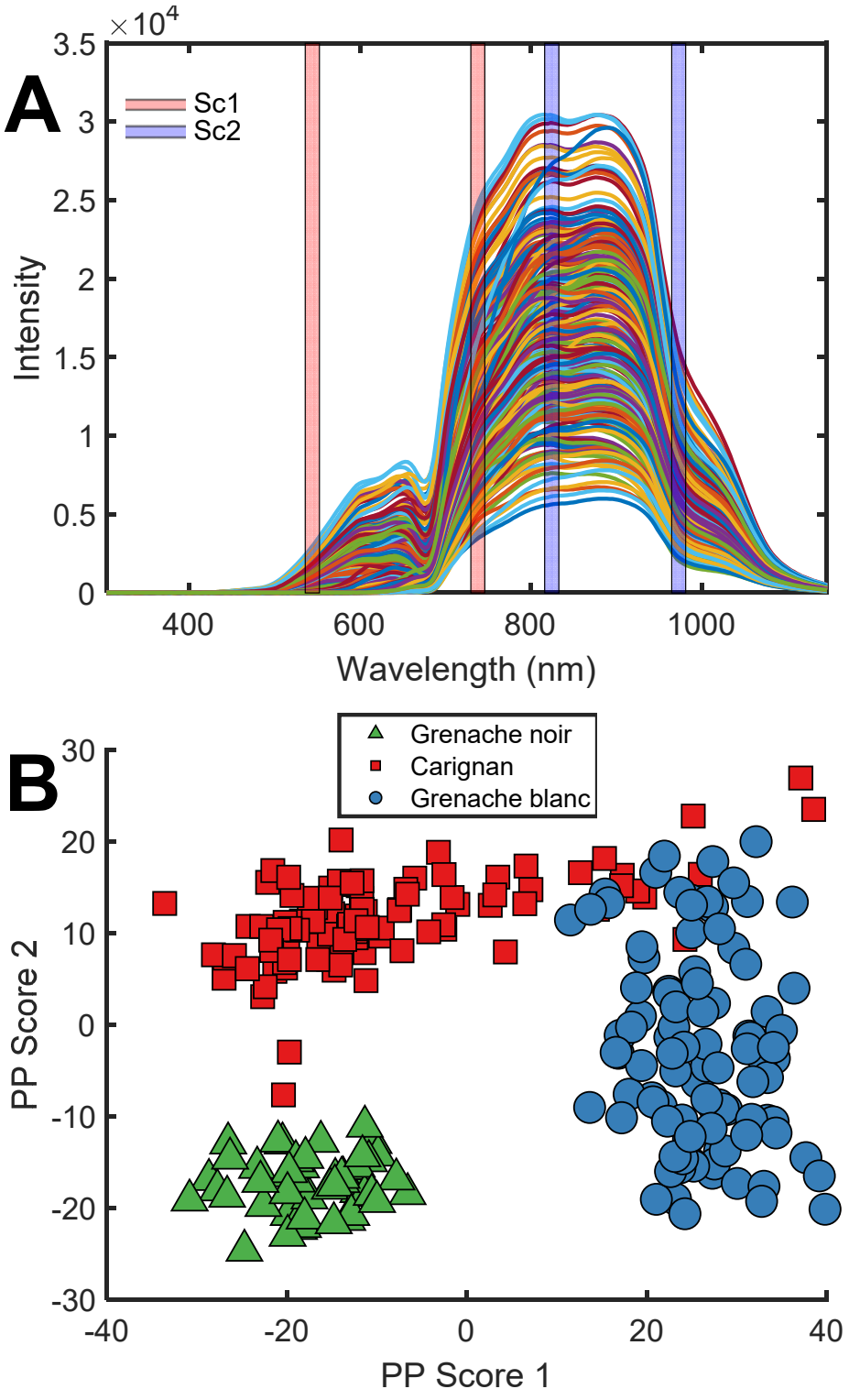

Figure 6.1: Result of applying banded univariate SPPA to the Vis-NIR Wine Grape Data Set with $a = 2$ bands, $w = 15$ nm in size (5 variables), for each of the two dimensions.

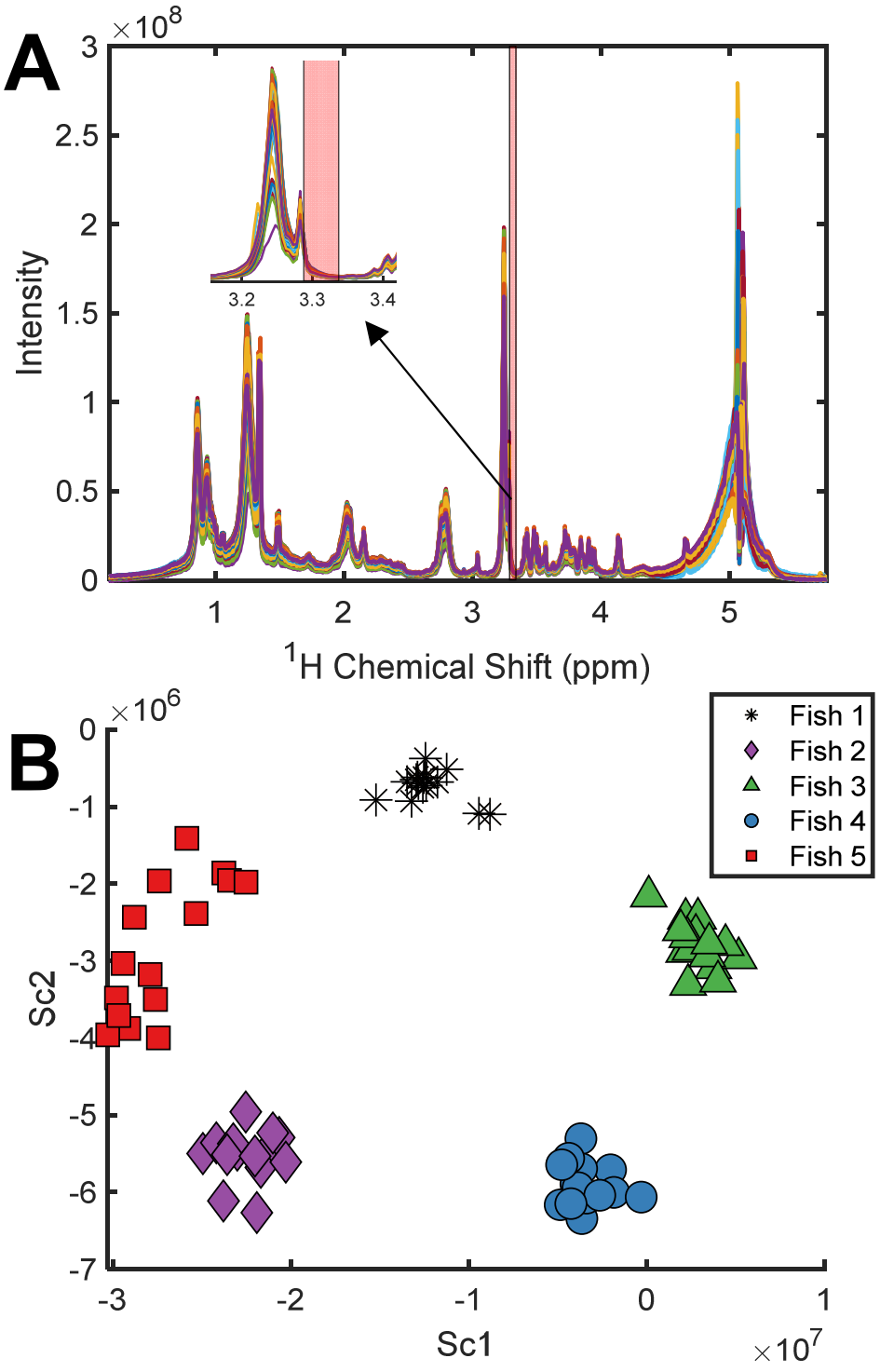

Figure 6.2: Result of applying banded multivariate SPPA to the salmon data set ($a = 1$ band, $w = 0.05$ ppm in size (5 variables)).

be stored and analyzed. For banded SPPA, since the search space is much more confined, the selected variables should be more stable (a smaller number of unique variables selected over 100 runs). In other words, the banded SPPA algorithm is less likely to converge on local optima. To explore the stability of the selected variables by banded SPPA relative to that of ordinary SPPA, banded SPPA was applied to the Wine Grape Vis-NIR Data Set 100 times. Figure 6.3 shows a histogram (bin size of 5) of the selected variables in each dimension over 100 runs for both ordinary SPPA and banded SPPA. Comparing the selected variables frequencies by ordinary SPPA in dimension 1 (Figure 6.3(a)) and banded SPPA in dimension 1 (Figure 6.3(c)) it is clear that the banded algorithm is providing more consistent solutions over repeated applications of the algorithm. The set of variables selected by banded SPPA over the 100 runs is dominated by two bands starting at 535 nm and 730 nm. It is important to note that these regions in the spectra have a modest amount of selection frequency in the ordinary SPPA results as well, but they are not the highest frequency regions. In dimension 2, the two bands selected by the single application of banded SPPA in Figure 6.1, 816 nm, and 965 nm, are also frequent in the 100 runs shown in Figure 6.3(d), but there appears to be an equally frequent band in the 690 nm region. These three regions are also the dominant regions in the ordinary SPPA dimension 2 frequency results shown in Figure 6.3(b). These results suggest that the banded SPPA algorithm can dramatically improve variable interpretation by providing important variable information without the need of excessive runs.

6.5 Conclusions

By imposing a restriction on the genetic variable selection algorithm in the SPPA algorithm which only allows a specific number of bands of neighboring variables to be selected, it was shown that the problem of defunct variables in ordinary SPPA can be mitigated. Additionally, the proposed modification to SPPA, termed banded SPPA, was shown to provide improved selected variable interpretation. The obtained results and performance of banded SPPA depends on two additional parameters, the width of the bands, w, and the number of bands, a. Although banded SPPA was only applied to select spectroscopic data sets, another possible, and immediate, application of this methodology is for hyperspectral images (feature enhancement, edge detection, etc.).

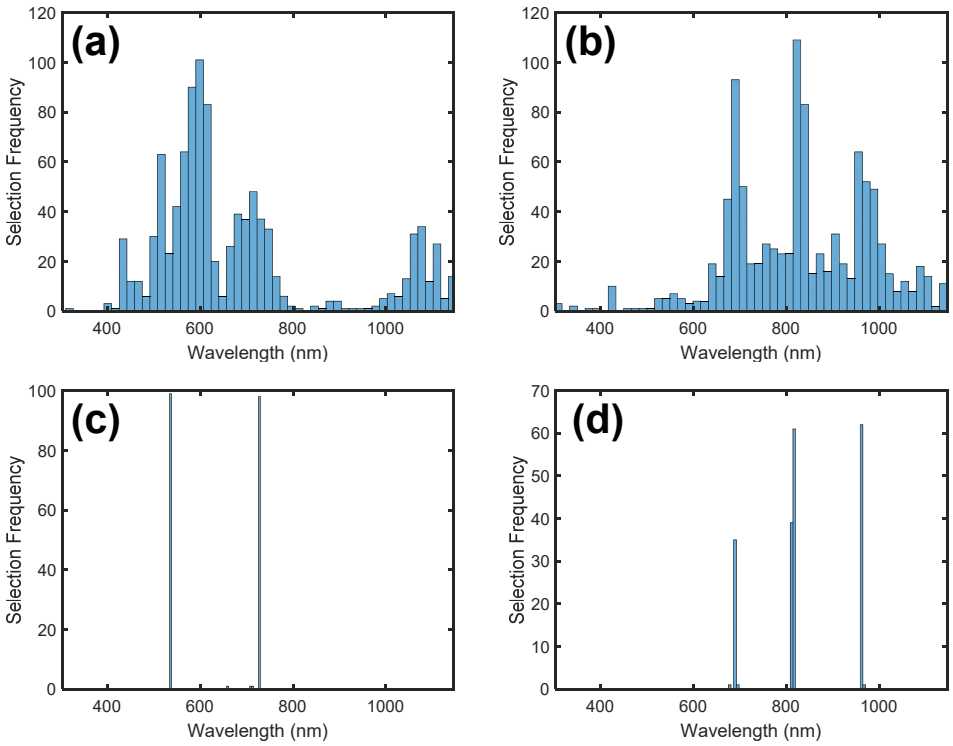

Figure 6.3: Histograms (bin size of 5, corresponding to a 15 nm width) of the selected variables in dimension 1 when applied Wine Grape Vis-NIR Data Set 100 times for (a) SPPA and (c) banded SPPA ($a = 2$, $w = 15$ nm). Data for dimension 2 is shown for (b) SPPA and (d) banded SPPA. It should be noted that for banded SPPA the histograms only reflect the frequency of the first variable in the selected band. As there are no bands in ordinary SPPA, all selected variables are represented in the histograms (a) and (b).

CHAPTER 7

AUGMENTED PROJECTION PURSUIT ANALYSIS

Kurtosis-based projection pursuit analysis (PPA), an unsupervised exploratory data analysis algorithm, has been shown to reveal meaningful clusters in multivariate chemical data that are inaccessible by traditional data analysis methods such as PCA and HCA. However, PPA often provides poor results when applied to unbalanced data sets (i.e. data sets that exhibit unequal class sizes) due to the nature of the projection index (univariate kurtosis). Additionally, since the stepwise univariate PPA algorithm is limited to a binary separation space in each dimension of visualization, there is a geometric restriction imposed on the analysis that favors data sets with a number of classes equal to a power of two (2, 4, 8). The work in this chapter demonstrates that it is possible to reveal clusters in an unbalanced data set using a simple augmentation strategy referred to as augmented PPA (augPPA). Additionally, it is shown that by applying augPPA with specific augmentation parameters it is possible to sequentially extract clusters from both simulated and real data. This sequential extraction using augPPA is referred to as seqPPA.

7.1 Introduction

Perhaps the area most lacking in method diversity in modern multivariate chemical data analysis is exploratory data analysis. This is unfortunate as exploratory methods are a key component in modern analytical chemistry applications such as metabolomics, clinical analysis, food analysis, forensics, and many others. Exploratory methods are often used to confirm the hypothesis that an experimental method is able to discriminate between classes

of interest (e.g. malignant vs. benign tumours) in cases where multivariate data make the application of traditional hypothesis testing difficult due to high false discovery rates. Principal components analysis (PCA) and hierarchical cluster analysis (HCA) dominate the scientific literature as the "go-to" exploratory methods due to their simplicity, performance, and intuitive visual presentation of results. Although these methods are useful for revealing information in multivariate data, such as clusters of similar samples (classes) or identifying possible outliers, they operate on variance (PCA) and distance metrics (HCA), which represent only two ways of displaying the complex information in the data in a reduced space. Therefore, it is important to realize that just because information to support a hypothesis is not found using one of these exploratory methods, it does not mean that the relevant information is not contained in the data. For example, in a multivariate data set designed to discriminate two classes (*e.g.* healthy and diseased), if the information required to separate these classes is expressed in variables that have relatively low variance, then PCA will not be able to reveal the class separation in the first few principal components. This is a problem inherent to the method, and although some judicious preprocessing of the data (autoscaling, for example) prior to applying PCA may increase the likelihood of revealing the class separation, all preprocessing methods have limitations based on their assumptions and may even make the data analysis results worse (*e.g.* by amplifying background noise). Rather than attempting the virtually unlimited options of preprocessing methods that may not work, a promising alternative is moving beyond the variance-based paradigm.

Kurtosis-based projection pursuit analysis (PPA), an alternative exploratory method proposed by Hou and Wentzell, offers a different approach to finding reduced representations of high-dimensional data. Univariate PPA is an unsupervised method that seeks a projection vector that minimizes the kurtosis of the samples when they are projected onto this vector. This tends to lead to projections that reveal binary separation of clusters in each dimension. One of the main advantages of PPA is that it requires little to no preprocessing of the data. This is due to the fact that kurtosis is not affected by changes in location or scale. PPA has been shown to perform as well as traditional methods and even reveal clusters in situations where they fail to do so. Despite this, PPA has not yet been picked up by the analytical chemistry community as a standard exploratory method for two main reasons. First, the development of PPA for applications in chemistry is in its

infancy relative to PCA and HCA, therefore, it has not seen the same general exposure. Second, and perhaps more important from a practical standpoint, is that PPA has some requirements that must be met for the algorithm to produce useful results.

One of the limitations of ordinary stepwise univariate PPA (the most useful algorithm) is that the classes in the data set should be balanced. In other words, there should be roughly an equal number of samples in all classes. This requirement is met for many data sets, especially simple two-class problems based on designed experiments. However, it is certainly not uncommon to have data sets that are unbalanced due to the differences in the availability of samples. In this case, the current implementation of stepwise univariate PPA may fail to correctly separate the unbalanced classes. This is because kurtosis is a simple univariate metric that depends on both the separation of object groups and the numbers within those groups. Thus, the minimization will attempt to partition objects into well-separated groups, but also make those groups equal in number. For unbalanced data sets, these two objectives work in opposition to one another and inevitably result in a compromise. This is evident through an examination of Figure 5.1 (Chapter 5). The unbalanced bimodal distribution in Figure 5.1(f) has a kurtosis almost equal to that of the balanced case in Figure 5.1(d), even though the latter is more poorly separated, and it is higher than that in Figure 5.1(b), which shows no separation. This problem becomes more pronounced as the differences in class membership become larger. This is an obvious limitation of the current implementation of ordinary PPA and, therefore, the first purpose of this work was to develop an improved method, termed augmented projection pursuit analysis (augPPA), for exploring unbalanced multivariate chemical data sets.

Due to the nature of the stepwise univariate kurtosis algorithm, another limitation of PPA is that the number of classes in the data set should be a power of 2 (2, 4, or 8), referred to as binary classification. This is connected to the problem of balanced classes. For example, if a data set contains four balanced classes of 50 samples each, the stepwise application of PPA would attempt to first partition the data into two groups of 100 (for example, by grouping classes 1 and 2 and classes 3 and 4). At the next stage, an orthogonal partitioning (*e.g.* separating classes 1 and 3 from 3 and 4) would also generate two groups of samples. Overall, this would result in a successful separation of four groups at the corners of a square in the space of the scores. However, if there were only three groups, the initial balanced partitioning is not possible, making the separation of clusters less likely (although

not impossible). The separation of three groups would be more feasible if class sizes were 100, 50 and 50, allowing the initial binary partitioning. This complex relationship between class size and number of classes is a problem that needs to be addressed if PPA is to be more widely applied. Therefore, a secondary purpose of this work is to demonstrate how augPPA can be applied to sequentially extract clusters from data sets regardless of the number of classes.

7.2 Theory

7.2.1 Kurtosis-Based Projection Pursuit

To formulate the proposed algorithm for augPPA, a brief overview of univariate PPA (minimization) is required. PPA seeks the projection vector $\mathbf{p}$ that minimizes the kurtosis, K, of the data projected onto $\mathbf{p}$. The univariate kurtosis of the projected data onto $\mathbf{p}$ can be written as

$$K = \frac{m \sum_{i=1}^{m} (\mathbf{p}^{\mathrm{T}} \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_i \mathbf{p})^2}{(\mathbf{p}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \mathbf{p})^2}, \tag{7.1}$$

where m is the number of samples, $\mathbf{x}_i$ is the ith row of the $m \times n$ column mean centered data matrix $\mathbf{X}$ and $\mathbf{p}$ is the $n \times 1$ projection vector. To minimize Equation 7.1 with respect to $\mathbf{p}$, it has been shown that the following learning algorithm can be applied:

$$\mathbf{p}_j(\text{new}) \leftarrow \left[\sum_{i=1}^{m} (\mathbf{x}_i \mathbf{p}_j)^2 \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_i \right]^{-1} \left(\mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j \right) \mathbf{p}_j. \tag{7.2}$$

Here, $\mathbf{p}_j$ indicates the projection vector for the jth dimension of the stepwise procedure and $\mathbf{X}_j$ represents the data matrix after deflation from the previous step. The PPA algorithm starts by providing a random guess for $\mathbf{p}_1$ (unit length) and then applying Equation 7.2 until the solution converges on a minimum. However, Equation 7.2 does not guarantee a global minimum, so several different guesses of $\mathbf{p}_1$ are used and the final projection vectors are screened for the smallest kurtosis value. The default number of guesses in the PPA algorithm is 100 as it seems to provide consistent solutions over a wide range of data sets without too much computational effort. This methodology provides one projection vector that tends to segregate the data into binary clusters. To extend this to multiple dimensions, the data matrix is deflated to remove the information associated with the

first dimension and that the scores in the second dimension will be orthogonal to those in the first dimension. The process is then repeated, starting with a random $\mathbf{p}_2$, to obtain a projection vector in the second dimension. Due to the nature of univariate kurtosis, the PPA algorithm results in projections that provide binary separations of classes in each dimension. This means that PPA will look to separate the data into two clusters in one dimension, and 4 and 8 clusters in two and three dimensions, but the projection on each dimension is binary.

7.2.2 Unbalanced Classes

In the case of class size imbalance, the current implementation of PPA may fail to find the projection where the unbalanced clusters are separated because the kurtosis of this projection is not a minimum. To illustrate this, a simple two-dimensional, two-class data set was simulated with a class ratio ranging from 1 to 5, where the size of the larger class, class 1, was held constant (200 samples). The samples were drawn from a bivariate normal distribution with mean of (-2, 0) and (2, 0) for class 1 and class 2, respectively. Both distributions had a covariance matrix with a diagonal of $\begin{bmatrix} 0.5 & 2 \end{bmatrix}$ and all off-diagonal elements set to zero. Figure 7.1 shows the optimal kurtosis value obtained using ordinary PPA (one dimension only) and the kurtosis of the ideal projection (i.e. the projection that discriminates the classes) as a function of the class ratio. When there are balanced classes (class ratio=1) the ideal and optimal projections have the same kurtosis value. As the class ratio increases there is a deviation in the kurtosis value for the optimal and ideal projections at a class ratio of about 3. After this point, the optimal projection obtained with ordinary PPA has a lower kurtosis value, but the classes can no longer be discriminated. Conversely, the ideal projection has a large kurtosis value, so it will never be obtained by the ordinary PPA algorithm. Although the point of divergence for the "ideal" and "optimal" solutions will depend on the distribution of the objects in multidimensional space for real problems, it is clear that there will be a performance breakdown at some non-unity class ratio.

There are two general approaches to solving the problem of unbalanced classes: altering the optimization algorithm or altering the data. Algorithmic modifications were considered first. One strategy would be to find an objective function which still targeted binary separation, but was less sensitive to class balance. Such an objective function would have to be carefully designed to avoid, for example, separating one or two samples as a single class. Even if such a function could be identified, a second problem is the

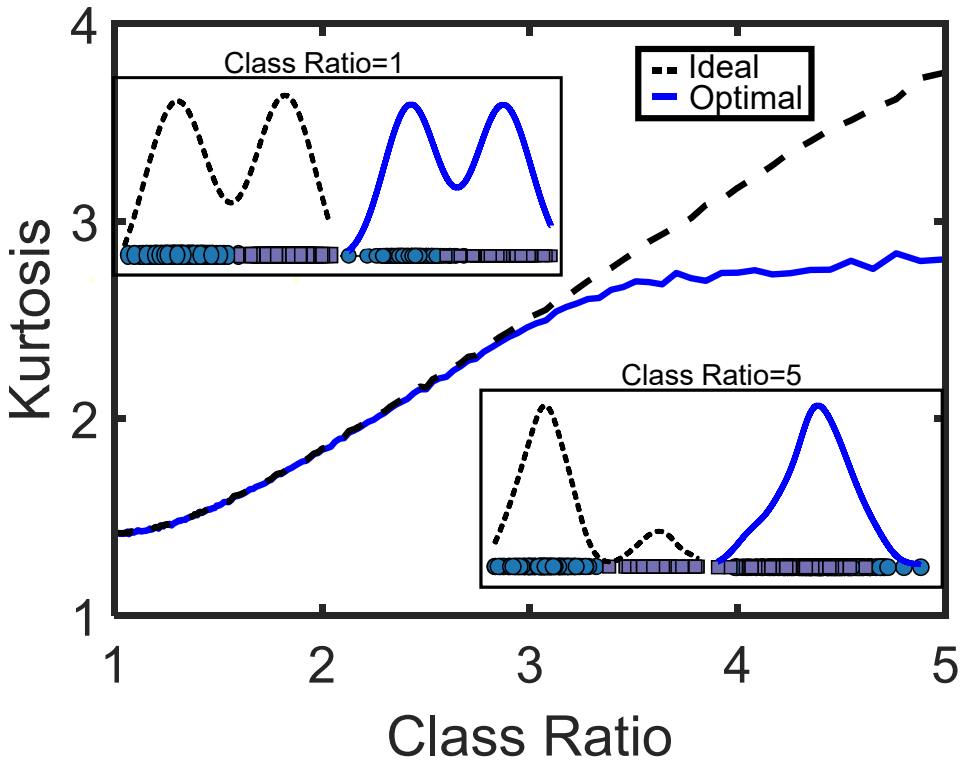

Figure 7.1: Mean univariate kurtosis for the optimal projection found with ordinary PPA (blue solid line) and the mean univariate kurtosis of the ideal projection (*i.e.* the projection that discriminates the classes, black dashed line) as a function of the class ratio between the two classes created through simulation (means are obtained from 100 realizations of the simulation).

efficient optimization, since it would be unlikely to be adaptable to the quasi-power method currently used. Another strategy is to adapt the the current kurtosis calculation to accommodate the class imbalance. While a weighted kurtosis calculation can certainly be carried out, this can only be done if the samples in the smaller class are identified in advance. However, identifying classes would make PPA a supervised method and defeat the purpose of exploratory data analysis. As there are much more efficient supervised methods this was not seen as a useful direction.

Earlier efforts at algorithmic modification for imbalance resulted in technique called recentered PPA (rcPPA).[55] This was based on the premise that the main issues associated with the imbalance could be addressed by redefining the population mean used in the kurtosis calculation. In this modification, a two-step learning algorithm is used to simultaneously define the optimal projection vector and the position of the multivariate mean. While this algorithm demonstrated some success, the simultaneous optimization of two vectors also made the optimization slower and less reliable, so alternatives were sought.

Failing efforts to develop a solution based on algorithmic modification, this work focused on modification of the data provided to the standard PPA algorithm through augmentation of the experimental data through resampling. This is described in the section that follows.

7.2.3 Data Augmentation Strategies

The principle of data augmentation is to artificially rebalance the classes by adding to the class with fewer objects (samples) in an unbiased way. To this end, the discussion will focus on data sets with only two classes. The extension of thesis ideas to multiple classes (sequential PPA) is discussed in Section 7.2.5.

Resampling is a standard statistical approach where new samples are drawn from existing data. In the current context several strategies can be employed. For the sake of illustration, a hypothetical case will be employed where class 1 ($\mathbf{X}_1$) is three times as large as class 2 ($\mathbf{X}_2$), with the two classes making up the entire data set, $\mathbf{X}$. The various strategies are illustrated in Figure 7.2 and described below.

The first strategy is one of subsampling of the larger class, perhaps more accurately described as sample elimination. If we simply discard sufficient samples from the larger data set, creating a smaller $\mathbf{X}_i$ from $\mathbf{X}$, as shown in Figure 7.2A, then balance can be restored. While this may work, it has a number of disadvantages. First, the selection of retained samples may be biased unless random sampling is used or selection is based

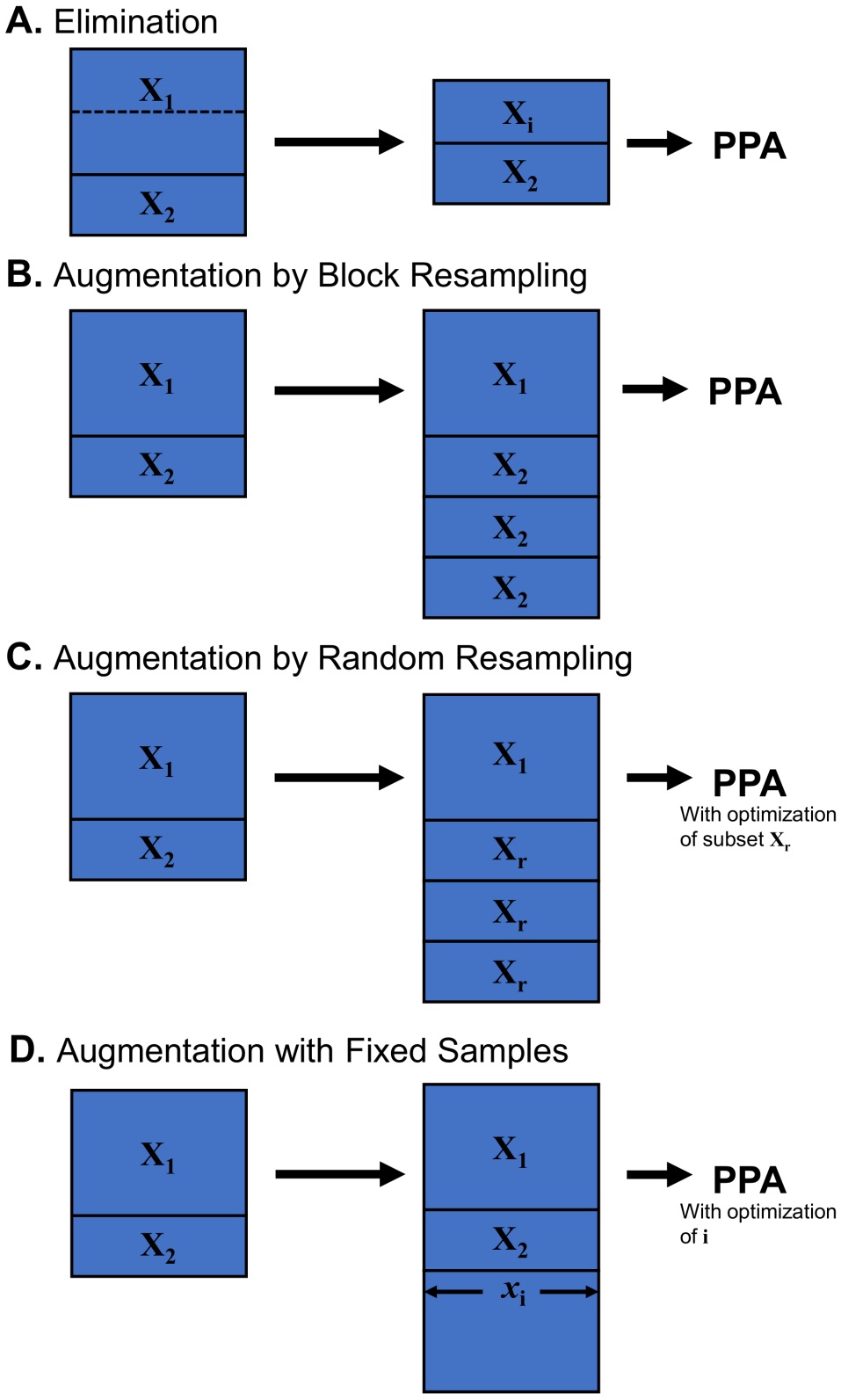

Figure 7.2: Different strategies for restoring class balance through data augmentation.

on an appropriate algorithm such as that of Kennard and Stone.[154] From a philosophical perspective, however, we are throwing away information when samples are discarded. For PPA, this has real consequences since the reduction in the total number of samples reduces the number of variables we can retain to avoid overmodelling.

A second approach, shown in Figure 7.2B, would be to increase the number of samples in the smaller class by drawing samples from that class and adding them to the data matrix. In effect, we make copies of $\mathbf{X}_2$ and add them to the bottom of $\mathbf{X}$ in sufficient number to rebalance the classes (adjustments for partial copies may be necessary, depending on the class ratio). This is essentially equivalent to the method of weighted kurtosis described in the previous section and suffers from the same drawback in that it requires prior knowledge of the classes and means that PPA is no longer an unsupervised method. Therefore, this option, while effective, was rejected.

While block resampling was rejected because it introduced prior knowledge of the classes into the algorithm, it is clear that, for resampling to work, it was at least necessary to have prior knowledge of the class ratios. Because this does not require specific assignment of the classes, however, it was not considered as diminishing the unsupervised exploratory nature of PPA.

A third approach, to avoid prior knowledge of sample class, is to build augmentation matrices randomly as shown in Figure 7.2C. In this approach, random samples are taken from $\mathbf{X}$ (without replacement) sufficient to equal the number of samples in $\mathbf{X}_2$. These form the matrix $\mathbf{X}_r$, which is used to augment $\mathbf{X}$ in the same manner as described above. Of course, it is hoped that all of these samples will be drawn from $\mathbf{X}_2$, but this is highly unlikely, so some sort of random or directed optimization of $\mathbf{X}_r$ (on top of the kurtosis minimization) is required to find the optimal subset. Various optimization strategies were employed in this work including iterative approaches and genetic algorithms, but these were all slow and prone to becoming trapped in local minima.

A less ideal but far more practical alternative to random sampling is shown in Figure 7.2D, where a single sample drawn from $\mathbf{X}$ is repeated a sufficient number of times to augment class 2. The idea here is that, while full resampling of $\mathbf{X}_2$ is statistically the most ideal, the use of a simple representative sample may be sufficient to produce a nearly equivalent result. Moreover, the optimization of the sample is computationally very fast, since it only involves testing each sample in the data set once. This was the approach

chosen for this work.

7.2.4 Augmented PPA

The algorithm for augmented PPA (augPPA) is specifically designed for two class separation problems involving unbalanced classes. Although the method could, in principle, be extended to more classes in certain circumstances, multiple non-binary class separation with unbalanced classes is more readily treated by sequential PPA, described in Section 7.2.5.

The augPPA algorithm requires the specification of the number of augmented samples to employ, p, which is based on the ratio of class sizes and the total number of samples. The augmentation of $\mathbf{X}$ with p samples is designed to give balanced class sizes. The algorithm consists of an outer loop which repeats m times to extract each row of the original $\mathbf{X}$ to be used as the augmentation vector. The extracted vector, x_i, is then copied p times to the end of $\mathbf{X}$ and univariate PPA (one dimension) is applied to the augmented matrix in the usual way. At the end of the loop, the solution which produced the minimum kurtosis is then retained. The scores and loadings are returned. The loadings are reported for data in the original space of $\mathbf{X}$, which avoids any inconsistencies between the column means of the original data and the augmented data.

In applications involving more than two classes and higher dimensions, application of augPPA is possible but with some caveats. First, additional steps would have to be carried out with augmentation of the the deflated $\mathbf{X}$. Second, because of the nature of the augmentation, the augmentation would need to represent a single class. For example, if three classes with sizes of 200, 75, and 50 were to be separated, p would have to be set to 225 (275-250) and 175 (250-75) to accommodate the separation of the three classes in two dimensions. Additionally, this method does not make full use of the space, so the maximum number of classes that could be extracted in three dimensions is four. Because of these complications, augPPA has not been extended to more than two classes at present. A better approach is to use sequential PPA (seqPPA), which uses the sample principles of augmentation in a different way, as described below.

7.2.5 Sequential Cluster Extraction

If $\mathbf{X}$ contains multiple classes, balanced or unbalanced, it is proposed here that sequential extraction of the classes should be possible by repeatedly implementing the augPPA

algorithm described above. First, one of the classes containing r_1 objects is targeted. This choice is somewhat arbitrary, but if there are multiple classes of the same size, it is not possible to target one of them unambiguously. augPPA is then applied with the length of augmentation, p, set to $p_1 = m_1 - 2r$, where m_1 is the number of rows in the original $\mathbf{X}$. If the scores plot shows successful isolation of the r_1 samples, these are removed from $\mathbf{X}$, creating $\mathbf{X}_2$ ($m_2 \times n$) and another class of objects, with size r_2 and $p_2 = m_2 - 2r_2$ is targeted. This process is repeated until all of the classes have been successively isolated.

Note that this procedure can continue only if it is successful at each step. However, if the class sizes are different, the process can be re-initiated with a different sequence of targets to try and resolve the impasse.

This process, unlike PCA or standard PPA, does not try to project all of the samples into a single space. It proceeds in a manner analogous to classification trees[155] which employ multiple steps to subdivide samples. However, the goal here is not to perform classification, but rather to confirm that the measurements contain the information necessary for classification. In this regard, it is a very powerful tool for problems involving a large number of classes.

7.3 Methods

To illustrate the proposed augPPA method, a simple two-class, two-dimensional, data set was simulated to confirm the expected behaviour of the algorithm. To extend this to higher dimensions and more classes, a data set that contained seven classes at the vertices of a 6-simplex was simulated. A simplex geometry was chosen as this geometry ensures that a projection vector exists where each class is isolable by the augPPA algorithm. This data set was used to test the seqPPA algorithm. The algorithms were then applied to two different sets of multivariate chemical data and their performance was compared to PCA and ordinary PPA.

7.3.1 Simulated Data Sets

Data set 1. A simple two-dimensional, two-class, unbalanced data set was simulated to investigate the kurtosis as a function of the projection vector angle for both the non-augmented and augmented data. The non-augmented data set contained data drawn from two bivariate normal distributions. The covariance matrix for both of these populations

was set to

$$\mathbf{\Sigma} = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix}$$

and the population means were $\mu_1 = \begin{bmatrix} -2 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mu_2 = \begin{bmatrix} 2 & 0 \end{bmatrix}^{\mathrm{T}}$. To create the unbalance, 200 samples were drawn from the first population and 50 samples were drawn from the second. These data were then rotated by $45°$ to introduce correlation between the two variables.

Data set 2. This data set was simulated by placing seven multivariate normal clusters of size 100 at the vertices of a 6-simplex centered around zero with the length of each edge being constant (Euclidean distance of 12.2). Each cluster had a covariance matrix equal to the 6×6 identity matrix.

7.3.2 Experimental Data Sets

To demonstrate the performance of augPPA and seqPPA on experimental data, two data sets were chosen.

Breast cancer: This well known data set (569×30) was obtained from the UCI Machine Learning Repository (Breast Cancer Wisconsin (Diagnostic), 1995). The data set consists of 30 features computed for cell nuclei from "digitized images of a fine needle aspirate of a breast mass". Examples of these features include the radius, perimeter, and area of the nuclei. It consists of 357 benign and 212 malignant samples.

Ink data: This data set was obtained from a study designed for the classification of blue pen by ink Fourier-transformed infrared (FTIR) spectroscopy.[151] The original data set contains 10 brands of ink, each with 6 batches, and 10 spectra per pen, which totals to 60 spectra per brand. This results in a 600×3351 data matrix. This is the same data set analysed in Chapter 5, but with all 10 brands of pen ink included.

7.3.3 Computational Details

All calculations were performed in MATLAB R2018a (MathWorks, Natick, MA)[81] using scripts written in-house. Code for implementing the proposed method can be found in Appendix D. As standard practice, column mean centering was performed before applying PCA, PPA, augPPA, and seqPPA.

7.4 Results

7.4.1 Simulated Data Sets

To explore the simplest case of the optimization space explored by PPA with augmented data, the kurtosis of the projected scores was calculated as a function of the angle of the projection vector to the horizontal line at $y = 0$. Figure 7.3A shows the simulated two-dimensional data. Qualitatively, an ideal projection vector would be $\sim 135°$.

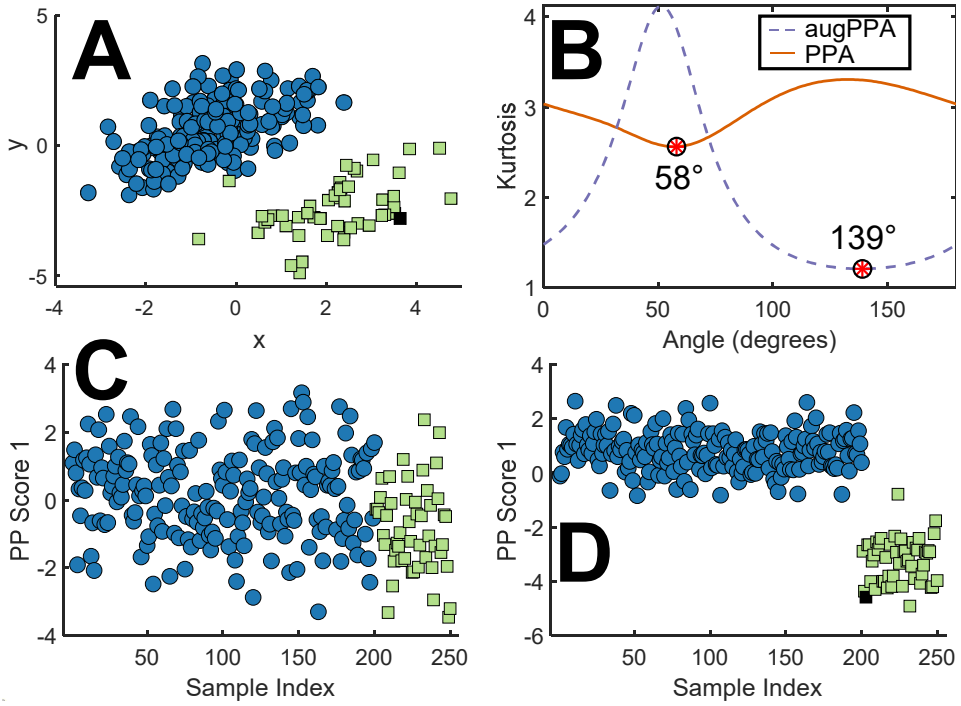

Figure 7.3: **A** Simulated two-dimensional, two-class, unbalanced data set with 200 samples in class 1 (blue circle markers) and 50 in class 2 (green square markers). **B** Kurtosis as a function of the angle of the projection vector for both PPA and augPPA (sample with black square marker in **A** indicates the sample used for augmentation of size 150). **C** Scores obtained when the data in **A** are projected onto the vector with an angle of 58°. **D** Scores obtained when the data in **A** are projected onto a vector with an angle of 139°of the data in **A** augmented with 150 samples of the sample with the black square marker.

As shown in Figure 7.3B, the projection vector with the minimum kurtosis of the non-augmented data (which is equivalent to the PPA algorithm) is at 58°. The scores of the samples when they are projected onto this vector are shown in Figure 7.3C. As expected, it is clear that there is no separation of the classes in this space. Conversely, Figure 7.3B shows that for the data augmented with 150 samples, the minimum kurtosis value is found

at a projection vector angle of 139°. The black square marker shows the sample that was determined to be optimal for the augmentation, which was identified using the simple search algorithm proposed in Section 7.2.3. The scores projected onto this vector are shown in Figure 7.3D, which show a clear separation of the two classes even though one is unbalanced. This confirms that augPPA is able to provide superior results to PPA in a simple two-dimensional, two-class, unbalanced data set.

Data set 2 contains 7 clusters on the vertices of a 6-simplex which was simulated to confirm the expected behaviour of augPPA in the sequential extraction of clusters from high-dimensional data (seqPPA). Figure 7.4(A-F) shows the scores plots for each step in the sequential extraction using augPPA. In each step, augPPA was able to find a projection that isolates one cluster from the remaining clusters. Of course, for real data, the quality of the results would be a function of the cluster positions relative to each other in the multidimensional space. Data set 2 was simulated as an ideal case, where each cluster is clearly defined with no overlap between the classes. However, these results suggest, that under ideal circumstances, seqPPA is able to sequentially extract multiple clusters and is not restricted to data sets with binary class sizes.

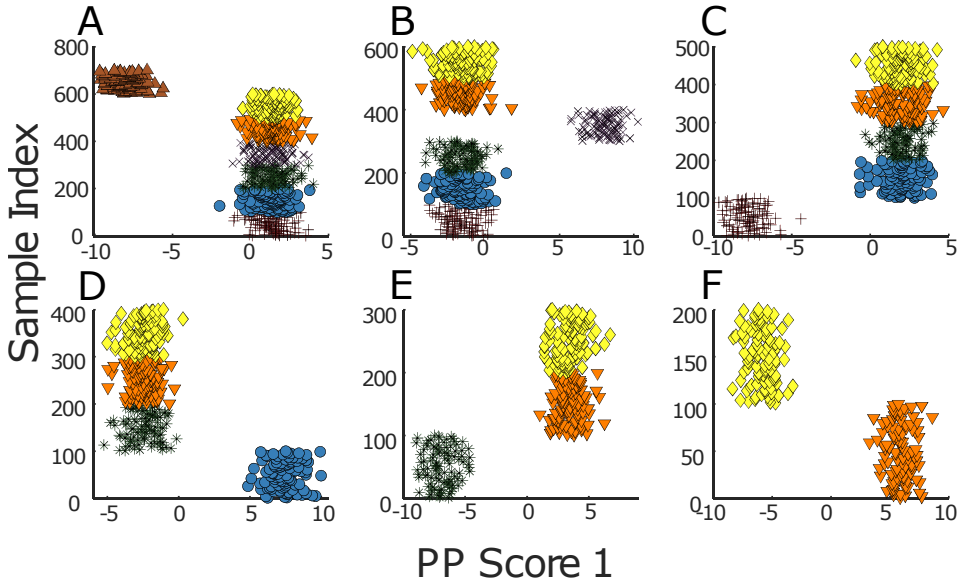

Figure 7.4: Scores plots obtained from applying seqPPA on a simulated 7 class data set.

7.4.2 Experimental Data Sets

The breast cancer data set was employed to demonstrate a case where PCA does a satisfactory job at separating binary classes while ordinary PPA fails to do so. Figure 7.5 shows

the first scores obtained with PCA, PPA, and augPPA for the breast cancer data set. Since the data set is unbalanced, augPPA was applied using an augmentation size of $p = 145$ with sample 237 (malignant) determined as the optimal augmentation location using the simple search algorithm described in Section 7.2.3. The PCA results (Figure 7.5A) show a tight clustering of the benign class relative to the malignant class and some overlap between the two classes. These results support the hypothesis that these measurements can provide discriminatory information between the benign and malignant classes. In contrast, the scores obtained with PPA (Figure 7.5B) show both classes heavily overlapped with no clear discrimination between the two. The augPPA results (Figure 7.5C) are similar to the PCA results, showing clustering of the the two classes with a slight overlap between them. These results suggest that the reason PPA is unable to find this projection is due to the imbalance in the class sizes. Employing the augmentation strategy balances the classes out and obtains a lower kurtosis value for the projection of interest.

The Ink Data Set was chosen to determine if the sequential extraction of clusters using seqPPA proposed in Section 7.2.5 and demonstrated on simulated data in Section 7.4.1 could be possible on a real data set. Since the ink data set contains 10 classes, each of size 60, augPPA was applied 8 times with the appropriate augmentation. Since this data set contains 3351 response variables, PCA compression was performed such that a sample to variable ratio of 10:1 was achieved before applying augPPA to the PCA scores. This was performed for each sequential extraction with the augmentation parameters $p = m - 120$, where m is the number of samples for the data matrix in the current step, and the augmentation vector which was chosen by the simple search algorithm presented in section 7.2.3. To make the procedure fully automated, the samples extracted in each step were removed by determining the 60 samples furthest away from the median of the data. Figure 7.6A–I shows the scores obtained for each step of this process.

Overall, the seqPPA method was able to extract one cluster of the ten total clusters in each step. As expected with real data, the quality of the separation in each step varies. For example, Figure 7.6F and I show an example of the extracted class being more dispersed in the score space compared to classes extracted in other steps. For the step that extracts Brand 5 (Figure 7.6F), for example, this could be due to a number of reasons. Since the proposed method uses a very simple procedure for determining the optimal augmentation vector, it may be that, for clusters that have irregular or multivariate non-normal distributions, the

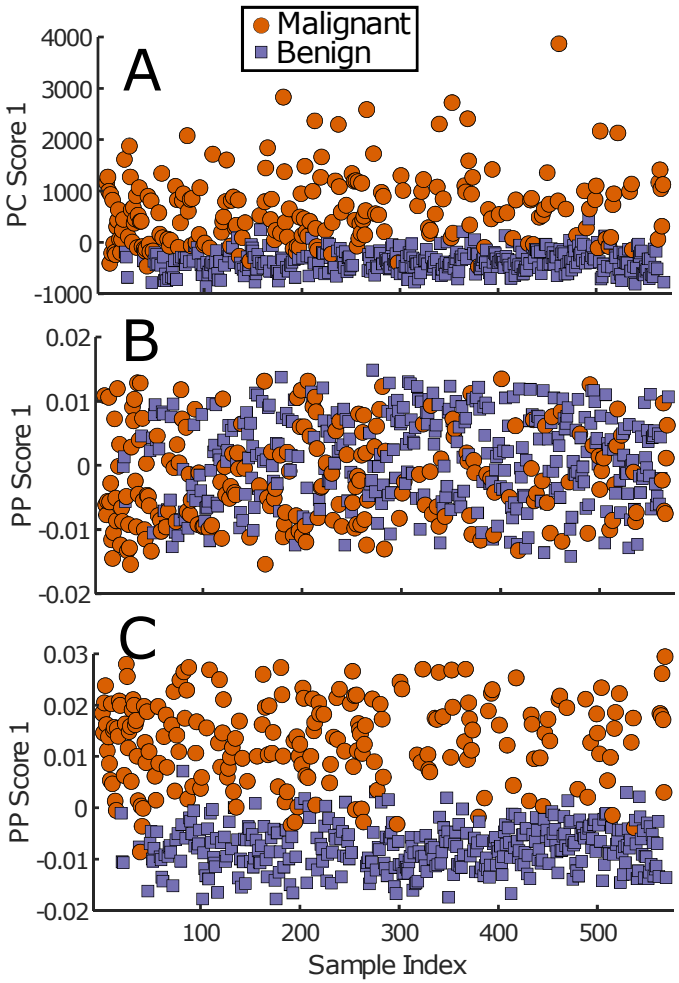

Figure 7.5: Scores plot for the breast cancer data set produced by (A) PCA, (B) PPA, and (C) augPPA.

procedure is not accurate enough in determining the augmentation vector to fully extract the class. However, these are minor deviations from the behaviour observed with the simulated data set, which was designed to be the ideal case.

7.5 Conclusions

A major drawback of stepwise univariate kurtosis-based projection pursuit is that it fails to identify clusters in unbalanced data sets. Using a simple augmentation strategy, it was shown that it is possible to identify samples in underrepresented classes and reveal clusters in unbalanced data sets by rebalancing the data through augmentation. Furthermore, we show that sequential class separation can be achieved by applying this method iteratively,

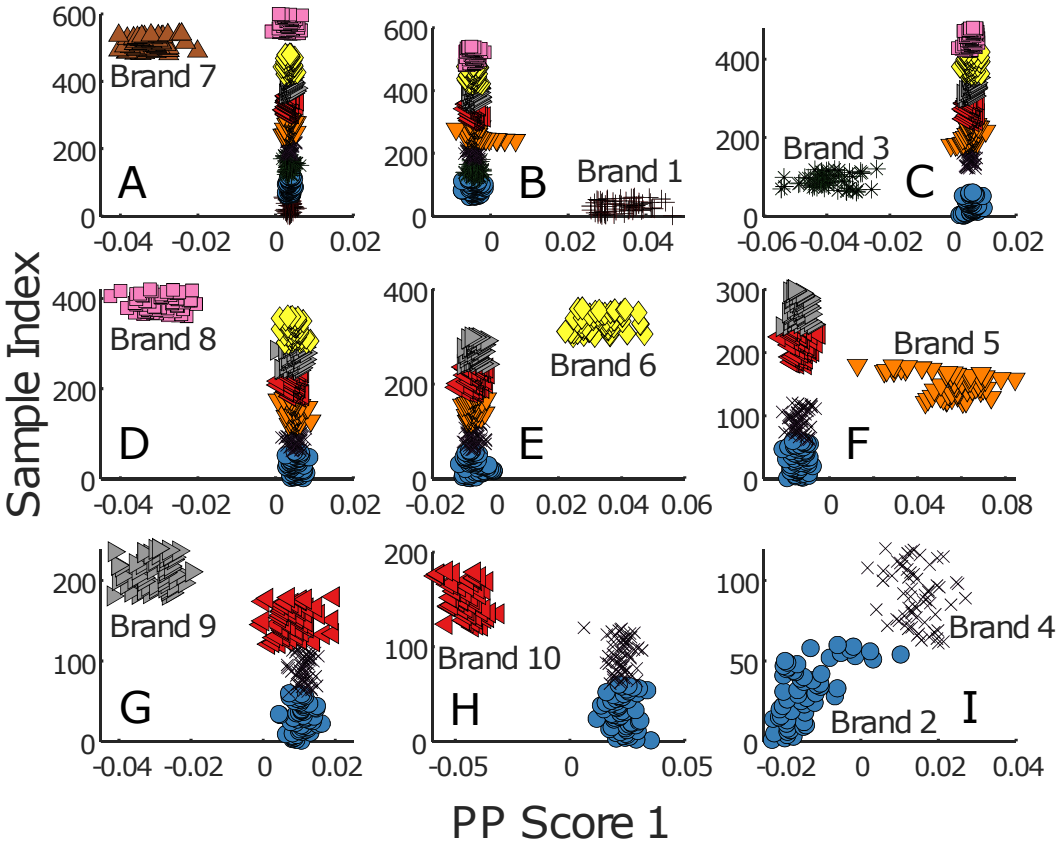

Figure 7.6: 10 clusters sequentially extracted from the ink data set using seqPPA.

which is a good starting point for a kurtosis-based projection pursuit algorithm that is not confined to the traditional 2^n separation geometry in n-dimensions.

CHAPTER 8

CONCLUSION

A central goal in the application of chemometrics to multivariate chemical data is to separate the relevant chemical variance (changes in the signal that are related to chemical changes of interest) from measurement noise (variance associated with analytical sources of error) and chemical noise (chemical variance related to chemical changes not of interest). The relevant chemical variance can then be correlated with the chemical information of interest, such as class membership (classification), concentration (multivariate calibration), or a model of system behaviour (multivariate curve resolution). The work presented here has attempted to address various aspects related to this core problem. This section presents the main conclusions from this thesis and provides some direction for future work.

The ability to isolate chemical variance from measurement noise variance is a key aspect of any chemometric tool. It is now widely accepted that this capability is closely linked to the structure of measurement noise in a given system and the assessment of data analysis tools and/or preprocessing methods for a given application requires consideration of relevant error structures. Therefore, the first part of this thesis focuses on methods to simulate various types of noise structures and demonstrating the utility of this strategy in assessing data analysis methods.

Despite the importance of measurement noise, there are many instances where its characteristics have not been elucidated, limiting applications of methods that require the availability of such information to be effective (*e.g.* MLPCA). Likewise, such methods are ineffective when chemical noise limits the extraction of chemical information. PPA is a powerful alternative to variance-based methods for exploratory analysis and data visualization, but is subject to certain practical limitations. The second part of this thesis

addresses some of these limitations with a goal of extending the capabilities of PPA.

The results of Chapter 2, titled Simulation of $1/f^{\alpha}$ Noise for Analytical Measurements, and Chapter 3, titled NoiseGen - Analytical Measurement Error Simulation Software, provide an approachable methodology to perform realistic noise simulation for evaluating data processing algorithms. Prior to the publication of the work described in these chapters, much of the reported noise simulation in the analytical chemistry literature was comprised only of single- component noise simulation. Together, these two chapters provide a workflow for the simulation of any type and combination of noise by rotation and scaling of white noise via the ECM of the target noise. The advantage of formulating the proposed simulation methodology around the ECM is twofold. First, it allows the simulation of any type of noise as long as an expression for the ECM is known. For example, previous simulation strategies for generating $1/f^{\alpha}$ noise were not trivial; formulating the simulation from the standpoint of the ECM proved beneficial in both interpretation of the noise structure and its simulation. Second, the simulation of combinations of different noise types is simple by assuming an additive noise model of the ECM. In attempt to make this method of noise simulation even more approachable, the NoiseGen package was developed for the simulation of common analytical noise types. Although the NoiseGen algorithm is fast, it was not a goal of our investigations to optimize its speed. This leaves room for further optimization that could almost certainly lead to faster generation of noise structures (for example, NoiseGen is currently implemented with no parallelization). However, this is a minor point as, on modern hardware, the software can simulate realistic noise structures for use in algorithm evaluation in time frames on the order of fractions of a second to seconds.

The impact of the realistic noise simulation work in this thesis should be observed in years to come as researchers continue to develop and evaluate more advanced analysis algorithms. It is important to note that, while these two chapters were structured around noise observed in chemical analysis platforms, these types of noise are observed in almost all measurements regardless of discipline. As such, the application of these tools are not limited to just chemical data, but to data from all disciplines concerned with measurement error.

Chapter 4, titled Data Fusion with Noisy Measurements, provides a theoretical framework for mitigating the impact errors have on chemical subspace estimation for fused data

using a maximum likelihood approach. The benefits of the proposed method were shown using the simulation of fused data with different error structures. The proposed method was then applied to a real fused data set concerned with the clustering of various olive oils analyzed by UV-Vis, NIR, and MIR spectroscopy. Because this data set contained many replicates, an accurate estimate of the fused ECM could be calculated. The results of applying MLPCA to the fused data with the fused ECM proved to greatly increase the quality of the projection compared to PCA and PCA applied to the autoscaled data. However, this method requires a conscious decision from the researcher designing the experiment to make sure the design is able to capture the structure of the ECM for each instrument used in the fusion. Although MLPCA seems like the best alternative to PCA for differentiating between chemical variance and noise variance, the requirement of an accurate representation of the ECM makes it difficult to apply to data sets that do not contain this information, and, unfortunately, the majority of published data is not designed for this purpose.

As mentioned in the introduction of this thesis, another strategy for differentiating between noise variance and chemical variance is projection pursuit analysis. Chapters 5, 6, and 7 presented modifications of a kurtosis-based projection pursuit algorithm, which provides an alternative approach to accessing useful chemical subspaces in high-dimensional chemical data. The goal of projection pursuit analysis (PPA) is to find "interesting" projections based on a projection index, which, in this case, is the kurtosis. PPA has been shown to isolate relevant chemical variance from variance due to measurement and chemical noise in data by searching for projections with low kurtosis. This has practical implications in exploratory data analysis, as PPA is not prone to the scaling issues of PCA and does not require information of the error covariance matrix to correct for these issues (the result of applying maximum likelihood principal component analysis). However, despite the initial success of PPA, there are two issues with the algorithm that prevent it from having the same utility of traditional exploratory methods. First, PPA requires "skinny data", or data with more samples than variables. With chemical data, this condition is usually met by compressing the data with PCA to k components such that an acceptable sample-to-variable ratio is achieved (previous applications of PPA have determined that at least a $\sim 5:1$ sample-to-variable ratio is required to avoid overfitting). The problem with this methodology is that useful information may be lost at this initial PCA compression

step, which is also sensitive to scaling. The second issue relates to the performance of ordinary PPA on unbalanced and non-binary data sets. As illustrated in Chapter 7, ordinary PPA, performs poorly on unbalanced data due to the behaviour of the univariate kurtosis value with unbalanced distributions. The results in this thesis have provided solutions to both of these problems.

Chapter 5, titled Sparse Projection Pursuit Analysis- An Alternative for Exploring Chemical Data, presents a sparse implementation of kurtosis-based PPA that circumvents the need for PCA compression. This work is a major advancement in PPA for multivariate data as it does not rely on PCA and, further, provides chemical interpretation of the observed projection through repeated variable selection. In ordinary PPA, the interpretation of the "loadings" can be difficult if not impossible. SPPA facilitates direct interpretation of the variables which leads to an improved understanding of important chemical information in the system. In this work, SPPA was found to out-perform PCA in all cases when applied to a diverse collection of data sets, supporting the notion that SPPA offers a competitive alternative to PCA in the exploratory analysis of multivariate chemical data.

Chapter 6 presented a modification to the genetic variable selection SPPA algorithm, termed banded SPPA, which constrains the variables selected by SPPA to bands of neighboring variables. This was shown to provide improved selected variable interpretation without the need for repeated applications of SPPA to determine the important variables responsible for an informative projection. As shown in the application of repeated runs of banded SPPA to the Wine Grape Vis-NIR Data Set, where two dominant bands were found in the first dimension and three dominant bands were found in the second dimension, there may be a need for allowing a different number and size of bands in each dimension. This would ultimately give the algorithm more flexibility and may provide even more stability in the selected variables. This chapter represents one way of implementing a banded SPPA method and it is possible that other implementations could perform better. For example, another possible strategy could be to calculate the average response in each band of variables in the genetic selection algorithm. This would reduce the number of variables being used to perform PPA as one band of variables would be represented as one variable (the average response in the band). In terms of performance, this strategy would likely perform better than the current banded SPPA strategy in situations with a small number of samples.

Chapter 7, titled Augmented Projection Pursuit Analysis, describes a data augmentation strategy that can be used to reveal useful projections in unbalanced and non-binary data sets. As illustrated in this work, ordinary univariate PPA fails to provide useful projections in unbalanced data sets due to the behaviour of univariate kurtosis. Through simulation it was discovered that, in the case of unbalanced data, the projection that discriminates unbalanced classes does not necessarily have the minimum univariate kurtosis. To remedy this, augmentation of the unbalanced class with the appropriate number of "dummy" samples prior to applying PPA was proposed. To keep the method unsupervised, the augPPA algorithm builds trial solutions of the augmented data using only the samples in the original data set and prior knowledge of the class size ratio. Ordinary PPA is applied to the original data augmented with each trial augmentation matrix. The sample that gives the lowest kurtosis value when used for the augmentation matrix is used for the final solution. This strategy proved useful in revealing clusters in both simulated and real unbalanced data sets. A secondary finding of this work was that, by redefining the size of the augmentation matrix, it is possible to sequentially extract clusters from both simulated and real data sets. As a demonstration of the algorithm's utility, the method was applied to sequentially extract 10 clusters from the ink FTIR data set, a truly remarkable result for an unsupervised method.

In general, the kurtosis-based PPA variants developed as a part of this thesis have been shown to provide an alternative way of searching multivariate chemical data for interesting subspaces. The application of these methods to both simulated and real multivariate data sets proved to reveal informative subspaces that are unobtainable by traditional methods such as PCA. However, the PPA methods proposed in this thesis are all still kurtosis-based methods and only represent a few ways of performing unsupervised exploration on multivariate data. Therefore, there is room for more exploratory methods to further diversify the number of analysis tools available to researchers. An obvious place to begin, in the realm of projection pursuit, is to explore different projection indices that may have desirable properties for exploring data. For example, another possible approach to solving the unbalanced data issue with ordinary PPA is to use a skewness-invariant measure of the kurtosis as the projection index.[156] In terms of revealing clusters in general, there are a couple of well defined metrics that could be used, such as the bimodality coefficient[157] and Ashman's D statistic.[158] However, optimization of these metrics as a function of the

projection vector would be a major component of the research involved with the practical implementation of these methods.

The differentiation between informative chemical variance and all other variance is the quintessential challenge in analyzing high-dimensional chemical data. The early work in this thesis lays the foundation for realistic noise simulation for evaluating and developing algorithms that attempt to tackle this challenge. The final sections of this thesis contain solutions to the issues preventing one of the most promising alternative exploratory methods, kurtosis-based projection pursuit analysis, from being applied alongside PCA and HCA in the analytical chemistry literature.

BIBLIOGRAPHY

[1] Booksh, K. S.; Kowalski, B. R. Theory of analytical chemistry. *Anal. Chem.* **1994**, *66*, 782A–791A.

[2] Brereton, R. G.; Jansen, J.; Lopes, J.; Marini, F.; Pomerantsev, A.; Rodionova, O.; Roger, J. M.; Walczak, B.; Tauler, R. Chemometrics in analytical chemistry—part I: history, experimental design and data analysis tools. *Anal. Bioanal. Chem.* **2017**, *409*, 5891–5899.

[3] Wold, S. Chemometrics; what do we mean with it, and what do we want from it? *Chemom. Intell. Lab. Syst.* **1995**, *30*, 109 – 115.

[4] Bro, R.; Smilde, A. K. Principal component analysis. *Anal. Methods* **2014**, *6*, 2812–2831.

[5] Lavine, B. K.; Mirjankar, N. Clustering and classification of analytical data. In *Encyclopedia of Analytical Chemistry*; Meyers, R. A., Brown, S. D., Eds.; Wiley: New Jersey, United States, 2012.

[6] Wold, S.; Sjöström, M.; Eriksson, L. PLS-regression: a basic tool of chemometrics. *Chemom. Intell. Lab. Syst.* **2001**, *58*, 109 – 130.

[7] Lamichhane, S.; Sen, P.; Dickens, A. M.; Hyötyläinen, T.; Orešič, M. An overview of metabolomics data analysis: current tools and future perspectives. In *Comprehensive Analytical Chemistry*; Jaumot, J., Bedia, C., Tauler, R., Eds.; Elsevier: Amsterdam, Netherlands, 2018; Chapter 14, pp 387–413.

[8] Lasch, P.; Stämmler, M.; Zhang, M.; Baranska, M.; Bosch, A.; Majzner, K. FT-IR hyperspectral imaging and artificial neural network analysis for identification of pathogenic bacteria. *Anal. Chem.* **2018**, *90*, 8896–8904.

[9] Prats-Montalbán, J.; de Juan, A.; Ferrer, A. Multivariate image analysis: A review with applications. *Chemom. Intell. Lab. Syst.* **2011**, *107*, 1 – 23.

[10] Gredilla, A.; de Vallejuelo, S. F.-O.; Elejoste, N.; de Diego, A.; Madariaga, J. M. Non-destructive Spectroscopy combined with chemometrics as a tool for Green Chemical Analysis of environmental samples: A review. *Trends Anal. Chem.* **2016**, *76*, 30 – 39.

[11] Risum, A. B.; Bro, R. Using deep learning to evaluate peaks in chromatographic data. *Talanta* **2019**, *204*, 255 – 260.

[12] Luts, J.; Ojeda, F.; Van de Plas, R.; De Moor, B.; Van Huffel, S.; Suykens, J. A. A tutorial on support vector machine-based methods for classification problems in chemometrics. *Anal. Chim. Acta* **2010**, *665*, 129–145.

[13] Skoog, D.; West, D.; Holler, F.; Crouch, S. *Fundamentals of Analytical Chemistry*; Brooks Cole: United States, 2013.

[14] Skoog, D.; Holler, F.; Crouch, S. *Principles of Instrumental Analysis*; Brooks Cole: United States, 2006.

[15] Hirschfeld, T. The hy-phen-ated methods. *Anal. Chem.* **1980**, *52*, 297A–312A.

[16] Meher, A. K.; Chen, Y.-C. Combination of Raman spectroscopy and mass spectrometry for online chemical analysis. *Anal. Chem.* **2016**, *88*, 9151–9157.

[17] Cumeras, R.; Figueras, E.; Davis, C. E.; Baumbach, J. I.; Gràcia, I. Review on Ion Mobility Spectrometry. Part 2: hyphenated methods and effects of experimental parameters. *Analyst* **2015**, *140*, 1391–1410.

[18] Räntzsch, V.; Wilhelm, M.; Guthausen, G. Hyphenated low-field NMR techniques: combining NMR with NIR, GPC/SEC and rheometry. *Magn. Reson. Chem.* **2015**, *54*, 494–501.

[19] Saccenti, E.; Hoefsloot, H. C.; Smilde, A. K.; Westerhuis, J. A.; Hendriks, M. M. Reflections on univariate and multivariate analysis of metabolomics data. *Metabolomics* **2014**, *10*, 361–374.

[20] Johnson, R. A.; Wichern, D. W. *Applied multivariate statistical analysis*; Prentice-Hall Inc: New Jersey, United States, 1992.

[21] Tukey, J. W. We need both exploratory and confirmatory. *Am. Stat.* **1980**, *34*, 23–25.

[22] Sudarikov, K.; Tyakht, A.; Alexeev, D. Methods for the metagenomic data visualization and analysis. *Curr. Issues. Mol. Biol.* **2017**, *24*, 37–58.

[23] Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Phil. Mag.* **1901**, *2*, 559–572.

[24] Hotelling, H. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **1933**, *24*, 417.

[25] Bishop, C. M. *Pattern recognition and machine learning*; Springer: Berlin, Germany, 2006.

[26] Riedl, J.; Esslinger, S.; Fauhl-Hassek, C. Review of validation and reporting of non-targeted fingerprinting approaches for food authentication. *Anal. Chim. Acta* **2015**, *885*, 17–32.

[27] Bro, R.; Kjeldahl, K.; Smilde, A.; Kiers, H. Cross-validation of component models: a critical look at current methods. *Anal. Bioanal. Chem.* **2008**, *390*, 1241–1251.

[28] Shi, L.; Westerhuis, J. A.; Rosén, J.; Landberg, R.; Brunius, C. Variable selection and validation in multivariate modelling. *Bioinformatics* **2018**, *35*, 972–980.

[29] Szymańska, E.; Saccenti, E.; Smilde, A. K.; Westerhuis, J. A. Double-check: validation of diagnostic statistics for PLS-DA models in metabolomics studies. *Metabolomics* **2012**, *8*, 3–16.

[30] Westerhuis, J. A.; Hoefsloot, H. C.; Smit, S.; Vis, D. J.; Smilde, A. K.; van Velzen, E. J.; van Duijnhoven, J. P.; van Dorsten, F. A. Assessment of PLSDA cross validation. *Metabolomics* **2008**, *4*, 81–89.

[31] Gromski, P. S.; Muhamadali, H.; Ellis, D. I.; Xu, Y.; Correa, E.; Turner, M. L.; Goodacre, R. A tutorial review: Metabolomics and partial least squares-discriminant analysis–a marriage of convenience or a shotgun wedding. *Anal. Chim. Acta* **2015**, *879*, 10–23.

[32] Placidi, G.; Alecci, M.; Sotgiu, A. Post-processing noise removal algorithm for magnetic resonance imaging based on edge detection and wavelet analysis. *Phys. Med. Biol.* **2003**, *48*, 1987.

[33] Jaumot, J.; Gargallo, R.; Tauler, R. Noise propagation and error estimations in multivariate curve resolution alternating least squares using resampling methods. *J. Chemom.* **2004**, *18*, 327–340.

[34] Ahmadvand, M.; Parastar, H.; Sereshti, H.; Olivieri, A.; Tauler, R. A systematic study on the effect of noise and shift on multivariate figures of merit of second-order calibration algorithms. *Anal. Chim. Acta* **2017**, *952*, 18–31.

[35] Stoica, P.; Moses, R. *Spectral analysis of signals*; Prentice Hall: New Jersey, United States, 2005.

[36] Wentzell, P. D. Measurement errors in multivariate chemical data. *J. Braz. Chem. Soc.* **2014**, *25*, 183–196.

[37] Cooley, J. W.; Tukey, J. W. An algorithm for the machine calculation of complex fourier series. *Math. Comput.* **1965**, *19*, 297–301.

[38] B. Johnson, J. The Schottky effect in low frequency circuits. *Phys. Rev.* **1925**, *26*, 71–85.

[39] Schottky, W. SmallShot effect and flicker effect. *Phys. Rev.* **1926**, *28*, 1331–1331.

[40] Hou, S.; Wentzell, P. Fast and simple methods for the optimization of kurtosis used as a projection pursuit index. *Anal. Chim. Acta* **2011**, *704*, 1–15.

[41] Wentzell, P. D.; Andrews, D. T.; Hamilton, D. C.; Faber, K.; Kowalski, B. R. Maximum likelihood principal component analysis. *J. Chemom.* **1997**, *11*, 339–366.

[42] Wentzell, P. D. Other topics in soft-modeling: maximum likelihood-based soft-modeling methods. In *Comprehensive chemometrics: chemical and biochemical data analysis;* Brown, S. D., Tauler R., Walczak, B.; Eds.; Elsevier: Amsterdam, Netherlands, 2009; pp 507-558.

[43] Wentzell, P. D.; Cleary, C. S.; Kompany-Zareh, M. Improved modeling of multivariate measurement errors based on the Wishart distribution. *Anal. Chim. Acta* **2017**, *959*, 1–14.

[44] Leger, M. N.; Vega-Montoto, L.; Wentzell, P. D. Methods for systematic investigation of measurement error covariance matrices. *Chemom. Intell. Lab. Syst.* **2005**, *77*, 181–205.

[45] Wentzell, P. D.; Andrews, D. T.; Kowalski, B. R. Maximum likelihood multivariate calibration. *Anal. Chem.* **1997**, *69*, 2299–2311.

[46] Schreyer, S. K.; Bidinosti, M.; Wentzell, P. D. Application of maximum likelihood principal components regression to fluorescence emission spectra. *Appl. Spectrosc.* **2002**, *56*, 789–796.

[47] Tauler, R.; Viana, M.; Querol, X.; Alastuey, A.; Flight, R.; Wentzell, P.; Hopke, P. Comparison of the results obtained by four receptor modelling methods in aerosol source apportionment studies. *Atmospheric Environ.* **2009**, *43*, 3989–3997.

[48] Wentzell, P. D.; Karakach, T. K.; Roy, S.; Martinez, M. J.; Allen, C. P.; Werner-Washburne, M. Multivariate curve resolution of time course microarray data. *BMC Bioinform.* **2006**, *7*, 343.

[49] Friedman, J. H.; Tukey, J. W. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.* **1974**, *100*, 881–890.

[50] Ifarraguerri, A.; Chang, C.-I. Unsupervised hyperspectral image analysis with projection pursuit. *IEEE Trans. Geosci. Remote Sens.* **2000**, *38*, 2529–2538.

[51] Posse, C. Projection pursuit exploratory data analysis. *Comput. Stat. Data Anal.* **1995**, *20*, 669 – 687.

[52] Huber, P. J. Projection Pursuit. *Ann. Stat.* **1985**, *13*, 435–475.

[53] Akritas, M. G. Projection pursuit multi-index (PPMI) models. *Stat. Probabil. Lett.* **2016**, *114*, 99–103.

[54] Croux, C.; Filzmoser, P.; Oliveira, M. R. Algorithms for projection–pursuit robust principal component analysis. *Chemom. Intell. Lab. Syst.* **2007**, *87*, 218–225.

[55] Hou, S.; Wentzell, P. D. Re-centered kurtosis as a projection pursuit index for multivariate data analysis. *J. Chemom.* **2014**, *28*, 370–384.

[56] Hou, S.; Wentzell, P. D. Regularized projection pursuit for data with a small sample-to-variable ratio. *Metabolomics* **2014**, *10*, 589–606.

[57] Deming, S.; Michotte, Y.; Massart, D. L.; Kaufman, L.; Vandeginste, B. *Chemometrics: a Textbook*; Elsevier: Amsterdam, The Netherlands, 1988; Vol. 2.

[58] De Klerck, K.; Vander Heyden, Y.; Mangelings, D. Exploratory data analysis as a tool for similarity assessment and clustering of chiral polysaccharide-based systems used to separate pharmaceuticals in supercritical fluid chromatography. *J. Chromatogr. A* **2014**, *1326*, 110–124.

[59] Pereira, J. F. Q.; Silva, C. S.; Braz, A.; Pimentel, M. F.; Honorato, R. S.; Pasquini, C.; Wentzell, P. D. Projection pursuit and PCA associated with near and middle infrared hyperspectral images to investigate forensic cases of fraudulent documents. *Microchem. J.* **2017**, *130*, 412–419.

[60] Martinez-Farina, C. F.; Driscoll, S.; Wicks, C.; Burton, I.; Wentzell, P. D.; Berrue, F. Chemical barcoding: a nuclear-magnetic-resonance-based approach to ensure the quality and safety of natural ingredients. *J. Agric. Food Chem.* **2019**, *67*, 7765–7774.

[61] Wentzell, P. D.; Wicks, C. C.; Braga, J. W.; Soares, L. F.; Pastore, T. C.; Coradin, V. T.; Davrieux, F. Implications of measurement error structure on the visualization of multivariate chemical data: hazards and alternatives. *Can. J. Chem.* **2018**, *96*, 738–748.

[62] Ingle, J.; Crouch, S. *Spectrochemical Analysis*; Prentice Hall: New Jersey, United States, 1988.

[63] Loock, H. P.; Wentzell, P. D. Detection limits of chemical sensors: Applications and misapplications. *Sens. Actuator B-Chem.* **2012**, *173*, 157–163.

[64] Wentzell, P. D.; Tarasuk, A. C. Characterization of heteroscedastic measurement noise in the absence of replicates. *Anal. Chim. Acta* **2014**, *847*, 16–28.

[65] Jaumot, J.; de Juan, A.; Tauler, R. MCR-ALS GUI 2.0: new features and applications. *Chemom. Intell. Lab. Syst.* **2015**, *140*, 1–12.

[66] Bro, R. Multivariate calibration: What is in chemometrics for the analytical chemist? *Anal. Chim. Acta* **2003**, *500*, 185–194.

[67] Allegrini, F.; Olivieri, A. C. Recent advances in analytical figures of merit: heteroscedasticity strikes back. *Anal. Methods* **2017**, *9*, 739–743.

[68] Weissman, M. $1/f$ noise and other slow, nonexponential kinetics in condensed matter. *Rev. Mod. Phys.* **1988**, *60*, 537.

[69] Kawaguchi, T.; Mineshige, S.; Machida, M.; Matsumoto, R.; Shibata, K. Temporal $1/f^{\alpha}$ fluctuations from fractal magnetic fields in black-hole accretion flow. *Publ. Astron. Soc. Jpn.* **2000**, *52*, L1–L4.

[70] Lipsitz, L. A.; Goldberger, A. L. Loss of "complexity" and aging. Potential applications of fractals and chaos theory to senescence. *JAMA* **1992**, *267*, 1806–1809.

[71] Lipsitz, L. A. Dynamics of stability: the physiologic basis of functional health and frailty. *J. Gerontol. A Biol. Sci. Med. Sci.* **2002**, *57*, B115–B125.

[72] Musha, T.; Higuchi, H. The $1/f$ fluctuation of a traffic current on an expressway. *Jpn. J. Appl. Phys.* **1976**, *15*, 1271.

[73] Stephany, J. F. Frequency limits of $1/f$ noise. *J. Phys. Condens. Matter* **2000**, *12*, 2469.

[74] Ince, A. T.; Williams, J. G.; Gray, A. L. Noise in inductively coupled plasma mass spectrometry: some preliminary measurements. *J. Anal. At. Spectrom.* **1993**, *8*, 899–903.

[75] Hayashi, Y.; Matsuda, R. Deductive prediction of measurement precision from signal and noise in liquid chromatography. *Anal. Chem.* **1994**, *66*, 2874–2881.

[76] Hayashi, Y.; Matsuda, R. Deductive prediction of measurement precision from signal and noise in fluorometry. *Anal. Sci.* **1995**, *11*, 929–934.

[77] Voigtman, E. Effect of source $1/f$ noise on optical polarimeter performance. *Anal. Chem.* **1992**, *64*, 2590–2595.

[78] Mittermayr, C.; Lendl, B.; Rosenberg, E.; Grasserbauer, M. The application of the wavelet power spectrum to detect and estimate $1/f$ noise in the presence of analytical signals. *Anal. Chim. Acta* **1999**, *388*, 303 – 313.

[79] Shumway, R.; Stoffer, D. *Time Series Analysis and Its Applications: With R Examples*; Springer: New York, United States, 2006.

[80] Kasdin, N. J. Discrete simulation of colored noise and stochastic processes and $1/f^{\alpha}$ power law noise generation. *Proc. IEEE.* **1995**, *83*, 802–827.

[81] MATLAB, *Release 2018a*; The MathWorks Inc.: Natick, Massachusetts, United Sates, 2018.

[82] Ashby, N. Probability distributions and confidence intervals for simulated power law noise. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2015**, *62*, 116–128.

[83] Savitzky, A.; Golay, M. J. Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.* **1964**, *36*, 1627–1639.

[84] Savitzky, A. A historic collaboration. *Anal. Chem.* **1989**, *61*, 921A–923A.

[85] Blackman, R. B.; Tukey, J. W. *The measurement of power spectra, from the point of view of communications engineering*; Dover Publications Inc.: United States, 1959.

[86] Bjorck, A.; Golub, G. H. Numerical methods for computing angles between linear subspaces. *Math. Comput.* **1973**, *27*, 579–594.

[87] Wedin, P. Å. On angles between subspaces of a finite dimensional inner product space. In *Matrix Pencils;* Kagstrom B., Ruhe A.; Eds.; Springer: Berlin, Heidelberg, 1983; pp 263–285.

[88] Winning, H.; Larsen, F. H.; Bro, R.; Engelsen, S. B. Quantitative analysis of NMR spectra with chemometrics. *J. Magn. Reson.* **2008**, *190*, 26–32.

[89] Eilers, P. H. A perfect smoother. *Anal. Chem.* **2003**, *75*, 3631–3636.

[90] Hamming, R. W. *Digital filters*; Courier Corporation: New York, United States, 1998.

[91] Wentzell, P. D.; Doherty, T. P.; Crouch, S. R. Frequency response of initial point least squares polynomial filters. *Anal. Chem.* **1987**, *59*, 367–371.

[92] Wentzell, P. D.; Brown, C. D. Signal processing in analytical chemistry. In *Encyclopedia of Analytical Chemistry;* Meyers, R.A., Eds.; Wiley: Chichester, United Kingdom, 2000; pp 9764–9800.

[93] Wentzell, P. D.; Hou, S. Exploratory data analysis with noisy measurements. *J. Chemom.* **2012**, *26*, 264–281.

[94] Wentzell, P. D.; Lohnes, M. T. Maximum likelihood principal component analysis with correlated measurement errors: theoretical and practical considerations. *Chemom. Intell. Lab. Syst.* **1999**, *45*, 65–85.

[95] Brown, C. D.; Vega-Montoto, L.; Wentzell, P. D. Derivative preprocessing and optimal corrections for baseline drift in multivariate calibration. *Appl. Spectrosc.* **2000**, *54*, 1055–1068.

[96] Braga, J. W. B.; Allegrini, F.; Olivieri, A. C. Maximum likelihood unfolded principal component regression with residual bilinearization (MLU-PCR/RBL) for second-order multivariate calibration. *Chemom. Intell. Lab. Syst.* **2017**, *170*, 51–57.

[97] McClain, R. L.; Wright, J. C. Description of the role of shot noise in spectroscopic absorption and emission measurements with photodiode and photomultiplier tube detectors: information for an instrumental analysis course. *J. Chem. Educ.* **2014**, *91*, 1455–1457.

[98] Driscoll, S.; Dowd, M.; Wentzell, P. D. Simulation of $1/f^{\alpha}$ noise for analytical measurements. *J. Chemom.* e3137, 10.1002/cem.3137.

[99] Wentzell, P. D.; Andrews, D. T.; Kowalski, B. R. Maximum likelihood multivariate calibration. *Anal. Chem.* **1997**, *69*, 2299–2311.

[100] Wentzell, P. D.; Lohnes, M. T. Maximum likelihood principal component analysis with correlated measurement errors: theoretical and practical considerations. *Chemom. Intell. Lab. Syst.* **1999**, *45*, 65–85.

[101] Brown, C. D.; Vega-Montoto, L.; Wentzell, P. D. Derivative preprocessing and optimal corrections for baseline drift in multivariate calibration. *Appl. Spectroscs.* **2000**, *54*, 1055–1068.

[102] Schreyer, S. K.; Bidinosti, M.; Wentzell, P. D. Application of maximum likelihood principal components regression to fluorescence emission spectra. *Appl. Spectroscs.* **2002**, *56*, 789–796.

[103] Tauler, R.; Viana, M.; Querol, X.; Alastuey, A.; Flight, R.; Wentzell, P.; Hopke, P. Comparison of the results obtained by four receptor modelling methods in aerosol source apportionment studies. *Atmospheric Environ.* **2009**, *43*, 3989–3997.

[104] Khaleghi, B.; Khamis, A.; Karray, F. O.; Razavi, S. N. Multisensor data fusion: A review of the state-of-the-art. *Inform. Fusion* **2013**, *14*, 28–44.

[105] Ramos, P. M.; Ruisánchez, I.; Andrikopoulos, K. S. Micro-Raman and X-ray fluorescence spectroscopy data fusion for the classification of ochre pigments. *Talanta* **2008**, *75*, 926–936.

[106] Di Anibal, C. V.; Callao, M. P.; Ruisánchez, I. ^{1}H NMR and UV-visible data fusion for determining Sudan dyes in culinary spices. *Talanta* **2011**, *84*, 829–833.

[107] Peré-Trepat, E.; Tauler, R. Analysis of environmental samples by application of multivariate curve resolution on fused high-performance liquid chromatography–diode array detection mass spectrometry data. *J. Chromatogr. A* **2006**, *1131*, 85–96.

[108] Workman Jr, J. J. Quantification of LDPE [low density poly (ethylene)], LLDPE [linear low density poly (ethylene)], and HDPE [high density poly (ethylene)] in polymer film mixtures "as received" using multivariate modeling with data augmentation (data fusion) and infrared, Raman, and near-infrared spectroscopy. *Spectrosc. Lett.* **1999**, *32*, 1057–1071.

[109] Bajoub, A.; Medina-Rodríguez, S.; Gómez-Romero, M.; Bagur-González, M. G.; Fernández-Gutiérrez, A.; Carrasco-Pancorbo, A. Assessing the varietal origin of extra-virgin olive oil using liquid chromatography fingerprints of phenolic compound, data fusion and chemometrics. *Food Chem.* **2017**, *215*, 245–255.

[110] Borràs, E.; Ferré, J.; Boqué, R.; Mestres, M.; Aceña, L.; Busto, O. Data fusion methodologies for food and beverage authentication and quality assessment–A review. *Anal. Chim. Acta* **2015**, *891*, 1–14.

[111] Biancolillo, A.; Bucci, R.; Magrì, A. L.; Magrì, A. D.; Marini, F. Data-fusion for multiplatform characterization of an Italian craft beer aimed at its authentication. *Anal. Chim. Acta* **2014**, *820*, 23–31.

[112] Dantas, C.; Tauler, R.; Ferreira, M. M. C. Exploring in vivo violacein biosynthesis by application of multivariate curve resolution on fused UV–VIS absorption, fluorescence, and liquid chromatography–mass spectrometry data. *Anal. Bioanal. Chem.* **2013**, *405*, 1293–1302.

[113] Ramos, P. M.; Ruisánchez, I. Data fusion and dual-domain classification analysis of pigments studied in works of art. *Anal. Chim. Acta* **2006**, *558*, 274–282.

[114] Måge, I.; Smilde, A. K.; van der Kloet, F. M. Performance of methods that separate common and distinct variation in multiple data blocks. *J. Chemom.* **2019**, *33*, e3085.

[115] Doeswijk, T.; Smilde, A.; Hageman, J.; Westerhuis, J.; Van Eeuwijk, F. On the increase of predictive performance with high-level data fusion. *Anal. Chim. Acta* **2011**, *705*, 41–47.

[116] Waaijenborg, S.; Korobko, O.; van Dijk, K. W.; Lips, M.; Hankemeier, T.; Wilderjans, T. F.; Smilde, A. K.; Westerhuis, J. A. Fusing metabolomics data sets with heterogeneous measurement errors. *PloS one* **2018**, *13*, e0195939.

[117] Felizardo, P.; Baptista, P.; Menezes, J. C.; Correia, M. J. N. Multivariate near infrared spectroscopy models for predicting methanol and water content in biodiesel. *Anal. Chim. Acta* **2007**, *595*, 107–113.

[118] Fernández-Cabanás, V.; Garrido-Varo, A.; Pérez-Marín, D.; Dardenne, P. Evaluation of pretreatment strategies for near-infrared spectroscopy calibration development of unground and ground compound feedingstuffs. *Appl. Spectrosc.* **2006**, *60*, 17–23.

[119] Thennadil, S.; Martens, H.; Kohler, A. Physics-based multiplicative scatter correction approaches for improving the performance of calibration models. *Appl. Spectrosc.* **2006**, *60*, 315–321.

[120] Martens, H.; Bruun, S. W.; Adt, I.; Sockalingum, G. D.; Kohler, A. Pre-processing in biochemometrics: correction for path-length and temperature effects of water in FTIR bio-spectroscopy by EMSC. *J. Chemom.* **2006**, *20*, 402–417.

[121] De Winter, J. C.; Dodou, D. Factor recovery by principal axis factoring and maximum likelihood factor analysis as a function of factor pattern and sample size. *J. Appl. Stat.* **2012**, *39*, 695–710.

[122] Driscoll, S.; Wentzell, P. D. NoiseGen - Analytical measurement error simulation software. *Chemom. Intell. Lab. Syst.* **2019**, *189*, 155 – 160.

[123] Based on a search of the journal website.

[124] Jolliffe, I. T.; Cadima, J. Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc. A* **2016**, *374*, 20150202.

[125] Drab, K.; Daszykowski, M. Clustering in analytical chemistry. *J. AOAC Int.* **2014**, *97*, 29–38.

[126] Forina, M.; Armanino, C.; Raggio, V. Clustering with dendrograms on interpretation variables. *Anal. Chim. Acta* **2002**, *454*, 13–19.

[127] Tran, T. N.; Wehrens, R.; Buydens, L. M. Clustering multispectral images: a tutorial. *Chemom. Intell. Lab. Syst.* **2005**, *77*, 3–17.

[128] Worley, B.; Powers, R. Multivariate analysis in metabolomics. *Curr. Metabolomics* **2013**, *1*, 92–107.

[129] Worley, B.; Powers, R. PCA as a practical indicator of OPLS-DA model reliability. *Curr. Metabolomics* **2016**, *4*, 97–103.

[130] Brereton, R. G.; Lloyd, G. R. Partial least squares discriminant analysis: taking the magic away. *J. Chemom.* **2014**, *28*, 213–225.

[131] Brereton, R. G. Consequences of sample size, variable selection, and model validation and optimisation, for predicting classification ability from analytical data. *Trends Anal. Chem.* **2006**, *25*, 1103–1111.

[132] Lay Jr, J. O.; Liyanage, R.; Borgmann, S.; Wilkins, C. L. Problems with the "omics". *Trends Anal. Chem.* **2006**, *25*, 1046–1056.

[133] van den Berg, R. A.; Hoefsloot, H. C.; Westerhuis, J. A.; Smilde, A. K.; van der Werf, M. J. Centering, scaling, and transformations: improving the biological information content of metabolomics data. *BMC genomics* **2006**, *7*, 142.

[134] Kurskal, J. *Statistical Computation*; Academic Press: New York, United States, 1969; pp 427–440.

[135] Hou, S.; Wentzell, P. D.; Riley, C. B. Simple methods for the optimization of complex-valued kurtosis as a projection index. *J. Chemom.* **2015**, *29*, 224–236.

[136] Wentzell, P. D.; Hou, S.; Silva, C. S.; Wicks, C. C.; Pimentel, M. F. Procrustes rotation as a diagnostic tool for projection pursuit analysis. *Anal. Chim. Acta* **2015**, *877*, 51–63.

[137] Hassanzadeh, Z.; Ghavami, R.; Kompany-Zareh, M. Radial basis function neural networks based on the projection pursuit and principal component analysis approaches: QSAR analysis of fullerene [C60]-based HIV-1 PR inhibitors. *Med. Chem. Res.* **2016**, *25*, 19–29.

[138] Hassanzadeh, Z.; Ebrahimi, P.; Kompany-Zareh, M.; Ghavami, R. Radial basis function neural networks based on projection pursuit approach and solvatochromic descriptors: single and full column prediction of gas chromatography retention behavior of polychlorinated biphenyls. *J. Chemom.* **2016**, *30*, 589–601.

[139] Zou, H.; Hastie, T.; Tibshirani, R. Sparse principal component analysis. *J. Comput. Graph. Stat.* **2006**, *15*, 265–286.

[140] Rasmussen, M. A.; Bro, R. A tutorial on the Lasso approach to sparse modeling. *Chemometr. Intell. Lab. Syst.* **2012**, *119*, 21–31.

[141] Filzmoser, P.; Gschwandtner, M.; Todorov, V. Review of sparse methods in regression and classification with application to chemometrics. *J. Chemom.* **2012**, *26*, 42–51.

[142] Hsu, Y.-L.; Huang, P.-Y.; Chen, D.-T. Sparse principal component analysis in cancer research. *Transl. Cancer Res.* **2014**, *3*, 182.

[143] Gajjar, S.; Kulahci, M.; Palazoglu, A. Selection of non-zero loadings in sparse principal component analysis. *Chemometr. Intell. Lab. Syst.* **2017**, *162*, 160–171.

[144] Xiaobo, Z.; Jiewen, Z.; Povey, M. J.; Holmes, M.; Hanpin, M. Variables selection methods in near-infrared spectroscopy. *Anal. Chim. Acta* **2010**, *667*, 14–32.

[145] Mehmood, T.; Liland, K. H.; Snipen, L.; Sæbø, S. A review of variable selection methods in partial least squares regression. *Chemometr. Intell. Lab. Syst.* **2012**, *118*, 62–69.

[146] Shahlaei, M. Descriptor selection methods in quantitative structure–activity relationship studies: a review study. *Chem. Rev.* **2013**, *113*, 8093–8103.

[147] Roque, J. V.; Cardoso, W.; Peternelli, L. A.; Teófilo, R. F. Comprehensive new approaches for variable selection using ordered predictors selection. *Anal. Chim. Actia* **2019**, *1075*, 57–70.

[148] Leardi, R.; Gonzalez, A. L. Genetic algorithms applied to feature selection in PLS regression: how and when to use them. *Chemometr. Intell. Lab. Syst.* **1998**, *41*, 195–207.

[149] Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. *Physica A* **2012**, *391*, 2193–2196.

[150] Roger, J.; Palagos, B.; Bertrand, D.; Fernandez-Ahumada, E. CovSel: Variable selection for highly multivariate and multi-response calibration: Application to IR spectroscopy. *Chemometr. Intell. Lab. Syst.* **2011**, *106*, 216–223.

[151] Silva, C. S.; Borba, F. d. S. L.; Pimentel, M. F.; Pontes, M. J. C.; Honorato, R. S.; Pasquini, C. Classification of blue pen ink using infrared spectroscopy and linear discriminant analysis. *Microchem. J.* **2013**, *109*, 122–127.

[152] Karakach, T. K.; Wentzell, P. D.; Walter, J. A. Characterization of the measurement error structure in 1D ^{1}H NMR data for metabolomics studies. *Anal. Chim. Acta* **2009**, *636*, 163–174.

[153] Ormseth, O.; S., B.; DeRobertis, A.; Horne, J.; McGowan, D.; Rand, K.; Wang, S. Temporal and spatial axes of variability in the structure of Gulf of Alaska forage fish communities. **2018**, 292–325.

[154] Kennard, R. W.; Stone, L. A. Computer aided design of experiments. *Technometrics* **1969**, *11*, 137–148.

[155] Loh, W.-Y. Classification and regression trees. *Wires. Data Min. Knowl.* **2011**, *1*, 14–23.

[156] Jones, M. C.; Rosco, J. F.; Pewsey, A. Skewness-invariant measures of kurtosis. *Am. Stat.* **2011**, *65*, 89–95.

[157] Ellison, A. M. Effect of seed dimorphism on the density-dependent dynamics of experimental populations of atriplex triangularis (chenopodiaceae). *Am. J. Bot.* **1987**, *74*, 1280–1288.

[158] Ashman, K. A.; Bird, C. M.; Zepf, S. E. Detecting bimodality in astronomical datasets. *Astron. J.* **1994**, *108*, 2348.

APPENDIX A

MATLAB CODE FOR SIMULATING $1/f^{\alpha}$ NOISE

```
function [NoiseOut,ECMout,coeff]=SI_Driscoll(samp,chan,a,p)
% Function to generate 1/f^\alpha noise
% Inputs:
%         samp is the number of sequences to be generated
%         chan is the number of channels to simulate the noise over (defined
%         as n in the manuscript
%         a is the alpha value in 1/f^\alpha
%         p is rho, which controls the amount of correlation in the noise
%         sequence
% Outputs:
%          NoiseOut is the samp x chan matrix of simulated power noise
%          ECMout is the theoretical ECM of the noise
%          coeff is a vector containing the coefficients of the filter
%
% Example: SI_Driscoll(100,500,1,1) generates 100 samples of pink noise 500
% cahnnels in length with a rho=1

% Steve Driscoll 2019, stephen.driscoll@dal.ca, stvdrisc.me

w=round(chan/2);
m=2*w+1; % Filter width
f=[1/(3*m) 1/(2*w):1/(2*w):0.5]; % Frequency vector
H=abs(1./(f.^(a/2)));
H=[H fliplr(H(2:end))]; % Mask
Z=fft(H);
c=sqrt(real(Z).^2+imag(Z).^2); % Coefficients
c=[c(w+1:end) c(1:w)];
win=0.54-0.46*cos(2*pi*[1:length(c)]/length(c)); % Hamming window
c=c.*win;
c=c/norm(c); % Normalize coefficients

% ECM via filter matrix
ECMout=zeros(chan,chan);
npts=chan+2*w;
F=zeros(npts,chan);
for i=1:chan
    indx=i+2*w;
    F(i:indx,i)=c';
end
ECMout=p^2*(F'*F); % Construct ECM with filter

```

```
42  %%ECM via self convolution of c (might be slightly slower, but don't have to store F)
43  % ECMout=zeros(chan,chan);
44  % sc2=conv(c,c);
45  % for i = 1:chan
46  %       k=1;
47  %       for j = i:chan
48  %           ECMout(i,j)=sc2(k+m);
49  %           k=k+1;
50  %       end
51  % end
52  % ECMout=triu(ECMout)+triu(ECMout,1)';
53
54  % Noise
55  [~,S,V]=svd(ECMout);
56  P=V*diag(sqrt(diag(S)));
57  NoiseOut=randn(samp,chan)*P';
58  coeff=c;
```

APPENDIX B

NOISEGEN COMMAND LINE MATLAB CODE

```
function [X,COV,COVM] = NoiseGen(chan,reps,varargin)
%%
%                       NoiseGen    v. 1.1
%
% This function simulates multiple types of instrument noise commonly
% observed in analytical chemistry using a 6-term model of the error covariance
% matrix. (See PAPER REFERENCE). The explicit inputs passed to the function are:
%
%   chan    -     the length of the output signal.
%
%   reps    -     the number of signals to generate.
%
% Additional variable length input arguments are:
%    'BO'   - Random baseline offset noise
%     [X, COV, COVM]=NoiseGen(chan,reps,'BO',SD);
%     SD (1 x 1)
%
%    'MO'   - Multiplicative offset noise
%     [X, COV, COVM]=NoiseGen(chan,reps,'MO',RSD,ReferenceSignal);
%     [X, COV, COVM]=NoiseGen(chan,reps,'MO',RSD,ReferenceSignal,Proportionality);
%     RSD (1 x 1) or (1 x reps)
%     ReferenceSignal (1 x chan) or (reps x chan)
%
%    'IID'   - White noise
%     [X, COV, COVM]=NoiseGen(500,400,'IID',SD);
%     SD (1 x 1)
%
%    'MN'   - Multiplicative noise
%     [X, COV, COVM]=NoiseGen(500,400,'MN',0.1,y,1);
%
%    'PL'   - Power law noise (1/f^alpha noise)
%     [X, COV, COVM]=NoiseGen(500,400,'PL',0.1,y,1);
%
%    'PPL'  - Proportional power law noise (1/f^alpha proportional noise)
%     [X, COV, COVM]=NoiseGen(500,400,'PPL',0.1,y,1);
%
%
% The returned variables are:
%
%   X     -     the (reps x chan) matrix of simulated noise signals.
%
```

```
%   COV   -     the (chan x chan) covariance matrix for X (X'*X/(reps-1)).
%
%   COVM  -     the (chan x chan) theoretical covariance matrix
%               If different ECM for each row, then it has dimension (chan x
%               chan x reps)
%
% Author: Steve Driscoll
% Trace Analysis Research Centre, Department of Chemistry,
% Dalhousie University, Halifax, Nova Scotia, Canada B3H 4J3
% Email: Stephen.Driscoll@dal.ca
% Website: http://groupwentzell.chemistry.dal.ca/
% April 2017; Last revision: Nov 2018
% V1.1 change log: - Added functionality to add multiple noise sources of
% the same type
%%

% Comb varargin to get parameter structure
Inpt=varargin; % Ressign input
% Check for any invalid paramter calls
s2={'MO','BO','IID','MN','PL','PPL'};
for i=1:length(Inpt)
    if iscellstr(Inpt(i))==1
        P_check=strcmpi(Inpt{i},s2);
        if isempty(find(P_check==1,1))==1
            tempT=['Unrecognized input string "', (Inpt{i}), '" (Possibilities are MO, BO, IID, MN, PL, and ...
                PPL)'];
            error(tempT)
        end
    end
end
% Build boolean vector of Inpt structure wrt valid parameters
cellfind=@(string)(@(cell_contents)(strcmpi(string,cell_contents))); % Inline function for strcmpi --> boolean
% Define cypher for paramater list [1=MO 2=BO 3=IID 4=MN 5=PN 6=PPN]
BooPa=cellfun(cellfind('MO'),Inpt);
BooPa=BooPa+2*cellfun(cellfind('BO'),Inpt);
BooPa=BooPa+3*cellfun(cellfind('IID'),Inpt);
BooPa=BooPa+4*cellfun(cellfind('MN'),Inpt);
BooPa=BooPa+5*cellfun(cellfind('PL'),Inpt);
BooPa=BooPa+6*cellfun(cellfind('PPL'),Inpt);
% Now search boolean vector for model paramters according to cypher
Pos=find(BooPa); % Find where noise types are in input
PosID=BooPa(Pos); % Find what noise types are in input
k=2; % Counter for # of params associated with each noise type after param string (2 because (1) is the ...
    noise type string)
ECM_count=1; % Counter for # of ECMs to be summed
ECM_MO_Row_Flag=0; % Flag for multiplicative offset noise per row
ECM_MN_Row_Flag=0; % Flag for multiplicative noise per row
ECM_PN_Row_Flag=0; % Flag for power law noise per row
Mcount_BO=1; % Counter for multiple BO offset contributions
Mcount_MO=1; % Counter for multiple MO offset contributions
Mcount_PN=1; % Counter for multiple PL offset contributions
Mcount_PPN=1; % Counter for multiple PPL offset contributions
Mcount_IID=1; % Counter for multiple IID offset contributions
Mcount_MN=1; % Counter for multiple MN offset contributions
ECM=zeros(chan,chan,1); % Pre-allocating
for i=1:length(BooPa)
    if BooPa(i)~=0 % a 0 in BooPa means we are not interested in element value
        if BooPa(i)==1 % MO noise begin
            RefSig=zeros(1,chan); % Force default
            c=1; % Counter for hopping along parameters (variable length)
            if length(Pos)==1
                for j=i+1:length(BooPa)
                    P1(c)=Inpt(j);
                    c=c+1;
                end
            elseif PosID(end)==1 && numel(find(PosID==1))>1 && Mcount_MO<numel(find(PosID==1))
                for j=i+1:Pos(k)-1
```

```
107                         P1(c)=Inpt(j);
108                         c=c+1;
109                     end
110                     Mcount_MO=Mcount_MO+1;
111                 elseif PosID(end)==1 && numel(find(PosID==1))>1 && Mcount_MO==numel(find(PosID==1))
112                     for j=i+1:length(BooPa)
113                         P1(c)=Inpt(j);
114                         c=c+1;
115                     end
116                 elseif PosID(end)==1
117                     for j=i+1:length(BooPa)
118                         P1(c)=Inpt(j);
119                         c=c+1;
120                     end
121                 else
122                     for j=i+1:Pos(k)-1
123                         P1(c)=Inpt(j);
124                         c=c+1;
125                     end
126                 end
127                 if length(P1{1})==1 && (isrow(P1{2})==1 || iscolumn(P1{2})==1) % One RSD, One ref vector
128                     if length(P1)==3
129                         Contr=P1{1}; % Contribution
130                         RefSig=P1{2}; % Reference signal
131                         Prop=P1{3}; % Proprtionality
132                     elseif length(P1)==2
133                         Contr=P1{1}; % Contribution
134                         RefSig=P1{2}; % Reference signal
135                         Prop=1; % Default
136                     else
137                         error('MO noise requires a reference signal (1 x chan)')
138                     end
139                     if isrow(RefSig)==1
140                         RefSig=RefSig';
141                     end
142                     if length(RefSig)~=chan
143                         error('Length of the reference signal should be equal to chan (1 x chan)')
144                     end
145                     ECM(:,:,ECM_count)=[RefSig.^(2*(Prop))]*[Contr.^2]*[RefSig.^(2*(Prop))]'; % Build ECM
146                 elseif length(P1{1})==1 && isrow(P1{2})==0 && iscolumn(P1{2})==0 % MO prop each row single prop
147                     ECM_MO_Row_Flag=1;
148                     if length(P1)==3
149                         Contr=P1{1}; % Contribution
150                         RefSig=P1{2}; % Reference signal
151                         Prop=P1{3}; % Proprtionality
152                     elseif length(P1)==2
153                         Contr=P1{1}; % Contribution
154                         RefSig=P1{2}; % Reference signal
155                         Prop=1; % Default
156                     else
157                         error('MO noise requires a reference signal (1 x chan)')
158                     end
159                     [rowe, cole]=size(RefSig);
160                     if rowe~=reps && cole~=chan
161                         error('Expecting reference signal matrix either 1 x chan or m x chan')
162                     end
163                     if length(RefSig)~=chan
164                         error('Length of the reference signal should be equal to chan (1 by chan)')
165                     end
166                     for RR=1:reps
167                         ECM_Row_MO(:,:,RR)=[RefSig(RR,:).^(2*(Prop))]'*[Contr.^2]*[RefSig(RR,:).^(2*(Prop))]; % ...
                                Build ECM
168                     end
169                 elseif length(P1{1})>1  % MO prop each row different prop
170                     ECM_MO_Row_Flag=1;
171                     if isrow(P1{2})==1 || iscolumn(P1{2})==1
172                         error('Expecting reference signal matrix of size reps by chan')
```

```
                end
                if length(P1)==3
                    Contr=P1{1}; % Contribution
                    if length(Contr)~=reps
                        error('Length of contributions should be equal to reps')
                    end
                    RefSig=P1{2}; % Reference signal (stacked m by n)
                    if length(RefSig)~=chan
                        error('Length of reference signal should be equal to chan')
                    end
                    Prop=P1{3}; % Proprtionality
                elseif length(P1)==2
                    Contr=P1{1}; % Contribution
                    if length(Contr)~=reps
                        error('Length of contributions should be equal to reps')
                    end
                    RefSig=P1{2}; % Reference signal
                    Prop=1; % Default
                else
                    error('Vector of contributions detected (reps by 1), expecting reference signal matrix ...
                        reps by chan')
                end
                for RR=1:reps
                    ECM_Row_MO(:,:,RR)=[RefSig(RR,:).^(2*(Prop))]'...
                    *[Contr(RR).^2]*[RefSig(RR,:).^(2*(Prop))];% Build ECM
                end
            end
            k=k+1;
            ECM_count=ECM_count+1;
        end
        if BooPa(i)==2 % BO noise start
            c=1;
            if length(Pos)==1
                for j=i+1:length(BooPa)
                    P1(c)=Inpt(j);
                    c=c+1;
                end
            elseif PosID(end)==2 && numel(find(PosID==2))>1 && Mcount_BO<numel(find(PosID==2))
                for j=i+1:Pos(k)-1
                    P1(c)=Inpt(j);
                    c=c+1;
                end
                Mcount_BO=Mcount_BO+1;
            elseif PosID(end)==2 && numel(find(PosID==2))>1 && Mcount_BO==numel(find(PosID==2))
                for j=i+1:length(BooPa)
                    P1(c)=Inpt(j);
                    c=c+1;
                end
            elseif PosID(end)==2
                for j=i+1:length(BooPa)
                    P1(c)=Inpt(j);
                    c=c+1;
                end
            else
                for j=i+1:Pos(k)-1
                    P1(c)=Inpt(j);
                    c=c+1;
                end
            end
            if length(P1{1})==1
                Contr=P1{1};
                ECM(:,:,ECM_count)=[ones(chan,1)]*[Contr.^2]*[ones(chan,1)]';
            else
                error('Expecting scalar for BO contribution')
            end
            k=k+1;
            ECM_count=ECM_count+1;
```

```
239             end
240             if BooPa(i)==3 % IID
241                 c=1;
242                 if length(Pos)==1
243                     for j=i+1:length(BooPa)
244                         P1(c)=Inpt(j);
245                         c=c+1;
246                     end
247                 elseif PosID(end)==3 && numel(find(PosID==3))>1 && Mcount_IID<numel(find(PosID==3))
248                     for j=i+1:Pos(k)-1
249                         P1(c)=Inpt(j);
250                         c=c+1;
251                     end
252                     Mcount_IID=Mcount_IID+1;
253                 elseif PosID(end)==3 && numel(find(PosID==3))>1 && Mcount_IID==numel(find(PosID==3))
254                     for j=i+1:length(BooPa)
255                         P1(c)=Inpt(j);
256                         c=c+1;
257                     end
258                 elseif PosID(end)==3
259                     for j=i+1:length(BooPa)
260                         P1(c)=Inpt(j);
261                         c=c+1;
262                     end
263                 else
264                     for j=i+1:Pos(k)-1
265                         P1(c)=Inpt(j);
266                         c=c+1;
267                     end
268                 end
269                 if length(P1{1})==1
270                     Contr=P1{1};
271                     ECM(:,:,ECM_count)=Contr.^2*diag(ones(1,chan));
272                 else
273                     error('Expecting scalar for IID contribution')
274                 end
275                 k=k+1;
276                 ECM_count=ECM_count+1;
277             end
278             if BooPa(i)==4 % MN
279                 RefSig=zeros(1,chan); % Force default
280                 c=1; % Counter for hopping along parameters
281                 if length(Pos)==1
282                     for j=i+1:length(BooPa)
283                         P1(c)=Inpt(j);
284                         c=c+1;
285                     end
286                 elseif PosID(end)==4 && numel(find(PosID==4))>1 && Mcount_MN<numel(find(PosID==4))
287                     for j=i+1:Pos(k)-1
288                         P1(c)=Inpt(j);
289                         c=c+1;
290                     end
291                     Mcount_MN=Mcount_MN+1;
292                 elseif PosID(end)==4 && numel(find(PosID==4))>1 && Mcount_MN==numel(find(PosID==4))
293                     for j=i+1:length(BooPa)
294                         P1(c)=Inpt(j);
295                         c=c+1;
296                     end
297                 elseif PosID(end)==4
298                     for j=i+1:length(BooPa)
299                         P1(c)=Inpt(j);
300                         c=c+1;
301                     end
302                 else
303                     for j=i+1:Pos(k)-1
304                         P1(c)=Inpt(j);
305                         c=c+1;
```

```
306                 end
307             end
308             if length(P1{1})==1 && (isrow(P1{2})==1 || iscolumn(P1{2})==1) % One RSD, One ref vector
309                 if length(P1)==3
310                     Contr=P1{1}; % Contribution
311                     RefSig=P1{2}; % Reference signal
312                     Prop=P1{3}; % Proprtionality
313                 elseif length(P1)==2
314                     Contr=P1{1}; % Contribution
315                     RefSig=P1{2}; % Reference signal
316                     Prop=1; % Default
317                 else
318                     error('MN noise requires a reference signal (1 x chan)')
319                 end
320                 if isrow(RefSig)==1
321                     RefSig=RefSig';
322                 end
323                 if length(RefSig)~=chan
324                     error('Length of the reference signal should be equal to chan (1 by chan)')
325                 end
326                 ECM(:,:,ECM_count)=Contr.^2*diag(RefSig.^(2*(Prop))); % Build ECM
327             elseif length(P1{1})==1 && isrow(P1{2})==0 && iscolumn(P1{2})==0 % MN prop each row single prop
328                 ECM_MN_Row_Flag=1;
329                 if length(P1)==3
330                     Contr=P1{1}; % Contribution
331                     RefSig=P1{2}; % Reference signal
332                     Prop=P1{3}; % Proprtionality
333                 elseif length(P1)==2
334                     Contr=P1{1}; % Contribution
335                     RefSig=P1{2}; % Reference signal
336                     Prop=1; % Default
337                 else
338                     error('MN noise requires a reference signal (1 x chan)')
339                 end
340                 [rowe, cole]=size(RefSig);
341                 if rowe~=reps && cole~=chan
342                     error('Expecting reference signal matrix either 1 by chan or m by chan')
343                 end
344                 if length(RefSig)~=chan
345                     error('Length of the reference signal should be equal to chan (1 by chan)')
346                 end
347                 for RR=1:reps
348                     ECM_Row_MN(:,:,RR)=Contr.^2*diag(RefSig(RR,:).^(2*(Prop))); % Build ECM
349                 end
350             elseif length(P1{1})>1  % MN prop each row different prop
351                 ECM_MN_Row_Flag=1;
352
353                 if isrow(P1{2})==1 || iscolumn(P1{2})==1
354                     error('Expecting reference signal matrix of size reps by chan')
355                 end
356                 if length(P1)==3
357                     Contr=P1{1}; % Contribution
358                     if length(Contr)~=reps
359                         error('Length of contributions should be equal to reps')
360                     end
361                     RefSig=P1{2}; % Reference signal (stacked m by n)
362                     if length(RefSig)~=chan
363                         error('Length of reference signal should be equal to chan')
364                     end
365                     Prop=P1{3}; % Proprtionality
366                 elseif length(P1)==2
367                     Contr=P1{1}; % Contribution
368                     if length(Contr)~=reps
369                         error('Length of contributions should be equal to reps')
370                     end
371                     RefSig=P1{2}; % Reference signal
372                     Prop=1; % Default
```

```
373                     else
374                         error('Vector of contributions detected (reps by 1), expecting reference signal matrix ...
                                reps by chan')
375                     end
376                     for RR=1:reps
377                         ECM_Row_MN(:,:,RR)=Contr(RR).^2*diag(RefSig(RR,:).^(2*(Prop))); % Build ECM
378                     end
379                 end
380                 k=k+1;
381                 ECM_count=ECM_count+1;
382             end
383             if BooPa(i)==5 % PN
384                 c=1; % Counter for hopping along parameters
385                 if length(Pos)==1
386                     for j=i+1:length(BooPa)
387                         P1(c)=Inpt(j);
388                         c=c+1;
389                     end
390                 elseif PosID(end)==5 && numel(find(PosID==5))>1 && Mcount_PN<numel(find(PosID==5))
391                     for j=i+1:Pos(k)-1
392                         P1(c)=Inpt(j);
393                         c=c+1;
394                     end
395                     Mcount_PN=Mcount_PN+1;
396                 elseif PosID(end)==5 && numel(find(PosID==5))>1 && Mcount_PN==numel(find(PosID==5))
397                     for j=i+1:length(BooPa)
398                         P1(c)=Inpt(j);
399                         c=c+1;
400                     end
401                 elseif PosID(end)==5
402                     for j=i+1:length(BooPa)
403                         P1(c)=Inpt(j);
404                         c=c+1;
405                     end
406                 else
407                     for j=i+1:Pos(k)-1
408                         P1(c)=Inpt(j);
409                         c=c+1;
410                     end
411                 end
412                 if length(P1)==3
413                     Contr=P1{1}; % Contribution
414                     Alpha=P1{2}; % Alpha
415                     Rho=P1{3}; % Rho
416                 elseif length(P1)==2
417                     Contr=P1{1};
418                     Alpha=P1{2};
419                     Rho=1;
420                 elseif length(P1)==1
421                     Contr=P1{1};
422                     Alpha=1;
423                     Rho=1;
424                 end
425                 ECM(:,:,ECM_count)=fgen(chan,Alpha,Contr,Rho);
426                 k=k+1;
427                 ECM_count=ECM_count+1;
428             end
429             if BooPa(i)==6 % PPN
430                 c=1; % Counter for hopping along parameters
431                 if length(Pos)==1
432                     for j=i+1:length(BooPa)
433                         P1(c)=Inpt(j);
434                         c=c+1;
435                     end
436                 elseif PosID(end)==6 && numel(find(PosID==6))>1 && Mcount_PPN<numel(find(PosID==6))
437                     for j=i+1:Pos(k)-1
438                         P1(c)=Inpt(j);
```

```
439                     c=c+1;
440                 end
441                 Mcount_PPN=Mcount_PPN+1;
442             elseif PosID(end)==6 && numel(find(PosID==6))>1 && Mcount_PPN==numel(find(PosID==6))
443                 for j=i+1:length(BooPa)
444                     P1(c)=Inpt(j);
445                     c=c+1;
446                 end
447             elseif PosID(end)==6
448                 for j=i+1:length(BooPa)
449                     P1(c)=Inpt(j);
450                     c=c+1;
451                 end
452             else
453                 for j=i+1:Pos(k)-1
454                     P1(c)=Inpt(j);
455                     c=c+1;
456                 end
457             end
458             if length(P1{1})==1 && (isrow(P1{2})==1 || iscolumn(P1{2})==1) % One RSD, One ref vector
459                 if length(P1)==4
460                     Contr=P1{1};
461                     RefSig=P1{2};
462                     Alpha=P1{3};
463                     Rho=P1{4};
464                 elseif length(P1)==3
465                     Contr=P1{1};
466                     RefSig=P1{2};
467                     Alpha=P1{3};
468                     Rho=1;
469                 elseif length(P1)==2
470                     Contr=P1{1};
471                     RefSig=P1{2};
472                     Alpha=1;
473                     Rho=1;
474                 end
475                 if isrow(RefSig)==1
476                     RefSig=RefSig';
477                 end
478                 if length(RefSig)~=chan
479                     error('Length of the reference signal should be equal to chan (1 by chan)')
480                 end
481                 ECM=fgen(chan,Alpha,1,Rho);
482                 fcovp=ECM.*(RefSig*RefSig');
483                 ECM(:,:,ECM_count)=Contr^2*fcovp; % Build ECM
484             elseif length(P1{1})==1 && isrow(P1{2})==0 && iscolumn(P1{2})==0 % PN prop each row single prop
485                 ECM_PN_Row_Flag=1;
486                 if length(P1)==4
487                     Contr=P1{1};
488                     RefSig=P1{2};
489                     Alpha=P1{3};
490                     Rho=P1{4};
491                 elseif length(P1)==3
492                     Contr=P1{1};
493                     RefSig=P1{2};
494                     Alpha=P1{3};
495                     Rho=1;
496                 elseif length(P1)==2
497                     Contr=P1{1};
498                     RefSig=P1{2};
499                     Alpha=1;
500                     Rho=1;
501                 end
502                 [rowe, cole]=size(RefSig);
503                 if rowe~=reps && cole~=chan
504                     error('Expecting reference signal matrix either 1 by chan or m by chan')
505                 end
```

```
506                     if length(RefSig)≠chan
507                         error('Length of the reference signal should be equal to chan (1 by chan)')
508                     end
509                     for RR=1:reps
510                         ECM=fgen(chan,Alpha,Contr,Rho);
511                         fcovp=ECM.*(RefSig(RR,:)'*RefSig(RR,:));
512                         ECM_Row_PPN(:,:,RR)=fcovp; % Build ECM
513                     end
514
515                 elseif length(P1{1})>1  % PN prop each row different prop
516                     ECM_PN_Row_Flag=1;
517
518                     if isrow(P1{2})==1 || iscolumn(P1{2})==1
519                         error('Expecting reference signal matrix of size reps by chan')
520                     end
521                     if length(P1)==4
522                         Contr=P1{1}; % Contribution
523                         Alpha=P1{3};
524                         Rho=P1{4};
525                         if length(Contr)≠reps
526                             error('Length of contributions should be equal to reps')
527                         end
528                         RefSig=P1{2}; % Reference signal (stacked m by n)
529                         if length(RefSig)≠chan
530                             error('Length of reference signal should be equal to chan')
531                         end
532                     elseif length(P1)==3
533                         Contr=P1{1}; % Contribution
534                         Alpha=P1{3};
535                         Rho=1;
536                         if length(Contr)≠reps
537                             error('Length of contributions should be equal to reps')
538                         end
539                         RefSig=P1{2}; % Reference signal
540                     elseif length(P1)==2
541                         Contr=P1{1}; % Contribution
542                         Alpha=1;
543                         Rho=1;
544                         if length(Contr)≠reps
545                             error('Length of contributions should be equal to reps')
546                         end
547                         RefSig=P1{2}; % Reference signal
548                     else
549                         error('Vector of contributions detected (reps by 1), expecting reference signal matrix ...
                                reps by chan')
550                     end
551                     for RR=1:reps
552                         ECM=fgen(chan,Alpha,Contr(RR),Rho);
553                         fcovp=ECM.*(RefSig(RR,:)'*RefSig(RR,:));
554                         ECM_Row_PPN(:,:,RR)=fcovp; % Build ECM
555                     end
556                 end
557                 k=k+1;
558                 ECM_count=ECM_count+1;
559             end
560         end
561     end
562     % All done building ECMs, now check if we should generate per row
563     if ECM_MO_Row_Flag>0 || ECM_MN_Row_Flag>0 || ECM_PN_Row_Flag>0
564         COVMt=sum(ECM,3);
565         for i=1:reps
566             if ECM_MO_Row_Flag>0
567                 COVM_Ri=ECM_Row_MO(:,:,i);
568             else
569                 COVM_Ri=zeros(chan,chan);
570             end
571
```

```
572             if ECM_MN_Row_Flag>0
573                 COVM_Ri2=ECM_Row_MN(:,:,i);
574             else
575                 COVM_Ri2=zeros(chan,chan);
576             end
577
578             if ECM_PN_Row_Flag>0
579                 COVM_Ri3=ECM_Row_PN(:,:,i);
580             else
581                 COVM_Ri3=zeros(chan,chan);
582             end
583             COVM(:,:,i)=COVM_Ri + COVM_Ri2 + COVM_Ri3 + COVMt;
584             [~,S,V]=svd(COVM(:,:,i));
585             eiid=randn(1,chan);
586             esig=eiid*sqrt(S)*V';
587             X(i,:)=esig;
588
589         end
590         COV=(X'*X)/(reps-1);
591     else % if not, then sum the ECMs and generate noise
592         COVM=sum(ECM,3);
593         [~,S,V]=svd(COVM);
594         eiid=randn(reps,chan);
595         esig=eiid*sqrt(S)*V';
596         COV=(esig'*esig)/(reps-1);
597         X=esig;
598     end
599     end
600
601     %% Function to generate 1/f.\alpha ECM
602     function [ECMout]=fgen(chan,a,p,rho)
603     w=round(chan/2);
604     m=rho*2*w+1; % Filter width
605     f=[1/(3*m) 1/(2*w):1/(2*w):0.5]; % Frequency vector
606     H=abs(1./(f.^(a/2)));
607     H=[H fliplr(H(2:end))]; % Mask
608     Z=fft(H);
609     c=sqrt(real(Z).^2+imag(Z).^2); % Coefficients
610     c=[c(w+1:end) c(1:w)];
611     win=0.54-0.46*cos(2*pi*[1:length(c)]/length(c)); % Hamming window
612     c=c.*win;
613     c=c/norm(c); % Normalize
614
615     % ECM via filter matrix
616     npts=chan+2*w;
617     F=zeros(npts,chan);
618     for i=1:chan
619         indx=i+2*w;
620         F(i:indx,i)=c';
621     end
622     ECMout=p^2*(F'*F); % Construct ECM with filter
623
624     %% ECM via self convolution of c (might be slightly slower, but don't have to store F)
625     % ECMout=zeros(chan,chan);
626     % sc2=conv(c,c);
627     % for i = 1:chan
628     %     k=1;
629     %     for j = i:chan
630     %         ECMout(i,j)=sc2(k+m);
631     %         k=k+1;
632     %     end
633     % end
634     % ECMout=triu(ECMout)+triu(ECMout,1)';
635     end
```

APPENDIX C

MATLAB CODE FOR IMPLEMENTING SPPA

```
function [T,V,Var,kurt]=SPPA_73(X,varargin)
%SPPA sparse projection pursuit analysis using genetic algorithm.
%
%Outputs:
%
%T = SPPA(X) performs sparse projection pursuit on the m*n (samples x variables)
%       matrix X, and returns the scores in T, a m*dim matrix, where dim is
%       the number of dimensions of separation.
%
%[T,V] = SPPA(X) also returns the vectors in V, a dim*n matrix containing the
%       projection vectors for each dimension.
%
%[T,V,VAR] = SPPA(X) returns the variables in VAR, a dim*nvars matrix, with each
%       column containing the chosen variables for each dimension of separation.
%
%[T,V,VAR,KURT] = SPPA(X) also returns the kurtosis value(s) for the
%       solution, in a dim*1 matrix, with each dimension having a kurtosis
%       value associated with the separation.
%Inputs:
%
%SPPA(X,...,Optionname,Optionvalue,...) performs sparse projection
%       pursuit according to the options provided. Optionname is a string,
%       as identified below, and Optionvalue is either a scalar or a string,
%       to identify the value.
%
%Options:
%
%    Dim: Number of dimensions along which separation is desired (1-3)
%           Default: 2
%           Recommendation: 1 dim for 2 classes, 2 dim for 3-4 classes,
%                           3 dim for 5-8 classes
%    Nvars: Number of variables
%           Default: 5
%           Recommendation: Nvars ≠ nsamp/25, such that nvars ≥ 3
%    Mutrate: Mutation rate (decimal between 0 and 1)
%           Default: 0.1
%    Popsize: Population size
%           Default: 100
%    Meth: Univariate (uni) or multivariate (mul) kurtosis
%           Default: uni
%    Opt: Ordinary (ord) or recentered (rec) kurtosis
```

```
%            Default: ord
%    Maxtime: Maximum time (in seconds), after which the algorithm stops if
%            it hasn't converged
%            Default: 300
%    Ctoff: Fitness value below which all individuals are considered
%            equivalent in mating, because they give adequate separation.
%            Default: 1.5 (univariate), 4.5 (multivariate), 2.25 (prod),
%            3 (sum).
%
%
%Example: To perform sparse projection pursuit with 10 variables,
% using recentered kurtosis and a mutation rate of 20%, enter:
%   [T,V,VAR] = SPPA(X,'nvars',10,'opt','rec','mutrate',0.2)

%Demo: Th

p=inputParser;
addOptional(p,'objclass',[])
addOptional(p,'classlist',{},@iscell)
addParameter(p,'dim',2,@(x) assert(0<x && x<4));
addParameter(p,'nvars',5,@(x) assert(x>1));
addParameter(p,'mutrate',0.1,@(x) assert(0≤x && x<1));
addParameter(p,'popsize',100,@(x) assert(x>0));
addParameter(p,'opt','ord',@(x) assert(isscalar(strfind('ordrec',x))));
addParameter(p,'meth','uni',@(x) assert(isscalar(strfind('unimul',x))));
addParameter(p,'maxtime',300,@(x) assert(x>0));
addParameter(p,'pctrecomb',0.3,@(x) assert(x>0));
addParameter(p,'sumprod','norm',@(x) assert(isscalar(strfind('normsumprod',x))));
addParameter(p,'exponent',4,@(x) assert(x>0));
addParameter(p,'ctoff',1.5,@(x) assert(x>0));
addParameter(p,'stat',50,@(x) assert(x>0)); %Definition of a static population (how many generations)
parse(p,varargin{:})
f=p.Results;
if strcmp(f.meth,'mul') && strcmp(f.sumprod,'norm')==false
    error('Multivariate kurtosis is incompatible with sum/product options')
end
if strcmp(f.opt,'rec') && strcmp(f.sumprod,'norm')==false
    warning('Use of recentered kurtosis is not recommended with sum/product options')
end
if strcmp(f.sumprod,'norm')
    [T,V,Var,kurt,pop]=ppga2(X,varargin{:});
else
    [T,V,Var,kurt]=ppga6(X,varargin{:});
end
end
%% -------------Dimension-by-dimension separation algorithm--------------------%%
function [scores,vectors,variables,kurt,pops]=ppga2(X,varargin)
%PPGA2  Variable selection for projection pursuit analysis, with a genetic
%       algorithm (version 2.0) with dimension-by-dimension separation.

p=inputParser;
addOptional(p,'objclass',[])
addOptional(p,'classlist',{},@iscell)
addParameter(p,'dim',2,@isnumeric)
addParameter(p,'nvars',5);
addParameter(p,'maxgen',1000,@(x) assert(x>0));
addParameter(p,'maxtime',300,@(x) assert(x>0));
addParameter(p,'meth','uni',@(x) assert(isscalar(strfind('unimul',x)))); %Univariate vs multivariate kurtosis
addParameter(p,'opt','ord',@(x) assert(isscalar(strfind('ordrec',x)))); %Ordinary vs recentered kurtosis
addParameter(p,'mutrate',0.1,@(x) assert(0≤x && x<1));
addParameter(p,'popsize',100,@(x) assert(x>0));
addParameter(p,'pctrecomb',0.3,@(x) assert(x>0));
addParameter(p,'stat',50,@(x) assert(x>0)); %Definition of a static population (how many generations)
addParameter(p,'exponent',4,@(x) assert(x>0));
addParameter(p,'ctoff',1.5,@(x) assert(x>0));
addParameter(p,'sumprod','norm',@(x) assert(isscalar(strfind('normsumprod',x))));
parse(p,varargin{:})
```

```
f=p.Results;
%Determine how many sets of variables it will output (dimvar)
%Determine how many dimensions to ask from the projection
%   pursuit alg (pursuitdim)
if strcmp('mul',f.meth)
    dimvar=1;
    pursuitdim=f.dim;
    f.ctoff=4.5;
else dimvar=f.dim;
    pursuitdim=1;
end

[nsamp,totvars]=size(X); %total number of variables
scores=zeros(nsamp,f.dim); %Will hold pp scores
variables=zeros(dimvar,f.nvars); %Will hold chosen variables
vectors=zeros(totvars,f.dim);
if rem(f.popsize,2) %Set number of elite individuals
    numret=1;
else numret=2;
end
nchild=f.popsize-numret; %the remaining population is made of retained individuals
%Mean center the data:
Morig=ones(nsamp,1)*mean(X);
X=X-Morig;
X0=X;
tic
for d=1:f.dim
%     disp(['Dimension ' num2str(d)])
    %% Define default variables
    pop=zeros(f.popsize,f.nvars);
    populations=zeros(f.popsize,f.nvars,f.maxgen); %3d matrix to store all of the historical populations
    fitness=zeros(f.popsize,f.maxgen); %Matrix storing fitness of all previous individuals

    %% Construct initial population
%     disp('Generation 1')
    for i=1:f.popsize
        pop(i,:)=randperm(totvars,f.nvars);
        [¬,¬,PPOUT]=projpursuit(X(:,pop(i,:)),pursuitdim,1,f.meth,f.opt);
        fitness(i,1)=PPOUT.K;
    end
    [fitness(:,1),isort]=sort(fitness(:,1)); %sort individuals by fitness
    for i=1:f.popsize
        pop(i,:)=pop(isort(i),:);
    end

    %%
    populations(:,:,1)=pop; %Store first generation
    for k=2:f.maxgen
        %% Elite selection
        popelite=pop(1:numret,:);
        %% Mating
        popchild=mating7(f.nvars,pop,nchild,fitness(:,k-1),popelite,totvars,f.pctrecomb,f.ctoff,f.exponent);
        %% Random mutation
        pop=mutation(nchild,f.mutrate,f.nvars,totvars,popelite,popchild);
        %% Evaluate fitness of each member
%         disp(['Generation ', num2str(k)])
        fitness2=fitness;
        for i=1:f.popsize
            [¬,¬,PPOUT]=projpursuit(X(:,pop(i,:)),pursuitdim,1,f.meth,f.opt);
            fitness(i,k)=PPOUT.K;
            %% Check for better fitness previously
            r=1;
            while r<51 && r<k %Check last 50 generations in sequence
                loctf=ismember(populations(:,:,k-r),pop(i,:),'rows'); %Find matches
                if sum(loctf)>0
                    loc=find((loctf==true),1);
                    fitness(i,k)=min(fitness(i,k),fitness2(loc,k-r));%Replace
```

```
                    break
                end
                r=r+1;
            end
        end
        [fitness(:,k),isort]=sort(fitness(:,k)); %sort individuals by fitness
         %% Intermediate plotting of fitness
          if k==2
            figure(50)
            clf
            drawnow
        end
%         plot(median(fitness(:,1:k)),'r','LineWidth',2.5)
%         hold on
%         plot(min(fitness(:,1:k)),'b','LineWidth',2.5)
%         legend('Median kurtosis','Minimum kurtosis')
%         xlabel('Generation number')
%         ylabel('Fitness')
%         set(gca,'LineWidth',2,'FontSize',9,'FontWeight','bold')
%         drawnow
%
    if k>2
        delete(h1);
        delete(h2);
    end
    plot(median(fitness(:,1:k)),'r','LineWidth',2.5)
    hold on
    plot(min(fitness(:,1:k)),'b','LineWidth',2.5)
    grid on
    axis tight
    xlabel('Generation number')
    ylabel('Kurtosis')
    h1=annotation('textbox',[.6 0.7 .3 .2],'String',strcat('Median Kurtosis: ...
        ',num2str(median(fitness(:,k)))),'EdgeColor','none','Color','red');
    h2=annotation('textbox',[.6 0.65 .3 .2],'String',strcat('Minimum Kurtosis: ...
        ',num2str(min(fitness(:,k)))),'EdgeColor','none','Color','blue');
    drawnow
        oldpop=pop;
        for i=1:f.popsize
            pop(i,:)=oldpop(isort(i),:);
        end

        %% Store population
       populations(:,:,k)=pop;
        %% Test convergence
        if k==f.maxgen
            disp(['Failed to converge after ' num2str(k) ' generations'])
            break
        end
        if k>f.stat && isequal((populations(1,:,k)),(populations(1,:,k-f.stat)))
            disp(['Stopping due to static population (' num2str(k) ' generations)'])
            break
        end
        if toc>f.maxtime*d
            disp(['Maximum time reached (' num2str(f.maxtime) ' seconds)'])
            break
        end
    end

    %% Evaluate class separation
    figure(50)
    hold off
    variables(d,:)=sort(pop(1,:));
    if strcmp('mul',f.meth)==true
        [scores,vectors,PPOUT]=projpursuit(X(:,variables(d,:)),pursuitdim,'mul');
        kurt=PPOUT.K;
        pops=pop;
```

```
            break
        end
        [T_1,V_1,PPOUT]=projpursuit(X(:,variables(d,:)),1,f.opt);%Re-run with 100 guesses
        kurt(d)=PPOUT.K; %Store kurtosis
        V2=zeros(totvars,1);
        V2(variables(d,:))=V_1;
        scores(:,d)=T_1 + (ones(nsamp,1)*mean(X)*V2) - (ones(nsamp,1)*mean(X0)*V2) ;
        vectors(:,d)=V2'; %Prepare vectors for output
        pops(:,:,d)=pop;
        if d<f.dim %Deflation
            t=X*V2;
            T1(:,d)=t;
            P(:,d)=X'*t/(t'*t);
            X=X0-(T1*P');
            if d==2
                t=(scores(:,1)).*(scores(:,2));
                T1(:,3)=t;
                P(:,3)=X'*t/(t'*t);
                X=X0-(T1*P');
            end
        end
    end

    V=vectors*inv(P'*vectors);
    vectors=V;
    scores=X0*V;

    %% Plotting
    % figure
    % if iscell(f.objclass) %Convert cell array to matrix if necessary
    %     f.objclass=cell2mat(f.objclass);
    % end
    % if size(f.classlist)>1 %Reduce objclass cell array to one dimension if necessary
    %     f.classlist=f.classlist(:,1);
    % end
    % if isempty(f.objclass)==false
    %     if isempty(f.classlist)
    %         for i=1:max(f.objclass) %Make generic group names if no classs names
    %             f.classlist{i}=['Group ' num2str(i)];
    %         end
    %     end
    %     Color=[255 57 33;215 25 232;53 33 255;29 156 207; ...
    %         12 178 85;122 43 12;0 0 0;29 84 74]./255;
    %     if f.dim==1
    %         for i=1:max(f.objclass) %Plot in 2d with colour
    %             plot(scores(f.objclass==i,1),rand(sum(f.objclass==i),1),'.','Color',Color(i,:),'MarkerSize',5)
    %             hold on
    %         end
    %         legend(f.classlist,'Location','BestOutside')
    %     end
    %     if f.dim==2
    %         for i=1:max(f.objclass) %Plot in 2d with colour
    %             plot(scores(f.objclass==i,1),scores(f.objclass==i,2),'o','Color',Color(i,:),'MarkerSize',5)
    %             hold on
    %         end
    %         legend(f.classlist,'Location','BestOutside')
    %     elseif f.dim==3
    %         for i=1:max(f.objclass) %Plot in 3d with colour
    %             hold on
    %             plot3(scores(f.objclass==i,1),scores(f.objclass==i,2), ...
    %                 scores(f.objclass==i,3),'.','Color',Color(i,:),'MarkerSize',10)
    %         end
    %         legend(f.classlist,'Location','BestOutside')
    %     end
    %     hold off
    % else
    %     if f.dim==2 %Plot all in black if no classes are given
```

```
%           plot(scores(:,1),scores(:,2),'.k','MarkerSize',5)
%       elseif f.dim==3
%           plot3(scores(:,1),scores(:,2), ...
%               scores(:,3),'.k','MarkerSize',5)
%       end
% end
toc
end

%% -------------------Mutation function--------------------------%%
function [pop]=mutation(nchild,mutrate,nvars,totvars,popelite,popchild)
for i=1:nchild
    while sum(ismember(sort([popelite;popchild],2),sort(popchild(i,:),2),'rows'))>1 %No duplicate individuals
    mutloc=rand(1,nvars)/mutrate; %generate a matrix saying where to mutate, for each child
    loc=find(mutloc<1);
    for j=1:length(loc)
        popchild(i,loc(j))=randi(totvars,1); %Replace with a random variable
        while sum(popchild(i,loc(j))==popchild(i,:))>1 %No duplicate variables
            popchild(i,loc(j))=randi(totvars,1);
        end
    end
    end
end
pop=[popelite;popchild]; %Concatenate elite and children
end

 %% ------------Fitness based mating for dim by dim-------------------------%%
 function [popchild]=mating7(nvars,pop,nchild,fitness,newpop,totvars,pctrecomb,ctoff,exponent)
popchild=zeros(nchild,nvars);
fitness(fitness<ctoff)=ctoff;
y=1./fitness.^exponent;
ranks=cumsum(y)/sum(y); %Generate cummulative distribution function
for i=1:2:nchild-1  %Go 2 children at a time
    it=0;
    parents=zeros(2,nvars);
    while (any(diff(sort(popchild(i,:)))==0)||any(diff(sort(popchild(i+1,:)))==0)) ... % No duplicate variables
            || sum(ismember(sort([newpop;popchild],2),sort(popchild(i,:),2),'rows'))>1 ... % No . . .
                duplicate individuals
            || sum(ismember(sort([newpop;popchild],2),sort(popchild(i+1,:),2),'rows'))>1
        while parents(1,:)==parents(2,:) %Ensure no duplicate parents
            parents(1,:)=pop(find(ranks>=rand(1),1),:); %Choose parents
            parents(2,:)=pop(find(ranks>=rand(1),1),:);
        end
        crossover=rand(1,nvars)/pctrecomb;
        loc1=crossover<1;
        loc2=randperm(nvars,sum(loc1));
        popchild([i,i+1],:)=parents;
        popchild(i,loc1)=parents(2,loc2);
        popchild(i+1,loc2)=parents(1,loc1);
        it=it+1;
        if it>30 %Give up and make random individuals
            popchild(i,:)=randperm(totvars,nvars);
            popchild(i+1,:)=randperm(totvars,nvars);
        end
    end
end
 end

%% -------------Sum/product separation algorithm--------------------------------%%
function [scores,vectors,variables,kurt]=ppga6(X,varargin)
%PPGA6  Variable selection for projection pursuit analysis, with a genetic
%       algorithm (version 6.0), simultaneous optimization

p=inputParser;
addOptional(p,'objclass',[])
addOptional(p,'classlist',{},@iscell)
addParameter(p,'dim',2,@(x) assert(x>0 && x<4))
```

```
addParameter(p,'nvars',5);
addParameter(p,'maxgen',1000,@(x) assert(x>0));
addParameter(p,'maxtime',300,@(x) assert(x>0));
addParameter(p,'opt','ord',@(x) assert(isscalar(strfind('ordrec',x)))); %Ordinary vs recentered kurtosis
addParameter(p,'mutrate',0.1,@(x) assert(0≤x && x<1));
addParameter(p,'popsize',100,@(x) assert(x>0));
addParameter(p,'sumprod','prod',@(x) assert(isscalar(strfind('sumprod',x))));
addParameter(p,'stat',50,@(x) assert(x>0)); %Definition of a static population
addParameter(p,'exponent',4,@(x) assert(x>0));
addParameter(p,'ctoff',2.25,@(x) assert(x>0));
parse(p,varargin{:})
f=p.Results;
f.sumprod=str2func(f.sumprod);
%Determine how many sets of variables it will output (dimvar)
scores=zeros(size(X,1),f.dim); %Will hold pp scores
variables=cell(1,f.dim); %Will hold chosen variables
[nsamp,totvars]=size(X); %total number of variables
vectors=zeros(totvars,f.dim);
kurt=zeros(f.dim,1);
if rem(f.popsize,2) %Set number of elite individuals
    numret=1;
else numret=2;
end
if strcmp('sum',f.sumprod)
    f.ctoff=3;
else f.ctoff=2.25;
end
nchild=f.popsize-numret; %the remaining population is made of retained individuals
%Mean center the data:
X=X-ones(nsamp,1)*mean(X);
X0=X; %Store initial X (because X is modified with deflation for each individual)
tic

    %% Define default variables
    pop=zeros(f.popsize,f.nvars*f.dim);
    populations=zeros(f.popsize,f.nvars*f.dim,f.maxgen); %3d matrix to store all of the historical populations
    fitness=zeros(f.popsize,f.maxgen); %Matrix storing fitness of all previous individuals

    %% Construct initial population
%     disp('Generation 1')
    for i=1:f.popsize
            T=zeros(nsamp,f.dim); %Initialize
            T1=zeros(nsamp,1);
            P=zeros(totvars,1);
            tempkurt1=zeros(f.dim,1);
            X=X0;
        for d=1:f.dim
            pop(i,((1+f.nvars*(d-1)):f.nvars*d))=randperm(totvars,f.nvars); %Create random pop
            [T(:,d),V,PPOUT]=projpursuit(X(:,pop(i,((1+f.nvars*(d-1)):f.nvars*d))),1,1,f.opt);
            tempkurt1(d)=PPOUT.K;
            V2=zeros(totvars,1);
            V2(pop(i,((1+f.nvars*(d-1)):f.nvars*d)))=V;
            if d<f.dim %Deflation
                t=X*V2;
                T1(:,d)=t;
                P(:,d)=X'*t/(t'*t);
                X=X0-(T1*P');
                if d==2 %Special deflation step to push for 8 groups in 3rd dim
                    t=(T(:,1)).*(T(:,2));
                    T1(:,3)=t;
                    P(:,3)=X'*t/(t'*t);
                    X=X0-(T1*P');
                end
            end
        end
        fitness(i,1)=f.sumprod(tempkurt1); %Calculate fitness based on sum or prod, as chosen
    end
```

```
441         [fitness(:,1),isort]=sort(fitness(:,1)); %sort individuals by fitness
442         for i=1:f.popsize
443             pop(i,:)=pop(isort(i),:);
444         end
445
446         %%
447         populations(:,:,1)=pop; %Store first generation
448         for k=2:f.maxgen
449             %% Selection of elites
450             popelite=pop(1:numret,:);
451             %% Mating
452             popchild=mating6(f.nvars,pop,nchild,fitness(:,k-1),popelite,totvars,f.dim,f.ctoff,f.exponent);
453             %% Random mutation
454             pop=mutation(nchild,f.mutrate,f.nvars*f.dim,totvars,popelite,popchild);
455             %% Evaluate fitness of each member
456 %             disp(['Generation ', num2str(k)])
457             fitness2=fitness; %Necessary if fitness is evaluated in parallel
458             for i=1:f.popsize
459                 X=X0; %Reset X for every individual
460                 tempkurt=zeros(f.dim,1);
461                 T=zeros(nsamp,f.dim);
462                 T1=zeros(nsamp,1);
463                 P=zeros(totvars,1);
464                 for d=1:f.dim
465                     [T(:,d),V,PPOUT]=projpursuit(X(:,pop(i,(1+f.nvars*(d-1)):f.nvars*d)),1,1,f.opt);
466                     tempkurt(d)=PPOUT.K;
467                     V2=zeros(totvars,1);
468                     V2(pop(i,((1+f.nvars*(d-1)):f.nvars*d)))=V;
469                     if d<f.dim %Deflation
470                         t=X*V2;
471                         T1(:,d)=t;
472                         P(:,d)=X'*t/(t'*t);
473                         X=X0-(T1*P');
474                         if d==2
475                             t=(T(:,1)).*(T(:,2));
476                             T1(:,3)=t;
477                             P(:,3)=X'*t/(t'*t);
478                             X=X0-(T1*P');
479                         end
480                     end
481                 end
482                 fitness(i,k)=f.sumprod(tempkurt); %Calculate fitness based on sum/prod
483                 %% Check for better fitness previously
484                 r=1;
485                 while r<51 && r<k %Check last 50 generations in sequence
486                     loctf=ismember(populations(:,:,k-r),pop(i,:),'rows'); %Find matches
487                     if sum(loctf)>0 %If found
488                         loc=find((loctf==true),1); %Where are they
489                         fitness(i,k)=min(fitness(i,k),fitness2(loc,k-r));%Replace
490                         break
491                     end
492                     r=r+1;
493                 end
494             end
495             [fitness(:,k),isort]=sort(fitness(:,k)); %sort individuals by fitness
496             %% Intermediate plotting of fitness
497             if k==2
498                 figure(50)
499                 clf
500                 drawnow
501             end
502             plot(median(fitness(:,1:k)),'r')
503             hold on
504             plot(min(fitness(:,1:k)),'b')
505             legend('Median fitness','Minimum fitness')
506             xlabel('Generation number')
507             ylabel('Fitness')
```

```
            drawnow
            oldpop=pop;
            for i=1:f.popsize
                pop(i,:)=oldpop(isort(i),:);
            end

            %% Store population
            populations(:,:,k)=pop;
            %% Test convergence
            if k==f.maxgen
                disp(['Failed to converge after ' num2str(k) ' generations'])
                break
            end
            if k>f.stat && isequal((populations(1,:,k)),(populations(1,:,k-f.stat)))
                disp(['Stopping due to static population (' num2str(k) ' generations)'])
                break
            end
            if toc>f.maxtime
                disp(['Maximum time reached (' num2str(f.maxtime) ' seconds)'])
                break
            end
        end
        %% Evaluate class separation
        X=X0;
        T1=zeros(nsamp,1);
        P=zeros(totvars,1);
        for d=1:f.dim
        variables{d}=sort(pop(1,(1+f.nvars*(d-1)):f.nvars*d));
        [T_1,V_1,PPOUT]=projpursuit(X(:,variables{d}),1,f.opt); %Full PP with 100 guesses
        kurt(d)=PPOUT.K; %Store kurtosis
        scores(:,d)=T_1; %Store scores
        V2=zeros(totvars,1);
        V2(variables{d})=V_1; %Construct vectors
        vectors(:,d)=V2';
        if d<f.dim %Deflation
            t=X*V2;
            T1(:,d)=t;
            P(:,d)=X'*t/(t'*t);
            X=X0-(T1*P');
            if d==2 %Special deflation step to push into 8 quadrants
                t=(scores(:,1)).*(scores(:,2));
                T1(:,3)=t;
                P(:,3)=X'*t/(t'*t);
                X=X0-(T1*P');
            end
        end
        end
    %% Plotting
    figure
    if iscell(f.objclass) %Convert cell array to matrix if necessary
        f.objclass=cell2mat(f.objclass);
    end
    if size(f.classlist)>1 %Reduce objclass cell array to one dimension if necessary
        f.classlist=f.classlist(:,1);
    end
    if isempty(f.objclass)==false
        if isempty(f.classlist)
            for i=1:max(f.objclass) %Make generic group names if no classs names
                f.classlist{i}=['Group ' num2str(i)];
            end
        end
        Color=[255 57 33;215 25 232;53 33 255;29 156 207; ...
            12 178 85;122 43 12;0 0 0;29 84 74]./255;
        if f.dim==1
            for i=1:max(f.objclass) %Plot in 2d with colour
                plot(scores(f.objclass==i,1),scores(f.objclass==i,1),'.','Color',Color(i,:),'MarkerSize',5)
                hold on
```

```
575             end
576             legend(f.classlist,'Location','BestOutside')
577         end
578         if f.dim==2
579             for i=1:max(f.objclass) %Plot in 2d with colour
580                 plot(scores(f.objclass==i,1),scores(f.objclass==i,2),'.','Color',Color(i,:),'MarkerSize',10)
581                 hold on
582             end
583             legend(f.classlist,'Location','BestOutside')
584         elseif f.dim==3
585             for i=1:max(f.objclass) %Plot in 3d with colour
586                 hold on
587                 plot3(scores(f.objclass==i,1),scores(f.objclass==i,2), ...
588                     scores(f.objclass==i,3),'.','Color',Color(i,:),'MarkerSize',10)
589             end
590             legend(f.classlist,'Location','BestOutside')
591             hold off
592         end
593     else
594         if f.dim==2 %Plot all in black if no classes are given
595             plot(scores(:,1),scores(:,2),'.k','MarkerSize',5)
596         elseif f.dim==3
597             plot3(scores(:,1),scores(:,2), ...
598                 scores(:,3),'.k','MarkerSize',5)
599         end
600     end
601     toc
602     end
603       %% ------------Fitness based mating for sum/prod------------------%%
604      function [popchild]=mating6(nvars,pop,nchild,fitness,newpop,totvars,dim,ctoff,exponent)
605     popchild=zeros(nchild,nvars);
606     fitness(fitness<ctoff)=ctoff;
607     y=1./fitness.^exponent;
608     ranks=cumsum(y)/sum(y);
609     for i=1:2:nchild-1
610         it=0;
611         parents=zeros(2,nvars*dim);
612         while (any(diff(sort(popchild(i,:)))==0)||any(diff(sort(popchild(i+1,:)))==0)) ... %no duplicate variables
613                 || sum(ismember(sort([newpop;popchild],2),sort(popchild(i,:),2),'rows'))>1 %no duplicate individuals
614             while parents(1,:)==parents(2,:) %No duplicate parents
615                 parents(1,:)=pop(find(ranks>=rand(1),1),:); %Select parents based on cum. dist. function (ranks)
616                 parents(2,:)=pop(find(ranks>=rand(1),1),:);
617             end
618             for d=1:dim
619                 varsel=randperm(nvars*2);
620                 pool=parents(:,(1+(d-1)*(nvars)):d*nvars);
621                 popchild(i,(1+(d-1)*(nvars)):d*nvars)=pool(varsel(1:nvars));
622                 popchild(i+1,(1+(d-1)*(nvars)):d*nvars)=pool(varsel((nvars+1):(nvars*2)));
623             end
624             it=it+1;
625             if it>30 %Generate random individual if it cannot find a new unique one after 30 iters.
626                 popchild(i,:)=randperm(totvars,nvars*dim);
627                 popchild(i+1,:)=randperm(totvars,nvars*dim);
628             end
629         end
630     end
631      end
632
633      %% -----------------Projection pursuit algorithm---------------------------------%%
634       function [ T,V,ppout ]=projpursuit(X,varargin)
635     %PROJPURSUIT  Projection Pursuit Analysis
636     %   T = PROJPURSUIT(X) performs projection pursuit analysis on the
637     %   matrix X, using default algorithmic parameters (see below) and
638     %   returns the scores in T.  The matrix X is mxn (objects x variables)
639     %   and T is mxp (objects x scores), where the default value of p is 2.
640     %
641     %   Projection pusuit (PP) is an exploratory data analysis technique that
```

```
%   seeks to optimize a projection index to find "interesting" projections
%   of objects in a lower dimensional space.  In this algorithm, kurtosis
%   (fourth statistical moment) is used as the projection index.
%
%   T = PROJPURSUIT(X,P) returns the first P projection pursuit scores.
%   Usually P is 2 or 3 for data visualization (default = 2).
%
%   T= PROJPURSUIT(X,P,GUESS) uses GUESS initial random starting points for
%   the optimization.  Larger values of GUESS decrease the likelihood of a
%   local optimum, but increase computation time.  The default value is
%   GUESS=100.
%
%   T = PROJPURSUIT(X,...,S1,S2,...) specifies algorithmic variation of
%   the PP analysis, where S1, S2, etc. are character strings as specified
%   with the options below.
%
%      Stepwise Unvariate ('Uni') or Multivariate ('Mul') Kurtosis
%      Ordinary ('Ord') or Recentered ('Rec') Kurtosis
%      Orthogonal Scores ('SO') or Orthogonal Loadings ('VO')
%      Minimization ('Min') or Maximization ('Max') of Kurtosis
%      Shifted ('Sh') or Standard ('St') Optimization Method
%
%   In each case, the default option is the first one.  These variations
%   are discussed in more detail below under the heading 'Algorithms'.
%
%   [T,V] = PROJPURSUIT(...) returns the P loading vectors in V (nxp).
%
%   [T,V,PPOUT] = PROJPURSUIT(...) returns additional outputs from the PP
%   analysis in the structured variable PPOUT. These vary with the
%   algorithm selected, as indicated below.
%         PPOUT.K:   Kurtosis value(s) for the optimum subspace. Can
%                    otherwise be found by searching for the max/min of
%                    PPOUT.kurtObj. For multivariate methods, this is a
%                    scalar; for univariate methods, it is a 1xP vector
%                    corresponding to the optimum value in each step.
%         PPOUT.kurtObj: Kurtosis values for different initial guesses.
%         PPOUT.convFlag: Convergence status for different initial guesses.
%         PPOUT.W:   If the scores are made orthogonal for univariate
%                    methods, W and P are intermediate matrices in the
%                    calculation of deflated matrices. The loadings are not
%                    orthogonal in this case and are given by V=W*inv(P'*W).
%                    If the projection vectors are set to be orthogonal, or
%                    multivariate algorithms are used, these are not
%                    calculated.
%         PPOUT.P:   See PPOUT.W.
%         PPOUT.Mu:  The estimated row vector subtracted from the data
%                    set, X, for re-centered methods.
%
%   Algorithms:
%
%   Univariate vs. Multivariate
%      In the stepwise univariate PP algorithm, univariate kurtosis is
%      optimized as the projection vectors are extracted sequentially,
%      with deflation of the original matrix at each step. In the
%      multivariate algorithm, multivariate kurtosis is optimized as
%      all of the projection vectors are calculated simultaneously.
%      Univariate is best for small numbers of balanced clusters that can
%      be separated in a binary fashion and runs faster than the
%      multivariate algorithm.
%
%   Minimization vs Maximization
%      Minimization of kurtosis is most often used to identify clusters.
%      Maximization may be useful in identifying outliers. Maximization
%      is not an option for recentered algorithms.
%
%   Orthogonal Scores vs. Orthogonal Loadings
%      This option is only applicable to stepwise univariate algorithms
```

```
%       for P>1 and relates to the deflation of the data matrix in the
%       stepwise procedure. Orthogonal scores are generally preferred,
%       since these avoid correlated scores in multiple dimensions.
%       However, the projection vectors (loadings) will not be orthogonal
%       in this case.  For multivariate methods, the loadings are always
%       orthogonal.
%
%   Ordinary vs. Recentered Algorithms
%       For data sets that are unbalanced (unequal number of members in each
%       class, the recentered algorithms may provide better results than
%       ordinary PP.
%
%   Shifted vs. Standard Algorithms
%       This refers to the mathematics of the quasi-power method. The
%       shifted algorithm should be more stable, but the option for the
%       standard algorithm has been retained. The choice is not available
%       for recentered algorithms, and the shifted algorithm may still be
%       implemented if solutions become unstable.

%%
%                               Version 1.0
%
% Original algorithms written by Siyuan Hou.
% Additional modifications made by Peter Wentzell and Chelsi Wicks.
%

%% Set Default Parameters
MaxMin='Min';
StSh='Sh';
VSorth='SO';
Meth='Uni';
CenMeth='Ord';
p=2;
guess=100;
ppout.W=[];
ppout.P=[];
ppout.Mu=[];

%% Check for valid inputs and parse as required

if ~exist('X','var')
    error('PP:DefineVar:X','Provide data matrix X')
elseif ~isa(X,'double')
    error('PP:InvalVar:X','Invalid data matrix X')
end

% Extract numeric variables if present
opt_start=1;        % Marks beginning of algorithmic options in varargin
if nargin>1
    if isa(varargin{1},'double')   % Second argument is p?
        p=round(varargin{1});
        opt_start=2;
        if nargin>2
            if isa(varargin{2},'double') % No. of guesses given?
                guess=round(varargin{2});
                opt_start=3;
            end
        end
    end
end

% Check numeric variables
[m,n]=size(X);          % Check numeric variables
if numel(p)~=1 || p<1  % Check if p is valid
    error('PP:InvalVar:p','Invalid value for subspace dimension.')
elseif numel(guess)~=1 || guess<1    % Check no. of guesses
    error('PP:InvalVar:guess','Invalid value for number of guesses.')
```

```
776  elseif m<(p+1) || n<(p+1)   % Check X
777      error('PP:InvalVar:X','Insufficient size of data matrix.')
778  end
779
780  % Extract string variables if present
781  Allowd_opts='unimulordrecsovominmaxshst';
782  OptStrg='';                          % String to concatenate all options
783  for i=opt_start:size(varargin,2)
784      if ischar(varargin{i})
785          temp=lower(varargin{i});
786          if isempty(strfind(Allowd_opts,temp))
787              error('PP:InvalVar:OptStrg','Invalid option syntax.')
788          end
789          OptStrg=strcat(OptStrg,temp); %creates string of all character options
790      else
791          error('PP:InvalVar:OptStrg','Invalid option syntax.')
792      end
793  end
794
795  % Set options for algorithm
796
797  if strfind(OptStrg,'max')
798      if strfind(OptStrg,'min')
799          error('PP:InvMode:MaxMin','Choose either to minimize or maximize.')
800      elseif strfind(OptStrg,'rec')
801          error('PP:InvMode:MaxMin','Maximization not available for recentered PP.')
802      else
803          MaxMin='Max';
804      end
805  end
806
807  if strfind(OptStrg,'st')
808      if strfind(OptStrg,'sh')
809          error('PP:InvMode:StSh','Choose either the standard or shifted method')
810      else
811          StSh='St';
812      end
813  end
814
815  if strfind(OptStrg,'vo')
816      if strfind(OptStrg,'so')
817          error('PP:InvMode:VSorth','Choose for either the scores or the projection vectors to be orthogonal')
818      else
819          VSorth='VO';
820      end
821  end
822
823  if strfind(OptStrg,'mul')
824      if strfind(OptStrg,'uni')
825          error('PP:InvMode:UniMul','Choose either univariate or multivariate method')
826      else
827          Meth='Mul';
828      end
829  end
830
831  if strfind(OptStrg,'rec')
832      if strfind(OptStrg,'ord')
833          error('PP:InvMode:OrdRec','Choose either the ordinary or recentred method')
834      else
835          CenMeth='Rec';
836      end
837  end
838
839  %% Carry out PP using appropriate algorithm
840
841  if strcmp(Meth,'Mul')
842      if strcmp(CenMeth,'Rec')
```

```
843             %disp('Performing recentered multivariate PP')  % Diagnostic
844             [T,V,R,K,Vall,kurtObj,convFlag]=rcmulkurtpp(X,p,guess);
845             ppout.K=K;
846             ppout.kurtObj=kurtObj;
847             ppout.convFlag=convFlag;
848             ppout.Mu=R;
849         else
850             %disp(['Performing ordinary multivariate PP(' StSh ')'])  % Diagnostic
851             [T,V,Vall,kurtObj,convFlag]=mulkurtpp(X,p,guess,MaxMin,StSh);
852             ppout.K=min(kurtObj);
853             ppout.kurtObj=kurtObj;
854             ppout.convFlag=convFlag;
855         end
856     else
857         if strcmp(CenMeth,'Rec')
858             %disp(['Performing recentered univariate PP(' VSorth ')'])  % Diagnostic
859             [T,V,R,W,P,kurtObj,convFlag]=rckurtpp(X,p,guess,VSorth);
860             ppout.K=min(kurtObj);
861             ppout.kurtObj=kurtObj;
862             ppout.convFlag=convFlag;
863             ppout.W=W;
864             ppout.P=P;
865             ppout.Mu=R;
866         else
867             %disp(['Performing ordinary univariate PP(' StSh ',' VSorth ')'])  % Diagnostic
868             [T,V,W,P,kurtObj,convFlag]=okurtpp(X,p,guess,MaxMin,StSh,VSorth);
869             ppout.K=min(kurtObj);
870             ppout.kurtObj=kurtObj;
871             ppout.convFlag=convFlag;
872             ppout.W=W;
873             ppout.P=P;
874         end
875     end
876       end
877
878     %% Original Univariate Kurtosis Projection Pursuit Algorithm
879     function [T,V,W,P,kurtObj,convFlag]=okurtpp(X,p,guess,MaxMin,StSh,VSorth)
880     %% Quasi-power methods to optimize univariate kurtosis
881     %
882     %%
883     % Input:
884     %       X:        The data matrix. Rows denote samples, and columns denote variables.
885     %       p:        The number of projection vectors to be extracted.
886     %       guess:    The number of initial guesses for optimization, e.g. 100.
887     %                 The more dimensions, the better to have more initial guesses.
888     %       MaxMin:   A string indicating to search for maxima or minima of kurtosis.
889     %                 The available choices are "Max" and "Min".
890     %                    "Max": To search for maxima of kurtosis
891     %                    "Min": To search for minima of kurtosis
892     %                 Projections revealing outliers tend to have a maximum
893     %                 kurtosis, while projections revealing clusters tend to
894     %                 have a minimum kurtosis. Maximization seems more important
895     %                 in ICA to look for independent source signals, while
896     %                 minimization appears useful in PP to looks for clusters.
897     %       StSh:     A string indicating if the standard or the shifted algorithm
898     %                 is used. The available choices are "St" and "Sh".
899     %                    "St": To use the standard quasi-power method.
900     %                    "Sh": To use the shifted quasi-power method.
901     %       VSorth:   A string indicating whether the scores or projection
902     %                 vectors are orthogonal. The available choices are
903     %                    "VO": The projection vectors are orthogonal, but
904     %                          scores are not, in general.
905     %                    "SO": The scores are orthogonal, but the projection
906     %                          vectors are not, in general.
907     %                 If not specified (empty), the scores are made orthogonal.
908     % Output:
909     %       T:          Scores.
```

```
910 %       V:          Projection vectors.
911 %       W & P:      If the scores are made orthogonal, they appear in the
912 %                   deflation steps. They can be used to calculate the final
913 %                   projection vectors with respect to the original matrix.
914 %                   If the projection vectors are set to be orthogonal, they
915 %                   are not needed.
916 %       kurtObj:    Kurtosis values for different initial guesses.
917 %       convFlag:   Convergence status for different initial guesses.
918
919 %% Note:
920 %
921 % The scores orthogonality is based on mean-centered data. If the data
922 % are not mean-centered, the mean scores are added to the final scores and
923 % therefore the final scores may not be not orthogonal.
924 %
925 % For minimization of kurtosis, the standard method (st) may not be stable
926 % when the number of samples is only slightly larger than the number of
927 % variables. Thus, the shifted method (sh) is recommended.
928
929 % Author:
930 % S. Hou, University of Prince Edward Island, Charlottetown, PEI, Canada, 2012.
931 %
932 % Version, Nov. 2012. This is the updated version. The original version was
933 % reported in the literature: S. Hou, and P. D. Wentzell, Fast and Simple
934 % Methods for the Optimization of Kurtosis Used % as a Projection Pursuit
935 % Index, Analytica Chimica Acta, 704 (2011) 1-15.
936 %%
937 if exist('VSorth','var')
938     if (strcmpi(VSorth,'VO')||strcmpi(VSorth,'SO'))
939         % Pass
940     else
941         error('Please correctly choose the orthogonality of scores or projection vectors.')
942     end
943 else
944     VSorth='SO';
945 end
946 %
947 if strcmpi(StSh,'St') || strcmpi(StSh,'Sh')
948     StSh0=StSh;
949 else
950     error('Please correctly choose "St" or "Sh" method.')
951 end
952
953 %% Mean center the data and reduce the dimensionality of the data
954 % if the number of variables is larger than the number of samples.
955 Morig=ones(size(X,1),1)*mean(X);
956 X=X-Morig;
957 rk=rank(X);
958 if p>rk
959     p=rk;
960     display('The component number larger than the data rank is ignored.');
961 end
962
963 [Uorig,Sorig,Worig]=svd(X,'econ');
964 X=Uorig*Sorig;
965 X=X(:,1:rk);
966 Worig=Worig(:,1:rk);
967 X0=X;
968 %% Initial settings
969 [r,c]=size(X);
970 maxcount=10000;
971 convFlag=cell(guess,p);
972 kurtObj=zeros(guess,p);
973 T=zeros(r,p);
974 W=zeros(c,p);
975 P=zeros(c,p);
976 %%
```

```
977  for j=1:p
978      cc=c+1-j;
979      convlimit=(1e-10)*cc;          % Set convergence limit
980      wall=zeros(cc,guess);
981      [U,S,Vj]=svd(X,'econ');
982      Vj=Vj(:,1:cc);                 % This reduces the dimensionality of the data
983      X=X*Vj;                        % when deflation is performed.
984      if strcmpi(MaxMin,'Max')       % Option to search for maxima.
985          invMat2=1./diag(X'*X);     % Note X'*X is diagonal due to SVD previously
986      elseif strcmpi(MaxMin,'Min')   % Option to search for minima.
987          Mat2=diag(X'*X);
988          VM=zeros(cc*cc,r);         % This is used to calculate "Matla" later
989          for i=1:r
990              tem=X(i,:)'*X(i,:);
991              VM(:,i)=reshape(tem,cc*cc,1);
992          end
993      else
994          error('Please correctly choose to maximize or minimize the kurtosis.')
995      end
996  %% Loop for different initial guesses of w
997      for k=1:guess
998          w=randn(cc,1);   % Random initial guess of w for real numbers
999          w=w/norm(w);
1000         oldw1=w;
1001         oldw2=oldw1;
1002         StSh=StSh0;
1003         count=0;
1004         while 1
1005             count=count+1;
1006             x=X*w;
1007 %% Maximum or minimum search
1008             if strcmpi(MaxMin,'Max')          % Option to search for maxima.
1009                 w=invMat2.*(X'*(x.*x.*x));
1010             elseif strcmpi(MaxMin,'Min')      % Option to search for minima.
1011                 Mat1=sum(VM*(x.*x),2);
1012                 Mat1=reshape(Mat1,cc,cc);
1013                 w=Mat1\(Mat2.*w);
1014             end
1015 %% Test convergence
1016             w=w/norm(w);
1017             L1=(w'*oldw1).^2;
1018             if (1-L1) < convlimit
1019                 convFlag(k,j)={'Converged'};
1020                 break   % Exit the "while ... end" loop if converging
1021             elseif count>maxcount
1022                 convFlag(k,j)={'Not converged'};
1023                 break   % Exit if reaching the maximum iteration number
1024             end
1025 %% Continue the interation if "break" criterion is not reached
1026             if strcmpi(StSh,'Sh')             % Shifted method
1027                 w=w+0.5*oldw1;
1028                 w=w/norm(w);
1029             elseif strcmpi(MaxMin,'Min')      % "St" method & minimization
1030                 L2=(w'*oldw2).^2;             % If "St" method is not stable,
1031                 if L2>L1 && L2>0.99           % change to shifted method
1032                     StSh='Sh';
1033                     display('Warning: "St" method is not stable. Change to shifted method.');
1034                 end
1035                 oldw2=oldw1;
1036             end                   % "St" method & maximization: do nothing
1037             oldw1=w;
1038         end
1039 %% Save the projection vectors for different initial guesses
1040         wall(:,k)=w;
1041     end
1042 %% Find the best solution from different initial guesses
1043     kurtObj(:,j)=kurtosis(X*wall,1,1);
```

```
1044        if strcmpi(MaxMin,'Max')          % Find the best projection vector for maximum search.
1045            [tem,ind]=max(kurtObj(:,j));
1046        elseif strcmpi(MaxMin,'Min')      % Find the best projection vector for minimum search.
1047            [tem,ind]=min(kurtObj(:,j));
1048        end
1049        Wj=wall(:,ind);                   % Take the best projection vector as the solution.
1050
1051    %% Deflation of matrix
1052        if strcmpi(VSorth,'VO')           % This deflation method makes the
1053            t=X*Wj;                       % projection vectors orthogonal.
1054            T(:,j)=t;
1055            W(:,j)=Vj*Wj;
1056            X=X0-X0*W*W';
1057        elseif strcmpi(VSorth,'SO') % This deflation method makes the scores orthogonal.
1058            t=X*Wj;           % This follows the deflation method used in the non-linear partial
1059            T(:,j)=t;         % least squares (NIPALS), which is well-known in chemometrics.
1060            W(:,j)=Vj*Wj;
1061            Pj=X'*t/(t'*t);
1062            P(:,j)=Vj*Pj;
1063            X=X0-T*P';            % This uses the Gram-Schmidt process for complex-valued vectors
1064        end
1065    end
1066    %% Transform back into original space
1067    W=Worig*W;              % The projection vector(s) are tranformed into original space.
1068    if strcmpi(VSorth,'VO')
1069        V=W;
1070        W=[];
1071        P=[];
1072        T=T+Morig*V;        % Adjust the scores. Mean scores are added.
1073    else
1074        P=Worig*P;          % Vectors in P are tranformed into original space.
1075        V=W*inv(P'*W);      % Calculate the projection vectors by V=W*inv(P'*W)
1076        T=T+Morig*V;        % Adjust the scores. Mean scores are added.
1077        tem=sqrt(sum(abs(V).^2));
1078        V=V./(ones(size(V,1),1)*tem); % Make the projection vectors be unit length
1079        T=T./(ones(size(T,1),1)*tem); % Adjust T with respect to V
1080        P=P.*(ones(size(P,1),1)*tem); % Adjust P with respect to V
1081    end
1082    end
1083    %% ================== End of the function ======================
1084    %%
1085
1086    %% Original Multivariate Kurtosis Projection Pursuit Algorithm
1087    function [T,V,Vall,kurtObj,convFlag]=mulkurtpp(X,p,guess,MaxMin,StSh)
1088    %
1089    % Quasi-power method to optimize multivariate kurtosis.
1090    %%
1091    % Input:
1092    %       X:      The data matrix.
1093    %       p:      The dimension of the plane or heperplane (Normally, 2 or 3).
1094    %       guess:  The number of initial guesses for optimization.
1095    %               The more dimension, the better to have more initial guesses.
1096    %       MaxMin: A string indicating to search for maxima or minima of kurtosis.
1097    %               The available choices are "Max" and "Min".
1098    %                   "Max": To search for maxima of kurtosis
1099    %                   "Min": To search for minima of kurtosis
1100    %               Projections revealing outliers tend to have a maximum
1101    %               kurtosis, while projections revealing clusters tend to
1102    %               have a minimum kurtosis.
1103    %       StSh:   A string indicating if the standard or the shifted algorithm
1104    %               is used. The available choices are "St" and "Sh".
1105    %                   "St": To use the standard quasi-power method.
1106    %                   "Sh": To use the shifted quasi-power method.
1107    % Output:
1108    %       T:          Scores.
1109    %       V:          Projection vectors.
1110    %       Vall:       All the projection vectors found based on different initial guesses. The
```

```
%                   best projection vectors are chosen as the solutions and put in V
%         kurtObj:  Kurtosis values for different projection vectors.
%         convFlag: Convergence status for the initial guesses..
%%
%  Mean center the data and reduce the dimensionality of the data if the number
%  of variables is larger than the number of samples.
Morig=ones(size(X,1),1)*mean(X);
X=X-Morig;
rk=rank(X);
[Uorig,Sorig,Vorig]=svd(X,'econ');
X=Uorig*Sorig;
X=X(:,1:rk);
Vorig=Vorig(:,1:rk);
[r,c]=size(X);
%%
% Initial settings
maxcount=10000;
convlimit=1e-10;
Vall=cell(1,guess);
kurtObj=zeros(1,guess);
convFlag=cell(1,guess);
%%
for k=1:guess
    V=randn(c,p); % Random initial guess of V
    V=orth(V);
    oldV=V;
    count=0;
    while 1
        count=count+1;
        A=V'*X'*X*V;
        Ainv=inv(A);
%%
%           kurt=0;
%           Mat=zeros(c,c);
%           for i=1:r
%               scal=(X(i,:)*V*Ainv*V'*X(i,:)');
%               kurt=kurt+scal^2;
%               Mat=Mat+scal*X(i,:)'*X(i,:);
%           end
        scal=sqrtm(Ainv)*V'*X';
        scal=sqrt(sum(scal.^2,1));
        Mat=((ones(c,1)*scal).*X');
        Mat=Mat*Mat';
% The four lines replace the above loop to increase the speed.

%%
        if strcmpi(MaxMin,'Max')        % Option to search for maxima.
            M=inv(X'*X)*Mat;
            if strcmpi(StSh,'St')
                V=M*V;
            elseif strcmpi(StSh,'sh')
                V=(M+eye(c)*trace(M)/c)*V;
            else
                error('Please correctly choose to standard or shifted method.')
            end
        elseif strcmpi(MaxMin,'Min')    % Option to search for minima.
            M=inv(Mat)*(X'*X);
            if strcmpi(StSh,'St')
                V=M*V;
            elseif strcmpi(StSh,'sh')
                V=(M+eye(c)*trace(M)/c)*V;
            else
                error('Please correctly choose to standard or shifted method.')
            end
        else
            error('Please correctly choose to maximize or minimize the kurtosis.')
        end
```

```
%%
        [V,TemS,TemV]=svd(V,'econ');          % Apply SVD to find an orthonormal basis.
        if sum((oldV-V).^2)/(c*p)<convlimit % Test convergence.
            convFlag(1,k)={'Converged'};
            break
        elseif count>maxcount
            convFlag(1,k)={'Not converged'};
            break
        end
        oldV=V;
    end
    kurtObj(1,k)=r * sum( (sum( (sqrtm(Ainv)*V'*X').^2, 1 ) ).^2 ); % Calculate kurtosis.
    %%
    [U,S,V]=svd(X*V*V','econ');
    Vall{1,k}=Vorig*V(:,1:p);
end
%%
if strcmpi(MaxMin,'Max')          % Find the best projection vector for maximum search.
    [tem,ind]=max(kurtObj(1,:));
elseif strcmpi(MaxMin,'Min')      % Find the best projection vector for minimum search.
    [tem,ind]=min(kurtObj(1,:));
end

V=Vall{1,ind};              % Store the projection vectors
T=X*Vorig'*V+Morig*V;       % Calculate the scores.
end
%% ================== End of the function ======================
%%

%% Recentered Univariate Kurtosis Projection Pursuit Algorithm
function [T,V,R,W,P,kurtObj,convFlag]=rckurtpp(X,p,guess,VSorth)
%
% Algorithms for minimization of recentered kurtosis. recentered kurtosis
% is proposed as a projection pursuit index in this work, aiming to deal with
% unbalanced clusters.
%
%%
% Input:
%       X:          The data matrix.
%       p:          The number of projection vectors to be extracted.
%       guess:      The number of initial guesses for optimization.
%                   The more dimensions, the better to have more initial guesses.
%       VSorth:     A string indicating whether the scores or projection
%                   vectors are orthogonal. The available choices are
%                     "VO": The projection vectors are orthogonal, but
%                           scores are not, in general.
%                     "SO": The scores are orthogonal, but the projection
%                           vectors are not, in general.
%                  If not specified (empty), the scores are made orthogonal.
% Output:
%       T:          Scores.
%       V:          Projection vectors.
%       R:          The estimated row vector subtracted from the data set X.
%       W & P:      If users choose scores are orthogonal, they appear in the
%                   deflation steps. They can be used to calculate the final
%                   projection vectors with respect to the original matrix X.
%                   If the projection vectors are set to be orthogonal, they
%                   are not needed.
%       kurtObj:    Kurtosis values for different initial guesses.
%       convFlag:   Convergence status for different initial guesses.

%% Note:
% Users have the option to make the projection vectors or scores orthogonal.
% The scores orthogonality is based on mean-centered data. If the data
% are not mean-centered, the mean scores are added to the final scores and
% therefore the final scores may not be not orthogonal.
%% Author:
```

```
% S. Hou, University of Prince Edward Island, Charlottetown, PEI, Canada, 2012.
%
% This algorithm is based on the Quasi-Power methods. The Quasi-power
% methods were reported in the literature: S. Hou, and P. D. Wentzell,
% Fast and Simple Methods for the Optimization of Kurtosis Used as a
% Projection Pursuit Index, Analytica Chimica Acta, 704 (2011) 1-15.
%
%%
if exist('VSorth','var')
    if (strcmpi(VSorth,'VO')||strcmpi(VSorth,'SO'))
        % Pass
    else
        error('Please correctly choose the orthogonality of scores or projection vectors.')
    end
else
    VSorth='SO';
end
%
%% Mean center the data and reduce the dimensionality of the data
% if the number of variables is larger than the number of samples.
Morig=mean(X);
X=X-ones(size(X,1),1)*Morig;
rk=rank(X);
if p>rk
    p=rk;
    display('The component number larger than the data rank is ignored.');
end
%
[Uorig,Sorig,Worig]=svd(X,'econ');
X=Uorig*Sorig;
X=X(:,1:rk);
Worig=Worig(:,1:rk);
X0=X;
%% Initial settings
[r,c]=size(X);
maxcount=10000;
convFlag=cell(guess,p);
kurtObj=zeros(guess,p);
T=zeros(r,p);
W=zeros(c,p);
P=zeros(c,p);
ALPH=zeros(1,p);
%%
for j=1:p
    cc=c+1-j;
    convlimit=(1e-10)*cc;          % Set convergence limit
    wall=zeros(cc,guess);
    alphall=zeros(1,guess);
    [U,S,Vj]=svd(X,'econ');
    Vj=Vj(:,1:cc);                 % This reduces the dimensionality of the data
    X=X*Vj;                        % when deflation is performed.
    for k=1:guess
        w=randn(cc,1);   % Random initial guess of w for real numbers
        w=w/norm(w);
        alph=mean(X*w);
        oldw1=w;
        oldw2=oldw1;
        count=0;
        while 1
            count=count+1;
            x=X*w;
            xalph=(x-alph);
            alph=alph + sum(xalph.^3) / (3*sum(xalph.^2)); % Update alpha (alph) value
            mu=alph*w';                % Updata mu, given w and alpha (alph)
            tem=(x-alph).^2;
            dalph_dv=(X'*tem)/sum(tem); % Calculate dalpha/dv
            tem1=X'-dalph_dv*ones(1,r);
```

```
                tem2=X-ones(r,1)*mu;
                Mat1=((ones(cc,1)*tem').*(tem1))*(tem2);
                Mat2=tem1*tem2;
                w=Mat1\(Mat2*w);                % updata w
%% Test convergence
                w=w/norm(w);
                L1=(w'*oldw1)^2;
                if (1-L1) < convlimit
                    convFlag(k,j)={'Converged'};
                    break    % Exit the "while ... end" loop if converging
                elseif count>maxcount
                    convFlag(k,j)={'Not converged'};
                    break    % Exit if reaching the maximum iteration number
                end
%% Continue the interation if "break" criterion is not reached
                L2=(w'*oldw2)^2;
                if L2>L1 && L2>0.95
                    w=w+(rand/5+0.8)*oldw1;
                    w=w/norm(w);
                end
                oldw2=oldw1;
                oldw1=w;
            end
%% Save the projection vectors for different initial guesses
            wall(:,k)=w;
            alphall(1,k)=alph;
        end
%% Take the best projection vector as the solution
        kurtObj(:,j)=( r*sum((X*wall-ones(r,1)*alphall).^4) ./ ( (sum((X*wall-ones(r,1)*alphall).^2)) .^2) )';
        [tem,ind]=min(kurtObj(:,j));
        Wj=wall(:,ind);                 % Take the best projection vector as the solution.
        for i=1:cc
            if Wj(i)~=0;
                signum=sign(Wj(i));       % Change the sign of w to make it unique
                break
            end
        end
        Wj=Wj*signum;
        ALPH(1,j)=alphall(1,ind)*signum;
%% Deflation of matrix
        if strcmpi(VSorth,'VO')         % This deflation method makes the
            t=X*Wj;                     % projection vectors orthogonal.
            T(:,j)=t;
            W(:,j)=Vj*Wj;
            X=X0-X0*W*W';
        elseif strcmpi(VSorth,'SO') % This deflation method makes the scores orthogonal.
            t=X*Wj;          % This follows the deflation method used in the non-linear partial
            T(:,j)=t;        % least squares (NIPALS), which is well-known in chemometrics.
            W(:,j)=Vj*Wj;
            Pj=X'*t/(t'*t);
            P(:,j)=Vj*Pj;
            X=X0-T*P';
        end
    end
%% Transform back into original space
    W=Worig*W;              % The projection vector(s) are tranformed into original space.
    if strcmpi(VSorth,'VO')
        V=W;
        W=[];
        P=[];
        T=T+ones(r,1)*Morig*V;          % Adjust the scores. Mean scores are added.
        R=ALPH*V'+Morig;
    else
        P=Worig*P;          % Vectors in P are tranformed into original space.
        V=W*inv(P'*W);  % Calculate the projection vectors by V=W*inv(P'*W)
        T=T+ones(r,1)*Morig*V;          % Adjust the scores. Mean scores are added.
        R=ALPH*(P'*W)*W'+Morig;
```

```
1379        tem=sqrt(sum(abs(V).^2));
1380        V=V./(ones(size(V,1),1)*tem); % Make the projection vectors be unit length
1381        T=T./(ones(size(T,1),1)*tem); % Adjust T with respect to V
1382        P=P.*(ones(size(P,1),1)*tem); % Adjust P with respect to V
1383    end
1384    end
1385    %% ========================== End of the function ==========================
1386    %%
1387
1388    %% Recentered Multivariate Kurtosis Projection Pursuit Algorithm
1389    function [T,V,R,K,Vall,kurtObj,convFlag]=rcmulkurtpp(X,p,guess)
1390    %
1391    % Algorithms for minimization of re-centered multivariate kurtosis that is
1392    % used as a project pursuit index. This algorithm aims to deal with
1393    % unbalanced clusters (multivariate version). The effect of dimension is
1394    % taken into account by introducing a dimension term in the constraint.
1395    %%
1396    % Input:
1397    %       X:      The data matrix. X cannot be singular.
1398    %       p:      The dimensionality of the plane or heperplane (Normally, 2 or 3).
1399    %       guess:  The number of initial guesses for optimization.
1400    % Output:
1401    %       T:      Scores of the chosen subspace (with the lowest multivariate
1402    %               kurtosis value).
1403    %       V:      Projection vectors for the chosen subspace.
1404    %       R:      The estimated row vector subtracted from the data set X.
1405    %       K:      Multivariate kurtosis value for the chosen subspace.
1406    %       Vall:   All the projection vectors found based on different initial guesses. The
1407    %               best projection vectors are chosen as the solutions and put in V.
1408    %       kurtObj:   Kurtosis values for the projection vectors of different initial guesses.
1409    %       convFlag: Convergence status for the different initial guesses.
1410    %
1411    %%
1412    % This algorithm extends the Quasi-Power methods reported in two papers:
1413    % (1) S. Hou, and P. D. Wentzell, Fast and Simple Methods for the Optimization
1414    %     of Kurtosis Used as a Projection Pursuit Index, Analytica Chimica Acta,
1415    %     704 (2011) 1-15. (featured article)
1416    % (2) S. Hou, and P. D. Wentzell,Re-centered Kurtosis as a Projection Pursuit
1417    %     Index for Multivariate Data Analysis, Journal of Chemometrics, 28
1418    %     (2014) 370-384.   (Special issue article)
1419    %
1420    % Author:
1421    % S. Hou, University of Prince Edward Island, Charlottetown, PEI, Canada, 2014.
1422    %
1423    %% Mean-center the data
1424    [n,m]=size(X);
1425    Morig=mean(X);
1426    X=X-ones(n,1)*Morig;
1427
1428    %% Initial settings
1429    maxcount=10000;
1430    convlimit=1e-10;
1431    Vall=cell(1,guess);
1432    rall=cell(1,guess);
1433    kurtObj=zeros(1,guess);
1434    convFlag=cell(1,guess);
1435
1436    %% Loop
1437    for i=1:guess
1438        count=0;
1439        V=randn(m,p);       % Random initial guess of V
1440        V=orthbasis(V);
1441        oldV1=V;
1442        R=mean(X)';
1443        while 1
1444            count=count+1;
1445
```

```
%% Update r
        Y=(X-ones(n,1)*R'/p)*V;       % Note p is in the denominator
        invPsi=inv(Y'*Y);
        gj=diag(Y*invPsi*Y');
        Yj=Y*invPsi*(sum(Y))';
        J=(2* Y'* ((Yj*ones(1,p)).*Y) * invPsi - eye(p)*(sum(gj)+2))/p;  % Jacobian matrix
        f=sum(Y'.*(ones(p,1)*gj'),2);
        R=R-V*(J\f);                  % Newton' method

%% Update V
% Calculate b1 and b2
        XX=X-ones(n,1)*R';            % Note p is not in the denominator
        Z=XX*V;
        S=Z'*Z;
        invS=inv(S);
        ai=diag(Z*invS*Z');
        Z_ai=(ai*ones(1,p)).*Z;
        Si_ai=Z'*Z_ai;

        b1=-J'\(invS*Si_ai*invS*(sum(Z))');
        b2=-J'\(invS*(sum(Z_ai))');

% Calculate the 8 matrices
        Yj_b1_Yj=(Y*b1*ones(1,p)).*Y;
        Yj_b2_Yj=(Y*b2*ones(1,p)).*Y;
        Xj_gj=sum((gj*ones(1,m)).*X);

        M1=X'*Z*invS*Si_ai;
        M2=-Xj_gj'*b1'*S;
        M3=2*X'*Y*(invPsi*Y'*Yj_b1_Yj*invPsi*S);      % Parentheses added to speed up
        M4=-2*X'*Yj_b1_Yj*invPsi*S;

        M5=(X'.*(ones(m,1)*ai'))*XX;                  % Full rank
        M6=-Xj_gj'*b2'*Z'*XX;                         % Not full rank
        M7=2*X'*Y*(invPsi*Y'*Yj_b2_Yj*invPsi*Z'*XX); % Parentheses added to speed up
        M8=-2*X'*Yj_b2_Yj*invPsi*Z'*XX;

% Calculate new V
        V=(M5+M6+M7+M8)\(M1+M2+M3+M4);
        V=orthbasis(V);

% Test convergence
        L=abs(V)-abs(oldV1);
        L=trace(L'*L);
        if L<convlimit*p
            convFlag(1,i)={'Converged'};
            break
        elseif count>maxcount
            convFlag(1,i)={'Not converged'};
            break
        end
        oldV1=V;
    end

% Save the subspaces for different initial guesses. Note the basis of the
% subspace has been changed in accordance with PCA (mean-centered) criterion.
    kurtObj(1,i)=n*sum(diag(Z*inv(Z'*Z)*Z').^2);
    [Utem,Stem,Vtem]=svd(X*V,'econ'); % X has been mean-centered.
    Vtem=V*Vtem;
    Vall(1,i)={Vtem};
    rall(1,i)={(R'*Vtem*Vtem')}; % r is saved as a row vector now.
end

%% Take the best projection vector as the solution
[tem,ind]=min(kurtObj);
V=Vall{ind};
R=rall{ind};
```

```
1513  T=X*V;
1514  K=kurtObj(ind);
1515
1516  %% Add mean value
1517  T=T+ones(n,1)*Morig*V; % Adjust the scores (The scores of mean vector are added).
1518  R=R+Morig;             % Adjust r (mean vector is added).
1519  end
1520  %% ============== End of the function ====================
1521
1522  function [V]=orthbasis(A)
1523  % Calculate an orthonormal basis for matix A using Gram-Schimdt process
1524  % Reference: David Poole, Linear Algebra - A Modern Introduction,
1525  % Brooks/Cole, 2003. pp.376.
1526  %
1527  % Input:
1528  %   A: a matrix
1529  % Output:
1530  %   V: an orthonormal matrix
1531
1532  %%
1533  c=size(A,2);
1534  V(:,1)=A(:,1)/norm(A(:,1));
1535  for i=2:c
1536      tem=A(:,i)-V*V'*A(:,i);
1537      V(:,i)=tem/norm(tem);
1538  end
1539  end
```

APPENDIX D

MATLAB CODE FOR IMPLEMENTING AUGPPA

```
function [T,ix] = augPPA(X,sub)
% Inputs:
%       X - samples by response variables data matrix
%     sub - size of the class to pull out
%
%Outputs:
%       T - scores of original data
%      ix - index of the sample used to augment
%
% Steve Driscoll 2019
[m,¬]=size(X);
kurts=zeros(1,m);
sub=m-2*sub;
for i=1:m
    Xaug=[X; repmat(X(i,:),[sub 1])];
    [¬,¬,¬,¬,f]=PPA(Xaug,1,1,'Min','Sh','SO');
    kurts(i)=min(f);
    disp(['Augementing with sample ' num2str(i)])
end
[¬,ix]=min(kurts);
Xaug=[X; repmat(X(ix,:),[sub 1])];
[T]=PPA(Xaug,1,100,'Min','Sh','SO');
T=T(1:m,:);
end
```

APPENDIX E

REQUIRED COPYRIGHT REPRODUCTION AGREEMENTS

JOHN WILEY AND SONS LICENSE TERMS AND CONDITIONS

Oct 17, 2019

This Agreement between Dalhousie University -- Stephen Driscoll ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

License Number	4691300589152
License date	Oct 17, 2019
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	Journal of Chemometrics
Licensed Content Title	Simulation of 1/fα noise for analytical measurements
Licensed Content Author	Stephen Driscoll, Michael Dowd, Peter D. Wentzell
Licensed Content Date	Jun 20, 2019
Licensed Content Volume	0
Licensed Content Issue	0
Licensed Content Pages	14
Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley article
Format	Print and electronic
Portion	Full article
Will you be translating?	No
Title of your thesis / dissertation	EXPLORATION OF MULTIVARIATE CHEMICAL DATA IN NOISY ENVIRONMENTS: NEW ALGORITHMS AND SIMULATION METHODS
Expected completion date	Dec 2019
Expected size (number of pages)	210
Requestor Location	Dalhousie University 6299 South St Halifax, NS B3H 4R2 Canada Attn: Dalhousie University
Publisher Tax ID	EU826007151
Total	0.00 USD
Terms and Conditions	

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a"Wiley Company") or handled on behalf of a society with which a Wiley Company has exclusive publishing rights in relation to a particular work (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright

Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

- The materials you have requested permission to reproduce or reuse (the "Wiley Materials") are protected by copyright.

- You are hereby granted a personal, non-exclusive, non-sub licensable (on a stand-alone basis), non-transferable, worldwide, limited license to reproduce the Wiley Materials for the purpose specified in the licensing process. This license, **and any CONTENT (PDF or image file) purchased as part of your order,** is for a one-time use only and limited to any maximum distribution number specified in the license. The first instance of republication or reuse granted by this license must be completed within two years of the date of the grant of this license (although copies prepared before the end date may be distributed thereafter). The Wiley Materials shall not be used in any other manner or for any other purpose, beyond what is granted in the license. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Wiley Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Wiley Material. Any third party content is expressly excluded from this permission.

- With respect to the Wiley Materials, all rights are reserved. Except as expressly granted by the terms of the license, no part of the Wiley Materials may be copied, modified, adapted (except for minor reformatting required by the new Publication), translated, reproduced, transferred or distributed, in any form or by any means, and no derivative works may be made based on the Wiley Materials without the prior permission of the respective copyright owner.**For STM Signatory Publishers clearing permission under the terms of the STM Permissions Guidelines only, the terms of the license are extended to include subsequent editions and for editions in other languages, provided such editions are for the work as a whole in situ and does not involve the separate exploitation of the permitted figures or extracts,** You may not alter, remove or suppress in any manner any copyright, trademark or other notices displayed by the Wiley Materials. You may not license, rent, sell, loan, lease, pledge, offer as security, transfer or assign the Wiley Materials on a stand-alone basis, or any of the rights granted to you hereunder to any other person.

- The Wiley Materials and all of the intellectual property rights therein shall at all times remain the exclusive property of John Wiley & Sons Inc, the Wiley Companies, or their respective licensors, and your interest therein is only that of having possession of and the right to reproduce the Wiley Materials pursuant to Section 2 herein during the continuance of this Agreement. You agree that you own no right, title or interest in or to the Wiley Materials or any of the intellectual property rights therein. You shall have no rights hereunder other than the license as provided for above in Section 2. No right, license or interest to any trademark, trade name, service mark or other branding ("Marks") of WILEY or its licensors is granted hereunder, and you agree that you shall not assert any such right, license or interest with respect thereto

- NEITHER WILEY NOR ITS LICENSORS MAKES ANY WARRANTY OR REPRESENTATION OF ANY KIND TO YOU OR ANY THIRD PARTY, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO THE MATERIALS OR THE ACCURACY OF ANY INFORMATION CONTAINED IN THE MATERIALS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, USABILITY, INTEGRATION OR NON-INFRINGEMENT AND ALL SUCH WARRANTIES ARE HEREBY EXCLUDED BY WILEY AND ITS LICENSORS AND WAIVED BY YOU.

- WILEY shall have the right to terminate this Agreement immediately upon breach of this Agreement by you.

- You shall indemnify, defend and hold harmless WILEY, its Licensors and their respective directors, officers, agents and employees, from and against any actual or threatened claims, demands, causes of action or proceedings arising from any breach of this Agreement by you.

- IN NO EVENT SHALL WILEY OR ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, PROVISIONING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

- Should any provision of this Agreement be held by a court of competent jurisdiction to be illegal, invalid, or unenforceable, that provision shall be deemed amended to achieve as nearly as possible the same economic effect as the original provision, and the legality, validity and enforceability of the remaining provisions of this Agreement shall not be affected or impaired thereby.

- The failure of either party to enforce any term or condition of this Agreement shall not constitute a waiver of either party's right to enforce each and every term and condition of this Agreement. No breach under this agreement shall be deemed waived or excused by either party unless such waiver or consent is in writing signed by the party granting such waiver or consent. The waiver by or consent of a party to a breach of any provision of this Agreement shall not operate or be construed as a waiver of or consent to any other or subsequent breach by such other party.

- This Agreement may not be assigned (including by operation of law or otherwise) by you without WILEY's prior written consent.

- Any fee required for this permission shall be non-refundable after thirty (30) days from receipt by the CCC.
- These terms and conditions together with CCC's Billing and Payment terms and conditions (which are incorporated herein) form the entire agreement between you and WILEY concerning this licensing transaction and (in the absence of fraud) supersedes all prior agreements and representations of the parties, oral or written. This Agreement may not be amended except in writing signed by both parties. This Agreement shall be binding upon and inure to the benefit of the parties' successors, legal representatives, and authorized assigns.
- In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall prevail.
- WILEY expressly reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.
- This Agreement will be void if the Type of Use, Format, Circulation, or Requestor Type was misrepresented during the licensing process.
- This Agreement shall be governed by and construed in accordance with the laws of the State of New York, USA, without regards to such state's conflict of law rules. Any legal action, suit or proceeding arising out of or relating to these Terms and Conditions or the breach thereof shall be instituted in a court of competent jurisdiction in New York County in the State of New York in the United States of America and each party hereby consents and submits to the personal jurisdiction of such court, waives any objection to venue in such court and consents to service of process by registered or certified mail, return receipt requested, at the last known address of such party.

WILEY OPEN ACCESS TERMS AND CONDITIONS

Wiley Publishes Open Access Articles in fully Open Access Journals and in Subscription journals offering Online Open. Although most of the fully Open Access journals publish open access articles under the terms of the Creative Commons Attribution (CC BY) License only, the subscription journals and a few of the Open Access Journals offer a choice of Creative Commons Licenses. The license type is clearly identified on the article.

The Creative Commons Attribution License

The Creative Commons Attribution License (CC-BY) allows users to copy, distribute and transmit an article, adapt the article and make commercial use of the article. The CC-BY license permits commercial and non-

Creative Commons Attribution Non-Commercial License

The Creative Commons Attribution Non-Commercial (CC-BY-NC)License permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.(see below)

Creative Commons Attribution-Non-Commercial-NoDerivs License

The Creative Commons Attribution Non-Commercial-NoDerivs License (CC-BY-NC-ND)

Sparse Projection Pursuit Analysis: An Alternative for Exploring Multivariate Chemical Data

Author: Stephen P. Driscoll, Yannick S. MacMillan, Peter D. Wentzell

Publication: Analytical Chemistry

Publisher: American Chemical Society

Date: Dec 1, 2019

PERMISSION/LICENSE IS GRANTED FOR YOUR ORDER AT NO CHARGE

This type of permission/license, instead of the standard Terms & Conditions, is sent to you because no fee is being charged for your order. Please note the following:

- Permission is granted for your request in both print and electronic formats, and translations.
- If figures and/or tables were requested, they may be adapted or used in part.
- Please print this page for your records and send a copy of it to your publisher/graduate school.
- Appropriate credit for the requested material should be given as follows: "Reprinted (adapted) with permission from (COMPLETE REFERENCE CITATION). Copyright (YEAR) American Chemical Society." Insert appropriate information in place of the capitalized words.
- One-time permission is granted only for the use specified in your request. No additional uses are granted (such as derivative works or other editions). For any other uses, please submit a new request.

BACK CLOSE WINDOW