Detecting Anomalies in Underwater Ambient Noise Using ARMA Models

Yngve Åm

Thesis submitted for the degree of
Master of Science in
Electronics and Computer Technology
60 credits

Department of Physics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

January 2017

Abstract

This thesis explores two alternative methods for detecting anomalies in underwater ambient noise. Both methods are based on ARMA modeling. The anomalies in question are unknown sound sources such as ships or biological organisms that mix with the ambient noise and alter the overall characteristics of the received acoustic signal. The performance of the two methods is tested on simulated and real data sets, and is compared to that of an energy detector.

Anomaly detection is done either by successively estimating ARMA parameters and calculating Mahalanobis distances to a baseline model, or by computing prediction errors from the model and testing the errors for serial correlation. Both methods give rise to a chi-squared test statistic.

Results show that the ARMA methods perform better than an energy detector. The energy detector itself is observed to perform significantly better when differencing is applied to the signal before mean square values are calculated.

Of the two ARMA methods, the prediction method is found to be the better one. It is both able to detect fainter anomalies than the successive estimation method and is found to have more desirable statistical properties.

The two ARMA methods are viewed as different approaches to feature extraction in an overall classification framework where the ambient noise represents the normal class and anomalies such as ships are examples of an anomaly class.

A supervised method for reducing the dimensionality of the feature space is proposed that can be seen as a combination of principal component analysis and linear discriminant analysis. Experiments on simulated and real data sets appear promising for supervision in general but further study is needed to determine if the proposed algorithm has any real benefit over regular linear discriminant analysis.

During the course of the work, a MATLAB program has been developed that implements all the methods discussed in the thesis. Output figures from the program are used throughout, and the most important functions are listed in the Appendix.

Acknowledgements

This work would not have been possible without the help of my two excellent supervisors. My university supervisor, Andreas Austeng at the Department of Informatics, helped me set up a 10-credit special curriculum in advanced statistical signal processing. Discussing this curriculum with him and others from the Digital Signal Processing and Image Analysis group was an ideal preparation for my thesis work. Since the outset, he has offered invaluable advice on methodology and academic writing.

My external supervisor, Lars Ødegaard at the Norwegian Defence Research Establishment provided the original project proposal for the thesis. His feedback has encouraged me to keep looking for new ways to improve my solution. As a researcher in the field of underwater acoustic signal processing, he has given me several important course corrections along the way which have helped increase the relevance of my work.

I would like to thank both my supervisors for all their help, as well as their patience, since this work has been carried out on a part-time basis over a longer period than what is usual for a master's thesis.

I would also like to thank my friends and family for their support and encouragement throughout this period. I am looking forward to spending more time with you all now that the work is done.

Yngve Åm
Oslo, January 2017

Contents

1 Introduction

In passive sonar, underwater microphones, or *hydrophones*, are used to record acoustic signals. Unlike active sonar, passive sonar does not rely on emitting sound waves to its surroundings. This makes it a quiet way of monitoring ocean activity and detecting objects like ships, submarines or biological organisms.

When passive sonar is used for detection, the processing performed on the acoustic signal is often designed to look for specific acoustic signatures of known objects. But when the goal is to detect unknown objects with an unknown acoustic signature, such an approach is not feasible.

The work documented in this thesis has been carried out in response to a project proposal written by Lars Ødegaard at the Norwegian Defence Research Establishment (FFI). In it he suggested that detection of unknown objects could be done by building a statistical model of the underwater ambient noise, and performing tests on new data designed to detect deviations from this model. More specifically, the suggested model was the *autoregressive moving average (ARMA)* model, which is a fundamental tool in time series analysis and statistical signal processing. The project proposal did not say how detection should be performed, but cited some examples of research in various fields where ARMA models had been used to detect changes from a normal state.

In this thesis, some possible ways of using ARMA models to detect anomalies in ambient noise will be examined. Hydrophone data from the Lofoten-Vesterålen Ocean Observatory (LoVe) has been provided by FFI. This data will be used to build a model of the ambient noise and to test the detection methods.

1.1 Anomaly Detection Using ARMA Models

When the phrase "anomaly detection" is used in the context of time series analysis, it often refers to finding individual points in a time series that can be classified as outliers or anomalies. See for example Louni (2005) and the references therein. These methods are not particularly interesting in the context of anomaly detection as it is defined here. The reason is that the acoustic signals from hydrophones typically consist of thousands of samples per second. So instead of looking for single samples that do not conform with our model, it is more desirable to work on whole segments of data and try to find ways of characterizing the segments that are sensitive to changes in the underlying system.

In other fields, ARMA models have been used to characterize sensor data this way. For instance, in the field of structural health monitoring, Gul and Catbas (2009) estimated AR parameters for segments of accelerometer data and used statistical analysis of the parameter vectors to find outliers corresponding to damaged structures. A similar approach was used by Bao et al. (2012) to detect damages in subsea pipeline systems. Both methods relied on scoring the

parameter vectors by their Mahalanobis distance (see Section 2.1.1) or some function of this distance.

ARMA models have also received some attention in the field of EEG analysis. Subasi (2005) investigated AR modeling as an approach to detecting epileptiform discharges. Similarly, Faust et al. (2007) explored AR modeling in the context of detecting epileptic and alcoholic states. In both articles, the AR models were used to obtain estimates of the power spectral density (see Section 2.2.1) of the underlying process.

The above methods are examples of what will be referred to as detection through successive estimation. In this approach, segments of data are obtained by some measurement process and an AR or ARMA model is fitted to each segment. Detection is done by using some function of the parameter vector to characterize the state of the underlying system and label it as either normal or abnormal.

As explained in Aggarwal (2013, pp. 225–232), another way to detect anomalies using ARMA models is to first fit a model to data from the normal state and subsequently compare new data with predictions from this model. This has for instance been used to detect anomalies in communication networks (Pincombe, 2005). When such an approach is taken, one can make use of the fact that the prediction errors, or *residuals*, of a true ARMA model will be normally distributed during the normal state of the system. Consequently, one can use the normal distribution to detect outliers among individual residuals or aggregates of residuals.

When searching for methods that rely on tests being performed on whole data segments using a prediction-based approach, I have not been able to find much literature. For the most part, it seems like prediction is used to detect individual outlier points in the data.

However, in time series analysis, a common way of checking the validity of a fitted ARMA model is to perform a hypothesis test on the model residuals called the Ljung-Box test (Box et al., 2008). This test takes the data that was used to fit the model and computes the residuals from the model. It then looks for serial correlations in the residuals to determine if the ARMA model fits the data. It is not unnatural, then, to ask if a test such as the Ljung-Box test can be used to look for lack of fit in new data and thus be employed to detect deviations from a normal state.

The two methods I have chosen to focus on in this thesis, are based on successive estimation and prediction. The successive estimation method can be seen as a simplified version of the approach in the structural health monitoring articles referred to above.

Unlike in those articles, no preprocessing such as normalization will be performed on the time series data. Additionally, p-values will be calculated directly from the squared Mahalanobis distance instead of some other function of the distance that is assumed to be normally distributed.

The prediction method is an adaptation of the Ljung-Box test, where the only difference is the degrees of freedom used for the distribution of the test statistic.

1.2 Thesis Outline

Chapter 2 deals with the theory of multivariate normal (MVN) variables and ARMA modeling of stochastic processes that will be used in Chapter 3 to perform anomaly detection. Special attention is given to whitening transformations and prediction errors, as these two concepts are fundamental to the methods examined in subsequent chapters.

Chapter 3 starts by specifying the hypothesis tests that the detection methods are based on, and goes on to describe the general approach taken in this thesis to designing and testing detectors. Next, an ARMA model is built from hydrophone data and subsequently used to simulate ambient noise in order to test the performance of the detection methods on sinusoids in noise. Finally, an algorithm for supervised dimensionality reduction is introduced and demonstrated on simulated data. The algorithm may very well have been proposed by others in the past, but since I haven't found evidence of that in literature, I introduce it as my own generalization of Fisher's linear discriminant (see Section 2.1.3).

In Chapter 4, experiments are conducted on a real hydrophone data set containing both ambient noise at different wind speeds, ship noise and whale sounds. As in the previous chapter, the detection methods are compared to each other and to an energy detector. In some of the experiments, simulated noise is added to the data set in order to create a more challenging situation for the detectors and test their performance on fainter anomalies.

1.3 Notes on Bibliography

Much of what is covered in the theory chapter and used throughout the thesis is material found in typical textbooks in statistical signal processing, time series analysis and pattern recognition. But readers coming from different fields may find some topics more obscure than others.

For general theory on statistical signal processing, Scharf (1991) and Hayes (1999) have been consulted extensively. For theory on spectral estimation, Stoica and Moses (2005) has been a valuable companion. The latest edition of the classic work by Box et al. (2008) has been the primary reference for time series analysis. Duda et al. (2001) has been the principal source for general classification theory and Aggarwal (2013) for the theory of anomaly detection.

2 Theoretical Background

The following is a summary of the theoretical foundations of the methods examined in this thesis. The reader is expected to be familiar with the material covered by a typical undergraduate course in each of the following subjects: linear algebra, multivariate calculus, univariate statistics and digital signal processing. For readers who are also familiar with statistical signal processing, time series analysis and multivariate statistics, this chapter will serve as a review of the relevant material, as well as provide some motivation for the methods presented later on.

2.1 Multivariate Normal Variables

In multivariate statistics, the ideas and methods of univariate statistics are generalized to vectors made up of several random variables. By employing the language of linear algebra, one is able to represent these generalized methods and ideas in a compact form, useful both for analysis and software implementation. Consequently, many general results and techniques from linear algebra can be readily applied to solve statistical problems. In addition, the geometric perspective offered by linear algebra notation can sometimes contribute significantly to our understanding of the problems at hand.

2.1.1 The Multivariate Normal Distribution

A vector $\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_d]^{\mathrm{T}}$ where all the elements are random variables, is known as a *random vector*.[1] When all the elements of the vector are normally distributed, their joint probability distribution is called a *multivariate normal distribution*. Like its univariate analog, the MVN distribution is defined by two parameters – the mean vector and the covariance matrix:

$$\begin{aligned} \mathbf{x} &\sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \boldsymbol{\mu} &= E[\mathbf{x}] = [\mu_1 \; \mu_2 \; \cdots \; \mu_d]^{\mathrm{T}} \\ \boldsymbol{\Sigma} &= E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}] = \{\sigma_{ij}\} \\ \sigma_{ij} &= E[(x_i - \mu_i)(x_j - \mu_j)] \end{aligned} \tag{2.1}$$

The probability density function (PDF) of the MVN distribution is given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\Big], \tag{2.2}$$

[1] The notation in this thesis does, for the most part, not distinguish between a random variable and a realization of that variable. Often, upper case letters would be used for the variable itself while lower case letters would be used for the realization. But to avoid confusion between random variables and matrices, boldface capital letters will only be used to denote matrices.

where $|\mathbf{\Sigma}|$ is the determinant of the covariance matrix. This function assigns a probability density (a scalar value) to a point $\mathbf{x}$ in $\mathbb{R}^d$. For $d = 1$ it reduces to the familiar PDF of the univariate normal distribution.

The multivariate normal distribution has been widely studied as it is a convenient way to represent sets of correlated normal random variables. The treatment here will be limited to properties of the distribution that are needed for this thesis. More details can be found in Scharf (1991) and Duda et al. (2001).

The quadratic form in the exponent of the PDF is known as the squared *Mahalanobis distance* from the point $\mathbf{x}$ to the distribution mean $\boldsymbol{\mu}$, or simply the squared Mahalanobis distance of $\mathbf{x}$:

$$r^2(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{2.3}$$

By substituting this into the equation for the PDF, we see that the density is just an exponential function of the squared Mahalanobis distance, multiplied by a constant:

$$f(\mathbf{x}) = k e^{-\frac{1}{2} r^2(\mathbf{x})} \tag{2.4}$$

It can be shown that the scaling constant k ensures that the integral over $\mathbb{R}^d$ is equal to 1, which is necessary for the function to be a probability density.

The equation $r = c$, for some constant c, defines a level curve of constant density for the PDF. In general, this will be a *hyperellipsoid* in $\mathbb{R}^d$, which is a generalization of the ellipse to higher dimensions. The MVN distribution is therefore known as an *elliptical distribution*. The semi-axes $\boldsymbol{a}_i$ of the hyperellipsoid can be found from the eigenvectors and eigenvalues of the covariance matrix:

$$\begin{aligned} \mathbf{\Sigma} &= \mathbf{V} \mathbf{\Lambda} \mathbf{V}^{\mathrm{T}} \\ \mathbf{V} &= [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_d] \\ \mathbf{\Lambda} &= \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d) \\ \boldsymbol{a}_i &= r \sqrt{\lambda_i} \mathbf{v}_i \end{aligned} \tag{2.5}$$

Figure 2.1 shows the PDF of a two-dimensional MVN distribution as a surface in three dimensions. An ellipse of constant density is shown below the surface, along with its two semi-axes. In addition, 400 points have been drawn from the distribution and plotted in the same plane as the ellipse. The plane has been shifted down on the z-axis to make it more visible. The parameters of the distribution are

$$\boldsymbol{\mu} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} 0.7 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}. \tag{2.6}$$

Figure 2.2 shows a similar plot of the multivariate standard normal distribution, *i.e.* $\mathbf{x} \sim N(\mathbf{0}, \mathbf{I})$, which means that the vector elements are independent standard normal variables. Note that in this special case, the semi-axes are equal in length and line up with the coordinate axes. The level curves then become circles with radius r centered at the origin. For higher dimensions, the hyperellipsoids reduce to hyperspheres in $\mathbb{R}^d$. The multivariate standard normal distribution is therefore known as a *spherical distribution*.

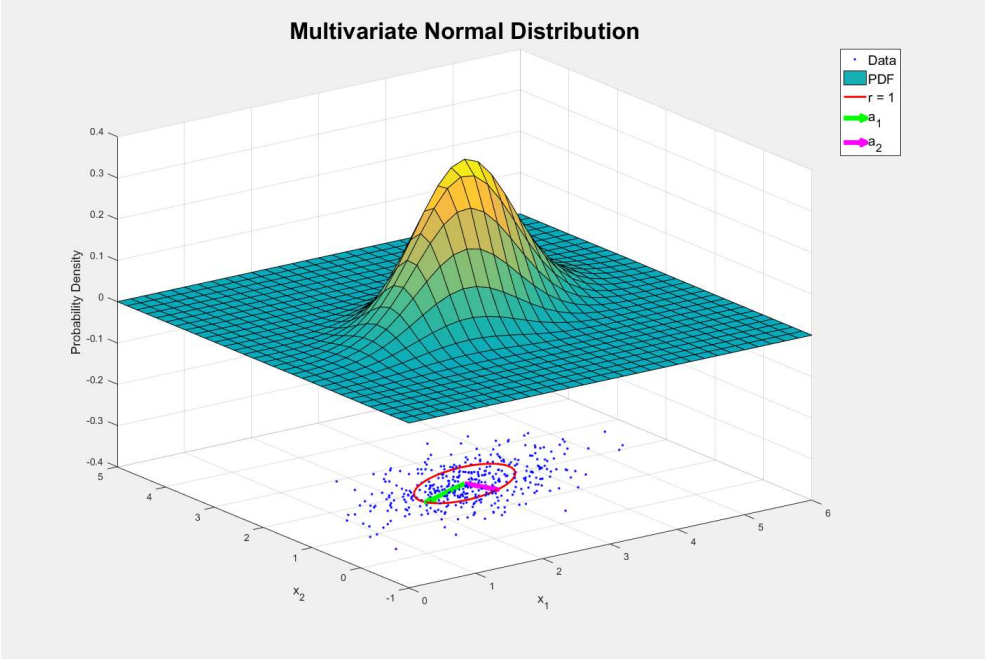

Figure 2.1: A two-dimensional multivariate normal distribution.

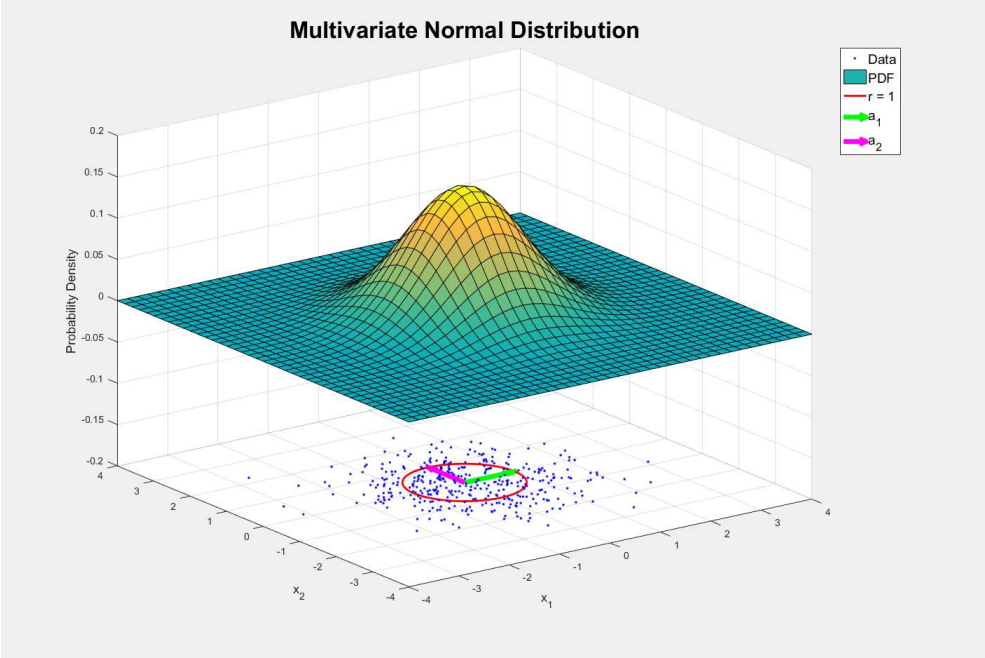

Figure 2.2: The two-dimensional multivariate standard normal distribution.

Distribution of the Squared Mahalanobis Distance

To further understand the Mahalanobis distance, we will need to use some properties of linear transformations of MVN random variables. When such transformations are formed using a full rank matrix $\mathbf{A} \in \mathbb{R}^{dxd}$, the resulting variable will be MVN with parameters as given below:

$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma});\ \mathbf{y} = \mathbf{A}\mathbf{x}\ \Rightarrow \mathbf{y} \sim N(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}}) \tag{2.7}$$

An important feature of the Mahalanobis distance is that it can be seen as the Euclidian norm of a standard normal random vector (Wicklin, 2012). This fact allows us to derive the distribution of r^2. Given a matrix $\mathbf{A} = \boldsymbol{\Sigma}^{-1/2}$ such that $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \boldsymbol{\Sigma}^{-1}$, the squared Mahalanobis distance becomes

$$r^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}(\mathbf{x} - \boldsymbol{\mu}) = \|\mathbf{A}(\mathbf{x} - \boldsymbol{\mu})\|^2 = \|\mathbf{z}\|^2. \tag{2.8}$$

That $\mathbf{z}$ is standard normal follows from the properties of linear transformations given above:

$$\begin{aligned} \mathbf{z} &= \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}) \\ E[\mathbf{z}] &= \mathbf{A}E[\mathbf{x} - \boldsymbol{\mu}] = \mathbf{A}(\boldsymbol{\mu} - \boldsymbol{\mu}) = \mathbf{0} \\ E[\mathbf{z}\mathbf{z}^{\mathrm{T}}] &= \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}} = (\boldsymbol{\Sigma}^{-1/2})\boldsymbol{\Sigma}(\boldsymbol{\Sigma}^{-1/2})^{\mathrm{T}} = \mathbf{I} \\ &\Rightarrow \mathbf{z} \sim N(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{2.9}$$

Since the squared norm of a standard normal vector is just a sum of squared independent standard normal variables, r^2 will be chi-squared distributed with d degrees of freedom:

$$r^2 = \|\mathbf{z}\|^2 = \sum_{i=1}^{d} z_i^2 \Rightarrow r^2 \sim \chi_d^2 \tag{2.10}$$

The chi-squared probability density function is given by

$$f(x) = \frac{1}{\Gamma(L/2)2^{L/2}} x^{(L/2)-1} e^{-x/2}; \quad x \geq 0, \tag{2.11}$$

where $\Gamma(\cdot)$ is the gamma function. The probability of observing a point $\mathbf{x}$ with Mahalanobis distance c or higher is thus

$$P(r \geq c) = \int_{c^2}^{\infty} f(x)\, dx = 1 - F(c^2), \tag{2.12}$$

where $F(\cdot)$ is the cumulative density function (CDF) of the chi-squared distribution.

Whitening Transformations

We arrived at the distribution of r^2 by assuming that we could find a matrix $\mathbf{A}$ such that $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{\Sigma}^{-1}$. In fact, there will be an infinite number of matrices $\mathbf{A}$ that satisfy this condition (Kessy et al., 2016). Such matrices can be used to transform a normal random vector into a standard normal one. This is often referred to as a *whitening transformation*, as it transforms the vector into an uncorrelated white noise vector (see Section 2.2.1).

Note that since a whitening transformation is easily reversible, no information is lost when a random vector is decorrelated. It is just a means of representing the vector in a way that can be useful under certain circumstances. When the covariance matrix is known, a common choice for $\mathbf{A}$ is found from the Cholesky factors of $\mathbf{\Sigma}$:

$$\begin{gathered} \mathbf{\Sigma} = \mathbf{L}\mathbf{L}^{\mathrm{T}} \\ \mathbf{\Sigma}^{-1} = (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} = (\mathbf{L}^{\mathrm{T}})^{-1}\mathbf{L}^{-1} = (\mathbf{L}^{-1})^{\mathrm{T}}\mathbf{L}^{-1} \\ \Rightarrow \mathbf{A} = \mathbf{L}^{-1} \end{gathered} \tag{2.13}$$

More on Cholesky factorization and its applications in signal processing can be found in Scharf (1991).

In practice, the covariance matrix is rarely known. However, given observations drawn from the distribution, we can get an estimate $\widehat{\mathbf{\Sigma}}$ of the true covariance matrix. Whitening transformations can then be constructed from this

estimate, or directly from the matrix of observations. In the next section, we will see an example of the latter, where a whitening transformation is constructed from the singular value decomposition (SVD) of the data matrix.

2.1.2 Decorrelation by SVD

The singular value decomposition of an N by p matrix $\mathbf{X}$, where $N \geq p$, is a factorization given by

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{2.14}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{S}$ is a diagonal matrix. The SVD is widely used in statistics and has many interesting properties not covered here. Details on the SVD can be found in Scharf (1991).

Given N observations of a normal random vector, an unbiased estimator for the covariance matrix is the *sample covariance matrix*

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathrm{T}}, \tag{2.15}$$

where

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{2.16}$$

is the *sample mean*. If the observations are organized as the rows of a *data matrix*

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{Nxd}; \quad N > d \tag{2.17}$$

and we form a matrix of repeated sample means,

$$\mathbf{M} = \begin{bmatrix} \hat{\boldsymbol{\mu}}^{\mathrm{T}} \\ \hat{\boldsymbol{\mu}}^{\mathrm{T}} \\ \vdots \\ \hat{\boldsymbol{\mu}}^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{Nxd}, \tag{2.18}$$

Eq. (2.15) can be rewritten as

$$\begin{aligned} \widehat{\boldsymbol{\Sigma}} &= \frac{1}{N-1}(\mathbf{X} - \mathbf{M})^{\mathrm{T}}(\mathbf{X} - \mathbf{M}) \\ \widehat{\boldsymbol{\Sigma}} &= \frac{1}{N-1}\mathbf{X}_c^{\mathrm{T}}\mathbf{X}_c \end{aligned} \tag{2.19}$$

where $\mathbf{X}_c = \mathbf{X} - \mathbf{M}$ is called the centered data matrix.

Recall that a matrix $\mathbf{A}$ such that $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \boldsymbol{\Sigma}^{-1}$ can be used to decorrelate a normal vector. Start by expressing $\widehat{\boldsymbol{\Sigma}}^{-1}$ in terms of the data matrix as

$$\widehat{\boldsymbol{\Sigma}}^{-1} = (N-1)(\mathbf{X}_c^{\mathrm{T}}\mathbf{X}_c)^{-1}. \tag{2.20}$$

Now, let

$$\mathbf{X}_c = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{2.21}$$

be the SVD of the centered data matrix. Then

$$\begin{aligned} \mathbf{X}_c^{\mathrm{T}}\mathbf{X}_c &= \mathbf{V}\mathbf{S}^2\mathbf{V}^{\mathrm{T}} \\ (\mathbf{X}_c^{\mathrm{T}}\mathbf{X}_c)^{-1} &= (\mathbf{V}^{\mathrm{T}})^{-1}\mathbf{S}^{-2}\mathbf{V}^{-1} \\ (\mathbf{X}_c^{\mathrm{T}}\mathbf{X}_c)^{-1} &= (\mathbf{V}\mathbf{S}^{-1})(\mathbf{S}^{-1}\mathbf{V}^{\mathrm{T}}), \end{aligned} \tag{2.22}$$

since $\mathbf{V}^{\mathrm{T}} = \mathbf{V}^{-1}$. The inverse sample covariance matrix is thus

$$\widehat{\boldsymbol{\Sigma}}^{-1} = (N-1)(\mathbf{V}\mathbf{S}^{-1})(\mathbf{S}^{-1}\mathbf{V}^{\mathrm{T}}) \tag{2.23}$$

To absorb the factor $(N-1)$, form a new diagonal matrix

$$\mathbf{D} = \frac{\mathbf{S}}{\sqrt{N-1}}. \tag{2.24}$$

and define $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{D}^{-1}\mathbf{V}^{\mathrm{T}}. \tag{2.25}$$

Then it follows from (2.23) and (2.25) that

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \widehat{\boldsymbol{\Sigma}}^{-1}. \tag{2.26}$$

This means that the linear transformation $\mathbf{y} = \mathbf{A}(\mathbf{x} - \hat{\boldsymbol{\mu}})$ is an estimate obtained from data of the "true" whitening transformation $\mathbf{y} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x} - \boldsymbol{\mu})$ that could be formed if $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ were known parameters.

Note from (2.22) and (2.5) that $\mathbf{V}$ is in fact the matrix of eigenvectors of the sample covariance matrix, while $\mathbf{D}^2$ is the diagonal matrix of corresponding eigenvalues. Another way of obtaining the transformation is therefore to do an eigenvalue decomposition of $\widehat{\boldsymbol{\Sigma}}$.

Since the transformation is an estimate, the quality of this estimate depends on the amount of data used. In particular, it depends on the ratio N/d. Since we

are essentially estimating $d+1$ parameters ($\hat{\boldsymbol{\mu}}$ and the rows of $\mathbf{A}$) from N data points (the rows of $\mathbf{X}$), it is generally desirable to have $N \gg d$.

To perform the transformation on all the vectors in a data matrix, define it in terms of the transpose, so that

$$\mathbf{y}^{\mathrm{T}} = (\mathbf{x} - \hat{\boldsymbol{\mu}})^{\mathrm{T}}\mathbf{V}\mathbf{D}^{-1}. \tag{2.27}$$

This leads to

$$\mathbf{Y} = (\mathbf{X} - \mathbf{M})\mathbf{V}\mathbf{D}^{-1}, \tag{2.28}$$

where $\mathbf{M}$ is made to have the same number of rows as $\mathbf{X}$. After the transformation has been created and $\mathbf{A}^{\mathrm{T}} = \mathbf{V}\mathbf{D}^{-1}$ and $\hat{\boldsymbol{\mu}}$ have been stored, it is therefore a simple task to perform it on another data matrix later on.

An instructive way to view (2.28), is as a coordinate transformation, or change of basis. Multiplying by $\mathbf{V}$ effectively represents the data in a coordinate system where the eigenvectors of $\widehat{\boldsymbol{\Sigma}}$ are the coordinate axes. Multiplication by $\mathbf{D}^{-1}$ then scales the variance along each of these axes to unity.

The ordered structure of $\mathbf{V}$ and $\mathbf{D}$ further ensures that the first element of the transformed vectors will be coordinates along the first eigenvector, and so on. This means that the n first columns of $\mathbf{Y}$ represent normalized coordinates along the n first eigenvectors of $\widehat{\boldsymbol{\Sigma}}$. By selecting these and removing the rest, we will have represented the data in terms of n vectors or *components* along which the spread of the data is the highest. This is known as doing a *principal component analysis (PCA)* of the data. It can be understood as projecting the data onto an n-dimensional subspace of $\mathbb{R}^d$ spanned by the most descriptive eigenvectors of $\widehat{\boldsymbol{\Sigma}}$. More on PCA can be found in Shlen (2003).

2.1.3 Dimensionality Reduction in Classification Problems

In classification theory, the general problem consists of designing a *classifier* that turns observations into decisions. The observations come in the form of random vectors whose elements are called *features*. These features are the result of some kind of processing done on raw data. Given a feature vector, the classifier needs to determine which among a discrete set of classes the vector comes from. The classes may represent types of real-world objects or something more abstract. In many situations, the feature vectors from each class can be modeled as multivariate normal variables.

The success of any classifier depends heavily on the features it uses to distinguish between classes. Feature selection is therefore an important part of building a classification system. After some extraction algorithm has been implemented to produce a set of features from raw data, one cannot simply assume that they are all equally useful for discriminating between classes. One might find that some features can be discarded altogether, while others may be strongly correlated and can be combined into a single feature.

A classical approach to dimensionality reduction in classification is performing a principal component analysis on the data from one or multiple classes. This amounts to forming linear combinations of the original features that represent coordinates along the directions in which the data varies the most. A problem with PCA in the context of classification is that, while the combination of features it forms are optimal for *describing* the data under the assumption of normality, they may not be the best for *discriminating* between different classes (Duda et al., 2001).

Another classical method for dimensionality reduction is known as *linear discriminant analysis (LDA)*. Here, the focus is on finding linear combinations of the features that best separate the data from different classes. In the two-class problem, one seeks to find a vector $\mathbf{w}$, formed so that when the data is projected onto it by $\mathbf{w}^{\mathrm{T}}\mathbf{x}$, the difference in the sample means is large relative to the combined sample variance of the classes. In other words, one wants the means to be far apart and the sum of the variances to be small for the projected data. This can be expressed by the following criterion:

$$J(\mathbf{w}) = \frac{[\mathbf{w}^{\mathrm{T}}(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)]^2}{\mathbf{w}^{\mathrm{T}}\left(\widehat{\boldsymbol{\Sigma}}_1 + \widehat{\boldsymbol{\Sigma}}_2\right)\mathbf{w}}, \tag{2.29}$$

where the subscripts refer to the two classes. $J(\mathbf{w})$ can be understood as a signal-to-noise ratio (SNR) where the difference in means is measured against the noise contributed by the combined variance of the two classes. It can be shown that the vector that maximizes the SNR is given by

$$\mathbf{h} = \arg\max_{\mathbf{w}} J(\mathbf{w}) = \left(\widehat{\boldsymbol{\Sigma}}_1 + \widehat{\boldsymbol{\Sigma}}_2\right)^{-1}(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1). \tag{2.30}$$

This is called *Fisher's linear discriminant (FLD)*. Details on the derivation of the discriminant can be found in Duda et al. (2001, pp. 117–121).

The fact that LDA uses labeled data from both classes to form the projection, makes it a *supervised learning* technique. If such labeled data is available, it is generally a good idea to make use of the information it contains when designing a classifier, even if the goal is just to detect anomalies (Aggarwal, 2013, p. 169).

An advantage of PCA over LDA, however, is the flexibility one has in choosing the dimension of the lower-dimensional subspace that the data is projected into. In LDA with two classes one is forced to work in just one dimension, although this one-dimensional subspace is optimal in the sense just described.

In this thesis, the general problem of detecting anomalies in underwater ambient noise is framed as a classification problem with two classes, where ambient noise represents the normal class and anomalies in the form of ship noise or biological sounds is grouped together as an anomaly class. The remainder of this chapter is devoted to the feature extraction approach that will be used to

turn the raw data of sonar time series into useful features for classification – namely ARMA modeling.

2.2 ARMA Modeling of Stochastic Processes

In the following sections, the general concept of ARMA modeling in time series analysis and statistical signal processing is introduced. Special attention is paid to prediction errors and the view of ARMA modeling as a means of decorrelating a stochastic process. Next, some common algorithms for ARMA parameter estimation are presented, as well as some ways of selecting the order of an ARMA model.

2.2.1 Stochastic Processes

An indexed set of random variables is known as a *stochastic process*. Discrete-time signals that cannot be explained by a deterministic model, are commonly modeled as the realization of a stochastic process. Just as there may be an infinite number of possible realizations of a random variable, a stochastic process may have an infinite number of possible realizations.

An important property of a stochastic process x_t is its *autocorrelation function (ACF)*, defined as $r_{t,l} = E[x_t x_l]$. It is a deterministic function that describes the degree of correlation between variables of the process at different points in time.

A discrete-time stochastic process is said to be *stationary* if all the statistical properties of the random variables stay the same over time. This is, however, a very strict requirement. More commonly, the models used to describe discrete-time signals, or *time series*, assume only that the process is *wide sense stationary (WSS)*. A stochastic process is said to be WSS if the mean and variance are constant and the ACF only depends on the difference $k = t - l$. The last condition is another way of saying that the correlation between the variables of the process is not a function of time, but only of how far apart they are in time. If a WSS process has non-zero mean, it can easily be transformed to a zero-mean process by simply subtracting the mean. It will therefore be assumed that any WSS process referred to here has zero mean.

In a similar way that a deterministic signal can be represented in the frequency domain via the discrete-time Fourier transform (DTFT), the *power spectral density (PSD)* or *power spectrum* provides a frequency domain representation of a WSS stochastic process. It is defined as the DTFT of the autocorrelation function:

$$P(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_k e^{-jk\omega} \tag{2.31}$$

The PSD is a continuous, real-valued function of the radian frequency ω that describes how the power of the process is distributed across frequencies. Since it

is a density, the total power in a specific frequency band is proportional to the integral over that band.

A stochastic process whose autocorrelation function is zero for all lags except lag zero, *i.e.* $r_k = \sigma^2 \delta_k$, is known as *white noise* with variance σ^2. When the variables of a white noise process are normally distributed, it is referred to as *Gaussian white noise*. Note that if we compute the power spectrum of a white noise process using Eq. (2.31), it will yield a constant value of σ^2 for all frequencies. This means that the power is evenly distributed across all frequencies.

2.2.2 The ARMA Model

A common way to model WSS processes is the autoregressive moving average model, where the process y_t is assumed to be governed by the difference equation

$$\begin{gathered} \sum_{n=0}^{p} a_n y_{t-n} = \sum_{n=0}^{q} b_n u_{t-n}\,; \qquad a_0, b_0 = 1 \\ u_t \sim N(0, \sigma^2); \qquad E[u_t u_l] = \sigma^2 \delta_{t-l}. \end{gathered} \tag{2.32}$$

Here, $\{a_n\}$ are the autoregressive (AR) coefficients and $\{b_n\}$ are the moving average (MA) coefficients, while u_t is Gaussian white noise with variance σ^2. A model with p AR coefficients and q MA coefficients is referred to as ARMA(p, q). Eq. (2.32) can be rewritten as

$$y_t = -\sum_{n=1}^{p} a_n y_{t-n} + \sum_{n=1}^{q} b_n u_{t-n} + u_t \tag{2.33}$$

This shows that each value of y_t is a linear combination of past values of y_t and u_t. Since u_t is assumed to be normally distributed, this implies that y_t will also be normal, since it is a linear combination of normal random variables.

In (2.33) y_t can be seen as the output of a causal linear shift-invariant filter with a rational transfer function, where the input is white noise:

$$\{y_t\} = \frac{B(z)}{A(z)}\{u_t\}, \tag{2.34}$$

where

$$\begin{aligned} B(z) &= 1 + \sum_{n=1}^{q} b_n z^{-k} \\ A(z) &= 1 + \sum_{n=1}^{p} a_n z^{-k} \end{aligned} \tag{2.35}$$

For the ARMA process to be WSS, the filter is required to be stable. This implies that all the poles and zeros, *i.e.* the roots of the polynomials defined by

$\{a_n\}$ and $\{b_n\}$, must lie inside the unit circle in the complex plane. Furthermore, for the ARMA process to be real-valued, the coefficients are required to be real, which translates to the system function having $2p$ and $2q$ complex conjugated poles and zeros.

Many non-stationary processes encountered in the real world, can still be found to exhibit WSS behavior in some sense. For instance, given a process where the mean is changing over time, the *difference* in values from one point in time to the next could still be WSS. The *autoregressive integrated moving average (ARIMA)* model is a generalization of the ARMA model particularly common in econometrics, which allows for such processes. It has the form

$$C(z)\{y_t\} = \frac{B(z)}{A(z)}\{u_t\}, \tag{2.36}$$

where

$$C(z) = (1 - z^{-1})^d \tag{2.37}$$

is the dth order difference operator. Typically, d is no higher than two (Box et al., 2008). Intuitively, taking the first difference of $\{y_t\}$ is the same as creating a new time series $\{y_t - y_{t-1}\}$ where each value is the difference between values of $\{y_t\}$ at time t and $t-1$. Taking the second difference is the same as taking the first difference of an already differenced time series, and so on. After differencing, the signal is treated as the realization of a regular ARMA process.

It can be shown that passing a stochastic process x_t through a linear shift-invariant filter results in a process y_t with power spectrum given by

$$P_y(e^{j\omega}) = P_x(e^{j\omega})|H(e^{j\omega})|^2, \tag{2.38}$$

where $H(e^{j\omega})$ is the frequency response of the filter (Hayes, 1996). Since we know that the power spectrum of the white noise process is just σ^2, we can immediately write down the power spectrum of the ARMA process as

$$P(e^{j\omega}) = \sigma^2 \frac{|B(e^{j\omega})|^2}{|A(e^{j\omega})|^2}. \tag{2.39}$$

Since the power spectrum is uniquely defined by the set of ARMA parameters $\{a_n\}$, $\{b_n\}$ and σ^2, estimation of these parameters can be seen as a form of *parametric spectrum estimation*.

2.2.3 Prediction Errors

Eq. (2.33) can be rewritten as

$$y_t = \hat{y}_{t|t-1} + u_t, \tag{2.40}$$

where

$$\hat{y}_{t|t-1} = -\sum_{n=1}^{p} a_n y_{t-n} + \sum_{n=1}^{q} b_n u_{t-n} \tag{2.41}$$

is the predicted value of y_t based on past values of y_t and u_t. This shows that u_t can be seen as a prediction error:

$$u_t = y_t - \hat{y}_{t|t-1} \tag{2.42}$$

If we knew the ARMA coefficients, as well as the p past values of y_t and the q past values of u_t, we could compute this error recursively from new samples of y_t:

$$u_t = y_t + \sum_{n=1}^{p} a_n y_{t-n} - \sum_{n=1}^{q} b_n u_{t-n} \tag{2.43}$$

Or equivalently, using filter notation:

$$\{u_t\} = \frac{A(z)}{B(z)}\{y_t\}, \tag{2.44}$$

where the filter would be initialized with the known previous values of y_t and u_t.

In any practical situation, we have to settle for estimates of the coefficients. And the white noise input signal is by definition unknown, so we cannot know its past values. The prediction errors obtained using estimated coefficients and a filter initialized with zeros (or something else) as past values, will therefore only be an estimate of the white noise input signal:

$$\{e_t\} = \{\hat{u}_t\} = \frac{\hat{A}(z)}{\hat{B}(z)}\{y_t\} \tag{2.45}$$

This quantity is often referred to as the residuals of the model, especially when it is computed from the data that was used to estimate the ARMA coefficients. It can be understood as the part of the data that is "unexplained" by the model. If the residuals of a model deviate significantly from what would be expected from a white noise process, it can be taken as a sign that the model is inadequate.

Eq. (2.45) offers an interesting view of ARMA parameter estimation and subsequent inverse filtering as a means of transforming a WSS process to white noise. Given a set of N samples from the process y_t, we can further view this as

an attempt to transform the normal random vector $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^{\mathrm{T}}$ into a white noise vector $\mathbf{e} = [e_1 \quad e_2 \quad \cdots \quad e_N]^{\mathrm{T}}$, where $\mathbf{e} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$. Note, however, the differences between this whitening transformation and the one we saw in Section 2.1.2. There we had many observations of a random vector with relatively few elements that was assumed to have a MVN distribution. The assumption of normality then allowed us to estimate a linear transformation taking the vector to a standard normal distribution. ARMA estimation, on the other hand, uses only *one* sample of a random vector $\mathbf{y}$ with N elements to create a recursive and hence non-linear transformation to a white noise vector $\mathbf{e}$. The reason this can work with just one sample is the special covariance structure of $\mathbf{y}$ caused by it being from a WSS process, where the autocorrelation is only a function of lag. Thus, increasing the number of elements in $\mathbf{y}$ actually just adds more information about the underlying covariance structure, thereby making it easier to transform $\mathbf{y}$ into a white noise vector.

2.2.4 Testing for Serial Correlation

Given a time series $\{x_t\}$ sampled from a random process we can define the sample autocorrelation function as

$$\hat{r}_k = \frac{1}{(N-1)\hat{\sigma}^2} \sum_{t=1}^{N-k} (x_t - \hat{\mu})\,(x_{t+k} - \hat{\mu}); \quad k \in [0, L], \tag{2.46}$$

where N is the number of samples in the series, $\hat{\mu}$ is the sample mean and $\hat{\sigma}^2$ is the sample variance. This is a normalized estimator for the ACF, so that $\hat{r}_0 = 1$ regardless of the variance σ^2.

If the time series is sampled from a white noise process, and L is small relative to N, Box and Pierce (1970) noted that the sample *autocorrelation vector (ACV)* $\mathbf{r} = [\hat{r}_1 \quad \hat{r}_2 \quad \cdots \quad \hat{r}_L]^{\mathrm{T}}$ will be asymptotically distributed as multivariate normal with mean $E[\mathbf{r}] = \mathbf{0}$ and covariance matrix

$$\mathbf{D} = E[\mathbf{r}\mathbf{r}^{\mathrm{T}}] = \frac{1}{N(N+2)} \begin{bmatrix} N-1 & 0 & \cdots & 0 \\ 0 & N-2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N-L \end{bmatrix}. \tag{2.47}$$

Given a realization of this vector, we can compute its squared Mahalanobis distance by

$$\mathrm{B}(\mathbf{r}) = \mathbf{r}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{r}, \tag{2.48}$$

where the inverse of $\mathbf{D}$ is

$$\mathbf{D}^{-1} = N(N+2)\begin{bmatrix} \frac{1}{N-1} & 0 & \cdots & 0 \\ 0 & \frac{1}{N-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{N-L} \end{bmatrix}. \tag{2.49}$$

Written out as a summation, the distance becomes

$$\mathrm{B}(\mathbf{r}) = N(N+2)\sum_{k=1}^{L}\frac{\hat{r}_k^2}{N-k}. \tag{2.50}$$

This is known as the Ljung-Box (LB) test statistic, and was introduced in Ljung and Box (1978). It is normally not presented as a distance measure, but this perspective will prove useful later on when we apply the statistic. It also allows us to immediately write down the distribution of $\mathrm{B}(\mathbf{r})$, as this has already been derived in (2.10):

$$\mathrm{B}(\mathbf{r}) \sim \chi^2_L \tag{2.51}$$

Starting from a null hypothesis that the sampled signal comes from a white noise process, we can therefore calculate the probability of observing a value β or higher of the test statistic from the CDF of the chi-squared distribution:

$$P(\mathrm{B} \geq \beta) = 1 - F(\beta). \tag{2.52}$$

This is known as a p-value. Note that the p-value is not the probability that the signal comes from a white noise process. Like any other hypothesis test, the LB test only seeks to find evidence *against* the null hypothesis. If it fails to do so, that does not prove that the data comes from white noise. However, the more the signal deviates from white noise, the lower the probability of observing its value of B will be, which in turn strengthens our belief that the signal is *not* generated by a white noise process.

To express this quantitively, we would define a threshold α on the p-values, and reject the null hypothesis if a p-value falls below this threshold. The threshold is referred to as the *significance level* of the hypothesis test. The significance level may be understood as the probability of committing a *type I error*, *i.e.* the probability of falsely rejecting the null hypothesis.

Ljung and Box introduced their test statistic as a way to evaluate the goodness of fit of an ARMA model by testing the model residuals for whiteness. They found, however, that when applied to the residuals of a fitted model and not an ordinary white noise time series, the chi-squared distribution with L degrees of freedom (DOF) did not provide a good fit for the test statistic. To

remedy this, they suggested a modification of the test in which the DOF parameter is set to $L - p - q$ when the test is applied to the residuals of a fitted ARMA model.

An intuitive explanation of why the degrees of freedom may need to be adjusted in such a situation, is that the decorrelation achieved on the data set used to fit the model, will tend to be better than what can be achieved on an "unseen" data set. In the extreme case one could imagine fitting an AR(p) model to a data set containing only $2p$ values. If the first half of the data were taken as previous values of $\{y_t\}$ and we wanted to minimize the prediction errors on the second half, Eq. (2.43) could be rewritten as

$$y_t + \sum_{n=1}^{p} a_n y_{t-n} = 0; \ t \in [p+1, 2p]. \tag{2.53}$$

This is a set of p linear equations with p unknowns that can be solved exactly for the coefficients $\{a_n\}$. All the residuals for the second half of the data set would therefore be zero, and hence uncorrelated. However, the prediction errors from this model on new data sampled from the same process, might be far from uncorrelated. This would be an (extreme) case of what is known as *overfitting*. Thus, adjusting the DOF parameter can be seen as a way of compensating for the bias introduced by using the same data to both fit and validate the model.

Generally, though, in situations where plenty of data from the process is available, it is more natural to use independent data sets for model fitting and model validation. This is known as *out-of-sample validation* and is what will be employed later in this thesis. As we shall see, in such a situation the Ljung-Box test statistic can be used with L degrees of freedom for the chi-squared distribution.

2.2.5 Parameter Estimation

The techniques for estimating ARMA parameters fall into two broad categories – iterative and non-iterative methods. Iterative methods are generally slower but more accurate. This means that they should be preferred in *offline* estimation, where there are no particular time-constraints. Non-iterative methods, on the other hand, are faster but less precise, making them better candidates for *online* estimation in real-time systems. In the special case of pure AR models, iterative methods offer very little improvement over non-iterative ones, so for such models there is no particular downside to using a fast non-iterative method both offline and online.

Over the last 60 years or so many different ARMA estimation methods have been proposed, and no attempt will be made here to give an overview of all of these. Instead, we will focus on two commonly used approaches – linear least squares (LS) and maximum likelihood (ML). Linear LS is a non-iterative approach, while ML is an iterative approach.

Least Squares for AR Models

Given N samples from a process assumed to be AR(p), Eq. (2.43) can be written as

$$y_t + \sum_{n=1}^{p} a_n y_{t-n} = e_t;\ t \in [p+1, N] \tag{2.54}$$

where the p first values of $\{y_t\}$ are used as previous values of the signal. This set of equations can be written in matrix form as

$$\begin{bmatrix} y_{p+1} & y_p & \cdots & y_1 \\ y_{p+2} & y_{p+1} & \cdots & y_2 \\ \vdots & \vdots & & \vdots \\ y_N & y_{N-1} & \cdots & y_{N-p} \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} e_{p+1} \\ e_{p+2} \\ \vdots \\ e_N \end{bmatrix}, \tag{2.55}$$

or more compactly:

$$\mathbf{Ya} = \mathbf{e}. \tag{2.56}$$

If we seek a set of coefficients that minimize the sum of squared prediction errors in these equations, they can be found as follows. Separate out the first column of $\mathbf{Y}$ and the coefficients in $\mathbf{a}$.

$$\begin{aligned} \begin{bmatrix} \mathbf{y} & \mathbf{X} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{\theta} \end{bmatrix} &= \mathbf{e} \\ \mathbf{y} + \mathbf{X}\boldsymbol{\theta} &= \mathbf{e} \end{aligned} \tag{2.57}$$

Define the least squares (LS) criterion:

$$\varepsilon^2(\boldsymbol{\theta}) = \|\mathbf{y} + \mathbf{X}\boldsymbol{\theta}\|^2 \tag{2.58}$$

By equating the gradient of this criterion with zero, one can find the vector $\hat{\boldsymbol{\theta}}$ that minimizes it:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \varepsilon^2(\boldsymbol{\theta}) = -(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{2.59}$$

where $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$ is called the pseudo-inverse of $\mathbf{X}$. To obtain an estimate of the last parameter $\hat{\sigma}^2$, note that $\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}}$ is an estimate of the white noise input signal for $t \in [p+1, N]$. A natural estimate of the noise variance will therefore be the mean squared value of $\hat{\mathbf{e}}$:

$$\hat{\sigma}^2 = \frac{1}{N-p}\left\|\mathbf{y} + \mathbf{X}\hat{\boldsymbol{\theta}}\right\|^2 \tag{2.60}$$

In the above formulation of the least squares AR method, t is set to run from $p+1$ to N, so that the p first values of the data can be used as previous values. This formulation is known as the *covariance method* of AR estimation. Another common formulation is the *autocorrelation method*, where t is set to run from 1 to $N+p$, and y_t is assumed to be zero for $t<1$ and $t>N$. The matrix representation then becomes:

$$\begin{bmatrix} y_1 & 0 & \cdots & 0 \\ y_2 & y_1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ y_{p+1} & y_p & \cdots & y_1 \\ y_{p+2} & y_{p+1} & \cdots & y_2 \\ \vdots & \vdots & & \vdots \\ y_N & y_{N-1} & \cdots & y_{N-p} \\ 0 & y_N & \cdots & y_{N-p+1} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & y_N & y_{N-1} \\ 0 & 0 & 0 & y_N \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N+p} \end{bmatrix}. \tag{2.61}$$

From the matrix $\mathbf{Y}$, a matrix $\mathbf{X}$ and a vector $\mathbf{y}$ can be formed in the same way as for the covariance method, and the parameter estimates will be given by

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= -(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \\ \hat{\sigma}^2 &= \frac{1}{N+p}\left\|\mathbf{y}+\mathbf{X}\hat{\boldsymbol{\theta}}\right\|^2 \end{aligned} \tag{2.62}$$

Although the two LS methods produce very similar estimates for large N, there are some differences worth noting. First, it can be shown that estimates obtained with the autocorrelation method are guaranteed to produce a stable filter $1/\widehat{A}(z)$. This is not the case for the covariance method, although unstable estimates rarely occur. Secondly, empirical evidence suggest that the covariance method is more accurate, perhaps because it does not make any assumptions about the values of the data outside the interval $[1, N]$ (Stoica and Moses, 2005, p. 94).

Two-Stage Least Squares for ARMA Models

If the white noise input signal were known, the ARMA parameter estimation would be quite simple. The idea of the two-stage least squares method given in Stoica and Moses (2005) is to form an estimate of the input signal using a high-order AR model, and then replace the true input noise by the estimated input noise in Eq. (2.43). This is a generalization to full ARMA models of the MA estimation technique introduced by Durbin (1959). According Hernandes et al. (2008), Durbin's method is probably the most commonly used non-iterative MA estimation technique.

Given an AR(K) model of y_t, the input noise in the two-stage least squares method is first estimated as

$$\{e_t\} = \hat{A}(z)\{y_t\} \tag{2.63}$$

The model then becomes

$$\begin{gathered} y_t + \sum_{n=1}^{p} a_n y_{t-n} - \sum_{n=1}^{q} b_n e_{t-n} = e_t \\ t \in [m+1, N];\ m = \max(p,q) + K \end{gathered} \tag{2.64}$$

This can be written in matrix form as

$$\mathbf{z} + \mathbf{Z}\boldsymbol{\theta} = \mathbf{e}, \tag{2.65}$$

where

$$\begin{gathered} \mathbf{z} = [y_{m+1} \quad y_{m+2} \quad \cdots \quad y_N]^{\mathrm{T}} \\ \mathbf{e} = [e_{m+1} \quad e_{m+2} \quad \cdots \quad e_N]^{\mathrm{T}} \\ \boldsymbol{\theta} = [a_1 \quad a_2 \quad \cdots \quad a_p \quad b_1 \quad b_2 \quad \cdots \quad b_q]^{\mathrm{T}} \\ \mathbf{Z} = \left[\begin{array}{ccc|ccc} y_m & \cdots & y_{m-p+1} & -e_m & \cdots & -e_{m-q+1} \\ y_{m+1} & \cdots & y_{m-p+2} & -e_{m+1} & \cdots & -e_{m-q+2} \\ \vdots & & \vdots & \vdots & & \vdots \\ y_{N-1} & \cdots & y_{N-p} & -e_{N-1} & \cdots & -e_{N-q} \end{array}\right]. \end{gathered} \tag{2.66}$$

The least squares solution is given by the pseudoinverse like for LS AR estimation:

$$\hat{\boldsymbol{\theta}} = -(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{z} \tag{2.67}$$

When p is set to zero in the above equations, this is equivalent to Durbin's method. As with pure AR estimation, the variance parameter is given by the mean square value of the prediction errors:

$$\hat{\sigma}^2 = \frac{1}{N-m}\left\|\mathbf{z} + \mathbf{Z}\hat{\boldsymbol{\theta}}\right\|^2 \tag{2.68}$$

A problem with the two-stage LS method is that its accuracy highly depends on the choice of the parameter K, and there is no simple way to find the best value of this parameter (Hernandes et al., 2008). For processes with zeros close to the unit circle, a high value of K is needed to properly decorrelate the process. One option is to employ some algorithm that first tries to find an optimal choice for K, but this may cause an unacceptable increase in complexity for a method whose main appeal is its simplicity.

Another drawback not mentioned in Stoica and Moses but which can be observed in experiments, is that the polynomial $\widehat{B}(z)$ is not guaranteed to be stable. If the estimate is going to be used to generate prediction errors, an unstable ARMA filter is useless, since the prediction error process will not be stationary, and may go off to infinity.

Maximum Likelihood

As we have seen earlier, the prediction error vector $\mathbf{e} = [e_1 \quad e_2 \quad \cdots \quad e_N]^{\mathrm{T}}$ will be distributed as

$$\mathbf{e} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}) \tag{2.69}$$

when it is computed from known past values and the true ARMA coefficients. Suppose we know the p past values of y_t and the q past values of e_t. Then we can derive the Maximum Likelihood (ML) estimate of the ARMA parameters as follows. Start by writing out the PDF of $\mathbf{e}$, noting its dependence on the parameter vector $\boldsymbol{\phi} = [\sigma^2 \quad \boldsymbol{\theta}^{\mathrm{T}}]^{\mathrm{T}}$:

$$\begin{aligned} f(\mathbf{e}|\boldsymbol{\phi}) &= \frac{1}{(2\pi)^{\frac{N}{2}}(\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{\mathbf{e}^{\mathrm{T}}\mathbf{e}}{2\sigma^2}\right] \\ f(\mathbf{e}|\boldsymbol{\phi}) &= \frac{1}{(2\pi)^{N/2}(\sigma^2)^{N/2}} \exp\left[-\frac{\|\mathbf{y} - \hat{\mathbf{y}}(\boldsymbol{\theta})\|^2}{2\sigma^2}\right] \end{aligned} \tag{2.70}$$

where

$$\begin{aligned} \mathbf{y} &= [y_1 \quad y_2 \quad \cdots \quad y_N]^{\mathrm{T}} \text{ and} \\ \hat{\mathbf{y}} &= [\hat{y}_{1|0} \quad \hat{y}_{2|1} \quad \cdots \quad \hat{y}_{N|N-1}]^{\mathrm{T}} \end{aligned} \tag{2.71}$$

are the two vectors that make up $\mathbf{e}$. If we see $\boldsymbol{\phi}$ as the dependent variable and $\mathbf{y}$ as the constant in Eq. (2.70), the resulting function $f(\boldsymbol{\phi}|\mathbf{y})$ is known as the *likelihood function* of $\boldsymbol{\phi}$, given some realization of $\mathbf{y}$. Maximizing the likelihood function is equivalent to maximizing its logarithm, the *log-likelihood function*:

$$\ln f(\boldsymbol{\phi}|\mathbf{y}) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{\|\mathbf{y} - \hat{\mathbf{y}}(\boldsymbol{\theta})\|^2}{2\sigma^2} \tag{2.72}$$

This leads to the *conditional* ML estimates in Eq. (2.13). They are referred to as conditional because they are conditioned on knowledge of the past values of y_t and e_t.

$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \arg\max_{\boldsymbol{\phi}} \ln f(\boldsymbol{\phi}|\mathbf{y}) \\ \hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \hat{\mathbf{y}}(\boldsymbol{\theta})\|^2 \\ \hat{\sigma}^2 &= \frac{1}{N}\left\|\mathbf{y} - \hat{\mathbf{y}}\left(\hat{\boldsymbol{\theta}}\right)\right\|^2\end{aligned} \tag{2.73}$$

Maximizing the conditional likelihood of $\boldsymbol{\theta}$ is thus equivalent to finding a vector $\hat{\boldsymbol{\theta}}$ that minimizes the sum of squared prediction errors for a given realization of $\mathbf{y}$.

Deriving the criterion for the *unconditional* ML ARMA estimate is more complicated, but it is sufficient to mention here that the unknown previous values are first estimated using "back-forecasting", *i.e.* a model of the time series where time is running backwards, so that estimating past values becomes a prediction problem. Then, a similar expression to Eq. (2.73) can be derived involving the conditional expectation of the error vector and of the back-forecasted previous values:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|E[\mathbf{e}|\mathbf{y}, \boldsymbol{\theta}]\|^2 + E[\mathbf{a}]^{\mathrm{T}} \boldsymbol{\Omega}^{-1} E[\mathbf{a}] \tag{2.74}$$

where $\mathbf{a} = [\hat{y}_{1-p} \quad \cdots \quad \hat{y}_0 \quad \hat{e}_{1-q} \quad \cdots \quad \hat{e}_0]^{\mathrm{T}}$ is a vector of back-forecasted previous values, and $\boldsymbol{\Omega}$ is the covariance matrix of $\mathbf{a}$. For details on how the derivation of this expression and recommendations on how to compute the estimate in practice, can be found in Box et al. (2008).

In any case, for full ARMA models a linear method of finding the ML estimate is not possible, since we cannot write the error vector as a linear function of the parameter vector. Instead, each element of $\mathbf{e}$ must be computed recursively from (2.43) for any given vector $\boldsymbol{\theta}$. The problem of finding the ML estimate $\hat{\boldsymbol{\theta}}$ in the case of full ARMA models is therefore referred to as a *non-linear least squares* problem.

The way to approach the non-linear estimation problem, is to use an *iterative optimization algorithm*. Such an algorithm starts with some initial guess for $\hat{\boldsymbol{\theta}}$, and then computes the criterion function to be minimized. It then uses one of several ways to find a direction in the parameter space which is likely to improve the estimate. Next, it updates the estimate by moving the parameter vector a small step in that direction and computes the criterion function again. It continues this way until it has a satisfactory value of the criterion, or the maximum number of iterations has been reached. One common algorithm for iterative optimization is *gradient descent*, in which the gradient of the criterion function is estimated on each iteration, and the step is taken in the opposite direction of the gradient.

Prediction Error Minimization

A class of methods closely related to maximum likelihood is the *prediction error methods (PEM)* used in the field of system identification. There the goal is to estimate the parameters of a more general class of systems of the form

$$\{y_t\} = G(z)\{v_t\} + H(z)\{u_t\}, \tag{2.75}$$

where v_t is a known input, and u_t is white noise. Generally, one would define a criterion function of the prediction error and select the parameters of $G(z)$ and $H(z)$ that minimize this criterion. A common choice of criterion is the quadratic norm, like in conditional ML ARMA estimation:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \hat{\mathbf{y}}(\boldsymbol{\theta})\|^2 \tag{2.76}$$

As in ML, the solution is found through iterative optimization. Since the ARMA model is just a special case of the more general model used in system identification, *i.e.* one where the known input v_t is zero, all the parameter estimation methods developed in the field can be used for ARMA estimation. In these methods y_t and u_t are often taken to be zero for $t < 1$. This means that they are not exactly computing the conditional ML ARMA estimate, but the effect of initial values becomes negligible for high N.

In system identification, a lot of work has gone into refining the iterative optimization algorithms used for parameter estimation. For general treatments, see Söderström and Stoica (1989) or Ljung (1999).

2.2.6 Order Selection

In time series analysis, there are two common approaches to selecting the order of an ARMA model. The first is based on visually inspecting estimates of the autocorrelation function and *partial autocorrelation function (PACF)* of the process.

The PACF α_k is defined as the autocorrelation between y_t and y_{t+k} with the linear dependence on y_{t+1} up to y_{t+k-1} removed. The closed form expression for this is rather complicated and will be omitted here for readability. When estimating the PACF from data, a fitted AR(k) model is used to estimate α_k. Details on PACF estimation can be found in Box et al. (2008).

The ACF and PACF of the three model types AR(p), MA(q) and ARMA(p, q) have the following properties: The ACF of an AR(p) process decays gradually, while its PACF has a sharp drop in magnitude after lag p. In contrast, the ACF of an MA(q) process has a sharp drop in magnitude after lag q, while its PACF decays gradually. For an ARMA(p, q) process, both functions decay gradually and other methods are needed to determine p and q.

For a wide sense stationary process, the sample ACF will usually decay fairly quickly. When this is not the case, and the sample ACF decays very slowly in a

seemingly linear way, it could mean that the process is not WSS. This may be resolved by differencing the time series (see Section 2.2.2).

If examination of the estimated ACF and PACF shows that neither a pure AR model nor a pure MA model is sufficient to describe the data, the common approach to finding the appropriate orders of the ARMA(p, q) model is to make use of an *information criterion*. Such a criterion weighs the computed likelihood of a given model against the number of parameters in the model, and sometimes the number of samples from the process. It is defined so that a model is rewarded for a high likelihood (*i.e.* a "good fit") and punished for a high number of parameters. In other words, it favors simple models with a high likelihood. By fitting many different models to the data and computing the information criterion for each of them, we can select the model that gives the lowest value for the criterion. The information criterion approach is described in more detail in Stoica and Moses (2005, pp. 387–398).

A common choice of information criterion in ARMA modeling is the *Bayesian information criterion (BIC).* The BIC of an ARMA(p, q) model is defined as

$$\text{BIC} = -2\ln f(\boldsymbol{\phi}|\mathbf{y}) + (p + q + 1)\ln N, \tag{2.77}$$

where $\ln f(\boldsymbol{\phi}|\mathbf{y})$ is the log-likelihood function, as defined in Eq. (2.72) and N is the number of samples from the process.

2.2.7 Distribution of Estimates

When the parameters of an ARMA process are estimated several times from independent realizations of the process, it is often assumed that the parameter vector has a multivariate normal distribution. In fact, whenever the Mahalanobis distance is used to score parameter estimates, such as in Gul and Catbas (2005), we are essentially assuming that the distribution of the parameter vector can be described by the mean and the covariance matrix alone, *i.e.* that it is MVN.

In maximum likelihood estimation, the ARMA parameter vector can be shown to approach a multivariate normal distribution (Yao and Brockwell, 2006). However, we will not use maximum likelihood estimates in a way that requires knowledge of their distribution.

For the case of AR coefficients estimated using least squares, a derivation of the normality of the estimates was given in Mann and Wald (1943). However, it is unclear to me whether the full parameter vector, with $\hat{\sigma}$ included, can really be normally distributed. The simple fact that $\hat{\sigma}$ cannot be negative, makes this seem unlikely. Furthermore, Stoica and Moses (2005) do not discuss the distribution of the estimates obtained by their two-stage least squares ARMA algorithm. Nevertheless, for now we will assume that the parameter vectors obtained from least squares estimation can be modeled as MVN variables, or at least that they are close enough that the Mahalanobis distance is a valid way of scoring the estimates.

We saw in Section 2.2.4 that by testing the prediction errors of an ARMA model on new data, we could assess how well the data fitted our model. This was accomplished by computing the sample autocorrelation vector of the prediction errors and taking its Mahalanobis distance. Now, if the assumption of normality holds, we have another way of assessing the fit of new data – namely to model the estimated parameter vector $\hat{\phi}$ as MVN and take the Mahalanobis distance of new estimates obtained from data. These two approaches form the basis of the detection methods presented in the next chapter.

3 ARMA-Based Anomaly Detection

This chapter will first give the details of the detection methods examined in this thesis. It then goes on to describe how the experiments on the methods will be conducted. Next, an ARMA model of underwater ambient noise is built from hydrophone data. This model is used to simulate ambient noise and test the detection methods on sinusoids in noise of varying amplitudes. At the end of the chapter, we will look at how anomalous data can be incorporated into the model building procedure to influence the detector's sensitivity towards certain types of anomalies.

3.1 Hypothesis Testing

Given a segment of hydrophone data, the basic problem of detecting anomalies can be formulated as a hypothesis test where the two hypotheses in their most general form are as given below.

H_0: The data segment is a realization of ambient noise.

H_1: The data segment is not a realization of ambient noise.

If a test statistic can be constructed that has a known probability distribution under the null hypothesis, then the value of this statistic can be converted to a p-value. If the p-value for the data segment is lower than some threshold, the null hypothesis is rejected and the data segment is classified as anomalous.

From the theoretical discussion in the previous chapter, two ways of making the hypotheses above more specific have emerged. Both are based on modeling the ambient noise as a realization of an ARMA process, and both give rise to a chi-squared test statistic.

3.1.1 Successive Estimation Method

In the successive estimation method, the model of the ambient noise is a distribution model of the ARMA parameter estimates. We have seen that when parameters are estimated from independent realizations of the same underlying ARMA process, the parameter vector

$$\hat{\boldsymbol{\phi}} = \begin{bmatrix} \hat{\sigma} & \hat{a}_1 & \cdots & \hat{a}_p & \hat{b}_1 & \cdots & \hat{b}_q \end{bmatrix}^{\mathrm{T}}, \tag{3.1}$$

can often be modeled as a multivariate normal variable, *i.e.*

$$\hat{\boldsymbol{\phi}} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.2}$$

The parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of this model are not known in advance, but they can be estimated from data. Given estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the precise hypotheses can be expressed as below.

> H_0: The ARMA parameters estimated from the data segment are drawn from the distribution $N\big(\hat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}\big)$
>
> H_1: The ARMA parameters estimated from the data segment are not drawn from the distribution $N\big(\hat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}\big)$

By taking the squared Mahalanobis distance of $\widehat{\phi}$ from the distribution mean,

$$D(\hat{\boldsymbol{\phi}}) = (\hat{\boldsymbol{\phi}} - \hat{\boldsymbol{\mu}})^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}^{-1} (\hat{\boldsymbol{\phi}} - \hat{\boldsymbol{\mu}}), \tag{3.3}$$

we obtain a test statistic that has a known distribution under the null hypothesis:

$$D(\hat{\boldsymbol{\phi}}) \sim \chi^2_{p+q+1} \text{ under } H_0. \tag{3.4}$$

From $D(\hat{\boldsymbol{\phi}})$ we can calculate the probability of observing $\hat{\boldsymbol{\phi}}$ or something more extreme, given that the true distribution is $N\big(\hat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}\big)$.

3.1.2 Prediction Method

In the prediction method, an ARMA(p, q) model is first fitted to ambient noise data. We have seen that when the estimated parameters are equal to the true parameters, the resulting prediction errors in Eq. (2.45) of a given data segment will be a realization of white noise. If the data segment deviates from what would be expected by our model, this will introduce correlations in the prediction errors, making them non-white. In Section 2.2.4 we saw that to test for such correlations, we can calculate the Ljung-Box test statistic B($\mathbf{r}$) from (2.48), where the sample autocorrelation vector $\mathbf{r} = [\hat{r}_1 \quad \hat{r}_2 \quad \cdots \quad \hat{r}_L]^{\mathrm{T}}$ is found from the prediction errors $\{e_t\}$ by

$$\hat{r}_k = \frac{1}{(N-1)\hat{\sigma}_e^2} \sum_{t=1}^{N-k} (e_t - \hat{\mu}_e)\,(e_{t+k} - \hat{\mu}_e); \quad k \in [1, L]. \tag{3.7}$$

The precise hypotheses being tested by the LB statistic can be expressed as

> H_0: The true autocorrelation of the prediction errors is zero for all lags from lag one up to lag L.
>
> H_1: The true autocorrelation of the prediction errors is non-zero for at least one lag from lag one up to lag L.

We noted that $\mathrm{B}(\mathbf{r})$ is simply the squared Mahalanobis distance of $\mathbf{r}$ from the distribution mean $E[\mathbf{r}] = \mathbf{0}$, and will hence be chi-squared distributed with L degrees of freedom, provided that the null hypothesis is true. That is,

$$\mathrm{B}(\mathbf{r}) \sim \chi^2_L \text{ under } H_0. \tag{3.9}$$

Thus, we have made the general hypothesis test more concrete by turning it into a test for correlation in the prediction errors of a fitted ARMA model. In this test, we are free to choose the number of lags L that will be tested, although it is recommended to set L to at least 20 (Box et al., 2008).

Note that this is only one of several possible hypothesis tests that could have been performed on the prediction errors. In experiments not included in the final version of this thesis, I tried performing a *t*-test on the mean value of the prediction errors. In those experiments, the ARMA model order had to be set very high to get a test statistic that fitted well with the *t*-distribution. Additionally, the LB test showed better detection performance on the data set in question. As a consequence, the *t*-test was not pursued further. However, a more thorough and systematic comparison of possible prediction error hypothesis tests would be needed to conclusively say which approach is the best one in general.

3.1.3 ARMA Estimators

Both the methods above depend on having a reliable way of estimating ARMA parameters from data. Some common ways of doing this are described in Section 2.2.5. MATLAB functions implementing each of the estimation methods there are listed in the Appendix and will be described here briefly.

The function `LS_ARMA` is an implementation of the least squares estimation methods. For pure AR estimation, it uses the covariance method, and for full ARMA models it follows the two-stage least squares approach with K set to 20. Since there are no clear guidelines on how to choose the parameter K, I estimated some spectra using different values and found that 20 seemed to give a good fit with the true spectrum. Since the two-stage method is a fast, non-iterative algorithm it will be used for the successive estimation method, where new ARMA estimates are needed for each data segment.

The function `ML_ARMA` uses functions from MATLAB's Econometrics Toolbox to compute the unconditional maximum likelihood estimate of the ARMA parameters. Since the estimation algorithm is iterative, this function will be used for the prediction method, where the parameters only need to be estimated once, and the test statistic is obtained from subsequent inverse filtering.

`PEM_ARMA` is an alternative to `ML_ARMA` that uses functions from MATLAB's System Identification Toolbox. It estimates the ARMA parameters by iteratively minimizing the prediction errors. Experiments have shown that `ML_ARMA` and

`PEM_ARMA` produce practically identical results. But `PEM_ARMA` has been observed to work somewhat faster. Still, the method is significantly slower than least squares and will therefore only be used for the prediction method.

3.2 Unsupervised Detection

This section is an outline of how ARMA-based detectors will be constructed when no prior knowledge of the anomalies is available. This is known in classification theory as *unsupervised* detection. In section 3.6 a supervised version of the methods will be introduced where examples of anomalous data are also used in constructing the detectors.

The approach has grown out of my own attempts at constructing ARMA-based detectors in MATLAB using a typical pattern classification framework, as the one presented in Duda et al. (2001). As such, both this section and Section 3.6 can be seen as an overview of the MATLAB program I have built to run the various experiments presented later on. The graphical user interface of the program is shown in Figure A.1 in the Appendix. All the methods discussed in this chapter correspond to different configurations of the user settings in that figure.

3.2.1 Overview of Procedure

In both of the detection methods in the previous section, segments of raw data are transformed into a MVN variable with known or estimated parameters. This can be seen as a form of feature extraction, where the data segments are mapped to a low-dimensional space, and the elements, or features, of the random vectors are believed to contain valuable information about the data segments. The test statistic is then formed by taking the Mahalanobis distance of the feature vector. This means that we can view the methods in the same overall framework for detector design, where the two methods represent alternative approaches to feature extraction. The word *detector* should here be taken to mean a set of specifications such as feature extraction method, ARMA estimator and model parameters.

The procedure will consist of two stages: training and testing. In the training stage, a model is fitted to ambient noise data using one of the feature extraction methods. In the test stage, a data set containing both ambient noise and examples of anomalies is used to compute the test statistic, and the results are presented graphically.

Since we plan to calculate p-values from the test statistic, it is important to verify that it actually follows the assumed distribution under the null hypothesis. A goodness of fit (GOF) test with the chi-squared distribution is therefore integrated in the training stage[2]. This can be seen as a form of model validation.

[2] Goodness of fit is tested with the Kolmogorov-Smirnov test (`kstest` in MATLAB).

The training set is split into two parts, so that the data used for validation is independent of the data used to fit the model.

3.2.2 Details of Procedure

Figures 3.1 and 3.2 show an outline of the training and test procedures for the two feature extraction methods. In the figures, $\mathbf{Y}$ is a data matrix where the rows are made up of consecutive segments of time series data. The prediction error matrix $\mathbf{E}$ has the same dimension as $\mathbf{Y}$.

In the successive estimation method, the rows of $\mathbf{\Phi}$ are ARMA parameters estimated from the corresponding rows of $\mathbf{Y}$, while in the prediction method they are sample autocorrelation vectors computed from the corresponding rows of $\mathbf{E}$. For simplification, the ARMA estimation block is shown the same way when it estimates one parameter vector from an entire data matrix as when it estimates one parameter vector for each row of a data matrix.

The vectors $\mathbf{d}$ and $\mathbf{p}$ produced by the test procedure contain distances and corresponding p-values for each data segment of the test file. These vectors can be compared with fixed threshold values to label the segments as either normal or abnormal.

A number of user settings such as ARMA orders, segment length and autocorrelation lags, will also need to be specified to properly train and test a detector with this procedure. The choice of these settings will affect both the behavior of the test statistic on the validation data and the detector's performance on the test data.

If the end goal were to design a detector that worked on streams of time series data in real-time, such processing would include the same steps as the test procedure, but instead of getting a whole data matrix as input, the detector would process the data segments one at a time as they became available. Training and testing on stored data sets could still be performed offline before deploying the detector to a real-time system.

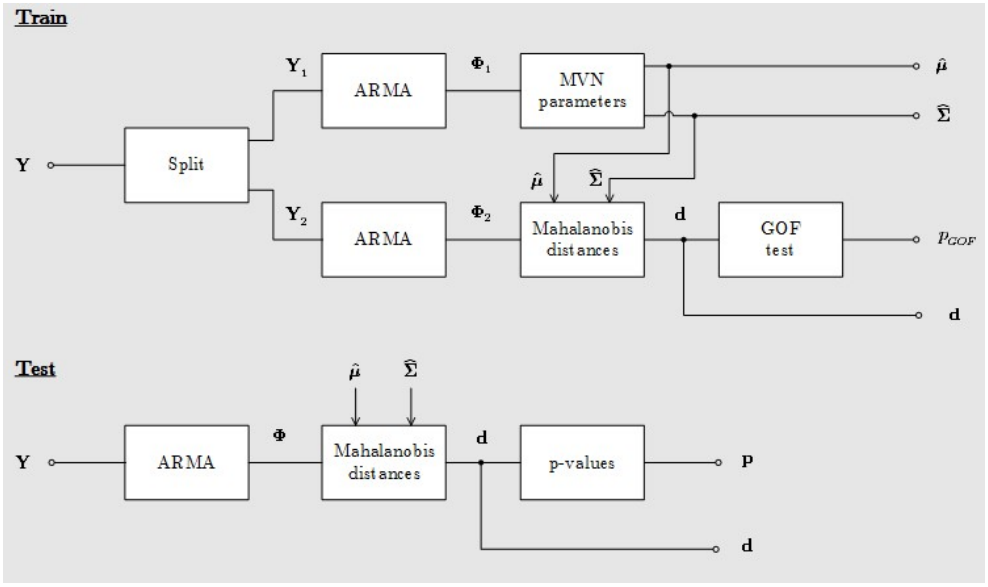

Figure 3.1: Overview of the detector training stage and test stage using successive ARMA estimation for feature extraction.

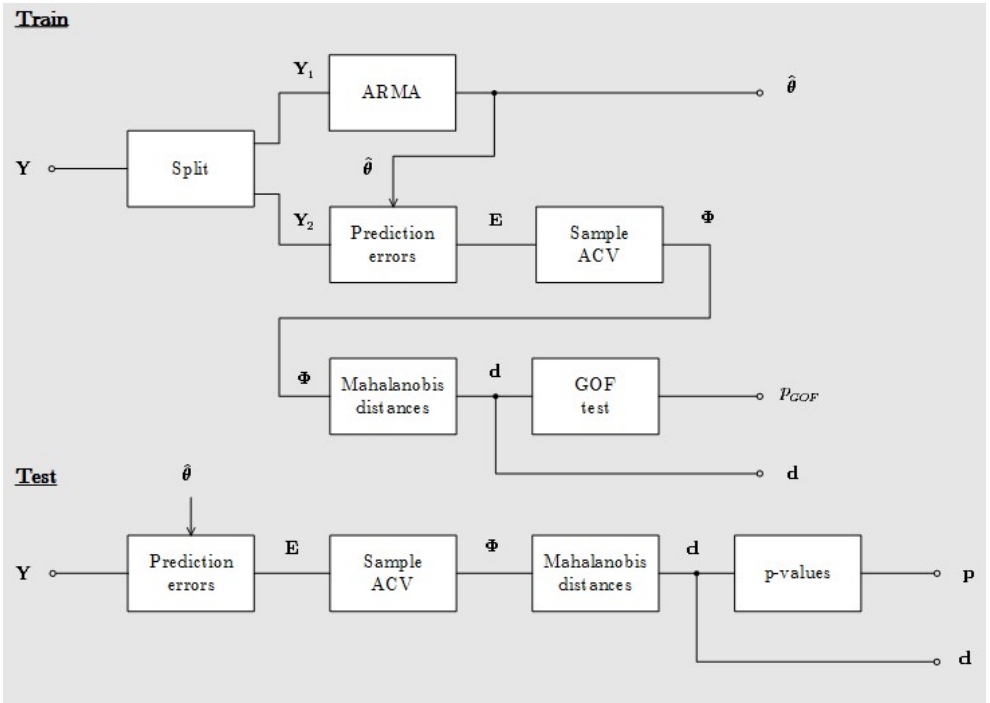

Figure 3.2: Overview of the detector training stage and test stage using the prediction method for feature extraction.

3.3 Modeling an Ambient Noise Signal

Before proceeding with experiments on simulated data, we first need to build a realistic model of the underwater ambient noise. To do this, we will use 10 seconds of hydrophone data from LoVe. The signal is sampled at 22.05 kHz with a surface wind speed of 18 m/s. A time-domain plot of the signal is shown in Figure 3.3.

The tools that will be used to build the model are the order selection methods given in Section 2.2.6 as well as visual inspection of the power spectrum of the estimated models plotted alongside a non-parametric PSD estimate. Non-parametric estimation is done here by Welch's method. This method has not been covered in the theory chapter, as it will only be used for visualization. More on Welch estimation can be found in Stoica and Moses (2005).

The hydrophone data set from LoVe was supplied by FFI with instructions on how to scale power spectral densities to the correct physical units. Throughout the thesis, the following formula has been applied to the PSD estimates.

$$P_s(e^{j\omega}) \text{ [dB re } \mu\text{Pa}^2\text{/Hz]} = 10\log_{10} P_u(e^{j\omega}) + 171, \tag{3.10}$$

where $P_u(e^{j\omega})$ is the unscaled power spectrum of the audio signal, and $P_s(e^{j\omega})$ is the scaled power spectrum in proper physical units. Figure 3.4 shows the Welch PSD estimate of the ambient noise signal converted to decibels using the above formula.

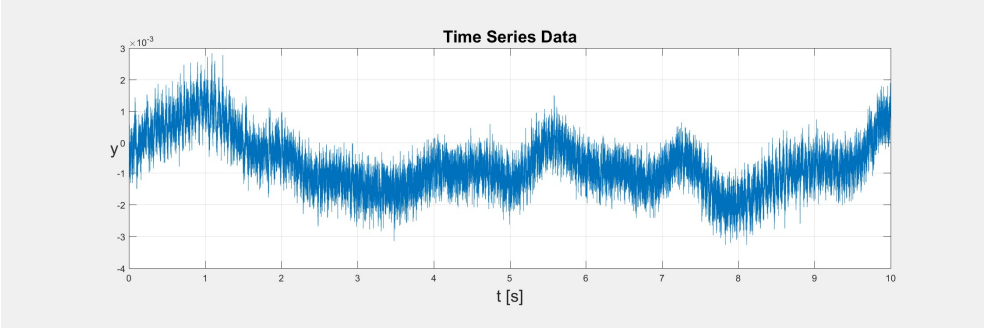

Figure 3.3: A plot of the 10 seconds of audio data used in this section to build a model of the ambient noise.

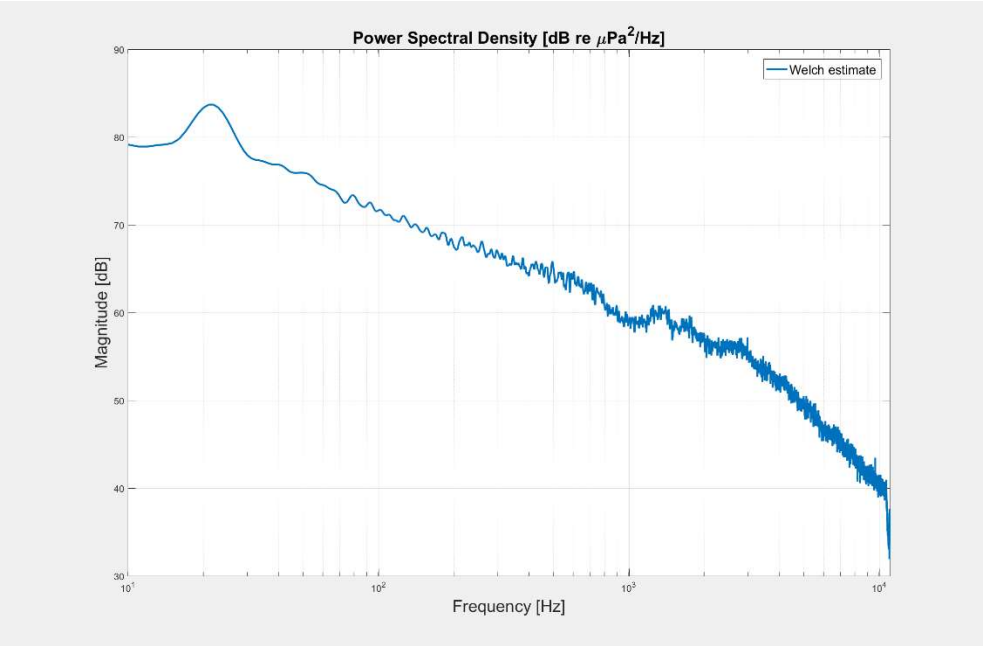

Figure 3.4: A Welch estimate of the power spectral density of the ambient noise signal.

Order selection

The first step in building an ARMA(p, q) model is to select the model orders p and q. As discussed in Section 2.2.6, this can be done by first inspecting estimates of the autocorrelation and partial autocorrelation functions of the signal. Estimates for 30 lags of these functions are shown in Figure 3.5. It is clear that neither the sample ACF nor the sample PACF drops to zero after a specific lag, which is indicative of a full ARMA process rather than a pure AR or MA process. Furthermore, the sample ACF decays very slowly, which is indicative of process with at least one pole close to unity (Box et al., 2008, pp. 196–197).

A slowly decaying ACF means that we could treat the process as non-stationary and instead try to model the first difference of the signal as ARMA(p, q). The effect of differencing would be to attenuate low frequencies, since the differencing operator in (2.37) is essentially a high-pass filter. This is not necessarily a problem when detection of anomalies is the only objective, but since the goal of this section is to build a realistic model of the signal for simulation purposes, we will treat the underlying process as wide sense stationary and avoid differencing.

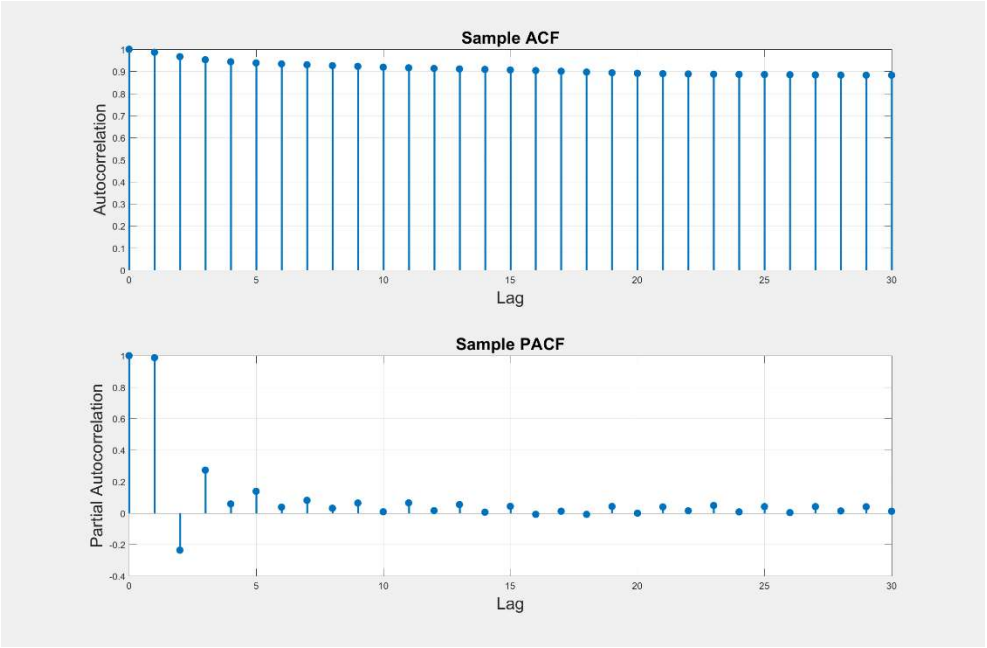

Figure 3.5: Plots of the sample autocorrelation function and sample partial autocorrelation function of the ambient noise signal.

Since the orders of a full ARMA process cannot be found by simple inspection of the sample ACF and PACF, another approach is required. As described in Section 2.2.6, a systematic way of finding the appropriate ARMA orders is to use an information criterion like the BIC. The function `BICorder` listed in the Appendix is adapted from an example on Mathworks.com (2016). It uses functions from the Econometrics Toolbox to compute the BIC for all combinations of AR orders from one up to `p_max` and MA orders from one up to `q_max`. It then returns the orders p and q that produce the lowest value for the BIC.

When the ambient noise data is fed to `BICorder` with `p_max` and `q_max` both set to 15, the resulting model is ARMA(11,4). The maximum orders were chosen so that the function would finish in reasonable time, as the maximum likelihood estimation done by `BICorder` can be very time-consuming for high orders and large amounts of data.

Parameter estimates

When an ARMA(11,4) model is fitted to the whole time series with `PEM_ARMA`, the resulting parameters are as given below.

$$\begin{aligned}\{a_n\} = \{&1.000, -2.152, 0.7093, 1.264, -1.129,\\ &0.3777, -0.09949, 0.06326, -0.02588,\\ &-0.02759, 0.04334, -0.02346\}\\ \{b_n\} = \{&1.000, -0.9117, -1.006, 0.8364, 0.08148\}\\ \sigma = &1.257 \text{ x } 10^{-4}\end{aligned} \tag{3.11}$$

These parameters define the model that will be used for simulation in the next section. The power spectrum of the estimated model is shown alongside the Welch estimate in Figure 3.6. Figure 3.7 shows the poles and zeros of the estimated ARMA model.

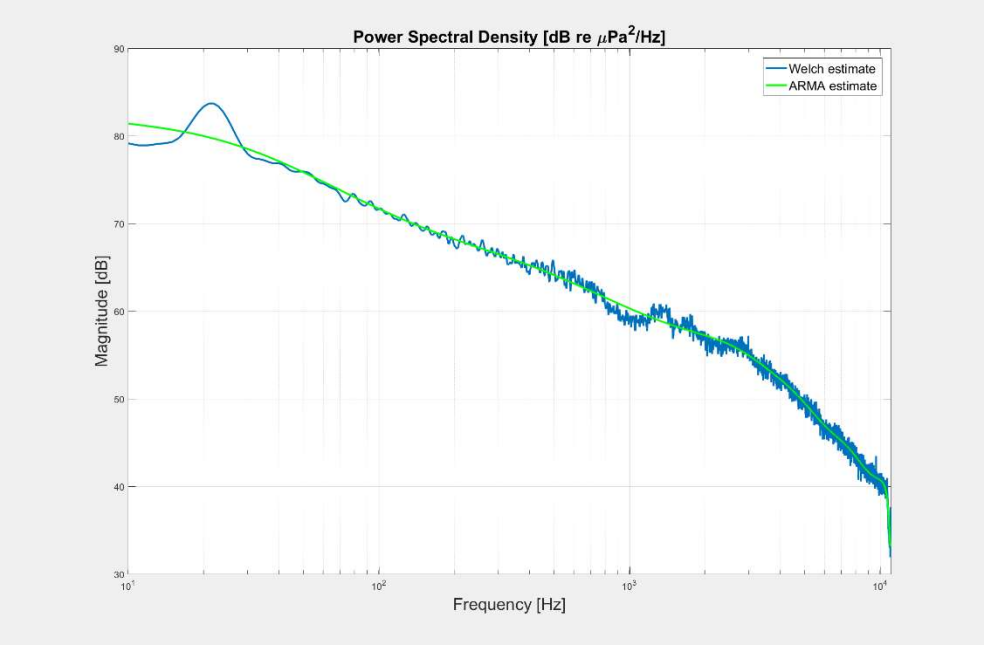

Figure 3.6: The ARMA PSD estimate and the Welch PSD estimate for the ambient noise signal.

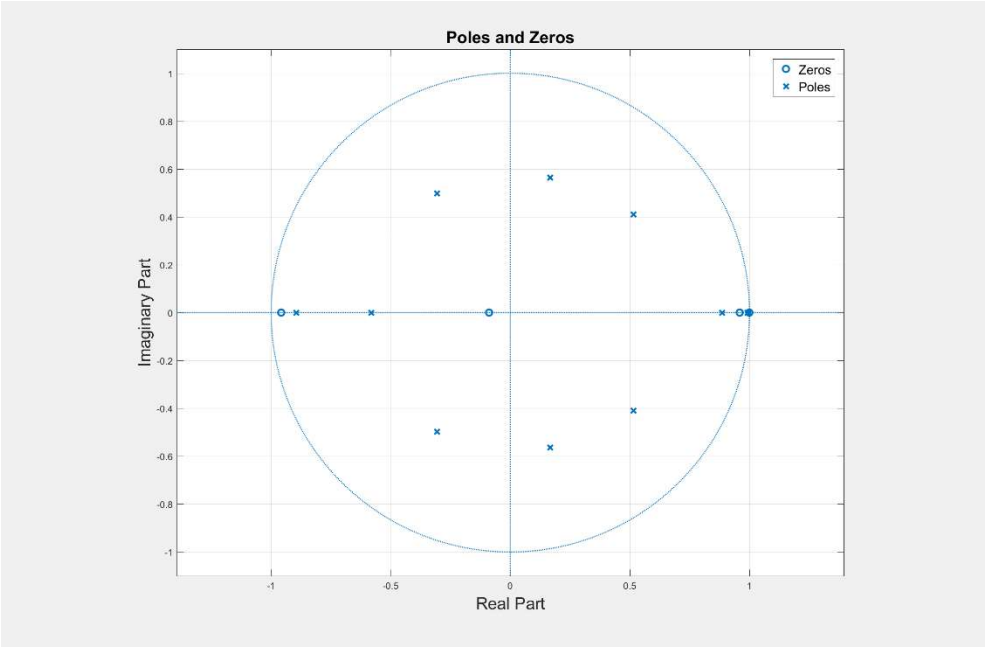

Figure 3.7: A pole-zero plot of the estimated ARMA model.

3.4 Detecting a Sinusoid in Noise

As a simple way of comparing the performance of the ARMA-based detection methods, we can run simulations of the model obtained in the previous section, and add sinusoids of varying amplitude to the signal. One minute of simulated data at a sampling frequency of 22.05 kHz will be generated from the model and the sinusoids will be added to the second half of the signal. The frequency of the sinusoids is 100 Hz. The amplitudes are given in decibel according to the following formula:

$$\sigma_s^2 \text{ [dB re } \mu\text{Pa}^2] = 10\log_{10}\sigma_u^2 + 171 \tag{3.12}$$

where σ_u^2 is the mean squared value of the signal. This will also be referred to as the total power of the signal. When referring to a sinusoid alone, the mean square value is simply $A^2/2$, where A is the multiplying factor of the sinusoid.

The energy detector mentioned earlier also uses mean square values to detect changes, but it assumes no prior knowledge of the signal and therefore has to rely on sample estimates of the total power. If we had a distribution model for the mean square signal values under ambient noise, we could set up a hypothesis test with a fixed significance level for the energy detector in the same way as for the ARMA-based methods. But no attempt has been made here to derive such a model. Instead, the energy detector will be evaluated by inspecting plots of the mean square value itself, not p-values. The detection thresholds in these plots

have simply been adjusted so that they could be suitable for the experiment in question.

In the experiments, Welch PSD estimates are used to form a *spectrogram* of the time series data. A spectrogram consists of PSD estimates stacked together along the time-axis to provide a graphical representation of how the power spectrum of a signal changes over time. The segment length of the spectrogram has been set to 0.5 seconds.

The segment length of the data used by the detectors will be $\Delta t = 0.04$ s. This is the same as what will be used in most of the experiments on real data in Chapter 4. The choice of the value is explained there.

The significance level chosen for the simulations is 10^{-5}. This may seem rather low but with $1/0.04 = 25$ p-values generated every second, this threshold means that we should expect around 25 x 60^2 x $10^{-5} = 0.9$ false alarms per hour of simulated noise data, given that the test statistic actually follows the assumed distribution.

3.4.1 Energy Detector

As is shown in Figure 3.8, at 90 dB the sinusoid is faintly visible in the spectrogram. Exactly when this occurs, has been observed to depend on the segment length used for the spectrogram. The longer the segments, the earlier the sinusoid becomes visible.

The total power of the signal does not seem to be affected by the 90-dB sinusoid at all. The mean square values also vary a lot. For a sinusoid to be detected in this situation, it needs to overcome the variance of these values. Figure 3.9 shows that this happens at an amplitude of around 118 dB.

Looking at the time series plot, the high variance in the mean square values appears to be due to power at some very low frequencies causing the signal to drift away from zero for several seconds at a time. This effect can also be observed in the plot of the ambient noise signal we used to build the model (Figure 3.3).

An interesting thing happens to the energy detector if we apply first order differencing to the signal before computing mean square values. This attenuates the low frequencies causing drift and significantly reduces the variance of the mean square values, as shown in Figure 3.10. When differencing is applied, the energy detector seems to be able to handle amplitudes down to about 113 dB, as shown in Figure 3.11. The threshold in these two figures has to be set differently since differencing has removed some of the total power of the signal.

It may very well be that differencing or some other form of prefiltering would have a beneficial effect on the ARMA-based detection methods as well, but this has not been studied here. In practice, differencing the signal means that the ambient noise model would become ARIMA(p, d, q) instead of ARMA(p, q).

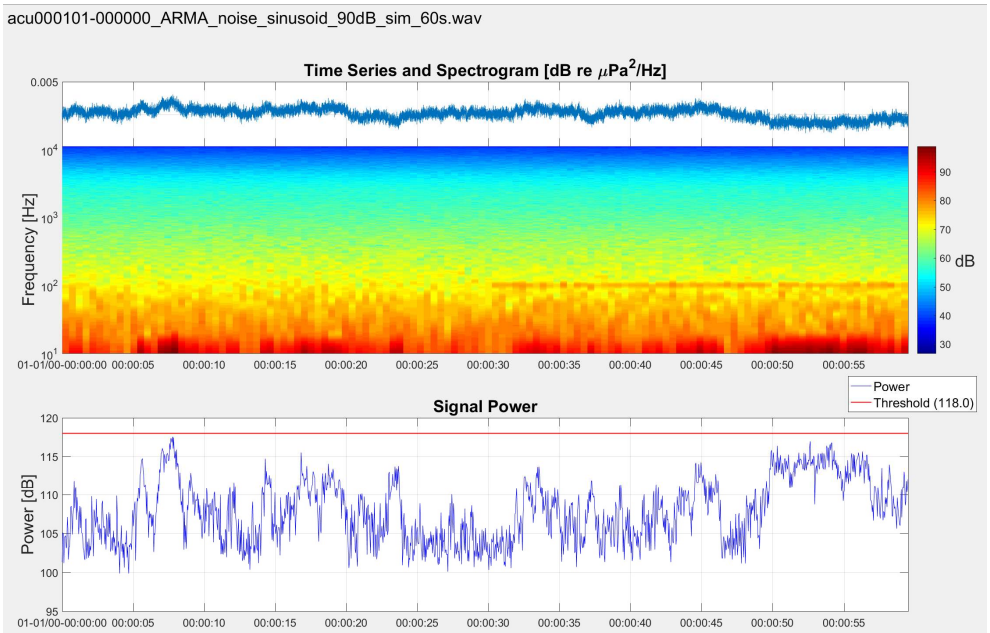

Figure 3.8: Energy detector tested on a 90-dB sinusoid in noise.

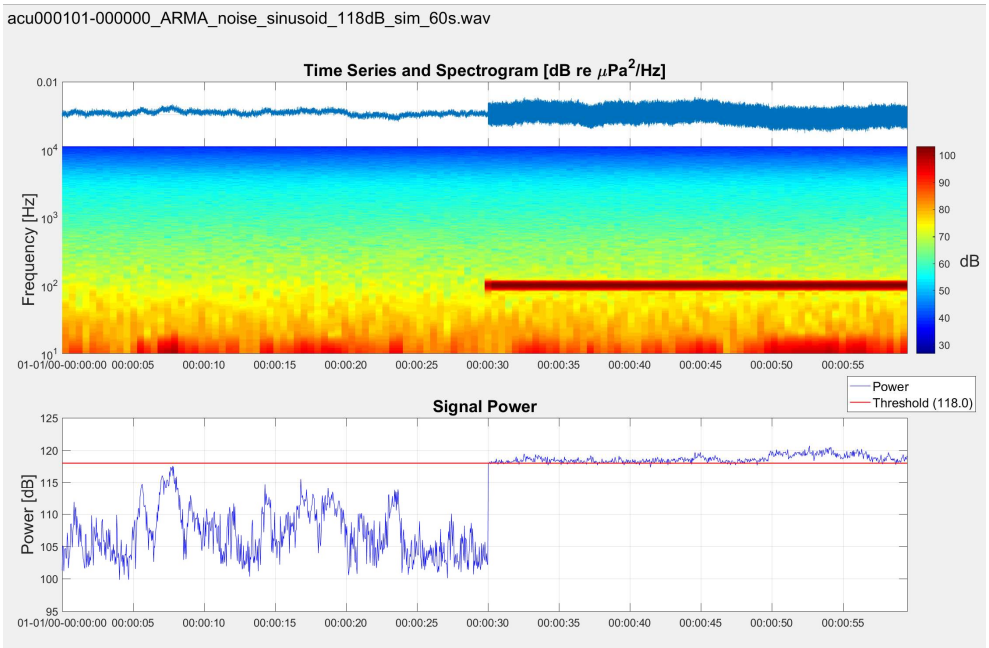

Figure 3.9: Energy detector tested on a 118-dB sinusoid in noise.

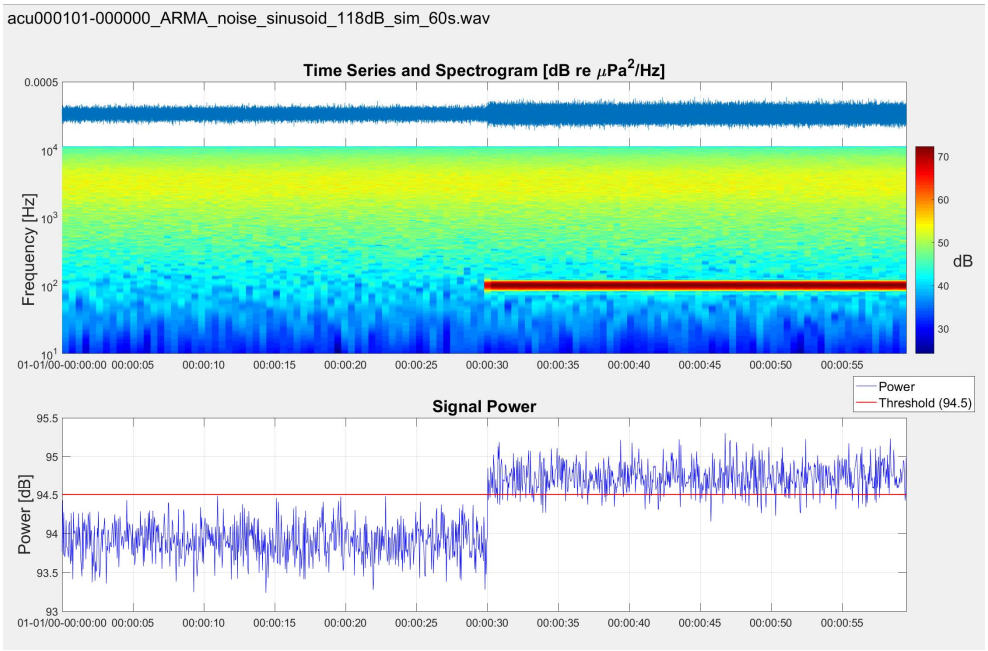

Figure 3.10: Energy detector with differencing tested on a 118-dB sinusoid in noise.

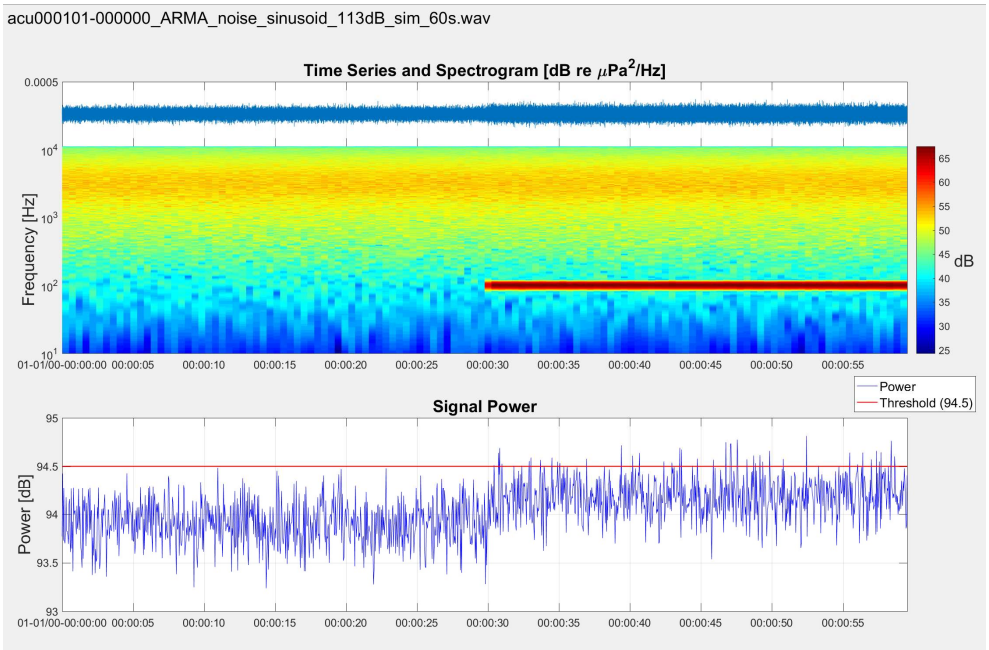

Figure 3.11: Energy detector with differencing tested on a 113-dB sinusoid in noise.

3.4.2 ARMA-Based Detectors

For each of the ARMA-based methods, two figures have been created. The first figure shows the results of the training procedure, *i.e.* model fitting and model validation. The second figure shows detector performance in the same way as was shown for the energy detector.

The training figures contain the following plots: the estimated power spectrum shown alongside a Welch estimate of the spectrum, a histogram of the test statistic values under validation shown alongside the chi-squared PDF, and a pole-zero plot of the estimated ARMA coefficients.

For the prediction method, the number of lags has been set to 20. 10 seconds of the simulated noise data has been used for model fitting and the remaining 20 seconds for model validation.

The training figures also show the computation time spent by MATLAB in fitting the model. This time will of course depend on the system MATLAB is running on[3]. Additionally, the iterative methods are not deterministic, so that estimation using two different data sets may take a different amount of time, even if the model orders and the number of samples are the same. Nevertheless, computation time is included in the figures to give a rough sense of how fast the methods are compared to each other.

Successive Estimation Method

When the parameters of an ARMA(11,4) model are successively estimated using least squares, the resulting test statistic does not quite fit the chi-squared distribution, as shown in Figure 3.12. If we choose to ignore this lack of fit and proceed to calculate p-values for the test data, several of the p-values are very low, even for the part of the signal that has no added sinusoid (Figure 3.13). In other words, the lack of fit increases the risk of false alarms.

As mentioned in Section 2.2.7, the assumption of normality may not be warranted for the parameter vector obtained from the two-stage least squares estimator. In general, when we observe a test statistic that does not fit well with the theoretical distribution, it could mean that some of the underlying assumptions are wrong.

However, experiments have shown that pure AR models tend to produce test statistics with a better fit. If we change the model to AR(7), for instance, the results are as shown in Figures 3.14 and 3.15. When the amplitude of the sinusoid

[3] The system used here is a desktop computer with the following specifications:

i7-6700 CPU
16 GB RAM
500 GB SSD
64-bit Windows 10 Home

Parallelization on four cores has also been employed wherever possible.

is increased to around 110 dB, the successive estimation method is able to detect it using an AR(7) model (Figure 3.16).

Experiments not shown here have further indicated that the test statistic for the successive estimation method will be more sensitive to the introduction of a sinusoid when the model order is higher, *e.g.* AR(20). But then the statistic fits poorly with the chi-squared distribution, which causes more misclassifications as with the ARMA(11,4) model.

A way around this problem could be to use another distribution model for the test statistic. Specifically, experiments have indicated that the generalized extreme value distribution can be used in place of the chi-squared distribution to obtain more well-behaved p-values under the null hypothesis. But this will not be pursued here. Instead, we will restrict our attention to models whose test statistic fits well with the chi-squared distribution.

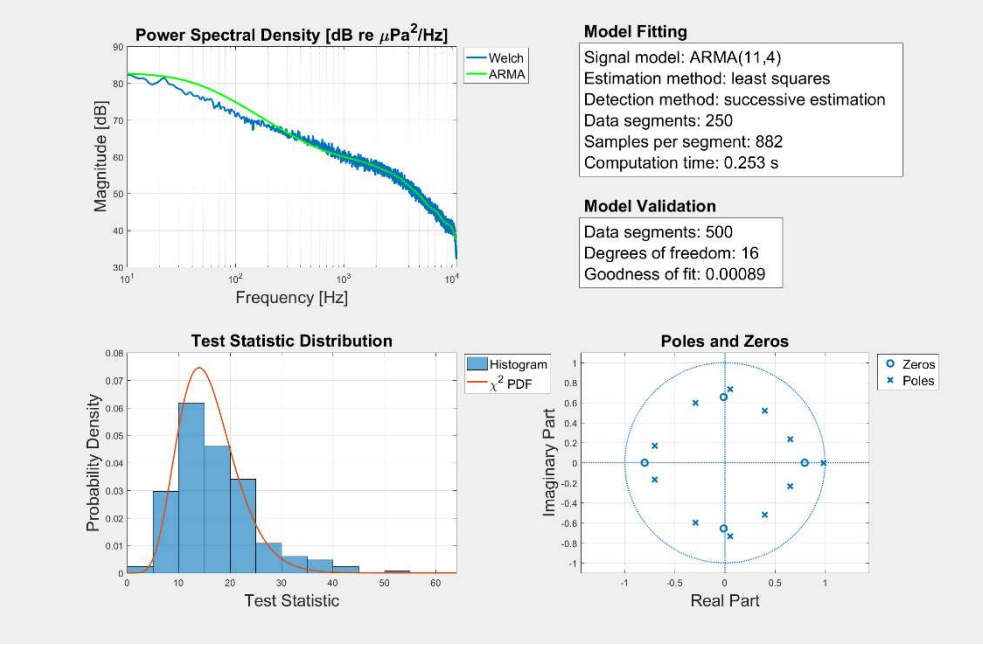

Figure 3.12: Training results for an ARMA(11,4) successive estimation detector.

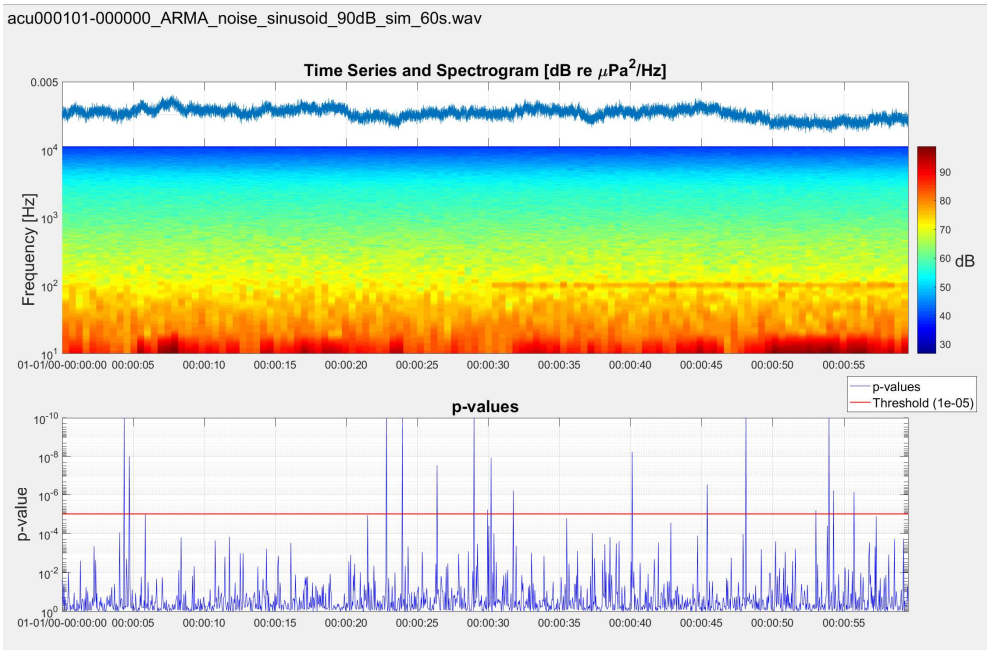

Figure 3.13: p-values for the successive estimation method using an ARMA(11,4) model of the ambient noise. The amplitude of the sinusoid is 90 dB.

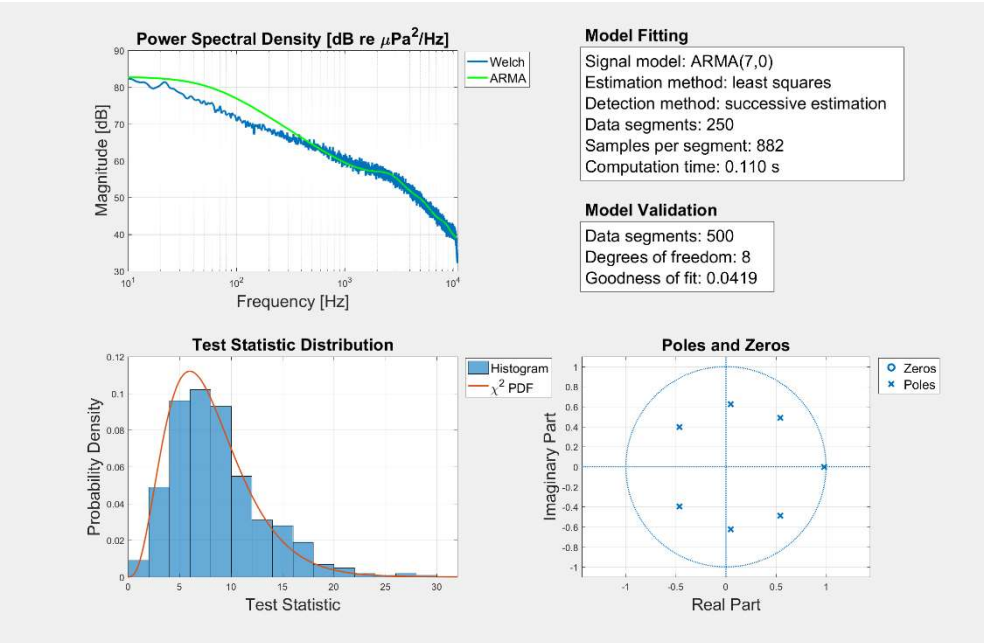

Figure 3.14: Training results for an AR(7) successive estimation detector.

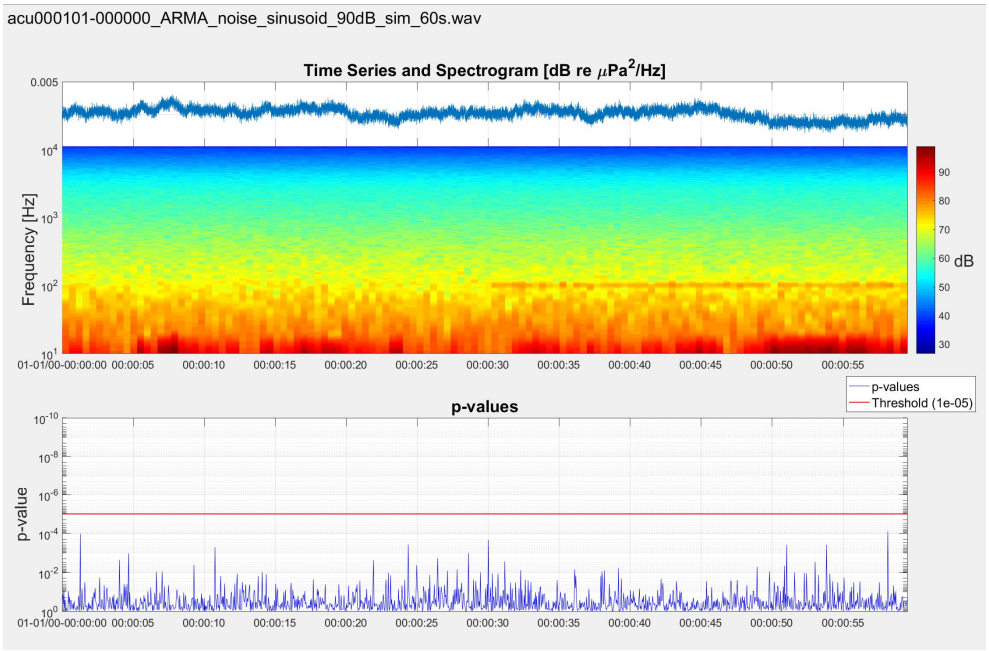

Figure 3.15: p-values for the successive estimation method using an AR(7) model of the ambient noise. The amplitude of the sinusoid is 90 dB.

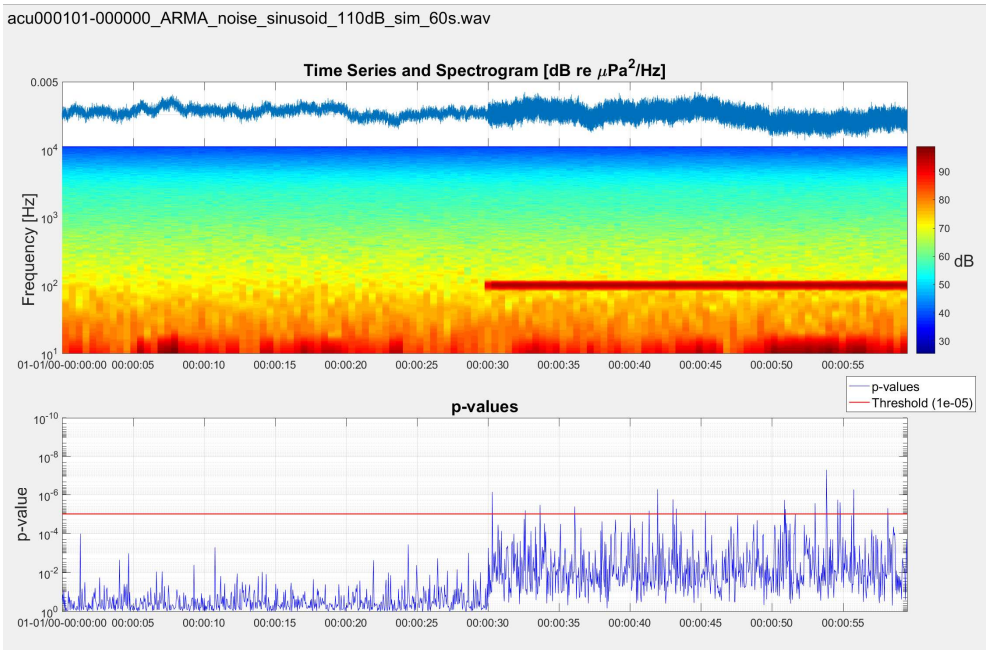

Figure 3.16: p-values for the successive estimation method using an AR(7) model of the ambient noise. The amplitude of the sinusoid is 110 dB.

Prediction method

The prediction method has not been observed to exhibit the same problems with lack of fit. As shown in Figure 3.17, the test statistic fits well with the chi-squared distribution with 20 degrees of freedom. In experiments not shown here, it has been observed to fit well for higher and lower number of lags as well. Detection of the sinusoid does not seem to be affected greatly by how many lags are chosen, so only 20 lags will be shown here and used in most of the experiments on real data as well. When the amplitude of the sinusoid is set to around 98 dB, the prediction method is able to reliably detect it (Figure 3.18).

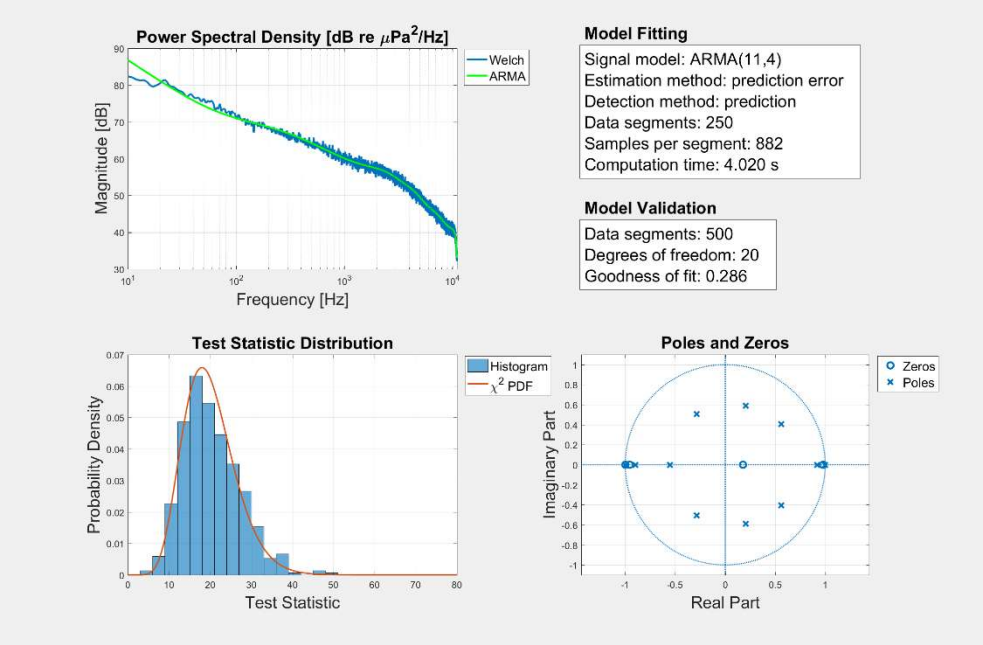

Figure 3.17: Training results for an ARMA(11,4) prediction-based detector.

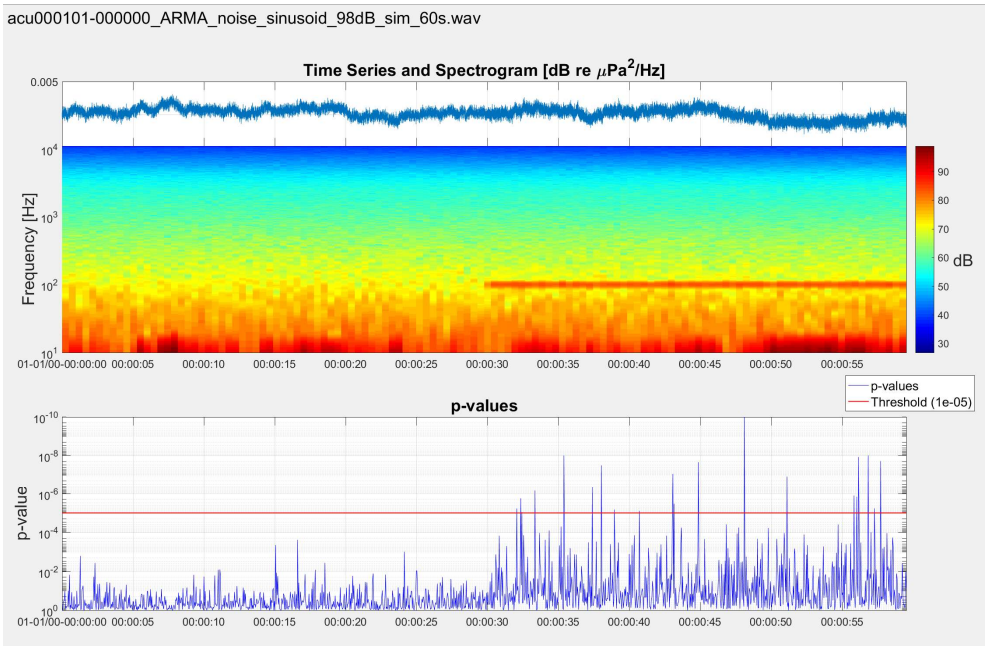

Figure 3.18: p-values for the prediction method using an ARMA(11,4) model of the ambient noise and 20 autocorrelation lags. The amplitude of the sinusoid is 98 dB.

3.4.3 Summary of Results

In the experiments above we have seen that a 100 Hz sinusoid can be detected by the examined methods at the following amplitudes:

Prediction method:	98 dB
Successive estimation method:	110 dB
Energy detector with differencing:	113 dB
Energy detector without differencing:	118 dB

An AR(7) model was used in place of the ARMA(11,4) model for successive estimation, as it produced a test statistic that fitted better with the chi-squared distribution – a necessary condition for obtaining reliable p-values.

3.4.4 Adding Noise to Real Data Sets

In simulated experiments, we are able to control the power of the anomalous signal. But when dealing with real data sets containing both ambient noise and anomalies mixed in, we don't have the same control over the relative power of the ambient noise and the anomalies. Consequently, it is harder on real data sets to evaluate detector performance on fainter anomalies.

We have already seen that that the ARMA-based methods can outperform an energy detector on simulated data. But in order to properly test this on the real data sets, some adjustments will be necessary. The modification that will be

done in the experiments in the next chapter consists of adding simulated noise to real hydrophone signals. This makes it possible to effectively increase the gain of the ambient noise, and thereby lower the signal-to-noise ratio (SNR). In practice, the same effect could of course be achieved by adding real data at different amplitudes to simulated noise with a constant amplitude, but we will not alter the real data, and instead just change the amplitude of the simulated noise.

3.5 Dimensionality Reduction

Because I observed that test statistics from successive estimation with high-order models were quite sensitive to changes in the noise but fitted poorly with the chi-squared distribution, I started to examine ways of fitting a high-order model and subsequently reducing its dimension in the hope that the resulting test statistic would provide a better fit, while retaining the sensitivity to changes of the higher-order model.

This led me to develop an algorithm which uses data both from ambient noise and examples of anomalies to reduce the dimension of the feature space. As such, it is a supervised approach to anomaly detection. The general algorithm is given in Section 3.5.5. Leading up to that, the ideas behind it will first be presented step by step in a less general way.

As Aggarwal (2013) explains in his chapter on supervised anomaly detection, the introduction of anomalous data in the training process, can be used to form a model of the normal class that is more sensitive to certain kinds of anomalies. In the context of ocean monitoring, this could be very useful since it means that we could potentially tune a detector to be more sensitive to a general class of anomalies such as submarines or whales, without having to look for specific acoustic signatures.

3.5.1 Analyzing the Feature Space

When the Mahalanobis distance is used to characterize a feature vector, all the features are assumed to be equally important for describing the underlying raw data. As discussed in Section 2.1.3, this is a strong assumption to make without doing any analysis of the feature space. In the same section, two classical approaches to reducing the dimension of the feature space were introduced, namely principal component analysis and linear discriminant analysis.

Both PCA and LDA seek to summarize the most useful information contained in the feature vector in a lower-dimensional space. The two methods differ in how "useful information" is defined, and in the dimension of the subspace that the feature vector is mapped to. In PCA, an n-dimensional subspace is created that retains the most of the variability of the feature vector, and the dimension n can be chosen freely. In LDA, supervised learning is used to find a one-dimensional subspace that provides optimal separation between feature vectors from two different classes.

The feature space typically has too many dimensions to be visualized, but by dividing it into two-dimensional subspaces, one can get an idea of how data from different classes is distributed for various features.[4] The subspaces will then consist of pairs of features. The features that make up a pair do not need to be related in any particular way. Grouping them together is simply an efficient way of visualizing many features at once.

As an example, when we fitted an ARMA(11,4) model to simulated noise using the successive estimation method in the previous section, the matrix $\mathbf{\Phi}$ contained a set of points in $\mathbb{R}^{16}$, one point for each segment of the signal. Now, assume we let a 118-dB sinusoid in simulated noise represent an example of the type of anomaly we wanted to be able to detect. We could then generate estimates for this signal and produce another matrix $\mathbf{\Phi}'$, also made up of points in $\mathbb{R}^{16}$. The data from both matrices could then be represented graphically in two-dimensional plots where each axis corresponds to a feature. Figure 3.19 shows what such plots would look like. The first feature σ has been given a separate subplot because its scale is quite different from the ARMA coefficients that make up the other features.

As a way of scoring the features, the SNR criterion used to derive Fisher's linear discriminant has been computed for each feature in the model. Eq. (2.29) for the criterion can be rewritten as

$$J(\mathbf{w}) = \frac{(\hat{\mu}_2 - \hat{\mu}_1)^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}, \tag{3.13}$$

where $\hat{\mu}_i$ and $\hat{\sigma}_i$ represent the sample mean and sample variance of the data for the two classes after it has been projected onto $\mathbf{w}$ (Duda et al., 2001). Since the directions we are examining are just the coordinate axes of the feature space, projecting simply means selecting one of the columns of the matrices $\mathbf{\Phi}$ and $\mathbf{\Phi}'$. The SNR value shown in the figures is $\mathrm{SNR}_{\mathrm{dB}}(\phi_i) = 10\log_{10} J(\phi_i)$.

When the features are viewed individually this way, the estimate of the white noise standard deviation σ, is the one that best discriminates between the two classes. It is important to note, however, that this plot does not give much information about the correlations that exist between features. Since the Mahalanobis distance also takes correlation into account, features that have a negative SNR may still contribute useful information if the correlation between them is significantly different for the two classes. Nevertheless, σ is the feature that is the most different for the two classes, and if we had to use only one feature to distinguish between them, σ would be the natural choice.

[4] For the rest of this section, "data" will refer to sets of feature vectors or points in feature space, not the raw data that the feature vectors seek to describe.

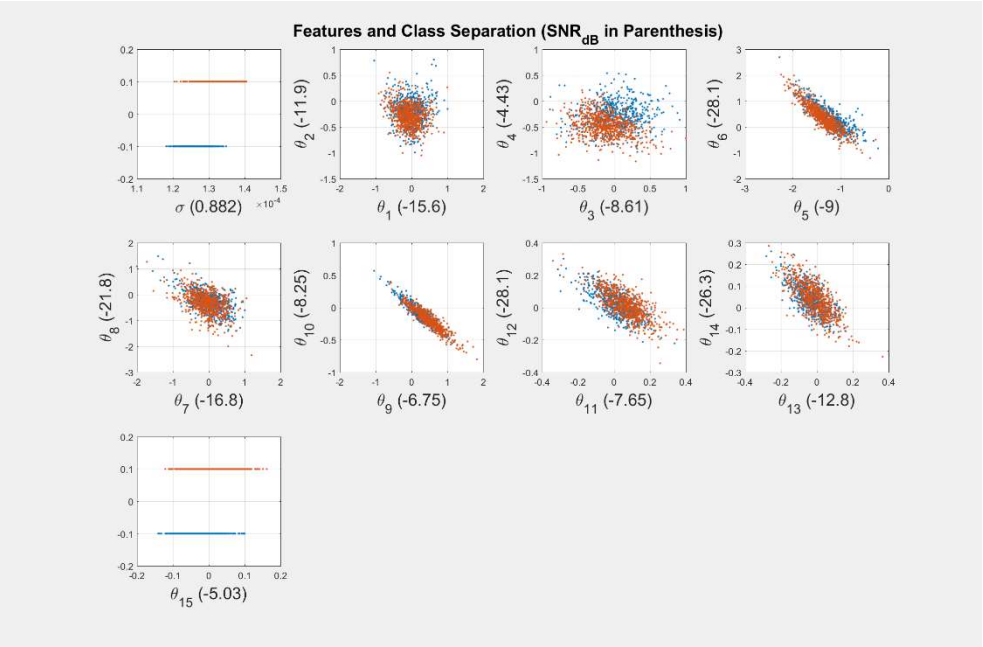

Figure 3.19: Visualization of the feature space of an estimation based detector that uses an ARMA(11,4) model. Each axis represents a parameter of the model. The blue points are estimates of the ambient noise, while the brown points are estimates of an 18-dB sinusoid in noise. In the one-dimensional plots, the points have been drawn on separate lines for ease of visualization.

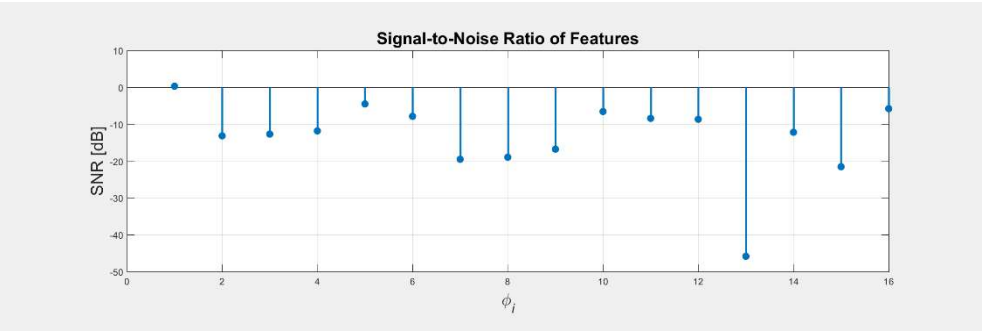

Figure 3.20: Signal-to-noise ratios of each parameter in the ARMA(11,4) model, computed from Fisher's criterion in Eq. (3.13).

3.5.2 Fisher's Linear Discriminant

Even though σ has the highest SNR among the features in the above example, its axis is not necessarily the direction which has the highest SNR of all directions in the feature space. This direction is given by Fisher's linear discriminant:

$$\mathbf{h} = \left(\widehat{\mathbf{\Sigma}}_{\phi} + \widehat{\mathbf{\Sigma}}_{\phi'}\right)^{-1}(\hat{\boldsymbol{\mu}}_{\phi'} - \hat{\boldsymbol{\mu}}_{\phi}) \tag{3.14}$$

When FLD is computed for the data set in this example, it yields a vector with an SNR of 10.4 dB. We cannot visualize this vector, but one way to understand it better is to calculate the angles between the vector and the coordinate axes. The general expression for the angle between two vectors in $\mathbb{R}^d$ is

$$\theta_{\mathbf{xy}} = \cos^{-1}\left(\frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right). \tag{3.15}$$

When this is calculated for FLD and all the coordinate axes, the angle with the first axis is only 0.060 degrees while all the other angles are around 90 degrees. This means that FLD for this example is almost in the same direction as the σ-axis and consequently almost orthogonal to all the other axes. But a small adjustment away from the σ-axis has resulted in an increase in SNR of 9.5 dB. The increased SNR means that the discriminant has absorbed some of the information present in the other features.

3.5.3 Finding an Orthogonal Component

Having found Fisher's linear discriminant in $\mathbb{R}^d$, we might be satisfied and start using that vector to discriminate between classes on new data. However, we could also ask if there is any more information left in the feature space that is not captured by FLD. More precisely: Are there directions that are orthogonal to Fisher's linear discriminant but still have a positive SNR? If we found one such vector, we could construct a two-dimensional subspace spanned by it and FLD that would capture more useful information than what FLD could possibly do alone.

To answer the above question, we first need a way of examining the directions that are orthogonal to FLD. This can be accomplished by employing the theory of projections from linear algebra. Scharf (1991) has been used in the following as a reference for this theory.

Since we are only interested in directions, not magnitudes, of the vectors, we can start by scaling FLD so that $\|\mathbf{h}\| = 1$. This simplifies some of the expressions to follow. Now, form the projection matrix

$$\mathbf{P}_{\mathbf{h}} = \mathbf{h}\mathbf{h}^{\mathrm{T}}. \tag{3.16}$$

When a vector $\boldsymbol{\phi}$ is multiplied by this matrix, it is projected into the one-dimensional subspace spanned by $\mathbf{h}$:

$$\mathbf{P}_{\mathbf{h}}\boldsymbol{\phi} = (\mathbf{h}^{\mathrm{T}}\boldsymbol{\phi})\mathbf{h} \tag{3.17}$$

Furthermore, it is known that any vector in $\mathbb{R}^d$ can be decomposed into two orthogonal components. That is, the vector $\boldsymbol{\phi}$ can be written as

$$\boldsymbol{\phi} = \mathbf{P}_{\mathbf{h}}\boldsymbol{\phi} + \mathbf{P}_{\mathbf{A}}\boldsymbol{\phi}, \tag{3.18}$$

where

$$\mathbf{P}_{\mathbf{A}} = \mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}} \tag{3.19}$$

is a projection onto the space $\langle \mathbf{A} \rangle$ that is orthogonal to $\mathbf{h}$. In other words, there exists a matrix $\mathbf{A}$ whose columns are basis vectors for the orthogonal space, so that $\mathbf{A}^{\mathrm{T}}\mathbf{h} = \mathbf{0}$.

Eq. (3.18) can be written as

$$\boldsymbol{\phi} = (\mathbf{P}_{\mathbf{h}} + \mathbf{P}_{\mathbf{A}})\boldsymbol{\phi}, \tag{3.20}$$

which implies that

$$\mathbf{P}_{\mathbf{h}} + \mathbf{P}_{\mathbf{A}} = \mathbf{I}. \tag{3.21}$$

So, from $\mathbf{h}$ we can form a projection that maps any vector in $\mathbb{R}^d$ to the space that is orthogonal to $\mathbf{h}$ by simply rearranging (3.21):

$$\mathbf{P}_{\mathbf{A}} = \mathbf{I} - \mathbf{P}_{\mathbf{h}}. \tag{3.22}$$

Now the entire data set can be projected into the orthogonal subspace by

$$\begin{aligned} \boldsymbol{\Phi}_{\mathbf{A}} &= \boldsymbol{\Phi}\mathbf{P}_{\mathbf{A}} \\ \boldsymbol{\Phi}'_{\mathbf{A}} &= \boldsymbol{\Phi}'\mathbf{P}_{\mathbf{A}} \end{aligned} \tag{3.23}$$

Remember that we wanted to find a direction in this subspace whose SNR is positive – if such a direction indeed exists. Surely, the most interesting direction to look for is the one that maximizes Fisher's criterion for the projected data set. However, we cannot compute Fisher's linear discriminant directly from (3.14) since that requires estimates of the covariance matrices, and the projected data is currently made up of vectors in $\mathbb{R}^d$ that lie in a subspace with $d-1$ dimensions. As a consequence, the covariance matrices are not full rank and cannot be inverted. To find FLD for the projected data set, we must therefore first represent it in $\mathbb{R}^{d-1}$.

To accomplish this, we can proceed by finding a set of $d-1$ basis vectors for $\langle \mathbf{A} \rangle$ in $\mathbb{R}^d$ and then do a coordinate transformation so that these vectors become the new coordinate axes. First, let

$$\mathbf{\Phi_A} - \mathbf{M} = \mathbf{USV}^{\mathrm{T}} \tag{3.24}$$

be the singular value decomposition of the centered projected data from the normal class. Here, $\mathbf{M}$ is a matrix of repeated sample means (see Section 2.1.2).

Since the rank of $\mathbf{\Phi_A}$ is $d-1$, the last singular value in $\mathbf{S}$ will be zero, and the corresponding eigenvector in $\mathbf{V}$ will be in the direction of Fisher's linear discriminant. This means that the vectors $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{d-1}\}$ are a basis for the orthogonal subspace $\langle \mathbf{A} \rangle$.

Next, form the reduced, full rank matrices

$$\begin{aligned} \mathbf{V}_r &= [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_{d-1}] \\ \mathbf{S}_r &= \mathrm{diag}(s_1, s_2, \cdots, s_{d-1}) \end{aligned} \tag{3.25}$$

and the scaled diagonal matrix

$$\mathbf{D}_r = \frac{\mathbf{S}_r}{\sqrt{N-1}}, \tag{3.26}$$

where N is the number of rows in $\mathbf{\Phi}$. Then perform a whitening transformation similar to the one given in Eq. (2.28) on the whole data set:

$$\begin{aligned} \mathbf{\Psi} &= (\mathbf{\Phi_A} - \mathbf{M})\mathbf{V}_r\mathbf{D}_r^{-1} \\ \mathbf{\Psi}' &= (\mathbf{\Phi}'_{\mathbf{A}} - \mathbf{M})\mathbf{V}_r\mathbf{D}_r^{-1} \end{aligned} \tag{3.27}$$

The matrices $\mathbf{\Psi}$ and $\mathbf{\Psi}'$ have $d-1$ columns representing coordinates along the basis vectors $\{\mathbf{v}_i\}$ of $\langle \mathbf{A} \rangle$. Incidentally, since we also chose to scale by $\mathbf{D}_r^{-1}$, the vectors in $\mathbf{\Psi}$ will be standard normal. That is,

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{\boldsymbol{\psi}} &= \mathbf{0} \\ \widehat{\mathbf{\Sigma}}_{\boldsymbol{\psi}} &= \frac{1}{N-1}\mathbf{\Psi}^{\mathrm{T}}\mathbf{\Psi} = \mathbf{I} \end{aligned} \tag{3.28}$$

Note that only data from the normal class was used to create the transformation, but data from both classes is transformed to the new coordinate system. The reason for this is that the underlying assumption of the whitening transformation is that the data is MVN. In our general model, this can only be assumed for the normal class and not for the pooled data from both classes.

After the data has been transformed to the new coordinate system, we can compute Fisher's linear discriminant in $\langle \mathbf{A} \rangle$ as

$$\mathbf{h_A} = \left(\mathbf{I} + \widehat{\mathbf{\Sigma}}_{\boldsymbol{\psi}'}\right)^{-1} \hat{\boldsymbol{\mu}}_{\boldsymbol{\psi}'}. \tag{3.29}$$

This vector is, however, still a point in the new coordinate system. If we want to use it along with the other linear discriminant to construct a two-dimensional subspace of $\mathbb{R}^d$, we first need to represent it in the original coordinate system by reversing the whitening transformation:

$$\mathbf{h}_2 = \mathbf{V}\mathbf{D}\begin{bmatrix}\mathbf{h}_\mathbf{A} \\ 0\end{bmatrix} + \hat{\boldsymbol{\mu}}_{\boldsymbol{\phi}}, \tag{3.30}$$

where

$$\mathbf{D} = \frac{\mathbf{S}}{\sqrt{N-1}}. \tag{3.31}$$

The vector $\mathbf{h}_2$ is now a vector in $\mathbb{R}^d$ that is orthogonal to Fisher's linear discriminant. Furthermore, of all the orthogonal directions, the direction of $\mathbf{h}_2$ is the one maximizes the SNR criterion for the data set.

If we called FLD $\mathbf{h}_1$ and scaled $\mathbf{h}_2$ so that its norm was also one, we could join the two vectors together in a matrix

$$\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2], \tag{3.32}$$

and form the projection

$$\mathbf{P}_\mathbf{H} = \mathbf{H}\mathbf{H}^\mathrm{T}, \tag{3.33}$$

which would be a mapping into a two-dimensional subspace of $\mathbb{R}^d$ providing, in some sense, optimal separation of the data from the two classes. Projecting a vector $\boldsymbol{\phi}$ this way can be written out as

$$\mathbf{P}_\mathbf{H}\boldsymbol{\phi} = (\mathbf{h}_1^\mathrm{T}\boldsymbol{\phi})\mathbf{h}_1 + (\mathbf{h}_2^\mathrm{T}\boldsymbol{\phi})\mathbf{h}_2, \tag{3.34}$$

which highlights the fact that the $\{\mathbf{h}_i\}$ are basis vectors of the subspace. This means that we can represent the vector $\boldsymbol{\phi}$ in $\mathbb{R}^2$ by simply extracting the coordinates $(\mathbf{h}_i^\mathrm{T}\boldsymbol{\phi})$ for each of the two axes. The resulting transformation is then

$$\mathcal{T}\colon \mathbb{R}^d \to \mathbb{R}^2\colon \mathbf{x} = \mathbf{H}^\mathrm{T}\boldsymbol{\phi}. \tag{3.35}$$

If $\boldsymbol{\phi}$ is assumed to be MVN, then $\mathbf{x}$ will be distributed as

$$\mathbf{x} \sim N(\mathbf{H}^\mathrm{T}\boldsymbol{\mu}_{\boldsymbol{\phi}}, \mathbf{H}^\mathrm{T}\boldsymbol{\Sigma}_{\boldsymbol{\phi}}\mathbf{H}), \tag{3.36}$$

which follows from the properties of linear transformations of MVN variables given in (2.7). When the transformation in (3.35) is performed on a data matrix, it becomes

$$\mathcal{T}\colon \mathbb{R}^{N\mathrm{x}d} \to \mathbb{R}^{N\mathrm{x}2}\colon \mathbf{X} = \boldsymbol{\Phi}\mathbf{H}. \tag{3.37}$$

After transforming the data, we can use Mahalanobis distances in $\mathbb{R}^2$ to classify new data in the same way as was done in $\mathbb{R}^d$ before the transformation.

3.5.4 Example: Mapping a Data Set from $\mathbb{R}^3$ to $\mathbb{R}^2$

To get a more intuitive understanding of what is going on in the above procedure, we can try to visualize it on a 3-dimensional data set. For simplicity, the data will be generated by drawing points from two Gaussian distributions. The parameters of the first distribution are given by

$$\boldsymbol{\mu} = \begin{bmatrix} -3.0 \\ -4.0 \\ 1.0 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1.2 & 0.3 & -0.4 \\ 0.3 & 1.5 & 0.2 \\ -0.4 & 0.2 & 2.0 \end{bmatrix}. \tag{3.38}$$

This will represent the normal class. The parameter of the second distribution, representing the anomaly class, are

$$\boldsymbol{\mu}' = \begin{bmatrix} 3.0 \\ 4.0 \\ 5.0 \end{bmatrix}, \quad \boldsymbol{\Sigma}' = \begin{bmatrix} 2.0 & -0.4 & 0.2 \\ -0.4 & 1.0 & 0.3 \\ -0.2 & 0.3 & 1.5 \end{bmatrix}. \tag{3.39}$$

From each of these distributions, 5000 points have been drawn and organized as rows of the data matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}'$.

Figure 3.21 shows the data set plotted as points in $\mathbb{R}^3$ along with the subspace $\langle \mathbf{h}_1 \rangle$ spanned by FLD and the subspace $\langle \mathbf{A} \rangle$ orthogonal to FLD. We see that in three dimensions, $\langle \mathbf{A} \rangle$ becomes a plane. The direction of $\mathbf{h}_1$ seems to make sense since it is roughly the direction $\boldsymbol{\mu}' - \boldsymbol{\mu}$, pointing from one mean to the other.

In Figure 3.22 the data has been projected onto the plane $\langle \mathbf{A} \rangle$ and the direction of $\mathbf{h}_2$ has been found as the direction that maximizes the SNR criterion in the subspace. We can imagine that the vector has been rotated around in the plane until the SNR is at its highest.

Figure 3.23 shows the original data set and the subspace $\langle \mathbf{H} \rangle$ spanned by $\mathbf{h}_1$ and $\mathbf{h}_2$, while figure 3.24 shows the data set projected into $\langle \mathbf{H} \rangle$. The data points are still points in three dimensions even though they are in a subspace. But by doing a coordinate transformation they can be represented in two dimensions, as shown in figure 3.25.

It is clear that the SNR is much better for $\mathbf{h}_1$ than $\mathbf{h}_2$. But $\mathbf{h}_2$ is still positive, which means it contains some information not captured by $\mathbf{h}_1$. To create an alternative but suboptimal representation of the data set we could find a vector $\mathbf{h}_3$ that is orthogonal to both $\mathbf{h}_1$and $\mathbf{h}_2$. That is, a vector that satisfies

$$\mathbf{H}^{\mathrm{T}}\mathbf{h}_3 = \mathbf{0}. \tag{3.40}$$

Geometrically, this is the normal vector to the plane spanned by $\mathbf{h}_1$ and $\mathbf{h}_2$. It can be found from an eigenvalue decomposition of $\mathbf{H}^{\mathrm{T}}$.

The result of representing the data by means of $\mathbf{h}_2$ and $\mathbf{h}_3$ is shown in Figure 3.26. It is clear that projecting the data onto $\mathbf{h}_3$ would cause the data from the two classes to be mixed together almost completely, as evidenced by the negative SNR. An explanation for this is that the first two directions have been formed in a way that maximizes our ability to classify the projected data. Consequently, less useful information for classification becomes available for the last direction. This could perhaps be viewed as entropy, or lack of information, being forced into the last orthogonal direction.

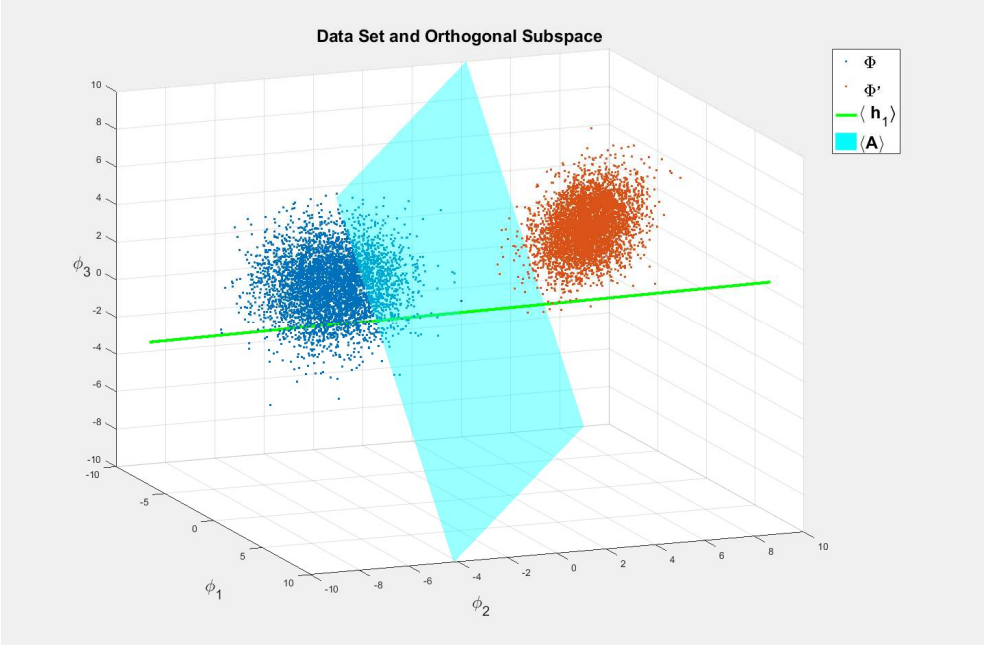

Figure 3.21: Two Gaussian clusters representing sets of feature vectors from two different classes. The green line is the subspace spanned by Fisher's linear discriminant. The blue plane is the subspace orthogonal to FLD.

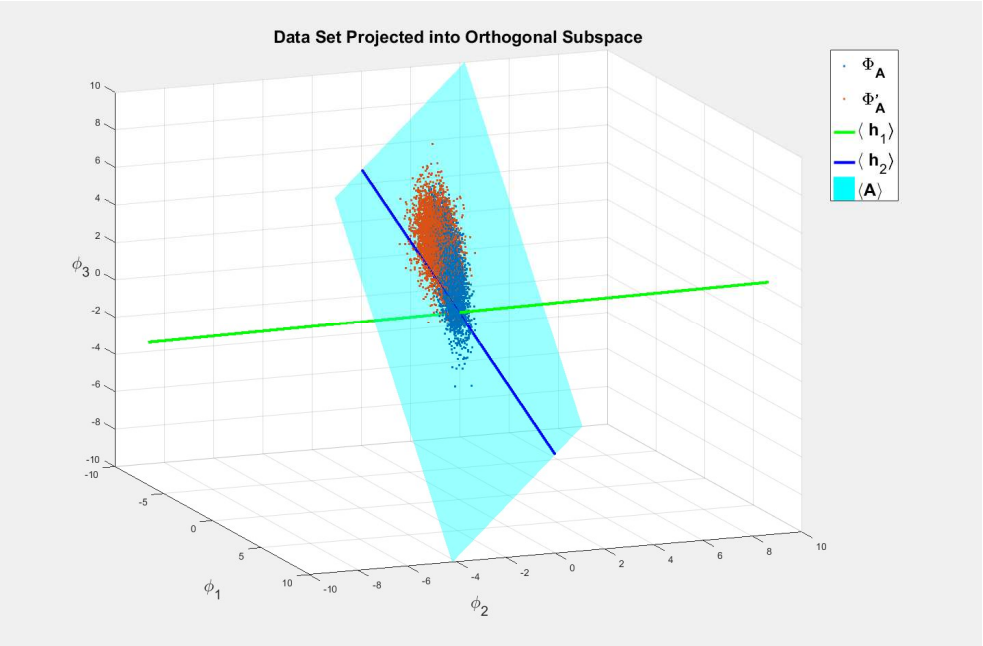

Figure 3.22: When all the feature vectors are projected into the orthogonal subspace, the direction of the blue line is the one that maximizes the SNR criterion and provides the best separation of the two classes in the orthogonal subspace.

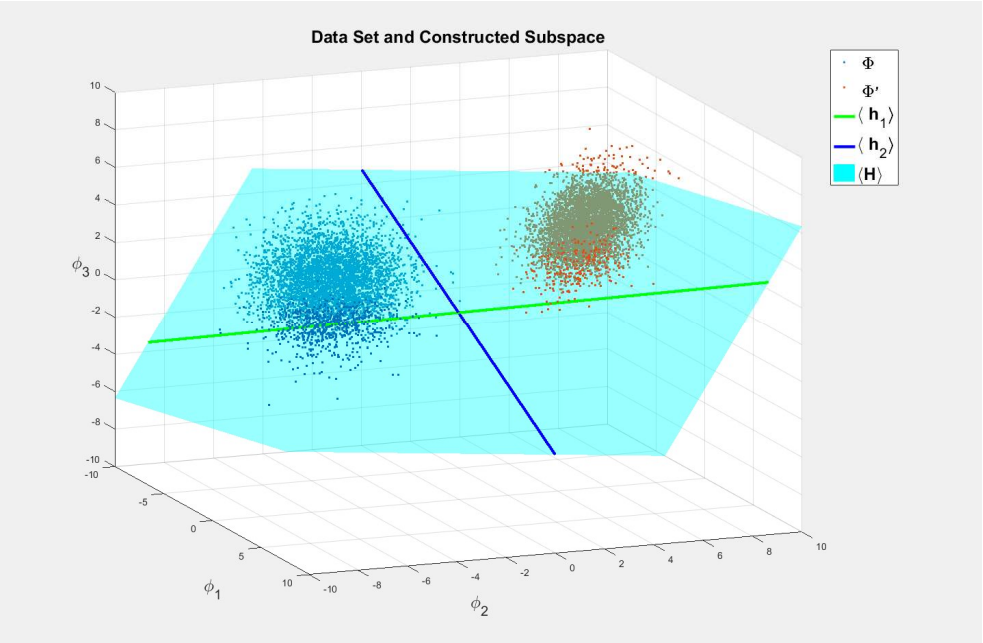

Figure 3.23: The feature vectors from the two classes shown with the subspace constructed from the two orthogonal components.

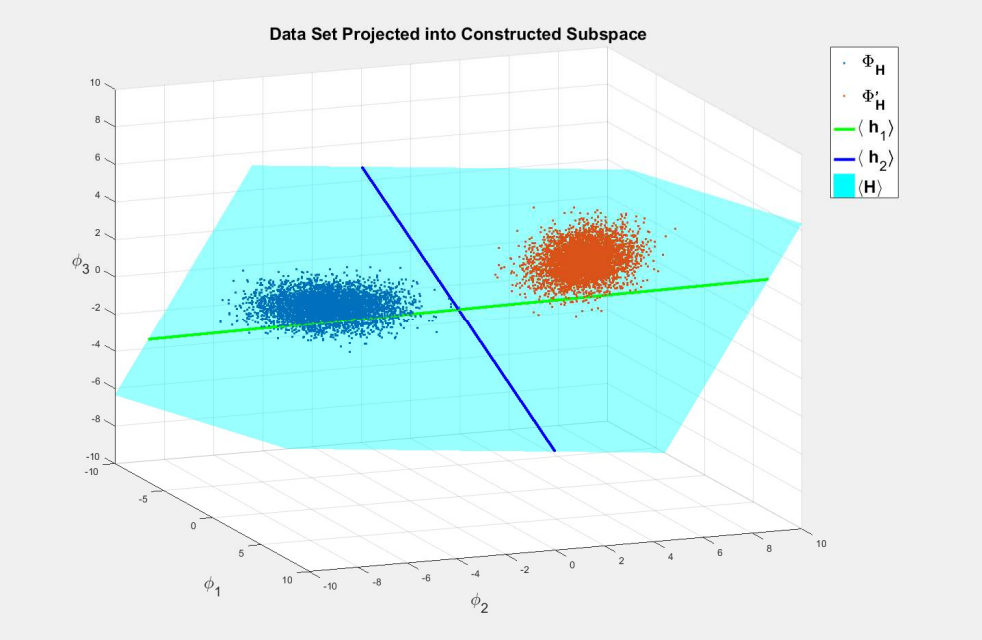

Figure 3.24: The feature vectors from the two classes projected into the constructed subspace.

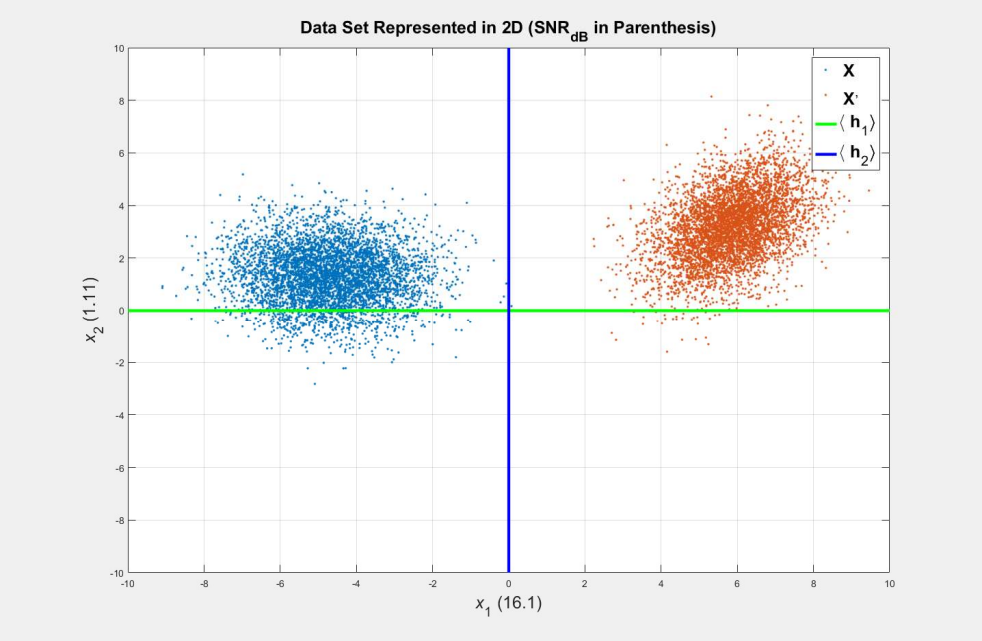

Figure 3.25: The projected feature vectors from the two classes represented in two dimensions.

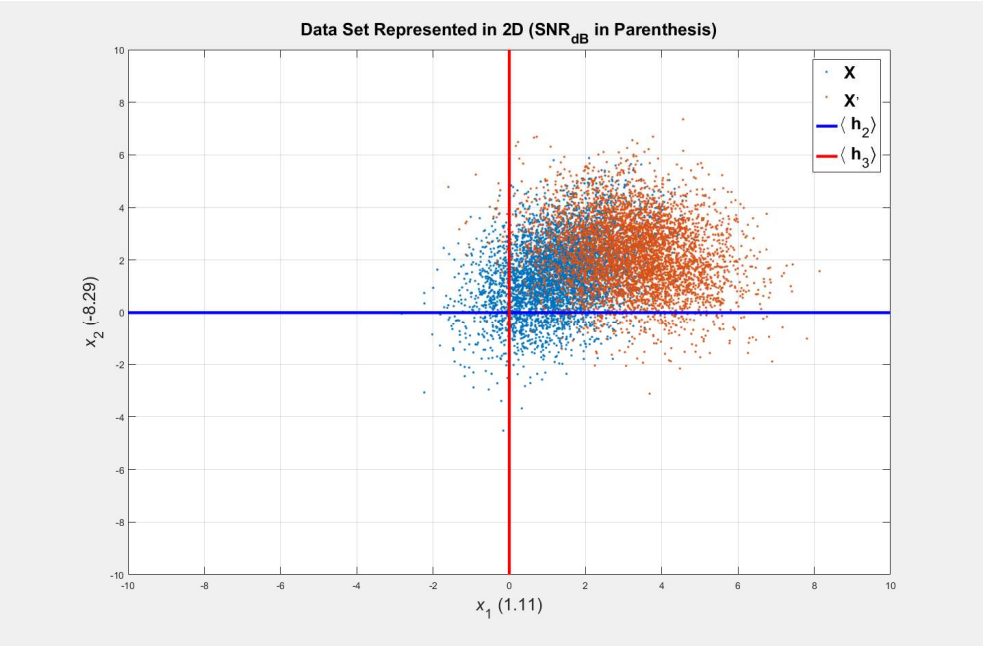

Figure 3.26: An example of a worse mapping where the feature vectors from the two classes are not well separated.

3.5.5 Finding Multiple Orthogonal Components

When working in a high-dimensional feature space, there is no need to stop after two orthogonal components have been found from the procedure described in the previous section. The method can fairly easily be generalized to find n orthogonal components in $\mathbb{R}^d$, where $n \leq d$. We just have to make sure that each new component is not only orthogonal to the preceding one, but to all previous components. That means that the projection matrix used to find the ith component $\mathbf{h}_i$, must be formed as

$$\mathbf{P}_{\mathbf{A}} = \mathbf{I} - \mathbf{H}_i \mathbf{H}_i^{\mathrm{T}}, \tag{3.41}$$

where

$$\mathbf{H}_i = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \cdots \quad \mathbf{h}_{i-1}]. \tag{3.42}$$

We can then project the data as before, and rotate a vector around in the orthogonal subspace until Fisher's criterion is maximized. Afterwards, we can represent the vector in $\mathbb{R}^d$ as $\mathbf{h}_i$ and move on to the next component.

Each component found this way will necessarily have a lower SNR than the preceding one. To see why this is the case, note that the directions we are searching through to find $\mathbf{h}_i$ are only a subset of the directions we have already searched through to find $\mathbf{h}_{i-1}$. So, $\mathbf{h}_1$ will be found by looking at the entire space, and will thus be Fisher's linear discriminant. $\mathbf{h}_2$ will then be formed by only considering the directions that are orthogonal to $\mathbf{h}_1$, and so on.

The fact that we end up with a set of ordered orthogonal components that can be used to represent the data in a lower-dimensional space, means that the procedure has a lot in common with PCA. The big difference is, of course, that the components are found by iteratively maximizing Fisher's SNR criterion under an orthogonality constraint, while in PCA it is the variance of the projected data that is maximized under an orthogonality constraint.

In the general version of the algorithm given below, one modification will be introduced that serves to simplify some of the steps. It consists of creating a whitening transformation based on data from the normal class and performing it on data from both classes before proceeding to find the components. This effectively represents the whole data set in a coordinate system where the distribution of the normal class is standard normal.

An advantage of the whitening transformation is that any time the data from the normal class is projected into a new subspace, the mean vector remains zero, which means it doesn't have to be recalculated on each iteration. The symmetry of the spherical distribution further ensures that the variance along any vector in the subspace is equal to one, which means we don't have to scale by the singular values when moving in and out of the subspace. The initial whitening transformation is done using the SVD in steps $1-3$ below. If the true mean and covariance matrix is known in advance, the transformation can instead be done by factoring the covariance matrix, as in (2.13).

Algorithm for finding n orthogonal components

1. Take the SVD of the data from the normal class:

$$\mathbf{\Phi} - \mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}, \tag{3.43}$$

   where the rows of $\mathbf{M}$ are all equal to $\hat{\boldsymbol{\mu}}^{\mathrm{T}}$, *i.e.* the sample mean of $\boldsymbol{\phi}$.

2. Form the diagonal matrix

$$\mathbf{D} = \frac{\mathbf{S}}{\sqrt{N-1}}, \tag{3.44}$$

   where N is the number of rows in the data matrix $\mathbf{\Phi}$.

3. Transform the data from both classes.

$$\begin{aligned} \mathbf{W} &= (\mathbf{\Phi} - \mathbf{M})\mathbf{V}\mathbf{D}^{-1} \\ \mathbf{W}' &= (\mathbf{\Phi}' - \mathbf{M})\mathbf{V}\mathbf{D}^{-1} \end{aligned} \tag{3.45}$$

4. Find the first component (FLD)

$$\mathbf{h}_1 = \left(\mathbf{I} + \widehat{\mathbf{\Sigma}}_{\mathbf{w}'}\right)^{-1} \hat{\boldsymbol{\mu}}_{\mathbf{w}'}. \tag{3.46}$$

5. Scale the component so that $\|\mathbf{h}_1\| = 1$.

6. Initialize $\mathbf{H}_1$ as an empty matrix

$$\mathbf{H}_1 = [\ \ ] \tag{3.47}$$

Repeat the following steps for $i = 2, 3, \cdots, n$.

7. Update $\mathbf{H}_i$

$$\mathbf{H}_i = [\mathbf{H}_{i-1} \quad \mathbf{h}_{i-1}] \tag{3.48}$$

8. Form the projection matrix

$$\mathbf{P}_{\mathbf{A}} = \mathbf{I} - \mathbf{H}_i \mathbf{H}_i^{\mathrm{T}} \tag{3.49}$$

9. Project the data into the orthogonal subspace $\langle \mathbf{A} \rangle$.

$$\begin{aligned} \mathbf{W}_{\mathbf{A}} &= \mathbf{W}\mathbf{P}_{\mathbf{A}} \\ \mathbf{W}'_{\mathbf{A}} &= \mathbf{W}'\mathbf{P}_{\mathbf{A}} \end{aligned} \tag{3.50}$$

10. Take the SVD of the projected data from the normal class.

$$\mathbf{W}_{\mathbf{A}} = \mathbf{\Omega}\mathbf{\Lambda}\mathbf{\Upsilon}^{\mathrm{T}} \tag{3.51}$$

11. Form the reduced matrix

$$\mathbf{\Upsilon}_r = [\gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_{d-i+1}]. \tag{3.52}$$

12. Represent the projected data from the second class in $\mathbb{R}^{d-n+1}$.

$$\mathbf{\Psi}' = \mathbf{W}_{\mathbf{A}} \mathbf{\Upsilon}_r \tag{3.53}$$

13. Find Fisher's linear discriminant in the new coordinate system.

$$\mathbf{h}_{\mathbf{A}} = \left(\mathbf{I} + \widehat{\mathbf{\Sigma}}_{\boldsymbol{\psi}'}\right)^{-1} \hat{\boldsymbol{\mu}}_{\boldsymbol{\psi}'}. \tag{3.54}$$

14. Represent $\mathbf{h}_{\mathbf{A}}$ in $\mathbb{R}^d$ as

$$\mathbf{h}_i = \mathbf{\Upsilon} \begin{bmatrix} \mathbf{h}_{\mathbf{A}} \\ \mathbf{0} \end{bmatrix}. \tag{3.55}$$

15. Scale the component so that $\|\mathbf{h}_i\| = 1$.

When the loop is complete, the matrix $\mathbf{H}_n = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \cdots \quad \mathbf{h}_n]$ is returned along with $\hat{\boldsymbol{\mu}}$ and the matrices $\mathbf{V}$ and $\mathbf{D}$ that were used for the initial whitening transformation. New feature vectors can then be mapped to $\mathbb{R}^n$ with the transformation

$$\mathcal{T}_{\boldsymbol{\phi}}\colon \mathbb{R}^d \to \mathbb{R}^n\colon \mathbf{x} = \mathbf{H}_n^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{V}^{\mathrm{T}}(\boldsymbol{\phi} - \hat{\boldsymbol{\mu}}), \tag{3.56}$$

which for a data matrix becomes

$$\mathcal{T}_{\boldsymbol{\Phi}}\colon \mathbb{R}^{N\mathrm{x}d} \to \mathbb{R}^{N\mathrm{x}n}\colon \mathbf{X} = (\boldsymbol{\Phi} - \mathbf{M})\mathbf{V}\mathbf{D}^{-1}\mathbf{H}_n. \tag{3.57}$$

We are, of course, free to create a matrix $\mathbf{T}_n = \mathbf{V}\mathbf{D}^{-1}\mathbf{H}_n$ and just use that for future transformations instead of keeping the three individual matrices.

In the algorithm above, the data is first mapped to a coordinate system in which the normal class has a spherical distribution. Multiplication by $\mathbf{H}_n$ then rotates the data while simultaneously reducing its dimension. It is possible to get back to the original coordinate system by reversing the whitening transformation, but if the goal is to calculate Mahalanobis distances this would be unnecessary. The reason is that the distribution of the mapped data is multivariate standard normal, and as we have seen, this reduces Mahalanobis distances to Euclidian distances. So, if we stay in the new coordinate system, we don't need a covariance matrix or a mean vector to calculate the Mahalanobis distance. We can simply take the Euclidian norm of the transformed vector **x.**

One might wonder why the covariance matrix of the transformed vector is still the identity after multiplication by $\mathbf{H}_n^{\mathrm{T}}$. This follows from the fact that $\mathbf{H}_n$ is orthogonal, and from the properties of linear transformations of MVN variables in Eq. (2.7):

$$E[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \mathbf{H}_n^{\mathrm{T}}\mathbf{I}_d\mathbf{H}_n = \mathbf{I}_n \tag{3.58}$$

Even though the algorithm can be stopped after n components have been found, it is usually more practical to set $n = d$ and find the whole set of components that span $\mathbb{R}^d$. With a little extra bookkeeping, one can also return the SNR of each component. It is possible, then, to do some analysis before deciding how many components to use. For instance, one could elect to use only components that have an SNR above some threshold (like zero). Having all the components available also makes it easier to run tests using different numbers of components on the same test data. The function `findBasis` listed in the Appendix is an implementation of the above algorithm that finds the whole set of basis vectors and also returns the SNR of each component.

Returning to the Simulated Example

At the beginning of this section on dimensionality reduction, we examined the feature space of an ARMA(11,4) model fitted to simulated ambient noise. We saw that the SNR of the different features varied a lot, with the first feature σ being the best for discriminating between the two classes.

Now, if the general algorithm for finding orthogonal components is applied to the data from that example, the results are as shown in Figures 3.27 and 3.28. There we clearly see how the components are ordered according to their SNR. Interestingly, there are now two components with a quite high SNR. In the next chapter, examples will be given where the two signals have less in common and even more components have a positive SNR.

Recall from Section 3.4.2 that validation of the ARMA(11,4) based detector produced a poor fit with the chi-squared distribution, resulting in unstable p-values under the null hypothesis. When we test this detector on the 118-dB sinusoid, the result is as shown in Figure 3.29. Even the lack of fit produces several false alarms, the detector clearly reacts to the introduction of the loud sinusoid.

Now, if we instead build a detector using only the first component, *i.e.* Fisher's linear discriminant, the GOF actually goes up to 0.39. When this detector is tested on the same file, the result is as shown in Figure 3.30. Two things are evident from the figure. Firstly, although there are still a few false alarms, the p-values are more well-behaved under the null hypothesis than before the mapping. Secondly, the p-values remain consistently low after the sinusoid is introduced.

If we decided to use two instead of one component, the result is as shown in Figure 3.31. This appears to increase the variance of the p-values somewhat but they are still consistently low for the second half of the signal. It is not clear from this example that adding another component has any benefit. In fact, it happens to produce one more false alarm than for just one component.

I have not conducted any systematic experiments to determine whether adding more components is beneficial in general. One way to test this, would be to introduce several examples of the same type of anomaly, *e.g.* different types of ship, in the training stage, and see if using more than one component in the test stage would improve the detectors ability on unseen anomalies of the same kind.

In the above example, the sinusoid used in detector training is the same as the one used to test the detector, but the ambient noise is generated from independent simulations of the same ARMA process. Nevertheless, the test does not show how the detector would react to other types of anomalies, such as sinusoids at different amplitudes. In the next chapter, we will see some examples of this when the detectors are tested on real hydrophone signals. Before proceeding, we will briefly return to the detector design procedure from Section 3.2 to see how supervised reduction of dimensionality can be incorporated there.

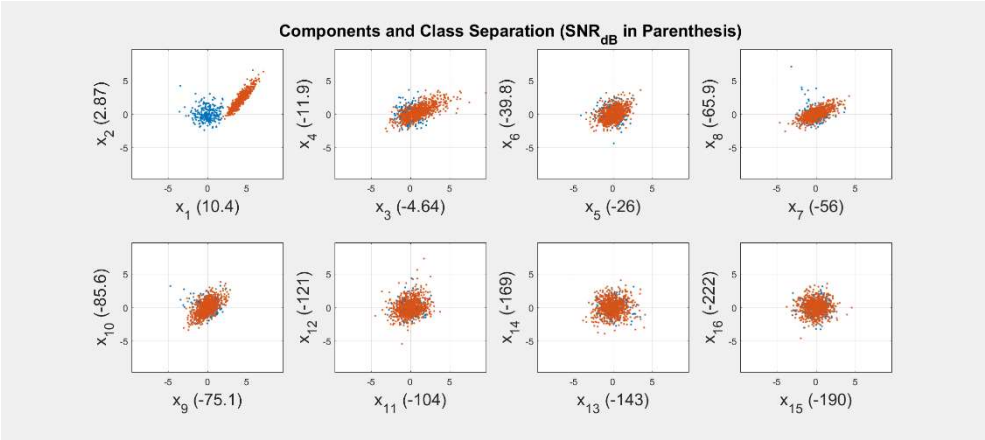

Figure 3.27: The feature space of the ARMA(11,4) example after the mapping has been applied.

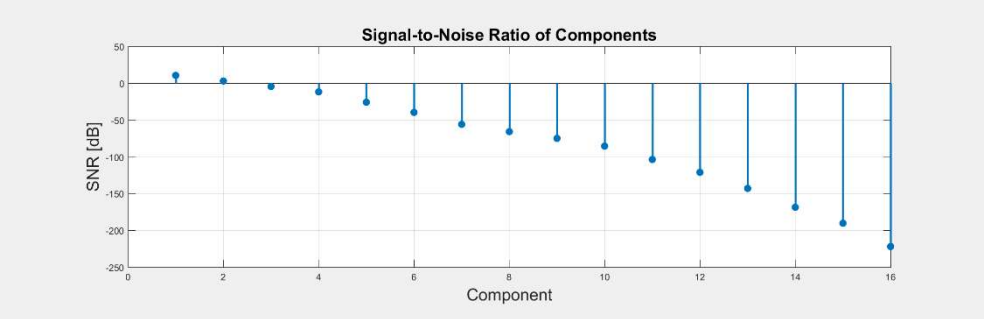

Figure 3.28: The SNR of all the components after the mapping has been applied.

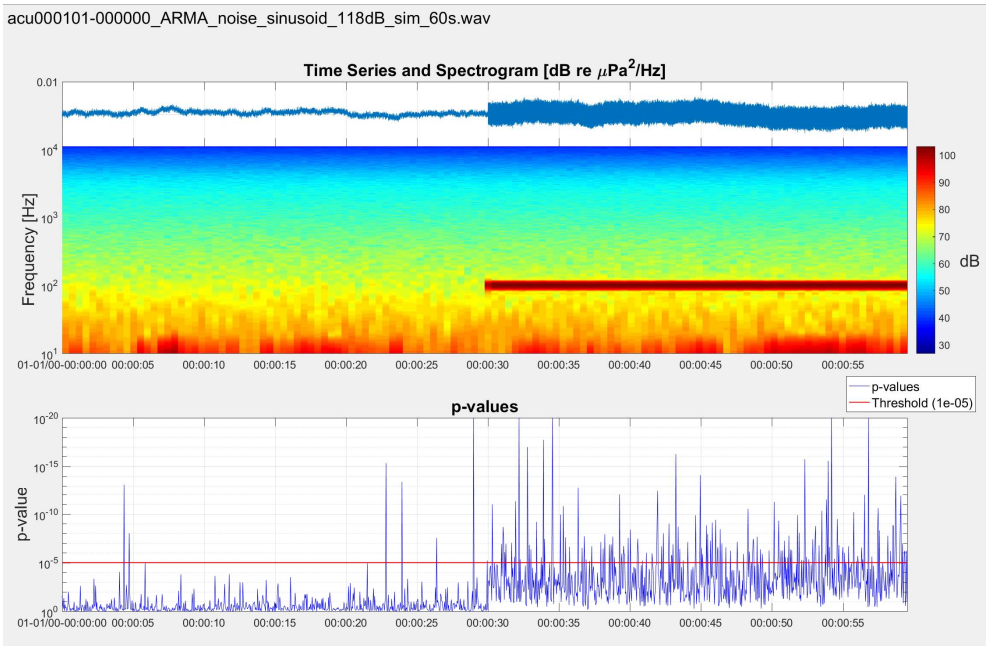

Figure 3.29: p-values calculated from Mahalanobis distances in the original feature space of the ARMA(11,4) model. The low GOF, as observed in Figure 3.12, causes unstable p-values under the null hypothesis.

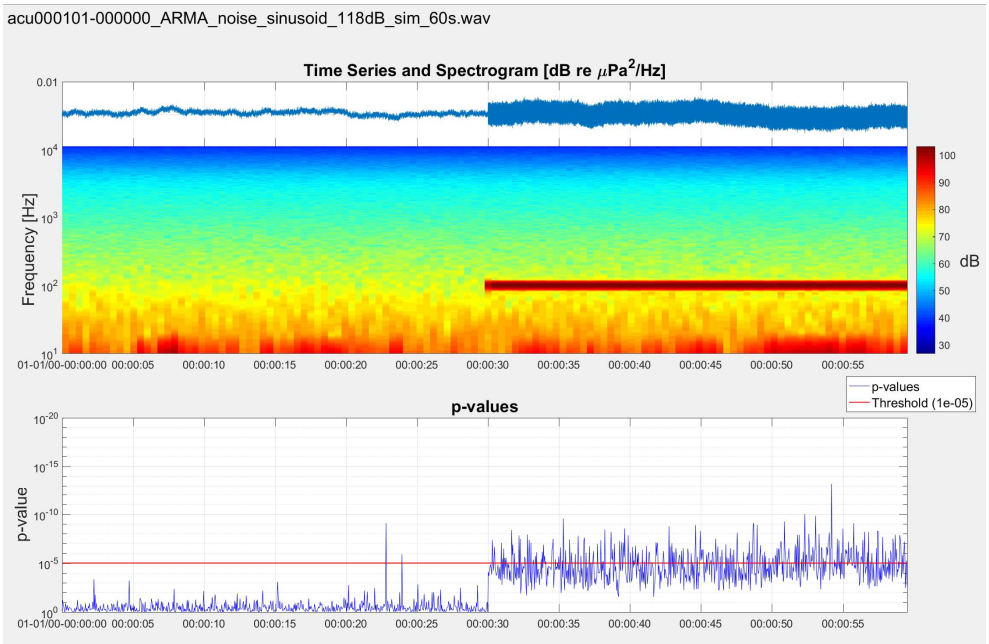

Figure 3.30: p-values calculated from distances along the first component, FLD. Training with this detector produced a GOF p-value of 0.39.

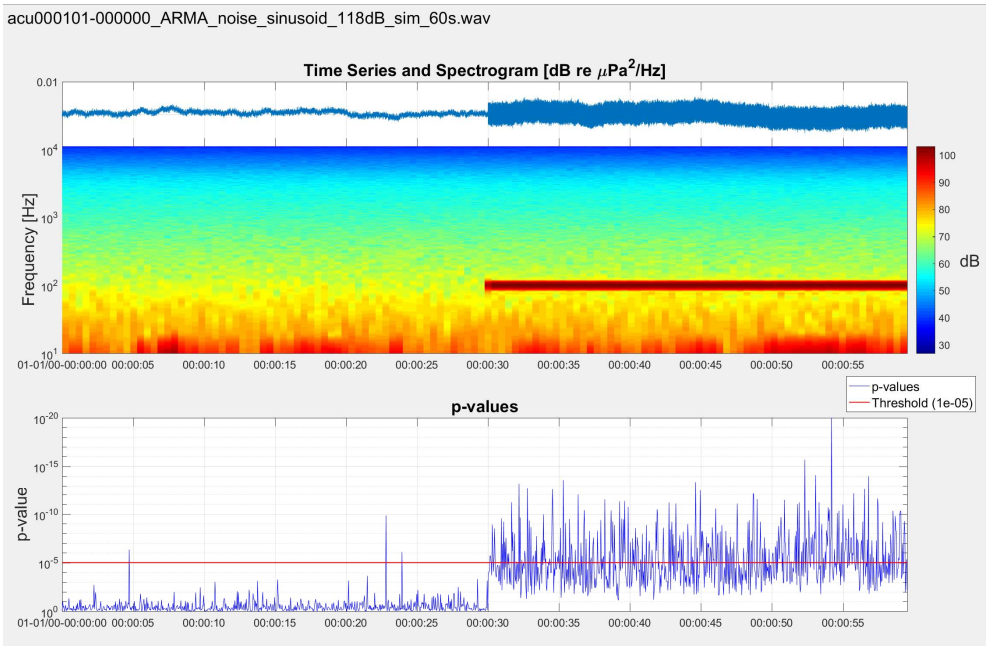

Figure 3.31: p-values calculated from Mahalanobis distances in the two-dimensional space spanned by the two components with the highest SNR. Training of this detector produced a GOF p-value of 0.43.

3.6 Supervised Detection

The dimensionality reduction algorithm in the previous section is a way of allowing the model of the normal state to be made more sensitive to certain types of anomalies. If examples of anomalies of specific interest are available, they can be fed to the reduction algorithm which then learns a linear transformation that makes the detector respond more to those types of anomalies. Whether there is any real benefit to using more components than just Fisher's linear discriminant in this setting, is unclear. Further work is necessary to determine the usefulness of the general algorithm.

Nevertheless, supervised reduction of dimensionality can be incorporated in the detector design approach outlined in Section 3.2. As shown in Figures 3.32 and 3.33, the algorithm for finding the orthogonal components is now part of the training procedure. The transformation matrix obtained can then be stored and used to map new feature vectors.

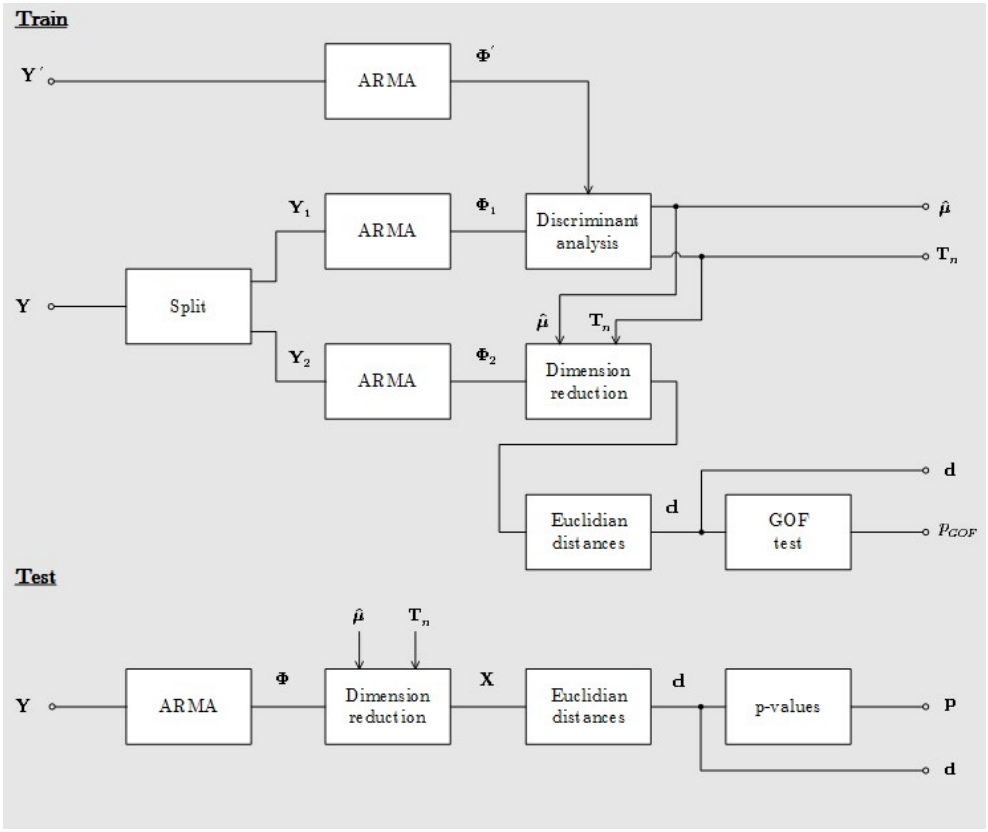

Figure 3.32: Overview of the detector training stage and test stage using the successive estimation method and supervised dimensionality reduction.

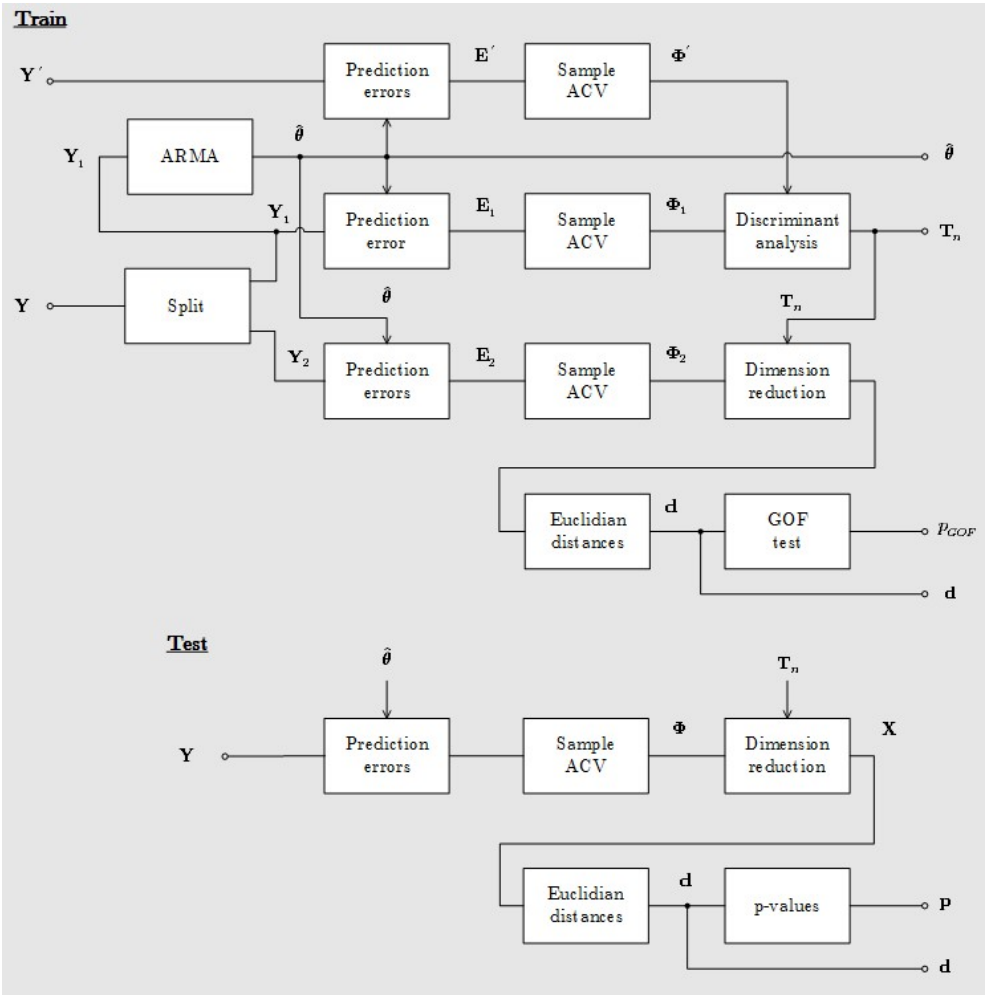

Figure 3.33: Overview of the detector training stage and test stage using the prediction method and supervised dimensionality reduction.

4 Experiments on Hydrophone Data

On the following pages, the methods described in the previous chapter will be tested on a data set provided by FFI. The data set consists of an 83 seconds long audio file made up of four different signals following each other. All the signals are 22.05 kHz hydrophone recordings from LoVe. The first signal is ambient noise recorded at 3 m/s wind speed. The second consists of the 10 seconds of 18 m/s ambient noise modeled in Section 3.3, plus five more seconds or so of the same noise. The third is ship noise, and the fourth is ambient noise at unknown wind speed interspersed with whale sounds.

The two detection methods will be tested in both their unsupervised and supervised form as laid out in Sections 3.2 and 3.6. The data that will be used to train the detectors is the 10 first seconds of the 18 m/s wind noise signal, as in Section 3.3. The remainder of the signal has been reserved for model validation.

In the supervised case, the ship noise will be taken as the example of anomalous data. This signal is around 13 seconds long. Only the first half will be used for training the detector, so that when we go through the test file, the second part of that signal will consist of unseen data.

The segment length chosen for most of the experiments is 0.04 seconds, meaning that each data segment consists of only 882 samples. This is probably a much higher temporal resolution than what is necessary in any practical situation for detecting ships and other sound sources in the ocean. In such a situation, one might prefer to only update the p-values once every second or so.

The segment length has been chosen this way because of the limited amount of data available for the experiments. Since successive estimation requires estimates of the mean and covariance matrix for the parameter vector, we need to obtain a fair number of samples of this vector during training. Checking the goodness of fit of the test statistic also requires more than just a few samples of the statistic.

In order to create more challenging situations for the detectors, simulated ambient noise of varying amplitudes will be added to the training data and the test data in some of the experiments. The model used for simulation is the one given in (3.11). The amplitude of the noise will be given in relation to the 18 m/s wind noise, so that a noise gain of K dB means that the added noise increases the total power of that signal by K dB.

The noise added to the training data and the test data will come from independent simulations of the ARMA model. But the random number generator is initialized in the same way for each experiment so that the added noise for different gains are scaled versions of the same simulated signals.

4.1 No Added Noise

4.1.1 Energy Detector

We start by running an energy detector on the unaltered test file. As shown in Figure 4.1, even though the detector is sensitive to the anomalies, it clearly varies a lot even in the areas where we are assuming that the signal is stationary. In Section 3.4.1 we saw that differencing the signal before computing mean square values produced more stable values. When the same is done here, the result is as shown in Figure 4.2. Clearly, the differenced version of the detector is much more stable. It will therefore be used for comparing the energy detector with the ARMA-based methods in the next experiment.

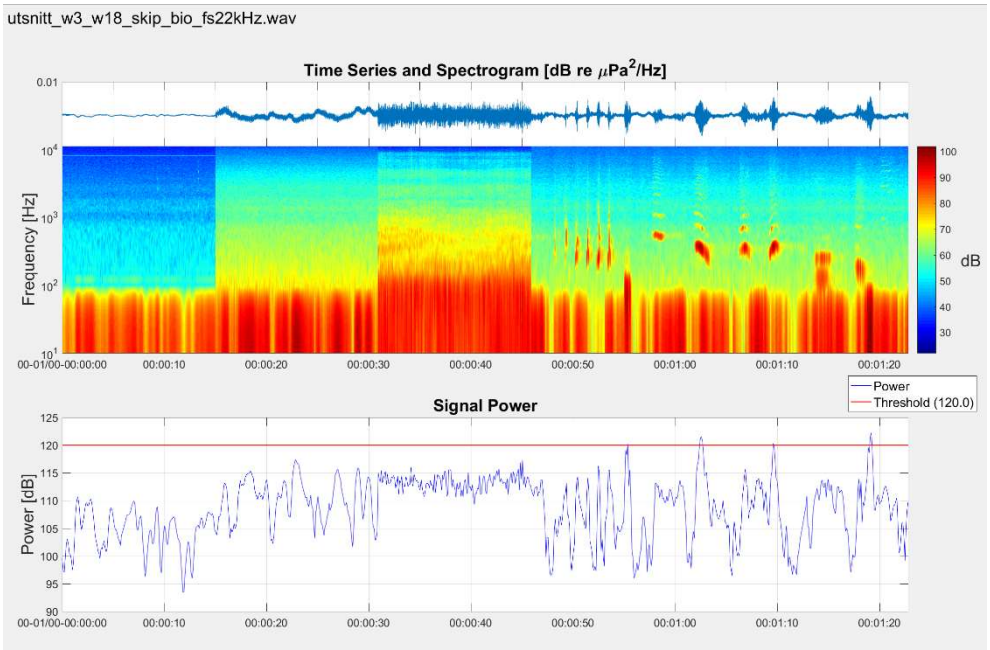

Figure 4.1: Energy detector tested on the sample file.

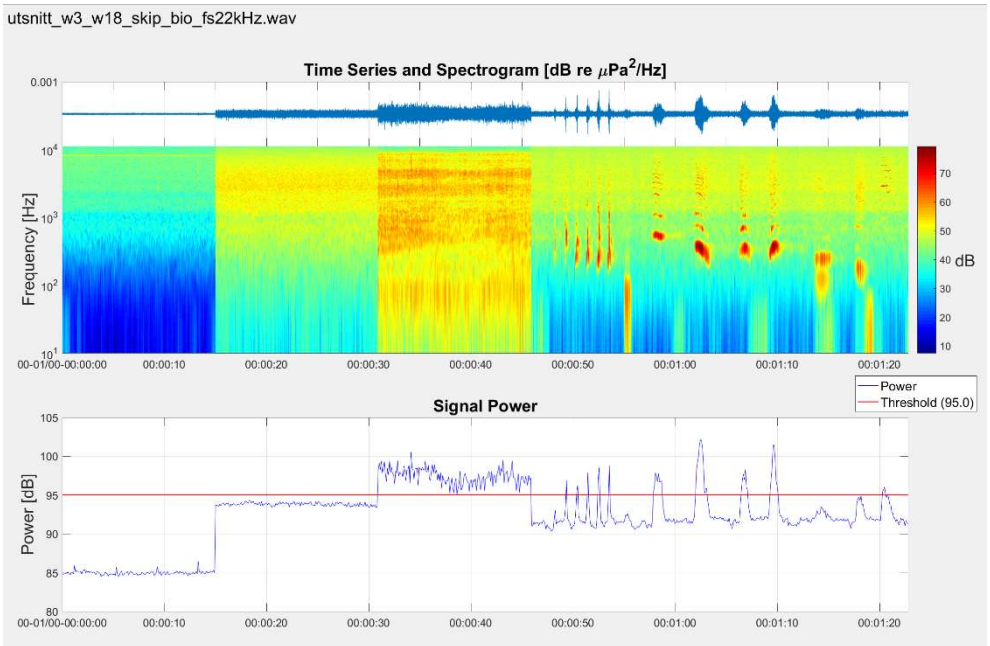

Figure 4.2: Energy detector tested on a differenced version of the sample file.

4.1.2 Successive Estimation

Since the simulation experiments showed that successive estimation produced a more well-behaved test statistic with a pure AR model than a full ARMA model, an AR(7) model will also be used to test the method on the real data set.

Figure 4.3 shows that the p-values produced by the method are extremely low for the ship noise and the whale sounds. Interestingly, the p-values for the other ambient noise signal at 3 m/s wind speed are also extremely low. This should perhaps not come as a surprise, since the detector is trained for very different weather conditions. From the point of view of the detector, another ambient noise signal may be just as "different" as a ship noise signal if it fits poorly with the trained model.

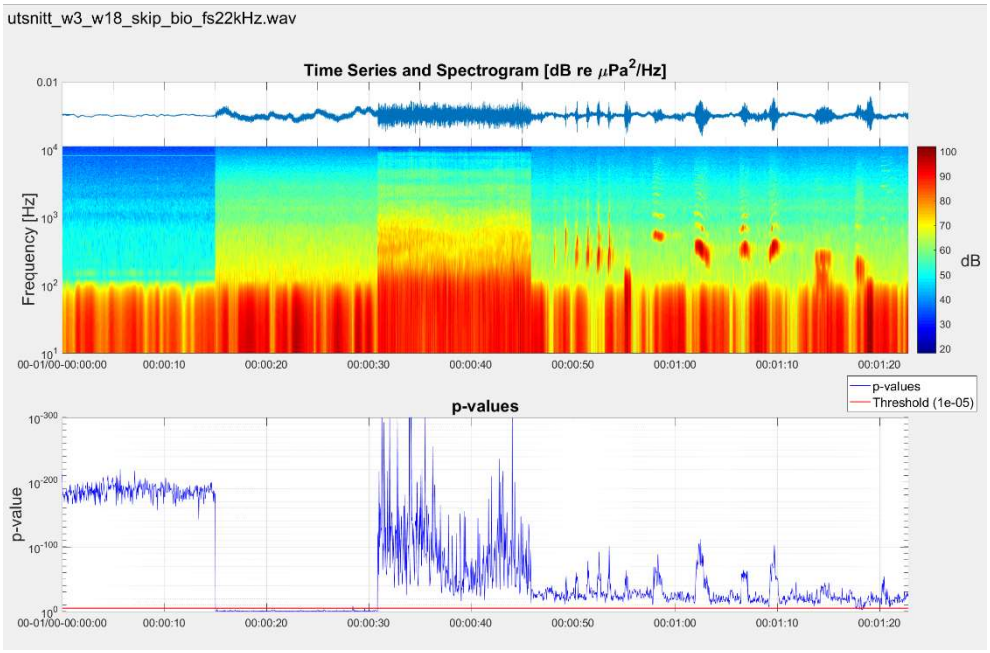

Figure 4.3: p-values for the successive estimation method using an AR(7) model.

When an AR(20) model is fitted using successive estimation and we run the algorithm in 3.5.5, it produces seven orthogonal directions in the feature space that have a positive SNR. The effect of projecting feature vectors into the space spanned by the six first of these is shown in Figure 4.5. While the ship noise has approximately the same p-values as before, the p-values for the other signals are not as extreme as in the unsupervised case. This shows that the detector has been made more sensitive to the ship noise relative to the other signals. The effect becomes even more evident if we project onto only the first component, *i.e.* Fisher's linear discriminant, as shown in Figure 4.6.

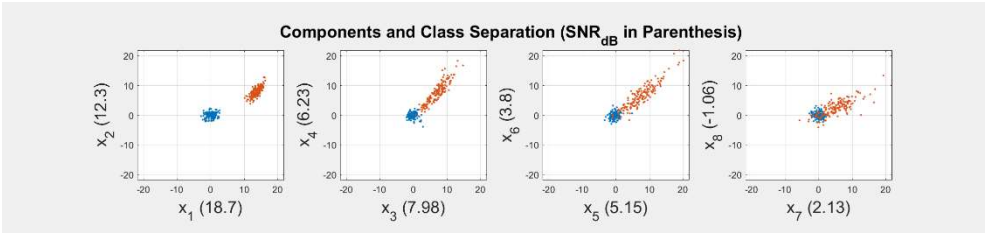

Figure 4.4: Orthogonal components obtained for the successive estimation method using an AR(20) model.

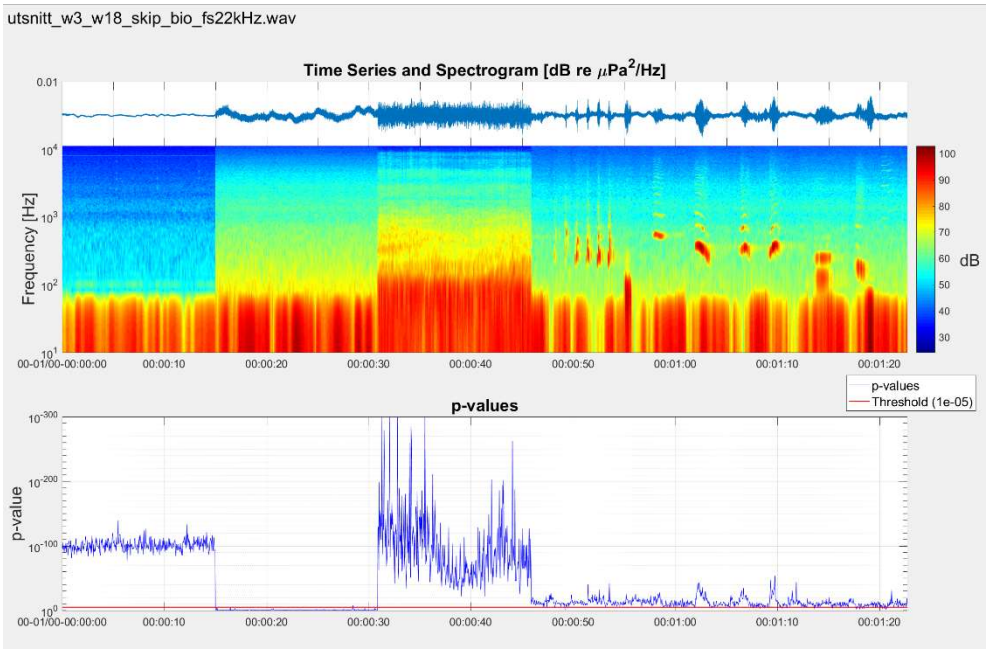

Figure 4.5: p-values for the successive estimation method after projecting onto the first six components.

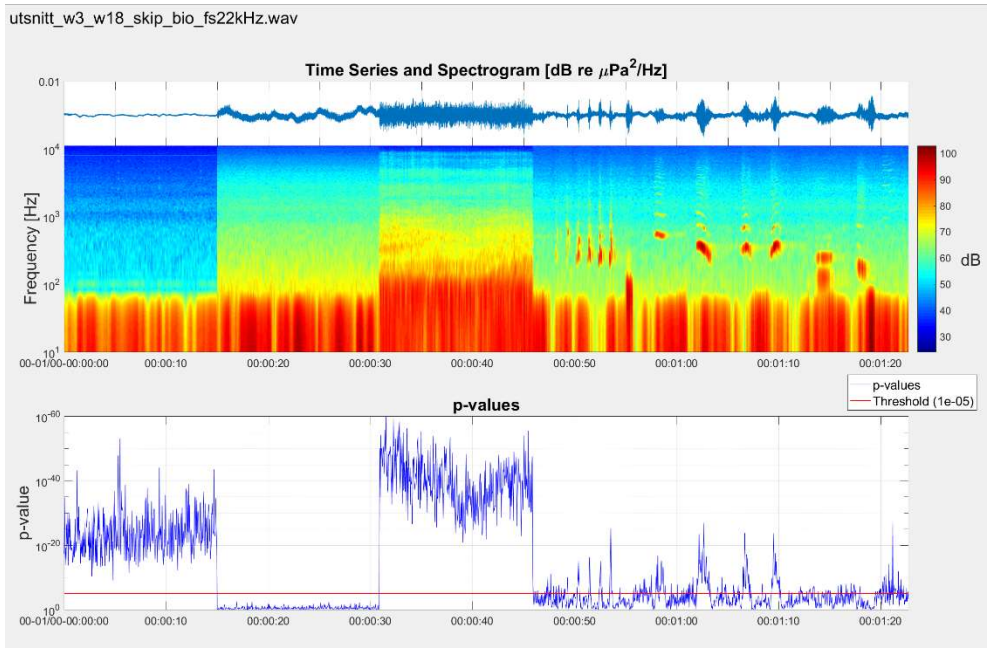

Figure 4.6: p-values for the successive estimation method after projecting onto only one component.

4.1.3 Prediction

As with successive estimation, the prediction method also produces extremely low p-values for both the other ambient noise signal and the whale sounds (Figure 4.7). The dimension of the feature space is 20 here, which is the number of autocorrelation lags being tested. By running the same algorithm as above, we can reduce the dimension from 20 to six by selecting the six first components shown in Figure 4.8. The result of projecting into this six-dimensional subspace is shown in Figure 4.9. Projecting onto only the first component has been done in Figure 4.10. As in the previous experiment, the detector becomes more sensitive to the ship noise relative to the 3 m/s wind noise and the whale sounds when supervision is used, and this is especially evident when using just Fisher's linear discriminant.

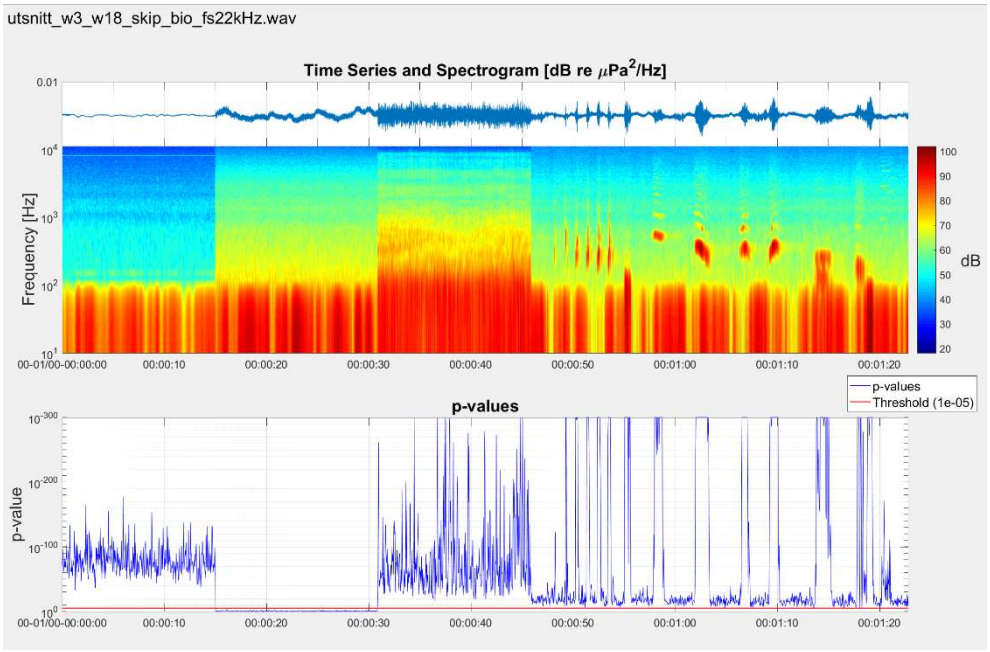

Figure 4.7: p-values for the prediction method using 20 autocorrelation lags.

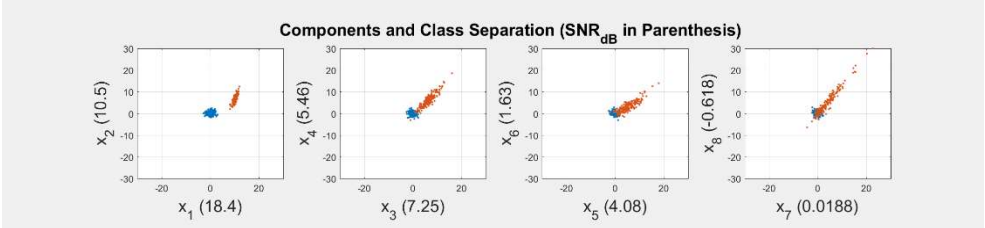

Figure 4.8: Orthogonal components obtained for the prediction method with 20 autocorrelation lags.

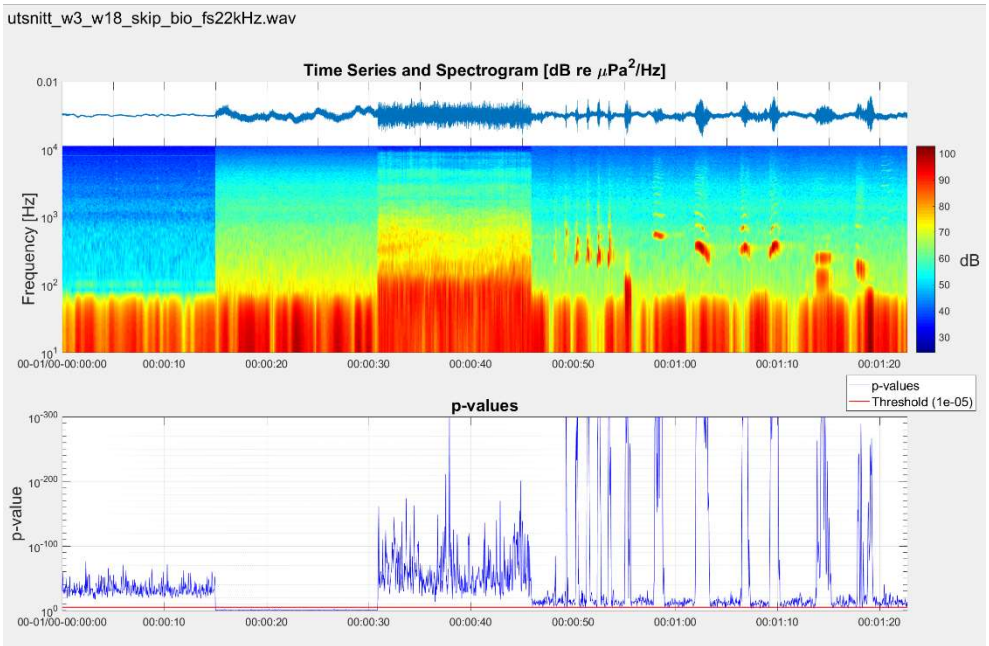

Figure 4.9: p-values for the prediction method after projecting onto the first six components.

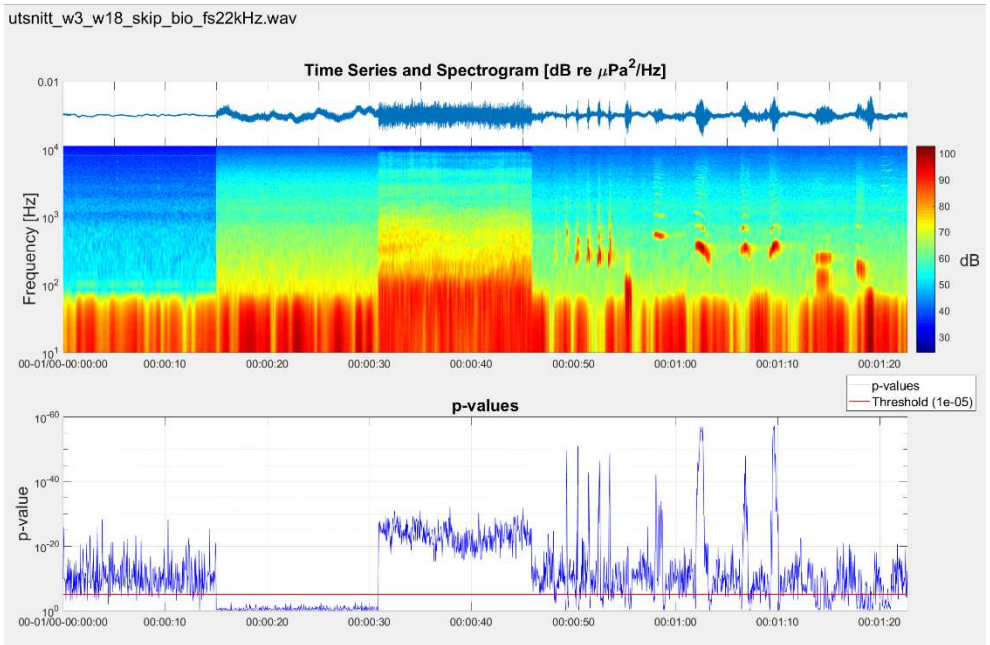

Figure 4.10: p-values for the prediction method after projecting onto only one component.

4.2 Added Noise

It is evident from the previous experiments that the signals in the test file are so different from each other that both the energy detector (with differencing) and the two ARMA methods have no problems distinguishing between them. In order to create a situation where the signals are more similar, a substantial amount of simulated ambient noise will now be added.

4.2.1 Energy Detector

Figure 4.11 shows how the energy detector performs when the noise gain is set to 12 dB and mean square values are calculated from the differenced signal. The detector still reacts to some of the anomalies, but it is hard to judge from the plot how well it could work in practice.

When experiments have been conducted with different amounts of added noise, the energy detector has been observed to gradually deteriorate from what is shown in Figure 4.2 to what is shown in the Figure 4.11. Since we don't have a distribution model for the mean square values, we cannot convert them to p-values and draw a significance level threshold like for the ARMA-based methods. Consequently, it is hard to say exactly when the detector becomes unusable, or to conclusively say that it is better or worse than some other method. Still, a noise gain of about 12 dB in this setting seems to be too much for the energy detector to reliably handle.

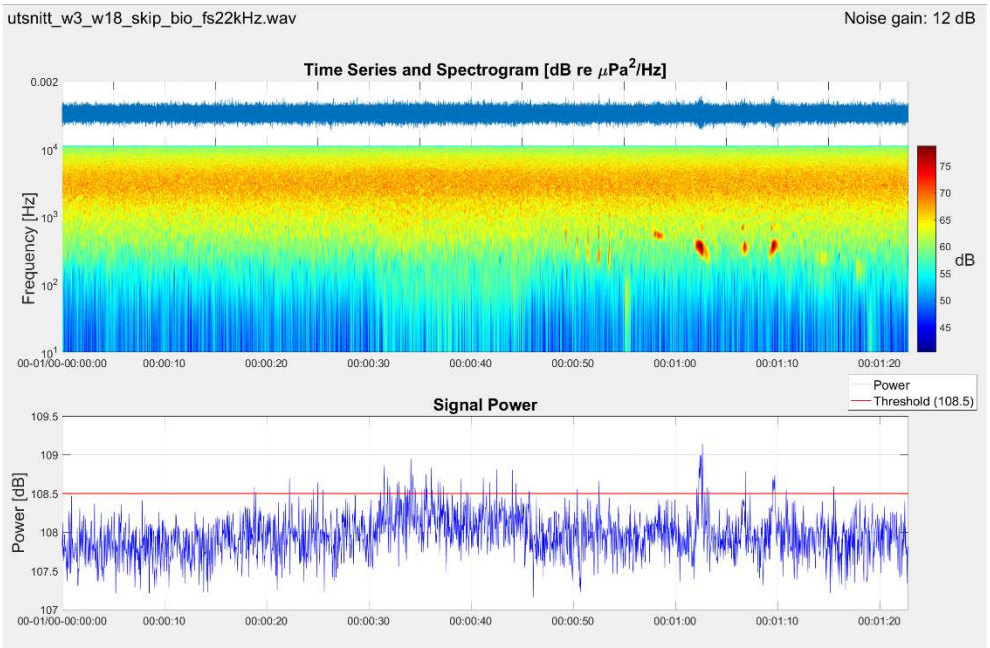

Figure 4.11: Energy detector with differencing tested at 12 dB noise gain.

4.2.2 Successive Estimation

When an AR(7) model is fitted to the training data with added noise using successive estimation, the performance of the detector on the test file is as shown in Figure 4.12. Here, most of the whale sounds are detected at the chosen significance level, but the ship noise is not. Nor is the ship noise visible in the spectrogram. The reader may need to consult one of the figures where no noise has been added to see where the various anomalies occur.

Now, if we fit an AR(20) model and try to reduce the dimension, it turns out that only the first component has a positive SNR. Projecting onto this component causes the ship to be detected, although only for the first part of the ship noise signal, not the unseen part (Figure 4.14). So, even though the whale sounds can still be detected, successive estimation struggles with the ship noise in this experiment.

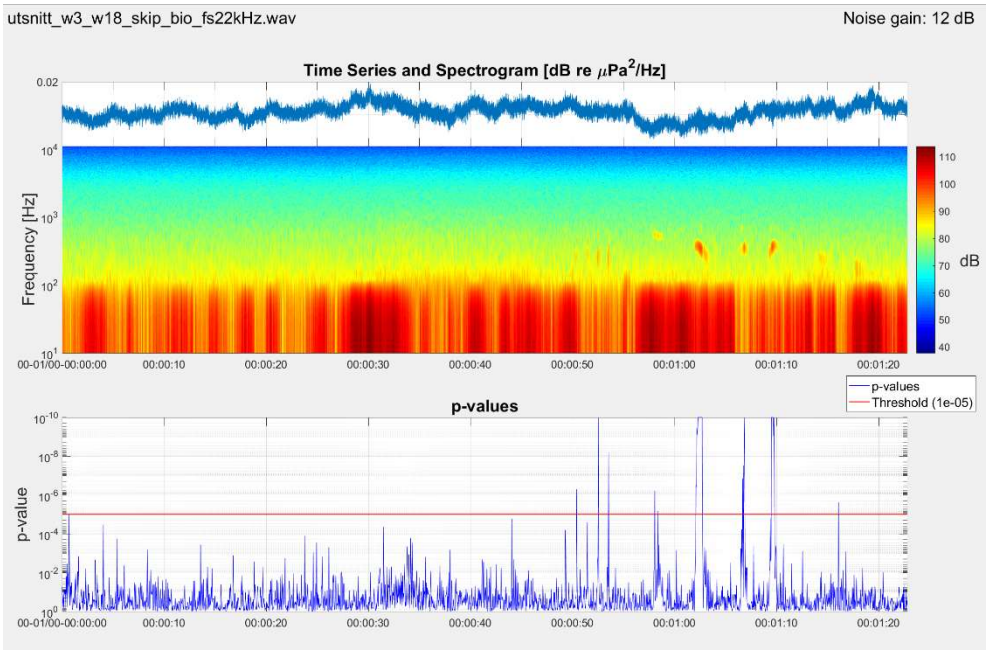

Figure 4.12: p-values for the successive estimation method using an AR(7) model.

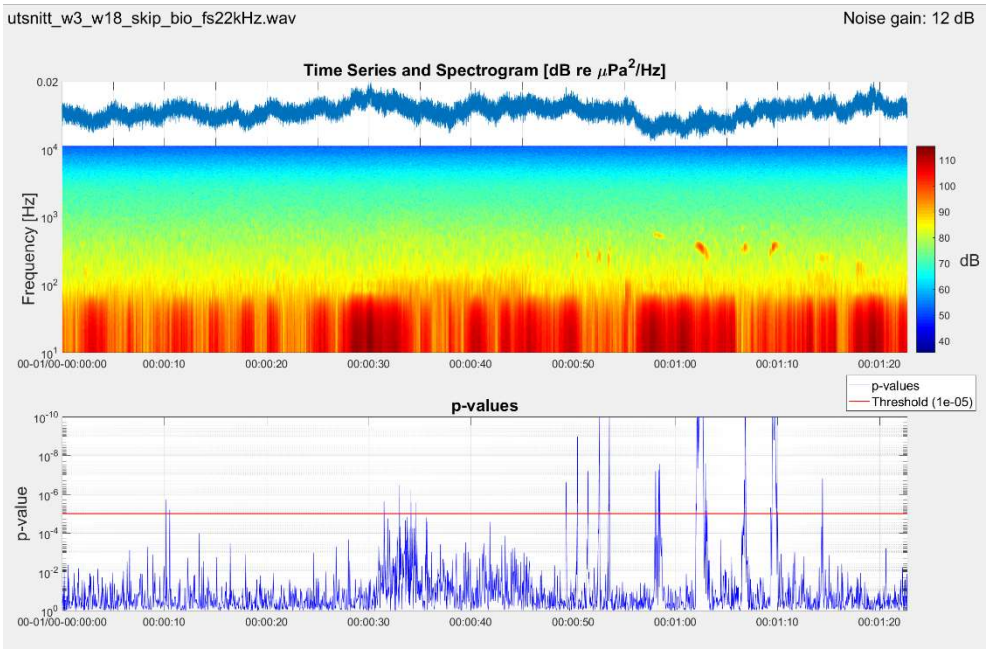

Figure 4.13: p-values for the successive estimation method after projecting onto the first component.

4.2.3 Prediction

The prediction method shows better performance in the unsupervised case (Figure 4.15), although some of the whale sounds at the end are not detected. As in the case of successive estimation, only one useful component is found from reducing the dimension. When the autocorrelation vectors are projected onto this component, the result is as shown in Figure 4.17. The detector is now more sensitive to the ship noise. Incidentally, one more whale sound is also detected after the projection.

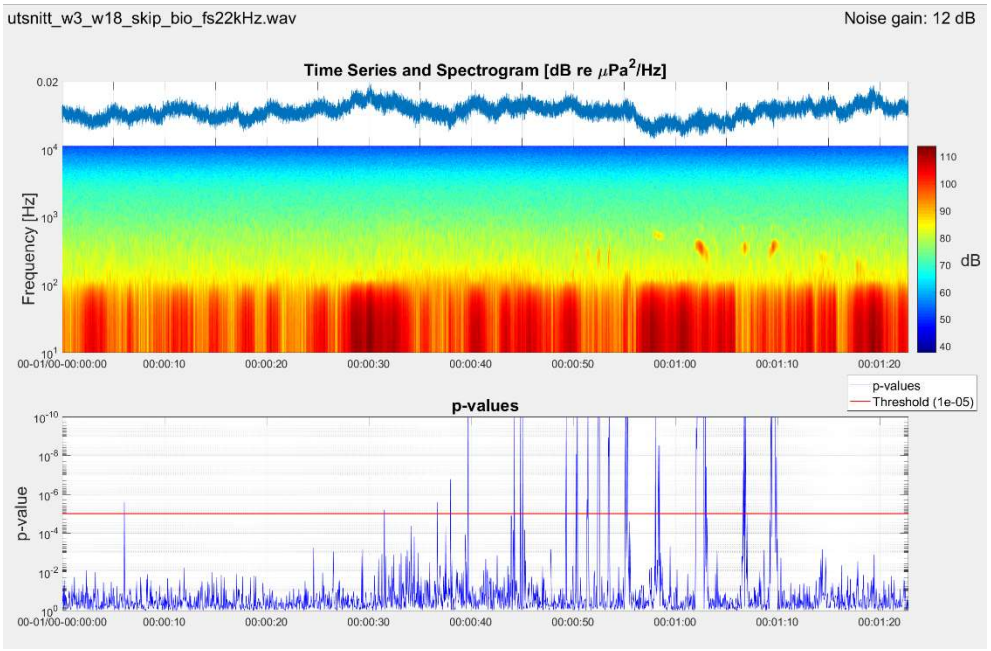

Figure 4.14: p-values for the prediction method using 20 autocorrelation lags.

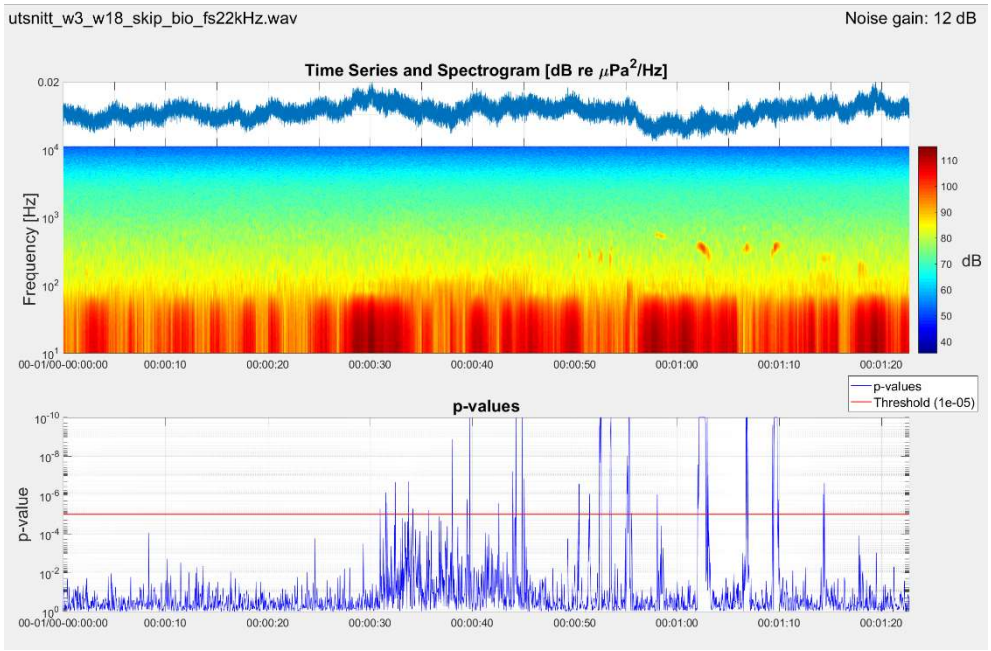

Figure 4.15: p-values for the prediction method after projecting onto the first component.

4.3 Adjusting Parameters

So far, we have kept the segment length constant at 0.04 seconds for all the methods, and the number of autocorrelation lags for the prediction method has been 20. As prediction seems to be the most promising of the methods, this section will briefly demonstrate the effect of adjusting the parameters of the prediction method.

In Figure 4.18 the ambient noise gain has been increased to 21 dB. The p-values obtained using the same settings as in the previous experiments appear to respond very little to the whale sounds, which are now almost completely buried in simulated noise.

When the segment length is increased to 0.4 seconds, however, Figure 4.19 shows that two of the whale sounds are detected. These two are also the ones that are the most visible in the spectrogram.

Since we now have more data for each estimated autocorrelation vector, we can increase the number lags as well. Figure 4.20 shows what happens when this value is set to 300. Apparently, one more whale sound is now detected.

So, in this example at least, increasing the segment length and the number of lags were both helpful in detecting faint sound sources buried in noise. The appropriate values of the two parameters will probably depend on what kind of anomalies we want to detect.

This experiment demonstrates a major advantage of the prediction method over the successive estimation method. The fact that the MVN parameters of the autocorrelation vector are known *a priori*, means that we can increase both the amount of data used to estimate the vector and the number of elements in the vector – without needing any more data to get the MVN parameters.

If we wanted to do successive estimation with 10 times longer segments, we would need 10 times as much raw data to obtain the same number of samples for MVN parameter estimation. And the successive estimation analog to using 300 lags is fitting AR(300) models to each segment and attempting to estimate the covariance matrix of the resulting parameter vector, which hardly seems realistic.

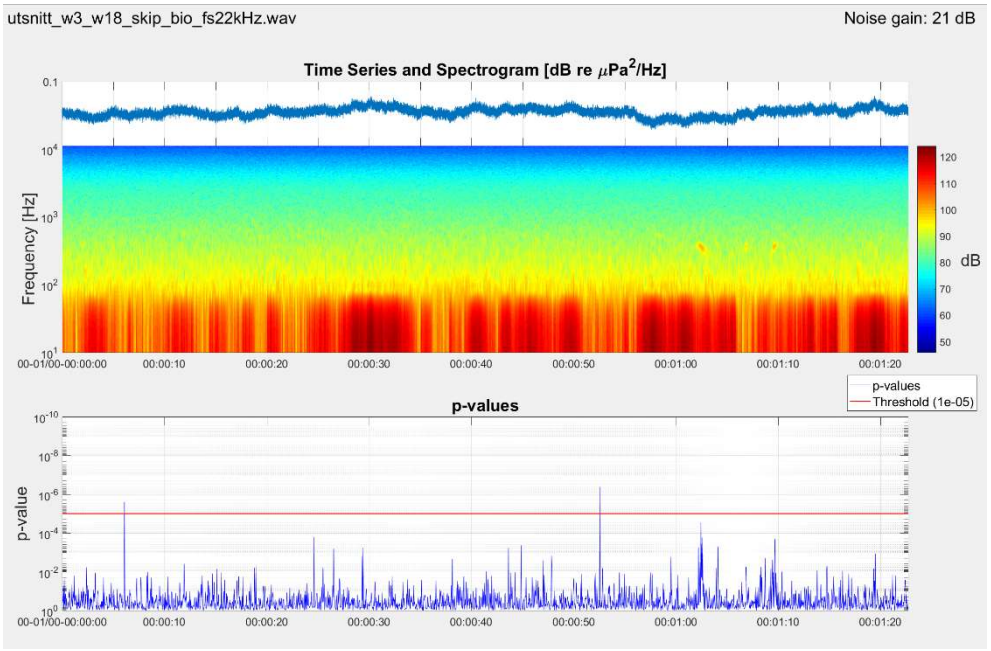

Figure 4.16: p-values for the prediction method using 20 autocorrelation lags and a segment length of 0.04 seconds.

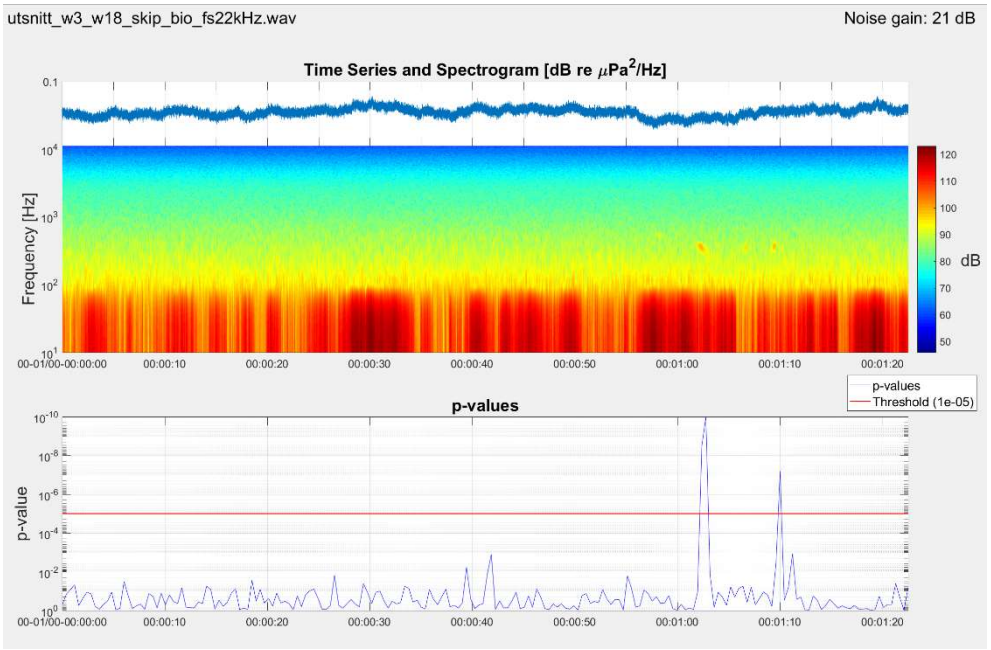

Figure 4.17: p-values for the prediction method using 20 autocorrelation lags and a segment length of 0.4 seconds.

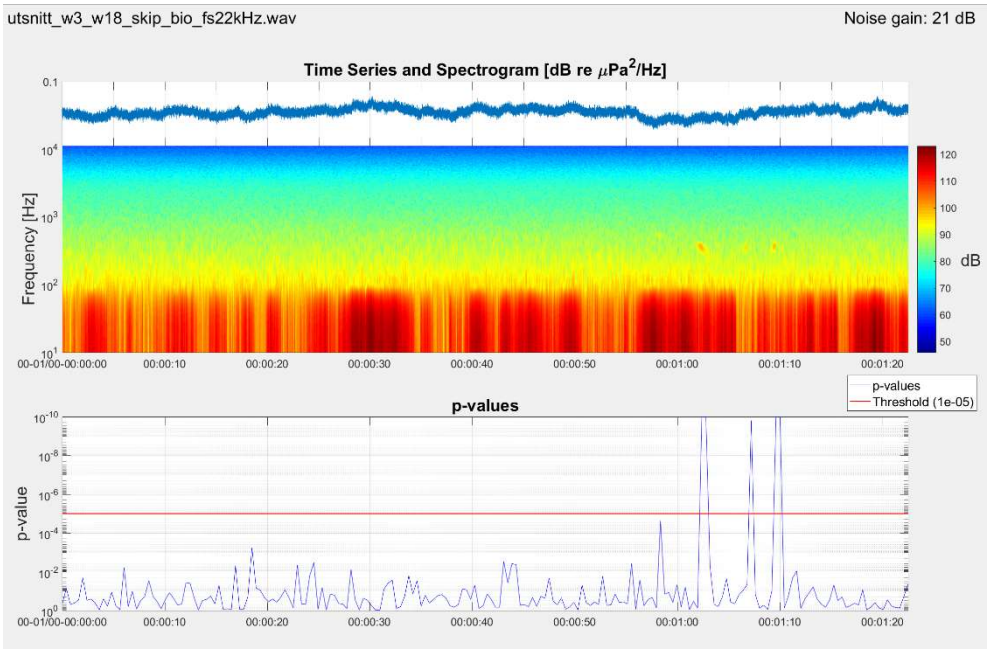

Figure 4.18: p-values for the prediction method using 300 autocorrelation lags and a segment length of 0.4 seconds.

5 Conclusion and Further Work

5.1 Conclusion

Tests on simulated and real data sets have indicated that the selected ARMA methods are better suited for detecting anomalies in underwater ambient noise than a simple energy detector. It was pointed out, however, that precise comparison with this detector cannot be made without having a distribution model for the mean square values the detector is based on. An attempt at building such a model was not made. The energy detector itself was found to produce significantly more stable values when differencing was applied to the signal before calculating mean square values. But even then, the ARMA methods showed better performance.

Of the two ARMA methods, the prediction method was found to be better than the successive estimation method in three ways. Firstly, its test statistic tended to fit better with the chi-squared distribution under the null hypothesis of ambient noise. This produced more reliable p-values and fewer false alarms. Secondly, it was able to detect fainter anomalies than the successive estimation method. Thirdly, our *a priori* knowledge of the MVN parameters of the sample autocorrelation vector meant that we could freely increase the length of the data segments without needing more data for training. It was demonstrated that both increasing the segment length and the degrees of freedom, *i.e.* the number of autocorrelation lags, were potential ways of improving detector performance for the prediction method.

Experiments with supervised dimensionality reduction indicated that this could be a useful way of building detectors that are more sensitive to certain kinds of anomalies without having to look for specific acoustic signatures. The proposed generalization of Fisher's linear discriminant allowed us to freely choose the subspace dimension and hence the degrees of freedom of the resulting test statistic, but it was unclear if this offered any practical benefit over just using FLD on its own.

5.2 Further Work

When the test statistic of the successive estimation method was observed to fit poorly with the chi-squared distribution, we had to resort to a fairly low-order AR model to get reliable p-values. This may have caused the detector to be less sensitive to changes in the process than it could otherwise have been using a full ARMA model. It would therefore be interesting to examine some other non-iterative ARMA estimation methods and see if any of them produced a test statistic with a better fit. For instance, the algorithm proposed by Hernandes et al. (2008) appears to be more accurate than the two-stage least squares method used here, which could perhaps affect the distribution of the test statistic in a positive way.

Even though the Ljung-Box test was found to work well for anomaly detection, it is by no means the only test that can be performed on the prediction errors of an ARMA model. As a continuation of the work done here, it could be useful to explore some other possible hypothesis tests and compare their performance. For instance, the Breusch-Godfrey test is another way of testing for serial correlation that could potentially be used instead of the Ljung-Box test (Breusch, 1978).

Since first order differencing was shown to improve the energy detector, it would be interesting to see if this could improve the ARMA-based methods as well. Some initial experiments have indicated that the performance is roughly the same, but I have not tested this systematically.

The tests done on the dimensionality reduction algorithm were not sufficient to determine if it offered any improvement over FLD. To properly test this, we would need to feed the algorithm examples of data from a general class of anomalies, such as ships or submarines, and then test detector performance on unseen data from the same class with one or more components. If it turns out that adding more components is beneficial, then the algorithm could perhaps be useful for detecting anomalies in other types of data as well, such as images.

A limitation of all the methods discussed here is that they need to be trained for certain conditions such as wind speeds, and will not work well if those conditions change too much. This problem can either be tackled by fitting several models for different weather conditions and allowing the detector to switch between models, or we could try to build a detector that continuously updates its model to the changing conditions. A challenge in this regard would be to distinguish between true anomalies and natural changes in ambient noise, so that only the right kind of changes are incorporated into the updated model.

List of Abbreviations

ACF	autocorrelation function
ACV	autocorrelation vector
AR	autoregressive
ARIMA	autoregressive integrated moving average
ARMA	autoregressive moving average
BIC	Bayesian information criterion
CDF	cumulative density function
DOF	degrees of freedom
DTFT	discrete-time Fourier transform
FFI	*Forsvarets forskningsinstitutt*, Norwegian Defence Research Establishment
FLD	Fisher's linear discriminant
GOF	goodness of fit
GUI	graphical user interface
LB	Ljung-Box
LDA	linear discriminant analysis
LoVe	Lofoten-Vesterålen Ocean Observatory
LS	least squares
MA	moving average
ML	maximum likelihood
MVN	multivariate normal
PACF	partial autocorrelation function
PCA	principal component analysis
PDF	probability density function
PEM	prediction error method
PSD	power spectral density
SNR	signal-to-noise ratio
SVD	singular value decomposition
WSS	wide sense stationary

Appendix: MATLAB Code

On the following pages, all the functions referred to in the thesis are listed. Most of the code for the thesis has been written so that it can be accessed from the user interface shown in Figure A.1. On the left side are settings related to the detector itself; on the right side are settings for graphical representations. When a detector has been trained and tested with the desired settings and input data, it can be saved as a MATLAB object to be retrieved later on or adapted to run on another system.

Much of the code is found inside the class definition file for the detector object, and the settings on the left side of the GUI are properties of this object. The class definition file itself will not be listed here, as it mostly deals with organizing the data and moving matrices around between different functions, as outlined in Sections 3.2 and 3.6.

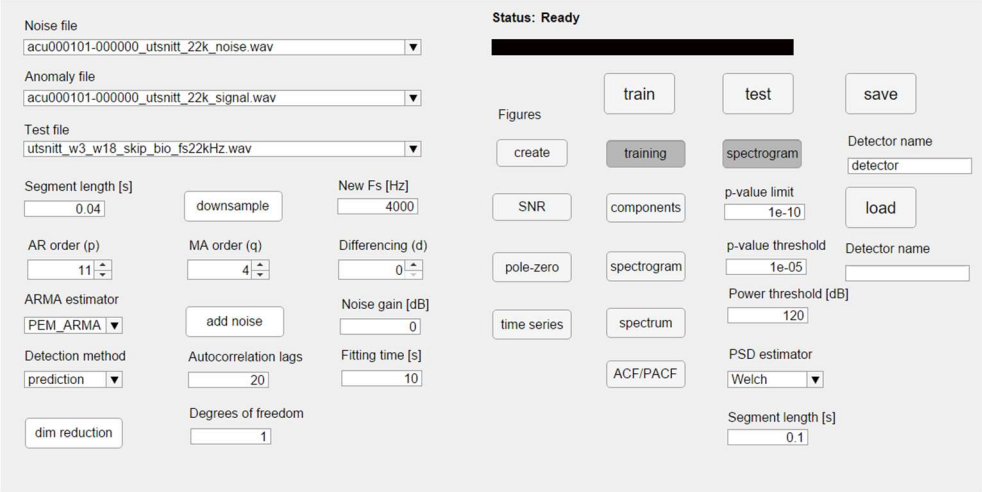

Figure A.1: Graphical user interface for the MATLAB program used to conduct the experiments in this thesis.

ARMA Estimation

```
function [ a, b, s ] = ML_ARMA( y, p, q )
% Computes the unconditional maximum likelihood estimate of the
% coefficients a and b, and the input noise standard deviation s, in the
% ARMA(p,q) model of the signal y. Requires the Econometrics Toolbox.

        mdl = arima( p, 0, q );
        estMdl = estimate( mdl, y, 'print', false );

        a = cell2mat(estMdl.AR)';
        b = cell2mat(estMdl.MA)';
        s = sqrt(estMdl.Variance);
        a = [1;-a];
        b = [1;b];

end
```

```
function [ a, b, s ] = LS_AR( y, p, method )
% Computes the least squares estimate of the coefficients a and the input
% noise standard deviation s in the AR(p) model of the signal y. The method
% used is either the autocorrelation method or the covariance method.

    y = y(:);
    N = length(y);

    switch method
        case 'acorr'
            Y = convmtx(y,p+1);
        case 'covar'
            Y = convmtx(y,p+1);
            Y = Y(p+1:N,:);
        otherwise
            disp('Method not recognized');
    end

        yp = Y(:,1);
        X = Y(:,2:p+1);
        theta = -(X\yp);
        a = [1;theta];
        b = 1;
        err = X*theta + yp;
        s = sqrt(mean(err.^2));

end
```

```
function [ a, b, s ] = LS_ARMA( y, p, q )
% Computes the two-stage least squares estimate of the coefficients a and
% b, and the input noise standard deviation s, of the ARMA(p,q) model of
% the signal y.

    y = y(:);
    N = length(y);

    if q == 0
        [a, b, s] = LS_AR( y, p, 'covar' );
    else
        K = floor(N/10);
        if K > 20
            K = 20;
        end

        aK = LS_AR( y, K, 'covar' );
        e_hat = filter(aK,1,y);
        m = max(p,q) + K;

        Y = convmtx(y,p+1);
        Y = Y(m+1:N,:);
        E = convmtx(-e_hat,q+1);
        E = E(m+1:N,:);

        z = Y(:,1);
        Y = Y(:,2:end);
        E = E(:,2:end);
        Z = [Y E];
        theta = -(Z\z);
        e = Z*theta + z;
        s_sq = (e'*e)/(N-m);
        s = sqrt(s_sq);
        a = [1;theta(1:p)];
        b = [1;theta(p+1:end)];

    end

end
```

```
function [ a, b, s ] = PEM_ARMA( y, p, q )
% Computes the prediction error method estimate of the coefficients a and
% b, and the input noise standard deviation s, in the ARMA(p,q) model of the
% signal y. Requires the System Identification Toolbox.

    data = iddata( y, [], 1 );
    sys = armax( data, [p q] );

    a = sys.A';
    s = sqrt(sys.NoiseVariance);
    b = sys.C';

end
```

Order Selection

```
function [ p, q, BICmin ] = BICorder( y, p_max, q_max )
% Adapted from Mathworks example:
% https://se.mathworks.com/help/econ/choose-arma-lags.html?s_tid=gn_loc_drop
% [Acessed 15 Jan. 2017].
% Computes the Bayesian information criterion for all combinations of ARMA
% model orders from 1 to p_max and 1 to q_max. Then selects the model
% orders for which the criterion is minimized. Requires the Econometric
% Toolbox and the Parallel Computing Toolbox. To remove parallelization,
% exchange "parfor" in line 15 with "for".

    N = length(y);
    LOGL = zeros(p_max,q_max);
    PQ = zeros(p_max,q_max);

    parfor p = 1 : p_max
        for q = 1 : q_max
            mdl = arima( p, 0, q );
            [ ~, ~, logL ] = estimate( mdl, y, 'print', false );
            LOGL(p,q) = logL;
            PQ(p,q) = p+q;
        end
    end

    LOGL = reshape(LOGL,p_max*q_max,1);
    PQ = reshape(PQ,p_max*q_max,1);

    [ ~, bic ] = aicbic( LOGL, PQ, N );

    BIC = reshape(bic,p_max,q_max);
    [BICmin,I] = min(BIC(:));
    [p,q] = ind2sub(size(BIC),I);

end
```

Dimensionality Reduction

```
function [ H, m, D, Vout, W, Ws, SNR ] = findBasis( PHI, PHIs, method, n )
% Finds a set of orthogonal basis vectors that span R^d, where d is the
% number of columns in PHI and PHIs. The basis vectors are orderered
% according to how well the points in PHI and PHIs are separated when
% projected onto the vector. The rows of PHI are assumed to be multivariate
% normal.
%
% Output arguments
% H: An othogonal matrix containing the basis vectors along its columns
% m: The mean vector of the points in PHI
% D: A diagonal matrix used for scaling variances
% Vout: An orthogonal matrix used for whitening along with D and m
% W: The whitened version of the input matrix PHI
% Ws: PHIs represented in the coordinate system of W
% SNR: The SNR of each basis vector

    [N,d] = size(PHI);
    Ns = size(PHIs,1);

    if strcmp( method, 'prediction' )

        V = eye(d);
        D = sqrt( rCovMat( n, d ) );
        m = zeros(1,d);
        M = zeros(N,d);
        Ms = zeros(Ns,d);

    else

        m = mean(PHI);
        M = repmat( m, [N 1] );
        Ms = repmat( m, [Ns 1] );

        [ ~, S, V ] = svd( PHI - M, 'econ' );
        D = S/sqrt(N-1);

    end

    W = (PHI-M)*V/D;
    Ws = (PHIs-Ms)*V/D;

    ms = mean(Ws)';
    I = eye(d);
    SIGMA = cov(Ws);

    hi = (I+SIGMA)\ms;
    hprev = hi/norm(hi);

    Hprev = [];
```

```
    Vout = V;
    SNR = zeros(d,1);

    df = hprev;
    SNR(1) = ((df'*ms)^2)/(df'*(I+SIGMA)*df);

    for i = 2 : d
        Hi = [Hprev hprev];
        PA = eye(d) - Hi*Hi';
        WA = W*PA;
        WAs = Ws*PA;
        [~,~,V] = svd(WA,'econ');
        Vr = V(:,1:d-i+1);
        PSIs = WAs*Vr;
        ms = mean(PSIs)';
        I = eye(d-i+1);
        SIGMA = cov(PSIs);
        df = (I+SIGMA)\ms;
        SNR(i) = ((df'*ms)^2)/(df'*(I+SIGMA)*df);
        hi = V*[df;zeros(i-1,1)];
        hprev = hi/norm(hi);
        Hprev = Hi;
    end

    H = [Hprev hprev];
    SNR = 10*log10(SNR);
    W = W*H;
    Ws = Ws*H;

end
```

```
function D = rCovMat( N, L )
% Forms the covariance matrix D of the normalized sample autocorrelation
% vectors, as computed by the MATLAB function autocorr.
% N is the number of samples used by autocorr. L is the number of lags.

    k = (1:L)';
    a = N - k;
    D = diag(a)/(N*(N+2));

end
```

Bibliography

Aggarwal, C. (2013). *Outlier Analysis*. 1st ed. [ebook] New York: Springer. Available at: http://link.springer.com/book/10.1007%2F978-1-4614-6396-2 [Accessed 11 Jan. 2017].

Breusch, T. (1978). Testing for Autocorrelation in Dynamic Linear Models. Australian Economic Papers, 17(31), pp. 334–355.

Bao, C., Hao, H. and Li, Z. (2012). Integrated ARMA model method for damage detection of subsea pipeline system. *Engineering Structures*, [online] Volume 48(2013), pp. 176–192. Available at: http://www.sciencedirect.com/science/article/pii/S0141029612005111 [Accessed 14 Jan. 2017].

Box, G., Jenkins, G. and Reinsel, G. (2008). *Time Series Analysis: Forecasting and Control*. 4th ed. Hoboken, New Jersey: Wiley.

Box, G. and Pierce, D. (1970). Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *Journal of the American Statistical Association*, 65(332), pp. 1509–1526.

Duda, R., Hart, P. and Stork, D. (2001). *Pattern Classification*. 2nd ed. New York: Wiley.

Durbin, J. (1959). Efficient Estimation of Parameters in Moving-Average Models. *Biometrika*, 46(3/4), pp. 306–316.

Faust, O., Acharya, R., Allen, A. and Lin, C. (2007). Analysis of EEG signals during epileptic and alcoholic states using AR modeling techniques. *IRBM*, [online] Volume 29(2008), pp. 44–52. Available at: http://www.sciencedirect.com/science/article/pii/S1297956207001209 [Accessed 14 Jan. 2017].

Fisher, R. (1938). The Statistical Utilization of Multiple Measurements. *Annals of Eugenics*, 8(4), pp. 376–386.

Gul, M. and Catbas, F. (2009). Statistical pattern recognition for Structural Health Monitoring using time series modeling: Theory and experimental verifications. *Mechanical Systems and Signal Processing*, 23(7), pp. 2192–2204.

Hayes, M. (1996). *Statistical Digital Signal Processing and Modeling*. New York: Wiley.

Hernandes, M., Stoica, P. and Rojas, M. (2008). ARMA Parameter Estimation: Revisiting a Cepstrum-Based Method. *IEEE International Conference on Acoustics, Speech and Signal Processing*, [online] pp. 3685–3688. Available at: http://ieeexplore.ieee.org/document/4518452/ [Accessed 11 Jan. 2017].

Kessy, A., Lewin, A. and Strimmer, K. (2016). *Optimal whitening and decorrelation*. [online] Cornell University. Available at: https://arxiv.org/abs/1512.00809 [Accessed 15 Jan. 2017].

Ljung, G. and Box, G. (1978). On a Measure of Lack of Fit in Time Series Models. *Biometrika*, 65(2), pp. 297–303.

Ljung, L. (1999). *System Identification: Theory for The User*. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall.

Louni, H. (2005). Outlier Detection in ARMA Models. *Journal of Time Series Analysis*, 29(6), pp. 1057–1065.

Mann, H. and Wald, A. (1943). On the Statistical Treatment of Linear Stochastic Difference Equations. *Econometrica*, 11(3/4), pp. 173–220.

Mathworks.com, (2016). *Choose ARMA Lags Using BIC*. [online] Available at: https://se.mathworks.com/help/econ/choose-arma-lags.html?s_tid=gn_loc_drop [Accessed 17 Jan. 2017].

Pincombe, B. (2005). *Anomaly Detection in Time Series of Graphs using ARMA Processes*. [online] ASOR. Available at: http://www.asor.org.au/publication/files/dec2005/Bra-paper.pdf [Accessed 14 Jan. 2017].

Scharf, L. (1991). *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*. Reading, Massachusetts: Addison-Wesley.

Shlen, J. (2003). *A Tutorial on Principal Component Analysis*. [online] Cornell University. Available at: https://arxiv.org/abs/1404.1100 [Accessed 15 Jan. 2017].

Söderström, T. and Stoica, P. (1989). *System Identification*. New York: Prentice Hall.

Subasi, A. (2005). Selection of optimal AR spectral estimation method for EEG signals using Cramer–Rao bound. *Computers in Biology and Medicine*, [online] Volume 37(2007), pp. 183–194. Available at: http://www.sciencedirect.com/science/article/pii/S0010482506000023 [Accessed 14 Jan. 2017].

Stoica, P. and Moses, R. (2005). *Spectral Analysis of Signals*. Upper Saddle River, New Jersey: Prentice Hall.

Wicklin, R. (2012). *What is Mahalanobis distance?* [online] Available at: http://blogs.sas.com/content/iml/2012/02/15/what-is-mahalanobis-distance.html [Accessed 15 Jan. 2017].

Yao, Q. and Brockwell, P. (2006). Gaussian Maximum Likelihood Estimation For ARMA Models I: Time Series. *Journal of Time Series Analysis*, 27(6), pp. 857–875.