Fourier Priors for Audio Restoration

Mair Allen-Williams

Master of Science

School of Informatics
University of Edinburgh

2005

Abstract

Traditional frequency-based audio restoration methods require complete data in order to perform a deterministic Fourier transform before handling the data in the frequency domain. We investigate the use of a probabilistic Fourier transform method, suggested by Storkey [27], in an audio restoration application, focussing on inferring missing and clipped points from music data. A practical implementation of the probabilistic Fourier transform method is necessary; we explore a number of optimisers and their associated performance and problems. We then investigate suitable models for Fourier priors for audio–in particular, music–data, and compare the performance of the probabilistic Fourier transform technique with Gaussian and t-distributed Fourier priors. We find that t-distribution priors are better suited to audio data than Gaussian, but in general it is difficult to perform effective audio restoration with incorporating perceptual information to the model.

Acknowledgements

Many thanks to my sister for keeping me sane and for listening to horrible crackly noises.

Thanks also go to Markus Svensen for his valuable assistance.

Finally, many thanks to my supervisor Amos Storkey for all his advice and encouragement.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Mair Allen-Williams*)

Table of Contents

Chapter 1

Introduction

Quantities of archived audio data exist in numerous contexts: examples include radio recordings, recordings of concerts and home-made tapes. Some of these may have been poor quality recordings, others may have degraded over time or been damaged at some point. Damaged data may be noisy or hissy, clipped, poorly compressed, or have gaps or clicks in the stream from damaged media.

There are a number of existing techniques for repairing audio data, and investigation into improving these techniques, and introducing new algorithms, is a lively research area. Techniques range from the simple interpolation methods attempted by most CD players through Bayesian models of varying complexity to neural networks and psychoacoustic analysis, using detailed knowledge about human perception.

Sound can be modelled either in the time domain, as sequence data, or in the frequency domain, as a combination of frequencies. Which domain is more appropriate will depend on the particular application and the way in which the data is to be modelled. State-of-the-art audio applications are likely to use combinations of time and frequency information to model the data.

Since time-domain applications are typically sequence based, they are not well suited to situations where there are likely to be large blocks of missing or inaccurate data. However, if it is known which data points are missing or faulty, then it may be possible to improve upon traditional frequency techniques, which do not take this information into account.

Storkey [27] has proposed a method for performing a probabilistic Fourier trans-

form on missing or noisy data. The technique uses prior probabilities over the components in the frequency domain. These can be updated to posterior probabilities based on the known data points: from this, a maximum likelihood Fourier transform can be computed if required. This project explores the practical use of this approach for the restoration of music files with missing or clipped data.

In §2, we give a brief overview of the relevant background to the project. In §3 we explain the probabilistic Fourier transform in detail, and in section §4 we discuss the practical issues associated with the implementation of the probabilistic Fourier transform. In §5 we explore in detail the nature of audio data in order to suggest suitable prior models for the frequency components. §6 develops these models. In §8 we present the results of the system on both synthetic and real damaged music data. In §9 we propose ways in which the system could be improved for music data and possible future applications in other domains. Finally, in §10 we summarise the conclusions we drew from the project.

Chapter 2

Background

2.1 Audio data

2.1.1 The nature of audio data

The human cochlea performs a frequency analysis of the signals which approach it. Human sound perception is not consistent over all frequencies; the ear is particularly sensitive to frequences around 3kHz, and frequencies outside the range 20Hz-20kHz can rarely be heard. *Fletcher-Munson* curves (figure 2.1[1]) describe audio frequency perception.

A perceived sound is rarely made up of a single frequency, and interactions between frequencies can further affect perception. The most common effect is "masking": strong frequencies may mask nearby weaker frequencies. However, the details of the masking depend on the particular frequencies and on the tonality of the masking frequency[2].

Finally, humans also respond differently to different types of noise; impulsive noise is more annoying to listeners than background hiss [12]. Successful audio restoration algorithms should take into account these perceptual features.

[1] source: `http://www.hibberts.co.uk/peals.htm`, with thanks to Bill Hibbert.
[2] source:`http://en.wikipedia.org/wiki/Psychoacoustics`

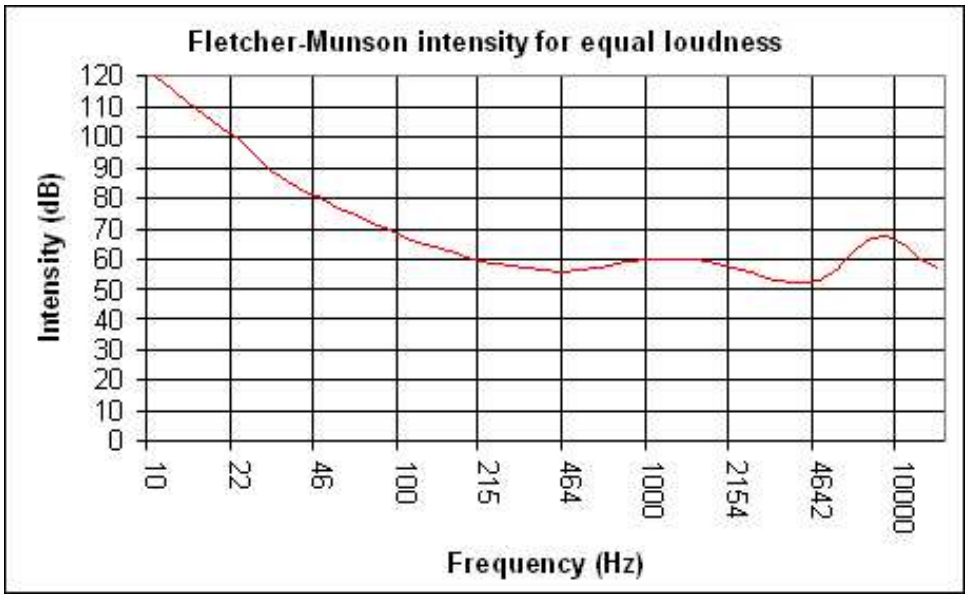


Figure 2.1: Human perception of loudness of audio frequencies. The curve depicts constant perceived loudness.

2.1.2 Damaged audio data

Forms of distortion which may occur on audio data include:

Missing data This may occur, for example, when an audio file is sent over a low-bandwidth connection. Lost packets may result in chunks of data missing from the received file.

Clicks Audible clicks may occur as a result of damaged media, such as a scratched record or CD. Other causes include slow sound drivers and nonconformant audio file creation software[3].

Clipping This occurs when the amplitude of the data has been constrained to be within some fixed limit L. All the data points which in the true signal are larger than L are represented as having the value L.

Quantisation This may take place in the amplitude domain, when a relatively small number of amplitude values are available to represent the data points, or in the

[3]source: `http://www.cdrfaq.org/faq03.html`

time domain, when the data has been sampled at a low rate.

Noise Stationary background noise, such as microphone hiss or drive motor hum, may be caused by the recording media. Non-stationary noise may occur on home recordings, caused by events in the background such as traffic or conversation.

Hiss Hiss is the result of random energy across the entire audio spectrum, generally referred to as "white noise".

There are many other forms of damage such as pitch variations ("wow"), crackling, in which the background contains random noises, and tonal distortions. Depending on the particular application, audio distortion effects may be subdivided in a number of ways. For example, many of the Bayesian sequencing methods discussed below handle linear distortions such as noise and hiss, and not non-linear distortions such as clipping and quantisation. Another possible categorisation is those types of distortion which can be treated as a missing data problem, perhaps alongside some other predictive technique: clips and clicks (once detected) may both be included in this category, albeit with potentially different models for the distribution and bounds for the inferred data points.

2.2 Overview of audio restoration techniques

Time-series methods State of the art time-series methods are typically based on Bayesian techniques. Godsill and Rayner [14] give an overview of Bayesian model-based approaches to removal of clicks (in the time domain), hiss (in the frequency domain) and other defects.

The Kalman filter [31] is based on a prediction of later states from current states and may be applied in either the time or the frequency domain. It is commonly used in the time domain to filter noise from a signal, but assumes Gaussianity and so may perform poorly if the correct model is non-Gaussian.

Monte Carlo methods form a similar approach. In particle filter methods the current datapoint is estimated based on the estimates for the preceding points. In particle smoothing methods the probability for a sequence is maximised. Fong and Godsill [10]

describe a way of exploiting substructure in Monte Carlo techniques to improve both efficiency and performance. Troughton [28] uses Bayesian models with Monte Carlo sampling to restore quantised data.

In [11], Fong and Godsill adapt the Monte-Carlo approach for non-linear distortion, testing it on clipped and quantised data. Although their experimental results on clipped and quantised data show some improvement in the audio signal, their discussion of this approach in [13] describes the computational requirements as 'prohibitive' and observe that they have had to make several simplifying assumptions.

Walmsley et. al [30] suggest a harmonic model for pitch estimation which could be used to differentiate between, for example, distinct instruments. Godsill and Davy [19] describe how to incorporate such a model into a Bayesian estimation system, using sinusoidal functions, but estimating in the time domain. Such a pitch estimation technique combines frequency and time information, and could potentially form a part of an audio restoration algorithm, although such an application is not described in the paper.

Spectral methods Simple frequency domain methods include the Wiener filter, which handles stationary additive noise and can be used for hiss reduction.

Time-domain interpolations are not well suited to filling in longer blocks of missing data. In [14], Godsill describes an interpolation which operates in the frequency domain and is suitable for filling in longer blocks.

The *multiresolution Fourier transform* [32] is a two-dimensional analysis which aims to examine a signal over a space of different frequency resolutions, in order to identify both short-term and long-term patterns in the data. Wilson et. al demonstrate how this can be used to pick out note and beat features from audio data. Scott and Wilson [25] discuss the use of this analysis to restore audio signals. Their approach relies on the existence of a prototype signal, possibly employing (for example) musicians to give a rendering of a recorded piece which is being restored. This may have some value for restoring data such as archived concert recordings, but is not widely applicable.

Abel and Smith [1] describe the restoration of a clipped signal by exact methods in a band-limited situation, making use of the DFT to provide bandwidth constraints on

the potential reconstructions. Their method is not probabilistic; if the possible signals which conform to the available constraints are not unique they propose the use of a norm to choose a reconstructed signal.

In [33], Wolfe and Godsill provide an overview of perceptually motivated approaches to background noise removal based on Fourier transforms of short blocks of data. These methods can include cost functions based on typical auditory perception, and can make use of knowledge about masking frequencies. Temporal information based on nearby blocks can be used to carry out the analysis on a particular block. These methods are particularly applicable to noise reduction, where there is a constant tension between removing enough of the noise to improve the signal, and not removing so much of the signal as to lose its integrity.

Neural networks Czyzewski [8] describes an approach to removing impulse noise based on learning a neural network. His approach uses two networks: one to detect the disturbances, and one to recover the original data. The networks are trained on both clean and distorted data. The disadvantage of his approach is that training neural networks is a time-consuming process, and distinct networks must be trained—requiring a good body of training material—for distinct classes of either audio signal or impulses. Czyzewski develops the use of neural networks further in [9], using the self-organising map of Kohonen and a neuro-rough controller (one which uses *rough* sets [16]—that is, creating approximations of sets) to remove non-stationary noise. The computational resources required for training remain an issue. Cocchi and Uncini [7] describe a neural network approach, operating in the frequency domain, to interpolating large blocks of missing data. They emphasize the advantages of non-linearity that neural networks provide. Their use of subband rather than full-band analysis ameliorates the performance difficulties somewhat.

Summary Clearly, audio restoration is a lively research area, and new approaches to audio restoration in a number of directions are being actively explored. State of the art restoration systems will typically combine a number of approaches, in both frequency and time domains. Audio data is treated by the human ear as frequency data, and so frequency approaches are well suited to capturing perceptual features of audio data and

for expressing short term periodicities in the data. Nevertheless, there is a wide range of effective sequence processing techniques for audio data which operate in the time domain.

2.3 Spectral approaches to audio data

Away from audio restoration research, there is work on using spectral techniques to improve audio analysis. This work could be incorporated into audio restoration techniques. For example, Cai et al [6] use various feature structure patterns to distinguish different types of audio signal. They also mention the use of principal component analysis in the frequency domain for de-noising.

Roweis [24] describes an approach to the source separation problem using a factorial hidden Markov model on spectrograms (a spectrogram is a representation constructed from the Fourier transform). Bach and Jordan [2] also use a spectrogram-based technique to solve the same problem. However, their methods are computationally heavy and resource intensive, involving large matrix calculations.

In [29], Tzanetakis and Cook explore feature-based classification of musical genre. They include pitch and beat features as well as frequency-based features, but find that frequency-based features provide the most accurate classifications. Pampalk [21] describes a number of spectral features which represent musical "similarity", implemented in a `MATLAB` toolbox. Although even on a very general feature set some genre distinctions are clear, application-specific information for a particular use could enhance the system.

One limitation of sequence processing techniques is the difficulty of applying them to relatively long consecutive chunks of data, as errors tend to compound. However, traditional spectral techniques require a translation into the frequency domain; something which can at best be estimated if there is missing data. Furthermore, the tendency of errors to compound means that first estimating the frequency transform and then estimating the repair may cause unwanted inaccuracies.

2.4 Probabilistic Fourier transforms

There has been some work on using a Bayesian derivation of the Fourier transform to estimate the transform of incomplete data. Gregory [15] describes a method due to Jaynes [17], using both systems with strong prior information and systems where the only prior information is in the choice of model.

Another interesting approach is that of Storkey [27], which exploits the graphical nature of the fast Fourier transform (FFT). This technique has the potential to be efficient as the FFT itself is an efficient technique. Storkey describes the application of the technique to missing audio data; it should also be applicable to noisy data of other kinds.

This approach could be incorporated into an audio restoration system. In its simplest form it would provide a way of estimating missing data, making full use of both frequency models (which we have suggested is appropriate for audio data) and the known time-series data. The same transform could be used for clipped data, using an appropriate model. The technique could be further extended to handle noisy data, or as a part of a larger system. In the next section we describe the probabilistic Fourier transform in detail.

Chapter 3

Probabilistic Fourier Transform

3.1 The Fourier Transform

Two well known periodic functions are the sine and cosine waves. It is a fact that any odd periodic function which fulfils the *Dirichlet conditions* (figure 3.2) can be expressed as a sum of cosine waves (an odd function is one for which f(x) = -f(-x); an even function is one where f(x) = f(-x)). Similarly, any even function fulfulling the same conditions can be expressed as a sum of sine waves (3.1).

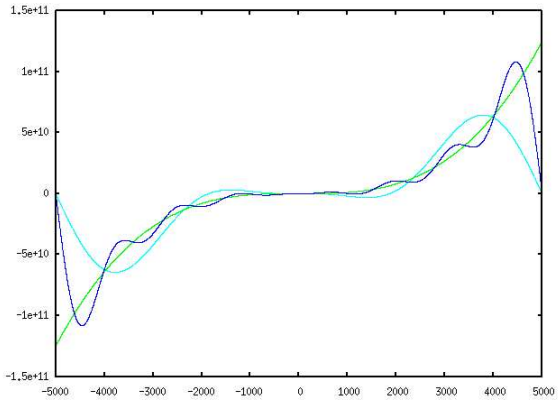
Furthermore, any function can be expressed as a sum of an odd and an even function:

$$\begin{aligned} f(x) = \quad & \tfrac{1}{2}(f(x)+f(-x)) + \tfrac{1}{2}(f(x)+f(-x)) \\ & = f_{even}(x) + f_{odd}(x) \end{aligned}$$

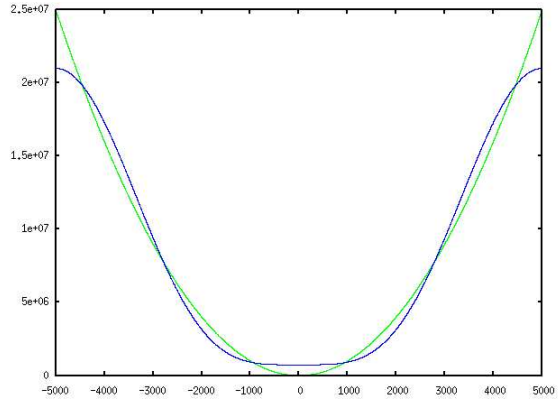
It is therefore possible to write any periodic function as the sum of a sine series and a cosine series.

$$f(x) = \sum_{n=0}^{\infty} a_n \sin(n\pi x/T) + \sum_{n=0}^{\infty} b_n \cos(n\pi x/T)$$

where T is the period of the function.



(a) $f(x) = x^3$
An odd function. The green line shows the function, the blue the sum of the first five sine components.



(b) $f(x) = x^2$
An even function. The green line shows the function, the cyan the sum of the first five cosine components, and the blue the sum of the first fifteen.

Figure 3.1: Functions as sums of series

1. $f(x)$ should be single-valued
2. $f(x)$ should have finitely many discontinuities
3. The discontinuities should be finite
4. There should be a finite number of maxima and minima per period
5. The integral over a single period of $|f(x)|$ should converge

Figure 3.2: Dirichlet conditions for convergence of the Fourier series of a function $f(x)$

Since $e^{ix} = \cos(x) + i\sin(x)$, we can reformulate the above as $< c_k e^{ixf_k} >$. Such a sequence is called a *Fourier series*. The frequencies f_k are obtained by $f_k = 2\pi k/T$.

The coefficients, c_k, are computed as

$$c_k = \frac{1}{T}\int_a^{a+T} f(x)e^{xf_k}dx$$

Any function which is only defined with a fixed range between two finite points a and b can be treated as a periodic function, where the range $(b-a)$ determines the period.

In practice, if a 'function' is determined by a finite set of data points, as in the case where the function is some audio signal, then we can treat it as a periodic function as described above. As we only have a finite set of data points, we replace the integral with a summation.

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} f(n)e^{2\pi ikn/N}$$

where the $f(n)$ are the datapoints. Computing the set of $< c_k >$ from the $f(n)$ is known as a *discrete Fourier transform* (DFT).

The transform can be inverted to obtain the original datapoints $f(n)$ from the Fourier components c_k:

$$f(n) = \sum_{k=0}^{N-1} c_k e^{2\pi ikn/N}$$

Figure 3.3 shows the DFT in action. The original curve, shown in green, is a short section of data from a music file[1]. The three curves show the original data offset with the curves formed from the sum of the first five, twenty-five and hundred and twenty-five Fourier components respectively. (There are a total of 256 data points in the section, so 256 Fourier components reconstruct the data exactly).

3.2 Fast Fourier Transform

The obvious method of computing a Fourier Transform is an $O(N^2)$ operation:

$$F = Af$$

where F is the N-dimensional vector of Fourier components:

$$F_k = \frac{1}{N}\sum_{n=0}^{N-1} f(n)e^{2\pi ikn/N}$$

f is the N-dimensional vector of data points, and A is the NxN matrix of coefficients:

$$A_i k = e^{-2\pi ikn/N}$$

However, it is possible to carry out the Fourier Transform more efficiently than this, using a recursive formula. We can construct the formula as follows (using F_k for the Fourier components):

$$F_k = \sum_{n=0}^{N-1} f(n)e^{2\pi ikn/N}$$

$$= \sum_{n=0}^{N/2-1} f(2n)e^{2\pi ik(2n)/N} + \sum_{n=0}^{N/2-1} f(2n+1)e^{2\pi ik(2n+1)/N}$$

$$= \sum_{n=0}^{N/2-1} f(2n)e^{2\pi ikn/(N/2)} + e^{2\pi ik/N}\sum_{n=0}^{N/2-1} f(2n+1)e^{2\pi ik(2n+1)/(N/2)}$$

[1] A Winter's Tale, Davis Essex

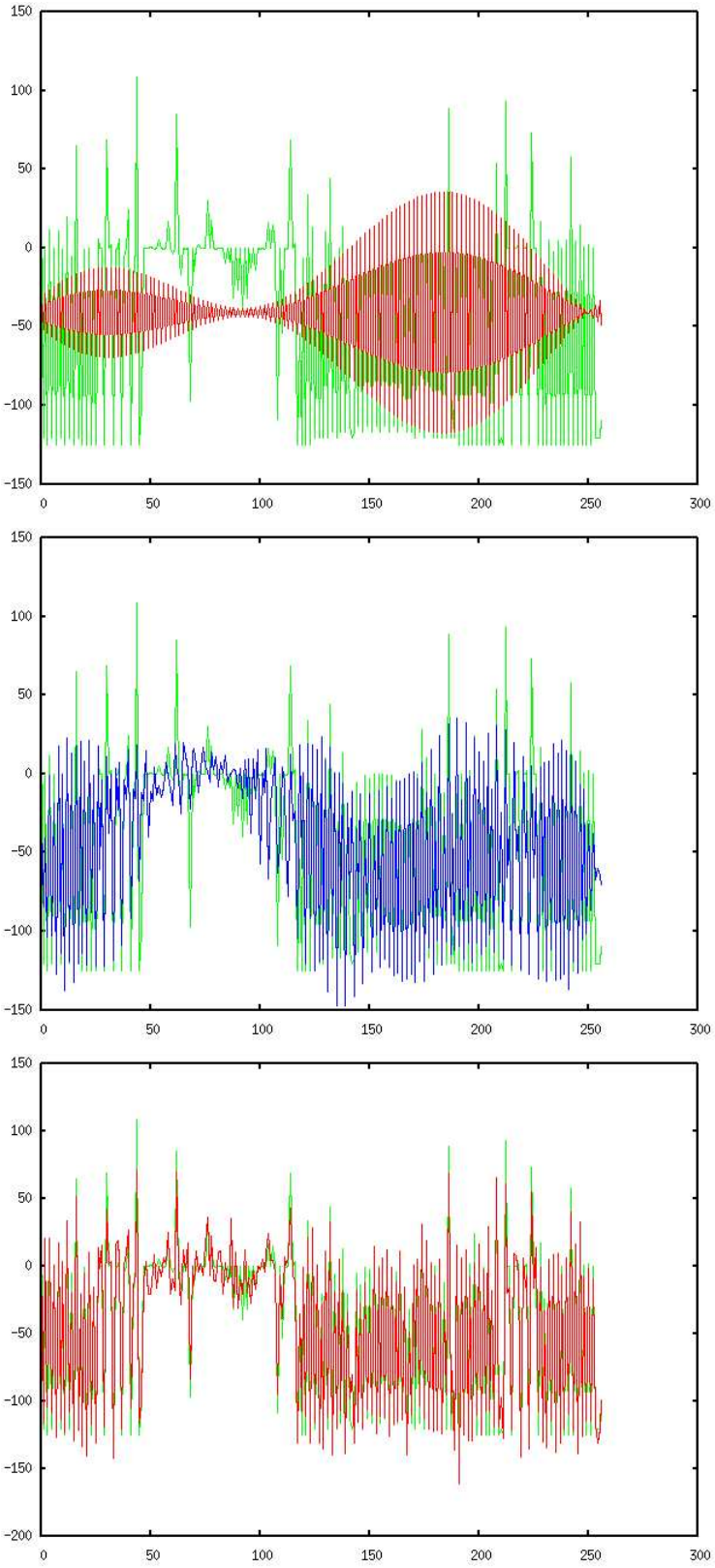


Figure 3.3: DFT example

$$= F_k^e + W^k F_k^o \tag{3.1}$$

where $W^k = e^{2\pi ik/N}$. F^e is the transform of length $N/2$ formed from the even components of the original f(n) and F^o the corresponding transform formed from the original components. (Proof due to Danielson and Lanczon, 1942. Reproduced in [23]).

As long as the data length, N, is a power of two, we can repeat this decomposition recursively until we need only perform the transform on a single point: that is, the identity operation. Next, it is necessary to determine which point corresponds to each single-point transform. A single point transform will be described in the notation above as

$$F^{eeo\ldots oeo}$$

for some pattern of es and os, where the es represent even tranforms and the os represent odd transforms. By reversing the order of es and os and letting $e = 0$ and $o = 1$, we have the binary value of n, where f(n) is the point used for this single-point transform. This is essentially because the last bit of the binary representation of the data determines its parity, thus which initial transform it is in.

These ideas make it possible to compute a Fourier transform in $O(\log_2 N)$ operations. First the data is sorted into reverse bit order to obtain the single point transforms (figure 3.4 shows an example). We then recursively combine the transforms according to equation 3.1. This process is called the *fast Fourier transform* (FFT). We can represent this graphically, as shown in figure 3.5 (taken from [27]). The arrows indicate the dependency of the data points on the Fourier components; they could equally be reversed to demonstrate the dependency of the Fourier components on the data.

If the length of the available data is not a power of two, then we must do some fudging to obtain the Fourier transform. One alternative is to pad the data with zeros to obtain a power of two. Another is to break the data up into more than one section, where the length of each section is a power of two, and then to perform the FFT on each section separately. If the sections are sufficiently large then this can be made to approximate the correct transform.

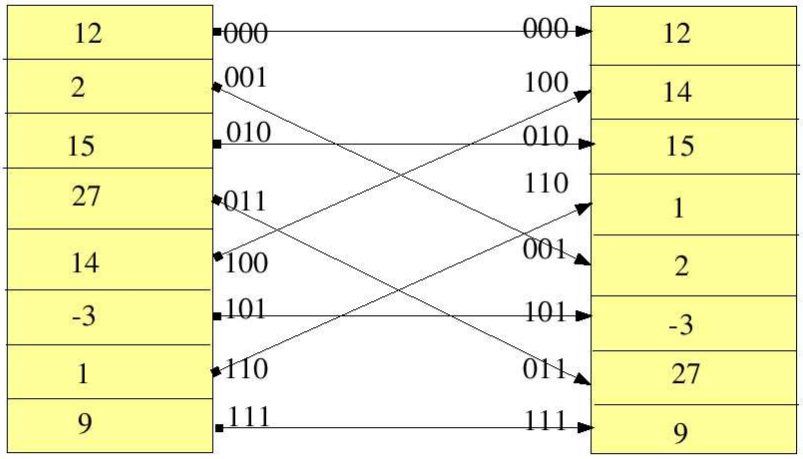


Figure 3.4: Sorting data into reverse bit order

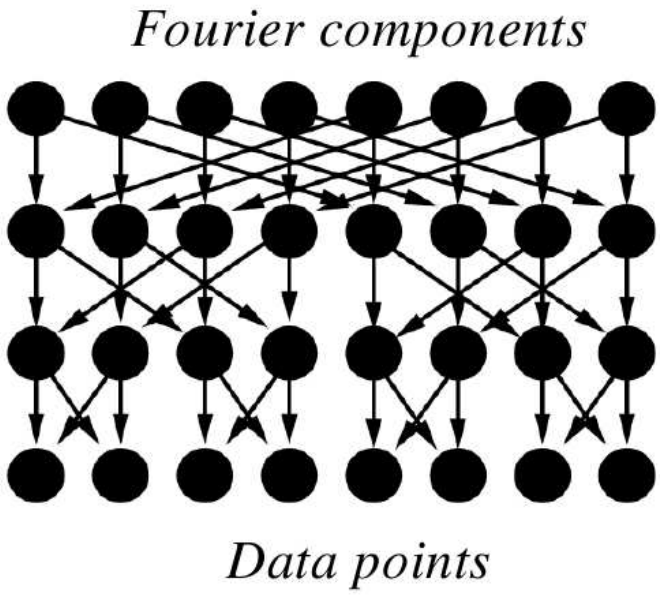


Figure 3.5: Network for fast Fourier transform

3.3 Probabilistic Fourier Transform

In order to compute a Fourier transform as described above, it is necessary to have all the data points. Typically, if some data is missing, it is filled in with zeroes before the Fourier transform is applied. However, it would be more correct to compute a posterior distribution over possible transforms, given the available data and some prior distribution on the frequency components. Given such a posterior distribution we could then take the maximum likelihood transform to use in a frequency-based application; invert the maximum likelihood transform to estimate the original data points, or use the full posterior distribution in an application.

3.3.1 Belief networks

A *belief network* is a representation of statements about the probability of events which allows efficient propagation of observations to update probabilities elsewhere in the network. The graphical structure for the FFT shown in figure 3.5 can be viewed as a belief network, where prior probabilities are placed on the Fourier components, and observations at the data points are propagated up through the network to obtain posterior probabilities. It would then be possible to propagate this information back through the network to obtain posterior probabilities for the un-observed nodes, given some prior distribution on the nodes (which could be uniform if there is no prior information about data points).

Note that this method applies equally well if it is the frequency data which is known, and the sequence data on for which we have prior information; we can merely replace the FFT with an inverse transform—that is, propagate information in the opposite direction.

There are a number of methods for propagating information through belief networks. The network, as shown in figure 3.5 is not singly connected; it contains (undirected) loops. This means that exact propagation methods such as Pearl's polytree algorithm [22] are not valid. Nonetheless, it is possible to make use of this algorithm as an approximation scheme. Some practical examples are dicussed in [20], but there is no guarantee of convergence. In practice Storkey [27] found that there were con-

vergence problems with the use of loopy propagation for FFT network of any size. A belief propagation scheme is used in which clusters of nodes are formed to try and avoid problems caused by the strong dependencies between the parents of a given node. Initial experimentation with this scheme produced good results. However, its scaling properties are not clear. Barber and Sollich [3] applied directed belief propagation using an "auxiliary variable trick" to the case in which the priors are Gaussian. They found that both the stability and the accuracy were improved over the undirected propagation used by Storkey. Unfortunately, this approach is not suitable for non-Gaussian systems.

3.3.2 Exact optimisation methods

The posterior probabilities for the Fourier components F can be computed by finding an exact solution to the equation

$$P(F|data) = \frac{P(data|F) * P(F)}{P(data)}$$

In the case where the data and hence $P(data)$ are fixed,

$$P(F|data) \propto P(data|F) * P(F)$$

Finding the maximum likelihood posterior for this system is an optimisation problem. Exact optimisation methods iterate towards finding a solution vector which maximises some objective function (a function of the solution vector). Many exact optimisation techniques use some form of hill climbing to move towards a solution, using the gradient of the objective function to determine a direction in which to move the test vector. The general form of this kind of search is an update to the estimated solution $\mathbf{w}$,

$$\mathbf{w} \leftarrow \mathbf{w} + \lambda \mathbf{d}$$

where $\mathbf{d}$ is the direction of search at that step and λ is chosen to minimise $E(\mathbf{w} + \lambda \mathbf{d})$ for the error function E. If we aim to minimise the objective function f, then $E = f$. The updates continue until the change in $\mathbf{w}$ falls below some small tolerance value.

Conjugate gradients are one such technique, using conjugate directions to determine the direction of travel, i.e. $\mathbf{d}$ is chosen such that

$$\mathbf{d}^T H\mathbf{d} = 0$$

where H is the Hessian (matrix of second derivatives) for the target function. Conjugate gradients are well suited to the solution of quadratic systems where they are guaranteed to converge in d steps, where d is the number of unknowns. For other distributions, there are no such guarantees, but the convergence properties are often found to be good in practical situations. A further advantage of the conjugate gradient method lies in its popularity: there are many implementations available.

3.4 Further steps

We have discussed finding a maximum likelihood Fourier transform found which is consistent with the original data. However, in a real application we may assume some measurement error on the observed data. If there is some transform T of data which is close to the observed data, and T has a much higher posterior probability than U, the precise transform of the observed data, then T may be a better posterior transform than U. In order to implement such a system, we would need to specify an acceptable measurement error on the original data. This should vary over the data, and may well depend on the data itself.

The theory described above applies equally well to any sequence data which can be profitably considered in its frequency domain. All that is needed is the specification of suitable priors on the Fourier components. In the next section, we discuss the implementation of this probabilistic FFT in the context of an audio restoration algorithm.

Chapter 4

Implementation

4.1 System design

- The core of the system is the module to perform the noisy FFT. This takes as inputs a block of data including unknowns and a model for the Fourier components, and returns the block of data with suggested values for the unknowns. Ideally, the posterior probability distribution over the data points should also be returned.
- The layer above this core handles a complete signal, converting stereo data into complex format, and passing each block to the probabilistic FFT core.
- Above this layer a third layer handles the loading of audio files and the saving of the repaired file. At this layer a set of model choices is provided and the user need merely select one.

Figure 4.1 illustrates this system.

4.2 Design decisions

Noisy FFT The core of the system is the probabilistic FFT code, which computes the maximum likelihood estimate for the unknown datapoints, given (1) the prior model for the Fourier components and (2) the observed data. This optimisation is time-critical,

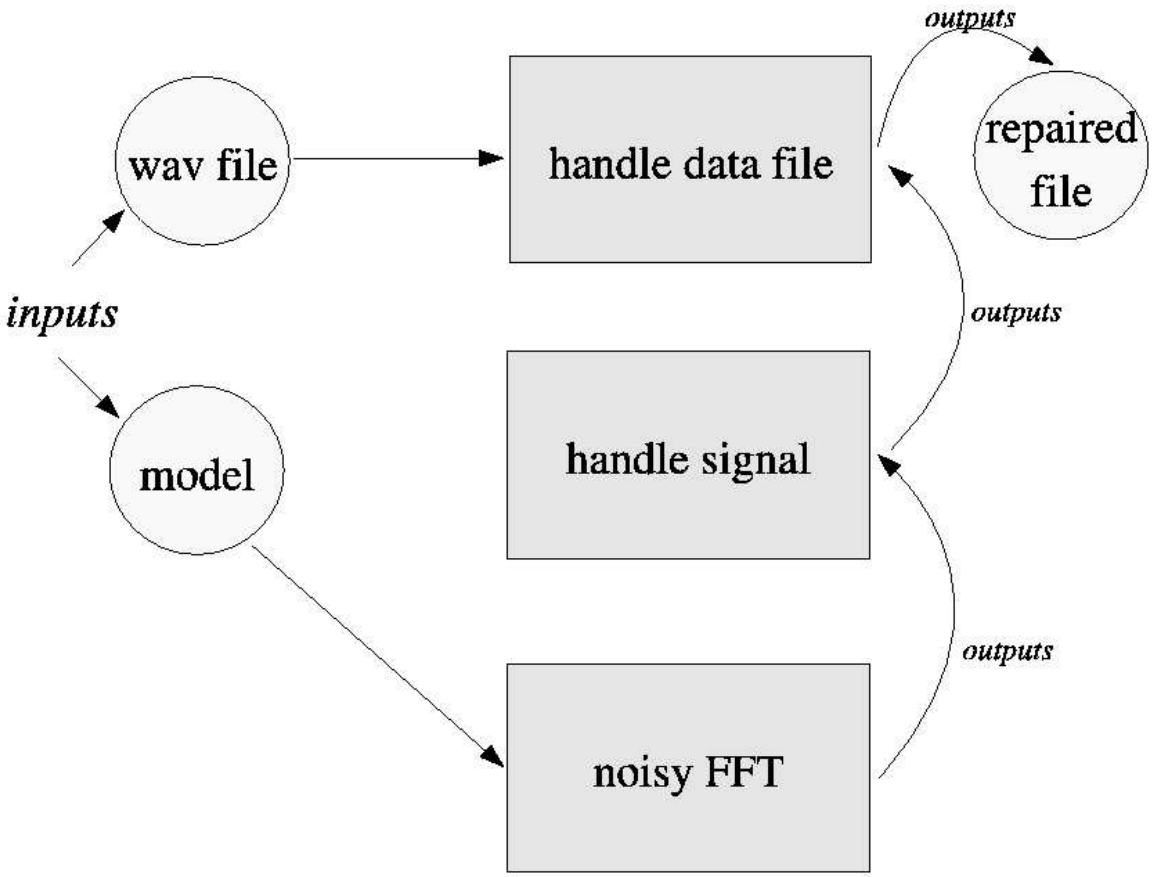


Figure 4.1: Module design

and forms the bottleneck in the system. It is therefore best implemented in a language such as C, which will be fast. By also making available a `MATLAB` implementation, which is simpler to write and use, the two versions could be compared. Optimisation methods are discussed separately in section §4.5.1; initially a conjugate gradient optimiser implemented in C was used.

Noisy FFT models In order to use a gradient optimiser, the model must be supplied in the form of an objective function and a gradient function. For simplicity, these were written in `MATLAB`. C wrappers which called the `MATLAB` functions were included in the C code.

Handling a data signal A stereo audio signal was treated as complex data, with the two parts of the signal forming the real and imaginary parts of the data. Mono data was not handled by the implementation, but it would be straightforward to modify the code to do so. The signal was broken into blocks of a fixed size—512, 1024, or 2048 points—and each block was handled separately by the probabilistic FFT core.

An arbitrary data signal is unlikely to be a multiple of a power of two. In order to handle the final points in the signal, the data can be padded with zeros. In practice, this experimental system simply ignored the final points as they represented at most one-twentieth of a second (no blocks larger 2048 points were operated on). For a practical system zero-padding would be appropriate.

Further discussion of audio data and clipped audio follows in §5, in which models for audio are explored.

Octave versus `MATLAB` `Octave` and `MATLAB` are very similar. The initial work was carried out in `Octave`, as it is free software which could be run on any machine. However, the tools available for `MATLAB` and its greater efficiency made it more suitable for this project. For example, there is no function to read a WAV format file into `Octave`; files must be converted to a format such as .au. `Octave` also has difficulty with memory management when loading large files.

4.3 Software engineering

The software engineering model was based on the spiral development model [5]. Development within this model involves building and testing a system with basic functionality, and then adding functionality and embellishments as a result of user feedback. In this case we began by implementing a very simple system which handled only one half of a stereo signal and which operated with the simplest models, and gradually built on this. Development of the the models ran parallel to development of the system, as new models threw up new issues and numerical as well as algorithmic problems.

4.4 A note on models

A full exposition of the models we used for the prior probabilities on the Fourier components is in the following section. We note here that we experimented with two sets of models, one with Gaussian priors and one with t-distribution priors. Both these probability functions involve the computation of the Mahalanobis distance,

$$\Delta^2 = (\mathbf{x}-\mu)^T \Sigma (\mathbf{x}-\mu)$$

where μ and Σ are parameters of the distribution and $\mathbf{x}$ is the data vector. Σ is a n x n matrix which we refer to below as the covariance matrix, and it is this computation on the covariance matrix that causes much of the complexity in the optimisation.

4.5 Implementation

4.5.1 Optimisation method

Conjugate gradients We considered two implementations of the conjugate gradient algorithm: `netlab`[1]'s `conjgrad`, and a C optimiser, `macopt`[2]. The netlab version was much slower, but provided useful sanity checks such as checks for the gradient

[1] `http://www.ncrg.aston.ac.uk/netlab/`
[2] `http://www.inference.phy.cam.ac.uk/mackay/c/macopt.html`

function. For a Gaussian, the conjugate gradient technique is guaranteed to converge in d steps, where d is the number of unknowns, so the maximum number of iterations can be set based on the dimensionality of the data. The t-distribution does not have this guarantee, but we found experimentally that the optimiser rarely reached the maximum number of iterations. There was some similarity between the occasions when it did reach the maximum for the Gaussian and the t-distribution. Since the failure with the Gaussian must be attributed to numerical error, we conclude that the failure with the t-distribution was also likely to be caused by numerical error.

The netlab version was also able to handle complex data. In order to use `macopt`, the complex signal had to be broken apart into the real and imaginary parts. Ultimately it was necessary to do this in any case, in order to handle data which had unknowns in only one part of the signal at any point (such as clipped data). It was still possible to provide complex parameters (mean and covariance) for the objective and gradient functions as these functions were implemented in `MATLAB` and the parameters were passed through to them untouched by the optimiser. The objective and gradient functions were modified to reconstruct complex data from the broken apart data.

As we will show in §5, the covariance matrices for blocks of audio data are often close to singular. This resulted in some numerical instability, causing the conjugate gradient optimiser to become very slow and to reach its maximum number of iterations without converging (since this convergence is guaranteed, this is an indication that some numerical error was occurring). A number of other optimisation techniques were therefore explored.

Exact optimisation This optimiser, based on code provided by Amos Storkey, computes the full posterior mean and covariance for the unknown data. The mean of this distribution is the maximum likelihood for the unknown value. This optimiser has the advantage of being “correct”, not suffering from convergence issues, and returning a distribution on the data points. However, it was very slow and scaled very badly as the block size (and hence the size of the covariance matrix) increased. This made it impractical for use on a selection of problems, although it was valuable for small tests and sanity checking on Gaussian problems. There was no equivalent code for t-distribution problems.

Quadratic programming Finding the maximum likelihood value in a Gaussian distribution is a quadratic programming problem; that is, we can state it in the form:

Optimise $\mathbf{x}^T Q\mathbf{x}$ subject to constraints $A\mathbf{x} \leq \mathbf{b}$ and $C\mathbf{x} = \mathbf{d}$.

(The constraints $A\mathbf{x} \leq \mathbf{b}$ can be used to express equality constraints, since

$$A\mathbf{x} \leq \mathbf{b} \cap -A\mathbf{x} \leq -\mathbf{b} \Rightarrow A\mathbf{x} = \mathbf{b}$$

However, it is conventional for convenience to express the equality constraints separately).

For the optimisation with Gaussian priors, the term $\mathbf{x}^T Q\mathbf{x}$ corresponds to the Mahalanobis distance and the constraints correspond to the known data points and are expressed as an equality constraint.

Code was provided by Amos Storkey which set up this problem and called the `MATLAB` quadratic programming optimiser `quadprog`.

`mosek` provides a C implementation of a quadratic programming optimiser for convex problems (that is, all the eigenvalues of Q are non-negative, as should be the case when Q is a covariance matrix). This implementation provides an interface which is compatible with `MATLAB`'s `quadprog` interface. It turned out that using the C version was much faster and returned better results. We did not investigate the reasons behind the difference in results; there is more than one way of solving quadratic programming problems and it may be that `mosek`'s algorithm is less susceptible to slight numerical error.

This code does not return the posterior distribution over the data points, nor is it suitable for non-quadratic problems, such as a t-distribution prior. However, it was satisfactory for many of the Gaussian problems in which the conjugate gradient optimiser failed to converge.

Comparisons Table 4.1 summarises the optimisers which were tested. "Sufficiently" fast is a rough measure, considered to be true if a ten-second stereo audio clip could be repaired using a joint Gaussian model in under half an hour on a 3.80GHz Pentium, while no other cpu-intensive processes are running.

Method	Arbitrary model	Posterior	"Sufficiently" fast	Numerically stable
`netlab` conj grad	Yes	No	No	No
`macopt` conj grad	Yes	No	When stable	No
Exact	No	Yes	No	Yes
`MATLAB` quadprog	No	No	No	Yes
`mosek` quadprog	No	No	For small covariances	Yes

Table 4.1: Optimisation methods

At the same time, we experimented with improving numerical stability in other ways, such as adding a jitter term or ensuring that all covariance matrices were genuinely symmetric (in particular slight numerical errors sometimes occurred in inverting the near-singular matrices, causing symmetry to be lost). These improvements permitted the conjugate gradient optimiser (the fastest) to work for some of the t-distribution problems which initially failed to converge, although it continued to run into difficulties with the Gaussian function.

4.5.2 Efficiency

In order to make the system usable, various steps were taken to improve the efficiency. Profiling helped to identify bottlenecks in the system. For example, passing the inverse covariance directly to the Gaussian p.d.f. rather than the covariance saved inverting the matrix on every iteration step. Careful ordering of equations could sometimes make computations faster.

The scaling properties of this system have two main dimensions: the way in which it scales with the fraction of the data which is unknown, and the way in which it scales with the block size. The exact and quadratic programming methods scale in the size of the covariance matrix, while the conjugate gradient method scales with the number of unknowns, which is constant over a file regardless of the blocksize. For example, the C implementation of quadratic programming on a missing data problem took two minutes to repair a file using 512-blocks, but 45 minutes to repair the same file using 2048-blocks. Although the conjugate gradient methods do not have such drastic scaling issues, in practice the number of iterations per block is rarely the theoretical maximum,

and does increase with the size of the covariance matrix. Furthermore, the work done per iteration involves multiplying by the full covariance matrix, which is more costly as the size of the matrix increases. Therefore, the conjugate gradient methods do also take longer to solve a problem as the block size is increased.

Eigenvector decomposition When working with distributions with large square covariance matrices or similar parameters, it may possible to improve performance through eigenvector analysis. If only a fraction of the eigenvalues of the covariance matrices are significant, the rest being very small, then a reduced-dimensionality approximation to the covariance matrix can be constructed

$$C = X^T \Lambda X$$

where $X = (x_1 \ldots x_n)$ consists of the eigenvectors corresponding to the n largest eigenvalues, and Λ is the n x n diagonal matrix with the n largest eigenvalues on the diagonals.

In the case of audio data, we find that many of the eigenvalues in the covariance matrix are indeed very small (the details are discussed in §5.5.10). The eigenvector approximation could therefore be appropriate, in particular for Gaussian distributions. It can be applied both when using the exact method and when using the conjugate gradient method, to simplify the computation of the Mahalanobis distance. For the exact method, there is no efficiency gain if we are modifying the covariance matrix at each step, as the eigenvectors and eigenvalues must be recomputed each time. For the conjugate gradient techniques when many iterations are carried out at each step, there may be some efficiency gain.

In practice, we found that using the an eigenvector decomposition within the conjugate gradient method, for a joint Gaussian distribution, reduced the numerical problems with the method, but was very slow. This might have been ameliorated by further rewriting the equations to reduce the amount of computation done at each step; we did not investigate further.

Preconditioning Preconditioning can be used to try to reduce the number of iterations in a conjugate gradient optimisation problem. We did not experiment with pre-

conditioning as the conjugate gradient method was for the most part sufficiently fast for my purposes. Preconditioning works by pre-multiplying the linear system by a preconditioner matrix, so it is possible that applying a preconditioner might have reduced the numerical problems with the conjugate gradient method.

The above two approaches—eigenvector decompostion and preconditioning—would have a dual effect on the conjugate gradient optimisation; the eigenvector decomposition reducing the per-step time, and the preconditioning reducing the number of steps. Further speedups might be obtained by implementing the objective and gradient functions in C rather than as `MATLAB` functions. However, this would add considerably to the complexity of use of the system and the likelihood of bugs in those functions. It would also not be possible to use netlab's `gradchek` to check the gradient function.

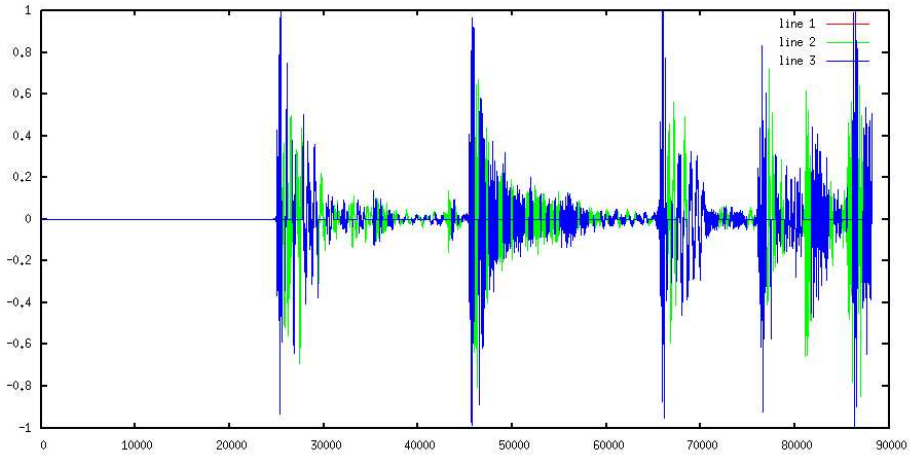
4.6 Testing

- Probabilistic FFT optimiser with conjugate gradients

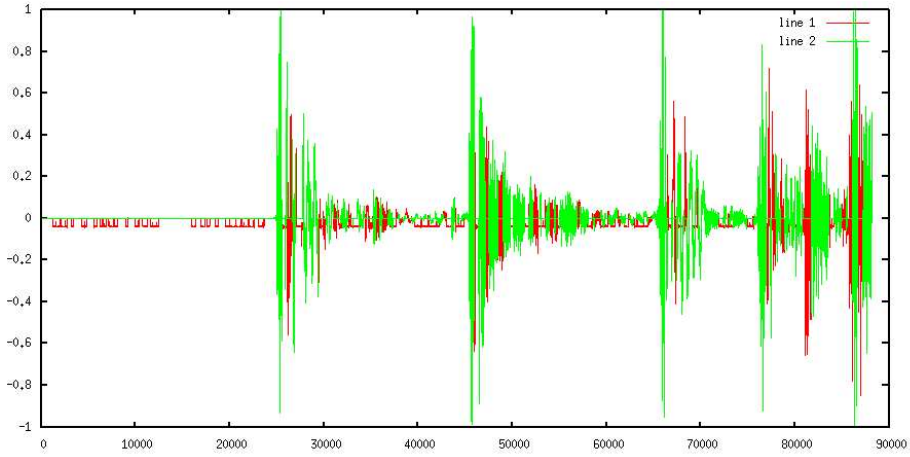
    1. Test with small vectors (4 elements) of data and a very simple model (objective function is $data - 4$, diagonal covariance). Test with vectors containing all NaNs, no NaNs, and NaNs in one or two places.
    2. Extend to non-diagonal covariance.
    3. Test with a more complex model (objective function is $data^2 - 4$). Test vectors as above. Diagonal covariance and then non-diagonal.
    4. Test with an independent Gaussian model
    5. Increase the size of the test vectors to 32 points
    6. Test on a short section of a one-dimensional audio signal, using mean and variance extracted from the signal and Fourier transforms of length 256. Figure 4.2 demonstrates the results.

- Once one optimiser was working, others could be tested by comparison.

- Results could be sanity-checked (although not guaranteed correct) by visualisation (as in figure 4.2) and on real audio data by listening (although apparently



(a) Broken data laid over clean data



(b) Repaired data laid over broken data

Figure 4.2: A small test. The data with blocks missing is shown in blue in 4.2(a), with the clean data behind in green. The repaired data is shown in red in 4.2(b), with the broken data laid over in green.

sane results do not necessarily sound good).

- At each stage of development with new models and new optimisers, we checked backward compatibility against the previous work.

4.7 Conclusions

As predicted, the bottleneck in the system was in the probabilistic FFT core. In particular when working with large covariance matrices it was necessary to experiment with techniques for accelerating the optimisers. Furthermore, the system was prone to numerical difficulties, which in places caused the optimisation techniques to struggle. A good deal of time was therefore spent in trying different optimisers and in attempting to work around numerical difficulties.

The final system was much as planned, but with a variety of choices of optimisers, not all of which were appropriate to arbitrary models. The core of the system does not depend on the input data being audio data; it could equally well be used in an application which handled image data, astrophysical series data, or geophysical data, to name three examples. In this project, we explore the use of the probabilistic FFT for audio restoration, focusing on music. We have mentioned the use of the Gaussian and the t-distribution in our experiments; in the next section we present our analyses of audio data and the justification for our model choices.

Chapter 5

Audio Data

In this project we focus on the modelling of music data. I collected a corpus of wav files containing music from a variety of genres. The files were initially stored as mp3 and good quality, typically taken from CDs.

Block size The probabilistic FFT code we have implemented will operate on chunks of data, not on the whole file at once: it is not practical to consider FFTs of a million data points in situations involving the use of large matrices or conjugate gradients. We therefore consider how large these chunks might practically need to be.

The files we used were recorded at a sampling rate (data points per second) of 44.1kHz. The Nyquist sampling theorem states that the maximum frequency which can be detected is half the sampling rate. Therefore, the maximum frequency which could be detected in these files is approximately 22kHz, just above typical human hearing limits. The lower limit of human hearing is approximately 20Hz. To detect signals at frequencies of 20Hz, we must sample twice the number of points which form a complete period of such a signal at this sampling rate. A signal at 20Hz undergoes a complete period in 2205 points when sampled at 44.1kHz. To detect these signals we would need to sample of 4410 points. The smallest power of two larger than this value is 8192, so in an ideal system we would use blocks of 8192 points. Practically, we would detect all audible signals using a block size of 4096 points.

However, for a real application it is not necessary to attain these theoretical optima for at least two reasons:

- It will be possible to estimate the presence of a signal without the full two periods theoretically required.
- Low-frequency sounds are not strongly perceived (see the Fletcher-Munson curves in figure 2.1) and are unlikely to contribute much to the signal as a whole.

We therefore chose to experiment with blocksizes no larger than 2048 points, and with a default block size of 512 points, sufficient to detect all frequencies over 200Hz (for comparison, telephones typically transmit above 300Hz).

5.1 Datasets

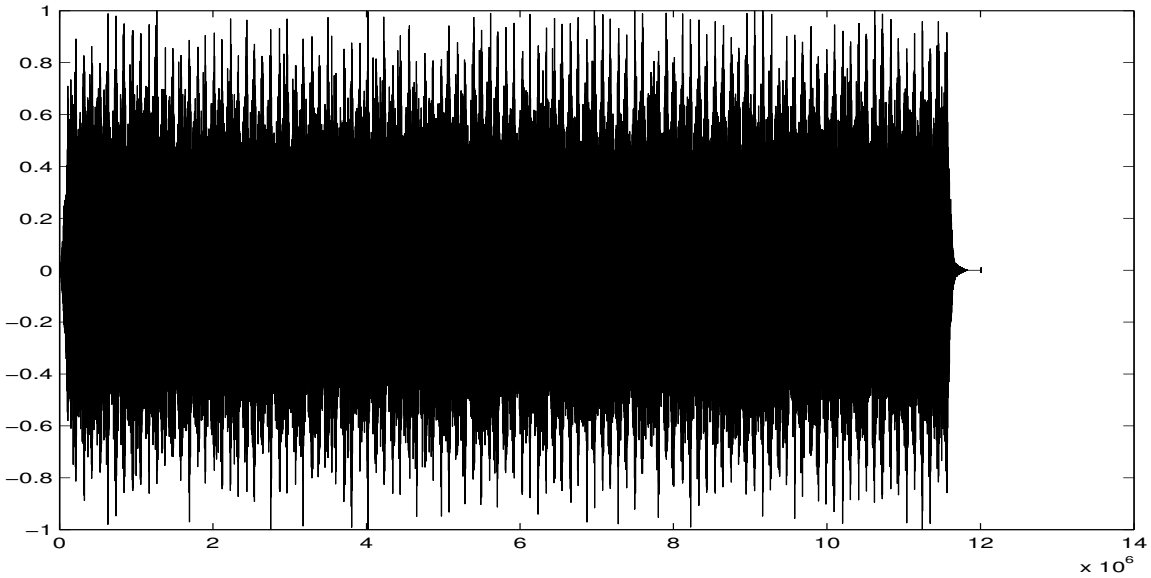
We worked with three data sets:

1. Bells: Sound of the Swan Bells, Perth
2. Songs: Songs from my collection–mainly pop songs
3. Classical: A variety of classical music from my collection

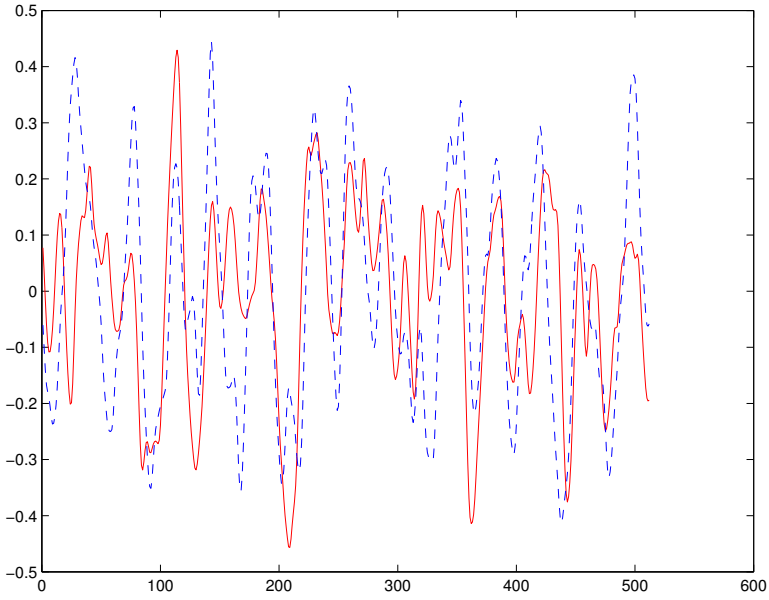
Songs and classical music provided two examples of different styles; there was a good deal of diversity within each of the two broad genres. The full data sets are listed in appendix A. The bells provided a slightly different type of data set: in these pieces there are a small number of similar instruments (the bells) each with a single note which has strong harmonics. Only one note is struck at any one time so the frequencies should be distinctive. We stored the data in .wav format which is readable by `MATLAB`.

5.2 General information

- The samples varied in length, mostly around four minutes. However, we restricted our work to ten second or thirty second samples from the files; either contiguous, or sampling small blocks (of the current block size: 512, 1024 or 2048) at random from throughout the file until we had 10s or 30s worth of data points.



(a) Whole file, left only



(b) A 512 block, left and right

Figure 5.1: Time series information for a sample file

- All the samples were recorded at 44.1kHz. It is worth commenting that if the songs are converted to 8kHz, there is no audible difference in the quality as played through a laptop and headphones, while the bells recordings are noticeably lower quality at 8kHz. This may be relevant to restoration applications as it may suggest that a higher error rate is acceptable for songs files than bells.

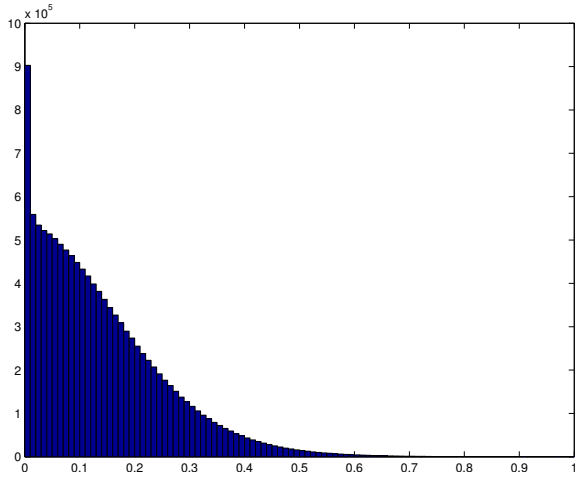
- All the samples contained stereo data. For the rest of this paper we refer to the "left" and "right" parts; the decision of which part is which is less directed, but the "left" part refers to the first column of data read into `MATLAB` by `wavread` and the "right" part the second. Where only one column of data is examined then it is the "left" part unless stated otherwise.

Figure 5.2 shows the left part of one of the "Bells" files (Bells 1), and both left and right parts of 512 block from that file. The large-scale regularity (5.1(a)) in the data points is a feature of the sound of bells ringing.

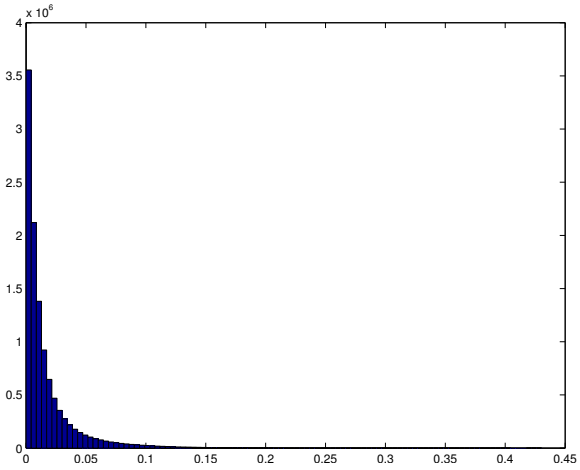
5.3 Data points

Figure 5.2 shows histograms of the size of the data points for three of the files, one from each data set. A number of differences can be seen. For example, "Morning Mood" is mostly both high and quiet, which combine to result in many of the datapoints having low amplitudes. It is also noticeable that the three histograms are shaped differently, indicating different distributions of datapoints. It is clear that these files would be represented by rather different models in the data domain. We hope that there will be more obvious regularities in the frequency domain.

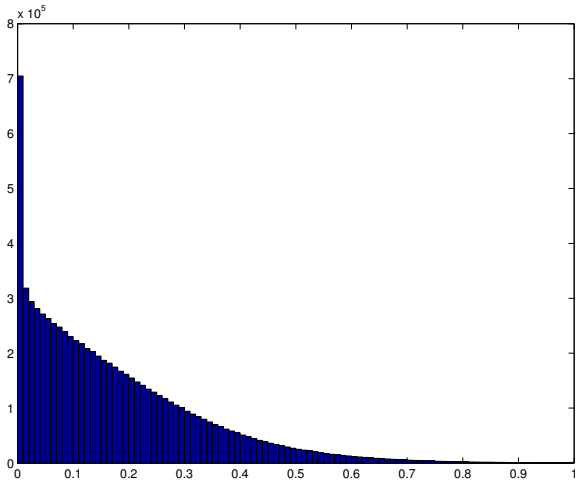
The mean value of a datapoint is around zero—silence—for each of the files. This is expected; data points are equally likely to be positive or negative. However, the mean absolute value of a datapoint varies between the files: for "Morning Mood" the mean absolute value for the left signal is 0.017; for "Wannabe" it is 0.17, ten times the size. Two of the audio tracks from the bells CD (bells 4 and bells 7) have similar tunes on the same number of bells, but are in different keys. The mean absolute values are 0.13 for the lower one and 0.083 for the higher. "Moonlight Shadow" and "Hometown



(a) Amplitude for Bells 1

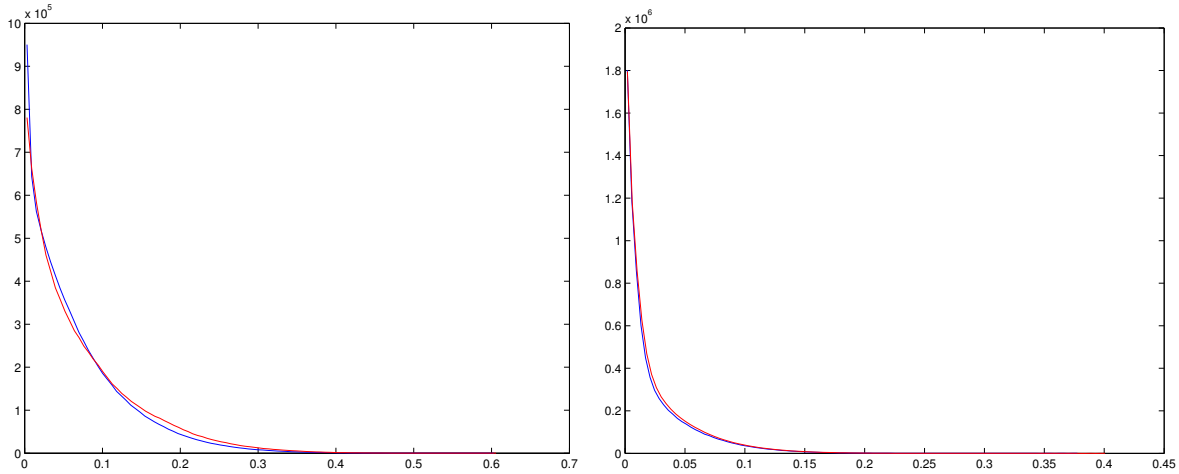


(b) Amplitude for "Morning Mood"



(c) Amplitude for "Wannabe"

Figure 5.2: Data point amplitudes



(a) Amplitudes for “Moon River”

(b) Amplitudes for “Sarabande”

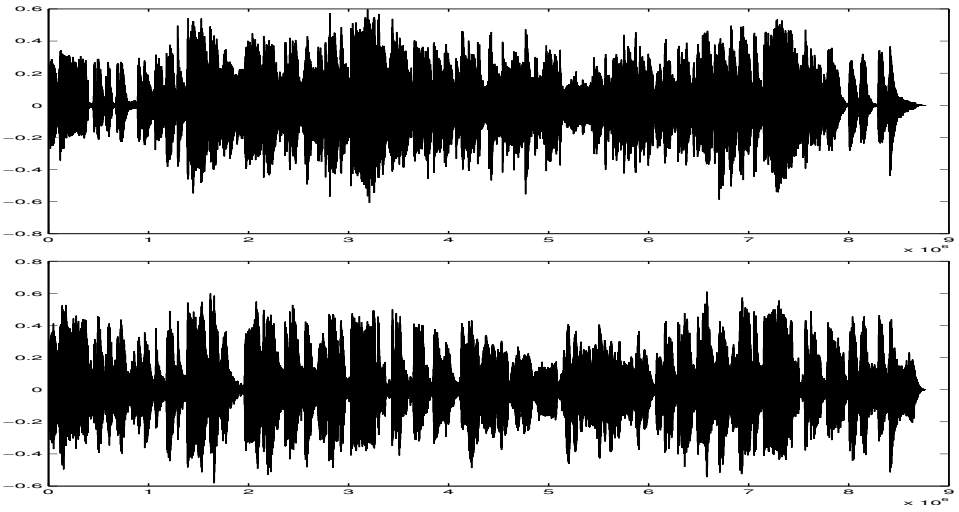
Figure 5.3: Histograms of data point values: left and right

Girl” have similar perceptual loudnesses, but the mean absolute value of datapoints in “Moonlight Shadow” is 0.077 while in“Hometown Girl” it is 0.091.

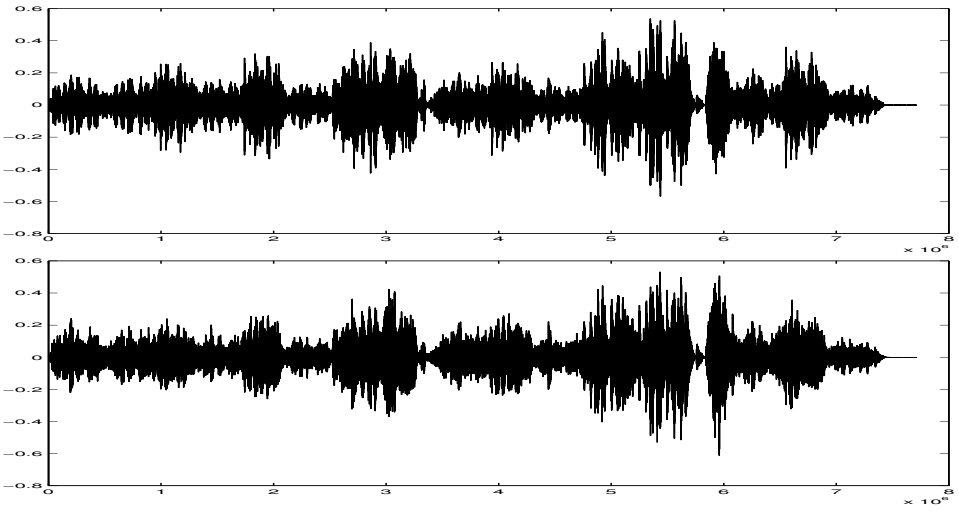
This gives us a brief overview of what kind of time-series data we are dealing with: data which does not have a clear distribution shape, data whose mean absolute value varies with both the loudness and the type of music.

5.4 Handling stereo data

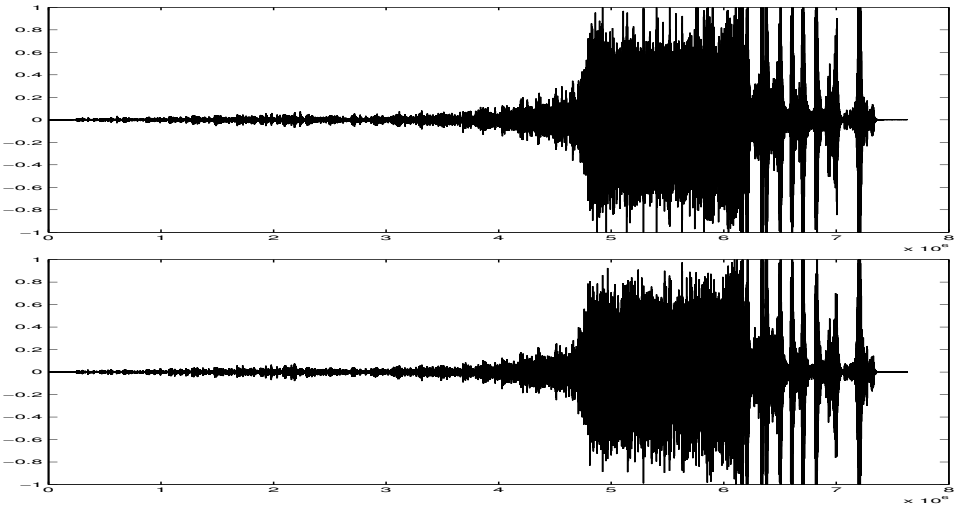
Figure 5.3 compares the histograms for the left and right parts of two files, “Moon River” (a song) and “Sarabande”: in general in music data the loudnesses of each part must be much the same, or one part would be likely to become less audible. In “Moon River” the two parts are, unusually, quite different (shown in figure 5.4), but the data histograms are still similar–in particular, they are shaped the same way. It may be that the ranges of frequencies used also vary in a similar way. In the “Hall of the Mountain King”, the left and right signals sound quite different in places, but the data plots look very similar.



Above: “Moon River”, left and right signals



Above: “Fur Elise”, left and right signals



Above: “Hall of the Mountain King”, left and right signals

Figure 5.4: Comparing the left and right signals

File	Correlation coefficient
Bells	0.416
“Hometown Girl”	0.725
“Moonlight Sonata”	0.407
“Allegro con Brio”	0.274
“Dressed for Success”	0.585

Table 5.1: Correlation coefficients between left and right data points

Here we use a method of displaying the histogram which allows us to view more than one histogram on a single plot: the curve represents a line drawn through an imaginary point at the top of each histogram bar. We will use this method in a number of later diagrams.

As seen above, the two parts generally (although not always) follow one another on a fairly large scale, although there will be differences at the small scale. The overall loudnesses will be similar, but it’s possible that some parts of the tune—a high-pitched voice or instrument, for example—may occur only on one part of the signal (this is the case with the woman’s voice which occurs briefly in “Hall of the Mountain King”, for example). Table 5.1 shows the correlation between the left and right data points for a selction of files. All of these coefficients are positive, indicating that the loudnesses of the two parts increase and decrease together, and non-zero, however they vary in magnitude and hence in significance.

In the frequency domain, this will be represented by left and right signals which tend to follow one another quite closely, although the signals themselves may vary in the details. The upsurge of one signal is likely to be near to the upsurge of the other. Looking back at figure 5.1(b) we can see this effect on the block of 512 datapoints depicted. Although there is one place at about point 160 where the two signals reflect one another, and although in places there are small “bumps” in the opposite direction on a larger downward (or upward) curve, for the most part they are indeed positively correlated.

Representation We can represent a stereo signal as complex data, with the real part representing the left part and the imaginary part representing the right part. Since Fourier transforms are complex, a complex representation of the data is appropriate for carrying out a Fourier analysis. If we were to perform Fourier transforms only on one part of the signal at a time, so that the Fourier transforms are on real data, then we will find that $F_j = F_{N-j}*$ (where * represents the complex conjugate). That is, only half the Fourier components are needed to specify the full transform.

5.5 Fourier components

In general, predictions on time series data are based on predicting data points based on the preceding data points. Predictive models must therefore capture regularities in the data. In this project we propose to try to encapsulate the data by providing prior information about the periodicity in the data—its frequency properties. The remainder of this chapter is devoted to an exploration of these properties for the music data we have.

As discussed, we intend to break the data up into chunks. Our prior information should therefore be a prior over the n Fourier components in a single chunk. In order to obtain statistical information about these Fourier components, we can examine a large number of such chunks, and extract from them appropriate patterns.

There are two steps in this process. The first is to consider the Fourier components in a chunk to be independent entities, and suggest n separate models, one per component. The second step is to consider the interactions between the components, and to propose a single n-dimensional model. The first case then becomes a special case of the second. Each of these models may be extended to a time-dependent version in which the model for a single block is dependent upon information abot the blocks which have already been seen.

The statistics were collected by selecting samples worth 30s of data from the audio files (as described in §5.2), breaking them into blocks, and computing the Fourier transform of each block. These collections of Fourier transforms could then be examined to suggest properties which could be used to model the components of the Fourier

transforms. Unless otherwise stated, we use blocks of 512 points.

We begin by examining some general properties of the individual Fourier components.

5.5.1 Phase independence

Figure 5.5 shows scatter plots of the real and imaginary components for a set of data (The "Bells 1" file). From figure 5.5(a) we can see that the frequency components at each end of the transform have independent real and imaginary parts. The first component, representing the data points, is rather the odd one out: we see some correlation in this component (shown in blue) as the amplitudes of the left and right parts of the signal will be correlated (as discussed previously). For the very small components around the middle of the transform (in this case, around 250; we saw similar effects for the components around 120 when looking at transforms on 256-blocks), we see that there is positive correlation between the two parts.

For a block size of N, the formula for the (N/2) Fourier component is

$$F_{N/2} = \sum_{n=1}^{N} f_n e^{-i\pi 2n(N/2)/N}$$

$$= \sum_{n=1}^{N} f_n e^{-in\pi}$$

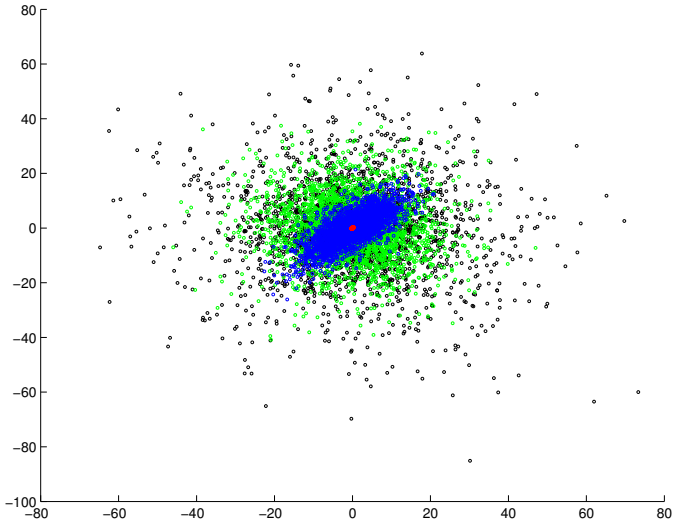
applying de Moivre's theorem:

$$F_{N/2} = \sum_{n=1}^{N} f_n(\cos(-n\pi) + i\sin(-n\pi)$$

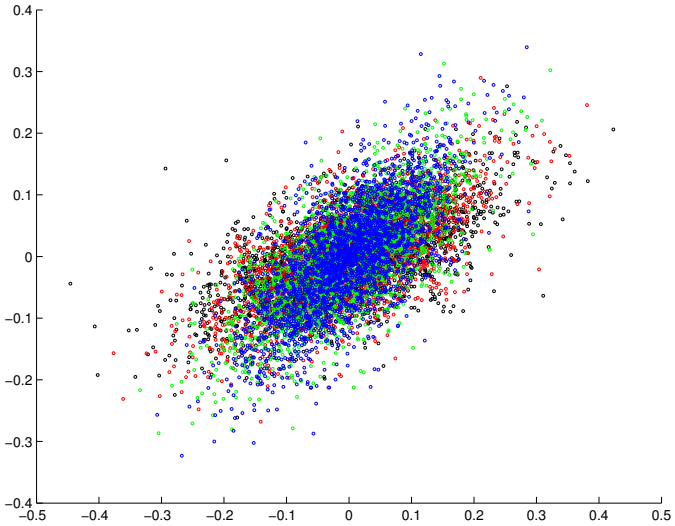
$$= \sum_{n=1}^{N} f_n((-1)^n)$$

$$= \sum_{n=1}^{N/2} f_{2n} - \sum_{n=1}^{N/2} f_{2n+1}$$

If, as we expect, there is no particular bias towards odd or even values, this will tend towards zero. There is no obvious reason for positive correlation to occur. It is possible that the small values are the results of small numerical inaccuracies. However, since



(a) Scatter plot showing real and imaginary parts for a sample of the fourier components: black: component number 510, green: component number 5, blue: component number 1, red: component number 255



(b) Scatter plot showing real and imaginary parts for a sample of the fourier components around 250: black: component number 200, green: component number 220, red: component number 250, blue: component number 270

Figure 5.5: Scatter plots comparing real and imaginary components of the Fourier transform

these components are generally so very small, it is unnecessary to be too concerned about the slight anomaly.

In conclusion, it will be sufficient, when examining the properties of the Fourier transforms, to look just at the real parts of the components; the spherical nature of the significant parts of the scatter plots (aside from the first component) allows us to assume that the real and imaginary parts will have the same properties.

5.5.2 Shape equivalence

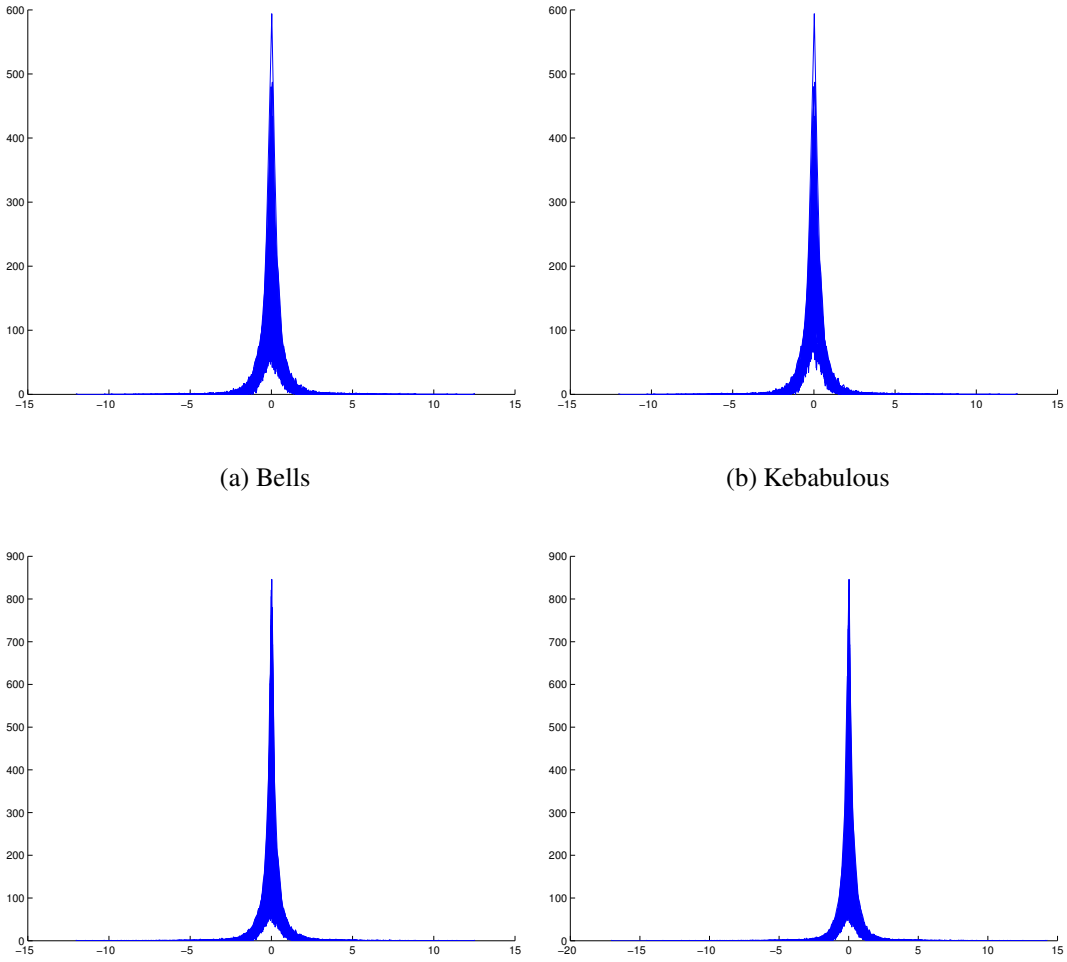
We consider just the shape of the Fourier components, after normalising the data to zero mean and unit variance. This enables us to see that the Fourier components for a particular music file all have the same shape—that is, they can be represented by the same distribution, although the (mean and variance) may not be the same for each. This provides us with justification for examining in detail only a small number of the Fourier components, and assuming that the behaviour of the other components will be similar.

It is also noticeable that the shapes differ very little between the four files, although the audio data in each is aurally very different.

5.5.3 Time dependence

We collected two sets of "30-second" data; one by taking blocks at random from throughout the file until there were sufficient blocks to represent 30 seconds of data (at 44.1kHz, that's 1323000 data points, or 2584 512-blocks), and one by taking 30 seconds of continuous data. Figure 5.7 compares some of the histograms for the tenth Fourier component. In each case the data taken randomly from throughout the file has a heavier-tailed distribution than the continuous data. Since there are the same number of data points contributing to each histogram, it must be the case that there is more variation over a shorter part of data than the average variation over the whole file. Thirty seconds represents a large number of data points and it is surprising that such a relatively large period of time still shows this effect.

From this, we can conclude that the use of available local information in the model



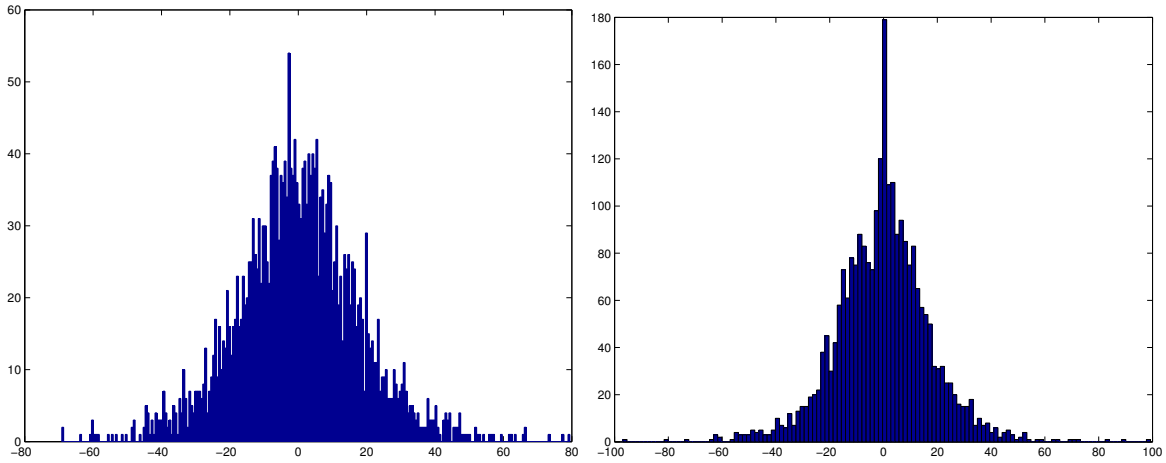
(a) Bells

(b) Kebabulous

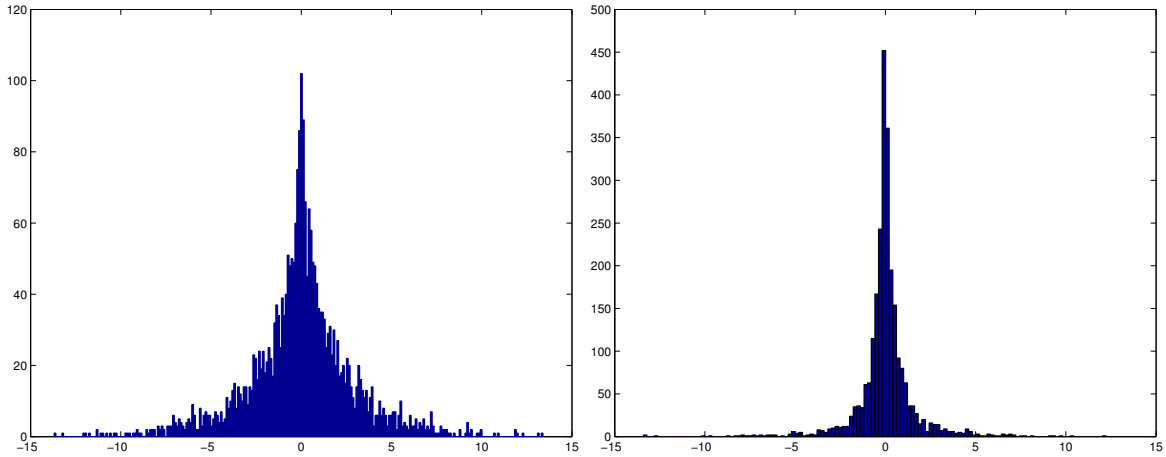
(c) Allegro Con Brio

(d) Ride of the Valkyries

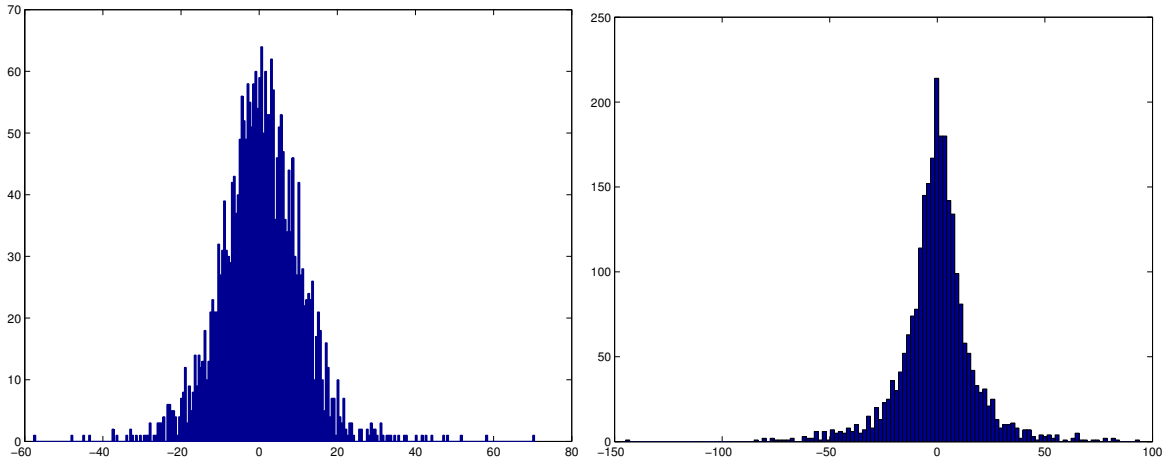
Figure 5.6: Overlapping histograms of the value of the normalised Fourier components for a selection of the data files



(a) Bells 1



(b) Morning Mood



(c) Caged

Figure 5.7: Histograms of the value of the tenth Fourier component for a block of contiguous data (left) and data taken from throughout the file (right).

Block size	Mean	Variance
512	0.3369 + 0.0877i	488
1024	-0.4752 + 0.1096i	1106
2048	-0.9624 + 0.8896i	3952

(a) Effect of block size: As Cool As I Am

Block size	Mean	Variance
512	0.1836 + 0.3049i	268
1024	-0.5559 + 0.6026i	368
2048	0.5516 - 0.1497i	1081

(b) Effect of block size: Never-Ending Story

Figure 5.8: Effect of varying block sizes on the parameters for the tenth Fourier component of two different files

may be helpful.

5.5.4 Block size

We used a default block size of 512, appropriate to cover all audible frequencies. However, it is worth noting what the effect of block size may be. Table 5.8 looks at the parameters for the 10th Fourier component for two of the files. In each case it is clear that as the block size increases, so does the variance, resulting in a broader distribution.

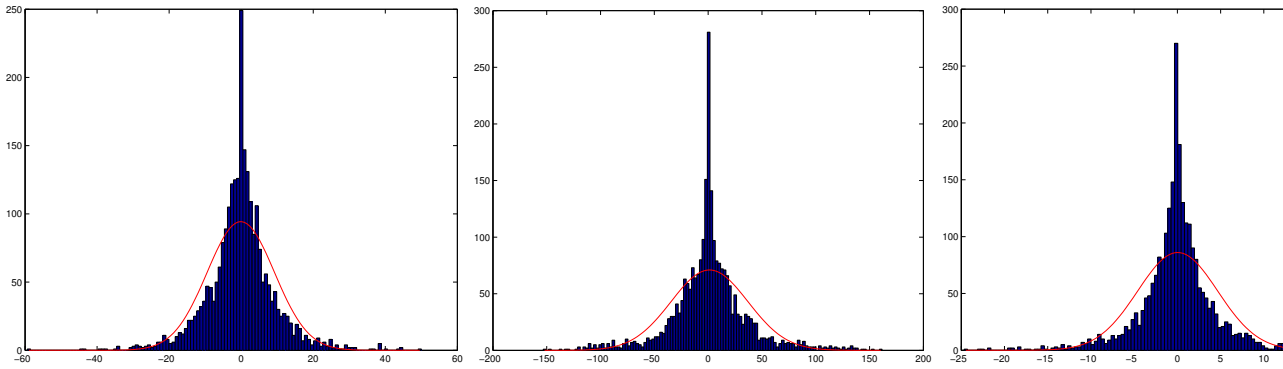
5.5.5 Model shape

Having looked at some general properties of the Fourier components, we have proposed that if we treat components independently, each of the components could be described by the same distribution, although with potentially different parameters §5.5.2. We now attempt to determine a suitable distribution to describe these components.

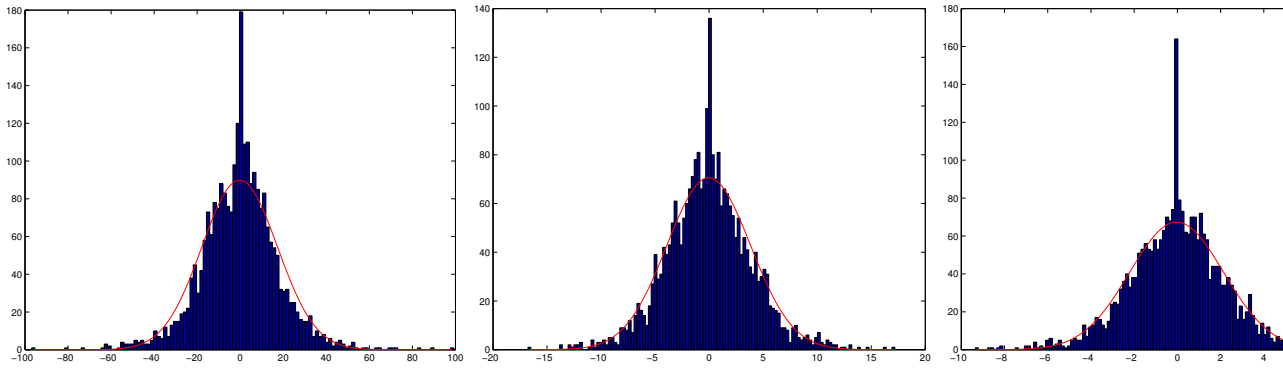
5.5.5.1 Normal distribution

A distribution which occurs commonly in many applications is known as the “normal distribution”, or the the “Gaussian distribution”. In this section we consider whether a Gaussian distribution is a good fit for our data.

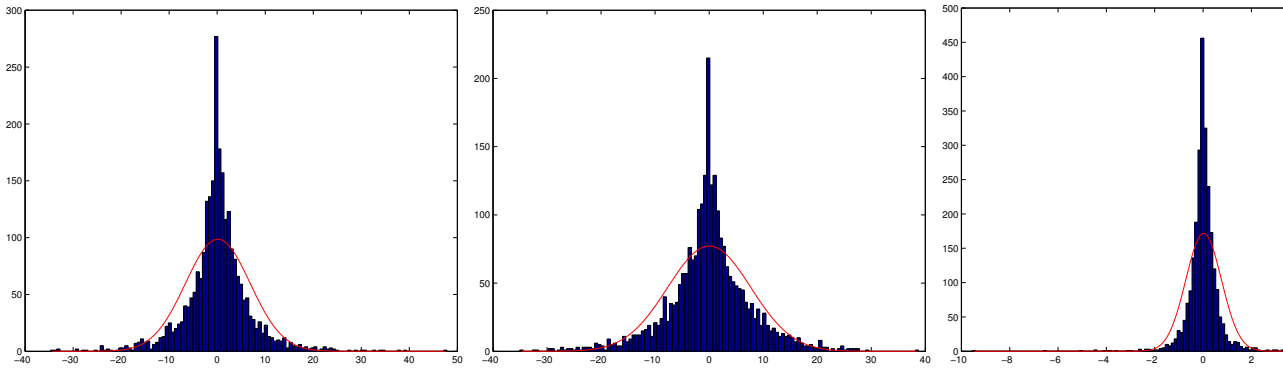
Figure 5.9 shows the histograms for a selection of Fourier components for several of the files with normal distributions (with the appropriate parameters) superimposed in red. The goodness of the fit varies between files and components; for the bells a



(a) Bananarama–"Venus": log likelihoods -15388, -18855, -13574



(b) Bells: log likelihoods -17023, -13117, -11622



(c) "Pomp and Circumstance": log likelihoods: -14545, -14890, -8823

Figure 5.9: Normal distributions over histograms for 10th (left), second (middle) and first (right) Fourier components

normal distribution is a fairly good fit; for "Pomp and Circumstance" it is repeatedly too broad. It is interesting, then, that the likelihood of the data is conistently higher for "Pomp and Circumstance" than the Bells. The likelihoods are higher for the less common components, perhaps indicating that these components are more the result of random effects than particular frequency information in the pieces.

5.5.5.2 t-distribution

Although the Gaussian distribution might form a useful first approximation to the data—useful because it is common and has useful analytical properties—these distributions are not so much bell-shaped as tight around the mean and heavy-tailed. The one-dimensional Student's t-distribution is such a heavy-tailed distribution. The t-distribution is described by the mean, a scale parameter S, and a third parameter ν, representing "degrees of freedom"; smaller values of ν result in heavier tails (this can be seen on the plots below). In the limit where $\nu \to \infty$, the distribution is Gaussian. The univariate t-distribution with parameters (μ, S, ν) is given by:

$$P(x|\mu, S, \nu) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)(S\nu\pi)^{1/2}} \left(1 + \frac{\Delta^2}{\nu}\right)^{-(\nu+1)/2}$$

where

$$\Delta^2 = \frac{(x-\mu)^*(x-\mu)}{S}$$

is the squared Mahalanobis distance from x to μ (* denotes the complex conjugate).

While for the Gaussian distribution there are closed forms to estimate the parameters of the distribution from data, for the t-distribution there are no such closed forms and the parameters must be learned via an iterative procedure. The parameters interact with each other, so cannot each be optimised separately. Ideally, all three parameters should be optimised together. [18] describes an E-M algorithm for doing this (which is also applicable to a multivarate t-distribution). However, optimising the three parameters together is difficult and may converge to local rather than global optima. We used code which was available to me based on the theory in [4] to learn values for (μ, S) for a given ν. To estimate an optimal value of ν, the following search procedure was used:

1. Try values at intervals of 5 between 1 and 100. Let the best be ν_1
2. Try values at intervals of 1 between $\nu_1 - 5$ and $\nu_1 + 5$. Let the best be ν_2
3. Try values at intervals of 0.1 between $\nu_2 - 1$ and $\nu_2 + 1$. Let the best be ν_3

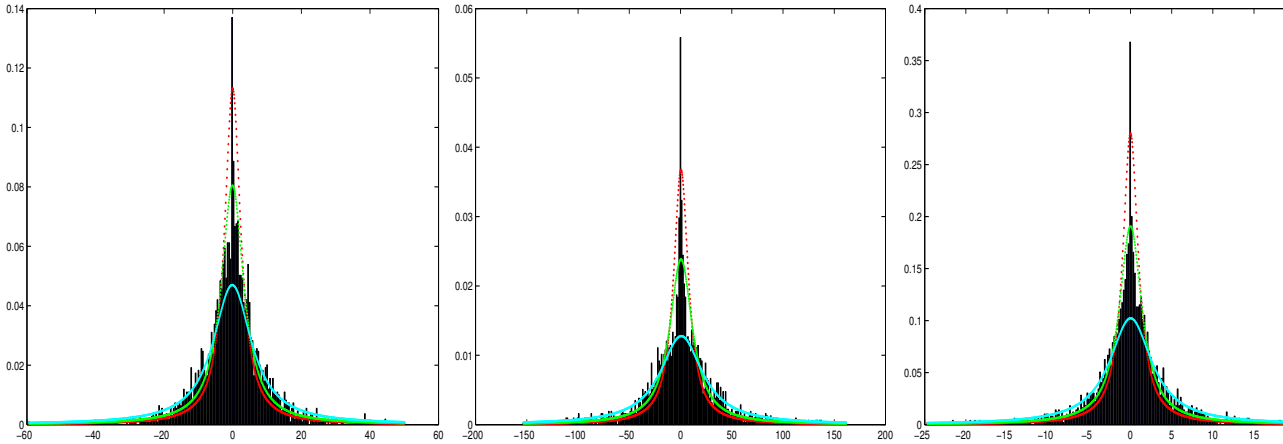
where 'best' refers to the greatest maximum likelihood of the data after learning the mean and covariance parameters. This procedure does not come with any guarantees, even to find a local optimum (beyond the one in step 3). However, we hoped that for most cases it would suggest a "good" value for ν. In practice we found that similar results were obtained on each of the data files. Taking the tenth Fourier component as representative of all the components in each case, we found that for 512-blocks, typically the ν selected was around 6 or 7; for 1024-blocks around 8, and for 2048-blocks around 10.

Figure 5.10 shows the components with a learnt t-distribution superimposed. These plots were generated using the aforementioned code. The likelihood for the best of the three curves is noted with the plots. In all cases there was little difference between the likelihoods for the three curves.

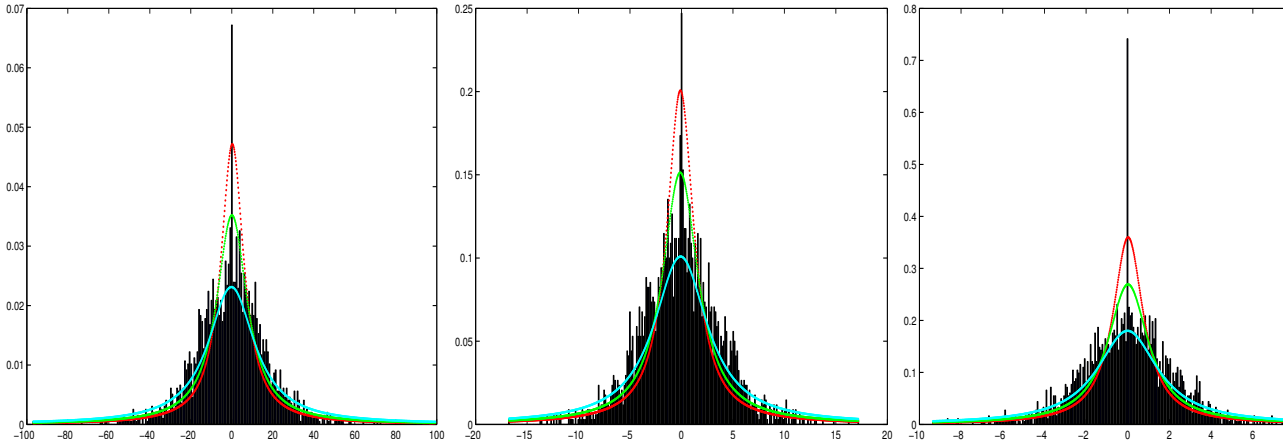
The likelihood values are higher than those for the normal distribution, and we can see the curves fitting better to the data histograms. Again the highest likelihoods occur with the component 50, generally the least used of the three. However, it is the sparser distribution which fits best to this component.

In figure 5.6 we observed that subject to normalisation to zero mean and unit variance, the histograms had the same shape. However, here it appears that different values of ν appear best for different distributions. This appearance may be misleading; we are looking only at a few specific values of ν so it may be the case that there is some other value which fits all of the histograms better. For each value of ν a different mean and covariance will be learnt.

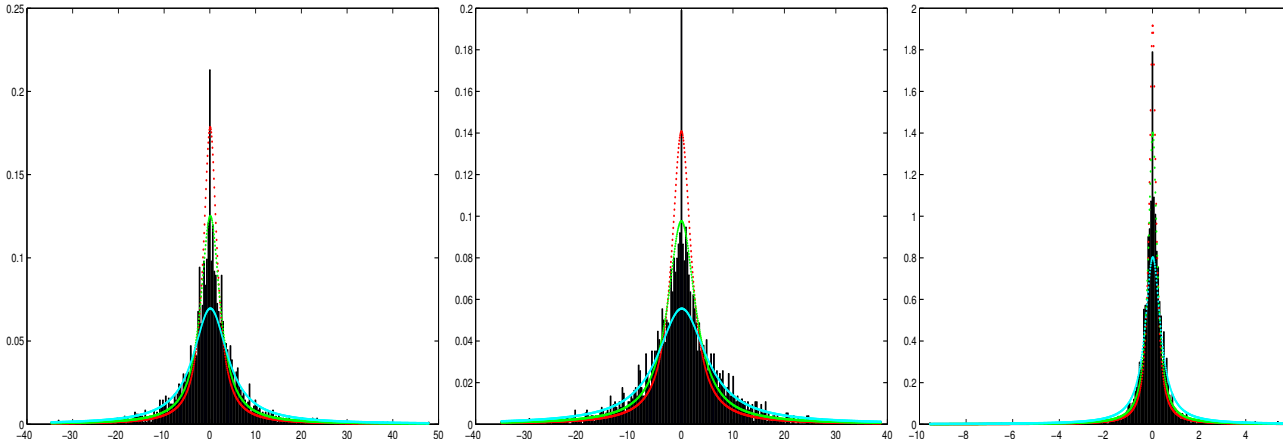
In conclusion, the Gaussian distribution may provide a convenient approximation to the probability density function, but we expect to obtain better results by using a student's t-distribution. Having looked at the general shape of these components, we now examine in detail some of the parameters of the distributions.



(a) Bananarama–"Venus": best likelihoods -9250($\nu = 5$), -12680 ($\nu = 5$), -7317 ($\nu = 1$)



(b) Bells: best likelihoods -10993 ($\nu = 5$), -7150 ($\nu = 5$), -5654 ($\nu = 5$)



(c) "Pomp and Circumstance": best likelihoods -8306 ($\nu = 5$), -8802 ($\nu = 5$,) -2068($\nu = 1$)

Figure 5.10: t-distributions: Red: $\nu = 0.1$, Green:$\nu = 1$, Cyan:$\nu = 5$

5.5.6 Fourier mean

This is the mean value of the Fourier transforms of the data. It can be computed by taking all the n-dimensional Fourier transforms as n-dimensional vectors, summing them, and dividing by the total number of these vectors. It should represent an "average" Fourier transform.

Since the Fourier transform is a linear transformation, and since the mean value of the datapoints is approximately zero, the mean of the Fourier transforms of the data is the Fourier transform of the mean of the data, and the Fourier transform of zero is zero (there are no frequency components). That is to say, there is no prior information about what the expected value of a particular Fourier component may be. This is because any component is as likely to be $-z$ as it is to be z, for any given z

5.5.7 Spectral mean

Although a particular Fourier component may be as likely to be $-z$ as it is to be z, it is not necessarily the case that its absolute value is as likely to be y as it is to be z, for given y and z. To investigate the expected absolute value of the components, we look at the power spectrum. If $\mathbf{F}$ is a vector of Fourier components, the corresponding power spectrum $\mathbf{P}$ is defined by:

$$\mathbf{P} = \sqrt{\mathbf{F}^* . \mathbf{F}}$$

where * denotes the complex conjugate. The expected power spectrum can be computed from a collection of power spectra using the same procedure as we proposed to compute the mean of the Fourier components.

Figure 5.11 shows the expected power spectra for a selection of the audio files. We can see that the Fourier components at either end of the spectrum—representing the highest and the lowest frequency components—have the highest coefficients, while the ones in the middle are non-existent. This explains a single Fourier transform in terms a small number of high frequency components which "sit on" a line defined by the combination of a small number of sinusoidals whose period is almost the whole width of the transform. Figure 5.12 demonstrates this shape for a case where there is

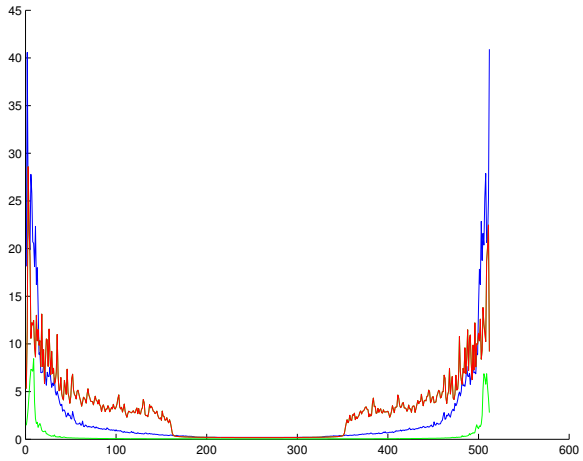


Figure 5.11: Power spectra–Red:"Dressed for Success", Blue:"Caged", Green:"Fur Elise"

one small frequency and one large. In the audio case there is more irregularity over the signal. We can also see that the range of frequencies used varies between files, with "Dressed for Success" (voice and backing) having a wider range than "Fur Elise" (piano).

Figure 5.13 shows the power spectra for "Never-ending Story" for a number of different blocksizes. We can see from figure 5.13(a) that although the Fourier components become almost zero at a frequency around 20, they do not quite reach zero on this smaller block size. This suggests that although most of these frequency components are only contributing tiny amounts to the spectrum these amounts are not non-zero. However, as explained above, the contribution should be very close to zero for the N/2 component; the small amounts may be numerical issues. This means that to obtain enough information to represent the full frequency range, we are unlikely to need the full 512-blocks. However, for the reconstruction process it may be beneficial to use larger block sizes in order to use a broader range of known information in the optimisation.

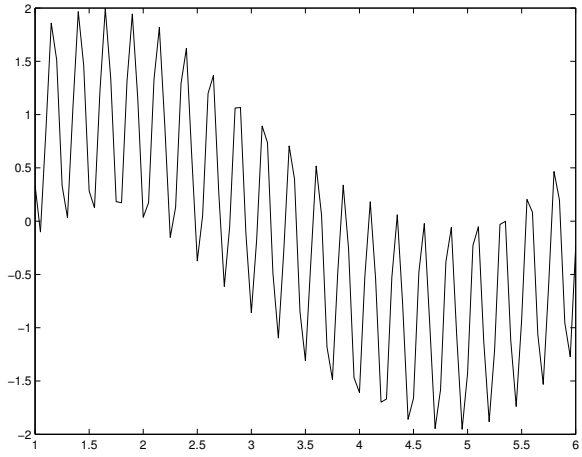


Figure 5.12: A simple periodic function consisting of the sum of a high and a low frequency component

5.5.8 Variance

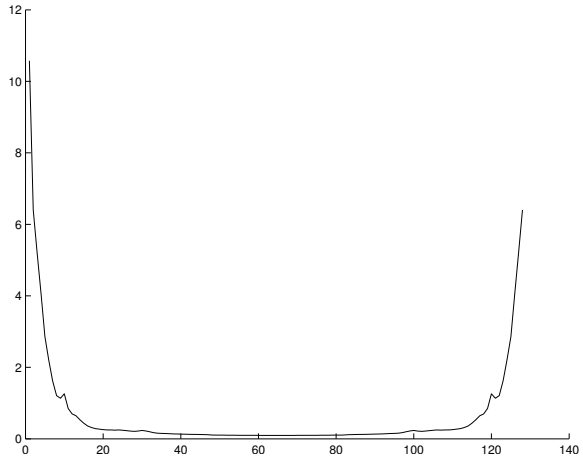
The standard deviation of a variable is roughly an indication of how likely a particular instance of the variable is to be close to the mean. Broadly, if the standard deviation is large, then the variable will often be much larger or smaller than its mean, as the standard deviation tends to zero then we can become increasingly certain of seeing the variable at its mean. The square of the standard deviation is known as the variance. For an arbitrary one-dimensional dataset, $(x_0 \ldots x_n$ with mean μ,

$$Var(x) = \sum_{k=0}^{n} (x_k - \mu)^* (x_k - \mu)$$

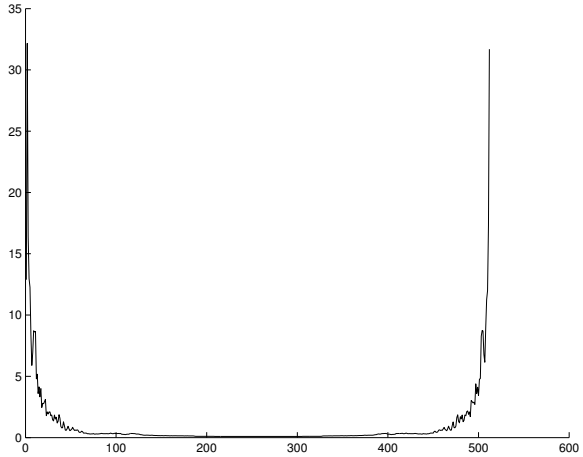
(where * denotes the complex conjugate).

Figure 5.14 shows the standard deviation of the Fourier components for a selection of the data files. The shape of the plot follows the power spectra: where values are likely to be large, so is the variance.

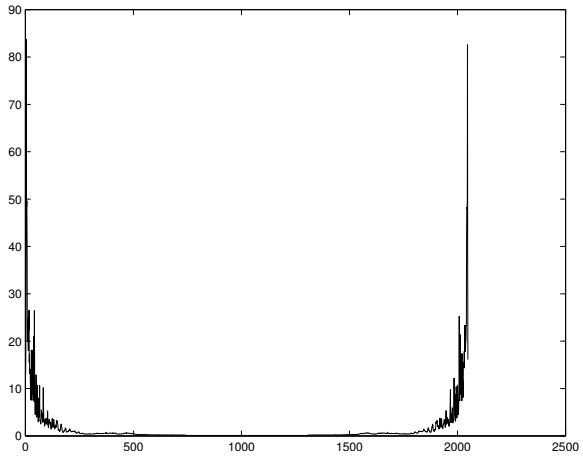
We can conclude that over all the data, some frequencies are more commonly used than others. For those which are more common, the values associated with them vary considerably over a file. The most commonly used frequencies vary from file to file, although they are in a similar range around the highest and lowest Fourier components.



(a) 128-blocks



(b) 512-blocks



(c) 2048-blocks

Figure 5.13: Power spectra–varying blocksizes

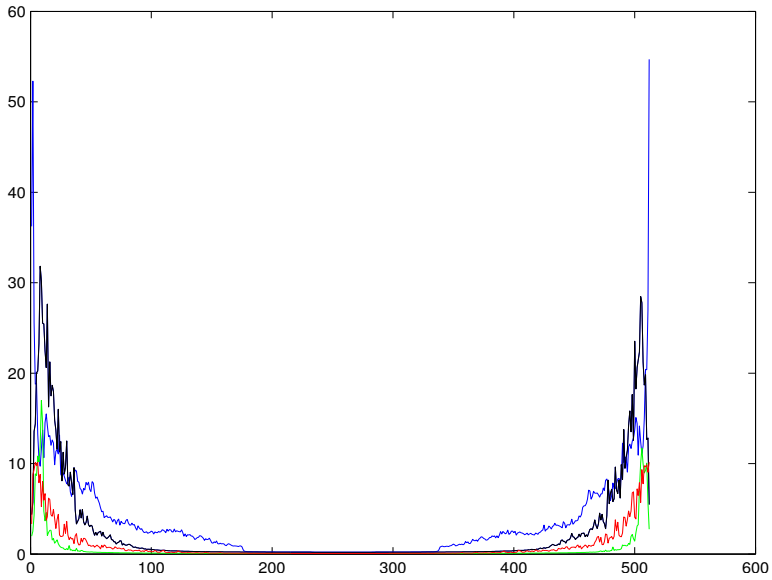


Figure 5.14: Standard deviation of Fourier components–Blue:"Venus" (Bananarama), Black: "Bells 1", Green: "Fur Elise", Red: "Pomp and Circumstance"

5.5.9 Covariance

The covariance is an extension of the variance to a multi-dimension dataset. A covariance matrix contains a representation of the interactions between variables,

$$C_{ij} = \sum_{k=0}^{n} (x_k^i - \mu^i) * (x_k^j - \mu^j)$$

Such a matrix has the variances on its diagonals. The value in C_{ij} is a measure of how much information we have about x_i if we are given x_j. This particular measure is a symmetric measure; covariance matrices are symmetric.

Since, as discussed, the mean of the data is approximately zero, the covariance matrix is what we expect to form the primary differentiator between files, and the nub of the models we will develop. We therefore spent some time examining the properties of the covariance matrices.

The covariances between two distinct components (i,j) are much smaller than the variances (the covariance between a component j and itself); for example, the maximum variance in "Hometown Girl" is 864 but the maximum covariance between distinct components has absolute value 91. The mean of the variances is 13; if we take

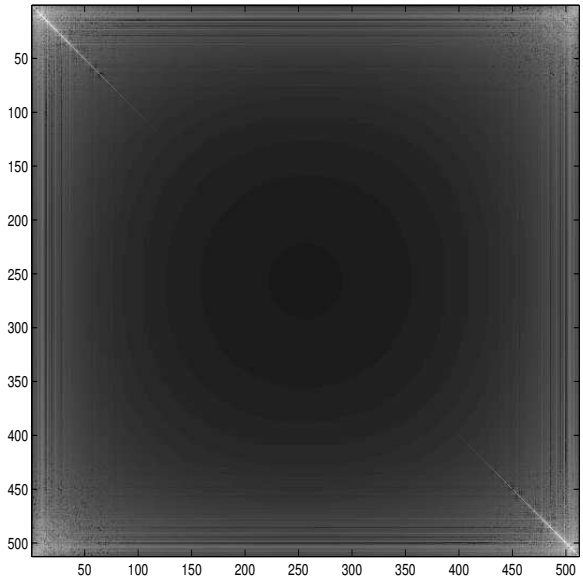
the maximum covariance for each component (that is, for each component we consider only the distinct component with which it is mostly highly correlated), the mean of these covariances is 3.7. For a bells file, the maximum variance is 379 and the maximum covariance between distinct components 133. The equivalent means are 11.7 and 5.0. For "Marlene on the Wall" the differences are even more pronounced: 460 and 2.7 for the two maxima, and 9.9 and 1.0 for the two means.

In all the diagrams shown below we take the log of absolute value of the covariances (which are complex); using the log is necessary if the high variances are not to dominate the plots. Unless stated otherwise, all the plots are for 512x512 matrices using data from transforms on 30 seconds worth of 512-blocks selected randomly from throughout the file.

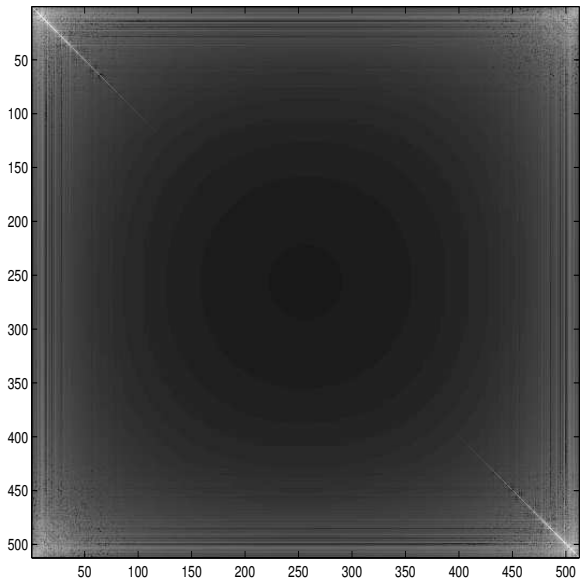
Figure 5.15 shows separate covariance matrices for the left and the right signals. Other files showed a similar effect to these: the same patterns for each part of the stereo signal, but one part tending towards higher covariances than the other (a brighter plot). This was true of all the files we looked at, including the Allegro Con Brio which has the lowest correlation coefficient among its left and right data points (table 5.1). We therefore conclude that we can see all the interesting features of the matrix by looking just at the covariance matrices for the combined complex signal.

Figure 5.16 shows a close up few of the corners of two of the covariance matrices. The corners, corresponding to the high and low frequencies (which we have already shown are the most commonly used) are where most of the action is to be found. Figure 5.16(a) shows a corner of the covariance matrix for 'Fur Elise', a piano piece. The white diagonals show the variance; in general the lighter colours represent higher values. From the figure we can see that there are some 'stripes' of high or low covariance, representing bands of frequencies which are more common in the piece, or under-represented. We can also see some blockiness around the higher frequencies (1–50), representing frequencies which tend to occur together. For the bells file 5.16(b), clear stripes can be seen, representing the notes of the bells and their harmonics, but there is almost no blockiness (interaction between frequencies) beyond some variation around the diagonal.

Figure 5.17 and 5.18 show the full 512x512 covariance matrix for "Fur Elise", and

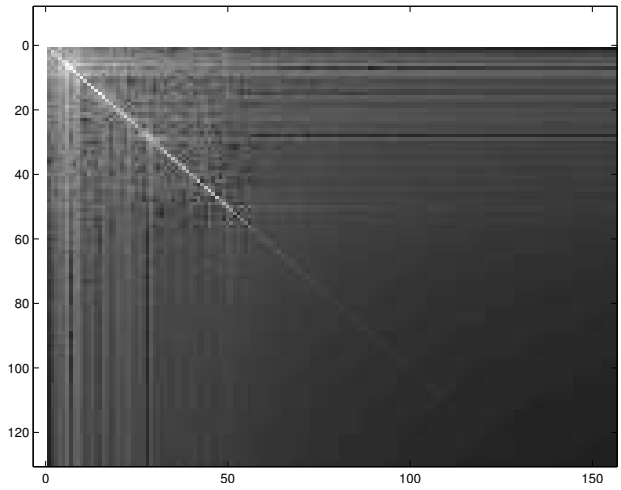


(a) Covariance matrix for left signal

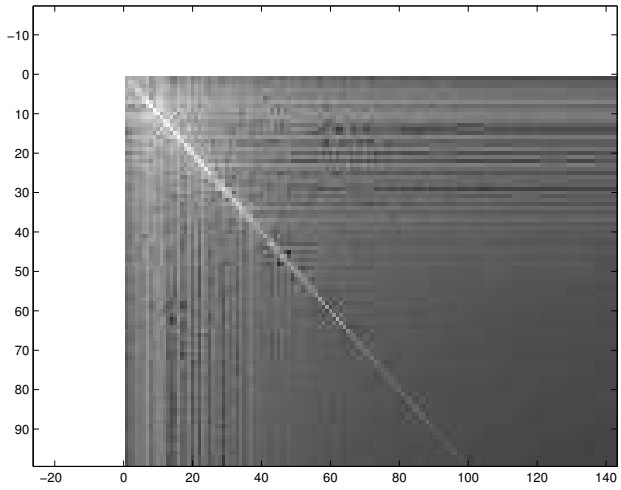


(b) Covariance matrix for right signal

Figure 5.15: Comparing real and imaginary parts of signal for bells file



(a) Corner of “Fur Elise” matrix



(b) Corner of “Bells 1” matrix

Figure 5.16: Close-ups of two covariance matrices

the equivalent matrices for a selection of other pieces: another piano piece, "Moonlight Sonata"; a woman's voice in "Dressed for Success" and a man's in "Hometown Girl"; a group of voices in "Wannabe"; a 'cello piece, and an orchestral piece, "Allegro con Brio". Looking at these we can start to see ways in which the covariance matrices can differentiate between pieces according to certain characteristics:

- The single instrument pieces (5.15, 5.17(a), 5.17(b), 5.18(b)) have a small number of stripes around the edge of the matrix, representing the frequencies associated with the instrument.

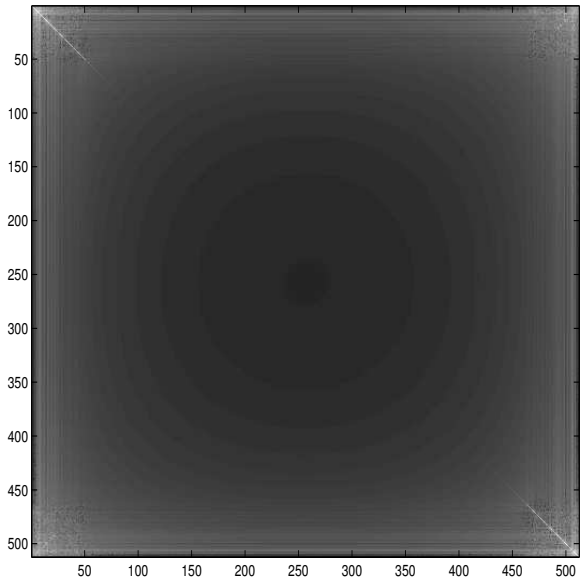
- The voice pieces (5.17(c), 5.17(d), 5.18(a)) have a block of lighter colour somewhere along the diagonal, presumably representing the frequency ranges of the singers. The block in 5.17(d) is more towards the centre of the matrix than the ones for the female voices, suggesting that the artist's voice is around 450Hz (Fourier component 100, corresponding to a signal with a complete period once every 100 datapoints), towards the low end of typical human voice frequency.

- The pieces with more singers (5.18(a)) or instruments (5.18(c)) tend to have larger blocks of interaction, representing a broader range of notes (frequencies) available.

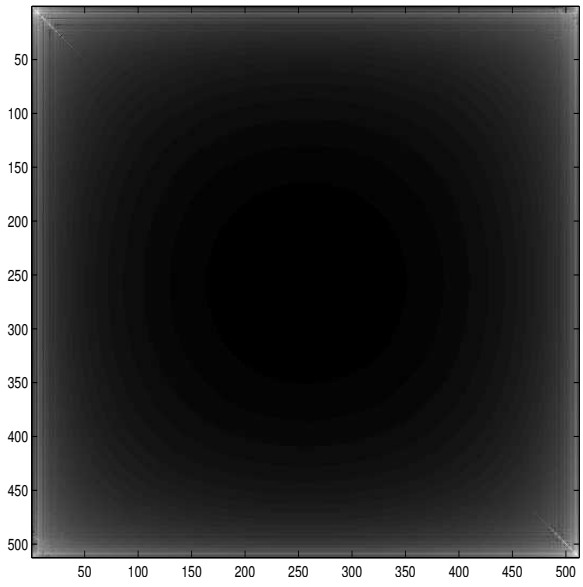
- The central part of the matrix has a circular pattern for instrumental pieces, and a square pattern for voice pieces. This suggests a much sharper cutoff in the frequencies in the voice pieces; perhaps a lack of harmonic information. The difference between the two similar piano pieces, 5.17(a) and 5.17(b) suggests that this effect is very much instrument-specific.

Figure 5.19 show close-ups of two songs by the same artist (male voice and guitar). It's possible to see similar shapes within the images, suggesting that the characteristics of the singer are important in determining the distribution of the frequency transforms. Figure 5.20(a) shows a close-up from the "Moonlight Sonata"; we can compare this with "Fur Elise" 5.16(a): the two have many similarities.

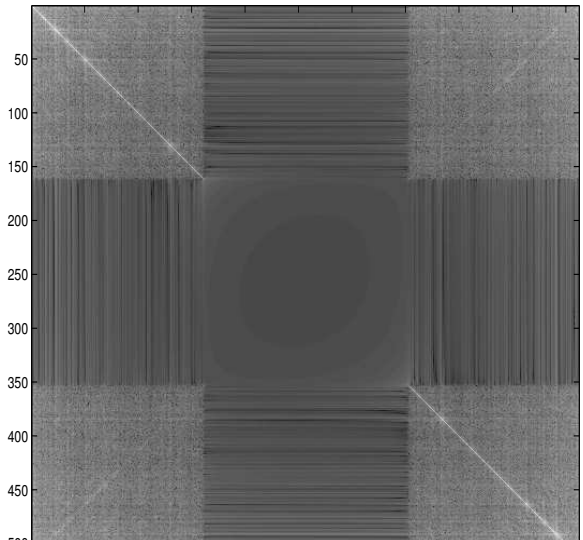
Finally, figure 5.21 shows the covariance matrices for two different pieces on exactly the same instruments: the Swan bells, Perth. These show the strong circular

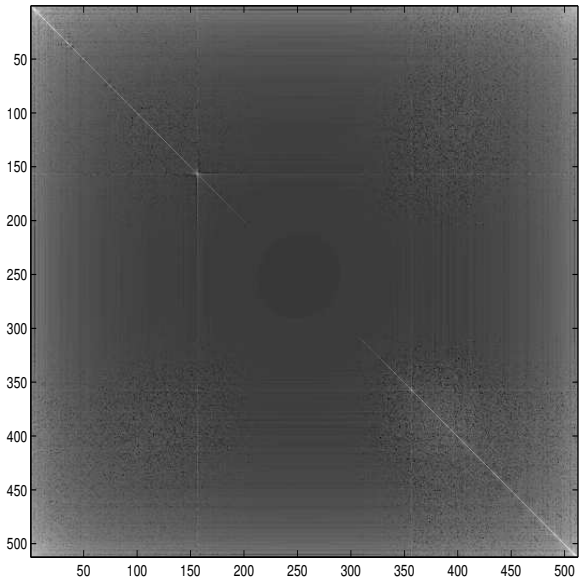


(a) Fur Elise

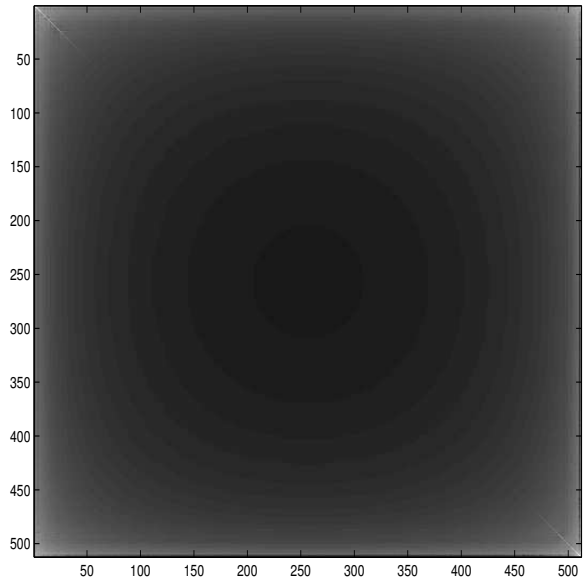


(b) Moonlight Sonata

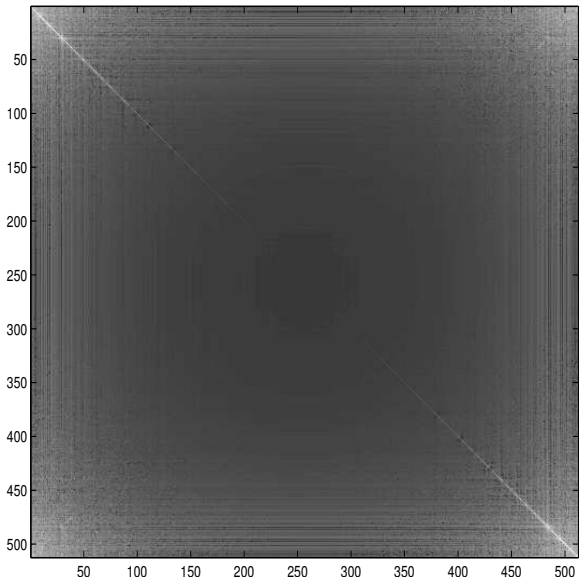




(a) Wannabe

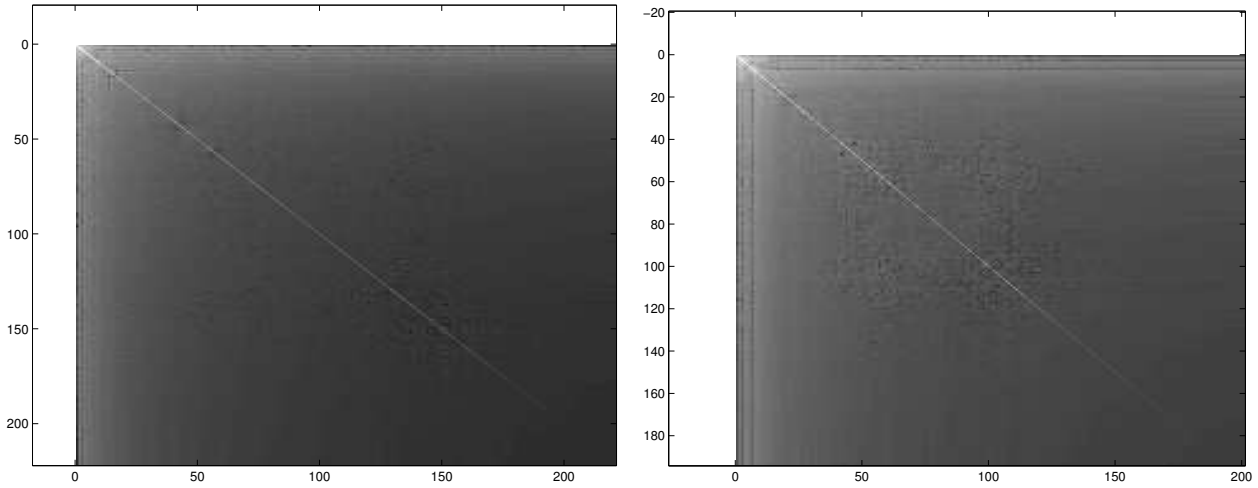


(b) Cello concerto



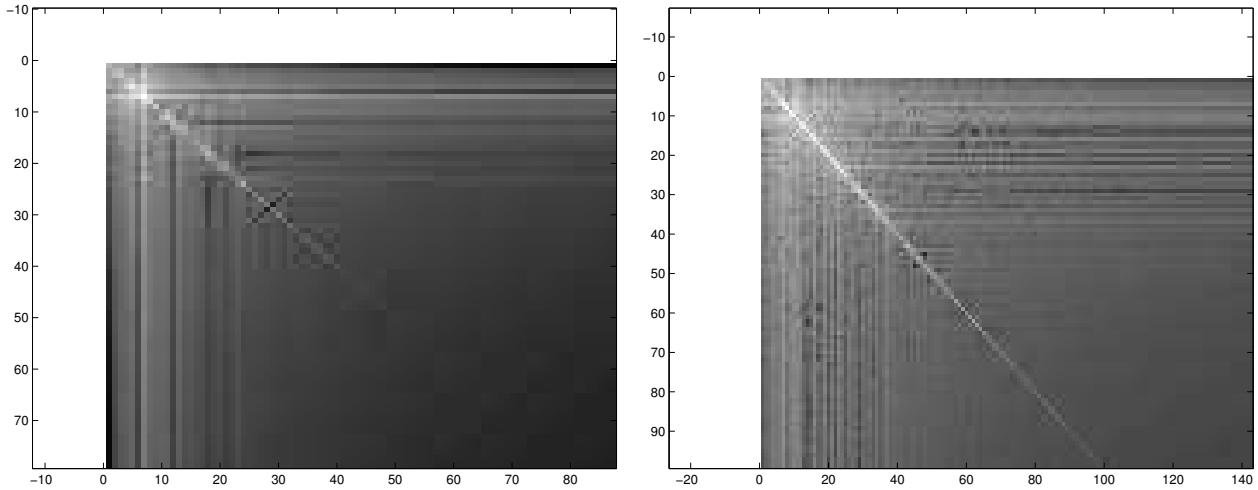
(c) Allegro Con Brio

Figure 5.18: Three covariance matrices



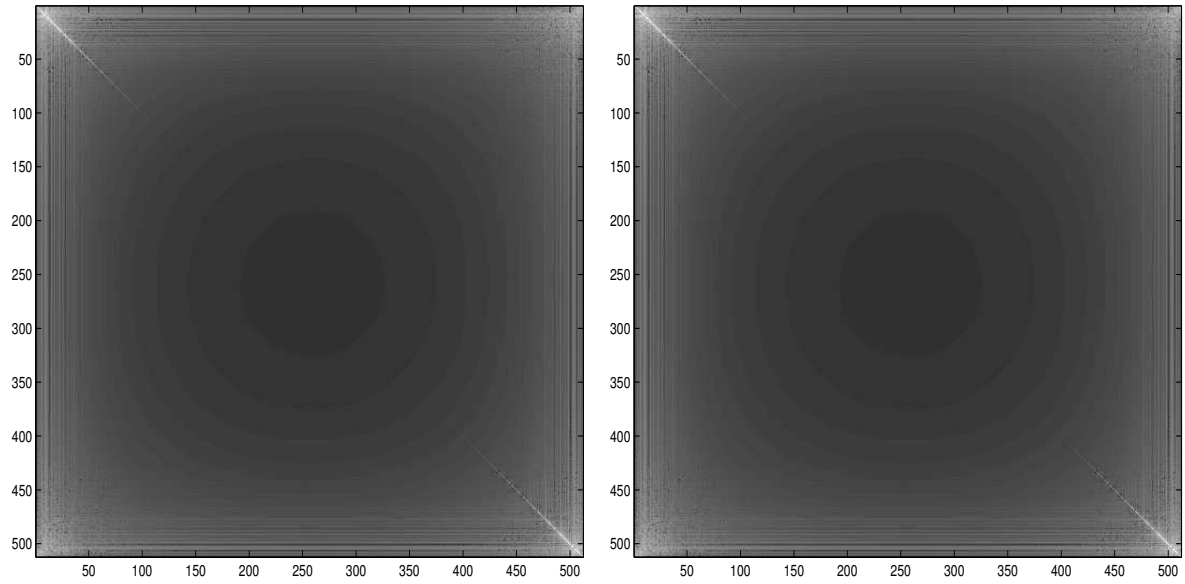
(a) “Kebabulous” (b) ‘Hometown Girl”

Figure 5.19: Close-up of covariance matrices for two songs by the same artist



(a) Moonlight Sonata (b) Bells

Figure 5.20: Close ups from two covariance matrices



(a) Bells 1

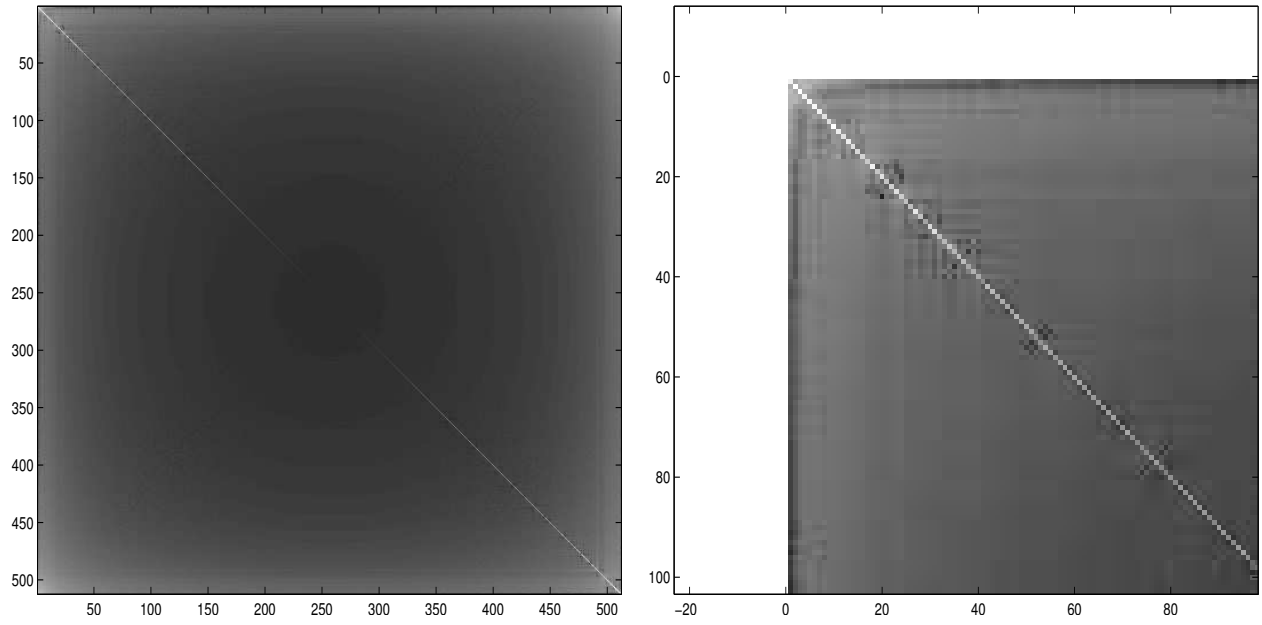
(b) Bells 11

Figure 5.21: Two different tunes on the Swan bells

pattern tending towards the centre. The stripes (figure 5.20(b)) pick out the notes and frequent harmonics associated with these bells. The two different tunes on these bells produce nearly identical covariance matrices.

It is clear from this that there are quite different covariance matrices associated with voice, piano, bells or other instrumental pieces. It seems that there are general patterns associated with voice, a single instrument, or orchestral pieces, and within each of those categories there are patterns associated with the particular voice or voices, or instruments. Finally, of course, every piece is slightly different. The covariance matrices mainly serve to pick out the frequency range or ranges typical to the piece, therefore identifying the instruments (or singers) used in the recording, but providing limited information beyond that already contained in the variances.

It is not therefore clear how much we will gain by considering a joint distribution over all the components rather than treating each Fourier component as independent. There may be some advantage: for example, some range of frequencies will correspond roughly to a “piano” piece, and so if we know that several of those frequencies are



(a) Covariance matrix

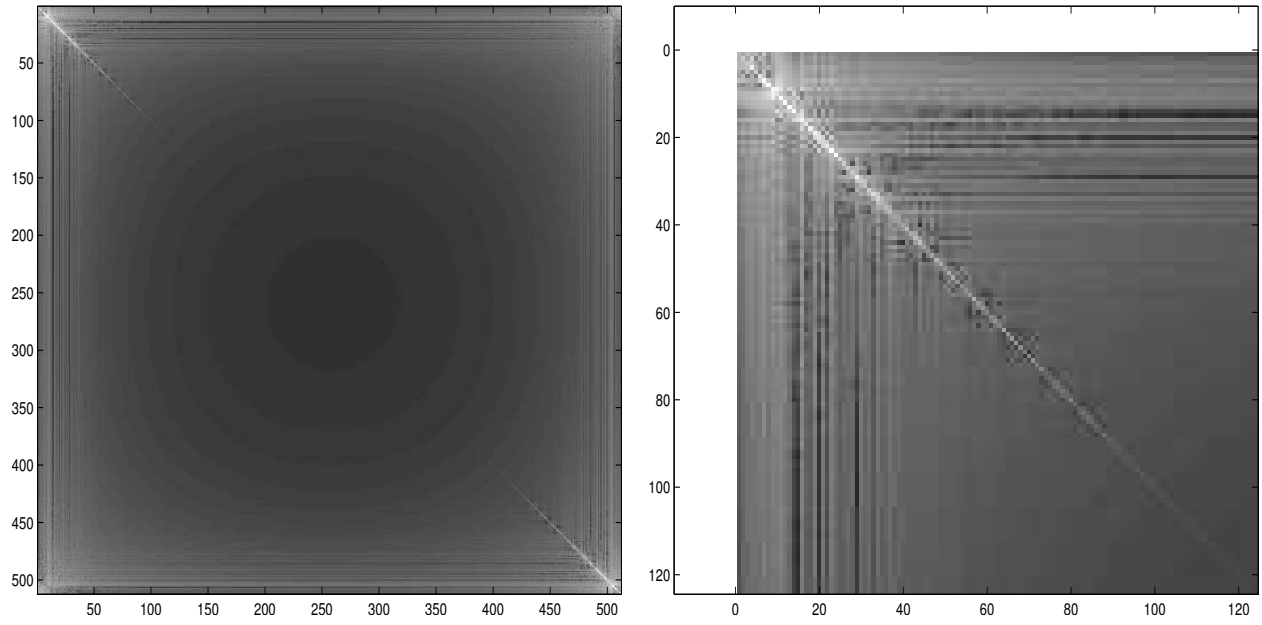
(b) Corner of covariance matrix

Figure 5.22: Averaged covariance matrix over all files

present we may suppose that the rest are likely to be.

5.5.9.1 Combined covariance matrices

Suggested by the above, we also computed some covariances as averages over collections of datasets (appendix B demonstrates the working for averaging covariance matrices). Figure 5.22 shows the covariance matrix taken over the whole dataset of 144 files from diverse sources, and a corner of this matrix. Compared with any individual matrix, there is some blockiness around the diagonal indicating some blurring of the frequencies there; more stripes indicating the wider range of frequencies that occur over the collection of files, and few blocky patches. Essentially this matrix represents the variance, with some blurring of frequencies around the diagonal. Figure 5.23 shows the covariance matrix over all the files from the Swan bells, and figure 5.24 shows the same information for all the songs by a single musician. In figure 5.24 some of the blocks characteristic to the artist can be seen, but much of it has been obscured by the



(a) Covariance matrix

(b) Corner of covariance matrix

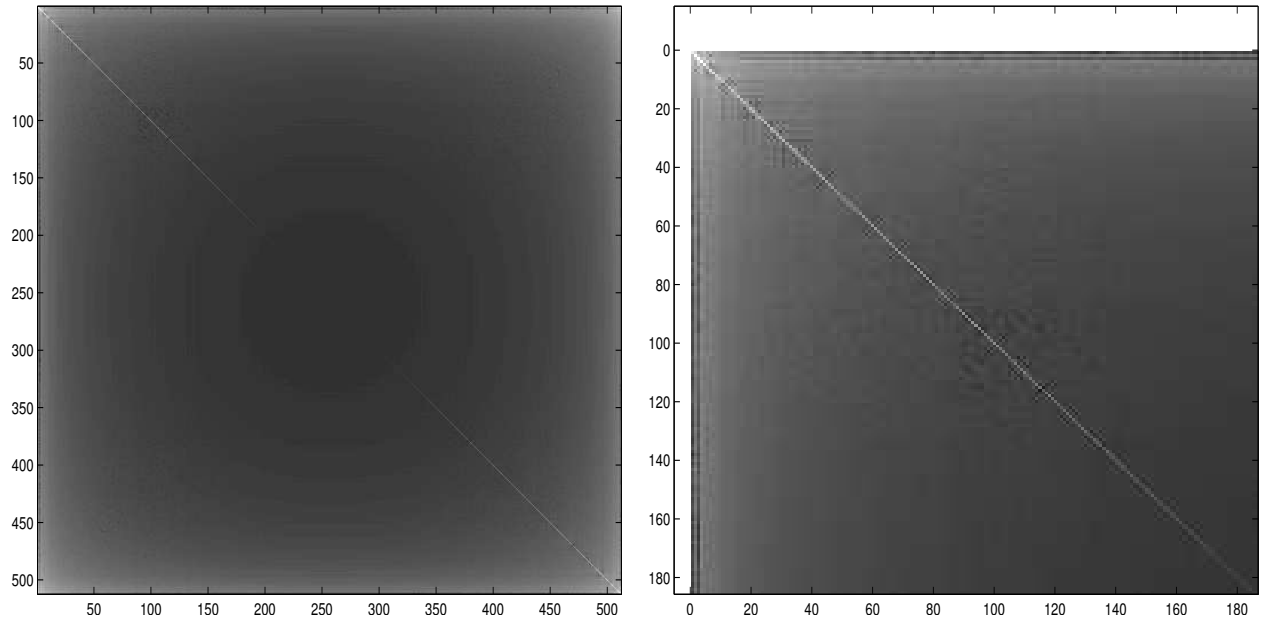
Figure 5.23: Averaged covariance matrix over bells files

averaging. For the bells, there is a similar distinctive pattern to that seen in the individual covariances although with more stripes; again there is very little blocking in this covariance matrix, merely picking out the common frequencies.

5.5.9.2 t-distribution

The means and covariances above form the maximum likelihood parameters for the Gaussian distribution. They capture certain properties of the data. There are equivalent maximum likelihood parameters for the multivariate t-distribution which capture similar properties of the data. These parameters, like those for the Gaussian distribution, are typically known as μ and Σ. They are estimated via iterative procedures and are dependent upon the value of ν.

As in the Gaussian, μ represents the mean of the data and in our audio context is zero. Figure 5.27 shows the magnitude of the computed μ for one of the files; it is similar in shape to the Gaussian mean.



(a) Covariance matrix

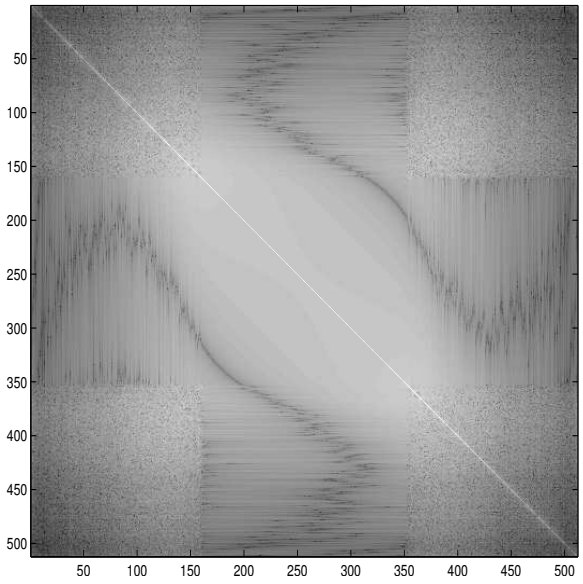
(b) Corner of covariance matrix

Figure 5.24: Averaged covariance matrix over Dave files

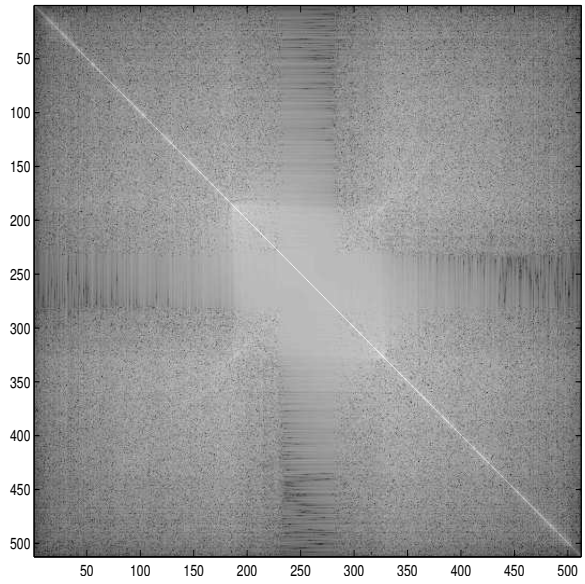
Figure 5.25 illustrates three of the Σ matrices found for a t-distribution, plotted in the same way as the covariance matrices above and figure 5.26 shows a close-up of one of these.

The Σ matrices have similarities to the Gaussian covariance, with blocky corners indicating the characteristic frequencies of the piece, but the frequency stripes are no longer obvious. The difference between the variance on the diagonal and the cross-variances is very obvious. the first component has a magnitude larger than 1. This may indicate a problem with the maximum likelihood estimation. Once again we see curving towards the centre for the more instrumental piece, 5.25(a), and a sharper cut-off for the voice piece, 5.25(c). "Solitude Standing" falls between the two.

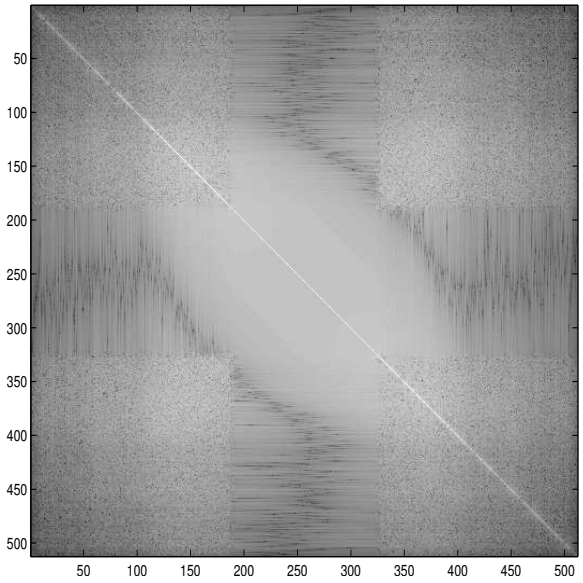
Optimising the parameters for a multivariate t-distribution with fixed degrees of freedom ist time consuming. The above parameters are taken for a t-distribution with the degrees of freedom fixed at 1. This is likely to be too low to obtain a good optimum, but should be sufficient to get some idea of the behaviour of the t-distribution. The likelihoods on a 512-dimensional data set with 1 degree of freedom were of the order



(a) Bells 1



(b) “Solitude Standing”



(c) “Pre-text”

Figure 5.25: Covariances for the t-distribution

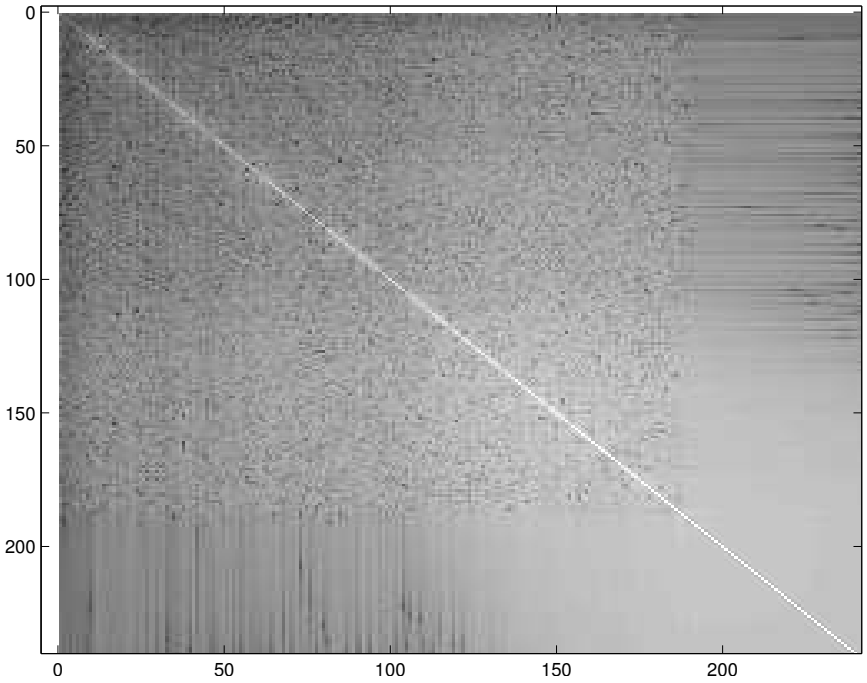


Figure 5.26: Close up of the t-distribution covariance parameter

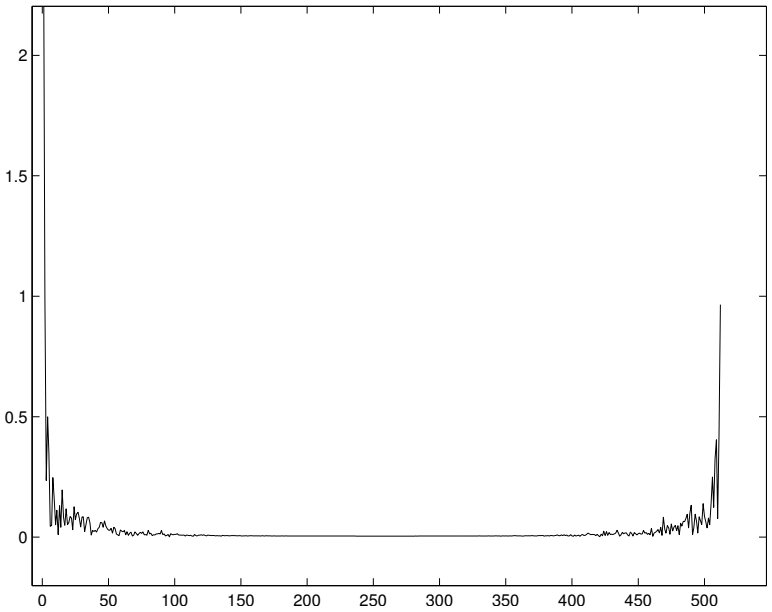
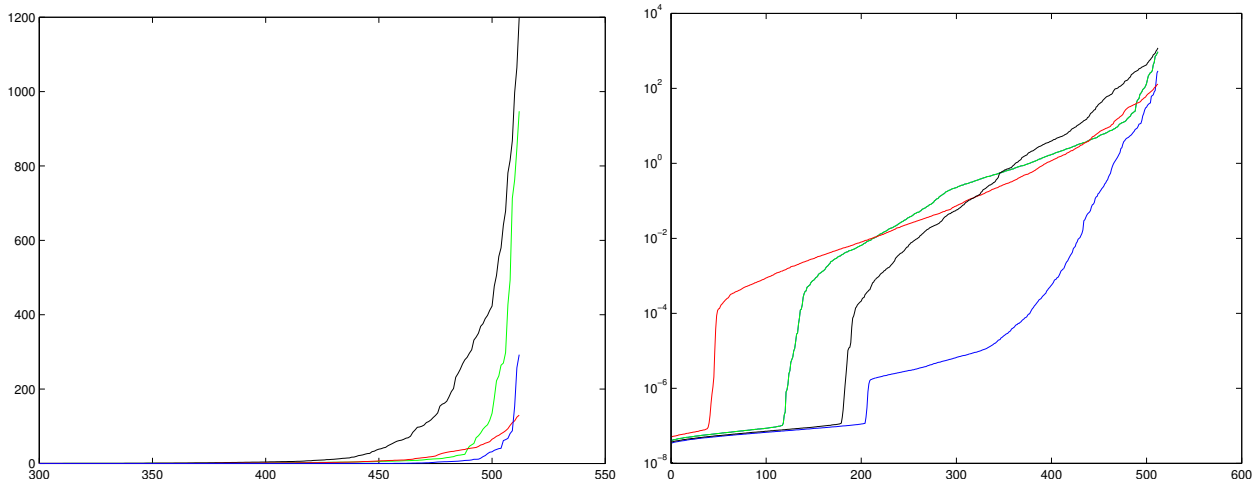


Figure 5.27: Value of μ-parameter for a multivariate t-distribution



(a) Eigenvalue plots:Blue: Moonlight sonata, Black: Bells, Red: Allegro con Brio, Green: Hometown Girl

(b) Log eigenvalue plots:Blue: Moonlight sonata, Black: Bells, Red: Allegro con Brio, Green: Hometown Girl

Figure 5.28: Eigenvalue plots

-1^{-5}, an order of magnitude lower than the one-dimensional likelihoods.

5.5.10 Principal Components

We looked at the eigenvalues of the Gaussian covariance matrices and their corresponding eigenvectors—these can be considered to be the "principal components" of the matrix and examining them may yield interesting results. We look at a piano piece, "Moonlight Sonata"; a song, "Hometown Girl"; and an orchestral piece, "Allegro Con Brio".

Figure 5.28 plots the eigenvalues on a linear scale(5.28(a)) and a log scale(5.28(b)). From the plots we can see that the magnitude of the eigenvalues varies between files, as does the magnitude of the covariances. However, for all the files there are only a few very high eigenvalues, and most of them are close to zero. Table 5.29 lists the top twenty eigenvalues for the three files.

When a large proportion of the eigenvalues are approximately zero, the covariance matrices are close to singular. This may cause problems in working with the inverted

matrices; care will have to be taken to avoid numerical difficulty.

Given a set of eigenvalues λ_i ordered by size of their associated eigenvectors x_i, we can reconstruct the original covariance matrix as

$$C = \Lambda^T X \Lambda$$

where $X = (x_1 \ldots x_n$ and Λ is the diagonal matrix with the λ_i on the diagonals.

We can use this formulation to reduce the dimensionality of C by using only $m < n$ of the eigenvectors. The relative error in reconstructing C is then

$$\frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

The remaining portion,

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

is called the fraction of the data "explained" by the first m eigenvectors. Table 5.30 shows the number of eigenvectors needed to explain various proportions of the covariances matrices for the example audio files. It is possible to explain "most" of the data, for some value of most, with a relatively small number of eigenvalues; in no case is it necessary to use as many as a quarter of the eigenvalues to explain 99% of the data.

Figure 5.31 shows the set of eigenvalues for one bells file's covariances plotted against the eigenvalues for another bells file's covariance. In figure 5.21 we saw that these two covariance matrices are very similar, so it is not surprising that their eigenvalues should scale the same way, although one set is larger than the other.

The eigenvectors associated with the m largest eigenvalues are known as the first m principal component. They represent the basis vectors for the lower dimensional representation of the data. Figure 5.32 compares the (real parts of the) first and second principal components for a number of the example files. Just the interesting parts of the vector are shown. They are all fairly similar and large in the same places as the power spectrum and variance are large.

5.5.11 Principal Components for t-distributions

The Σ matrix for a t-distribution has some similarities to the Gaussian covariance. However, it is clearly not the same thing. We examined briefly the eigenvalues and

Moonlight Sonata	Bells	Allegro con Brio	Hometown Girl
11.10	331.8	42.02	53.79
11.33	339.7	43.13	69.17
11.61	350.1	45.87	74.55
14.08	368.7	49.48	81.80
18.31	375.8	51.37	92.16
21.25	389.5	54.45	98.62
26.39	398.4	56.48	103.97
30.11	411.9	58.68	116.35
31.39	423.1	65.03	134.95
35.22	476.8	68.64	175.54
37.66	505.5	73.13	222.26
39.80	556.6	75.61	235.62
40.70	579.6	79.81	264.25
61.60	641.0	83.71	269.93
63.89	678.5	88.00	297.92
67.44	781.2	93.42	426.82
80.43	815.4	102.41	493.17
87.85	867.9	109.80	713.18
153.89	997.4	114.51	762.51
257.48	1064.7	124.54	847.15
292.57	1199.6	129.86	947.14

Figure 5.29: Eigenvalues

% explained	Moonlight Sonata	Bells	Allegro con Brio	Hometown Girl
50	3	12	14	4
66	6	20	24	8
75	9	26	30	10
90	18	44	52	22
95	26	57	72	43
99	41	96	127	115

Figure 5.30: Eigenvalues

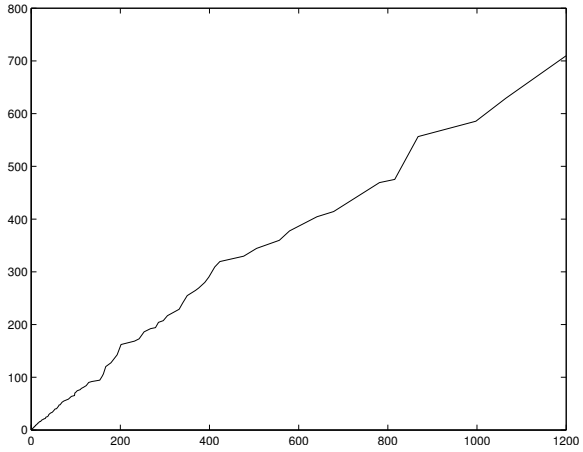
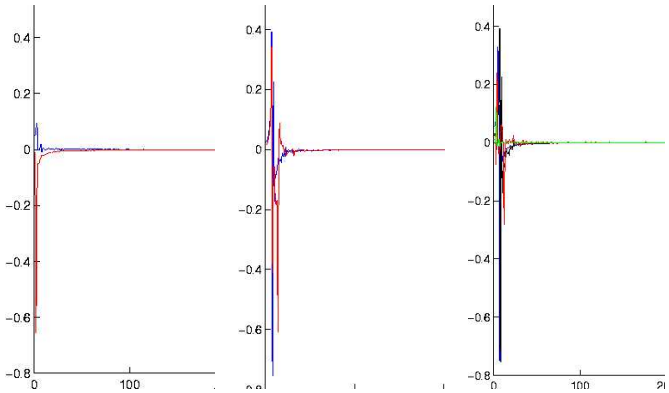


Figure 5.31: Eigenvalues for the two “bells” covariance matrices plotted against each other



(a) First principal component: “Hometown Girl” (blue), “Kebabulous” (red)

(b) Second principal component: “Bells-1” (blue), “Bells-2” (red)

(c) Second principal component: Blue: Moonlight sonata, Black: Bells, Red: Allegro con Brio, Green: Hometown Girl

Figure 5.32: First principal components

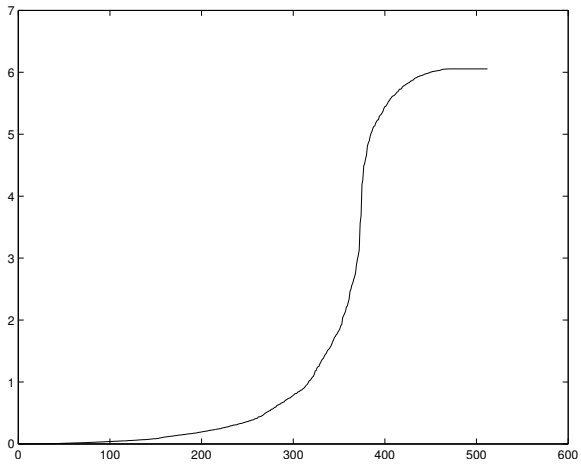
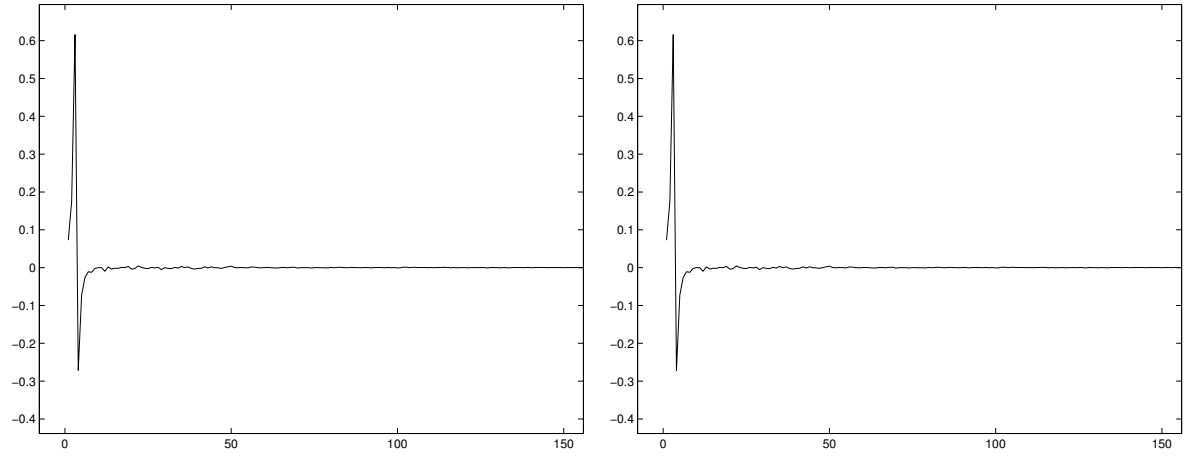


Figure 5.33: The eigenvalues for a t-distribution matrix



(a) First principal component of the t-distribution Σ matrix for "Solitude Standing"

(b) Second principal component of the t-distribution Σ matrix for "Solitude Standing"

Figure 5.34: Parts of the principal components for a t-distribution Σ matrix

eigenvectors for the "Solitude Standing" Σ matrix (figure 5.25(b)). Figure 5.33 plots the eigenvalues. As in the Gaussian case, the eigenvalues drop nearly to zero after the first 100. However, in this case there is a far smaller distinction between the largest eigenvalue (6.0547) and the smallest (0.0001). We also looked at the top principal components; these we found to have a similar shape to the Gaussian ones. Figure 5.34 shows parts of the top two components.

5.6 Summary

We have examined a selection of our music files from a range genres, considering a variety of instruments and voices. We have found that individual Fourier components have a characteristic distribution best described by a student's t-distribution with a mean of zero. We have observed that in general those Fourier components close to the maximum block size or the minimum block size are the ones which occur the most, or associated with the largest values. We have examined the interactions between Fourier components and concluded that the information stored in the distribution parameters for a Fourier prior of either t-distribution or Gaussian form contains primarily infor-

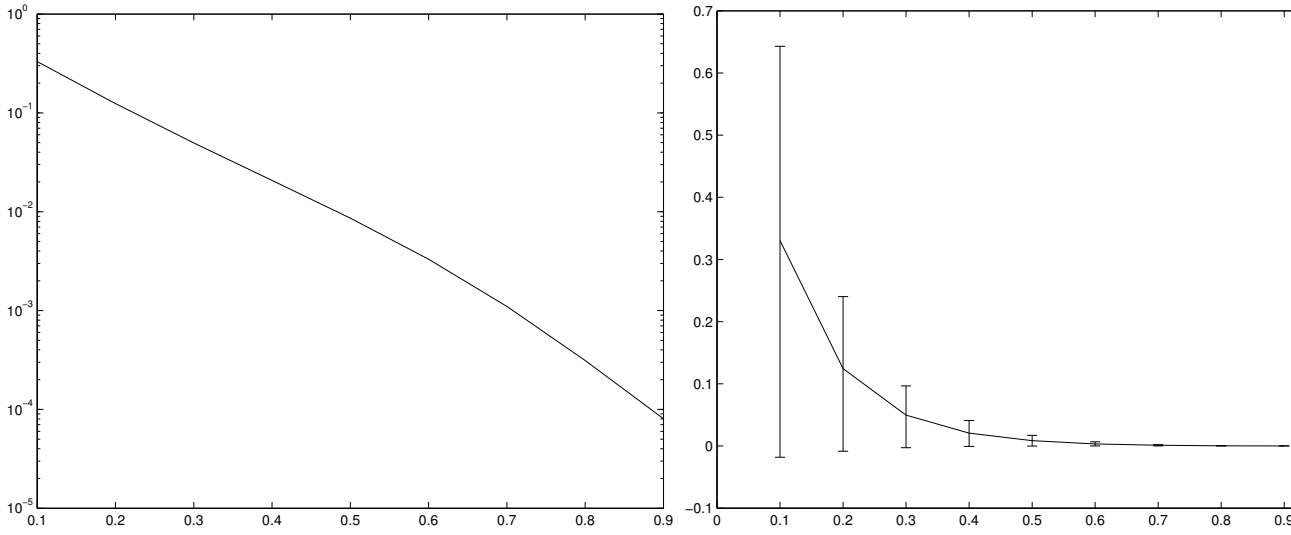


Figure 5.35: Rate of point removal at various clipping levels

mation about which frequencies are used in a piece. This differentiates pieces by the instruments or voices used in the recording.

5.7 Damaged data

When handling real clipped data, we will want to be able to estimate what the extent of the clipping is, in order to re-scale the data to leave room for the repaired larger amplitude data points. It would be possible to do this after the repair, but there are issues with doing so if the repair has been particularly optimistic. In particular, in practice, we often observed a small number of overoptimistic spikes. It may also be useful to be able to include bounds on the data points in carrying out the repair, although we do not examine doing so in this project. We therefore examined re-scaling before repairing the data.

Figure 5.35 shows the distribution of the fraction of points removed at a number of clipping levels on a log scale, and the distribution with error bars attached. It is clear that the removal of points is approximately proportional to the log of the number clipped, but also that it is difficult to be accurate about this without some prior information about the “loudness” of the file.

File	Mean dist. between points
Bells 1	0.022
Bells 2	0.013
Ride of the Valkyries	0.0038
Fur Elise	0.0021
Minuet in G	0.0077
I am cow	0.0043
Beautiful Morning	0.038
Streets of London	0.023
I vow to thee my country	0.00092
Wham! - Last Christmas	0.0079

Table 5.2: Continuity between data points

5.8 Evaluation metrics

There are two possible straightforward evaluation metrics on audio data of this form: the mean squared error of the data points, and the mean squared error of the frequency components. A third related measure is the mean squared error of the power spectrum. However, these measures do not necessarily capture the particular qualities of audio data. In this section we examine the data from the point of view of attempting to capture heuristic information which identifies a "clean" music file.

In order to obtain some evaluation metrics suitable for audio data, we considered a number of heuristics. By examining typical values and the variation in the values for the proposed heuristics for the known data we hoped to be able to determine heuristics which would be suitable for identifying "clean" data. We required two kinds of metrics: ones which could be used to compare a repaired file to its original "clean" file, and ones which could be used to describe the quality of a repaired file without access to a an original file.

5.8.1 Continuity

Audio data is suited to frequency-based approaches because of the typical wave-like shape of the signal. It is rare for two consecutive points to be a large distance apart. We therefore computed the mean distance between consecutive data points. Table 5.2 shows this metric for a selection of files. Note that in a clean file this value should not be too high; there should be a "good" degree of continuity. However, neither should it be too low; we do not expect to find blocks of data points which do not change at all from one to the next. In the table above, there is a difference of a factor of 40 between the smallest value (I vow to thee my country) and the the largest (Beautiful Morning). It is not immediately obvious what characteristics of the files might account for these differences. The amplitude of the file has some effect (I vow to thee my country is relatively quiet, and Beautiful Morning relatively loud), perhaps as a result of steeper curves, however it cannot account for all (or most) of the variation. Hence although if a continuity value is particularly large or small we can assume the file is poor quality, we must accept a broad range of values for an unknown file, meaning that a continuity-based metric would have limited usefulness.

5.8.2 Variance

As a consequence of the continuity constraint, small block of clean data will typically contain data points which are similar to one another. If there is typically large variation in the data points over a block of data, then the block is possibly erroneous. Table 5.3 shows the mean standard deviation on data points over blocks of 100 and 500 data points for a selection of files. They vary between Bells 1 (the largest value) and I vow to thee my country (the smallest value) by a factor of ten. Looking more closely, these differences can be approximately accounted for by considering the mean amplitude of the file. The final column of the table demonstrates this, showing considerably consistency over most of the files: "Last Christmas" is the exception, having a smaller σ than the mean amplitude would suggest. This suggests that dividing the mean variance of data points over a block by the mean amplitude of the data points may be a plausible heuristic.

File	σ: 100-blocks	σ: 500-blocks	amplitude	100-blocks / amplitude
Bells 1	0.0013	0.00027	0.10928	0.011
Bells 2	0.00067	0.00013	0.055862	0.012
Ride of the Valkyries	0.00019	4.1452e-05	0.016	0.011
Fur Elise	0.00031	6.7465e-05	0.02848	0.010
Minuet in G	0.00088	0.00023	0.09653	0.0091
I am cow	0.00029	6.26e-05	0.025	0.011
Beautiful Morning	0.0011	0.00027	0.11	0.010
Streets of London.mp3.wav	0.0012	0.00027	0.11	0.011
I vow to thee my country	0.00015	3.5e-05	0.014	0.010
Wham! - Last Christmas	0.00030	0.0001	0.042	0.0072

Table 5.3: Variances over blocks of data points

5.8.3 Penalties for grouped errors

A single point error in an audio signal at 44.1kHz is undetectable. One hundred single point errors distributed over a one-second clip are probably undetectable. However, one hundred single point errors gathered together in a block will produce briefly audible noise in the audio file. We investigated experimentally how various rates of per-block error sound. In order to do this, we took the data file, broke it into blocks of 500 points, and in every tenth block set a fraction of the data points, chosen randomly, to zero. We then ran a second set of tests setting the broken points to one. This provides a only a rough estimate of how errors might sound.

Table 5.4 shows the perceived effects on three different files, measuring quality on a scale of 1 (pure noise) to 10 (a perfect file). Estimating such effects qualitatively is difficult. Ideally we should ask a large sample of people to suggest a value, and take an average. In practice we were able to obtain a single second opinion. It was also noticeable that the errors sounded "different" between the different files: in the Bells file large proportions of zeros resulted in a throbbing effect, while in in Eine Kleine Nachtmusik the errors were audible as crackles. The errors set to one were audible as clicks in all cases. Given this table it seems that the error perception is roughly linear, the gain in constructing an evaluation metric which attempts to penalise grouped errors

% removed	filename	quality (removed=0)	quality (removed=1)
10	Bells	9	4
25		9	4
50		6	3
65		5	3
75		4	3
10	Boyzone – “Love me for a reason”	10	6
25		10	6
50		7	6
65		6	4
75		6	4
10	Eine Kleine Nachtmusik	10	6
25		9	6
50		8	5
65		7	5
75		7	4

Table 5.4: Perceived effects of rates of breakage

more heavily than individual error will be likely to be small.

However, one thing which is evident from this table is that guessing too high is far more audible than guessing too low. This will be because the mean amplitude of the data points is much nearer to zero than to one.

5.8.4 Frequency perception

At least two effects contribute to audible frequency perception:

1. Frequencies towards the middle of the human hearing range are perceived as louder than those towards the edges (Fletcher-Munson curves 2.1). The effect is non-linear and varies considerably from person to person.

2. Some frequencies may mask others. These effects are exploited in audio compression algorithms such as MPEG. In general, strong frequencies may mask

nearby weaker frequencies. However, the details of the masking depend on the particular frequencies and on the tonality of the masker[1].

For the purposes of this project it is unreasonable to consider complex measures such as frequency masking effects. It would however, be possible to apply a cost function to the frequency error computation, but this may not be worth doing. The perceptual qualities of the music may contribute to the frequency effects in the sound and thus incorporated implicitly in the model.

5.8.5 Other cues

Roweis [24] mentions heuristics based on common fate cues (frequencies which tend to have similar points of onset and off set), pitch, and associating harmonics. This is just the kind of information which we hope to be encapsulated by the covariance matrix and hence contained within a good model.

5.8.6 Summary

We have examined a small number of properties of the data which we hoped might lead to perceptually motivated evaluation metrics. Unfortunately, we have not identified any metrics which are obviously suitable for this purpose.

5.8.7 Comments

One of the primary aims of this project was to experiment with a variety of models and extend the use of the noisy FFT techniques beyond the independent Gaussian priors previously tested [27]. From the point of view of testing the effectiveness of this technique on real data, the perceptual qualities of the data are not relevant; unless the models as well as the evaluation metrics incorporate these qualities (explicitly or implicitly) we cannot expect improved models to perform better perceptually.

The use of these audio-specific evaluation metrics may therefore help to determine if the models are not supplying sufficient problem-specific information, but will not say

[1] source:`http://en.wikipedia.org/wiki/Psychoacoustics`

anything about the effectiveness of the noisy FFT technique itself, or its applicability to other types of problem.

Chapter 6

Models

6.1 Choosing models

Following the exploratory analysis we felt that the following points would be worth investigating:

- A Gaussian distribution as an approximation to the distribution of Fourier components. The Gaussian appears to be less good a fit than the t-distribution but the handling of Gaussian distributions is well-known and straightforward; a Gaussian distribution will therefore form a good baseline.
- A t-distribution appears to be a good fit to the distribution of Fourier components.
- Since there appears to be more variation over a whole file than over part of a file, the incorporation of some form of time-dependency into the model may be appropriate. This could take the form of updating the covariance matrix to more closely approximate the covariance matrix of the original file as we move along the file, or of using a distribution such as a time-dependent Gaussian distribution.
- Files appear similar in their Gaussian parameters if they contain similar instruments and combinations of instruments. Choosing priors based on some known information about the type of sound in the file may therefore be worthwhile.

- There is some interaction between frequencies which can be considered characteristic to a file. However, this is limited. Use of a full covariance matrix rather than an independent model may therefore not add enormously to the model.

This leads to the following set of models:

- Independent Gaussian (each component is considered separately)
- Multivariate Gaussian
- Time-dependent Gaussian (multivariate)
- Independent t-distribution (each component is considered separately)
- Multivariate t-distribution

There is no known (closed) form for a time-dependent t-distribution.

6.1.1 Modelling decisions

ν-values We use the same value of ν for every component within the independent t-distribution model. It is not clear from the exploratory analysis whether this is optimal. Initial experimentation showing the histograms having the same shape suggested that it should be. However, when fitting t-distribution curves we found that different values of ν seemed to be good for different components. This may just be because none of the choices were optimal.

Handling clipped data We chose to treat the clipped data as a missing data problem. All data points which were found by the system to be at the cut-off point for clipping were set to zero, and then handled in precisely the same way as missing data. There were two reasons for not incorporating a more complicated model even to the extent of forcing the data to be at least as large as the clipped value. One was the simplicity of handling the data uniformly in this way; the other was in order to investigate the extent to which the model succeeded in inferring the correct values for the data without the additional cues.

Handling real clipped data Synthesizing clipping problems is straightforward; one chooses a cut-off point and fixes all the data points originally larger than the cut-off at that point. The space between the cut-off point and 1 is then available for repaired data points. However, when real data has been clipped, it is necessary to rescale it either before or after repairing the file. If the rescaling is done before repairing the file, then it is necessary to estimate what the scaling factor should be, knowing only the distribution of the remaining data. In §5.7 we will discuss some of the issues in this estimation. Our system performed the rescaling before the repair, in order to base the scaling on the values in the correct data, and not on possibly inaccurate repaired data. The rescaling was done by a simple rule on the logarithm of the number of clipped points, as suggested by the results in §5.7.

(There would be potential for more experiment here: the algorithm performance may depend on the scale of the data, although this seems unlikely. More evident is the fact that the scale of the data does affect the noise perception, as discussed in §5.8.3.)

Multivariate time-dependent distribution We chose to move immediately from the multivariate Gaussian distribution to the multivariate time dependent distribution without considering a univariate time-dependent distribution. This stemmed from an initial assumption that much of the information propagating from one block to the next would be between Fourier components rather than on a particular component. Further reflection and our results suggest that this assumption may be invalid; we leave a univariate time-dependent Gaussian distribution as a suggestion for future work.

6.1.2 Parameters

Alongside this set of models I investigated the effects of changing the size of the Fourier blocks, of the initial parameters and of modifying the parameters during the restoration process. The next section develops the models mathematically.

6.2 Developing the models

6.2.1 General form of the model

Model Given a block of n data points containing m known points, $\mathbf{m} < observed_j >$, we aim to find a vector **unk** of $n-m$ points such that $P(\mathbf{block} = (\mathbf{observed}, \mathbf{unk}))$ is maximised; that is $P(\mathbf{block} = (\mathbf{observed}, \mathbf{unk}) | \mathbf{observed})$ is maximised.

From Bayes' rule,

$$P(\mathbf{block}|\mathbf{observed}) \propto P(\mathbf{observed}|\mathbf{block})P(\mathbf{block})$$

By monotonicity of log, this is maximised when

$$\log(P(\mathbf{observed}|\mathbf{block})P(\mathbf{block})) = \log P(\mathbf{observed}|\mathbf{block}) + \log P(\mathbf{block})$$

is maximised.

Since $\mathbf{block} = (\mathbf{observed}, \mathbf{unk})$, $P(\mathbf{observed}|\mathbf{block}) = 1$, and $\log P(\mathbf{observed}|\mathbf{block}) = 0$.

The prior probabilities we have are not for $P(\mathbf{block})$, but for $P(\mathbf{f}(\mathbf{block}))$, where $f(...)$ is the Fourier transform.

Applying rules for transformation of variables,

$$P(\mathbf{block}) = P(\mathbf{f}(\mathbf{block}))\,|J(f(block), block)|$$

where

$$J(X,Y) = \frac{\partial(x_1,\ldots,x_n)}{\partial(y_1,\ldots,y_n)} = \begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \cdots & \frac{\partial x_n}{\partial y_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial y_n} & \cdots & \frac{\partial x_n}{\partial y_n} \end{pmatrix}$$

is the Jacobian.

So

$$\log P(\mathbf{block}) = \log(P(\mathbf{f}(\mathbf{block}))\,|J(f(block), block))|$$

$$= \log P(\mathbf{f}(\mathbf{block})) + \log|(J(f(block), block))|$$

We therefore need to compute $J(f(block), block)$.

Letting $F = (f_1, \ldots, f_n)$ be the Fourier transform of $X = (x_1, \ldots, x_n)$,

$$\frac{\partial f_j}{\partial x_k} = \frac{\partial}{\partial x_k} \frac{1}{N} \sum_{n=0}^{N-1} x_i e^{-2\pi i jn/N}$$

$$= \frac{1}{N} e^{-2\pi ikn/N}$$

which is a constant, so for the purposes of maximisation,

$$\log P(\mathbf{block}) \propto \log P(f(\mathbf{block}))$$

Gradients In order to use gradient optimisation techniques, we must be able to compute the gradient of the probability function;

$$\frac{\partial}{\partial x_i} \log(P(\mathbf{F}))$$

where $\mathbf{F} = (f_1, \ldots, f_n) = f(x_1, \ldots, x_n)$.

Applying the chain rule,

$$\frac{\partial}{\partial x_i} \log P(\mathbf{F}) = \frac{\partial}{\partial F} \log P(\mathbf{F}) \frac{\partial F}{\partial x_i}$$

$$= \sum_{j=1}^{N} \frac{\partial}{\partial f_j} \log P(\mathbf{F}) \frac{\partial f_j}{\partial x_i}$$

$$= \sum_{j=1}^{N} \frac{\partial}{\partial f_j} \log P(\mathbf{F}) \frac{\partial f_j}{\partial x_i}$$

$\frac{\partial}{\partial f_j} \log P(\mathbf{F})$ depends on the probability distribution. Writing $lp_j = \frac{\partial}{\partial f_j} \log P(\mathbf{F})$,

$$\frac{\partial}{\partial x_i} \log P(\mathbf{F}) = \sum_{j=1}^{N} lp_j \frac{\partial f_j}{\partial x_i}$$

Where the $\frac{\partial f_j}{\partial x_i}$ are as computed above, giving

$$\frac{\partial}{\partial x_i} \log P(\mathbf{F}) = \frac{1}{N} \sum_{j=1}^{N} lp_j e^{-2\pi ijk/N}$$

$$= fft(\mathbf{LP})$$

Equivalently, we could replace the Fourier transform with the inverse Fourier transform throughout.

6.2.1.1 Gaussian Components

Objective function

$$P(\mathbf{x}) = \frac{1}{(2\pi)^d/2\sqrt{|\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)}$$

where d is the dimensionality of the data.

$$\log P(\mathbf{x}) = \log(\frac{1}{(2\pi)^d/2\sqrt{|\Sigma|}}) + -\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)$$

$$\propto -\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)$$

If the components are assumed independent, then Σ is a diagonal matrix, with the variances of the components on the diagonals. Letting $\mathbf{v} = (v_1, \ldots, v_n) = (\Sigma_{11}, \ldots, \Sigma_{nn})$, $\mathbf{v}^{-1} = (\frac{1}{v_1}, \ldots, \frac{1}{v_n})$ and $\mathbf{x} * \Sigma^{-1} = \mathbf{x}.\mathbf{v}^{-1}$.

Gradient function

$$\frac{\partial}{\partial \mathbf{x}} \log P(\mathbf{x}) \propto \frac{\partial}{\partial \mathbf{x}}(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu))$$

$$\propto \frac{\partial}{\partial \mathbf{x}}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)$$

$$= \frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}\mathbf{x} + \mu^T\Sigma^{-1}\mu$$

$$= 2(\Sigma^{-1}\mathbf{x} - \Sigma^{-1}\mu)$$

$$\propto \Sigma^{-1}(\mathbf{x}-\mu)$$

This can be similarly simplified if the components are independent, so Σ is diagonal.

6.2.1.2 Time-dependent Gaussian Components

We can improve the accuracy of the model by including not just the observations in the current block, but also the estimate for the previous block of data. That is, we compute

$$P(\mathbf{block_t}|\mathbf{observed_t}, \mathbf{block_{t-1}})$$

$$\propto P(\mathbf{observed_t}|\mathbf{block_t},\mathbf{block_{t-1}})P(\mathbf{block_t}|\mathbf{block_{t-1}})$$

$$= P(\mathbf{block_t}|\mathbf{block_{t-1}})$$

$$= \frac{P(\mathbf{block_t},\mathbf{block_{t-1}})}{P(\mathbf{block_{t-1}})}$$

If $P(\mathbf{block_{t+1}},\mathbf{block_t})$ is Gaussian with mean $(\mu_{t+1}\mu_t)$ and covariance matrix

$$\begin{pmatrix} Q^{tt} & Q^{t+1,t} \\ Q^{t,t+1} & Q^{t+1,t+1} \end{pmatrix}$$

then (by Gaussian partitioning theory) $P(\mathbf{block_{t+1}}|\mathbf{block_t})$ is Gaussian with mean

$$\mu_{t+1} + Q^{t+1,t}(Q^{t+1,t+1})^{-1}(\mathbf{block_t} - \mu_t) \tag{6.1}$$

and covariance

$$Q^{t+1,t+1} - Q^{t+1,t}(Q^{t+1,t+1})^{-1}Q^{t,t+1} \tag{6.2}$$

Note that if the covariance matrix is diagonal, these updates reduce to:

$$\mu_{t+1|t} = \mu_{t+1}$$

and covariance

$$Q_{t+1|t} = Q^{t+1,t+1}$$

This is unsurprising; if the covariance matrix is diagonal, then the datapoints give no information about other datapoints, and so knowing one set of datapoints cannot help infer anything about later points.

However, we should be clear about what the covariance matrix

$$\begin{pmatrix} Q^{tt} & Q^{t+1,t} \\ Q^{t,t+1} & Q^{t+1,t+1} \end{pmatrix}$$

contains in this application.

$P(\mathbf{block_{t+1}},\mathbf{block_t}))$ is proportional not to $P(F = fft(\mathbf{block_{t+1}},\mathbf{block_t})$, but to $P(F1 = fft(\mathbf{block_{t+1}}), F2 = fft(\mathbf{block_t}))$. The top-left and bottom-right corners of

the covariance matrix therefore contain (roughly speaking) the covariances of individual Fourier transforms, and the bottom-left and top-right corners contain the interactions between two transforms. It will therefore not necessarily be the case that considering the Fourier components independently (so that the covariances of individual transforms are diagonal) results in the time-dependent covariance matrix being diagonal.

Objective function, gradient function These functions are Gaussian, as above. However, the mean and covariance will be updated at each time step, using equations 6.1 and 6.2.

6.2.1.3 Student t-distribution

Objective function For the univariate t-distribution,

$$P(x|\mu,S,\nu) = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)(S\nu\pi)^{1/2}}\left(1+\frac{\Delta^2}{\nu}\right)^{-(\nu+1)/2}$$

where

$$\Delta^2 = \frac{(x-\mu)^*(x-\mu)^2}{S}$$

is the squared Mahalanobis distance from x to μ (* denotes the complex conjugate).

It is usual for convenience and necessary in order to avoid numerical issues (very small probabilities which appear as 0) to use the log probability,

$$L(x;\mu,S,\nu) = \log Z + P\log\left(1+\frac{\Delta^2}{\nu}\right)$$

where

$$Z = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)(S\nu\pi)^{1/2}}$$
$$P = -(\nu+1)/2$$

Gradient function

$$\frac{d}{dx}L(x;\mu,S,\nu) = \frac{P}{\left(1+\frac{\Delta^2}{\nu}\right)}\frac{2(x-\mu)}{S\nu}$$

6.2.1.4 Multivariate t-distribution

Objective function The t-distribution can be generalised to d dimensions, where it takes the form:

$$P(\mathbf{x}|\mu,\Sigma,\nu) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\nu/2)\,|\Sigma|^{1/2}\,(\nu\pi)^{d/2}}\left(1+\frac{\Delta^2}{\nu}\right)^{-(\nu+d)/2}$$

where

$$\Delta^2 = \frac{(x-\mu)^*(x-\mu)^2}{S}$$

and $\Gamma(.)$ is the Gamma function.

The log probability is given by

$$L(\mathbf{x};\mu,\Sigma,\nu) = \log Z + P\log\left(1+\frac{\Delta^2}{\nu}\right)$$

where

$$Z = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\nu/2)\,|\Sigma|^{1/2}\,(\nu\pi)^{d/2}}$$
$$P = -(\nu+d)/2$$

Gradient function The gradient function becomes

$$\frac{d}{dx}L(\mathbf{x};\mu,\Sigma,\nu) = P\left(1+\frac{\Delta^2}{\nu}\right)^{-1}\frac{2\Sigma^{-1}(\mathbf{x}-\mu)}{\nu}$$

Chapter 7

Problem Set

7.1 Data Files

To form a problem set, we selected a variety of files with different audio characteristics. We chose two files from the "Bells" dataset; one with sixteen bells (fast ringing) and one with just six (much slower). From the exploratory analysis we have seen that there is little variety in the frequency domain among files from this dataset. From the "Songs" dataset we selected two song sung by male voices ("Hometown Girl", "Moon River"), songs sung by a range of female voices ("This was Pompeii", "Marlene on the Wall", "Caged"), and a song sung by a group ('Wannabe"). These songs were also selected to have different backing music/instrumentals. From the "classical" dataset, we chose four pieces: "Fur Elise", on the piano, "Morning Mood", an instrumental piece with flute solos, the "Dance of the Sugar Plum Fairy", another instrumental piece, including a glockenspiel, and "the Ride of the Valkyries", an operatic female voice backed by full orchestra.

For each file, we took three ten-second clips; one from close to the beginning of the file (beginning ten seconds in), one from close to the end (beginning twenty seconds from the end), and one from the middle of the file. We carried out experiments on the three clips to obtain a reasonable representation of the results on each file without having to perform analysis on what summed to hours of audio data.

The intention was not to try every model on every file in the problem set, but to

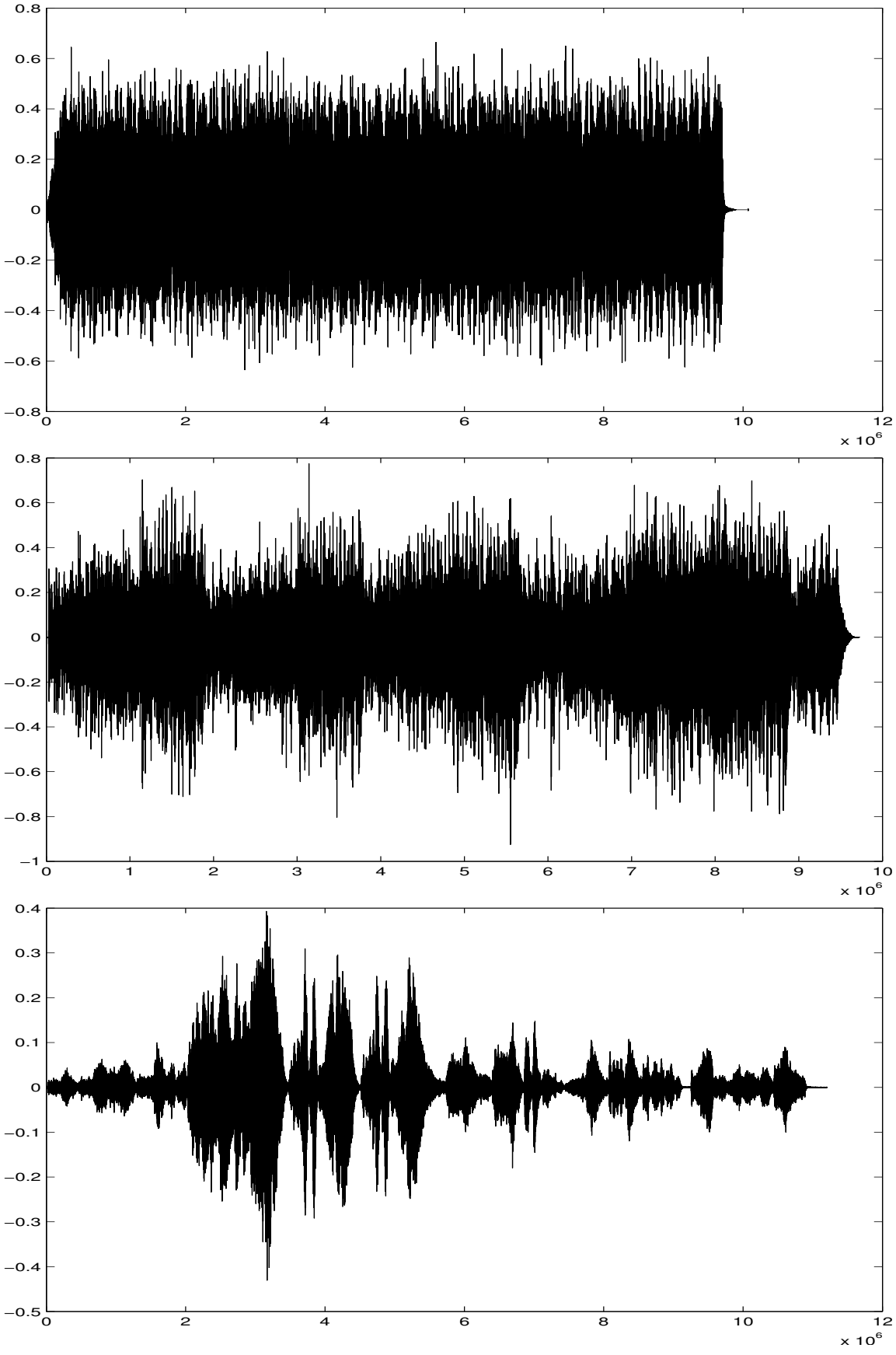


Figure 7.1: Signals for a selection of the problem set files.

Top: 6 bells; Middle: Marlene on the Wall; Bottom: Morning Mood

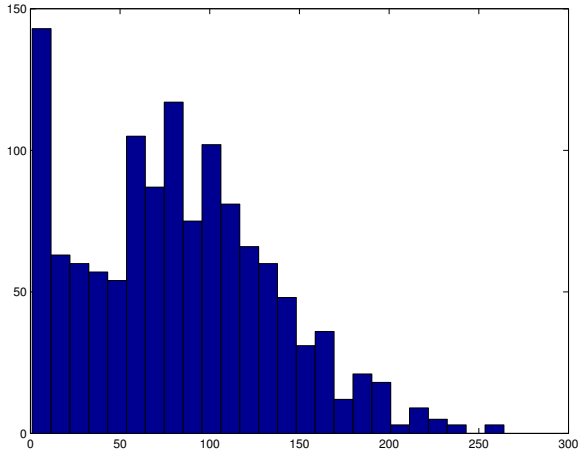
have a variety of files along with a variety of possible breakages, and to select examples for examining each model.

7.2 Missing data

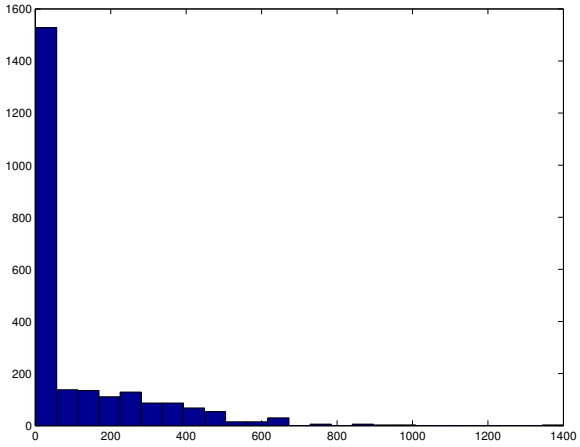
We experimented with four different average block sizes to remove. Using a Fourier block size of 512, it will mostly be interesting to consider missing blocks no larger than the Fourier block size, although as we propagate time-dependent information through the file it will be possible to consider large chunks of missing data. For the non-time-dependent distributions we compared average block size removals of 128 and 384. For the time-dependent distributions we also considered removing blocks with an average size of 768 points.

We used a normal distribution centred around the mean (128, 384, 768) to determine block size, and removed blocks from random points through the file. A ten-second clip at 44.1kHz contains 441,000 data points; 861512 blocks. 500 blocks of data were removed from each of the three clips.

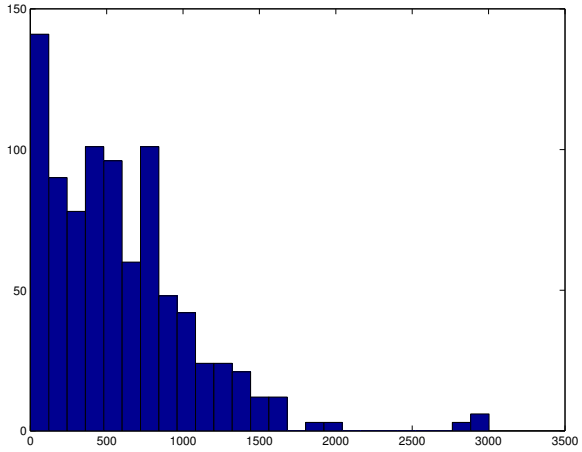
The files were stored with zeroes representing the missing data. Any zeroes in the files to start with were therefore treated as missing data for the reparation. In each case there were some isolated points at zero: in Morning Mood; 1742, in Caged; 88 and in Wannabe; 92. Of the 1.36 million data points, this is a tiny fraction. However, this explains the apparent non-Gaussianity in the plots. Table 7.1 provides some more statistics about the missing data distribution. These values are taken over all three ten-second clips. The counts are taken ignoring lone zeroes. There is evidence of some overlapping in the blocks which were removed: in no case does the total reach 1500. As the block size increases, so does the chance of overlap. Despite this, the mean block size removed is less than the intended mean, perhaps because of the prior presence in the data of some very small blocks of zeroes.



(a) Missing from Caged in 128 blocks



(b) Missing from Morning Mood in 384 blocks



(c) Missing from Wannabe in 768 blocks

Figure 7.2: Distributions of missing data

Piece	Breakage	Mean size removed	Contiguous blocks remov
Morning Mood	128	89.5384	1146
Caged	128	89.1705	1161
Sugar Plum Fairies	128	85.9438	1138
Suzanne Vega - Marlene On The Wall	128	89.6245	1153
Bells	128	82.6617	1150
Morning Mood	256	177.8526	1099
Caged	256	178.7635	1095
Sugar Plum Fairies	256	185.4366	1065
Suzanne Vega - Marlene On The Wall	256	195.265	1053
Bells	256	174.7635	1095
Morning Mood	384	259.653	977
Caged	384	292.7038	996
Sugar Plum Fairies	384	294.9652	978
Suzanne Vega - Marlene On The Wall	384	281.3457	1047
Bells	384	288.2688	1008
Morning Mood	768	623.9564	780
Caged	768	644.4703	774
Sugar Plum Fairies	768	648.285	786
Suzanne Vega - Marlene On The Wall	768	594.8589	801
Bells	768	653.599	813

Table 7.1: Distribution of missing data

7.3 Clipped data

The amplitude for data points varies between -1 and 1. To simulate clipping, we removed all data points whose absolute value was larger than some cut-off point. Four cut-off points were: 0.1, 0.3, 0.6 and 0.9. Most of the files were affected barely or not at all by the higher cut-off points (table 7.2) — only in "Wannabe" did as many as as 1 in a hundred datapoints have an amplitude above 0.6. However, even the small fractions of clipping were audible: blocks of points with a high proportion clipped cause a brief hiss; in the files clipped at higher cut-off points the hisses were rarer, occurring perhaps twice in a clip for "This was Pompeii" at a cut-off of 0.6, for example. However, the hisses themselves remained audible. For the cut-off of 0.9, the breakage was virtually undetectable in all but "Wannabe"—in some pieces it is possible to identify short problematic points by ear, but they do not disrupt the audio signal. Figure 7.3 shows histograms of the sizes of continuous blocks of clipped data. It is clear that for the most part data was not clipped continuously in large quantities, although many data points from a block of true data may have been removed.

7.4 Real data

7.4.0.4.1 Clipped data One way in which clipped data may arise in a real-world situation is when sound is recorded onto a computer with the volume set poorly. We created real examples of clipping in this way, recording audio signals from tapes onto my machine. These signals were recorded at 8kHz, which limits the maximum possible quality. Two of the clips were from songs of different styles: "Westside Story" and a children's tape. The third was the sound of bells. Table 7.3 shows the fraction of points which were at the maximum amplitude and therefore probably clipped, for each of the files.

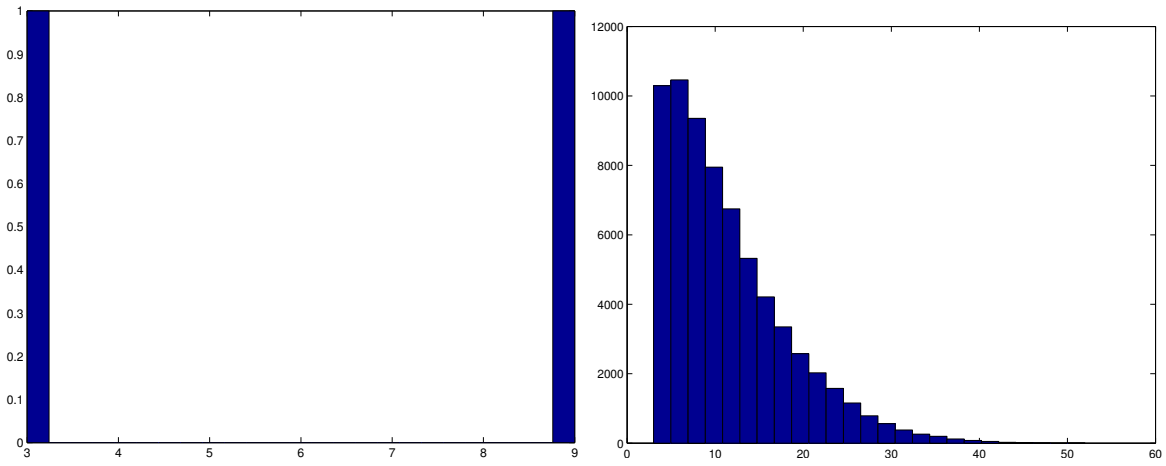
The distribution of the clipped points was different in each of the files. The bells were clipped consistently throughout (figure 7.4(a)), while the children's song had several "peaks", with the audio quality reasonable in between (figure 7.4(b)).

File	0.1	0.3	0.6	0.9
Caged	0.54	0.085	0.0015	7.18e-05
Dar Williams - 05 - This Was Pompeii	0.26	0.019	0.00076	0.00055
Suzanne Vega - Marlene On The Wall	0.26	0.0081716	0.00016175	0.00014
Hometown Girl	0.34	0.0089	0.00015	0.00011
Frank Sinatra - Moon River	0.21	0.0049	0.00043	0.00043
Spice Girls - Wannabe	0.605	0.18237	0.016	0.00076
Ride of the Valkyries	0.177	0.010515	0.00030	0.00022
Morning Mood	0.0013	0.0013	0.0013	0.0013
Sugar Plum Fairies	0.32	0.013	0.00016	0.00012
Fur Elise	0.038	0.00036	0.00036	0.00036
Bells–16 bells	0.584	0.11	0.0044	0.00013
Bells–6 bells 08	0.34	0.0090	0.00012	0.00012

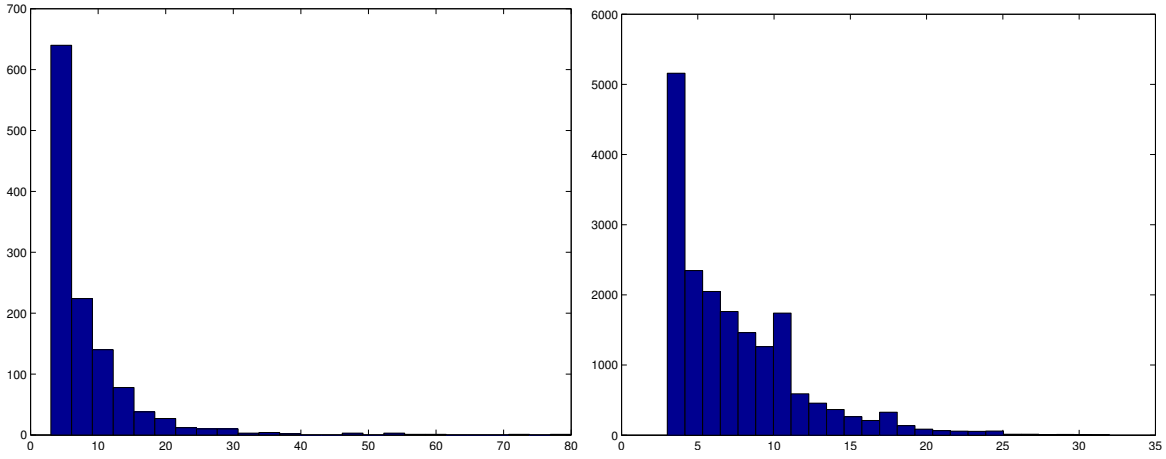
Table 7.2: Fraction of points removed from the problem set files (mean over the three clips)

File	Fraction clipped
Bells	0.29
Westside story	0.24
Children's song	0.22

Table 7.3: Clipping levels on real data

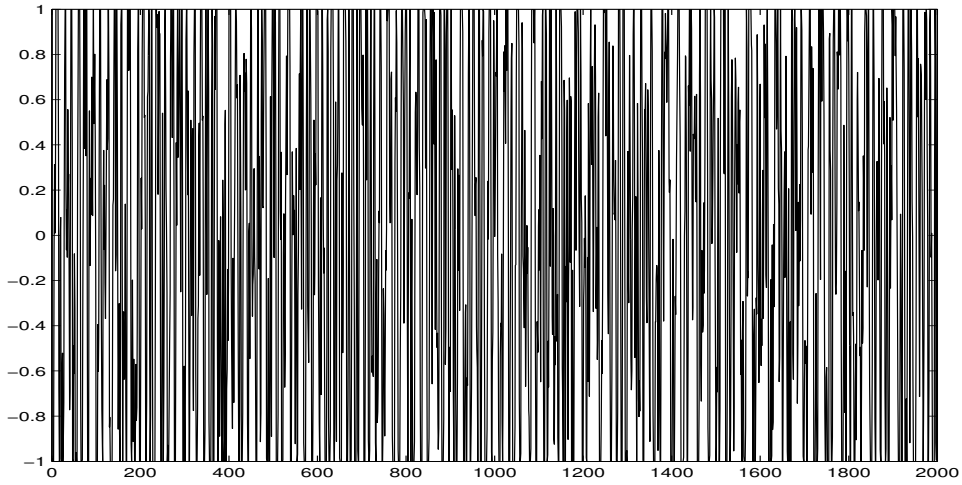


(a) Sizes of clipped blocks for files clipped at 0.1: Left, “Morning Mood”; Right, Bells 1

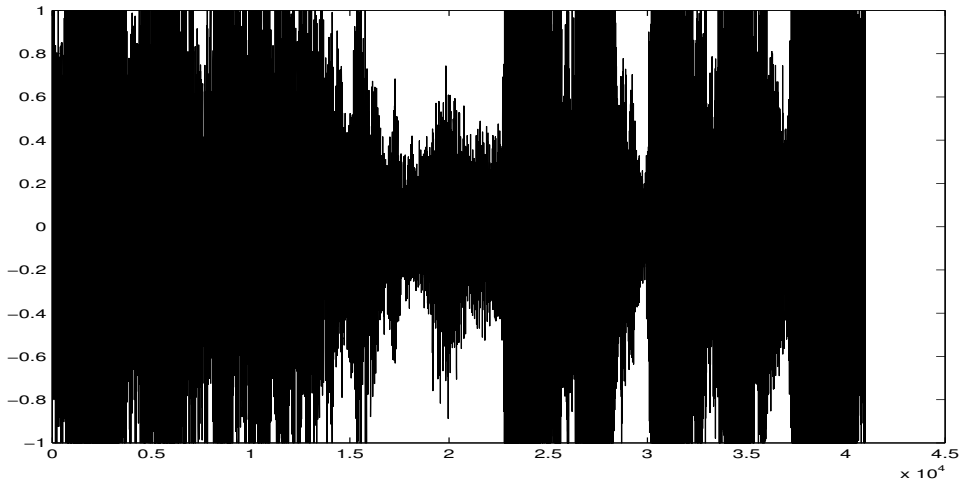


(b) Sizes of clipped blocks for files clipped at 0.3: Left, “Marlene on the Wall”; Right, Bells 1

Figure 7.3: Sizes of clipped blocks at two levels of clipping



(a) Short section of bells



(b) Children's song

Figure 7.4: Clipped audio files

7.5 Statistic sets

For the Gaussian distribution, estimating the parameters from data is straightforward. We collected three parameter sets:

1. An overall (μ, Σ), estimated from the entire data set.

2. Several sets of group-specific (μ, Σ): bells, Carpenters, Dar Williams, Dave, Suzanne Vega and Within Temptation. These represented a variety of the songs. We did not create group-specific parameters for the classical dataset as it was more difficult to determine what might consititute a group; music played by the same orchestra, perhaps.

3. a (μ, Σ) pair for each file.

We used the same mean for both the independent and joint Gaussian distribution, and took the diagonals of the joint covariance for the independent Gaussians.

Estimating the parameters for the time-dependent Gaussian distribution is similar.

For the t-distribution, an iterative process is used to estimate the parameters and both degrees of freedom and (μ, Σ) must be optimised for. This is time-consuming, and makes it more difficult to obtain parameters averaged over large data sets: all the data must be examined at once. Rather than obtain overall parameter sets, I computed a single set of parameters for each of the files in the problem set and some files "similar" to those in the problem set (songs by the same artists, for example). We estimated a value for ν by the procedure described in §

For the above distributions, we collected information using block sizes of 512, 1024 and 2048. For the multivariate t-distribution, maximum-likelihood parameter estimation was much slower, and we only considered block sizes of 512, using the same data sets as for the univariate t-distribution.

The methods for computing the parameters are as described during the exploratory analysis.

Chapter 8

Results and Analysis

8.1 Experiments with a single file

We began with experiments on a ten-second clip from the middle of "Caged". This clip contained only instrumental sound; primarily a bass guitar. It began with a burst of loud noise and then proceeded at a fairly even loudness. Most of the restoration attempts resulted in the poorest results on that first burst, succeeding better over the remainder of the file.

We examine the performance on the four problem set files which we'll refer to as Missing-128, Missing-384, Clipped-0.1, Clipped-0.3. Missing-128 and Clipped-0.3 were quite pleasant to listen to; in particular Clipped-0.3 was barely damaged. Missing-384 was less pleasant; the missing data throughout manifested itself in a throb which disturbed the true signal. Clipped-0.1 was little more than a harsh hiss. A selection of our results are presented below.

In the results below we have sometimes included the "perceived" quality of a file. It should be emphasized that this is a very rough measure which varies from person to person. However, it may give some idea of which files were better "restored" and which models are suited to audio data. Perceived quality is on a scale of 1 to 10 where 1 denotes pure noise and 10 denotes perfect quality. Where possible we enlisted unbiased opinions on file quality.

As stated previously, our intention was not to carry out every test on every file in

Distribution	MSE (data)	MSE (Fourier)	MSE (power)	Diffs	Variance	Perceived
Independent Gauss	0.0103	0.825	696.157	0.00312	0.00946	9
Multivariate Gauss	0.0098	0.791	640.280	0.0033	0.00948 8	
Time dependent Gauss	0.00967	0.783	609.828	0.00389	0.0095	7
Independent t	0.00620	0.505	480.356	0.0102	0.0100	9.5
Multivariate t	0.0246	1.977	2477.522	0.0152	0.0103	6

(a) Missing-128

Distribution	MSE (data)	MSE (Fourier)	MSE (power)	Diffs	Variance	Perceived
Independent Gauss	0.0157	5.88	20392.19	0.00602	0.00663	6
Multivariate Gauss	0.0229	8.59	22455.66	0.00300	0.00704	4
Time dependent Gauss	0.102	38.4	24939.169	0.0711	0.0797	1
Independent t	0.0205	7.68	20796.62	0.0087	0.00894	6
Multivariate t	0.0289	10.82	23212.49	0.0046	0.00986	4

(b) Clipped-0.1

Figure 8.1: Tests with the different distributions on two of the "Caged" problemset files. The "diffs" and variance:amplitude ratio of the clean file are respectively 0.01002 and 0.01012

the problem set, but to explore a selection of models over a variety of problems. Where there are gaps in the tables below, this indicates a test which we did not carry out either due to difficulties with numerical instability or because we considered it unnecessary.

8.1.1 Distribution

We compared the five distributions discussed earlier. Figure 8.1 shows the results of applying each of the distributions to the file, using a block size of 512. For the Gaussians, the overall μ and Σ were used. For the t-distribution, the ν, μ, and Σ estimated to be best for a different audio file, "Hometown Girl" were used. These we had determined experimentally behaved well as an "overall" parameter set. The column "Diffs" refers to the average difference in amplitude between two datapoints; the continuity

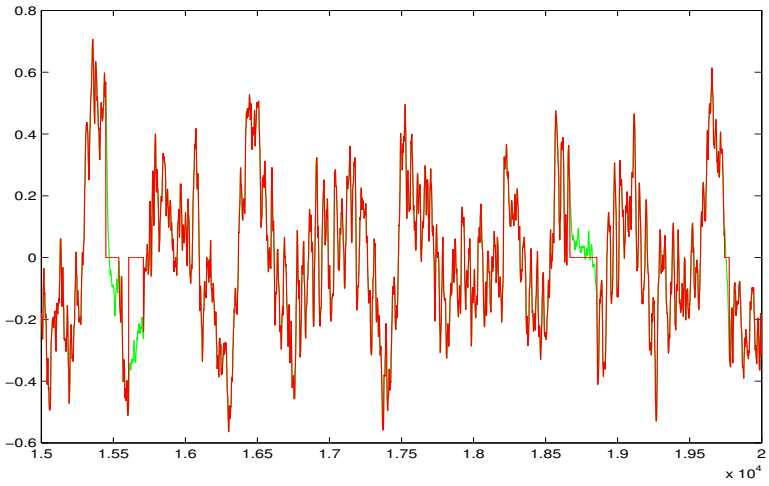


Figure 8.2: Time-dependent Gaussian on missing data. The green line indicates repaired data, the red line the original data

parameter discussed in §5.8.1

For the missing-128 file, the perceived quality approximately follows the MSE values, which match one another, although the scales vary. The continuity is consistently lower than the continuity in the file. It is highest for the perceptually "best" repair. The sound of the errors in the clipped-0.1 is quite different for the independent t-distribution and the independent Gaussian distribution. In the file repaired with independent-t distribution there is a sort of "underwater" noise throughout the file, while with the Gaussian distribution there is just a hissing noise obscuring the clean data. This may be partially attributable to the use of a less general parameter set for the t-distribution—the "underwater" noise may be the model attempting to force the characteristic sound of the file used for the parameters onto this file.

Figure 8.2 shows an extract from the results of the time-dependent Gaussian distribution on the missing-128 and the clipped-0.1 problems. The green line indicates repaired data, the red line the original data. We can see that the system has filled in the missing chunks with plausible values, apart from a tendency for too many higher frequency components to be included.

Figure 8.3 shows an extract from the results of the joint Gaussian on the clipped data. It is clear what is happening here; the interpolation is rarely sufficiently large.

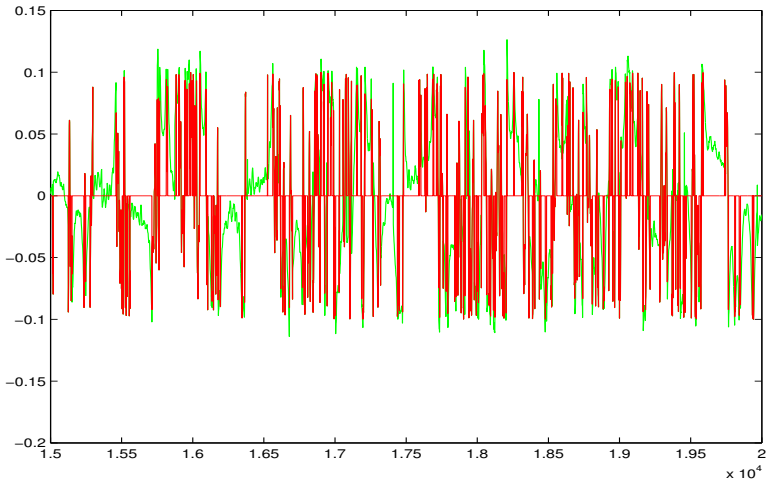


Figure 8.3: Joint Gaussian on clipped data. The red line represents the original data with points above 0.1 all set to zero. The green line represents the repaired values.

This could perhaps be improved upon by constraining the clipped data points to be above the level of clipping.

The time-dependent Gaussian distribution behaves particularly badly on the missing data. In terms of assault on a listener's ears, painfully badly! Figure 8.4 shows clearly why this is the case: after the first block, which are not time-dependent, the interpolated data is far too large to be correct. This occurs because the small values in the covariance matrix result in overly large estimated means at each time step. It might be possible to ameliorate this effect to some extent. For example by treating the individual Fourier components independently, although this would lose any benefits of exploiting correlations between frequencies.

Figure 8.6 shows an extract from the results of the two t-distributions on clipped data. The broken signal is shown in red over the "repair" in green. We can see that there are some similarities, but the multivariate t-distribution tends towards larger amplitudes, and in particular to upward spikes. There are several instances of upward spikes in the diagram for the multivariate t-distribution where the spikes are downward in the independent t-distribution, or where the spikes have twice the amplitude in the multivariate t-distribution. In the case of the independent t-distribution, we can pick out the form of a sinusoidal which undulates once during the block under examination.

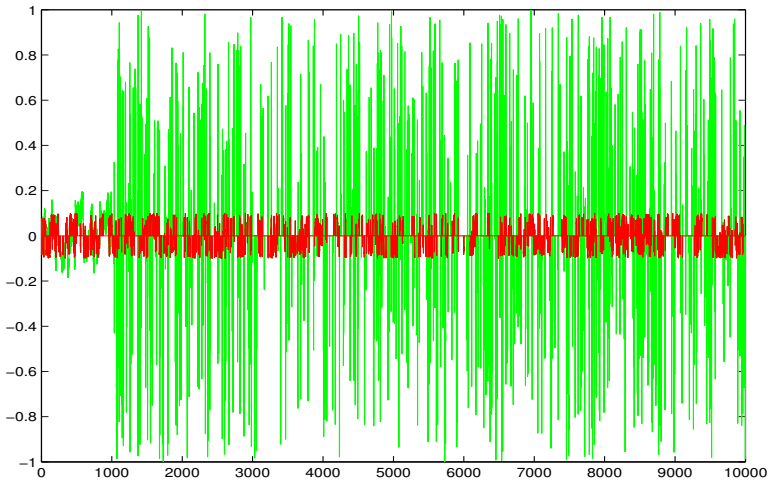


Figure 8.4: Time-dependent Gaussian on clipped data. The red line represents the original data with all points above 0.1 set to 0. The green represents the repaired data.

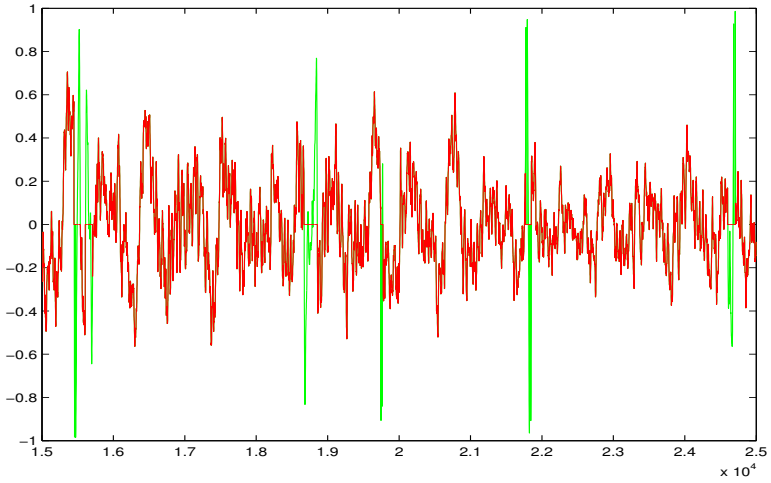
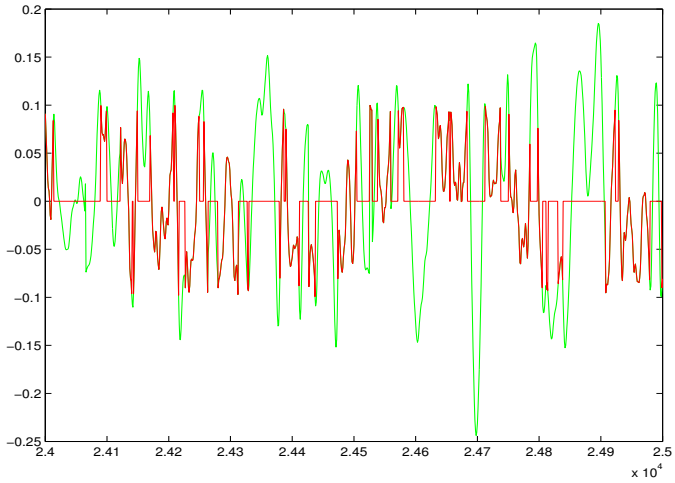
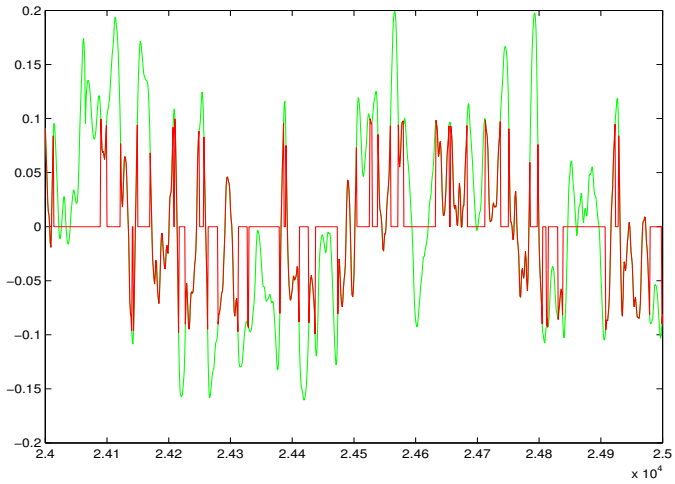


Figure 8.5: The multivariate t distribution on missing data



(a) Multivariate t



(b) Independent t

Figure 8.6: The t-distribution on clipped data

Distribution	Parameters	MSE data	MSE Fourier	MSE power	Diffs	Variances	Per
Independent Gauss	overall	0.0134	2.70	2701.4	0.0012	0.00819	
Independent Gauss	Never-ending Story	0.0152	2.80	2965.4	0.0026	0.0085	
Multivariate Gauss	overall	0.0123	2.75	3083.48	0.00074	0.0082	
Multivariate Gauss	Never-ending Story	0.020	4.56	4806.2	0.0086	0.0104	
Independent t	"overall"	0.0165	3.463	5583.8	0.0048	0.00882	
Independent t	Never-ending Story	0.0159	3.220	3574.4	0.00499	0.00955	

(a) Missing-384

Distribution	Parameters	MSE data	MSE Fourier	MSE power	Diffs	Variances	Per
Independent Gauss	overall	0.0157	5.88	20392.1	0.00603	0.0066	
Independent Gauss	Never-ending Story	0.014	5.46	18805.9	0.00841	0.00766	
Multivariate Gauss	overall	0.022	8.59	22455.6	0.003	0.00704	
Multivariate Gauss	Never-ending Story	0.0223	8.47	21191.8	0.0064	0.00948	
Independent t	"overall"	0.0204	7.68	20796.62	0.00873	0.00894	
Independent t	Never-ending Story	0.0104	3.93	16726.24	0.015	0.0092	

(b) Clipped-0.1

Table 8.1: Effect of initial parameters

This is not evident in the multivariate case.

The overall effect is that the errors caused by the multivariate distribution are louder errors. A similar problem occurs with the multivariate t-distribution on the missing data: the resulting file sounds worse than the original as the interpolated points are too large.

8.1.2 Parameters

Choice of initial parameters For each of the Gaussian distributions, we compared the effects of using the "overall" (μ, Σ) with using (μ, Σ) taken from another song by

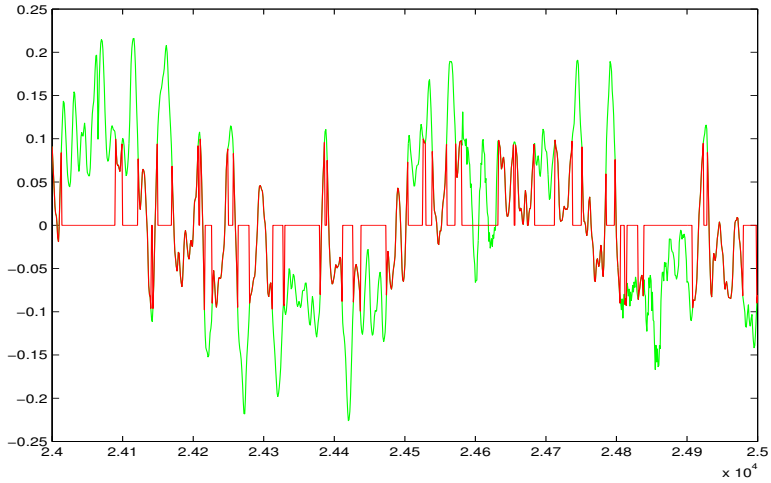
the same artists, "Never-ending Story". For the t-distributions we compared the ν, μ, and Σ from "Hometown Girl" with the t-distribution parameters from "Never-ending Story". Figure 8.1.2 shows some example results. For the missing data problem, there is little difference between the two options; the "Never-ending Story" parameters are slightly better for the t-distribution and slightly less good for the Gaussian, but there is little to choose between them. On the clipped data the results are more interesting: the "Never-ending Story" parameters with the t-distribution behave quite differently from the "overall" ("Hometown Girl") parameters. However, despite the much lower mean-squared errors, the effects of using the "Never-ending Story" parameters are less audibly pleasing; the underwater sound is worsened. Figure 8.7 shows short sections of each; it is difficult to see from this why the version using "Never-ending Story" has the underwater noise as its characteristic—it may be the effect of the bass guitar in Never-ending Story–but it is clear that the two examples are different in character, with the "Never-ending Story" version frequently having higher frequency components (which is not consistent with our hypothesis of a bass guitar causing the effect).

The degrees of freedom parameter for the t-distribution, ν, as found by our ad-hoc search procedure, was slightly higher (7) for "Never-ending Story" than it was for "Hometown Girl" (6). This may not be a sufficient difference to be significant, but may suggest that "Never-ending story" is slightly more randomly distributed and therefore slightly more appropriate for use as an "overall" parameter which attempts to approximate an average over diverse files.

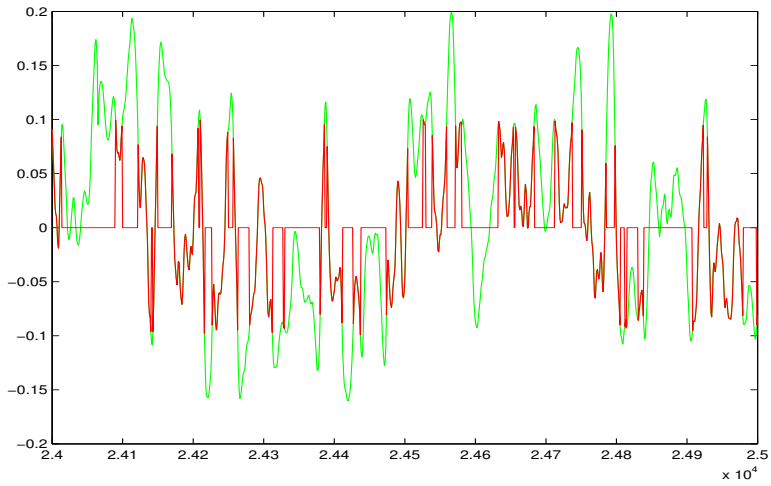
For the multivariate Gaussian, the "Never-ending Story" version sounds worse than the overall version as clicks can be heard throughout the piece. These represent points where the interpolation is too loud, as shown in figure 8.8.

Updating By the time we have repaired several seconds of a file, we should have enough data with which to accurately estimate the parameters for this file. We experimented with two settings for updating the covariance along a file: one weighted the most recent blocks much more highly than the other. It was only sensible to do this for the Gaussian distributions as determining t-distribution parameters requires maximum likelihood estimation which is time-consuming.

It is clear that updating the parameters in this way had virtually no effect on the



(a) Never-ending Story



(b) Hometown Girl

Figure 8.7: Effects of parameter set on t-distribution performance

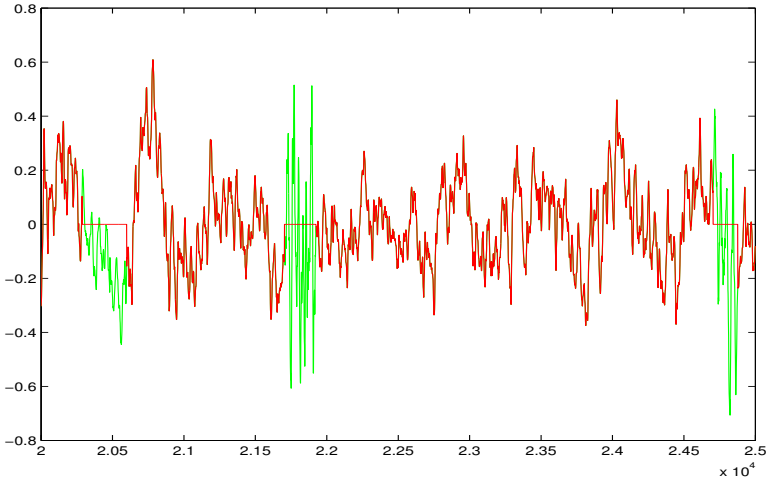


Figure 8.8: Multivariate Gaussian with Never-ending Story parameters

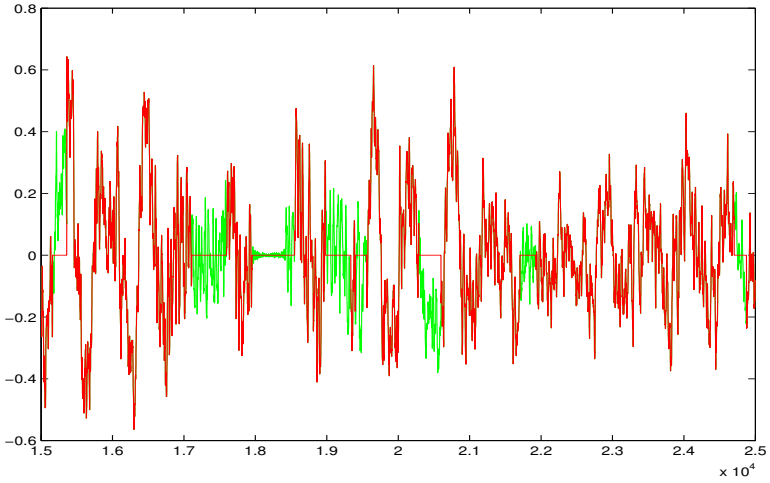


Figure 8.9: Independent t-distribution on missing-384 problem

Distribution	Update rate	MSE data	MSE Fourier	MSE power	Diffs	Variance
Independent Gauss	None	0.0103	0.825	696.2	0.00312	0.0094
Independent Gauss	Slow	0.0107	0.863	726.1	0.0024	0.00946
Independent Gauss	Fast	0.0112	0.902	783.5	0.00192	0.0094
Multivariate Gauss	None	0.0132	1.058	1391.443	7.94e-05	0.0096
Multivariate Gauss	Slow	0.0132	1.056	1392.45	5.8e-05	0.0096
Multivariate Gauss	Fast	0.0132	1.056	1391.29	5.5e-05	0.0096

(a) Missing-128

Distribution	Update rate	MSE data	MSE Fourier	MSE power	Diffs	Variance
Independent Gauss	None	0.0157	5.88	20392.2	0.00602	0.00663
Independent Gauss	Slow	0.0156	5.86	20377.7	0.00602	0.00663
Independent Gauss	Fast	0.0157	5.87	20395.8	0.00601	0.00663
Multivariate Gauss	None	0.0316	11.842	24111.05	0.000259	0.012
Multivariate Gauss	Slow	0.0316	11.845	24113.21	0.00024	0.0121
Multivariate Gauss	Fast	0.0316	11.84	24112.86	0.00024	0.0122

(b) Clipped-0.1

Table 8.2: Updating the parameters along the file

outcome. There was also no discernible audible effect. This corresponds to the limited effects of changing the parameters for the Gaussians shown in the previous section. If it were practical to update the t-distribution parameters, the gain might be greater. This is perhaps a result of the sparsity of the t-distribution; as we saw in the matrices for Σ for the t-distribution and the scale of the eigenvalues, the available information is better distributed within the Σ matrix, so the distinction between two files can be made stronger.

These experiments were very brief; there is room for much more analysis of decay settings.

8.1.3 Block size

We compared three block sizes: 512, 1024 and 2048.

Table 8.3 shows some of the metrics on the data points on the two independent distributions. It appears that the block size hs little effect on the perceived quality, although more so with the t-distribution than the Gaussian. However, in both cases the squared errors are reduced as block size increases; the effects being particularly noticeable with the t-distribution.

We do not show the results for the multivariate distributions as scaling difficulties made it impossible to carry out any but the simplest tests with larger block sizes on these distributions. We do not expect that the behaviour would be noticeably different, because the same number of frequencies have significant values on the larger blocks as the smaller ones. However, it would be interesting to experiment further when possible.

8.1.4 Type of damage

On both types of damage if there were large chunks of errors, we felt that fixing worked best perceptually if the fix attempt was more conservative than the correct value, as it was rare that an exactly correct match would be found.

8.1.5 Summary

Looking at the various parameters on this file, it seems that

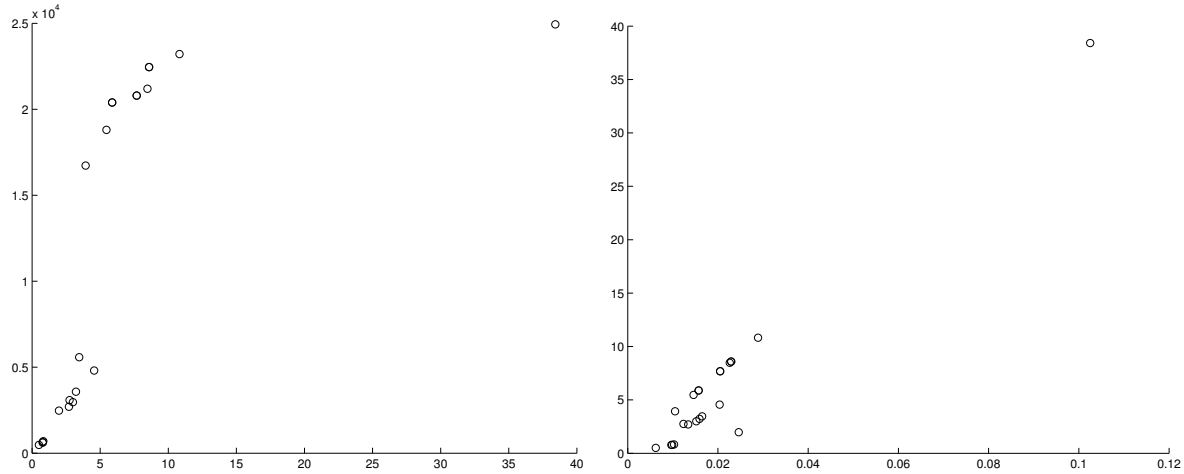
Distribution	MSE data: 512	1024	2048	MSE Fourier: 512	1024	2048
Independent Gauss	0.0103	0.00904	0.00842	0.825	0.733	0.683
Independent Gauss	0.0157	0.0151	0.0152	5.88	5.67	5.69
Independent Gauss	0.0134	0.0128	0.0129	2.71	2.55	2.46
Independent Gauss	0.0299	0.0258	0.027	0.46	0.404	0.422
Independent t	0.0127	0.00824	0.00394	1.02	0.669	0.323
Independent t	0.0205	0.0175	0.00557	7.68	6.55	2.09
Independent t	0.0164	0.0148	0.0133	3.41	2.61	1.88
Independent t	0.0234	0.017	0.0126	0.36	0.268	0.202

(a)

Distribution	Diffs: 512	1024	2048	Perceived: 512	1024	2048
Independent Gauss	0.00312	0.00311	0.00338	9	9	9.5
Independent Gauss	0.00603	0.00638	0.0066	7	7	7
Independent Gauss	0.00123	0.00119	0.00139	7	7.5	7
Independent Gauss	0.0676	0.071	0.072	9.5	9.5	9.5
Independent t	0.00825	0.0101	0.0108	8	9	9.5
Independent t	0.00874	0.00986	0.0218	6	7	8
Independent t	0.00507	0.00682	0.00768	6	7	8
Independent t	0.0902	0.098	0.114	9.5	9.5	9.5

(b)

Table 8.3: Effect of blocksize



(a) Fourier and power spectra errors

(b) Fourier and data point errors

Figure 8.10: Mean-squared errors

- Very noticeable improvements can be made to badly clipped data
- Using a joint distribution gains little and may in fact result in overestimation and loss of accuracy
- Increasing the block size can have some benefit, although given the way complexity scales with in particular the quadratic programming optimisers we might not consider the small benefits worth it for joint distributions
- The Gaussian distribution is not very sensitive to the choice of parameters
- The t-distribution is sensitive to the choice of initial parameters
- The time-dependent Gaussian distribution tends to over-estimate the data

8.2 Evaluation metric

For the most part in the above section, we have included all the proposed metrics. However, it is fairly clear that the three squared errors are proportional. Figure 8.10

demonstrates this with scatter plots of the error rates. It is also clear that for most part–although not in all cases–the MSE metrics are a reasonable metric for determining the perceived quality. We therefore focus on the MSE of data points in the following section. This means that when examining the results on real data, we have no better metric than the perceived quality.

Figure 8.11 demonstrates an example where the audio quality does not match the apparent accuracy of the restoration. The blue line represents the data clipped at a level of 0.1. The red line represents the true data without clipping. The green line represents the restoration attempt on the data. We can see that the green line for the t-distribution more accurately follows the shape of the true data, and is a better approximation to in terms of absolute value of the data points. It is therefore not surprising that the MSE is lower for the t-distribution restoration. However, the Gaussian restoration sounds much better than the t-distribution restoration—and much better than the broken data.

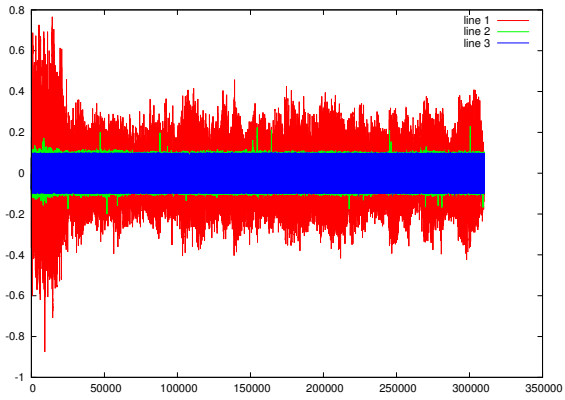
8.3 Evaluation on the problem set

Having examined some of the effects of the various parameters on a single file, we broadened my experiments to the problem set as defined in chapter 7. Running tests over a large number of files was necessary to thoroughtly compare the effects. However, it was not possible to listen to or view graphically all the repaired files; we rely on the evaluation metrics to determine how well the system was performs, listening to occasional files which appeared interesting or anomalous. We attempted to listen to at least one repaired example associated with each of the original problem set files.

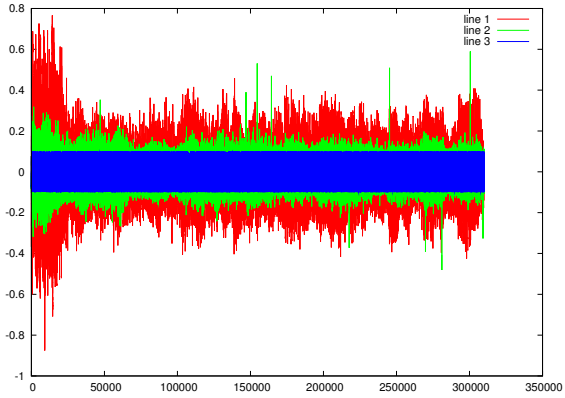
8.3.1 Distribution

The figures for the joint and time-dependent Gauss seem surprisingly small. For these tests we used just five files from the problemset: “Caged”, “Moon River”, “Bells 1”, “Ride of the Valkyries” and “Fur Elise”. The results are low because the performance is particularly good for “Fur Elise” and “Ride of the Valkyries”. All the multivariate distributions (t and the two Gaussian ones) performed particularly badly on “Caged”.

Looking more closely at “Fur Elise” and “Ride of the Valkyries”, it appears that



(a) Gaussian distribution



(b) Gaussian distribution

Figure 8.11: Restoration of heavily clipped data. The blue line represents the data clipped at a level of 0.1. The red line represents the true data without clipping. The green line represents the restoration attempt on the data.

Distribution	MSE on missing-128	missing-384	missing-768	clipped-0.1	clipped-0.3
Independent Gauss	0.0022	0.00228		0.0155	0.00921
Independent t	0.0022	0.00231		0.0160	0.0155
Joint Gauss	0.00187			0.0187	
Time-dependent Gauss	0.00183		0.00195	0.0175	
Multivariate t	0.00211	0.00253		0.012772	

(a) MSE

Distribution	Variance on missing-128	missing-384	missing-768	clipped-0.1	clipped-0.3
Independent Gauss	0.0023	0.0022		0.0125	0.00817
Independent t	0.00228	0.0022		0.0137	0.0313
Joint Gauss	0.00217			0.0249	
Time-dependent Gauss	0.00212		0.00195	0.024	
Multivariate t	0.00147	0.00189		0.0201	

(b) Standard Deviation

Figure 8.12: Results of different distributions on the problem set

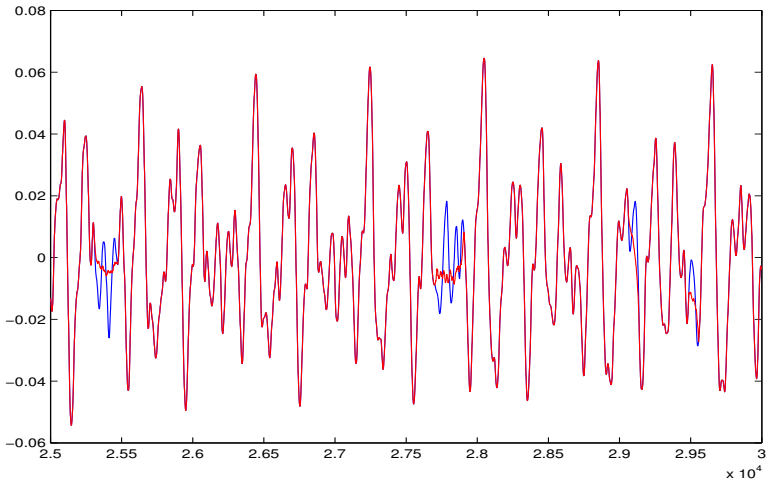


Figure 8.13: Fur Elise

Distribution	Parameters	MSE missing-128	missing-384	clipped-0.1	clipped-0.3
Independent Gauss	Overall	0.0022	0.00228	0.0155	0.00921
Independent Gauss	Group	0.0024	0.0027	0.018	0.0092
Multivariate Gauss	Overall	0.00187		0.0187	
Multivariate Gauss	Group	0.0051		0.0424	

Table 8.4: Effects of parameter choice for a Gaussian distribution

the performance is so good because these clips are themselves very quiet. Figure 8.13 shows a section from the "Fur Elise" multivariate Gaussian repair, with the true data in red and the repair data in blue. It's clear that the repair itself is not spectacular. However, the amplitude of the data and therefore the possible amplitude of any error varies between -0.08 and 0.08, a fraction of the possible space. The situation is similar for "Ride of the Valkyries"; the particular sections tested on these two files happen to be quiet.

8.3.2 Parameter choice

Some types of audio file responded more to the choice of parameters than others. For example, using the Bells mean on a Bells file showed a noticeable improvement over the overall mean; while the same effect was not evident for a song file.

Distribution	Parameters	MSE missing-128	missing-384	clipped-0.1	clipped-0.3
Independent t	Params 1	0.0022	0.0023	0.0159	0.0156
Independent t	Params 2	0.0021	0.0023	0.0156	0.0153
Multivariate t	Params 3	0.0022	0.0025	0.016	
Multivariate t	Self	0.0020	0.00234	0.0153	

Table 8.5: Effects of parameter choice for a t-distribution

File	Params 1	Params 2
daws-hometowngirl.mp3.wav	0.0084	0.0096
Bells 2	0.0220	0.0236
Dar Williams - 05 - This Was Pompeii.mp3.wav	0.0045	0.0061
Ride of the Valkyries.mp3.wav	0.0148	0.0163
Suzanne Vega - Marlene On The Wall.mp3.wav	0.0263	0.0322
Sugar Plum Fairies.mp3.wav	0.0087	0.0132
Frank Sinatra - Moon River.mp3.wav	0.0019	0.0019
Fur Elise.mp3.wav	0.0063	0.0063
Morning Mood.mp3.wav	0.0000	0.0000
Bells 1	0.0190	0.0184
Caged.mp3.wav	0.0393	0.0320
Spice Girls - Wannabe.mp3.wav	0.0398	0.0281

Table 8.6: Comparing the parameter sets for the test files on the clipped-0.1 problem

Distribution	512 blocks	1024	2048
Independent t	0.0156	0.0161	0.0196
Independent Gauss	0.0155	0.0141	0.0172
Multivariate Gauss	0.0187	0.0143	0.015

(a) Effects of block size on Clipped-0.1

Distribution	MSE on 512 blocks	1024	2048
Independent t	0.0023	0.0023	0.0023
Independent Gauss	0.0023	0.0022	0.0022

(b) Effects of block size on Missing-384

Figure 8.14: Effects of changing the block size on the MSE of the data points

From table 8.6 we can see that the "best" choice of parameter depends upon the file. The table has been sorted so that all those files for which parameter set 1 is the best are in the top half, and those for which parameter set 2 is the best are in the bottom half. It is difficult to make judgements on such a small sample, and the variation is rarely large. However, there do not seem to be any obvious audio characteristics which identify a file as being more suited to one parameter set or the other: in particular, the two "Bells" files, very similar in character, are in different parts of the table.

8.3.3 Block size

Figure 8.14 lists the effects of the changing the block size on two of the problem set files. It is clear that as was the case for Caged, changing the block size has little effect on the overall error rate. There is some benefit in the case of the multivariate Gaussian, going from a block size of 512 to 1024. It is not clear why there should be any reason for this to be the case. These results contrast with our observations on "Caged", suggesting that some files may be better suited to larger block sizes than others. In particular, "Caged" contains a variety of different instruments including bass instruments—these especially may require larger block sizes to correctly determine the

Distribution	Clipped-0.1 Fraction too small	Clipped-0.3 Fraction too small
Independent Gauss	0.33	0.28
Independent t	0.32	0.28

Table 8.7: Fraction of clipped data not estimated above the clip point

relevant frequency information.

8.3.4 Clipping

We stated earlier our intention of explicitly not including the clipping limits in the model, in order to see to what extent the system correctly estimated data above the level at which it had been clipped. Table 8.7 shows the results for the two independent distributions. It is noticeable that the error rate is not very different between the two clipping rates, although there is quite a lot of data between the levels of 0.1 and 0.3. Both the Gaussian and the t distribution perform equally well in this respect, despite our observations earlier that the t-distribution is likely to be less conservative about the size of a data point.

8.4 Evaluation on real data

We experimented with several of the distributions on the real data files in the problem-set, using blocks of 2048 points. For the data file containing bell music, we tested both the Bells parameters from the original data set and the overall parameters. For the other data files we merely the overall parameters (or, for the t-distribution, the parameters we used in lieu of overall parameters). For simplicity we considered only the independent distributions as these seemed to have been as effective as the more complicated distributions.

For this data, evaluation by MSE measures is not possible. We measured the perceived quality according to an unbiased listener. The listener was also supplied with clips recorded from the same tapes, but without clipping, to suggest the maximum achievable quality for the de-clipping process. Table 8.8 indicates the results.

File	Parameters	independent t	independent Gauss	perceived quality clipped	unclipped
Bells	overall	7	6	6	8
Bells	Bells	6	6	6	8
Westside Story	overall	10	9	9	10
Children's song	overall	7	6	7	9

Table 8.8: Results on real data

The results show that very little real improvement has been made to these files. In the children's song, the errors come in two peaks which are particularly high notes in the clip. Neither the t-distribution nor the Gaussian interpolates the high notes correctly, so the sound is blurred; slightly less of a "squawk" than in the original making it fractionally more pleasant to listen to, but no better quality. For both the distributions, the results are less good with the bell-specific parameters than with the overall parameters.

There is no need to be discouraged by the lack of improvement; the model we are using to handle clipped data remains naive and there is scope for further improvement. We might also obtain good results by incorporating the probabilistic FFT as just one step in a restoration procedure. We need not expect it to achieve everything on its own.

8.5 Summary

In this chapter we have carried out the second part of our aim: the first part was to implement the probabilistic FFT and the second part was to explore its use on audio data. We have tested a number of models and found that the t-distribution may be a suitable model for audio data. We have not demonstrated any conclusive benefits, but observed effects which are suggestive and could benefit from further investigation. In particular, we find that the t-distribution is senstitive to the choice of initial parameters. We have noted that a block size of 512 points is sufficient to perform restoration as effectively as we have succeed in performing it. Finally, we have shown that the probabilistic FFT can be used in a practical application.

8.6 Further investigation

There remains plenty of scope for further investigation within this simple missing-data restoration application:

- More general t-distributions should be investigated, computing the parameters over several data files.
- Different ν parameters for the multivariate t-distribution should be considered. It seems likely that ideally ν will be larger than the value of 1 we used.
- For the independent t-distribution, varying ν with the particular component could be investigated. This should have a limited effect on the outcome.
- For the t-distribution in particular, more experimentation with initial parameters would be useful.
- Ways of updating the t-distribution throughout the restoration could be investigated.
- A univariate time-dependent distribution could be investigate. It seems likely that this will be more practical than the multivariate time-dependent distribution and may have some benefits, while not being subject to the overestimation problems encountered with the multivariate time-dependent distribution.
- For random missing data, it should be possible to create models for estimating the parameters modelling the data file from the clean parts of the data. Whether this approach would be useful is not certain: one of the primary causes of missing data is streaming audio over a low bandwidth connection; in such an application waiting until the whole file is available might not be appropriate. (Such a consideration is to some extent academic, since our system is not fast enough to perform a repair in real time in any case).
- The effect of the scale of clipped data on the restoration could be investigated, in particular in the context of real-world clipped data.

- The optimisation methods could take into account the bounds on the unknown data points ($-/+1$ for missing data; $< -cor > c$, where c is the clip point, for clipped data.
- Ways of modifying the models based on knowledge about audio perception such as Fletcher-Munson curves could be investigated.
- The error rate over the course of a file should be examined.

Chapter 9

Future work

9.1 Noisy model

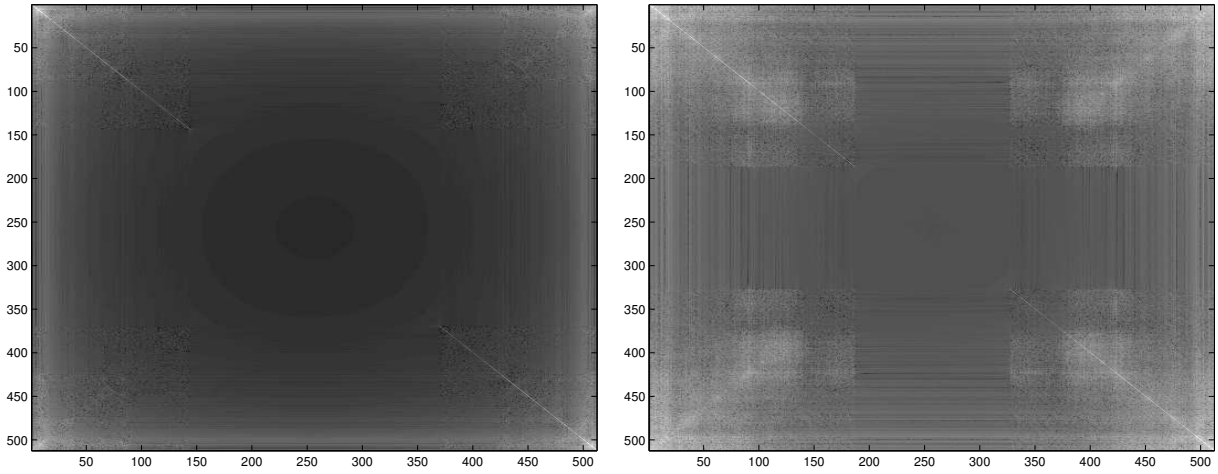
9.1.1 Flexibility in the data points

The first extension to this system is to treat all the points as noisy; both known and unknown, with some prior probabilities on the points. For the unknown points the prior probabilities will have a broad distribution; for the known points the prior probability should encapsulate the fact that the true data point is likely to be within a small value of the observed data point.

These priors can then be updated as before to generate a posterior distribution over the data set, and a maximum likelihood value over the full data set computed.

9.1.2 Detecting noise

The next step is to attempt to determine from the data what parts of the data are noisy. Identifying missing or clipped data is straightforward, and we have been doing that throughout the project. There are known Bayesian techniques for identifying clicks in the datastream which could be incorporated into the system to attempt to provide a better model for the data points. Kalman filter methods could also be applied to the data stream to modify the model.



(a) Speaker 1

(b) Speaker 2

Figure 9.1: Gaussian covariances for two different male speakers

9.2 Other kinds of data

Speech One important application involving speech data is source separation: breaking a single signal up into a number of signals representing the distinct speakers who contributed to the overall audio effect. This is commonly known as the "Cocktail Party Problem". The probabilistic FFT technique described here could be extended to the source separation problem, for example by treating a signal as a "pure" signal, corresponding to a single speaker plus "noise", corresponding to the rest of the speakers. The noise removal approach proposed above would then separate the two signals. The approach could be applied recursively if there are more than two speakers.

It is not clear how effective such a technique would be. From the examination of audio data, we have observed that the covariance matrices which represent voice data are often similar. similar. However, figure 9.1 shows two the Gaussian covariance matrices for two different speakers reading from the Bible. While there are obvious similarities, seems plausible that with a sufficiently detailed model, a signal modelled by a combination of these could be separated. Figure 9.2 shows the covariance matrix for a signal used as a benchmark for the source separation problem, containing signals from two speakers simultaneously. It is possible to visibly identify the frequency com-

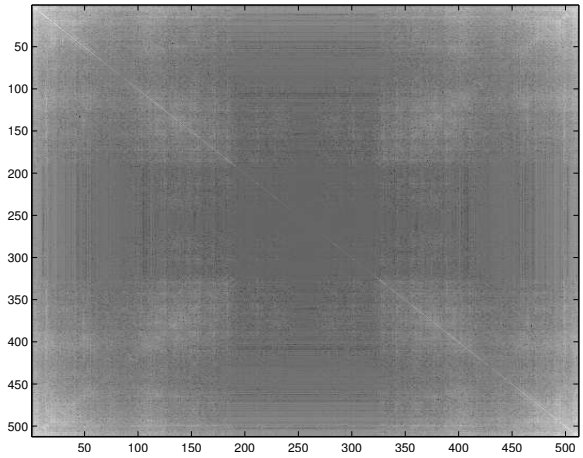


Figure 9.2: The covariance matrix for a signal containing two simulataneous voices

ponents belonging to the two speakers, and it should be possible to build a system with a probabilistic FFT core which can separate these two signals.

In [24], Roweis describes a hidden Markov model approach to source separation in which he focuses on spectral properties of the signal and spectral cues. Roweis's approach only works well if the original speech signals are distinct (a male and a female, for example). In our work we have also seen that the parameters for signals containing, for example, female voice, are all very similar; it is possible that any source separation based on simple Fourier priors will suffer from similar difficulties.

However, something we have investigated in this project which has not been much examined is the use of the t-distribution to model audio frequency components. It may be that the t-distribution would be better able to distinguish the minor frequency differences between similar speakers.

Another problem in Roweis's system is the necessity of training the model on the speakers themselves. It is possible that a source separation system based on the probabilistic FFT would have a similar problem; distinguishing between "male" and "female" might be straightforward, but distinguishing between two similar voices would be better done given prior knowledge of the specific frequencies in the voices.

Image The JPEG algorithm performs a cosine transform on quantised datapoints. Using spectral priors we can more accurately estimate the original datapoints, as de-

scribed in [26]. A two-dimensional extension of our probabilistic FFT could be used in an approach similar to this.

Other series data There are many other forms of series data; spectral approaches are appropriate to any series applications which exhibit some form of periodicity. This may include astrophysical data, medical data, geoscience data, economic data, or many others. The probabilistic FFT approach could be used for any application where frequency-based models are suited to encapsulating information about the data.

Chapter 10

Conclusions

We experiment with a number of different optimisation techniques in order to implement a system which could practically and effectively perform a probabilistic FFT on audio streams with missing or clipped data. We find that applying this technique efficiently requires some care in the choice of optimiser and in handling the near-singular covariance matrices which occur with Gaussian priors. We experiment mostly unsuccessfully with eigenvector techniques to improve efficiency.

We apply a number of different models for the Fourier component priors to the system, comparing independent, joint, and time-dependent Gaussians, and independent and joint t-distributions. We find that in general the joint distributions did not perform well on practical audio problems; this may be a consequence of the form of audio data. The t-distribution behaves slightly better on audio data than the Gaussian distribution, but the effect is not large.

We find that for many practical problems this technique did a poor job as a standalone audio restoration application, but for problems with small amounts of unknown or clipped data good results could be achieved. It seems likely that including perceptual characteristics of audio data in the model might improve matters. Audio data is sensitive to the scale of errors; "loud" errors are more noticeable than "quiet" errors, and often data at about the right volume which looks plausible sounds wrong. This behaviour may be different on other forms of data.

We experiment with a number of evaluation heuristics for audio data. It was hard to find metrics which are a reliable judge of perceived quality; most of the time the

mean-squared errors were sufficient. We find that the mean-squared errors in the data and the Fourier domains can be treated equivalently and provide as much information about the perceptual quality of the data as any other metric we considered.

Finally, we suggest a number of directions for future work both within our simple audio restoration application and for the probabilistic FFT in general.

Bibliography

[1] Jonathan S. Abel and Julius O. Smith. Restoring a clipped signal. In *International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991*, volume 3, pages 1745–1748, New York, May 1991. IEEE Press.

[2] Francis R. Bach and Michael I. Jordan. Blind one-microphone speech separation: A spectral learning approach. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 65–72. MIT Press, Cambridge, MA, 2005.

[3] David Barber and Peter Sollich. Stable directed belief propagation in Gaussian DAGs using the auxiliary variable trick. In *Neural Information Processing Systems*, 2005.

[4] Christopher M. Bishop and Markus Svensen. Robust Bayesian mixture modelling. In *European Symposium On Artificial Neural Networks*, 2004.

[5] Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[6] Rui Cai, Lie Lu, Hong-Jiang Zhang, and Lian-Hong Cai. Improve audio representation by using feature structure patterns. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2004.

[7] G. Cocchi and A Uncini. Subband neural networks prediction for on-line audio signal recovery. *IEEE Transactions on Neural Networks*, 13:867– 876, 2002.

[8] Andrzej Czyzewski. Learning algorithms for audio signal enhancement: Part 1 Neural network implementation for the removal of impulse distortions. *Journal of the Audio Engineering Society*, 10:815, 1997.

[9] Andrzej Czyzewski and Rafal Królikowski. Neuro-rough control of masking thresholds for audio signal enhancement. *Neurocomputing*, 36(1-4):5–27, 2001.

[10] W. Fong, S. Godsill, A. Doucet, and M. West. Monte Carlo smoothing with application to audio signal enhancement. *IEEE Transactions on Signal Processing*, pages 438–449, 2002.

[11] W. N. W. Fong and S. J. Godsill. Monte carlo smoothing for non-linearly distorted signals. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 3997–4000, 2001.

[12] Canazza S.; Coraddu G. and De Poli G.; Mian G.A. Objective and subjective comparison of audio restoration methods. *Journal of New Music Research*, 30:93–102, 2001.

[13] S. J. Godsill, P. J. Wolfe, and W. N. W. Fong. Statistical model-based approaches to audio restoration and analysis. *Journal of New Music Research*, 2:323–338, 2001.

[14] Simon J Godsill and Peter J W Rayner. *Digital Audio Restoration - A statistical model-based approach*. Springer-Verlag, 1998.

[15] P.C. Gregory. A Bayesian revolution in spectral analysis. In *American Institute of Physical Proceedings*, volume 568, pages 557–568, 2001.

[16] `http://www2.cs.uregina.ca/~roughset/`. Electronic bulletin of the rough set community.

[17] E. Jaynes. Bayesian spectrum and chirp analysis. In *Maximum Entropy and Bayesian Spectral Analysis and Estimation Problems*, pages 1–37. D. Reidel Publishing Company, Holland, 1987.

[18] Chuanhai Liu and Donald B. Rubin. Ml estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

[19] M.Davy and S. J. Godsill. Bayesian harmonic models for musical signal analysis. In *IEEE International Conference on Acoustics Speech and Signal Processing*, 2002.

[20] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–475, 1999.

[21] E. Pampalk. A Matlab toolbox to compute music similarity from audio. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, October 10-14 2004. Universitat Pompeu Fabra.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[23] William H. Press, Saul A. Teulosky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[24] Sam T. Roweis. One microphone source separation. In *Advances in Neural Information Processing Systems*, pages 793–799, 2000.

[25] H.R.R. Scott and R. Wilson. A multiresolution audio restoration algorithm. In *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, pages 151–154, 1995.

[26] Amos Storkey and Michael Allan. Cosine transform priors for enhanced decoding of compressed images. In *Intelligent Data Engineering and Automated Learning*, pages 533–539, 2004.

[27] Amos J Storkey. Generalised propagation for fast fourier transforms with partial or missing data. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[28] Paul T. Troughton. Bayesian restoration of quantised audio signals using a sinusoidal model with autoregressive residuals. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1999.

[29] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, volume 10, 2002.

[30] P. Walmsley, S. Godsill, and P. Rayner. Bayesian modelling of harmonic signals for polyphonic music tracking, 1999.

[31] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[32] R. Wilson, A.D. Calway, and E.R.S Pearson. A Generalized Wavelet Transform for Fourier Analysis : the Multiresolution Fourier Transform and its Application to Image and Audio Signal Analysis. *IEEE Transactions on Information Theory*, 38(2):674–690, 1992.

[33] Patrick Wolfe and Simon Godsill. Perceptually motivated approaches to music restoration. *Journal of New Music Research*, 30:83–92, 2001.

Appendix A

Audio Data

Those mentioned in the text have been marked with a * and highlighted in blue. The remainder were used to contribute to the statistics.

A.0.1 Bells: from the Swan Bells, Perth

File	Content
*1	Rounds and Call Changes on 16
2	Rounds and Call Changes on 12
3	Grandsire Caters on the back 10
*4	Stedman Triples on the back 8
5	Plain Hunt Maximus on the light 12
6	Call Changes on the light 10
*7	Little Bob Major on the middle 8
8	St. Martins Bob Doubles on the G# 6
9	Grandsire Triples on the light 6
10	Double Court Bob Minor on the front 6
11	Plain Hunt Sixteen

A.0.2 Songs

Name	Artist
16 going on 17	Sound of Music
A winter's tale	Davis Essex
America	Simon and Garfunkel
American Pie	Madonna
*As Cool As I Am	Dar Williams
As Time Goes By	Frank Sinatra
Bananaphone	Raffi
*Beautiful Morning	Boo Radleys
Beautiful Stranger	Madonna
Birdhouse in Your Soul	They Might Be Giants
Blonde Over Blue	Billy Joel
Blowin' in the Wind	Bob Dylan
Bridge Over Troubled Water	Simon and Garfunkel
Bring It All Back	S Club 7
*Caged	Within Temptation
Climb Ev'ry Mountain	Sound of Music
Colours of Day	Philip Heak
Cool Cat	Garfield
Cow Town	They Might Be Giants
Daddy Sang Bass	Johnny Cash
Diamonds Are Forever	Shirley Bassey
*Dressed for Success	Roxette
February	Dar Williams
Girls Just Wanna Have Fun	Cyndi Lauper
Goodbye To Love	Carpenters
Gypsy	Suzanne Vega
*Hometown Girl	Dave Allen-Williams
How Can I Keep From Singing	Enya
How Could You Say No	Julie Miller

*I am Cow	Unknown
I Had No Right	Dar Williams
*I Vow to Thee My Country	Unknown
Ice Queen	Within Temptation
Imagine	Beatles
In The Eye	Suzanne Vega
Iowa	Dar Williams
Jerusalem	Unknown
Just Wanna Make Love to You	Etta James
*Kebabulous	Dave Allen-Williams
*Last Christmas	Wham!
Like A Virgin	Madonna
Little Boxes	Pete Seeger
Love Me for a Reason	Boyzone
Man I Feel Like a Woman	Shania Twain
Manic Monday	Bangles
Maria	Sound of Music
*Marlene On The Wall	Suzanne Vega
Material Girl	Madonna
May The Road Rise To Meet You	Northumbrian Community
Mercedes Benz	Janis Joplin
Milk And Toast And Honey	Roxette
*Moonlight Shadow	Suzanne Vega
*Moon River	Frank Sinatra
Morning Has Broken	Neil Diamond
*Never-Ending Story	Within Temptation
Oops I did it again	Britney Spears
Party Generation	Dar Williams
*Pre-text	Dave Allen-Williams
Puff the Magic Dragon	Carpenters
Puppet on a String	Sandie Shaw

Rainy Days and Mondays	Carpenters
Remember You're A Womble	Wombles
Rotterdam	Beautiful South
Seek Ye First	Maranatha! Singers
Silver Rainbow	Genesis
Sing	Carpenters
*Solitude Standing	Suzanne Vega
Somewhere Over The Rainbow	from The Wizard of Oz
Southern California Wants to be Western New York	Dar Williams
Stop Smoking	Dar Williams
Stop!	Spice Girls
*Streets of London	Cat Stevens
The Bare Necessities	from The Jungle Book
The Christians and the Pagans	Dar Williams
The End of the Summer	Dar Williams
The Pointless Yet Poignant Crisis of a Co-Ed	Dar Williams
The Wombling Song	Wombles
There's a Kind of Hush	Carpenters
*This Was Pompeii	Dar Williams
Top of the World	Carpenters
Tragedy	Steps
Uptown Girl	Billy Joel
*Venus	Bananarama
Viva Forever	Spice Girls
*Wannabe	Spice Girls
We didn't start the fire	Billy Joel
What's up	4 Non Blondes
Wiggle It	Two In A Room

A.0.3 Classical

File	Composer
Air on a G-string	Bach
*Allegro Con Brio	Beethoven
Blue Danube	Strauss
Bolero	Ravel
Canon	Pachebel
Carmina Burana	Orff
Cello Concerto	Unknown
Dance of the Knights	Prokoviev
Eine Kleine Nachtmusik	Mozart
Flower Duet	Delibes
*Fur Elise	Beethoven
*Hall of the Mountain King	Grieg
Hallelujah	Handel
Jesus Bleibt Mein Freund	Bach
Jupiter	Holst
Mars	Holst
*Minuet in G	Bach
*Moonlight Sonata	Beethoven
*Morning Mood	Grieg
Nessun Dorma	Puccini
Ode to Joy	Beethoven
*Pomp and Circumstance	Elgar
Radetsky March	Strauss
Rhapsody in Blue	Gershwin
*Ride of the Valkyries	Wagner
Rigoletto	Verdi
Rondo alla Turca	Mozart
*Sarabande	Unknown
Sugar Plum Fairy	Tchaikovsky
Suiten fur Violoncello Solo	Unknown
Dance of the Merlitons	Tchaikovsky
The Magic Flute	Mozart
The Marriage of Figaro	Mozart
The New World	Dvorak
Toccata and Fugue	Unknown

A.0.4 Real

File	Artist
*Bells	Bells of St. Mary Redcliffe, Bristol (12 bells)
*Children's Song	Children's tape, unknown
*Westside Story	Westside Story, Bernstein

Appendix B

Covariance matrices

Figure B.1 demonstrates a formula for computing covariance matrices incrementally.

Consider the element C_{ij} of the covariance matrix:

$$C_{ij} = < (x_i - \mu_i)^T (x_j - \mu_j) >$$

$$= \frac{1}{N} \sum_{k=1}^{n} (x_i^k - \mu_i)^T (x_j^k - \mu_j)$$

Now suppose that we have not N data points, but N+1:

$$C_{ij} = \frac{1}{N+1} \sum_{k=1}^{N+1} (x_i^k - \mu_i)^T (x_j^k - \mu_j)$$

$$= \frac{1}{N+1} (\sum_{k=1}^{N} (x_i^k - \mu_i)^T (x_j^k - \mu_j) + (x_i^{N+1} - \mu_i)^T (x_j^{N+1} - \mu_j))$$

$$= \frac{N}{N+1} C_{ij}^{1..N} + \frac{1}{N+1} (x_i^{N+1} - \mu_i)^T (x_j^{N+1} - \mu_j)$$

Hence the covariance can be computed incrementally. Suppose now that we have N+M data points:

$$C_{ij} = \frac{1}{N+M} \sum_{k=1}^{N+M} (x_i^k - \mu_i)^T (x_j^k - \mu_j)$$

$$= \frac{1}{N+M} \sum_{k=1}^{N} (x_i^k - \mu_i)^T (x_j - \mu_j) + \frac{1}{N+M} \sum_{k=1}^{M} (x_i^{k+N} - \mu_i)^T (x_j^{k+N} - \mu_j)$$

$$= \frac{N}{N+M} C_{ij}^{1..N} + \frac{M}{N+M} C_{ij}^{N+1..N+M}$$

Finally, consider the case where $M = N$:

$$C_{ij} = \frac{N}{2N} C_{ij}^{1..N} + \frac{N}{2N} C_{ij}^{N+1..2N}$$

$$C_{ij} = \frac{1}{2} (C_{ij}^{1..N} + C_{ij}^{N+1..2N})$$

It is easy to see that this will extend to more than two covariance matrices. Therefore the covariance matrix of a large quantity of datapoints is the mean of the covariances of some partitioning of the datapoints. This theorem [!] is valuable in computing the covariance matrix over several files, or when there is too much data to reasonably load into `MATLAB` all at once.

We should bear in mind, however, that these equations assume that the mean for the whole data is known in advance, while `MATLAB`'s covariance routines compute the mean from the data. We therefore modified the `MATLAB`'s covariance routine to take the mean