Wavelet and Tomographic Reconstruction of Nonlinear Dynamics

David Allingham, B.Sc. (Hons)

This thesis is presented for the degree of
Doctor of Philosophy
of The University of Western Australia
Department of Mathematics & Statistics.
2002
And after a time, the boiling sea of blood and all the lopped and all the hacked-up humanity that swam within it drained from mah head, and from it rose a pillar of chaogenous calculus, cold and hard. And some serious weighing up of terms ensued. Yes, there, supine beneath a bold and brazen sun, ah struggled with some pretty eternal, some pretty adult problems. Listen.

Nick Cave

*And the Ass Saw the Angel*

Let \( X = X \).

Laurie Anderson

*Big Science*
Abstract

We began by using wavelets as a tool to model the dynamics of nonlinear dynamical systems. Due to the nature of dynamical systems data the standard wavelet transforms were not effective, and so a method was developed which fitted dictionaries of wavelet functions to the data, which incorporated an optimization step to ensure goodness of fit.

Following this we investigated modelling the distribution of data points in order to estimate the average mutual information of a system. Again, standard procedures proved unsuitable for our purposes, and a new method was developed from tomographic principles.

Being in possession of trusted one-dimensional density estimators (in this case the adaptive kernel estimator of Silverman), we wished somehow to lift these up into higher dimensions. This was achieved through the use of tomography, whereby many one-dimensional projections were calculated from projections of our data and kernel density estimators built from each of these. The main tool of tomographic reconstruction, the (inverse) Radon transform, was then used to reconstruct higher-dimensional probability density estimates.

From these we were able to perform simulations of the system as well as calculate important invariants such as Lyapunov exponents, entropy and mutual information. The technique was applied with success to both artificial and experimental data.
Contents

Abstract iii

Acknowledgements vii

1 Introduction 1

2 Preliminaries 7

  2.1 Embedding ............................................................... 7
  2.2 Description length ...................................................... 10
  2.3 Algorithmic probability ................................................ 13
  2.4 Data sets ................................................................. 14
    2.4.1 Artificial data – the Hénon map .............................. 14
    2.4.2 Experimental data – forced oscillating string ............ 15
    2.4.3 Experimental data – squid giant axons .................... 15

3 Wavelet Models 19

  3.1 Dictionary Wavelet Models ........................................... 19
  3.2 Optimized Wavelet Models ............................................ 26
  3.3 Noisy Hénon models .................................................. 31
  3.4 Modelling of a Forced Oscillating String ....................... 35

4 Applications of Probability Density Models 39

  4.1 Average mutual information ....................................... 39
  4.2 Estimating Lyapunov exponents .................................... 41
4.3 Simulation .................................................. 42

5 Density Estimation Models .................................. 45
  5.1 Adaptive partitioning estimator ......................... 46
    5.1.1 Perturbed partitions ................................ 49
  5.2 Adaptive kernel estimator ............................... 52
    5.2.1 Perturbed kernel estimators ....................... 56
  5.3 Tomographic imaging techniques ......................... 58
  5.4 The Radon transform and its inverse ................... 61

6 Tomographic Density Estimation .......................... 67
  6.1 Tomographic density reconstruction ..................... 67
  6.2 Constraints ............................................ 72
    6.2.1 Results using positivity constraints ............... 74
  6.3 Weighted optimization .................................. 77
    6.3.1 Distribution assessment ........................... 81
  6.4 Experimental data results ................................ 83
  6.5 Further work .......................................... 85

7 Conclusion .................................................. 89

Bibliography .................................................. 93
Acknowledgements

Many thanks must go to Alistair Mees for his excellent supervision, guidance and support over the past four years, and to Kevin Judd for stepping into the breach when Alistair was absent.

Thanks are also due to Matthew West and Devin Kilminster with whom I collaborated and without whose programming skills I would be still have been here in another four years. Matt wrote most of the programmes to implement the wavelet modelling method.

Of course, none of this would have been possible without Gina and my parents’ support and encouragement. To them all I owe a deep debt of gratitude.
CHAPTER 1

Introduction

Over the past 20 years, interest in nonlinear dynamical systems has sprung up from a variety of sources, due to the prevalence of such systems in fields such as mechanical and electronic engineering ([40] [45] [50]), medicine ([25] [27] [39]), mathematics ([30]), astronomy ([37] [70]), chemistry ([60]), biology ([26] [46]) and many disciplines of physics ([10] [42] [57]). Nonlinear dynamics have even been applied, however dubiously, to literary studies (see [67]).

The study of nonlinear dynamical systems, whilst not in its infancy by any means, is relatively new, and as is the case for any scientific discipline, new or not, uses are still being found for algorithms and techniques from other fields. In this thesis we examine the application of two such techniques to the modelling of experimental data from nonlinear dynamical systems. Original algorithms are presented for directly modelling the dynamics of time-series data and for modelling the spatial distribution of such data.

We were primarily concerned with producing useful models of time-series data when no theoretical model was available. We also wished to focus on reasonably small data sets, usually a few hundred points, and at most perhaps one or two thousand. In this thesis we are only concerned with "embedded" data. Here, a one-dimensional measurement of some value of the system is taken over time, and values of that measurement from various times are combined to form a higher-dimensional vector of points. This process is described in Section 2.1. When a dynamical system operating in a chaotic, that is, non-periodic and deterministic, regime is involved, these embedded points lie on a "strange attractor," a set consisting of all the possible states of the system. Examples of these are given in Section 2.4, which describes the data sets that we examined.

Once we have a model of the system under investigation, we can use it for several things. One is prediction; that is, given a particular state of the system,
we would like to know what will happen next. Due to the chaotic nature of the systems which we are interested in, this is impossible to achieve except for very short lengths of time. Also, since we must work with finite machine precision and finitely precise measurements of experimental data at the collection point, very small errors occur in defining the state of the system. In chaotic systems, points which start close together diverge very quickly, and so what our model says will happen given a particular starting condition quickly diverges from what happens in the actual system. Depending on the particular time-scales involved, predictions accurate for only, say, 5 steps ahead may be quite useful. However, this is not usually the case, and consequently we concern ourselves more with the qualitative nature of future behaviour, rather than quantitative. Some of this information can be obtained from the model itself, some from simulation. Given some starting values, perhaps setting the system in a state in which we are interested, we can generate long trajectories from the model.

The first tool which we seconded for our modelling purposes was wavelet decomposition. The term “wavelet” first began appearing in its current context in the early 1980s, referring to mutually orthogonal functions, often with compact support, used in a then new form of series expansion. This work began in a signal-processing setting, but quickly expanded into other fields as the compression properties of the wavelet transform became apparent. Most of the early applications of wavelets were in the field of geophysics, where field-studies produce enormous quantities of data, on the order of terabytes, in very short periods of time. Wavelet techniques can be used to compress the data, and are still in wide use (see, for example, [19] and [62]).

Due to the non-uniformly distributed nature of our data sets, we were unable to use the standard wavelet transforms to produce a series decomposition of the system. Instead, we used function series techniques already developed for other function classes, such as radial basis functions (see, for example, [13] or [31]). This work is presented in Chapter 3. Description length (see Section 2.2) was used to
determine the number of functions in the model. The results of this modelling technique were unsatisfactory, as the models produced had no generalization away from the data. To correct this, an optimization step was added which adjusted the wavelet parameters (see Section 3.2). Radial basis models use radially-symmetric, non-compact functions, which include parameters for position (centre) and width (radius), and wavelet functions have similar parameters, position and scaling. The addition of this step produced much better results, with good generalization. In standard wavelet transforms, scalings are in negative powers of 2, and positioned so that wavelets of the same scale are non-overlapping. We hoped that the compact, or close to compact, support of wavelet functions would result in improved models over radial basis techniques. Artificial data with added noise and experimental data (see Sections 3.3 and 3.4 respectively) were modelled successfully. This modelling method, using only wavelet functions, is original.

We were attempting to model the governing equations of the underlying dynamical system directly, and to produce deterministic models, rather than stochastic ones. For a deterministic model, a particular starting condition should evolve identically each time the model is run, although in practice rapid divergence often occurs, since we are dealing with chaotic systems. We cannot achieve the infinite precision which is required for exact determinism. Stochastic models, are based on probabilities, and values of the model evolve according to some distribution. Our next modelling technique involved such models.

The second technique to be employed was the principle of tomography. Tomography is very prevalent today in diagnostic medicine, but traces its origin back to the work of Radon around the time of World War I (see [55]). It refers to the reconstruction of functions from their projections, and the “development of computer assisted tomography” won the 1979 Nobel Prize in Physiology or Medicine for Cormack and Hounsfield [66].
The development of our tomographic density estimator is described in full in Chapters 5 and 6. We computed one-dimensional projections of the data under investigation using a known and trusted estimator, Silverman’s adaptive kernel estimator (see Section 5.2), and then applied tomography theory to obtain a reconstruction of the higher-dimensional distribution of the data. These results contained many artefacts, including negative values, which obviously should not appear in a probability density function. To remove these, we converted the problem into a quadratic program (see Section 6.2) and added positivity constraints. The addition of these constraints eliminated the negative values, but to remove the most stubborn of the artefacts, weightings were added to the one-dimensional kernel values (see Section 6.3). The application of tomographic techniques to modelling distributions of dynamical systems is original work.

Our initial aim in developing models of probability densities was to provide us with error estimates on calculations of average mutual information, described in Section 4.1. This statistic can be used to help determine lags for embedding the original one-dimensional time-series data, and we wished to check if the lags were able to be chosen more flexibly. Once we had established our model, we were able to use it to estimate other statistics, and we were particularly interested in calculating Lyapunov exponents. This is usually done by computing the divergence of points belonging to very long trajectories of a dynamical system, sometimes consisting of millions of points; these are temporal estimates of the Lyapunov exponents. Since we were restricting ourselves to small data sets, we wanted to calculate spatial estimates – knowledge of the distribution of the data is required for this, and was provided by our technique.

Following this, we explored further the applications of having high-dimensional probability density functions available to us, although lack of computing power was unfortunately a limiting factor here. In low-dimensional cases, however, we were able to use the densities as stochastic models of the data. Further improvements
which could be made are discussed in Section 6.5.
CHAPTER 2

Preliminaries

In this section we discuss some of the foundations of the study of nonlinear dynamical systems in general and of our techniques for the modelling of these in particular. We first describe the concept of time-lag embedding, a principle which allows us to reconstruct higher-dimensional data sets from one-dimensional measurements of systems. Next, we give details of description length, a method of assessing our models, and how we calculated it. Algorithmic probability is then described, which, coupled with description length, we hoped would enable us to give error estimates for statistics calculated from our models. Finally, the data sets that we examined in detail are described.

2.1 Embedding

In the investigation of time-series from nonlinear dynamical systems, we frequently use the technique of time-lag embedding. This approach is used because we believe that observations in the time-series have come from a system with dimension greater than one.

Suppose that the system under investigation is an unknown function \( f(x) \) from a space \( \mathbb{R}^d \) to itself, and that we are able only to observe this via a one-dimensional observation function \( g \). Also suppose that there is dynamical and observational noise present, denoted by \( \zeta_t \) and \( \eta_t \) respectively. That is, we have

\[
x_{t+1} = f(x_t + \zeta_t), \tag{2.1.1}
\]
\[
y_t = g(x_t) + \eta_t, \tag{2.1.2}
\]

and the only values which are known are the \( y_t \)'s. In order to model the system, we require a \( d \)-dimensional data set, which Taken's theorem says that we are able to reconstruct from the one-dimensional measurements \( y_t \) (see [64]). Let \( Y_t \in \mathbb{R}^D \) be the points in this reconstructed time-series. Taken's theorem only applies in the
noise-free case, but it is in widespread and successful use in the presence of noise, and we use it as justification for embedding our experimental data.

If our observed data comprises $N$ observations, $\{y_t\}_{t=1}^N$, then we will take $Y_t$ to be

$$Y_t = (y_t, y_{t-L}, \ldots, y_{k-(D-1)L})$$

(2.1.3)

for a uniform embedding with time-lag $L$, or

$$Y_t = (y_t, y_{t-L_1}, \ldots, y_{k-L_{D-1}})$$

(2.1.4)

for a more general, nonuniform, embedding with time-lags $L_1, \ldots, L_{D-1}$. The two parameters which we need in order to determine $Y_t$ are the embedding dimension $D$ and lags $L_k$. Usually, $D = d$, although there may be cases where this does not hold, such as if our modelling method is not sufficiently good ($D > d$), or if the data that we are dealing with is locally of a lower dimension than the system as a whole, for example co-planar points of a system in $\mathbb{R}^3$ ($D < d$).

The embedding lag $L$ can be chosen in a number of ways, and often a range of values are suitable. One simple method is to examine the data by eye and choose a lag corresponding to roughly one-quarter of the apparent "period" of the data. This "period" need not be a true periodicity in the data – it is more representative of time-scales of interest present in the data. Another method is to calculate the correlation of the data set $y_t$ with a time-lag delayed version of itself, $y_{t-L}$ (see, for example, [17] or [36]). This is known as the auto-correlation of the data and is given by

$$c_L(y) = \frac{\langle y_t y_{t-L} \rangle - \langle y \rangle^2}{\sigma^2},$$

(2.1.5)

where $\langle y \rangle$ and $\sigma^2$ are the time average and variance of $y$ respectively. The embedding lag is then chosen to be the first minimum of $c_L$. A related technique, and the one that we used, is to take the first minimum of the average mutual information between $y_t$ and $y_{t-L}$. See Section 4.1 for details of average mutual information. Informally,
the reason for using this value is that we are using points which together provide us with the maximum information about the system at the time $t$.

All of the described methods result in similar lags for uniform embeddings. For data in which there are a number of different time-scales present, non-uniform embeddings can be used, and the individual lags can be calculated using these methods (see [33]). Uniform embeddings were used on all the data sets in this thesis, although the work is in no way dependent upon this.

A common technique for deciding what the embedding dimension $D$ should be is to look at the number of "false nearest neighbours" present in embedded versions of the one-dimensional data (see, for example, [1], [3] or [36]). These are points which are neighbours when embedded in $\mathbb{R}^{k-1}$ but not when embedded in $\mathbb{R}^k$. This happens because in dimensions lower than $D$, points which are not actually close together become so under projection. As we increase the embedding dimension, the attractor is unfolded, and points move away from each other, and the number of false neighbours decreases until it drops to zero. $D$ is then chosen to be the first value of $k$ for which this happens.

We were interested in checking how good our estimate of the position of this minimum was, and so sought error bounds on the values of $I$. If the error bars were large, then it would be unclear which lag is most suitable for embedding the data. This might indicate a range of lags which we could use, rather than identifying a single value. This did not prove to be the case, but lead us to a new method for modelling dynamical systems data, described in Chapter 6.

Once we have embedded the data, we are ready to construct models of the system. For the wavelets method we needed to determine how many wavelet functions to include in the model, and for the tomographic method we needed to construct a one-dimensional kernel estimate. For these problems, the principle of "minimum description length" provided a solution.
2.2 Description length

Description length is the number of bits needed to be sent by a particular model of some data in order that the receiver is able to reproduce the data, to some chosen precision. It expresses a trade-off between model size and modelling error. Our reasons for wanting to calculate description length are two-fold. Firstly, we believe that models with shorter description lengths are in some sense better than ones with large description lengths, and so we need some way of determining the ideal size of our models. Secondly, we wish to be able to compare the performance of different models, even different types of models, of the same data set. Minimizing description length can be thought of as finding the smallest "good" representation for the data under study, and our models can be thought of as compressed versions of the data. In this way, description length provides us with a way of comparing different models to see which performs best using this criterion.

We followed Rissanen's idea of parsimony in building our models, which maintains that models with shorter description lengths are preferable to those with large description lengths (see [56]). This in itself is an expression of Occam's Razor which says that the simplest explanation (model) for an event (the data) is the most likely.

Description length calculations are an approximation to various information theoretic criteria such as Akaike's information criterion (see [4]), Schwarz's information criterion (see [58]), or, more generally, stochastic complexity. This expresses an ideal parametrization or partitioning for a data set. Stone has shown that cross-validation (from Equation (2.1.5)) is asymptotically equivalent to Akaike's information criterion (see [63]). We did not calculate any of these criteria directly, due to this asymptotic behaviour, and instead approximated them using description length, which, in fact, is also an approximation. We could make an allowance for the size of "pre-code" (such as an emulator, if required), however factors such as this are constant for all of our models and we can safely ignore them.
2.2. Description length

We used two different forms of description length. The first was used for parametric models in which we needed to determine the ideal number of parameters to include. The second was more general, and was used to compare models of different types.

When we are constructing parametric models, we need to address the issue of model size. Maximum likelihood methods have a known tendency to over-parametrize (see, for example, [51]), and so we turned to the technique of minimizing description length to determine the size of our models. A penalty term was included for each parameter that is added to the model, and we also assessed how well the model fitted the data, as in the case of maximum likelihood calculations.

This approach seeks a balance between modelling error and model size. We calculated the description length for our parametric models by using

\[
\text{DL(model)} = - \sum \log e_i + k \times b, \tag{2.2.1}
\]

where \(e_i\) is the prediction error of the \(i^{th}\) point, \(k\) is the number of parameters and \(b\) is the accuracy to which each parameter is specified, in bits. This equation is in fact correct up to a constant term which we ignored, as our different models were produced by the same code. It would be possible to have a large range of models to choose from which are decided beforehand, and then a few bits used to express which model was chosen. The description length for a particular model is likewise independent of implementation. This will work well for reasonably similar models. However, it does not allow a general comparison of models to take place.

In later work, it was necessary to calculate the description length accurately enough that we were able to compare it for quite different methods, and this was done using predictive encoding.

Predictive encoding can be used to compare description lengths for any algorithms which model the distribution of the data. This encoding is based upon a
look-up table of probabilities for each data point. The more probable a particular point is, the shorter the encoding needed to transmit it becomes using a prefix-free code such as a Huffman code (see, for example, [28]). A prefix-free code is one in which no codeword is a prefix of another, even though the codewords may be of different lengths. Often the look-up table will be an abbreviated version of the data – that is, it will contain probabilities for a particular partition which covers the data set. The location of the partition that a point is in is sent, followed by the “fine-tuning” needed to place the point exactly within its partition. Alternatively, a large number of partitions can be chosen in the first place, truncating the data to a certain precision.

Suppose that we have $N$ data points in $\mathbb{R}^D$ and that each is known to $b$ bits precision. Predictive encoding works by first sending the first $N_0 << N$ data points to full accuracy, so that the receiver can build an initial probability table. After that, the length of code transmitted for each point $x_i$ is

$$-\log_2 \left( 2^{-bD} P(x_i) \right) + e_i,$$ (2.2.2)

where $e_i$ is the “adjustment” that we need to make to get from our transmitted partition to the true point.

Points which occur more often, or which lie regions of the space which are visited more often, will have higher probabilities in the table and can be transmitted using fewer bits than “unlikely” points. As each point is sent, the probabilities in the table are updated.

To avoid potential problems with divisions by zero, it may be desirable to allocate a small, rather than zero, probability density to all regions of otherwise zero probability. If the value assigned to such regions is on the order of $1/N$, this can be thought of as adding one point to the data set and smearing it out over the entire space. In practice, this results in a bit cost for such points of $-\log_2 N^{-1} = \log_2 N$. This is large if we are dealing with a large data set, and a better approach would be
to send such points to full accuracy, at a cost of \( bD \) bits. Both of these methods were tried and work in practice. Results presented in this thesis use the latter method.

Using this method of calculating description length enables very different methods of estimating the distribution of the data to be directly compared. The method being used (for example, a partitioning method, or a adaptive kernel estimator) simply determines the entries in the look-up table. The main drawback is that if the data set is very large, it can be very time-consuming to re-run the chosen algorithm for each point that is being transmitted. We found that good results could be obtained for large data sets by using only the first \( N_{\text{max}} \) of the points available for calculating description length. We generally used 2000 points for this purpose for data sets larger than this. This restriction applies only to the points used in constructing the model. Once this number of points have been sent, the model of the distribution is considered fixed, and the bit cost of sending additional points \( x^+ \) is then simply \( -\log_2 P(x^+|x_1, \ldots, x_{N_{\text{max}}}) \).

Once we had calculated the description length for a particular model, we then needed to compare statistics calculated using those models. A method of weighting different models using description length was developed by Solomonoff.

### 2.3 Algorithmic probability

The work of Solomonoff on algorithmic probability (see [61]) provided us with a way of weighting different models that were constructed of the same data set. We could calculate various statistics for each of these models and then obtain a probability distribution for those statistic. Suppose that we wished to find a distribution for some statistic \( S \). The probability assigned to each model is given by

\[
P_{\text{Sol}}(\text{model}) = 2^{-L(\text{model})},
\]

where \( L(\text{model}) \) is the description length of the model being used.
One of the advantages of this approach is that we can include in our calculations any model that we build, even bad models with high description lengths. In fact, if we are careful about how we calculate description length, we can include models built in radically different ways. We did this using predictive encoding.

As we will only ever have a finite, but possibly large, number of models, which are representative of the system under examination, we scale the $P_{\text{Sol}}$'s so that they sum to one, and will denote these normalized probabilities by $P_N$. Once we have computed the probabilities for our different models, we let $P_N^{(i)}$ be the normalized Solomonoff probability for the $i$th model. The weighted mean and variance of $S$ are then given by

$$
\mu(S) = \sum_i P_N^{(i)} S_i \quad (2.3.2)
$$

and

$$
\sigma^2(S) = \left( \sum_i P_N^{(i)} S_i^2 - \mu^2(S_i) \right). \quad (2.3.3)
$$

2.4 Data sets

We applied our methods to three data sets. The first was an artificial nonlinear dynamical system, the well-known Hénon map, whilst the other two were experimental data sets from vibrating string and squid giant axon experiments.

2.4.1 Artificial data – the Hénon map

Often we will refer to this well-known and relatively simple map, given by

$$(x, y) \mapsto (1 + y - 1.4x^2, 0.3x). \quad (2.4.1)$$

The parameters 1.4 and 0.3 are chosen to place us within a chaotic regime of this map. 500 time-lag embedded values from the $x$ co-ordinate were scaled to the interval $[-0.5, 0.5]^2$, and are shown in Figure 2.4.1. The scaling was performed for programming convenience and does not alter the validity of our results.
2.4. Data sets

2.4.1 Experimental data – forced oscillating string We examined data taken from measurements of a vibrating wire collected by Molteno and Tufillaro (see [49]). The experiment involved the driven oscillation of a short segment of guitar string, and the resulting amplitude of oscillation was measured.

This data was sampled from two minutes of string vibration, and starts out in a chaotic regime. The system is non-stationary, that is, it involves parameters whose values are not constant with time. This is due largely to the heating the string experienced as the experiment progressed, and ultimately the oscillation settled down to a periodic orbit – the 200 points of data which we used were taken from the chaotic regime.

2.4.3 Experimental data – squid giant axons We examined voltage data from squid giant axons forced periodically by a voltage clamping electrode (see [46]
Figure 2.4.2: We examined 200 points of data from the vibrating string system, which form the orbits shown here.

for discussion of this data). We modified the data slightly to make it one-dimensional, in the following way.

An apparent transient was removed from the start of the data set, and then the data was “unfolded” to remove the cross-over that occurs in the upper right-hand region. The unfolded data set \( \{ y_k \} \) is given by

\[
y_k = \begin{cases} 
  x_{k+1} & \text{if } x_k < 0.45, \\
  x_{k+1} + 1 & \text{otherwise},
\end{cases}
\]

where \( \{ x_k \} \) is the original data. This transform is reversible since \( x_{k+1} = y_k \pmod{1} \).

We also noted that there were large intervals of \( y \) in which there were no data points. These would be assigned zero probability and so were removed to permit a smaller grid size to be used in order to speed up computation. The removed intervals were \([.81,.96] \) and \([1.31,1.55] \). Again, this is reversible, as these intervals can be replaced in the final density estimate. As with the Hénon map data, the final data set was scaled to \([-0.5,0.5]^2 \). The original data is shown in Figure 2.4.3, with
2.4. Data sets

Figure 2.4.3: The unaltered embedded squid giant neuron data.

the modified 400 points of data shown in Figure 2.4.4.
Figure 2.4.4: The 400 points of modified embedded squid giant neuron data.
Wavelet Models

Our approach to building models is founded on the idea of parsimony, as described in Section 2.2, and so we sought models which express the data very concisely. For compressing data via basis decomposition, Donoho has shown that unconditional bases, also called frames, perform optimally (see [18]). A frame is a spanning set of basis functions \( \{v_i\} \) for which there exists a constant \( C \) such that

\[
\sum_i |\langle f, v_i \rangle|^2 \leq C \|f\|^2 \tag{3.0.1}
\]

for all functions \( f \) in the space, where \( \langle .., .. \rangle \) is the inner product operator. Wavelets form an unconditional basis (see [48]) and so are an appealing technique for modelling dynamical systems.

3.1 Dictionary Wavelet Models

We used a dictionary model scheme, similar to other methods for dynamical systems modelling such as radial basis functions. A function \( f \) was represented by the superposition of a set of elements chosen from some large dictionary \( D \):

\[
f = \sum c_i d_i \tag{3.1.1}
\]

with coefficients \( c_i \in \mathbb{R} \) and \( d_i \in D \). We approximated the function by using a finite number of dictionary elements, so that

\[
f = \sum_{i \in I} c_i d_i + R_n \tag{3.1.2}
\]

where \( R_n \) is the residual. Our modelling problem was to chose the subset \( I \). A dictionary for which any function has multiple representations in terms of the dictionary elements is called over-complete. This contrasts with a complete dictionary in which a given function has a unique representation.

\footnote{Portions of the work in this chapter have appeared in [7], [8] and [9].}
For our modelling we used a dictionary composed of tensor product wavelet functions, but our conclusions also apply to more general classes. The elements of our wavelet dictionaries were of the form

\[ d_{a,b} \left( \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \right) = \prod_{i=1}^{n} \psi \left( \frac{x_i - b_i}{a_i} \right), \]  

(3.1.3)

where \( a_i, b_i \in \mathbb{R} \) are constants specifying \( d_{a,b} \) and \( \psi : \mathbb{R} \to \mathbb{R} \) is the chosen wavelet function, and was fixed for the dictionary. Typically \( \psi \) was chosen so that

\[ \int \psi(x) dx = 0. \]  

(3.1.4)

For signal processing applications, a highly oscillatory \( \psi \) is often chosen because the signals themselves are of this nature when seen as functions of time. For the modelling of dynamical systems however, we were interested in smooth interpolation on higher-dimensional spaces and we have found that better results are obtained using a \( \psi \) with relatively low curvature. We believe that the systems we modelled were smooth functions and so smooth wavelet functions were used. All the results presented here used the wavelet

\[ \psi(x) = (8x^2 - 1)e^{-4x^2}, \]  

(3.1.5)

shown in Figure 3.1.1. This is the Morlet wavelet (see, for example, [15] or [34]).

A point of significance concerns the coverage of the function domain. In signal processing, the dimension of the function domain is the same as the dimension of the space; this means that the data is not sparse on the region of the space that it occupies, and points are uniformly spread over time. For dynamical systems, however, the dimension \( D_{\text{att}} \) of the attractor which we are trying to model is usually less than the dimension \( D_{\text{emb}} \) of the space in which it lies. Typically it is the case that

\[ D_{\text{emb}} - 1 < D_{\text{att}} < D_{\text{emb}}, \]  

(3.1.6)
and the data points are scattered in a non-uniform way throughout that space. This sparseness means that the model is not very constrained, and so model trajectories can wander far from the true attractor, if the model is not carefully constructed.

For appropriately chosen sets of \(a\)'s and \(b\)'s the dictionary \(D = \{d_{a,b}\}\) is orthogonal, and so

\[
c_{a,b} = \frac{\langle f, d_{a,b} \rangle}{\langle d_{a,b}, d_{a,b} \rangle}^{\frac{1}{2}}. \tag{3.1.7}
\]

If the domain of \(f\) is discrete and \(f\) is defined on a uniformly spaced set of points, as is the case in signal processing, then this decomposition is known as the discrete wavelet transform (DWT) (see, for example, [15]). This transform is given by

\[
f_{N+1}(x_i) = \sum_{j=-N}^{N} \sum_{k=-\infty}^{\infty} b_{jk} \psi_{jk}(x_i), \tag{3.1.8}
\]

where \(f_{N+1}(x_i)\) is the approximation to \(f(x_i)\), defined at the uniformly spaced points \(x_i \in \mathbb{R}\), \(N\) is the number of scales used and the coefficients \(b_{jk}\) are defined by

\[
b_{jk} = \int_{-\infty}^{\infty} f(x_i) \psi_{jk}(x_i) dx. \tag{3.1.9}
\]
Wavelet compression involves the removal of coefficients smaller than some chosen threshold (see, for example, [16]); compression ratios of 20 : 1 are common. If we could use this transform on our data sets, we would seek to use description length to pick the threshold level and so produce a model of the data.

For modelling dynamical systems, we used time-lag embedding to define a finite portion of a trajectory in $\mathbb{R}^n$, and thus our function had a finite domain. Our data points were not, however, uniformly spaced on this domain and so standard wavelet dictionaries were no longer orthogonal, and the DWT could not be used. In fact, decomposition of the data using Equation (3.1.2) was no longer unique and the problem became one of selecting a suitable subset of the basis.

The work of both Donoho and Rissanen (see [18] and [56] respectively) suggests that the best models will be those containing relatively few basis elements. This was discussed in Section 2.2. For this reason we sought a decomposition which had few large coefficients. This is similar to the way that wavelet compression works, and if we had been able to perform the DWT, we would proceed at this step by choosing a threshold below which wavelet coefficients were discarded, by minimizing the model’s description length.

A well-known technique for building such small models is orthogonal matching pursuit (OMP), also called greedy selection (see, for example, [43] or [53]).

The OMP algorithm builds a model by iteratively adding dictionary elements which are not already included, according to description length. The element chosen for inclusion will be the one which gives the best improvement in description length. This process continued until a minimum description length was found. For this to happen, we allowed elements to be added to the model which actually increased the description length. That is, if no element was found which decreased the description length, we included the element which gave the smallest increase. Once the description length increased for a certain number of iterations, we declared
the last minimum to be our target minimum. This helped avoid local minima, and we allowed 10 steps of increase before declaring the stopping condition met. This technique produces better models when presented with over-complete dictionaries. We also expected that the functions we were modelling did not have unique representations in terms of the wavelet functions used. For these reasons, over-complete wavelet dictionaries were used.

There are a number of more elaborate techniques for basis selection, including the swapping method of Judd and Mees (see [31]) and the basis pursuit method of Chen, et al. (see [14]). Both of these require more complicated implementation than the OMP method. We investigated the performance of the Judd and Mees algorithm and found that it did not substantially alter the conclusions we arrived at. We did not believe that other basis selection methods would have a significant impact either, and so for reasons of computational efficiency the results presented here were obtained using orthogonal matching pursuit.

We modelled the Hénon map from Section 2.4.1 by selecting a subset of our wavelet dictionary. For these models we were interested in both one-step prediction and free-run simulation. As a quantitative measure of the performance of the model we used cross-validation. This involves calculating the prediction error on data from the system which was not used during model fitting. This is an approximation of the generalization error for the model and as such we expected the model with the smallest cross-validation error to give the best short-term free-run behaviour.

The results of fitting a wavelet model to 200 points of the Hénon attractor are shown in Figures 3.1.3 and 3.1.4. This model selected vectors from an over-complete dictionary of size 6400. The cross-validation error calculated from 10000 data points is plotted against the model size (the number of wavelets chosen) in Figure 3.1.2. From this plot it can be seen that the best model contained 108 wavelets. The difference between this model and the true Hénon function is plotted in Figure 3.1.4.
Chapter 3. Wavelet Models

The position of the data points to which the model was fitted are shown in this figure by green circles. From this we see that the model surface was extremely accurate on the data points but diverged quickly from the true function elsewhere, even in the region bounded by the attractor.

Despite this, the model still exhibited stable long-term free-run behaviour, with its attractor shown in Figure 3.1.3. From this we can see that the model attractor diverges prominently from the true attractor in several regions. Two such regions are enlarged in Figures 3.1.3(b) and 3.1.3(c). These show that the model coincided very accurately with the true system at the points on which the model was fitted, but had inaccuracies elsewhere.

To quantify the short-term predictive capabilities of this model we calculated the multi-step cross-validation error. This was done by taking a large number of points that were not used for constructing the model, and then iterating the model for a number of steps, giving us the predicted values of the system according to

Figure 3.1.2: Cross-validation error versus model size for OMP-selected models.
Figure 3.1.3: (a) shows the attractor for the OMP-selected model of size 108; (b) and (c) show details of regions 1 and 2 respectively. The true attractor is the thin line in (b) and (c), and the squares show the data points used for fitting. Note how the model attractor passes directly through each data point but may not behave well elsewhere.
our model. These predicted points were compared with the true values of the map, and the sum of squares error calculated. The multi-step cross-validation errors for our model are shown in Figure 3.1.5. This plot shows that there was little useful predictability beyond 10 steps.

An improvement was needed which improved these models’ generalization characteristics, and so we examined the way in which the model parameters were chosen.

3.2 Optimized Wavelet Models

In order to correct the problems of these wavelet models, we investigated continuous optimization of the parameters starting from some initial model. To perform least-squares optimization we used Levenberg-Marquardt optimization (LMO) (see, for example, [22] or [44]) and Nelder and Mead’s downhill simplex method (see, for
3.2. Optimized Wavelet Models

example, [54]), both standard optimizers with good performance in a wide range of problems. Occasionally Levenberg-Marquardt optimization would be extremely slow in finding its way off flat sections of the error surface. In this event we allowed the simplex algorithm to take over, later resuming the Levenberg-Marquardt technique because the simplex method was very slow to converge.

The use of these optimizers, however, was particularly sensitive to the choice of initial model. Given random starting conditions, LMO would usually become trapped in a local minimum of the error surface. This problem worsened as the number of wavelets in the model increased, with it being virtually impossible to obtain a good model using more than 5 wavelets. Although a reasonable model of Hénon can be built using only 5 wavelets, this approach clearly would not generalize well to more complex systems. When successful models were built with the added optimization step, however, they used very few wavelets and had lower cross-validation errors than the basis-selected models in Section 3.1.
To overcome this problem of sensitivity to initial conditions, we initialized the optimization with a model built by the basis-selection techniques outlined in Section 3.1. This hybrid method did in fact produce much better models. A plot of cross-validation errors versus model size is shown in Figure 3.2.1. This shows that the optimal model size was much smaller for this technique than when we used the greedy algorithm from Section 3.1 without LMO. From Figure 3.2.1 we see that the optimal model contained 15 wavelets.

The error surface for the model containing 15 wavelets is shown in Figure 3.2.2. Comparing this to Figure 3.1.4 we can clearly see the improved generalization characteristics of the optimized model. This was particularly reflected by the extended "plateau" surrounding the attractor. This improvement was also reflected in the free-run behaviour, both in the short-term, as demonstrated by the multi-step cross-validation, and in the long-term attractor. These are shown in Figures 3.2.3 and 3.2.4.
Figure 3.2.2: Error surface for the LMO model of size 15. Green circles show the data points used for fitting the model. Here the improved generalization of this model can be observed, with the accurate part of the model extending away from the attractor.

Figure 3.2.3: Multi-step cross-validation error for the LMO model of size 15. Comparing this to Figure 3.1.5 it is clear that this model had much improved free-run prediction capabilities.
Figure 3.2.4: The reconstructed attractor for the LMO model of size 15. Comparing this to Figure 3.1.3, we see that this attractor is much better than that built using our OMP selection method. It deviates from the true attractor by no more than $10^{-5}$. 
respectively. Comparing the first of these with Figure 3.1.5, we can see that this model had approximately twice as many steps of accurate short-term prediction as the previous technique. The model attractor shown in Figure 3.2.4 is virtually identical to the true attractor, deviating by no more than $10^{-5}$, demonstrating the success of our hybrid basis-selection/Levenberg-Marquardt technique.

### 3.3 Noisy Hénon models

In Sections 3.1 and 3.2 we described two approaches to modelling the dynamics of nonlinear systems. The data that we used to illustrate these methods was clean, taken from a mathematically generated system. This is very unlike any real data which might be encountered. In this section we show the results of modelling the Hénon attractor with added observational noise. We expected that this would cause our selection methods to choose smaller models which did not perform as well as the models for the clean Hénon data.

Plots of cross-validation error against model size for our OMP technique are shown in Figure 3.3.1 for various levels of noise. It can be seen that the optimal number of wavelets decreased as more noise was added. For Levenberg-Marquardt optimization with 3% noise the cross-validation plot is shown in Figure 3.3.2, from which it can be seen that the optimum model size occurs earlier and that the cross-validation error for that model is much greater than for the noise-free model (as shown in Figure 3.2.1). We chose the minimum model sizes for 3% noise to more fully compare the two techniques. For the OMP method, the minimum model size was 32, whilst for Levenberg-Marquardt optimization it was 6. The true attractor with noise is shown in Figure 3.3.3.

We compared these two models by considering multi-step cross-validation and the attractor geometry. For OMP, the model built for the noisy attractor was quite bad. It is shown in Figure 3.3.4. This model did not compare very well with the true
Chapter 3. Wavelet Models

Figure 3.3.1: Cross-validation error versus model size for OMP models with 1%, 3% and 5% additive observational noise.

Figure 3.3.2: Cross-validation error versus model size for LMO models with 3% additive observational noise. A lower sum of squares error is achieved, as well as a much smaller model, compared to the OMP models.
3.3. Noisy Hénon models

![Image of the Henon attractor with 3% additive observational noise.]

Figure 3.3.3: The Hénon attractor with 3% additive observational noise.

The attractor shown in Figure 3.3.3. Although some of the same structure was present, it was very distorted. The multi-step cross-validation error for the model is plotted in Figure 3.3.5. Comparing this to Figure 3.1.5 we can see that the model of the noisy data performed very badly, with no real predictive ability beyond about 4 points ahead. Clearly the performance of this modelling technique is much reduced in the presence of noise.

The model built using our hybrid method with 3% noise proved to be better than that built using only OMP. The multi-step cross-validation error is plotted in Figure 3.3.6. The model’s predictive capability extended to about 5 steps ahead, and the attractor closely resembles the true Hénon attractor for about 1000 iterations. After this the model settled down to a periodic orbit. It is clear from the number of transient points for this periodic orbit that the model is only weakly stable, and so a very small amount of dynamic noise would have removed this orbit. To check this, 0.01% dynamic noise was added during the free-run calculations, resulting in
Chapter 3. Wavelet Models

Figure 3.3.4: The attractor for the OMP model of size 32 with 3% additive observational noise.

Figure 3.3.5: Multi-step cross-validation for the OMP model of size 32 with 3% additive observational noise.
3.4 Modelling of a Forced Oscillating String

Ultimately the purpose of any modelling method is to model and study data from real systems. To illustrate such an application with our optimized wavelet basis-

Figure 3.3.6: Multi-step cross-validation for the LMO model of size 6 with 3% additive observational noise. Compared with Figure 3.3.5, we see that there is only a small improvement.

the non-periodic attractor shown in Figure 3.3.7, which very closely resembles the true attractor.

The models built for the noisy Hénon attractor demonstrate the importance of the continuous optimization step in our hybrid technique. Without this the models that were built performed very poorly, whereas the addition of this step allowed us to build quite reasonable models of the noisy system, suggesting that our technique might be able to successfully model some real-world dynamical systems. In the next section our hybrid method is applied to data from a vibrating guitar string.

3.4 Modelling of a Forced Oscillating String

Ultimately the purpose of any modelling method is to model and study data from real systems. To illustrate such an application with our optimized wavelet basis-
Figure 3.3.7: The attractor for the LMO model of size 6. The model was built from the noisy data in Figure 3.3.3, the Hénon map with 3% additive observational noise. This attractor is clearly much better than the one shown in Figure 3.3.4, and very closely resembles the true attractor.
3.4. Modelling of a Forced Oscillating String

selection method we modelled time-series data from the forced oscillating string experiment, described in Section 2.4.2.

In their analysis of this data, Judd and Mees demonstrated that it contains bifurcation structures consistent with a Shil'nikov mechanism (see [32]). This was accomplished by modelling the experimental data and then analysing properties of the model, and to test our modelling method we also adopted this approach, in order that we could compare our results with theirs.

We built a model of the initial chaotic region of the data using our optimized wavelet technique, resulting in a model containing 6 wavelets. This model accurately captured the periodic component of the behaviour which manifests itself in the hoop-shaped attractor seen in Figure 2.4.2. For this reason the multi-step cross-validation remained very low for many periods of free-run simulation.

To demonstrate that a Shil’nikov-type bifurcation might be present in the data it was necessary to calculate the fixed points of our model. The model had 3 fixed points, the locations of which are shown in two-dimensional projection in Figure 2.4.2, along with the trajectory formed by the 200 points of data used to build the model. Two of these fixed points lie far from the data and as such we can make no comment on the reality of their existence. The third fixed point, located at \((-0.71, -0.71, -0.71)\), was in good agreement with fixed point A found in the paper of Judd and Mees, and was in approximately the correct location for a Shil’nikov mechanism.

Calculating the eigenvalues of the linearised system at this point we obtained \(\lambda_1 = 1.5486\) and \(\lambda_2 = -0.0805 \pm 0.1148i\). This implied a stable two-dimensional manifold spiralling into the fixed point and a one-dimensional unstable manifold. This is the same type of fixed point as was found by Judd and Mees. Calculating the value \(\delta = -\log |\text{Re}(\lambda_2)|/\log \lambda_1\) for this fixed point we obtain 5.76. This compares poorly with the value of 0.74 obtained by Judd and Mees, and in fact does not
support the existence of a Shil’nikov mechanism which requires \( \delta < 1 \). Building optimized wavelet models of other segments of the time-series gave extremely similar results. There was always a fixed point very close to \((-0.70, -0.70, -0.70)\) and the eigenvalues of the map at this point differed very little from those above.

This stability of the fixed point and its characterisation, along with the accuracy of our models for free run predictive purposes, suggest that the disagreement of our results with those of Judd and Mees would seem to be due to the type of model we are using rather than the quality of the fit. The radial basis models used by Judd and Mees contain only Gaussian functions with standard deviations on the order of one third of the attractor width. For this reason their model surface will be relatively smooth compared to our wavelet model’s surface, and should provide a more accurate approximation to the true map away from the attractor. Nevertheless, the optimized wavelet model accurately captured many of the important characteristics of the system, justifying the use of this modelling approach for real world data.
Applications of Probability Density Models

Amongst the many applications of having computed the probability density function of a data set, which will be described in Chapters 5 and 6, are estimating Lyapunov exponents, estimating average mutual information, and free-run prediction. Others include finding periodic orbits and calculating various fractal dimensions.

4.1 Average mutual information

Average mutual information (see, for example, [28]), \( I \), is a measure of the information contained in one channel of data, or more generally one random variable, about another, or in several channels about several others. It represents the number of bits that knowledge of channel 2 contributes to knowledge of channel 1, and vice versa, since \( I \) is a symmetrical function, \( I(X,Y) = I(Y,X) \). Average mutual information is commonly expressed in bits, but other units may be used depending on the base of the logarithms in the defining equations given below. Another common unit, for which the natural logarithm is used, is nats.

Probability density function subscripts are omitted in this section as it will be clear from the variables which density is referred to. For example, \( P(x) \) means \( P_X(x) \), whilst \( P(x, y, z) \) means \( P_{XYZ}(x, y, z) \). Also recall that \( P_X(x) = \sum_{y,z} P_{XYZ}(x, y, z) \).

Let \( X, Y \) and \( Z \) be jointly distributed discrete random variables. The average mutual information between \( X \) and \( Y \) is given by

\[
I(X,Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \quad (4.1.1)
\]

\[
= H(X) + H(Y) - H(X,Y), \quad (4.1.2)
\]

where \( H(X) \) is the entropy of channel \( X \),

\[
H(X) = - \sum_x p(x) \log(p(x)), \quad (4.1.3)
\]
and $H(X,Y)$ is the joint entropy between channels $X$ and $Y$,

$$H(X,Y) = -\sum_{x,y} p(x,y) \log(p(x,y)). \quad (4.1.4)$$

The conditional entropy of $Y$ given $X$, $H(Y|X)$, is

$$H(Y|X) = -\sum_{x,y} p(x,y) \log(p(y|x)). \quad (4.1.5)$$

Goldie and Pinch give correspondences between set theoretic and information theoretic concepts are given which allow us to derive an equation for average mutual information in higher dimensions (see [28]). Some of these are

$$H(X) \leftrightarrow \mu(A), \quad (4.1.6)$$

$$H(X,Y) \leftrightarrow \mu(A \cup B), \quad (4.1.7)$$

$$H(Y|X) \leftrightarrow \mu(B \setminus A), \quad (4.1.8)$$

$$I(X,Y) \leftrightarrow \mu(A \cap B), \quad (4.1.9)$$

where $\mu$ is some non-negative additive set function. Using these and Equation (4.1.3), we can derive the average mutual information between $X,Y$ and $Z$.

Consider first that we wish to find $I(X,Y,Z)$, which according to the above correspondences is equal to $\mu(A \cup B \cup C)$, the value of $\mu$ on the intersection of the three sets $A$, $B$ and $C$. We can write this as

$$\mu(A \cup B \cup C) = \mu(A) + \mu(B \setminus A) + \mu(C \setminus A) - \mu(C \cap B) + \mu(A \cap B \cap C), \quad (4.1.10)$$

which is equivalent to

$$H(X,Y,Z) = H(X) + H(Y|X) + H(Z|X) - I(Z,Y) + I(X,Y,Z)$$

$$= H(X) + H(Y|X) + H(Z|X) - H(Y) + H(Y|Z) + I(X,Y,Z),$$

using Equation (4.1.1). Thus we have

$$I(X,Y,Z) = H(X,Y,Z) - H(X) + H(Y) - H(Y|X) - H(Y|Z) - H(Z|X)$$

$$= H(X,Y,Z) - H(X,Y) + H(Y) + H(Z) - H(Y,Z) + H(X) - H(X,Z)$$

$$= H(X,Y,Z) + H(X) + H(Y) + H(Z) - H(X,Y) - H(X,Z) - H(Y,Z),$$
4.2. Estimating Lyapunov exponents

Lyapunov exponents are an expression of the amount of expansion and contraction present in a dynamical system (see, for example, [2] or [72]). If the dimension of a system is \( D \), then there will be \( D \) Lyapunov exponents.

Let the Lyapunov exponents be \( \lambda_1, \lambda_2, \ldots, \lambda_D \), ordered such that \( \lambda_i \geq \lambda_{i+1} \forall i \), where \( D \) is the dimension of the system. For the system to be chaotic, at least one of these \( \lambda \)'s must be negative and one positive. In addition, the sum of the Lyapunov exponents must be negative, indicating that on average the system contracts (see [52]). Positive Lyapunov exponents denote stretching, whilst negative exponents indicate that contraction is occurring, and so for a system to remain bounded and interesting, both of these processes must occur.

One of our reasons for calculating these invariants was to compare values calculated using our techniques to those published elsewhere in order to establish the validity of our methods. We used the following method (see [24]) to estimate only the largest and smallest local Lyapunov exponents. This method uses spatial approximation of the exponents, rather than temporal approximation, which involves the use of extremely long orbits from a system. Once we have these local estimates and a model of the distribution of the data (see Chapter 6) we can estimate the Lyapunov exponents of the whole system.

Let the data set be \( \{x_i\}_{i=1}^n \), with each point \( x_i \in \mathbb{R}^D \); here \( D \) is the embedding dimension. For each point \( x_j, j = 1, \ldots, n - N_{nn} + 1 \), let \( x_f \) be the set consisting of

using Equations (4.1.4) and (4.1.5). Finally, we substitute the definitions of entropy to obtain

\[
I(X, Y, Z) = \sum_{x,y,z} p(x, y, z) \log \left\{ \frac{p(x, y)p(x, z)p(y, z)}{p(x, y, z)p(x)p(y)p(z)} \right\},
\]  

(4.1.11)
since for \( x, \sum_z p(x) = \sum_x p(x) = \sum_{x,y,z} p(x) \), and similarly for \( y \) and \( z \).
the $N_{nn}$ nearest neighbours of $x_j$ (note that $|I| = N_{nn}$ and $I \subset \{1, \ldots, n\}$). We know that $x_i \mapsto x_{i+1}$, and using a least squares approximation for a linear map $f(x_i) = x_{i+1}$, we obtained estimates of $a$ and $b$ for $f(x_{i+1}) = ax_i + b$. We were interested in the stretching that occurs along the unstable manifold, and so we examined the iterations of a randomly chosen unit vector, $u$. Since the system was assumed to be chaotic, as $m \to \infty$ the direction of $f^m(u)$ would rapidly converge to the direction of the unstable manifold. Let $\hat{u}(x)$ be the direction of the unstable manifold at $x$, and let $\rho(x) = |f'(x)\hat{u}(x)|$. Then $\lambda_1 = \int \ln \rho d\mu$. The rapid convergence that occurred in our results is shown later in Figure 6.3.5.

By reversing the order of the data points, we used the same method to estimate the smallest Lyapunov exponent, $\lambda_D$.

Having obtained these values for a two-dimensional system, we also estimated the Hausdorff dimension (see, for example, [11]) using

$$HD = 1 - \frac{\lambda_1}{\lambda_2}. \quad (4.2.1)$$

### 4.3 Simulation

One of our main goals in the modelling of nonlinear dynamical systems is the prediction of future behaviour. Due to the nature of these systems, however, we do not claim to be able to exactly predict trajectories for extended periods of time, and rather we seek to express the general types of behaviour that we would expect to see if the system was observed for a long time. Thus, if our models are good enough, we can produce qualitative information about system behaviour. This comes in the form of simulation, often called "free-run prediction".

In the wavelets work presented in Chapter 3, we were directly modelling the underlying system governing the dynamical system under investigation. Using this approach it is quite straightforward to produce free-run simulations of the data. We have some model $F(x)$ such that $x_{t+1} = F(x_t, x_{t-1}, \ldots, x_{t-D+1})$, where $D$ is the
embedding dimension, and to perform free-run prediction, we simply evaluated this function to obtain

\[
\hat{x}_{n+1} = F(x_n, x_{n-1}, \ldots, x_{n-k+1}), \quad (4.3.1)
\]
\[
\hat{x}_{n+2} = F(\hat{x}_{n+1}, x_n, \ldots, x_{n-k+2}), \quad (4.3.2)
\]

If we have constructed a good model, the behaviour of our predicted time-series \(\hat{x}_{n+1}, \hat{x}_{n+2}, \ldots\) will remain close to the behaviour of the original data set. If we compared these predicted values with the true values \(x_{n+1}, x_{n+2}, \ldots\), we would expect them to quickly diverge, due to sensitive dependence on initial conditions (the so-called "butterfly effect").

Models of probability density functions, to be described in Chapters 5 and 6, can also be used to model the dynamics of a system. The main difference that occurs is in the generation of the predicted points, which are now chosen from a distribution, rather than coming from a deterministic set of equations.

To perform free-run simulation from such a model, we need to know \(P(x_{t+1}|x_t, x_{t-1}, \ldots, x_{t-L+1})\) and so we must construct a distribution of points \((x_{t+1}, x_t, \ldots, x_{t-L+1})\) with an embedding dimension one higher than the dimension of the system. This gives us

\[
P(x_{t+1}|x_t, x_{t-1}, \ldots, x_{t-D+1}) = \frac{P(x_{t+1}, x_t, x_{t-1}, \ldots, x_{t-D+1})}{P(x_t, x_{t-1}, \ldots, x_{t-D+1})} \quad (4.3.3)
\]
\[
= \frac{\sum_{x_{t+1}} P(x_{t+1}, x_t, x_{t-1}, \ldots, x_{t-D+1})}{P(x_{t+1}, x_t, x_{t-1}, \ldots, x_{t-D+1})}. \quad (4.3.4)
\]

Starting values \(\hat{x}_t, \ldots, \hat{x}_{t-L+1}\) can be chosen either from the data or simply from \(P(x)_t\), and then \(\hat{x}_{t+1}\) is generated from the distribution in Equation (4.3.4). Further points are obtained in the same way, with the selection of \(\hat{x}_{t+2}\) being from the distribution

\[
P(x_{t+2}|x_{t+1}, x_t, \ldots, x_{t-D+2}), \quad (4.3.5)
\]

and so on after that.
In the case of the Hénon map data, described in Section 2.4.1, where \( D = 2 \), this would require us to construct a three-dimensional invariant density, \( P(x_t, x_{t-1}, x_{t-2}) \). This involves additional considerations, since the space that we will be working is now much larger, and more data points are required. If we maintain the same number of partitions along each axis, the number of grid points, and so the size of the inverse solution which we seek, become very large indeed. For the string data, we require a four-dimensional probability density function, and with 100 partition boxes on each axis, we require around \( 1.25 \times 10^8 \) projections, according to Equation (6.1.2), resulting in the number of variables and constraints each being on the order of \( 1 \times 10^8 \). This is much larger than the largest linear programming problem ever solved (see [29]), which involved \( 1.25 \times 10^7 \) constraints and \( 2.5 \times 10^7 \) variables, and thus is certainly larger than the largest quadratic programming problem ever solved, since quadratic programmes are more complex than linear ones. Also, we run into the problem of sparsity in higher dimensions. If the data is too sparse, our one-dimensional Silverman estimates will be very bad, resulting in a very inaccurate final distribution estimate. The only solution to this problem is to obtain more data points.

Fortunately, the squid axon data, described in Section 2.4.3, is very close to one-dimensional, and so we were able to approximate the underlying system using a conditional density of the form \( P(x_{t+1}|x_t) \). The results of this are given in Section 6.4.
CHAPTER 5

Density Estimation Models

In this chapter we first describe two existing methods for estimating the distribution of a data set. The first, in Section 5.1, uses adaptive partitions (see [23]) whilst the second, in Section 5.2, is an adaptive kernel estimator (see [59]).

An original method of reconstructing probability density functions is then presented in Sections 5.3 to 6.3. This technique utilizes the Radon transform and principles of tomography, and has been used to successfully model both artificial and experimental time-series data.

Typically the data that we investigated, from chaotic dynamical systems, formed trajectories lying on attractors of non-integer (“fractal”) dimension. It was the distribution of such points which we wished to model; however, such distributions are usually singular with respect to Lebesgue measure, and are thus poorly represented by smooth function which we might use for modelling other distributions. The problem with such functions for our application is that they tend to smooth in all directions, whereas we wish only to smooth along the attractor, that is, in the direction of the unstable manifold, and not across it, in the direction of the stable manifold.

The data sets which we were most interested in are experimental ones, and thus some amount of noise is almost always present. One need only compare Figures 2.4.1 and 2.4.3 to see the obvious difference in character between the data sets. The presence of such noise smears out the fractal features of the attractor below a certain level, removing the exact Lebesgue singularity. We still wished to minimize the smoothing which we perform across the attractor, but we were able to do so whilst smoothing sufficiently along the attractor.

Once we had constructed our estimates of the probability density function for a particular data set, we examined the possibility of estimating the mean and variance
for any statistic, $S$, computed with that estimator. We were most interested in
the average mutual information, mainly for embedding purposes, as discussed in
Section 2.1. Following the ideas of Solomonoff, discussed in Section 2.3, we needed
to compute the value of $S$ for a variety of description lengths. These values were then
weighted according to Equation (2.3.1), and their mean and variance calculated.

We needed to find a way of obtaining models of different description length,
constructed from the same data. If there were some random factor in our modelling
methods, we would simply have used the algorithms many times, but in fact all the
methods presented in this chapter are deterministic, and will produce exactly the
same model every time. Instead, we perturbed our models by making small changes
to their parameters. These perturbations are described for rectangular partitions in
Section 5.1.1, and for the adaptive kernel estimator in Section 5.2.1.

5.1 Adaptive partitioning estimator

This adaptive $n$-dimensional histogram method, due to Fraser, recursively di­
vides the space into boxes of varying size. See [23] for full details.

Beginning with one box which covers all the of the data points, this estimator
recursively subdivides boxes which show evidence of substructure – that is, divide
boxes whose points seem not to be distributed uniformly, according to a $\chi$-squared
test. In Fraser’s method, the new partition boundaries are set at the mean value of
the data in each dimension. This leads to a “bunching up” of points close to the
boundaries of the sub-partitions. An alternative would be to subdivide equally in
each dimension.

Consider a set of data points lying in (for convenience) a “cube” in $D$ dimensions,
and denote the first partition, containing $2^D$ boxes, by $P_1$, with $P_0$ being a cube large
enough to contain all of the data points. Let the current partition be $P_N$, with those
of its $2^{N\times D}$ boxes which are actually present denoted by $P_N^i$, $i = 0, \ldots, 2^{N\times D} - 1$. 
These boxes will be rectangular and generally of different sizes.

To decide whether or not to further partition a particular box \( P_N \) in \( P_N \), we temporarily partition it into \( R_{N+1}^j \) and \( R_{N+2}^{jk} \), \( j = 0, \ldots, 2^{N \times D} \), \( k = 0, \ldots, 2^{N \times D} \), that is, the next two sub-partitions. The \( R_{N}^{jk} \) boxes are the sub-partitions of the \( R_N^j \) boxes. The uniformity of each box in both \( R \) partitions is examined using \( \chi^2 \)-squared tests. So, we perform two levels of adaptive partitioning on the box in question, and then test the uniformity of these two new sub-partitions. The null hypothesis for the \( \chi^2 \)-squared tests is that the points in each new sub-partition are uniformly distributed. These tests (for a two-dimensional system with 20% confidence levels), are given by Fraser (in [23]) as

\[
\chi_3^2 = \frac{16}{9} \frac{1}{N} \sum_{j=0}^{3} \left( a_j - \frac{N}{4} \right)^2 < 1.547
\]

\[
\chi_{15}^2 = \frac{256}{225} \frac{1}{N} \sum_{j,k=0}^{3} \left( b_{jk} - \frac{N}{16} \right)^2 < 1.287.
\]

Here, \( N \) is the total number of data points in \( P_N \), and \( a_j \) and \( b_{jk} \) are the number of points in each of the boxes of \( R_{N+1}^j \) and \( R_{N+2}^{jk} \) respectively. For uniformly distributed data, in the limit \( N \to \infty \), we will have \( a_i = a_j = \frac{N}{4} \) for all \( i, j \) and \( b_{ij} = b_{k\ell} = \frac{N}{16} \) for all \( i, j, k, \ell \), and hence \( \chi_3^2 = 0 = \chi_{15}^2 \).

If either test fails, then it is concluded that some substructure exists in the new partitions, and the subdivision to \( R_{N+1}^j \) is kept. This process is repeated until no more partitions are necessary, or the number of points in a given box is too small, since the \( \chi \)-square tests are only accurate for asymptotically large data sets. We arbitrarily chose 10 as the minimum number of points that a box must contain in order to be considered for partitioning.

The final weight given to a box is the number of points that it contains, normalized according to the size (area) of the box to produce a probability density. Thus, if two boxes contain the same number of points, the one with the smaller area will be allocated a larger probability than the larger one.
Figure 5.1.1: The partition of the Hénon data produced by Fraser’s method, shown here with the 500 data points.

The partitioning for the 500 point Hénon data set from Section 2.4.1 is shown in Figure 5.1.1 and the corresponding three-dimensional histogram is shown in Figure 5.1.2. Notice how in Figure 5.1.1 the large boxes are very sparse, whilst around the attractor the boxes become small to capture the fine detail there. In Figure 5.1.2, note the flatness of the large empty boxes. Using this method of partitioning, the weighting given to these boxes extends to the edge of the bounding box and never becomes truly zero away from the data as we would like. Partitioning has stopped in all boxes due to too few points, rather than evidence of uniformity. Given the fractal nature of the attractor, we would expect this to be the case even for extremely large data sets.

The description length for Fraser’s method using predictive encoding, as de-
Figure 5.1.2: The density estimate generated for the Hénon data by Fraser's method. The result is very flat away from the data, but non-zero. Many more data points would be needed for a moderately convincing representation of the Hénon map’s density.

5.1.1 Perturbed partitions Following the ideas of Solomonoff described in Section 2.3, we constructed many different partitions so that we could estimate the distribution of statistics on our data. This was done by translating the partition grid boxes.

The partition method that we used was a regular grid in $D$ dimensions, rather than Fraser’s adaptive partition described in Section 5.1. Such a partition could be used, but for our purposes we simply wished to illustrate the drawbacks of using...
any partitioning method and the regular partition was far more easily implemented. We used \( k \) partition boxes in each co-ordinate, resulting in a total of \( k^D \) boxes. For simplicity, the notation for two dimensions will be used here, as the extension to higher dimensions is obvious.

Each box \( B_{ij} \) was allocated the box-counting probability \( p_{ij} = n_{ij}/N \), where \( n_{ij} \) was the number of points falling within \( B_{ij} \) and \( N \) was the total number of (embedded) data points. The probabilities \( p_i \) and \( p_j \) were given by \( \sum_{j=1}^{k} p_{ij} \) and \( \sum_{i=1}^{k} p_{ij} \) respectively. Once these probabilities were found, it was a simple matter to calculate the average mutual information, \( I \), from equation (4.1.1). For each particular partition we estimated the description length using predictive encoding, as described in Section 2.2.

Different partition sizes, that is, different values of \( k \), were tried, and each of these was also perturbed. For the Henon data, described in Section 2.4.1, we found that the minimum description length occurred with \( k = 14 \). The translations for each value of \( k \) were initially uniform random in a range equal to the width of the partition boxes, since any larger translation merely replicates a smaller one. After this we narrowed the range to around one-fiftieth of this size in order to obtain more useful values of description length.

For each partition, the normalized Solomonoff probability, described in Section 2.3, was calculated as well as the average mutual information. The results of a set of \( 10^6 \) perturbations are shown in Figure 5.1.3. From these, only 15000 were unique, and these are the values shown. It is clear from this figure that many of the translated partitions had much higher description length than the best few, resulting in very low \( P_N \) values. The weighted mean was 1.58, with a variance of \( 5.37 \times 10^{-5} \).

The main problem which arose from these results was that translations of the partition grid caused discrete changes to occur in the description length of the
model. For each point that moved into a neighbouring box, the probability of those two boxes changed by $1/N$. This meant that there was a certain minimum change which could be made to description length, and so we could not alter the description length smoothly. In order to use Solomonoff’s weighting from Section 2.3, we needed many models with very similar description lengths. Better estimates could have been obtained by performing many more perturbations, but since only around 1.5% of our translated partitions were used, this would be a very time consuming exercise. In order to overcome these issues, as well as those arising from the discrete nature of the density estimate itself, we turned to an adaptive kernel estimator.
5.2 Adaptive kernel estimator

This method, due to Silverman (see [59] for full details) places a kernel function at each embedded data point, $x_i, i = 1, \ldots, n$. The individual kernels have a local width variable $\lambda_i$ associated with them, and then a global smoothing parameter $h$ is used to combine the kernels.

The adaptive kernel estimates method is comprised of the following steps:

- Choose a pilot estimate $\tilde{p}(y)$ s.t. $\tilde{p}(x_i) > 0 \forall i$. That is, choose a pilot function $\tilde{p}$ which has positive value at the data points.

- Define the local width factors $\lambda_i$ by

$$
\lambda_i = \left( \frac{\tilde{p}(x_i)}{g} \right)^{-a}
$$

(5.2.1)

where $a$ is a sensitivity parameter in the range $[0, 1]$, and $g$ is the geometric mean of the $\tilde{p}(x_i)$'s,

$$
\log g = n^{-1} \sum \log \tilde{p}(x_i).
$$

(5.2.2)

We could eliminate $g$ since it is constant, but it allows $h$ to be independent of the scale of the data, allowing us to make comparisons of different results more easily.

- Define the adaptive kernel estimate $\hat{p}(y)$ by

$$
\hat{p}(y; h) = \frac{1}{n} \sum_{i=1}^{n} (h\lambda_i)^{-D} K \left( \frac{y - x_i}{h\lambda_i} \right),
$$

(5.2.3)

where $K$ is the chosen kernel function and $h$ is the global bandwidth, or smoothing parameter.

In the work presented here, $K$ was chosen to be a Gaussian kernel

$$
K(r) = \frac{1}{(\sqrt{2\pi})^D} \exp^{-\frac{1}{2}(r^TRr)},
$$

(5.2.4)
5.2. Adaptive kernel estimator

Figure 5.2.1: This figure shows the Gaussian and Epanechnikov kernels defined in Equations (5.2.4) and (5.2.5) respectively. Note the compact support of the Epanechnikov kernel.

although similar results were obtained using other kernels such as the Epanechnikov kernel (see [21]), defined by

$$K(r) = \begin{cases} \frac{3}{4\sqrt{5}}(1 - \frac{1}{8}r^2) & \text{if } -\sqrt{5} \leq r \leq \sqrt{5} \\ 0 & \text{otherwise.} \end{cases}$$

(5.2.5)

In both of these kernels, $D$ is the embedding dimension of the data. These two kernels are shown in Figure 5.2.1. Note that the Epanechnikov kernel has compact support, being zero outside the interval $[-\sqrt{5}, \sqrt{5}]$.

Silverman says that the adaptive kernel estimate is insensitive to the choice of pilot estimate, and in our case we chose the inverse of the distance to the nearest neighbour, that is, $\hat{p}(x_i) = |x_i - x^*_i|^{-1}$ where $x^*_i$ is the closest point to $x_i$. This was done because regions where the data points are closer together should have higher densities in our final estimate $\hat{p}(y; h)$. He also discusses the choice of $\alpha$; we used $\alpha = 1/2$ as recommended.
A variant of cross-validation is used by Silverman to determine the value of $h$ for a given model, which involves minimizing the score function

$$M_0(h) = \int p^2(y; h)dy - \frac{2}{n} \sum_i \hat{p}_{-i}(x_i; h),$$

(5.2.6)

where $\hat{p}_{-i}$ is the adaptive kernel function constructed from all data points except $x_i$.

Alternatively, description length can be minimized, as discussed in Section 2.2; this has advantages when dealing with dynamical systems, and enables us to compare different estimates of the density of a data set obtained through different methods. This was the technique which we used to find $h$.

Again using predictive encoding, from Section 2.2, the description length of the adaptive kernel estimator is

$$DL = b n_0 D - \sum_{i=n_0+1}^{n} \log_2(P(x_i|x_1, \ldots, x_{i-1}) \times 2^{-bD}),$$

(5.2.7)

where the $2^{-bD}$ term is included to integrate the probability density. Since the accuracy of the data is $b$ bits, points will be no closer than $2^{-b}$ in each co-ordinate, and so we assume that there is a box of this size around each point which has constant probability. Due to the discrete nature of a partition, this integral is implicit in the way we assign probabilities to a box rather than each point individually, and so no such term appears in the description length function for the adaptive partitioning method of Section 5.1, given in Equation (5.1.3).

We applied the adaptive kernel estimator to the 500 point Hénon data set from Section 2.4.1, and the resulting probability density function is shown in Figure 5.2.2. 30 contour levels are shown, and we can see that the algorithm has successfully picked up the slenderness of the measure around the data to a certain extent. A surface mesh plot of the same probability density function is shown in Figure 5.2.3. The density represents the data well, and goes to zero away from the data. It does appear to be quite broad close to the attractor, however, rather than very narrow as we would like.
Figure 5.2.2: The probability density function generated for the Hénon data using Silverman’s adaptive kernel estimator. 30 contour levels are shown.

Figure 5.2.3: This figure shows the probability density function generated for the Hénon data using the Silverman’s adaptive kernel estimator. The density captures the position of the data well, but spreads out too much close to the attractor.
5.2.1 Perturbed kernel estimators Using Silverman's technique we were able to overcome one of the problems of perturbation discussed in Section 5.1.1, that of obtaining arbitrarily small changes in description length by altering model parameters. Perturbations on this adaptive kernel estimator were performed by changing the value of the global smoothing parameter \( h \).

By smoothly varying the value of \( h \), the value of the model's description length was also changed smoothly, and so we could obtain perturbed models with description lengths as close to the minimum as we desired. Figure 5.2.4 shows this effect, whilst Figure 5.2.5 shows the near-linear relationship between \( h \) and average mutual information \( I \). Unfortunately, as we can see in Figure 5.2.4, the change in description length with \( h \) was not symmetric, resulting in a somewhat skewed distribution for our estimate of \( I \). We weighted the models according to their description length, as described in Section 2.3, and obtained a distribution for \( I \), shown in Figure 5.2.6. The skewing that has occurred is apparent here, but at least in this case we had a good representation of the distribution, compared to Figure 5.1.3. The weighted average is approximately 2.48, with a variance of \( 4.16 \times 10^{-4} \).

Due to the very small variance for average mutual information that we found, we concluded that we can reliably use of the first minimum of \( I \) as the lag for embedding data. The possibility of using probability density estimators such as Fraser and Silverman's for modelling dynamics was now considered, as so much work had gone into implementing computationally complicated algorithms for these techniques. The adaptive kernel estimator for our purposes is clearly superior to a discrete partitioning method, producing a probability density function that is both smooth, and zero-valued away from the data. However, it seems that the distribution obtained using Silverman's method, as seen, for example, in Figure 5.2.2, could be improved to capture important features of the data such as the narrowness of the attractor. A new approach was necessary to utilize the success of Silverman's method whilst overcoming this drawback.
Figure 5.2.4: Description length versus the global smoothing parameter $h$ for 300 evenly-spaced values of $h$. As $h$ smoothly changes, we see that the change to description length is also smooth, although somewhat skewed.

Figure 5.2.5: Average mutual information versus the global smoothing parameter $h$ for 300 evenly-spaced values of $h$. The near-linear relationship between these values is obvious.
5.3 Tomographic imaging techniques

Tomographic imaging is a technique used to reconstruct an image from projections of that image. Its primary application is in diagnostic medicine where it allows two- and three-dimensional examination of internal organs, bones, et cetera without surgery (the ubiquitous “CAT scan” stands for “computer aided tomography”).

The main principle behind tomographic imaging is that it is possible to reconstruct an object from a set of projections of it. Consider Figure 5.3.1, which we will use to describe tomographic imaging in two dimensions. Suppose we have an object whose boundary is the curve C. The scalar function $g$ can be thought of as a mass function which is zero outside this boundary. In medical applications $g$ might be a function of bone or organ density. The projection of $g$ onto the line AB lying at an
Figure 5.3.1: The function $g(x, y)$ is projected onto the line AB, the line of projection at an angle of $\theta$ to the x-axis. This is done by integrating $g$ along the lines $L(t)$ for all $t$.

The angle of $\theta$ to the x-axis is defined by

$$ q_\theta(t; g) = \int_{L(t)} g(x, y) dx dy, \quad (5.3.1) $$

where $L(t)$ is a line $\{(x^*, y^*) : X[\mathcal{R}_\theta^{-1}(x^*, y^*)] = t\}$, $X[.]$ is the x-co-ordinate of the argument and $\mathcal{R}_\theta$ is a rotation about the origin by $\theta$. For a known, continuous function $g$, an analytic expression for $q_\theta$ can be found. In medical applications, three-dimensional functions are projected onto two dimensions, and so $L(t)$ described above becomes a plane. The action of the projection operator on the function $g$ is called the Radon Transform. We describe this transform and its inverse in more detail in Section 5.4.

It is often the case that $g$ is an unknown function, but that we are able to obtain projections of $g$ with relative ease. For example, if $g$ is a bone density function, x-
Chapter 5. Density Estimation Models

rays provide us with two-dimensional projections of that three-dimensional function. Once we have the set of projections for many different values of $\theta$, the Fourier Slice Theorem allows us to reconstruct $g$.

The Fourier Slice Theorem (see, for example, [35] or [41]), sometimes called the Projection-Slice Theorem, is the basis upon which tomographic imaging is founded. This theorem relates the Fourier transform of a projection of the original object to the Fourier transform of the original object along the axis of projection. It states that

$$Q_\theta(\Omega) = G(\Omega \cos \theta, \Omega \sin \theta),$$  

(5.3.2)

where $Q_\theta(\Omega)$ is the standard Fourier transform of $q_\theta(t;g)$, given by

$$Q_\theta(\Omega) = \int_{s=-\infty}^{\infty} q_\theta(t;g) e^{-j\Omega s} ds,$$  

(5.3.3)

and $G$ is the Fourier transform of $g$.

What this means is that the Fourier transform of the one-dimensional projection of $g(x, y)$ is equal to the two-dimensional Fourier transform of $g(x, y)$ evaluated along a line passing through the origin and lying at an angle of $\theta$ to the $\Omega_1$-axis. If we knew $q_\theta(t;g)$ for an infinite number of angles, $0 \leq \theta < \pi$, we could reconstruct $G(\Omega_1, \Omega_2)$ perfectly. However, since we wish to work with real data and cannot calculate the projection for all angles, we must interpolate from the finite number of known slices of the two-dimensional Fourier transform.

To reconstruct $g(x, y)$, one calculates the Fourier transform $Q_\theta(\Omega)$ for each projection $q_\theta(t;g)$, using equation (5.3.3). The Fourier transform $G(\Omega_1, \Omega_2)$ is built from $Q_\theta(\Omega)$ via interpolation, and then the inverse transform is used to obtain an estimate of $g(x, y)$.

For the purposes of analysing time-series data, we must make some modifications to the above techniques. We are dealing with discrete points in space, and the unknown function $g$ that we seek to reconstruct is a probability density function
for those points. In order to perform the reconstruction, we need to first find the projections \( q_\theta(t; g) \). We do this by first projecting the data points onto the line AB, and then using a trusted one-dimensional estimator to convert this discrete set of projected points into a continuous function \( q_\theta(t; g) \).

5.4 The Radon transform and its inverse

The Radon transform describes the action of a projection on any function, including a density (see, for example, [55] or [35]). Specifically, the Radon transform of a scalar function \( f \) is a projection of \( f \) onto a line in the direction of the unit vector \( \hat{n} \), and is given by

\[
\rho_\hat{n}(t; f) = \int \delta(t - \hat{n} \cdot x) f(x) \, dx
\]  

(5.4.1)

Here, \( t \) is the projection of a point \( x \in \mathbb{R}^D \) onto the line, \( \delta \) is the Dirac delta function, and \( \hat{n} \cdot x \) is the inner product. In two dimensions, where \( \hat{n} \) lies along a line at angle \( \theta \) to the \( x \)-axis, and where \( f \) is a density \( p \), this becomes

\[
\rho_\theta(t; p) = \int \delta(t - x \cos \theta - y \sin \theta) \, p(x, y) \, dx \, dy.
\]  

(5.4.2)

The Radon transform is linear and is a convolution, so it can be inverted efficiently by Fourier transform methods. The main inversion result is the Fourier Slice Theorem, Equation (5.3.2). Software for calculating the standard Radon transform and its inverse is readily available; for example, the MATLAB Image Processing toolbox (see [65]) contains the functions \texttt{radon} and \texttt{iradon}.

In principle, this inversion requires knowledge of the transform \( \rho \) for all \( t \) and \( \theta \); in practice, of course, one uses a discretised version which requires \( \rho \) to be known at a finite number of points covering the object of interest. Fast Fourier Transform (FFT) methods enable inversion in time of order \( m \log m \), where \( m \) is the number of points of the transform which we wish to evaluate in the final result.
Later, in Section 6.2, we discuss added constraints to the inversion of the Radon transform, most importantly a non-negativity constraint. Such embellishments are not readily added to FFT methods, and so we used our own algorithm for performing the inversion. The exceptions to this were the first results, which used MATLAB's \texttt{iradon} command, to demonstrate the necessity of these constraints.

Our procedure was as follows. We chose a set of \( N \) projection angles \( \{ \theta_j \}_{j=1}^N \) and for each of these angles, projected the data points \( \{ x_k \}_{k=1}^n \) onto the line lying at an angle of \( \theta_j \) to the x-axis, as described in Section 5.3. The data points will be denoted by \( (x_k, y_\ell) \) when we wish to distinguish their components. One-dimensional kernel density estimates for each projection were then calculated on a uniform grid of \( M \) points lying along the line of projection, although it was not necessary that they be uniform. For this step we used Silverman's adaptive kernel estimator outlined in Section 5.2, to give us a set of one-dimensional kernels

\[
K_j(t) = \text{Silv}(q_{\theta_j}(t; \varphi)), \quad (5.4.3)
\]

for \( j = 1, \ldots, N \). Each \( K_j \) is a vector of length \( M \).

Consider a discretisation of Equation (5.4.2) into

\[
\rho_{\theta_j}(t; p) = \sum_{k, \ell} \Delta(i, x_k \cos \theta_j + y_\ell \sin \theta_j) \ p(x_k, y_\ell), \quad (5.4.4)
\]

where \( \Delta \) is a delta function which allocates the contents of cells to the partition boxes which lie along the rotated line of projection, and \( p(x, y) \) is a probability density function as before. In Figure 5.4.1, these cells are shown using dotted lines, whilst the line of projection lies at an angle of \( \theta \) to the x-axis. The function \( \Delta \) can be chosen in several different ways.

The first approach which we tried was simply to allocate a cell's contents to the partition box that the centre of the cell falls in. In this case, \( \Delta \) in Equation (5.4.4) acts like a discretised Dirac delta function, picking out cases where the projection
of \((x, y)\) was closer to \(t_i\) than to any other \(t\), and so

\[
\Delta(i, z) = \begin{cases} 
1 & \text{if } |t_i - z| < |t_j - z| \text{ for all } j \neq i, \\
0 & \text{otherwise},
\end{cases}
\]  

(5.4.5)

where, \(z_{kt}\) is the projection of the point \((x_k, y_k)\). We found that a big improvement could be obtained by using a more complicated algorithm to allocate cells' contents to the partition boxes on the line of projection.

Using this approach, we apportioned the grid cells amongst the \(t_i\)'s according to the portion of their area that projects onto \(t_i\), and so \(\Delta(i, z_{kt})\) from Equation (5.4.4) becomes the area of the box containing \((x_k, y_k)\) in the \((x, y)\) plane that projects onto \(t_i\), the \(i\)th partition box. This is illustrated in Figure 5.4.1. The cells are shown as dotted lines, and the line of projection (dashed) lies at an angle of \(\pi/20\) to the \(x\)-axis. We can see that the shaded region will now project onto the partition box \(t \in [0.5, 1]\). This region contains approximately 61.5\% of the area of its cell, and so this weighting will be applied to the contents of the cell when it is added to the partition box. For clarity, the partitions here are much larger than the ones we used. Typically, we used around 100 partition boxes on the interval \([-1, 1]\).

We emphasize that there is no requirement that the discretisations in any of the variables be uniform; if the value of \(p\) is of interest at particular points \((x, y)\), then these can be put into the discretisation. Likewise the projection angles can be chosen randomly, uniformly, or in some way that is guided by what is known about the system. As a general rule, even if angles are not chosen uniformly, we should ensure that the \(\theta_j\)'s cover the whole range \([0, 180)\) degrees, lest we miss important features of the data which are obscured except at certain ranges of angles.

Finally, the reader is reminded that if we have analytic information about the functions involved, which is not the case when analysing discrete data points, none of this discretisation is necessary. All the projections can be written analytically, and it is possible to calculate the inverse Radon transform using Fourier methods.
Figure 5.4.1: Cells are allotted proportionally to the partitions that they intersect on the rotated line of projection. The shaded area in this figure will project onto the partition box $[0.5, 1]$, and has an area of approximately 0.615. This weighting will be given to the contents of the two-dimensional cell when they are added to that partition box.
If constraints were to be added to this inverse problem, however, some modification would be required.
Tomographic Density Estimation

6.1 Tomographic density reconstruction

Our procedure was as follows. The kernels \( K_j(t_i), \ i = 1, \ldots, M, \ j = 1, \ldots, N \) were made into a single vector \( R \in \mathbb{R}^{MN} \) and the probabilities \( p(x_k, y_\ell) \) were made into a single vector \( P \in \mathbb{R}^m \), where \( m \) is the number of points at which we wished to evaluate \( P \) in the final result. We used \( m = M^D \); that is, we used the same number of grid points in each dimension of the result as we did in the evaluation of the one-dimensional kernel functions. Following these reconfigurations, Equation (5.4.4) becomes the matrix equation

\[
R = AP
\]

where the elements of the \( MN \times m \) matrix \( A \) are the \( \Delta \)'s from (5.4.4); that is, \( A \) is the matrix version of \( \Delta \). Solving Equation (6.1.1) for \( P \) allowed us to unpack \( P \) into \( p(x_k, y_\ell) \) and hence obtain a two-dimensional probability density estimate.

In general \( m \neq MN \), and so Equation (6.1.1) will not necessarily have a unique solution, or any solution. Since we are confident that the one-dimensional density estimates used to construct \( R \) are accurate, we should ensure that \( m > n \) by taking enough projections (making \( N \) large), and seek an approximate solution to the resulting over-determined set of equations. Thus, with our presently unconstrained problem, we simply find a least squares solution to Equation (6.1.1).

Above a certain number of projections, the results became somewhat independent of \( N \). Clearly, we required a certain number of projections so that the inverse Radon transform problem in Equation (6.1.1) is well-formed. This required is that

\[
MN \geq m.
\]

In all of the results here, we chose \( m = M^D \), where \( D \) is the dimension of the system, and in two dimensions, this results in \( N \approx M \). Once this number of projections has

\[1\] Portions of the work in this chapter have appeared in [5], [6], [38] and [47].
been reached, we might expect that using many more projections would begin to result in over-fitting of the probability density function, and that it might approach a pathologically over-fitted distribution with a spike at each data point and zero probability elsewhere. In fact, this does not occur, due to the reliability of our one-dimensional estimators. The amount of smoothing in each of these is quite independent of the number of projections, and since this smoothing was determined using description length, the one-dimensional projections will not become over-fitted, and so our reconstructed densities will not regress to spikes at each data point.

We illustrate the technique so far with the 500 point Hénon map data from Section 2.4.1. Figures 6.1.1 and 6.1.2 show the results of using the our technique using 25 projections. As mentioned above, these results used the function \texttt{iradon} from MATLAB's Image Processing Toolbox, which performs the inverse Radon transform described in Section 5.4.

It can be seen in these figures that there were many linear artefacts present in the reconstructed probability density function, as well as a great deal of rippling. This occurred most prominently away from the attractor, in regions which we know should have zero probability density. Added to this problem is the fact that there were many negative values, which is strictly forbidden for a probability density function.

Figures 6.1.3 and 6.1.4 show the results of using 91 projections, again with MATLAB's \texttt{iradon} function. It is clear that the worst of the artefacts have disappeared, as have the worst of the negative values of the density function. The presence of small but still significant values in the density function far from the data, and the fact that some of were are negative, was of concern to us. As the number of projections used increases, these fluctuations flatten out, but a very large number of projections are needed before they become negligible for our purposes, and so we sought ways around this problem.
6.1. Tomographic density reconstruction

Figure 6.1.1: Reconstruction of the probability density function of the data in Figure 2.4.1 from 25 projections using MATLAB's iradon function. 40 contour levels are shown. There are many artefacts and a great deal of ripple (leading to negative density estimates).
Figure 6.1.2: Reconstruction of the probability density function of the data in Figure 2.4.1 from 25 projections using MATLAB's iradon function.

One option was to take the absolute value of the reconstructed density function, with an appropriate renormalization. However this was unsatisfactory for a number of reasons, foremost of which was the allocation of non-zero probabilities to regions we believe should be zero.

Another way to remove the negative values would be to perform a smoothing operation on the kernel. Since the negative values occur alongside positive values, if the right filter was chosen, this might work well to reduce the negative values. Choosing such a filter would be very difficult though, as the oscillations to be removed have no specific period. Such an action would also smear out our probability density estimate, broadening it, instead of narrowing it, around the data points.

What we needed to implement was a constraint at a much earlier stage which required the reconstructed function to be positive everywhere.
Figure 6.1.3: Reconstruction of the probability density function of the data in Figure 2.4.1 from 91 projections using MATLAB's `iradon` function. The same 40 contour levels are shown as in Hénon other figures. There are still noticeable artefacts and a good deal of low-level detail that is certainly wrong, including negative values.
6.2 Constraints

There were several possible solutions to the problem of negative density values in our estimates, for example simply taking the absolute value of the estimator, and renormalizing so that the integral was 1. This had the undesirable effect of reducing the positive values, most noticeably around the peaks. This effect will also occur with any local smoothing filter. The problem is lessened if there are a large number projections, but will never disappear completely from the standard inversion. (Recall the Gibbs effect from elementary Fourier series theory; see, for example, [69].)

The two criteria for the reconstructed function to satisfy in order to be a probability density function are having an integral of 1 and being everywhere greater than or equal to zero. It failed to achieve these due to the nature of the inverse Radon transform; in the more usual applications of the transform to digital im-
ages, such oscillations as we encountered show up as background noise. As these, and also medical images, are scrutinized by eye, such artefacts are of little concern in these applications, or are reduced by using very large numbers of projections. For our purposes, however, they caused the reconstructed probability density to be unusable since it was not, in fact, a probability density at all.

We thus added a constraint to ensure non-negativity. The unconstrained problem can be expressed as

\[ AP = R + \varepsilon, \]  \hspace{1cm} (6.2.1)

which is Equation (6.1.1) with errors explicitly allowed for. To enforce non-negativity, we solved the problem

\[
\begin{align*}
\text{minimize} & \quad \varepsilon^T \varepsilon \\
\text{over} & \quad P, \varepsilon \\
\text{subject to} & \quad \varepsilon = AP - R \\
& \quad P \geq 0.
\end{align*}
\]

We could have included a normalization constraint on \( P \) here, but instead normalized after the optimal value of \( P \) had been found. In either case, care must be taken over normalization during the calculation of \( R \).

The above problem is a quadratic program, and solution methods have been well-known for many years (see, for example, [12] or [68]). It can be solved, for example, by methods derived from linear programming, by interior point methods, or by descent methods. If, instead of minimizing \( \varepsilon^T \varepsilon \), we minimize \( \sum |\varepsilon_i| \) or \( \max |\varepsilon_i| \), similar methods can be used.

We dealt with the case \( m > n \), since we were able to use enough projections to give us a set of over-determined equations, and sought an approximate solution to the problem. This approximation can be done conveniently by the minimization of a norm, using any of the \( \ell_1, \ell_2 \) or \( \ell_{\infty} \) norms; linear or quadratic programming are
adequate for all of these cases, which result in convex optimization problems with unique solutions. The $\ell_1$ norm corresponds to robust statistics, with less attention paid to outliers. The $\ell_\infty$ norm concentrates on getting the outliers right, which was not appropriate for us, as we believed that isolated points away from the main body of the data should be ignored. The $\ell_2$ norm was the most appropriate for our applications and was the norm that we used, though experiments with the other norms indicated that the method is fairly robust for different choices of norm.

It is perhaps worth remarking that since $m > n$, it is more efficient to solve the Lagrangian dual problem using most solution methods; however, this is a technical detail concerning implementation will not be discussed further.

6.2.1 Results using positivity constraints For simplicity, we performed computations of the inverse on a uniform grid, but as mentioned above, the method works for any selection of points where it is desired to evaluate the density. We used the optimization package MOSEK to perform these optimizations (see [20] for further information and references). For a grid size of 100 by 100 points and using 91 projections, there are approximately 13100 variables with around 5000 constraints. On a 350 MHz PC, this took a little less than 10 hours to solve using MOSEK's interior point linear/quadratic programming solver, including the time taken to calculate the one-dimensional projections.

The reconstructed probability density function for the Henon map data from Section 2.4.1, using non-negativity constraints, is shown in Figures 6.2.1 and 6.2.2. It compares favorably with Figure 6.1.3, and the function was zero-valued away from the attractor. However, some minor straight-line artefacts remained, which we sought to remove by weighting the equations in Equation (6.2.1). This will be described in Section 6.3.

It can be observed from these results that the outer edge of the probability density function was captured more accurately than the inner edge. The latter cannot
Figure 6.2.1: The reconstructed probability density function for the Hénon map data with 91 projections, with the added non-negativity constraints. The same 40 contour levels are shown as in other Hénon figures. Observe that there are still some artefacts but the resolution and the connectedness on the attractor have both improved over Figure 5.2.2. Low level ripples in the result have been entirely eliminated.
usually be observed in profile at any projection angle $\theta$, and is thus represented less well in our final estimate. The effect here is especially prominent, due to the distinctly concave shape of the Hénon attractor.

A possible solution to this problem is the introduction of elliptical projections, to supplement the standard linear projections described in Equation (5.3.1). In the elliptical case, the straight line $L(t)$ in that equation and in Figure 5.3.1 is replaced by an elliptical line of points passing through $x = t$. One elliptical projection corresponds to a set of ellipses, all with the same eccentricity. Using different families with various eccentricities is the equivalent of using straight lines at different angles to the x-axis. Appropriate terms could be added to the matrix $A$ in Equation (6.1.1) in a straightforward way.

For the experimental data which we examined, however, the attractors do not have a strongly convex shape, and so this problem did not arise except in the artificial case of the Hénon map. Depending on the data that is of interest, various exotic
projections could be included in the matrix $A$ to overcome problems associated with unusually shaped attractors.

### 6.3 Weighted optimization

We believed that the components of $\varepsilon$ from Equation (6.2.1) were approximately independent Gaussian with zero mean, but not identically distributed. This made it appropriate to apply a weight $w_i$ to the $i$th equation in Equation (6.2.1), and we used

$$w_i = \frac{1}{\max\{K_j, \delta\}},$$

(6.3.1)

where $K_j$ is the value of the kernel of the $j$th projection, from Equation (5.4.3), and for some small positive $\delta$, typically $10^{-7}$. That is, we placed more trust in values of the one-dimensional kernel which are close to zero, than those which were non-zero. This was because the non-zero values were more subject to noise, smoothing and interpolation by the one-dimensional kernel estimator, and large values arose from the accidental lining-up of points, which were not corrected by other projections. Due to the nature of the projections being performed, however, we know that any zero value in the projection represents a line of empty cells in the higher-dimensional reconstruction of the function, and so were given to our constrained version of the inverse Radon transform with high weights to indicate this.

In that case, letting $W = \text{diag}(w_i)$, we solved

$$\begin{align*}
\text{minimize} & \quad \varepsilon^T W^2 \varepsilon \\
\text{over} & \quad P, \varepsilon \\
\text{subject to} & \quad \varepsilon = AP - R \\
& \quad P \geq 0,
\end{align*}$$

where $W$ is a diagonal matrix whose entries are the vectors $w_i$ lined up along the diagonal. This is again a quadratic program, and it or its dual can be solved using
Figure 6.3.1: The reconstructed probability density function for the Hénon map data using 91 projections, non-negativity constraints and weighting. The same 40 contour levels are shown as in other Hénon figures. The distribution of the data points is accurately captured, with no artefacts and good connectivity along the direction of the attractor. The artefacts present in Figure 6.2.1 have been eliminated by the addition of the weighting step.

The weighting further improved our results, and entirely eliminated the linear artefacts, which were due to accidental lining-up of points in certain projections. The reconstructed probability density function for the Hénon map data is shown in Figures 6.3.1 and 6.3.2. Comparing these two figures with Figures 6.2.1 and 6.2.2 respectively, we can see that all of the artefacts have been eliminated, without losing the improvements over the standard kernel density estimator of Figure 5.2.2.

Slightly better results were obtained using 200 projections. The probability
6.3. Weighted optimization

The reconstructed probability density function for Hénon using 91 projections, non-negativity constraints and weighting.

The density function for this case is shown in Figures 6.3.3 and 6.3.4. This problem is overdetermined, and the probability density function has a slightly better shape – in particular, the top left-hand tip in Figure 6.2.1 has improved, and there is less spreading out in the bottom half of the attractor. Connectivity has been improved in numerous places around the attractor.

We estimated the Lyapunov exponents as described in Section 4.2, with 20 neighbours and 90 projections. The convergence of our estimate for two local points (one showing expansion, the other contraction) on the attractor is shown in Figure 6.3.5. The convergence occurs quite quickly, and so 10 steps were chosen as being sufficient for our calculation of the exponents. We found the values to be

$$\lambda_1 = 0.587 \text{ bits}$$
$$\lambda_2 = -2.32 \text{ bits}$$

$$HD = 1.25,$$
Figure 6.3.3: The reconstructed probability density function for the Hénon map data using 200 projections, non-negativity constraints and weighting. The same 40 contour levels are shown as in other Hénon figures. The attractor has a slightly better shape and is slightly better connected than when 91 projections were used.
6.3. Weighted optimization

Figure 6.3.4: The reconstructed probability density function for Hénon using 200 projections, non-negativity constraints and weighting.

which agree well with the figures published in [71] and [72] of

$$\lambda_1 = 0.603 \text{ bits}$$

$$\lambda_2 = -2.34 \text{ bits}$$

$$HD = 1.26,$$

although in the latter $\lambda_1$ and $\lambda_2$ are given in units of nats not bits, which use base $e$ logarithms rather than base 2.

6.3.1 Distribution assessment While it is possible to visually inspect plots of distributions which result from our method, such as those shown in Figures 6.3.1 and 6.3.2, and to discern some improvement obtained by using an increased number of projections, it would be better to be able to quantitatively assess this improvement. We attempted this by estimating the description length of a test set of points originating from the same system as those used to construct the distribution, but which were not used in that construction. Using the distribution produced by our
method, we again used predictive encoding (see Section 2.2) to find the description length. For the test set of points this was very fast, since we simply have to look up the value of the distribution for each point sent. The description length of the test points using this method is given by

$$DL_{test} = - \sum_{i=1}^{N_{test}} (\log P_{x,N}(y_i) + (bD - D \times \log(k))),$$

(6.3.3)

where $P_{x,N}$ is the distribution constructed using the data points $\{X_i\}$, $b$ is the number of bits to which each test point, $y_i, i = 1, \ldots N_{test}$, is specified, $D$ is the dimension of the points involved and $k$ is the number of boxes at which $P_{x,N}$ is specified in each dimension.

For the distributions produced so far for the Hénon data, using 25, 91 and 200 projections, the resulting description lengths were 10061, 9622 and 10462 bits respectively. From these few values, we can see that we do need a reasonable number

Figure 6.3.5: The rapid convergence of estimates of local stretching for two points, $x_1$ and $x_2$, from the Hénon map data is seen here. The top plot shows stretching ($\rho > 1$), whilst the lower plot shows contraction ($\rho < 1$).
of projections, but that 200 is probably too many.

In order to find the number of projections required for this description length to reach its minimum, we would need to produce distributions using our method for a large range of numbers of projections. This would be computationally extremely intensive, although a rough estimate could be found, and the processing power of computers is increasing all the time. A further problem arises for our experimental data from the squid giant axon, in that we do not have any spare data to use as a test set. For these results, 91 was again chosen as the number of projections, and the results are quite satisfactory, even if this is not the optimum number of projections to use.

6.4 Experimental data results

We also applied our final technique, incorporating both non-negativity and weighting constraints, to the modified squid giant axon data described in Section 2.4.3. These results can be seen in Figures 6.4.1 and 6.4.2. 91 projections were used.

We estimated the Lyapunov exponents as described in section (4.2), approximating the linear map with 40 neighbours and 10 steps. These exponents were found to be

\[
\begin{align*}
\lambda_1 &= 0.346 \\
\lambda_2 &= -1.55
\end{align*}
\]

(6.4.1)

and the Hausdorff dimension of the system was estimated to be \(HD = 1.22\).

Observe that the sum of the exponents is negative and that there is a positive exponent, suggesting chaotic behavior. The average mutual information for the probability density function was 2.82.

As the modified data was one-dimensional, we were able to calculate free-run simulations using the probability density estimator from Figure 6.4.1. A comparison
Figure 6.4.1: Reconstruction of the probability density function of the squid neuron data from 91 projections. 40 contour levels are shown.

Figure 6.4.2: Reconstruction of the probability density function of the squid neuron data from 91 projections.
6.5 Further work

Using the tomographic density estimation technique, we were able to calculate average mutual information, $I$, for the systems under study. These results confirmed that taking the first minimum of $I$ as the embedding lag is correct. We also had success in using the probability density functions as stochastic models of the dynamics of the system. We would like to be able to extend both of these successful aspects
Figure 6.4.4: The simulated data embedded in two dimensions. Outliers were few, but some discretisation is apparent, due to the chosen grid size. A finer grid (with appropriately longer computation time to find the estimator) could be used to obtain more accurate simulations.
6.5. Further work

Figure 6.4.5: A histogram of the simulated squid data from Figure 6.4.4 after embedding. Comparing this with Figure 6.4.2 we see that despite the discretisation that has occurred, the simulated points represent the original data very well.

of our work to higher dimensions, increasing the practical usefulness of our technique. Ways to do this are described below, but first we address one issue related to capturing the dynamics of a system.

Our main interest in further work concerns the tomographic density estimation technique. In this work, no constraints were added to enforce invariance of the resulting probability density function. Since the data which we were concerned with was an embedded one-dimensional time-series, we know that

$$\int p(x, y) dy = \int p(y, x) dy,$$

when the embedding dimension is 2, and similarly for higher dimensions. This is because the different co-ordinates of the embedded data actually contain the same values, shifted by the embedding lag, and excluding the beginning and end. Further work has been done (see [47] and [38]) in which invariance constraints are added,
and it has been shown in these papers that for the data considered in this thesis, such extra constraints make no discernible difference to the results. There was no noticeable increase in the solution time, either, and so these constraints should be used as a matter of course. An example is given in these papers of a dynamical system for which our technique fails to produce a good probability density function if the invariance condition is not used.

The main improvement which we would like to make to the tomographic method involves the size of the quadratic programming problem to be solved in Section 6.2. Currently, we solve for all points in a $D$-dimensional grid, typically around $10^4$ points for the 2-dimensional examples given in this thesis. We would like to be able to solve much larger problems to increase the resolution of the resulting functions, and to also solve for systems with $D > 2$. Since we know that the probability density function should quickly go to zero away from the data, we are able to allocate a zero value to a large number of points on the grid prior to solving the quadratic problem. This can be done in several ways, such as simply setting $P(I) = 0$ for all points $i \in I : |x_i - x_i^*| > C \times \bar{d}$ for some constant $C$, $\bar{d}$ is the average separation of the data points and $x_i^*$ is the closest data point to $x_i$, a point in the grid. By choosing a quite modest value for $C$, say 0.2, the data is still more than adequately covered, and the number of points that we must solve for is reduced by an order of magnitude. A better approach might be to use successively finer grids and chose the zero-valued points according to their values in the coarser grids.

The addition of this constraint would improve both the speed of calculations as well as reducing memory requirements. By reducing the size of the quadratic problem in this way it would become feasible to estimate probability density functions for higher-dimensional systems.
Conclusion

In this thesis we have discussed two new modelling methods for investigating time-series data from nonlinear dynamical systems. The first was a function series method, selecting wavelet functions from a large dictionary to construct a deterministic model of the system dynamics.

We hoped that the near-compactness of the wavelet functions involved would result in improved models when compared with other forms of function series model, such as radial basis function models. Initially we found that the models produced were not very good, although we were able to obtain stable free-run simulations from them. The attractors of these simulations were distorted approximations to the true attractor. The model had no generalization characteristics even in the region surrounded by the attractor, and clearly improvement was needed.

Different methods for selecting basis elements were tried, but the results did not vary markedly when one of the more complicated selection algorithms was used, and so a simple greedy selection method was used.

An optimization step was added, with the best results found using a combination of Levenberg-Marquardt optimization and the downhill simplex method. Wavelets were chosen for a model using the basis-selection method, and then the translation and dilation parameters for each wavelet were optimized. This resulted in much-improved models, whose generalization capabilities were extended to a region surrounding the attractor. The attractors formed by free-run simulations became indistinguishable from the true attractor.

Following this, we looked at the problem of choosing the embedding parameters of time-lag. A standard method for doing this is to choose the first minimum of average mutual information. We wondered how constrained this choice is, and so sought to estimate error bars for average mutual information calculations.
In order to calculate average mutual information, we needed to know the probability density function of the data points which we were dealing with. Many methods exist for doing this based on partitioning the data and variations of this theme, and one of these was described in this thesis. A different approach is to fit kernel functions to the density, combined with smoothing operations. Using Silverman’s adaptive kernel density estimator, we were able to produce two-dimensional probability density functions for time-series data.

From these estimates of the probability density function, we were able to calculate average mutual information, and by smoothly perturbing the global smoothing parameter, a distribution of the average mutual information was obtained. The resulting variance proved to be very small, however, confirming the practice of choosing the first minimum for the embedding lag. Since we had invested much time and effort into implementing Silverman’s algorithm, we then sought to use the resulting densities for other purposes.

These had a tendency to spread out too much and in the wrong directions - across the attractor as well as along it. However, as a one-dimensional estimator, Silverman’s algorithm worked very well, and so we incorporated it into a new approach.

Using tomographic techniques we took the one-dimensional Silverman estimators and reconstruct two-dimensional probability density functions with much improved characteristics. Unfortunately, the tomographic reconstruction also produced many artefacts, including negative values.

To remove these negative values, we converted the problem into a quadratic programme with positivity constraints. This had the desirable result, making the probability density function non-negative everywhere, as well as removing some of the worst linear artefacts. Additional weighting constraints eliminated these entirely. The final densities had good smoothing characteristics, with very little smoothing
across the attractor.

For both this technique and wavelet series, artificial and experimental data sets were successfully modelled, and we compared our results for the experimental data with previously published values. We found that although the wavelet models were able to capture the gross details of the dynamics, they were not so good at determining finer points of the system, such as the behaviour at fixed points. Features such as the fixed points themselves were in agreement with published results. The tomographic density estimator produced results for statistics such as Lyapunov exponents, and these compared very well with previously published values.
Bibliography


