An Investigation of Nonlinear Speech Synthesis and Pitch Modification Techniques

Iain Mann

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
September 1999

Abstract

Speech synthesis technology plays an important role in many aspects of man–machine interaction, particularly in telephony applications. In order to be widely accepted, the synthesised speech quality should be as human–like as possible. This thesis investigates novel techniques for the speech signal generation stage in a speech synthesiser, based on concepts from nonlinear dynamical theory. It focuses on natural–sounding synthesis for voiced speech, coupled with the ability to generate the sound at the required pitch.

The one–dimensional voiced speech time–domain signals are embedded into an appropriate higher dimensional space, using Takens' method of delays. These reconstructed state space representations have approximately the same dynamical properties as the original speech generating system and are thus effective models.

A new technique for marking epoch points in voiced speech that operates in the state space domain is proposed. Using the fact that one revolution of the state space representation is equal to one pitch period, pitch synchronous points can be found using a Poincaré map. Evidently the epoch pulses are pitch synchronous and therefore can be marked.

The same state space representation is also used in a locally–linear speech synthesiser. This models the nonlinear dynamics of the speech signal by a series of local approximations, using the original signal as a template. The synthesised speech is natural–sounding because, rather than simply copying the original data, the technique makes use of the local dynamics to create a new, unique signal trajectory. Pitch modification within this synthesis structure is also investigated, with an attempt made to exploit the Šilnikov–type orbit of voiced speech state space reconstructions. However, this technique is found to be incompatible with the locally–linear modelling technique, leaving the pitch modification issue unresolved.

A different modelling strategy, using a radial basis function neural network to model the state space dynamics, is then considered. This produces a parametric model of the speech sound. Synthesised speech is obtained by connecting a delayed version of the network output back to the input via a global feedback loop. The network then synthesises speech in a free–running manner. Stability of the output is ensured by using regularisation theory when learning the weights. Complexity is also kept to a minimum because the network centres are fixed on a data–independent hyper–lattice, so only the linear–in–the–parameters weights need to be learnt for each vowel realisation. Pitch modification is again investigated, based around the idea of interpolating the weight vector between different realisations of the same vowel, but at differing pitch values. However modelling the inter–pitch weight vector variations is very difficult, indicating that further study of pitch modification techniques is required before a complete nonlinear synthesiser can be implemented.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Department of Electronics and Electrical Engineering at The University of Edinburgh.

Iain Mann

Acknowledgements

> "An expert is a person who has made all the mistakes that can be made in a very narrow field." – *Niels Bohr*

While not claiming to have made *all* the mistakes possible in the field of nonlinear speech synthesis, I do know that I made a fair few. For helping me out at these times, as well as for other valuable assistance, I would like to acknowledge the following people.

First of all, thanks to my supervisors, Steve McLaughlin and Bernie Mulgrew, who have offered support and advice at all stages of this work. Also thanks to Mike Edgington and Julian Page at BT Labs for their continued interest.

I would like to thank all of my colleagues in the Signals and System Group, with particular mention to: Mark Tuffy and Nick Walton for some uproarious, and occasionally even useful, discussions during our time here; Charlie Unsworth for providing the mutual information software; Mark Cowper for useful discussions about radial basis functions; Damon Thompson for proof–reading this thesis.

Steve Isard of CSTR at Edinburgh University, Dave Broomhead and Jerry Huke at UMIST, and Gernot Kubin from the Technical University of Vienna all discussed aspects of the project with me. Thanks to them for giving their time and advice.

I acknowledge the financial support of EPSRC and BT Labs.

Finally, thanks to Alex Snow for her encouragement during the writing of this thesis.

Contents

List of figures

List of tables

Acronyms and abbreviations

CVC	Consonant–vowel–consonant
FD	Frequency domain
GCI	Glottal closure instant
IPA	International Phonetic Association
LP	Linear prediction
MBR	Multi–band resynthesis
MLP	Multi–layer perceptron
MSE	Mean square error
NLP	Natural language processor
NN	Neural network
PCM	Pulse code modulation
PSOLA	Pitch synchronous overlap add
RBF	Radial basis function
SNR	Signal–to–noise ratio
STFT	Short–time Fourier transform
SVD	Singular value decomposition
TD	Time domain
TTS	Text to speech
V/UV	Voiced/unvoiced

Nomenclature

a	number of evolve steps
$\mathbf{B}$	neighbourhood matrix
d	attractor dimension
$\mathbf{DG}(.)$	Jacobian of map $\mathcal{G}(.)$
$e(n)$	discrete excitation signal
$E(.)$	cost function
$f(n)$	PSOLA synthesis window function
F_0	fundamental frequency (pitch)
F_i	ith formant
F_s	sampling frequency
$\mathcal{F}$	mapping function
$G(.;.)$	Green's function
$h(n)$	PSOLA analysis window function
h	entropy
$\mathbf{h}$	flow vector
$I(A,B)$	mutual information between measurements A and B
m	embedding dimension
M	number of points in neigbourhood vector
P	number of RBF centres
$P(.)$	pitch contour function
$\mathbf{P}$	differential operator
R	pitch modification multiplier
R_n^τ	redundancy
T_0	pitch period length
$\mathbf{T}$	tangent map
$\mathbf{U}$	matrix of left singular vectors
$v(t)$	system variable
$\mathbf{V}$	matrix of right singular vectors
w	SVD window length

w_i	i–th RBF weight
$\mathbf{W}$	matrix of singular values
$x(n)$	discrete time signal
$x(t)$	continuous time signal
$\mathbf{x}$	state space vector
$y(n)$	modified / synthesised signal
Γ	nearest neighbour vector
δ	distance from stable manifold
$\delta(.)$	Dirac function
ϵ	hypersphere radius
λ_i	i–th Lyapunov exponent
λ	regularisation parameter (Chapter 7)
Σ	Poincaré section
ϕ_k	phase offset (Chapter 2)
ϕ	RBF kernel function (Chapters 6, 7)
σ	RBF kernel bandwidth
τ	embedding delay
ω	frequency

Chapter 1
Introduction

This thesis is concerned with the application of nonlinear techniques to the production of synthetic speech within a text–to–speech (TTS) synthesis system. TTS synthesis is the process whereby an unknown input text is converted into the spoken word, via a series of linguistic and speech analyses. Systems capable of this are currently in demand in diverse applications, such as reading machines for the blind, talking machines for people who have lost the use of their voice, and, increasingly, for access to electronically stored information by telephone (*e.g.* remote email reading, automatic airline booking services) [8].

Text–to–speech synthesis is a complex task, with difficulties arising at each stage of the process, not least with the generation of the synthesised sound. All TTS systems are made up of two main units. These are a natural language processor (NLP), followed by a speech generation module [9]. The NLP analyses the text in order to produce a description of the sounds to be synthesised, as well as the intonation, or *prosody*, required for each phrase under consideration. It is then the task of the speech generating unit to produce natural–sounding synthetic speech from this description. Despite more than four decades of research, even the most up–to–date systems are not capable of producing speech that sounds completely human, and this lack of naturalness is one of the reasons preventing the wholesale acceptance of speech synthesis systems by the general public.

The adaptation of nonlinear techniques has become an increasingly common approach over the last two decades to signal processing problems which have not been solved in a satisfactory manner by linear methods. This has included the field of speech processing, with nonlinear techniques applied to speech coding [10], and neural networks incorporated into speech recognition systems [11]. The use of nonlinear methods in conjunction with speech would appear justified, since it is generally accepted that there are a number of nonlinear mechanisms present in the speech production process [12]. Therefore, it is possible that a nonlinear approach will be better able to model the speech signal than its linear counterpart. This rational has also been applied to speech synthesis, where it is expected that some form of nonlinear model will be

able to produce natural–sounding synthetic speech, a goal not achieved by current synthesisers based on linear techniques.

The aim of this work is to investigate how nonlinear methods may be used in speech synthesis. In order to evaluate the suitability of a given technique for this task, two objectives must be satisfied. The first is clear from the above: the nonlinear model should generate natural–sounding synthetic speech. The second is less evident, but equally important: there must be the facility to impose the required prosody upon the synthesised signal. In practice, it will be seen that this means the ability to synthesise a signal at the required *pitch*, and for the required *duration*. Pitch is the underlying excitation frequency of the sound, and, along with the duration, is the most important factor in producing speech with the correct intonation.

The two classes of nonlinear model considered in this thesis reflect different synthesis strategies. Initially a locally–linear model is examined, which models the dynamics of the speech signal by a series of local approximations. This was proposed by Banbrook [5], and may be regarded as an extension of a *concatenative* synthesis approach, in that the original signal data is used as a template during synthesis. Although this technique produces very good quality synthesised speech of any required duration, no satisfactory method to modify the pitch is found.

Focus then shifts to an alternative model, based on the use of a radial basis function (RBF) neural network [13]. In turn, this can be seen as a form of *waveform* synthesis, where the speech waveform is modelled by a set of parameters. In many ways this is a more satisfactory approach, since it actually produces a parametric model of the speech sound. Again this results in good quality synthetic speech, with the possibility of generating any required length of signal. However pitch modification in this method is not resolved either, highlighting this factor as the weakness of these nonlinear approaches.

The achievements of the work described in this thesis are threefold. Firstly, separate from the synthesis issues, a novel technique for marking pitch synchronous points in speech has been developed, which offers a practical demonstration of the usefulness of nonlinear techniques in speech processing. Secondly, the limits of the locally–linear model for use in speech synthesis have been explored. Thirdly, a compact RBF structure, capable of synthesising speech sounds of any required duration in a stable manner, has been developed. This latter contribution will also be of interest to researchers in fields other than speech processing, since it demonstrates how the dynamics of many complex systems can be effectively modelled. In summary, the high

quality of the synthesised speech makes a nonlinear approach desirable, but, without the resolution of the pitch modification problem, the development of a complete nonlinear synthesiser is still not feasible.

1.1 Thesis layout

The remainder of this chapter describes the layout of the thesis, which is organised into a further seven chapters.

Chapter 2 starts with a brief discussion of phonetics, to provide a general description of speech production and the speech signal. It then examines all levels of the text–to–speech process, concentrating on the speech signal generation. The three main synthesis approaches are discussed, and each is shown to have some disadvantages. It is concluded that there is still a requirement for new, improved, speech synthesis techniques.

In Chapter 3 the background to nonlinear dynamical theory and speech as a nonlinear system is presented. Previous work on nonlinear aspects of speech, including possible chaotic behaviour, is discussed. The Lyapunov exponent estimation algorithm, used as an indicator of chaos, developed by Banbrook *et al.* is presented, and the chapter concludes with some new work which may help to explain why the results gained from this algorithm contradict those found by some other researchers.

In Chapter 4 the work on the use of Poincaré maps for pitch marking is presented. This novel piece of work uses a standard nonlinear processing technique as a simple and effective way to mark pitch synchronous points in a speech signal. It is concluded that the algorithm is effective when applied to simple stationary signals, and gives promising results for real, continuous, speech.

Chapter 5 deals with synthesis by the locally linear model. The algorithm as developed by Banbrook is described and its performance is evaluated. It is shown that the technique produces natural–sounding speech, but that pitch modification as proposed by Banbrook is not realistic. A possible new method for pitch modification is proposed, based on an examination of pitch variations in real vowel sounds. This is shown to work in principle, but fails when used in conjunction with the locally linear model. Therefore, it is concluded that this method is limited as a speech synthesis technique.

Chapter 6 begins the discussion of the radial basis function approach to synthesis. Other existing neural network speech synthesisers are reviewed, and then the RBF structure used for synthesis is described in detail. It is shown that the lack of stability in the output signal is the major problem with this technique. Various methods are used to investigate the stability of the network, and it is concluded that the RBF network model is more nonlinear than the original speech data, making it extremely sensitive.

In Chapter 7 regularisation theory is introduced to solve the stability problem in the RBF network, allowing stable resynthesis. The results of the RBF synthesis of vowel sounds, compared to the original speech and a linear prediction synthesiser, are presented. The use of the RBF synthesiser for pitch modification is then examined. A basic concept involving interpolation between sets of parameters derived from models at varying pitch levels is proposed. The chapter concludes with some simple experiments which show this method may be valid, although no way of effectively implementing it was found.

Finally, in Chapter 8, the main contributions of the thesis are highlighted, and areas for possible further work are suggested.

Chapter 2
Background to speech synthesis

2.1 Introduction

Text to speech synthesis is a complex task that aims to convert the written word into naturally–sounding speech. While working systems that produce intelligible speech have existed since the 1970's, the final aim of producing a synthesiser that is indistinguishable from a human speaker has still to be realised. There remain a number of problems at all stages of the process, including the actual generation of the speech signal itself with the required intonation. Improving this last stage is the motivation behind this work.

After a brief introduction to phonetics, which aims to define much of the vocabulary required to describe the various forms of speech sounds that will be encountered in the thesis, this chapter will concentrate on an overview of text to speech (TTS) synthesis. The various components required to build a working text to speech synthesiser are discussed in general, and then the twin tasks of speech signal generation and the imposition of intonation or *prosody* onto this signal are discussed in more detail.

2.2 A description of speech

The field of phonetics includes the study of speech production and the acoustics of the speech signal, and provides a way to effectively describe speech.

2.2.1 Articulatory phonetics

Speech signals are produced by the process of air being expelled from the lungs and travelling through the vocal tract. The resulting sound pressure wave radiates out from the lips. The various speech organs are shown in Figure 2.1 and, according to their positioning, a large variety of sounds can be produced. In order to discuss these sounds unambiguously, they are categorised into a series of distinct types according to how they are produced.

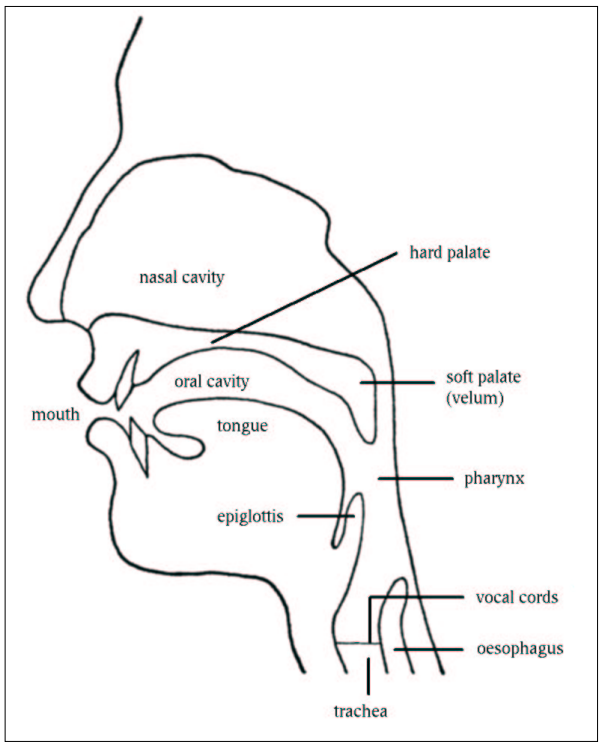


Figure 2.1: *The speech organs (after [1])*

Voicing

The larynx is at the base of the vocal tract and mainly comprises two bands of muscle and tissue called the vocal cords or folds. All air from the lungs must pass through the vocal folds, and they can impede its passage to a greater or lesser extent. In terms of speech production, the vocal folds can operate in three ways:

- vibrating in a pseudo–periodic manner to create *voiced* sounds. The frequency of this vibration is called the fundamental frequency, and corresponds to the tone heard by a listener which is called pitch[1];
- not vibrating for *unvoiced* sounds;
- stopped closed to produce a *glottal stop*, the glottis being the gap between the vocal cords;

[1] These terms are often used interchangeably due to their close correspondence [8].

Interference in the vocal tract

The different articulatory organs (*e.g.* tongue, lips, soft–palate) in the vocal tract can be positioned so as to modulate the flow of air through the tract in different ways.

1. Closure. As well as the glottal stop, the vocal tract may be closed at other places such as at the lips, or between the tongue and hard palate. If the velum is lowered, then air can flow out through the nose creating a *nasal* sound. However, if it is raised (as is more normal in English), there is no way for the air to escape. Therefore the pressure in the vocal tract increases, and when the closure is removed the air bursts out creating a *plosive* sound.

2. Narrowing. If rather than completely closing the vocal tract, two speech organs are instead brought close together, then the air flow past them becomes turbulent and produces *fricative* sounds. The narrowing can occur at any point in the vocal tract.

3. Open. With the speech organs sufficiently open so that no turbulence is produced in the air flow, *vowel* sounds are generated. These sounds are always voiced, and it is mainly the position of the highest part of the tongue that determines the vowel produced. This leads to a widely used description in which vowels are specified according to which part of the tongue is highest (front, central, back) and how high it is (close, mid, open).

Phonetic symbols

The International Phonetic Association (IPA) has produced a set of phonetic symbols to define all of the individual speech sounds (called phonemes) in terms of their place of articulation. For vowel sounds, the above description of tongue position is used. The IPA vowel chart is shown in Figure 2.2, and provides an alphabet of all vowel sounds by place of articulation.

2.2.2 Acoustic phonetics

Now it is necessary to consider how the actual speech waveforms themselves can be linked to the description of sounds explained above, which is based entirely on the mechanics of their production.

Figure 2.3 shows the time domain and frequency spectrum of the voiced part from the word

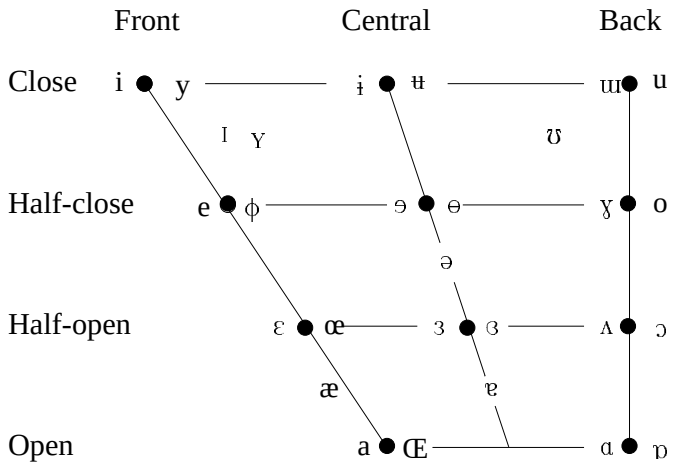


Figure 2.2: *The IPA vowel chart, showing vowel phonetic symbols according to tongue position.*

"hart", corresponding to the vowel /ɑ/. All of the transition from and to the other unvoiced sounds has been removed. The spectrum, which is limited to 4 kHz, shows a number of peaks at differing levels. These are due to the pseudo–periodic excitation signal and the subsequent modulation by the vocal tract. The first peak, at approximately 230 Hz, is at the fundamental frequency and the other peaks are harmonics of this. There are also areas of higher energy within the spectrum, corresponding to the resonant frequencies of the vocal tract. These are called formants, and are labelled incrementally F1, F2, F3, *etc*. In the frequency range shown, three formants are visible at approximately 700 Hz, 1300 Hz and 2700 Hz. Formant frequencies provide a useful method to characterise vowels acoustically, since they depend upon where the vocal tract is constricted and by how much – *i.e.* tongue position [14]. There are at least five formants present in a typical vowel, but in terms of using them for specification, only the first two or three are required. The most common representation is the two–dimensional formant chart, in which the first formant (F1) is plotted against the second (F2). Figure 2.4 shows the position of ten vowels on this representation, using average male values of F1 and F2 (from a study by Peterson and Barney [2]). Clearly there is considerable similarity between the position on the formant chart and tongue position as used in the IPA system, thus together giving a useful system for the specification of vowel sounds.

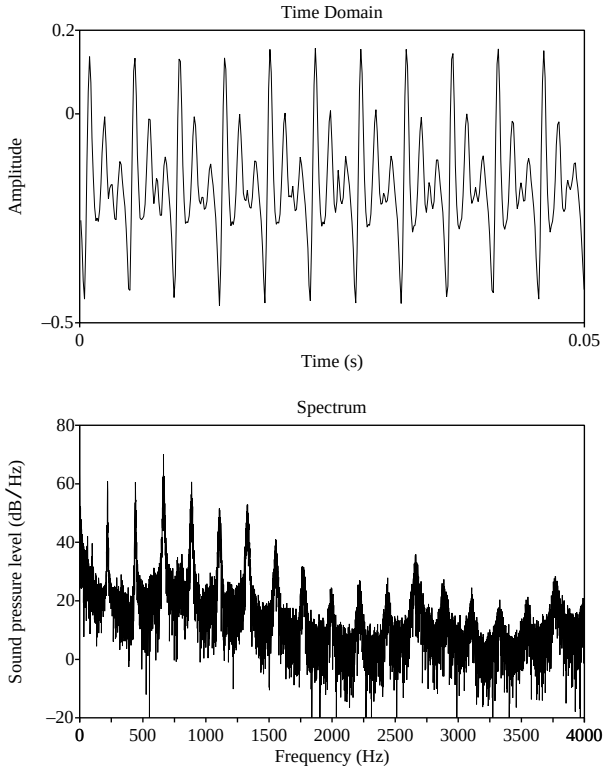


Figure 2.3: *Time domain and spectrum of the vowel part of the word "hart".*

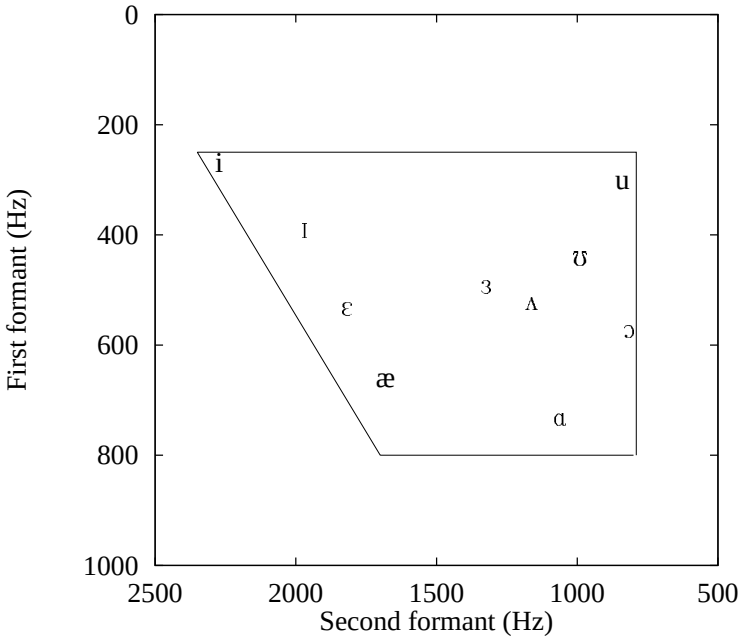


Figure 2.4: *Formant chart for average male vowel sounds using data from Peterson and Barney [2].*

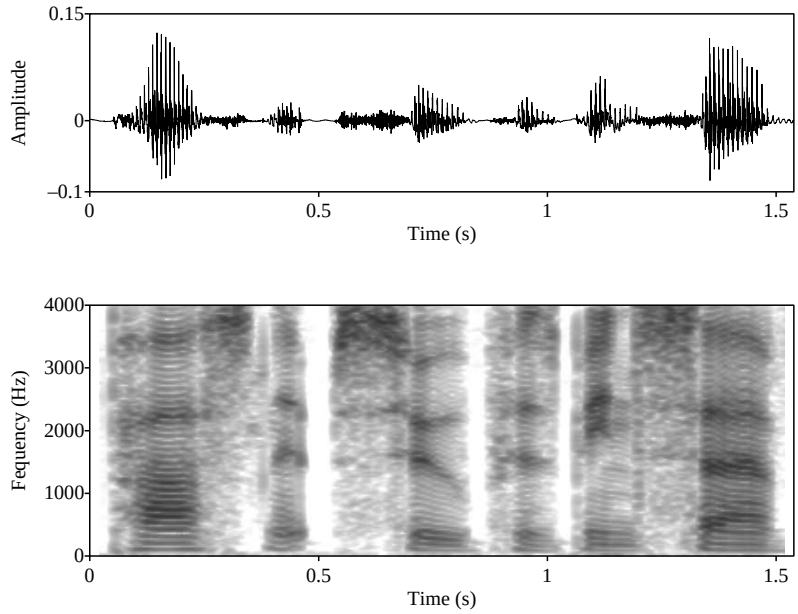


Figure 2.5: *Time domain and spectrogram of the utterance "parsnip soup has been served".*

2.2.3 Nonstationarity

Up until now only single units of speech (phonemes) have been considered. Obviously in real connected speech, these units will be joined together to produce words and sentences. The durations of the individual sounds will be short, meaning that normal spoken speech is highly nonstationary. It consists of many transitions from silence to speech, between different phonemes, and back to silence. This is clearly seen in Figure 2.5, which shows the time domain and spectrogram of the utterance "parsnip soup has been served". A spectrogram is a plot of time against frequency, with the energy at any particular time–frequency instant given by the intensity at that point. As well as demonstrating nonstationarity in the signal, the spectrogram is also useful for observing the formants during the vowel sounds, seen as dark horizontal bands.

2.3 Overview of TTS systems

A text to speech synthesis system must perform the task of converting an input text stream to speech. This is extremely difficult, especially since the text does not contain all of the information required for it to be rendered as speech. In particular, intonation – the information

on *how* to say the words – is inadequately specified [15]. In any system there are two main stages. The first is a textual and linguistic analysis, which provides a detailed description of exactly what sounds are required and the intonation, or prosody, to go with them. This is passed to the second speech generation stage, which produces the actual synthesised speech signal.

The various methods for speech signal generation, which are the main area of interest in this thesis, are described in detail in the next section. However, it is useful to first examine in general terms the higher level processing required to reach the generation stage. Further detail can be found in existing speech synthesis review articles, such as [16–18]. Figure 2.6 shows a flowchart of the various stages in a complete TTS synthesis system.

2.3.1 Text preprocessing

The input text is initially segmented into paragraphs, then sentences, then words. Simultaneously, any ambiguous structures must be resolved. These will include abbreviations, numbers and dates. This task becomes more difficult as less restrictions are placed upon the structure of the input text. A classic "free text" example would be the TTS conversion of an email, which would require processing of the header to extract the sender's name and other details, as well as being able to deal with a huge variety of styles in the message text itself.

2.3.2 Pronunciation

The pronunciation of each word in the text is computed at the next stage. In languages where words are pronounced exactly as they are written, a simple letter–to–sound rule approach can be used. Other languages, including English, which can have very different pronunciations compared to the spelling of words, must use a different method. Some form of dictionary is useful, but this can not cover all the words in a language in practice. Hence a morphological analysis is often used to break the word down into its component parts. The pronunciation of these can then be computed, possibly with the aid of a more limited dictionary.

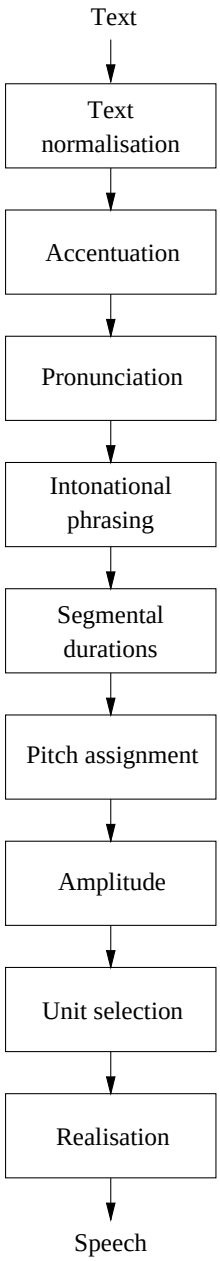


Figure 2.6: *Overview of the steps in the TTS synthesis process.*

2.3.3 Syntactic analysis

It is now necessary to consider how the words fit together as phrases and sentences, so as to generate the sentence structure. This will be used to compute the prosody. The first step is called tagging and involves defining which syntactic category each word belongs to (*e.g.* noun, verb, adjective). Then a syntactic parser is used to build up a representation of each sentence structure. This aims to identify phrases, which phrases are linked, and how they are linked within a hierarchy [18]. Using this information a decision can be made about the accentuation to assign to each part of each phrase.

2.3.4 Prosody generation

Prosody concerns variations in pitch, duration and volume which affect the perception of many features in speech including tempo, rhythm, accentuation and emphasis [19]. Thus at this stage the interest is not so much *what* is being said, but more *how* it is being said. An example of the need for correct prosody is shown by the phrases "It's in the box." and "It's in the box?". The first is a statement, and a human speaker would normally give fairly level emphasis over the whole phrase. The second is a question, which is often characterised by rising pitch at the end of the phrase, with perhaps added emphasis on the word "box". Using the results of the previous linguistic analysis, the prosody generator computes a pitch contour, segmental durations of the individual sound units, and changes in volume. Volume changes are easy to impose at a post–processing stage, and have also been identified as being perceptually the least important part of prosody [20], so more emphasis is placed on the duration and pitch computations.

Segmental durations

The duration of the individual sound segments is dependent upon a number of factors including the actual segment in question, whether the syllable of which it is a member is stressed or accented, and the position of the syllable in the phrase. Various models for segmental duration have been proposed (see for example [21–23]), using a segmented and labelled database that has a wide coverage of the unit sounds in a variety of linguistic situations.

Pitch contour

Pitch changes are not restricted to within single sound segments, but act more as a trend across a complete phrase. For this reason, a pitch contour that specifies pitch across whole phrases is used. Generating this contour involves selecting maximum and minimum pitch levels for the speaker in question and then setting appropriate pitch values according to the pre–defined accentuation of the phrase. Following this, the F0 contour is produced by interpolating between these values, with some additional smoothing [17].

2.4 Speech signal generation

Once the information about the speech segments to be synthesised and the prosody model has been generated, the final stage of the TTS system is to produce the synthesised speech signal. The main approaches to this task depend on the type of modelling used. This may be a model of the speech organs themselves (articulatory synthesis), a model derived from the speech signal (waveform synthesis), or alternatively the use of pre–recorded segments extracted from a database and joined together (concatenative synthesis).

2.4.1 Articulatory synthesis

Modelling the actual speech organs is an attractive approach, since it can be regarded as being a model of the fundamental level of speech production. An accurate articulatory model would allow all types of speech to be synthesised in a natural manner, without having to make many of the assumptions required by other techniques (such as attempting to separate the source and vocal tract parts out from one signal). The main steps in an articulatory model are: a source model for the vocal cord vibration; a saggital function to model the vocal tract in two dimensions; an area function for the third vocal tract dimension; a lip model to determine the shape of the lips [24]. These will be interlinked in practice, but are considered separately in the following.

The vocal fold vibration may be modelled by a two or more mass model, in which the movement of the vocal cords is simulated by masses coupled by springs, *e.g* [25, 26]. The coupling factor is nonlinear in order to take into account known nonlinear stretching properties in the larynx. An alternative approach simulates the glottal flow with a model based on the Navier–Stokes

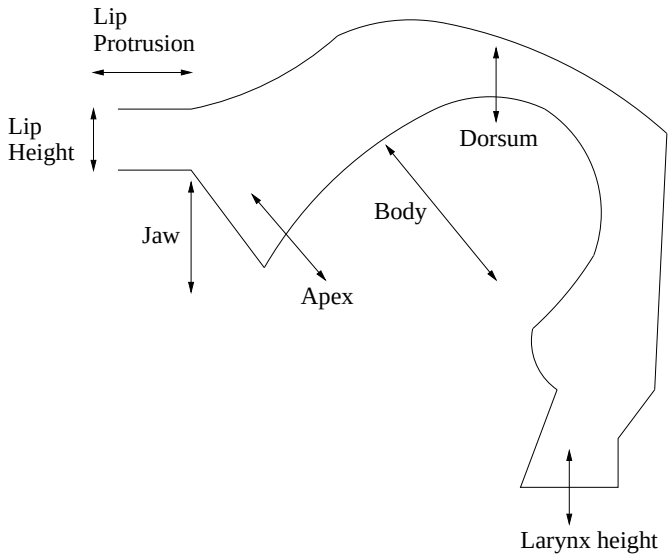


Figure 2.7: *The 7 parameters used in Maeda's vocal tract model, showing possible movement directions for different sounds.*

equations, again resulting in a nonlinear model [27].

Using X–ray data taken during speech production, cross–sections of the vocal tract for different speech sounds are obtained. These are used to build up statistical models of the vocal tract, such as that produced by Maeda [28]. Average values for each of the following parameters are calculated for each phoneme: lip protrusion, lip height, jaw, tongue body, tongue dorsum, tongue apex, larynx height, as shown in Figure 2.7. To this cross–section, a third dimension is added to create an area function. The transformation from the cross–section to the area function is achieved via a look–up table, which is derived from, for example, a cast of the vocal tract [29].

Lastly a lip model, defining the lip shape and area is included. This ensures that the lip area is a realistic size for the speech sound being produced, and may also be used in visual synthesis of lip movement.

It will be evident from the above that realistic articulatory synthesis is an extremely complex process, and that the data required is not at all easy to collect. As such, it has not to date found any commercial application and is still more of a research tool, although it may be that this technique will ultimately provide the best solution to speech synthesis.

2.4.2 Waveform synthesis

Waveform synthesisers derive a model from the speech signal as opposed to the speech organs. Generally this means attempting to model the resonant frequencies of the vocal tract in terms of poles in a filter or bank of filters. This is then excited by some source excitation model. Sometimes zeros are also included in the vocal tract filter to model the effects of the nasal cavity. This approach is derived from the linear source–filter theory of speech production [30], which is illustrated in Figure 2.8 for the case of a vowel sound.

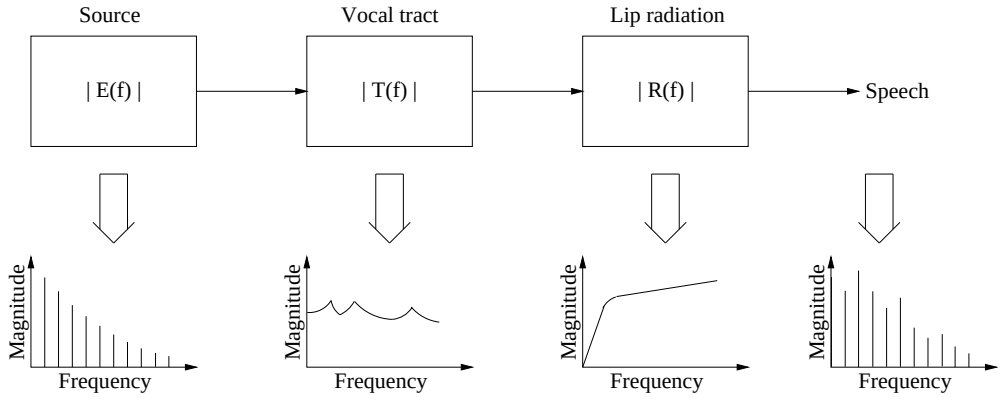


Figure 2.8: *The linear source–filter theory of speech production.*

In this model, the source is modulated first by the vocal tract resonances, then by the lip radiation model.

Perhaps the simplest form of waveform synthesis is based on linear prediction (LP) [31]. Here the above model is further simplified by combining the spectral characteristics of source, vocal tract and lip radiation into one filter. The current synthesised output $y(n)$ is found from a linear combination of previous outputs and the current input [32]:

$$y(n) = \sum_{k=1}^{P} a_k y(n-k) + e(n) \tag{2.1}$$

where there are P filter coefficients a_k. $e(n)$ is the current input sample, which is a Dirac pulse train for voiced speech, or white noise for unvoiced speech. This model is shown in Figure 2.9. The resulting quality is extremely poor for voiced speech, sounding very robotic. However, interestingly, it has been found that this method of generating unvoiced speech using a white noise source is perceptually almost identical to the original [33].

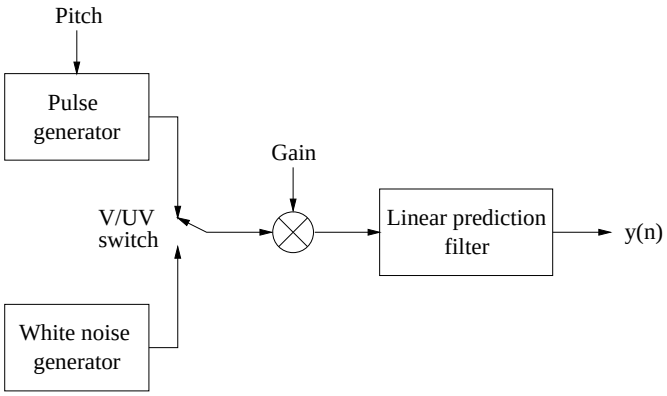
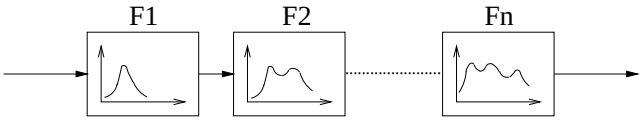
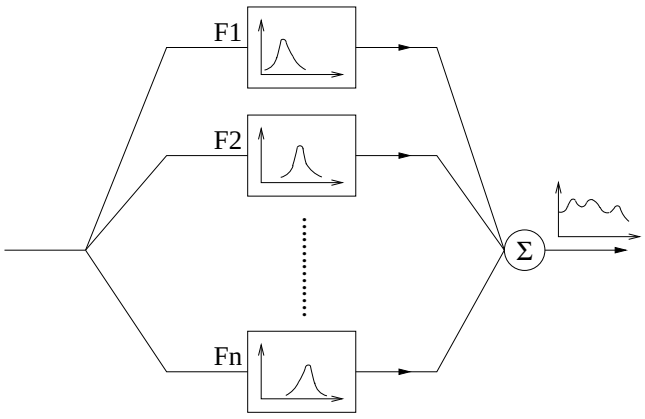


Figure 2.9: *The linear prediction speech model.*

Formant synthesis uses a bank of filters, each of which represents the contribution of one of the formants. The two basic structures are cascade and parallel, as shown in Figure 2.10.



(a) Cascade



(b) Parallel

Figure 2.10: *Cascade and parallel realisations of the formant synthesiser.*

Both are based on the same underlying theory, but when implemented the cascade design is better at modelling vowel sounds, whereas the parallel design is superior in the modelling of

plosives, fricatives and nasals [8]. These synthesisers also include sophisticated glottal pulse models and some element to represent lip radiation characteristics. The best known formant synthesiser is the Klatt synthesiser [34], which can be operated in either parallel or cascade mode and has been exploited commercially as DECTalk. The synthesised speech quality is considerably greater than that of the LP method, but still lacks naturalness, even when an advanced voice–source model is used. Thus, while being highly intelligible, formant synthesisers still tend to sound very robotic [35].

2.4.3 Concatenative synthesis

The idea behind concatenation methods is very simple, and involves joining together pre–recorded units of speech which are extracted from a database. It must also be possible to change the prosody of the units, so as to impose the prosody required for the phrase that is being generated.

The construction of the database is extremely important. It must be possible to cover all the speech segments required, and these segments must be stored in a compact manner which can be easily and quickly accessed. Typically, a database for the English language will contain all of the diphones as well as a selection of longer units [8]. A diphone consists of the transition from the centre of one phoneme to the centre of another. In English, there are approximately two thousand diphones, making this a reasonable trade–off between performance and database size. The use of the shorter phoneme unit is not possible, as it does not contain any transient information. Use of longer units, such as syllables, is also not possible due to the size of database this would imply [35]. After labelling, which is usually performed semi–automatically, the database is stored using some form of speech coding, such as linear predictive coding.

Ideally the database will contain more than one example of each diphone, covering different co–articulation effects which occur at syllable level or higher. In this way a search can be made for the best diphone among the possible selection to minimise perceptual differences between neighbouring concatenated elements. A particular diphone is chosen so as to minimise some criteria, such as the spectral difference between elements, or a unit concatenation penalty (where the penalty is calculated taking into account human auditory masking properties) [36].

Having selected the most suitable diphones, the correct prosody must be imposed onto the phrase. This inevitably causes some distortion which must be minimised. The two main tech-

niques for prosody modification are the pitch–synchronous overlap add algorithm, and the harmonic techniques based around a sinusoidal model. These are described in detail in the next section.

The concatenation technique provides the best quality synthesised speech available at the time of writing. It is used in a large number of commercial systems, including British Telecomm's Laureate [37] and the AT&T Next–Gen system [38]. However this technique is limited by its reliance on a database, and the number of different types of speech example that can be stored in it. In this respect it does not have, in theory at least, as great a potential as synthesis techniques that use some form of parametric speech model. Although there is a good degree of naturalness in the synthesised output, it is still clearly distinguishable from real human speech, and it may be that more sophisticated parametric models will eventually overtake it.

2.5 Time–scale and pitch modification

As most TTS synthesis systems currently in use generate the speech signal by the concatenation method, techniques for time and pitch scaling of sounds held in a database are extremely important. Even if the upper tiers of the system are capable of producing a good phonetic description with accompanying prosody model, the output will still sound very artificial if the imposition of the prosody model generates audible artifacts, or removes naturalness.

Two main techniques for time–scale and pitch modification in concatenative synthesis can be identified, each of which operates on the speech signal in a different manner. The pitch synchronous overlap add (PSOLA) approach is non–parametric as opposed to the harmonic method, which actually decomposes the signal into explicit source and vocal tract models.

2.5.1 Pitch synchronous overlap add

The PSOLA algorithm and its variants are all based on the principle of overlap–add (OLA) synthesis and the short–time Fourier transform (STFT). Although PSOLA itself does not actually require the calculation of the STFT, it is still useful to start at this point. Any STFT and OLA algorithm has three stages: analysis, modification and synthesis [3].

Analysis

Analysis consists of the calculation of the STFT at successive time locations through the speech signal. The STFT uses a window at a certain time instant which effectively restricts the Fourier analysis to the portion inside the window, by setting the rest of the signal to zero. If the window is of a short enough duration, the remaining signal can be considered stationary. Mathematically, given the speech signal $x(n)$ and defining $h(n)$ as the analysis window, then the sth short–time analysis signal $x(t_a(s), n)$ around the time instant $t_a(s)$ is given by the multiplication of the signal and window:

$$x(t_a(s), n) = h(n)x(t_a(s) + n) \tag{2.2}$$

and will have a short-time spectrum

$$X(t_a(s), \omega) = \sum_{n=-\infty}^{\infty} h(n)x(t_a(s) + n)\exp(-j\omega n) \tag{2.3}$$

$h(n)$ is a symmetric, finite duration window, centred around $t = 0$.

Modification

Some suitable modification can now be made, such as a compression or expansion of the frequency axis for pitch scaling. Generally, the spectra $X(t_a(s), \omega)$ are modified to produce a series of short–time synthesis spectra $Y(t_s(u), \omega)$, which are synchronised onto a new set of synthesis time instants, $t_s(u)$. These synthesis time instants are defined according to the required pitch and time–scale modifications, and there may be a differing number of synthesis time instants compared to analysis time instants.

Synthesis

Finally the short–time synthesis spectra are recombined to generate the modified signal $y(n)$. However, because of the modifications made, it is possible that the inverse STFT of $Y(t_s(u), \omega)$ will not exist. What is required is the closest spectrum to $Y(t_s(u), \omega)$ for which the STFT does exist. This is found, in the least–squares sense, by the overlap–add method [39], and gives the

synthesis signal as:

$$y(n) = \frac{\sum_n y_w(u, n - t_s(u)) f(n - t_s(u))}{\sum_n f^2(n - t_s(u))} \tag{2.4}$$

where $y_w(u, n)$ is the Fourier transform of $Y(t_s(u), \omega)$, and $f(n)$ is the synthesis window, which will have similar properties to $h(n)$.

2.5.1.1 PSOLA analysis–modification–synthesis

Modifying the pitch implies breaking the signal into some form of source–filter structure, imposing changes on the source, and then resynthesising the signal. PSOLA is able to perform this in a single, computationally efficient step, by implicitly estimating the transfer function of the filter thanks to the use of pitch synchronous windowing. An in–depth description of the complete PSOLA algorithm, including a discussion of this, is found in [40].

- The analysis stage decomposes the signal into the short–term analysis signals $x(t_a(s), n)$:

$$x(t_a(s), n) = h_s(n) x(n - t_a(s)) \tag{2.5}$$

  where $h_s(n)$ is the sth analysis window, whose length is equal to twice the local pitch period. Generally, a symmetric Hanning window is used, and the resulting short–time signals are also referred to as PSOLA bells [41]. The analysis is pitch synchronous, with each time instant $t_a(s)$ being set at the glottal closure instant (during voiced speech), corresponding approximately to the pitch onset time. In unvoiced segments the analysis time instants are set at a constant rate.

- Modification requires the calculation of the synthesis time instants, and proceeds in two steps. Time instants are computed according to the required pitch and time–scale modification, and then each synthesis time instant is associated with one or more time analysis instants. Following the notation of Moulines [42], the analysis time instants $t_a(s)$ give a pitch contour function $P(t)$ where

$$P(t) = P(t_a(s)) \qquad t_a(s) \leq t < t_a(s+1) \tag{2.6}$$

  To perform pitch modification, the synthesis time instants must also be positioned pitch synchronously, and must follow the synthesised pitch contour $P'(t)$. Using a recursive

algorithm, the synthesis pitch marks $t_s(u)$ are found so that

$$t_s(u+1) = t_s(u) + P'(t_s(u)) \tag{2.7}$$

with $P'(t_s(u))$ being approximately $1/\beta$ times the pitch of the original signal around that time instant, where β is the required pitch modification factor. Time scale modification operates similarly in order to generate a stream of synthesis time instants, while still preserving the pitch contour. Finally the mapping between the synthesis and analysis time instants is calculated by taking a weighted average of the two closest short–time signals:

$$y(u,n) = (1-\alpha_u)x(s,n) + \alpha_u x(s+1,n) \tag{2.8}$$

where α_u is the weighting factor, which gives greater weight to the closest analysis time instant to the synthesis time instant $t_s(u)$.

- In the final synthesis stage, the synthesised signal is obtained by combining together all of the short–term synthesised signals on the series of synthesised time instants $t_s(u)$, using OLA as in Equation 2.4. The synthesis window $f_u(n)$ is set equal to that used for the analysis window around analysis time instant $t_a(s)$, where $t_a(s)$ maps to the synthesis time instant $t_s(u)$

Figure 2.11 shows graphically how the PSOLA technique performs pitch modification. The technique is reported to give good quality, natural–sounding synthetic speech for moderate pitch and time modifications. Slowing down the speech by a large factor (greater than two) does introduce artifacts due to the repetition of PSOLA bells. Some tonal artifacts (*e.g.* whistling) also appear with large pitch scaling, especially for higher pitch voices, such as female speakers and children.

The above method operates entirely in the time domain, and consequently is known as time–domain or TD–PSOLA. Several variations are possible. Frequency–domain (FD) PSOLA involves actually calculating the STFT for each analysis window, with modifications carried out in the frequency domain. In principle this allows better control of the spectral envelope, although it is very much more complex than TD–PSOLA. Another option is to carry out the modifications on a linear prediction residual, giving LP–PSOLA [40, 43, 44]. Here at each time instant, a spectral envelope is estimated and this is used to extract the excitation. After

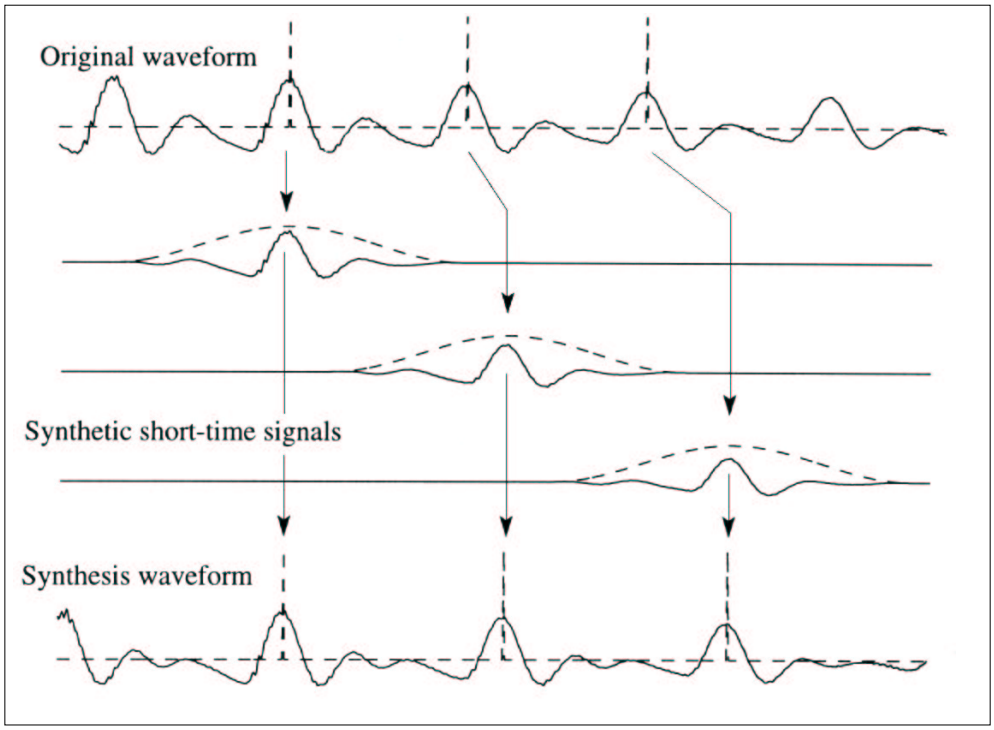


Figure 2.11: *Pitch modification by the PSOLA technique (after [3]).*

modification, the output signal is obtained by combining the modified source with the spectral envelopes, synchronised with the synthesis time instants. Problems often occur in concatenative synthesis when joining and modifying two segments extracted from different words in a TTS database, resulting in phase, pitch and spectral envelope mismatches. A solution to this is MBR–PSOLA (Multi–Band Re–synthesis PSOLA) [45] which resynthesises the entire database to a constant pitch. This is performed once off–line, using a modification of the MBE coder [46], and results in a database better adapted to the use of PSOLA. It should be noted that all of the PSOLA techniques rely to some extent on the linear source–filter model of speech, and so will inherit problems found in this model.

2.5.2 Harmonic transformation

McAulay and Quatieri developed a speech generation model that is based on a glottal excitation signal made up of a sum of sine waves [47]. They then used this model to perform time–scale and pitch modification [4].

Starting with the assumption made in the linear model of speech that the speech waveform $x(t)$

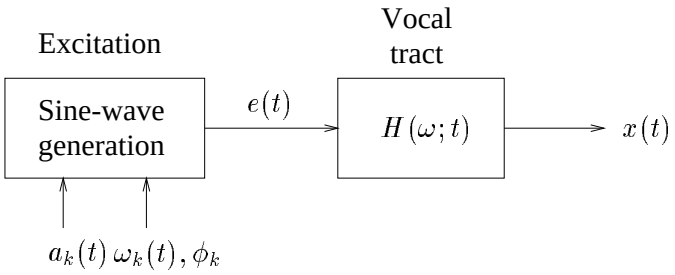


Figure 2.12: *The sinusoidal speech model (after [4]).*

is the output generated by passing an excitation waveform $e(t)$ through a linear filter $h(t)$, the excitation is defined as a sum of sine waves of arbitrary amplitudes, frequencies and phases:

$$e(t) = \sum_{k=1}^{N} a_k(t) \cos[\Omega_k(t)] \tag{2.9}$$

where for the kth sine wave, the excitation phase $\Omega_k(t)$ is the integral of the time–varying frequency $\omega_k(t)$:

$$\Omega_k(t) = \int_0^t \omega_k(\sigma)\, d\sigma + \phi_k \tag{2.10}$$

$a_k(t)$ is the time–varying amplitude, ϕ_k is a fixed phase offset and N is the number of sine waves at time t.

The time–varying vocal tract transfer function can also be written in terms of time–varying amplitude $M(\omega;t)$ and phase $\psi(\omega;t)$ as:

$$H(\omega;t) = M(\omega;t) \exp[j\psi(\omega;t)] \tag{2.11}$$

Therefore along each frequency trajectory $\omega_k(t)$, the system amplitude and phase are given by $M_k(t) = M[\omega_k(t);t]$ and $\psi_k(t) = \psi[\omega_k(t);t]$. The speech waveform is generated by passing $e(t)$ through the vocal tract representation, giving

$$x(t) = \sum_{k=1}^{N} A_k(t) \cos[\theta_k(t)] \tag{2.12}$$

where $A_k(t) = a_k(t)M_k(t)$ and $\theta_k(t) = \Omega_k(t) + \psi_k(t)$ are the amplitude and phase of each sine–wave component along the frequency trajectory $\omega_k(t)$. This is shown in Figure 2.12.

This model is applied to real speech by blocking the signal into overlapping frames and then calculating the 1024 point short–time Fourier transform of each frame. Then the excitation frequencies $\omega_k(m)$ are estimated by picking the peaks of the STFT magnitude, where m refers to the mth frame (the variables are now dependent on m, the frame variable, rather than t). The amplitude and phase, $A_k(m)$ and $\theta_k(m)$, of each sine wave is then given by the amplitude and phase of the STFT at the estimated peaks.

Synthesis reverses the process, but also includes tracking of the frequency trajectories $\omega_k(m)$ from one frame to the next, since these will change. A nearest–neighbour approach which includes a birth–death process performs this matching, allowing sine waves to come and go in time. The amplitude and phase components are then interpolated across frame boundaries. The final result is reported to be perceptually very close to the original for both voiced and unvoiced speech.

In order to perform time scaling, the system parameters corresponding to the vocal tract articulators are modified so that they move faster or slower in time, and the excitation parameters are modified to stretch or compress the frequency trajectories while maintaining the pitch. This modifies the basic sinusoidal model as shown in Figure 2.12, to that shown in Figure 2.13. ρ is the rate of change that is imposed. In this example, the rate change is fixed, but it is also possible to have time–varying rate change by making ρ time dependent.

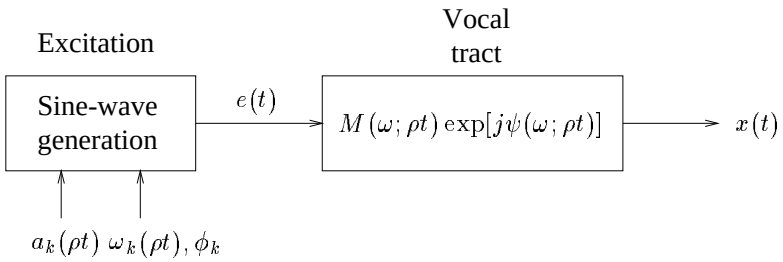


Figure 2.13: *Time–scale modification in the sinusoidal model (after [4]).*

Pitch modification operates by a similar principle, with the scaling of the frequency trajectories in the excitation function. The vocal tract model is unchanged which, as noted by Quatieri and McAulay, does not take into account changes in the vocal tract spectral characteristics that may occur in human pitch modification. Figure 2.14 shows the model for pitch modification with a scaling factor β. Again, time–varying pitch modification is possible by making β time dependent.

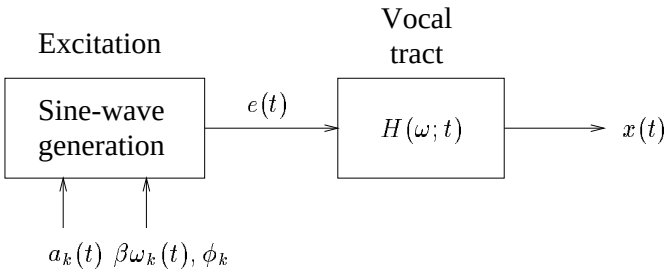


Figure 2.14: *Pitch modification in the sinusoidal model (after [4]).*

Time scaling with a factor of $0.5 \leq \rho \leq 3$, and pitch modification with a factor of $0.5 \leq \beta \leq 2$ is reported to give good results without artifacts. However there is some loss of naturalness in pitch modification, probably due to the vocal tract parameters being held constant. Some "tonality" is also reported for large time scaling. The temporal structure of the speech is also changed in this technique, although later work extended it to include shape invariance by including pitch marker information [48]. A number of variations on this basic method have been proposed, mainly based around incorporating a different, noise–like, model for the unvoiced components, while continuing to use the harmonic model for voiced speech (*e.g.* [49–51]). A limitation of all these techniques is that they use the linear model of speech as a basis.

2.6 Summary

In this chapter a brief overview of articulatory and acoustic phonetics has been presented. This provides a framework for later discussions of speech production and the speech signal, especially concentrating on vowel sounds and their characterisation. Following this, the chain of processes required in a text to speech synthesis system has been examined. The emphasis has been placed on methods used to generate the speech signal and the techniques for time–scale and pitch modification, since these are the areas of interest in this work.

Each of the three main synthesis techniques has been shown to have short–comings: the great complexity of articulatory synthesis; the reliance on linear source–filter modelling in waveform synthesis; the difficulty in choosing the best segments from a database in concatenative synthesis, and the subsequent problems of imposing prosody without creating artifacts. Current TTS systems do produce good quality speech, but still do not sound completely natural. Each stage requires improvement, including the speech signal generation. Therefore, there is still a need to search for improved synthesis techniques which can generate natural–sounding speech with the required prosody.

Chapter 3
Nonlinear dynamics and speech

3.1 Introduction

In this chapter, the concepts of nonlinear systems are introduced, since they will form the backbone of the work in this thesis. Firstly the theory of dynamical systems is presented to provide a background for subsequent work. This discussion includes chaos, reconstructing dynamical systems from a time series, dimension, and the use of Lyapunov exponents, which indicate if a system is chaotic.

Following this, speech is demonstrated to be a nonlinear process, giving the motivation to pursue speech processing from this perspective. Modelling speech as a nonlinear dynamical system is also discussed.

Finally, previous work on measuring the Lyapunov exponents in speech signals is examined. Results on measuring chaos in speech are still contradictory, and some new work is presented that may help to explain why this is so.

3.2 Dynamical systems

Dynamical theory provides a framework with which nonlinear deterministic systems can be examined. Important steps in the analysis are to find the system dimension; reconstruct from available data a representation of the system; investigate if chaos is present and determine the predictability of the system. The theory is now well enough established that a number of excellent tutorial papers [52–55] and books [56–58] exist, which give a thorough background to the subject. The following will be limited to an attempt to show some basic principles and examples which will lead on to subsequent work. For further detail, the reader should refer to the above references.

3.2.1 Dynamical theory

A dynamical system may be defined, for the continuous time case, by the equation:

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}) \tag{3.1}$$

where $\mathbf{x}$ is a vector of length d, defining a point in a d–dimensional space, $\mathcal{F}$ is some function (linear or nonlinear) operating on $\mathbf{x}$, and $\dot{\mathbf{x}}$ is the time derivative of $\mathbf{x}$. This system is deterministic, in that it is possible to completely specify its evolution, or flow of trajectories in the d–dimensional space, given the initial starting conditions.

Alternatively, for discrete time, the dynamical system can be defined as a map:

$$\mathbf{x}_{n+1} = \mathcal{G}(\mathbf{x}_n) \tag{3.2}$$

where $\mathbf{x}_j$ is again the d length vector, at time step j, and $\mathcal{G}$ is the operator function. Given the initial state, $\mathbf{x}_0$, it is possible to calculate the value of $\mathbf{x}_k$ for any $k > 0$.

A simple example of a map is the quadratic map:

$$x_{n+1} = \alpha - x_n^2 \tag{3.3}$$

where α is a control parameter. Plotting iterations of x_{n+1} against x_n gives the evolution of the solution, as seen in Figure 3.1. This type of plot is known as a phase portrait. This example also provides a stepping stone to defining the number of degrees of freedom of a system. Here knowledge of the variable x_n completely specifies the system; hence it can be seen that a quadratic map only has one degree of freedom. Now consider another simple system, that of a point mass on an ideal spring, *i.e.* Hooke's Law. The motion, $x(t)$, of this continuous time system is given by

$$x(t) = x_0 \cos \omega t + \frac{\dot{x}_0}{\omega} \sin \omega t \tag{3.4}$$

where

$$\omega = \sqrt{\frac{k}{m}}$$

ω is the angular frequency of the oscillations, found from the spring constant, k, and the mass

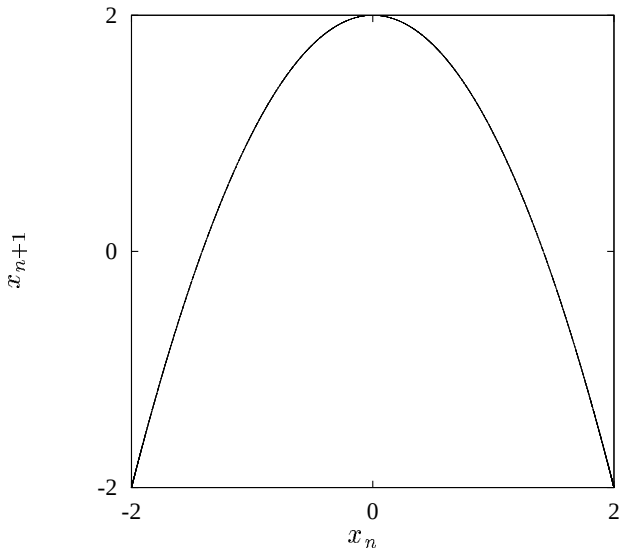


Figure 3.1: *Phase portrait for the quadratic map with* $\alpha = 2$.

of the particle, m. x_0 and $\dot{x}_0$ are the initial conditions at time $t = 0$. The velocity of the system is found by differentiating Equation 3.4:

$$\dot{x}(t) = -\omega x_0 \sin \omega t + \dot{x}_0 \cos \omega t \tag{3.5}$$

The motion and the velocity completely specify the system, so there are two degrees of freedom. Plotting x against $\dot{x}$ gives the phase portrait, as shown in Figure 3.2. This plot is also called the state space of the system, since it can be used to specify the state of the system at any moment in time.

If the spring was not ideal, then energy would be dissipated from the system through time. This would lead to a phase portrait of the type shown in Figure 3.3*a*. When all energy has been dissipated, the particle will be at rest, corresponding to the origin of the phase portrait. Hence this system has an *attractor* at the origin, since any damped (undriven) harmonic oscillator will converge towards it. This is known as a point attractor. In contrast, the addition of some driving force will lead to a closed curve, or limit cycle, attractor, as seen in Figure 3.3*b*. At this point the no–intersection theorem should be mentioned. This defines that two distinct state space trajectories cannot intersect, nor can a single trajectory cross itself [57]. This is due to the fact that the system is deterministic: the location in state space defines the state of the system and how it will evolve. If two trajectories cross, then this implies two possible system evolutions from one point, which is impossible for a deterministic system.

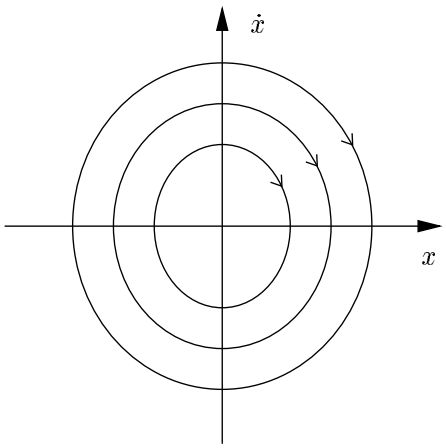


Figure 3.2: *Phase portrait for a point mass on an ideal spring. Each ellipse is a state space trajectory, associated with the amount of energy in the system.*

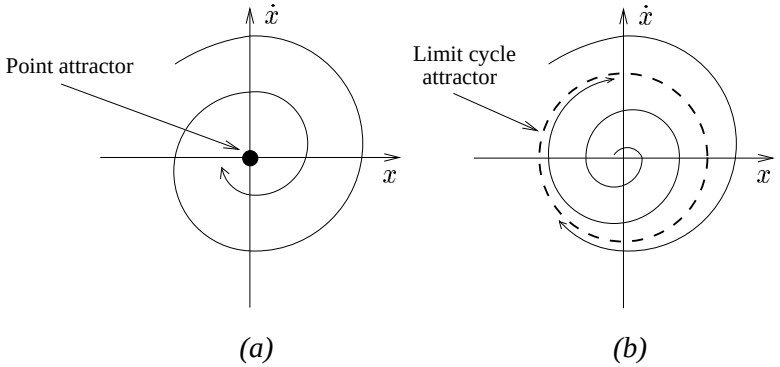


Figure 3.3: *(a) Point and (b) limit cycle (shown in dashed bold) attractors for a harmonic oscillator.*

3.2.2 Chaos

It is somewhat surprising to find that certain dynamical systems, with system equations not much more complicated than the previous examples, can exhibit extremely complex behaviour. Should only one variable be available for observation, then the resulting time series may seem entirely random; indeed the systems responsible were often written off as random and unpredictable. The discovery of chaos has changed this view–point dramatically. Consider the Lorenz equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= -xz + rx - y \\ \dot{z} &= xy - bz \end{aligned} \tag{3.6}$$

where σ, r and b are adjustable parameters. The system (derived as a model of fluid flow) consists of three coupled equations. By comparison to Equation 3.1, this can be seen as a nonlinear dynamical system. It is also autonomous, in that time is not part of the description. Plotting just one variable, such as x shown in Figure 3.4, would lead to the conclusion of random behaviour. However, plotting the state space (*i.e.* x, y and z), as in Figure 3.5, reveals a definite structure.

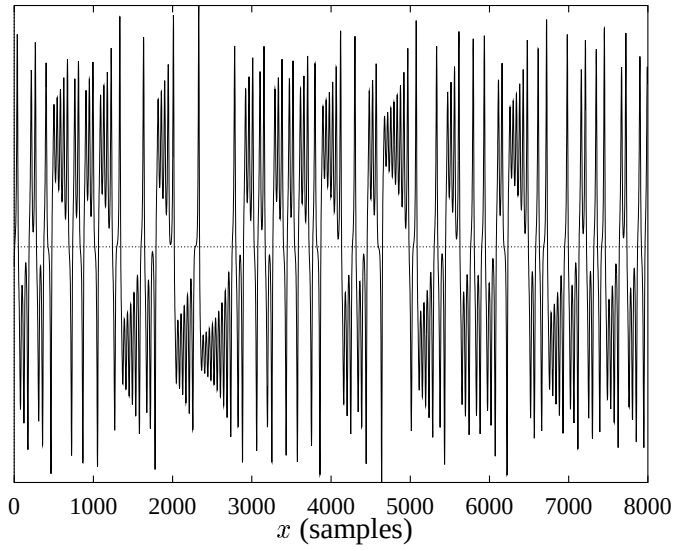


Figure 3.4: *Time series from the variable x of the Lorenz system.*

How can a deterministic dynamical system exhibit such behaviour? The answer lies in the

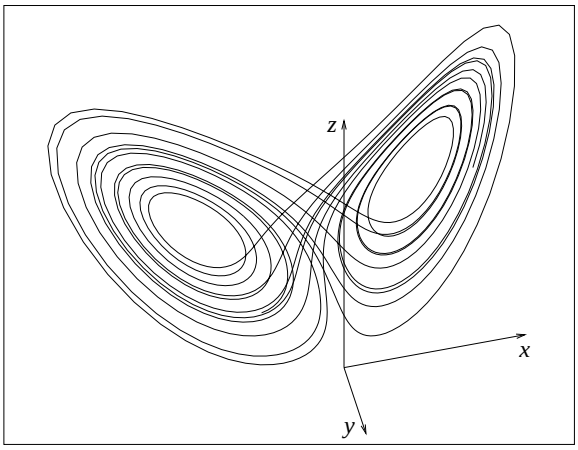


Figure 3.5: *The Lorenz attractor in 3D state space, with* $\sigma = 16.0$, $r = 45.9$ *and* $b = 4.0$.

sensitivity to initial conditions [52]. Closer inspection of the Lorenz attractor reveals that the trajectories in state space remain within a bounded area (the manifold), but never the less diverge by twisting and wrapping around each other. Thus two trajectories that are close to each other at some time t will eventually become separated. Therefore, two sets of initial conditions that are very close together to start with (*e.g.* to machine precision) will have trajectories very close together in the manifold, but not identical. They will become separated, hence leading to differing behaviour of the system. This is what is termed chaos, and attractors with this behaviour are often called strange attractors. How quickly the trajectories diverge is important, since it provides a limit on the short–term predictability of the system. It is important to note that although infinite prediction of a chaotic system is not possible, short–term prediction should be possible to some extent. Later, Lyapunov exponents will be shown to give a measure of how predictable a system is.

As a further example of chaos, a discrete time system is also examined: the Hénon map, described by the equations:

$$\begin{aligned} x_{n+1} &= 1 + y_n - \alpha x_n^2 \\ y_{n+1} &= \beta x_n \end{aligned} \tag{3.7}$$

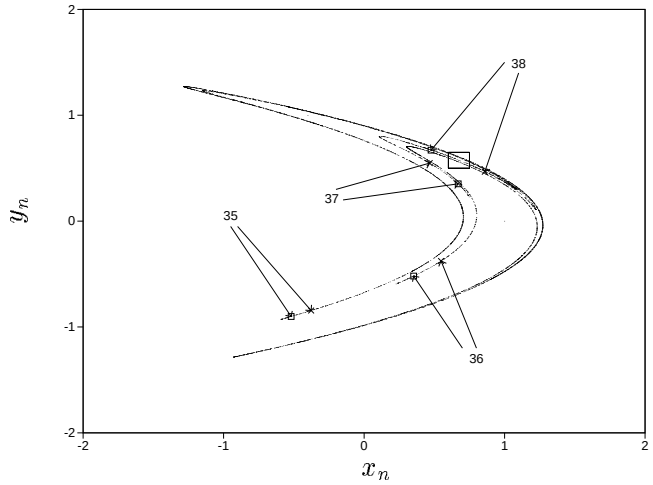


Figure 3.6: *The Hénon attractor, showing the divergence of trajectories, with* $\alpha = 1.4$ *and* $\beta = 0.3$.

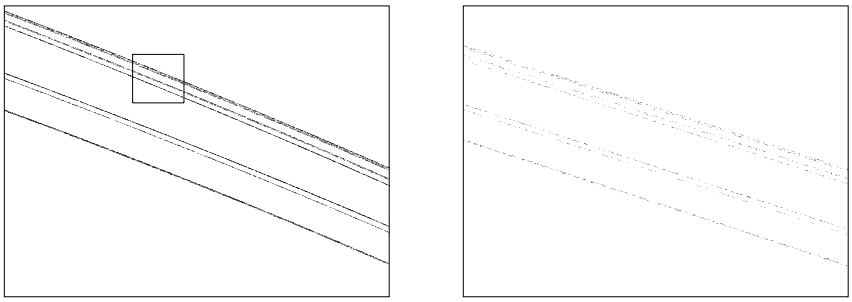


Figure 3.7: *Zooming in on the Hénon attractor, showing its fractal nature.*

where α and β are control parameters. This example is useful, since it clearly demonstrates both the fractal nature and sensitivity to initial conditions characteristic of strange attractors [58]. Figure 3.6 shows the phase portrait of the Hénon map. It can be seen from this how trajectories diverge within a strange attractor. The iteration of the map was carried out twice, with the same initial conditions, but to different machine precisions: firstly using single precision, and then using double precision. It is clear that even after only 35 iterations there is a significant divergence of the trajectories. Figure 3.7 shows subsequent zooms into the Hénon attractor. At each stage of magnification a further set of self–similar lines appear. This will be true, however many times the attractor is magnified, and is a characteristic of fractals.

3.2.3 Dimension

In the previous section, although the number of degrees of freedom of a system has been described, we have shied away from any mention of dimension. In terms of nonlinear dynamical systems, there are a number of different dimensions, all of which are important and will now be defined.

- *System* dimension refers to the number of variables that are required to describe the system dynamics. This is also the complete number of degrees of freedom of the system. In real world situations, the set of differential equations governing a system will not be known, and may be extremely large.
- *Fractal* dimension covers a wide number of measures which are designed to find the geometric dimension of the attractor. It can take a non–integer value, indicating a strange attractor and the possible presence of chaos.
- *Embedding* dimension is the dimension of the Euclidean space into which a one dimensional time series is projected to create a reconstruction of the system attractor.

The calculation of fractal dimension is important, since it is used to estimate the required embedding dimension when performing phase space reconstruction. Consequently it has been heavily studied, resulting in a range of different techniques to estimate its value. These include capacity dimension (also called box–counting dimension), correlation dimension and information dimension [59, 60].

3.2.4 Embedding

A now classic paper by Takens [61] formalises how the state space of a dynamical system can be reconstructed from a time series of only one observed variable. This theorem states that there will be a one to one mapping between the reconstruction and the actual attractor of the underlying system. If the embedding is carried out in a space of dimension of at least m, where

$$m \geq 2d + 1 \tag{3.8}$$

where d is the phase space dimension of the original attractor, then this mapping is guaranteed. Sauer *et al.* [62] then extended this to show that d can be taken as the fractal dimension of the

original attractor. This means that so–called invariant properties, such as Lyapunov exponents, can be calculated from the phase space reconstruction and yet will still correspond to the actual values of the underlying system attractor. The process of phase space reconstruction from a single time series is called time delay embedding. It can be better explained by starting with the case where it is possible to make m simultaneous measurements of the system, $\mathbf{x}(t) = (x_1(t), \ldots, x_m(t))$. Here

$$\mathbf{x} = \mathbf{F}(\mathbf{v}) = (f_1(\mathbf{v}), \ldots, f_m(\mathbf{v})) \tag{3.9}$$

The measurement variables $\mathbf{x}$ are related to the system variables $\mathbf{v}$ by the measurement function $\mathbf{F}$, and they lie in an m–dimensional *reconstruction* space. With m found from Equation 3.8, Takens' theorem tells us that there is a one to one mapping between $\mathbf{x}$ and $\mathbf{v}$.

Now consider the case where only a single measurement, $x(t)$, is available. $x(t)$ must be related to the full system variables by:

$$x(t) = f(\mathbf{v}(t)) \tag{3.10}$$

where f is the scalar measurement function, since the measurement only depends on the system. Now consider the delay coordinate vector, $\mathbf{H}(\mathbf{v}(t))$ [58]:

$$\mathbf{H}(\mathbf{v}(t)) = (x(t-\tau), \ldots, x(t-m\tau)) = (f(\mathbf{v}(t-\tau)), \ldots, f(\mathbf{v}(t-m\tau))) \tag{3.11}$$

where τ is a delay. As long as the system is invertible, $\mathbf{v}(t-\tau)$ for any $(t-\tau)$ is a function of $\mathbf{v}(t)$. This allows us to define $h_1(\mathbf{v}(t)) = f(\mathbf{v}(t-\tau)), \ldots, h_m(\mathbf{v}(t)) = f(\mathbf{v}(t-m\tau))$. Writing $\mathbf{x}(t) = (x(t-\tau), \ldots, x(t-m\tau))$, Equation 3.10 becomes:

$$\mathbf{x} = \mathbf{H}(\mathbf{v}) \tag{3.12}$$

where $\mathbf{H}(\mathbf{v}) = (h_1(\mathbf{v}), \ldots, h_m(\mathbf{v}))$. Clearly $\mathbf{H}$ is a special case of $\mathbf{F}$ in Equation 3.9, and provides, subject to a few caveats specified by Takens, a one to one mapping between the embedded reconstruction and the original system attractor.

In practice, forming the reconstructed trajectory matrix is relatively simple, and involves moving a window of length m through the data to form a series of $\mathbf{x}$ vectors. So for the discrete

time series $x_n := (x_0, x_1, x_2, \ldots, x_i, \ldots)$, the trajectory matrix $\mathbf{X}$ is:

$$\mathbf{X} = \begin{pmatrix} x_0 & x_\tau & \ldots & x_{(m-1)\tau} \\ x_1 & x_{(1+\tau)} & \ldots & x_{(1+(m-1)\tau)} \\ x_2 & x_{(2+\tau)} & \ldots & x_{(2+(m-1)\tau)} \\ \vdots & & & \vdots \end{pmatrix} \tag{3.13}$$

Singular value decomposition

In real systems, where noise is an issue, the reconstruction produced by time delay embedding inherits the noise of the time series which will adversely affect the results of any subsequent analysis. The singular value decomposition (SVD) embedding technique [63] was developed to resolve this problem. The principle behind it is to partition the state space into two subspaces, one containing the signal and the other the noise. As a first step, the time delay embedding matrix $\mathbf{X}$ is formed with $\tau = 1$ sample and a window length of w. w will be referred to as the SVD window length and is chosen to be much greater than the supposed required embedding dimension. The singular value decomposition of $\mathbf{X}$ is defined by:

$$\mathbf{X} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}} \tag{3.14}$$

where $\mathbf{W}$ is diagonal and contains the singular values $w_0 > w_1 > w_2 > \ldots \geq 0$ and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and contain the singular vectors associated with $\mathbf{W}$. The singular values are the root mean square projections onto the basis vectors, and the number of non–zero terms should be equal to the number of active degrees of freedom of the system, although the noise will generate extra non–zero values. However, if the last elements of $\mathbf{W}$ are much smaller than their predecessors, it should be legitimate to discard the final columns, thus reducing the dimension and removing noise effects [64]. Therefore the reduced trajectory matrix can be written as:

$$\mathbf{X}' = \mathbf{X}\mathbf{V}_d \tag{3.15}$$

where $\mathbf{V}_d$ only contains the columns of $\mathbf{V}$ corresponding to the significant values of $\mathbf{W}$.

Mutual information and entropy

To perform a time delay embedding, the delay τ between samples must be specified. An optimum choice will result in the phase space reconstruction being fully opened out, which will improve any subsequent analysis. Mutual information has been proposed as a method to choose τ. The mutual information between the measurement a_i, drawn from a set $A = \{a_i\}$, and the measurement b_j, drawn from the set $B = \{b_j\}$, defines the amount learnt about b_j by the measurement of a_i. Averaging over all of the measurements gives the average mutual information between the set of A measurements and the set of B measurements. Expressed in bits, this is given by [65]:

$$I(A,B) = \sum_{a_i,b_j} \mathrm{P}_{AB}(a_i,b_j) \log_2 \left[\frac{\mathrm{P}_{AB}(a_i,b_j)}{\mathrm{P}_A(a_i) P_B(b_j)} \right] \tag{3.16}$$

where $\mathrm{P}_{AB}(a,b)$ is the joint probability density of measurements A and B, resulting in values a and b. $\mathrm{P}_A(a)$ and $\mathrm{P}_B(b)$ are the individual probability densities for the measurements A and B. $I(A,B)$ measures the dependence between the measurements A and B. Substituting for A and B with $x(n)$ and a delayed version $x(n+\tau)$ gives

$$I(\tau) = \sum_{x(n),x(n+\tau)} \mathrm{P}(x(n),x(n+\tau)) \log_2 \left[\frac{\mathrm{P}(x(n),x(n+\tau))}{\mathrm{P}(x(n))\mathrm{P}(x(n+\tau))} \right] \tag{3.17}$$

which gives the mutual information between points separated by τ. Fraser and Swinney [66] suggest that the first minimum of the mutual information *vs.* τ plot should be used to determine the best value of τ for embedding. A justification for this is that the value used should allow $x(n)$ and $x(n+\tau)$ to be independent enough to form a useful delay vector, but not so independent as to be unrelated.

Mutual information is also used to estimate the entropy of a system, which can be used as a measure of chaotic behaviour. This involves the calculation of the mutual information at higher dimensions than the one dimensional case considered above. Following Bernhard and Kubin [67], the following definitions are made:

- $\mathbf{x}_m^\tau(n) = (x(n), x(n+\tau), x(n+2\tau), \ldots, x(n+(m-1)\tau))$
- X represents the ensemble $x(n)$
- $X(\delta\tau)$ specifies the ensemble of $x(n+\delta\tau)$

- $\mathbf{X}_m^\tau$ denotes the ensemble of vectors $\mathbf{x}_m^\tau(n)$

Then, given an m–dimensional vector ensemble generated (via Takens theorem) from one observed output, the mutual information $I(X(m\tau), \mathbf{X}_m^\tau)$ between the ensemble $X(m\tau)$ and the vector ensemble $\mathbf{X}_m^\tau$ is given by:

$$I(X(m\tau), \mathbf{X}_m^\tau) = \sum_{X(m\tau), \mathbf{X}_m^\tau} \mathrm{P}(X(m\tau), \mathbf{X}_m^\tau) \log_2 \left[\frac{\mathrm{P}(X(m\tau), \mathbf{X}_m^\tau)}{\mathrm{P}(X(m\tau))\mathrm{P}(\mathbf{X}_m^\tau)} \right] \tag{3.18}$$

This can be calculated by a recursive algorithm, such as those implemented by Fraser [68] or Bernhard and Kubin [69].

If m is set high enough, then the mutual information value will saturate, since it has reached the maximum value of predictable information for the chosen time delay τ. Then, using two different time delays, τ_1 and τ_2, the entropy rate can be found as:

$$h = \frac{I(X(\tau_1), \mathbf{X}_{m_s}^{\tau_1}) - I(X(\tau_2), \mathbf{X}_{m_s}^{\tau_2})}{\tau_2 - \tau_1} \tag{3.19}$$

where m_s is the dimension above saturation. If the entropy rate is greater than zero then the system from which it was calculated is chaotic, whereas an entropy rate of zero signifies a non–chaotic system [69]. Additionally, the magnitude of h gives an indication of the predictability of the system. The greater the entropy rate, the greater the information gain per unit time, and hence the less predictable the system.

3.2.5 Lyapunov exponents

One way of detecting chaos that has been previously indicated is the stretching and folding of trajectories in state space. Lyapunov exponents provide a quantitative characterisation of this property and as such are often used as an indicator of the presence or otherwise of chaos. Essentially, across the whole attractor, a positive exponent indicates the divergence of trajectories, whereas a negative exponent indicates convergence. If both positive and negative exponents are present then this indicates a strange attractor and chaos.

Calculating the exponents from a time series, when the system equations are not known, is a difficult process, due mainly to noise and short data lengths. Wolf [70] described a method for finding the largest exponent only. However, Eckmann and Ruelle [55] and Sano and Sawada [71]

propose methods that can calculate the entire spectrum. The basic principle is to embed the time series in order to reconstruct the local dynamics, and then consider the average growth of small perturbations in the trajectories. If small time steps are used, then the linearised evolution of the perturbations is governed by the Jacobian, which is the linear mapping between the trajectory tangent vectors. The exponents are then evaluated by averaging over the product of the Jacobians along the orbit of the state space reconstruction. As such the global Lyapunov exponents are defined as:

$$\lambda_i = \lim_{n \to \infty} \frac{1}{n} \ln \left(\mathrm{eig} \prod_{p=1}^{n} J(p) \right) \qquad 1 \leq i \leq d \tag{3.20}$$

where $J(k)$ is the kth Jacobian around the d–dimensional phase space reconstruction.

Most subsequent work in this area has continued in this vein, with improvements such as better approximation of the Jacobian [72], and noise–reduction for real world signals [73].

It should be noted that local Lyapunov exponents, which indicate local instabilities around the phase space structure, as opposed to global behaviour, can also be calculated [74]. These are discussed further in a later chapter.

3.3 Speech as a nonlinear dynamical system

Speech generation has classically been modelled as a linear system, which provides a convenient and simple mathematical formulation. However, a number of nonlinear effects are present in the physical process which limit the effectiveness of the linear model. An improved approach may be to view speech as a nonlinear dynamical system.

3.3.1 Speech production

Speech production is an extremely complex process. It can be described in basic terms as follows. Air pressure is generated by the lungs and this pressure wave is modulated as it flows through the larynx. Essentially, the larynx is made up of two almost symmetric masses known as the vocal folds which are capable of closing completely together or, as they move apart, creating a triangular opening called the glottis. During normal respiration and the production of so–called unvoiced sounds, air passes freely through the glottis. When the vocal folds vibrate

in a quasi–periodic manner then voiced sounds are produced. The frequency of this excitation is known as the fundamental frequency. The resulting glottal waveform excites the vocal tract, which is the region extending from the larynx to the lips. Different configurations of the vocal tract will result in different modulations of the glottal waveform and thus produce specific sounds [75].

3.3.2 Nonlinearities in speech

There are known to be a number of nonlinear effects in the speech production process. Firstly, it has been accepted for some time that the vocal tract and the vocal folds do not function independently of each other, but that there is in fact some form of coupling between them when the glottis is open [76]. This can cause significant changes in formant characteristics between open and closed glottis cycles. [77]. More controversially, Teager and Teager [78] have claimed (based on physical measurements) that voiced sounds are characterised by highly complex air flows in the vocal tract involving jets and vortices, rather than well behaved laminar flow. Turbulent flow of this nature is also accepted to occur during unvoiced speech, where the generation of sound is due to a constriction at some point in the vocal tract [12]. In addition, the vocal folds will themselves be responsible for further nonlinear behaviour, since the muscle and cartilage which comprise the larynx have nonlinear stretching qualities. Such nonlinearities are routinely included in attempts to model the physical process of vocal fold vibration, which have focussed on two or more mass models [25, 26, 79], in which the movement of the vocal folds is modelled by masses connected by springs, with nonlinear coupling. Observations of the glottal waveform reinforce this evidence, where it has been shown that this waveform can change shape at different amplitudes [80]. Such a change would not be possible in a strictly linear system where the waveform shape is unaffected by amplitude changes.

This highly complex process is generally modelled by the linear source–filter speech model [30]. As shown in Figure 3.8, there is a hard switch between voiced and unvoiced sounds. The glottal waveform for voiced sounds is generated from an impulsive periodic signal (often a Dirac pulse train), with period equal to the pitch period, whereas the generation of unvoiced sounds is modelled by white noise. These signals are then applied to a slowly time–varying linear filter whose transfer function represents the contribution of the vocal tract, followed by another filter to represent the radiation from the lips.

In order to arrive at this simplified model, a number of major assumptions are made. These

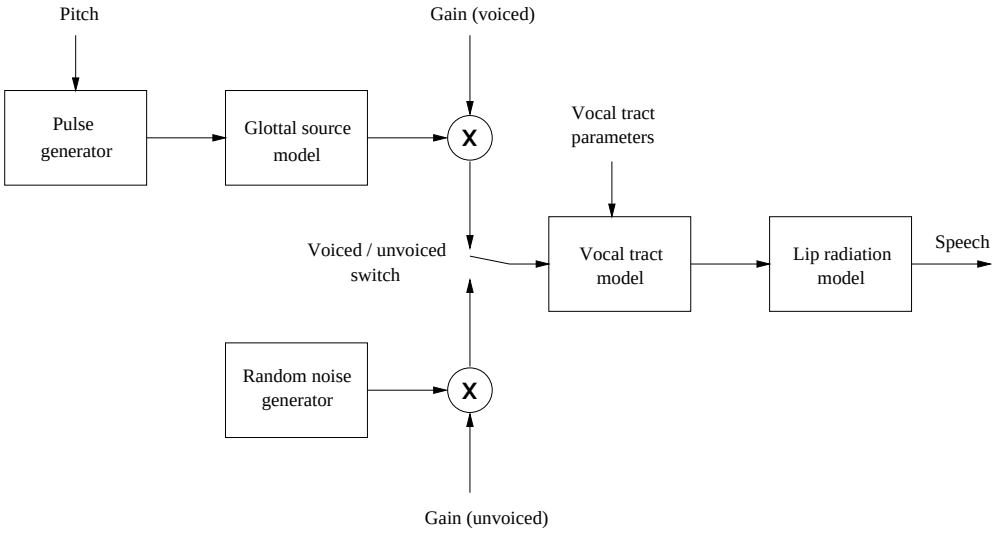


Figure 3.8: *Linear source/filter model of speech production.*

include assuming that:

- the vocal tract and speech source are uncoupled (thus allowing source–filter separation);
- airflow through the vocal tract is laminar;
- the vocal folds vibrate in an exactly periodic manner during voiced speech production;
- the configuration of the vocal tract will only change slowly;

Taking into account the evidence presented above for the nonlinearities in speech, this model clearly has drawbacks. The assumptions made imply a loss of information which, although allowing a simple, mathematically tractable, linear model to be found, means that the full speech signal dynamics can never be properly captured. These inadequacies can be seen in practice in speech synthesis where, at the waveform generation level, current systems tend to produce an output signal that lacks naturalness. This is true even of concatenation techniques which copy and modify actual speech segments, but is even more evident when an attempt is made to linearly model speech production. This occurs because the linear model has not fully modelled the speech signal and so cannot reproduce qualities which make the signal appear human–like, such as pitch variability.

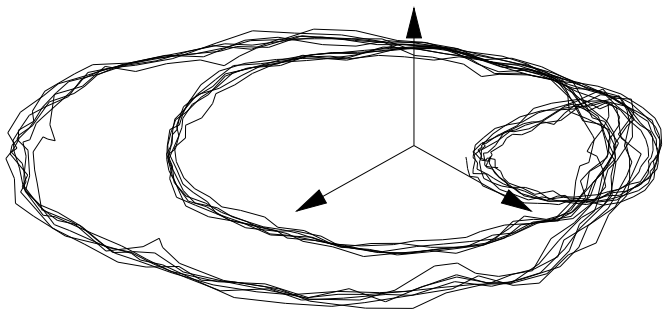


Figure 3.9: *Time delay embedding of the vowel /i/ in 3D space, with $\tau = 0.45$ msec.*

3.3.3 The nonlinear option

If instead of choosing a linear model, the nonlinear option is pursued, then it is not necessary to make many of the assumptions that are required in the linear case. Speech is modelled as the output of a autonomous dynamical system, which only has a low number of active degrees of freedom. By performing a time delay embedding of a stationary vowel sound, this approach appears justified. Figure 3.9 shows the vowel /i/ embedded in a 3D embedding space. τ was set using the mutual information approach at 0.45 msec. Examining the phase space reconstruction from all angles in three dimensions shows that there are no crossings of the manifold, implying a sufficient embedding according to the theory of no intersections. Therefore vowels can be reconstructed onto some form of attractor in a low dimensional space, and so it is possible to model them using a nonlinear dynamical approach by:

$$x_{i+1} = \mathcal{F}\{\mathbf{x}_i\} \tag{3.21}$$

where $\mathcal{F}$ is some nonlinear mapping between a vector of previous samples $\mathbf{x}_i$ and the next sample x_{i+1}. This low–dimensionality characteristic is shared by all voiced sounds, but may not be true for unvoiced sounds, and is discussed further later. By Takens theorem, there will be a one–to–one mapping between $\mathcal{F}$ and the underlying nonlinear system. Therefore correctly modelling $\mathcal{F}$ implies that the dynamics of speech production have been captured.

3.4 Speech and chaos

In Sections 3.3.1 and 3.3.2, physical evidence to suggest that speech is a nonlinear process in terms of speech production was presented. This led to a large amount of speculation about

whether the underlying mechanism of speech has any chaotic properties.

3.4.1 Dynamical analyses of speech

One of the first pieces of work on this subject was by Tishby [81], who constructed a nonlinear predictor based on dynamical systems. His most important finding in the context of dynamical analysis was an estimate of the correlation dimension. This was found to be between three and five for voiced speech, whereas unvoiced gave a higher dimension of between five and eight. This latter estimate is, however, unreliable due to the insufficient number of samples. It also seems that the signals analysed were not stationary, which could give misleading results.

Subsequent work does seem to generally back up these findings. Townshend reports a correlation dimension for vowels of 2.9 [82] or 3.3 [10], and both Townshend and Casdagli [83] implement efficient nonlinear predictors based on modelling the local dynamics. The method used by Casdagli can also indicate whether a system is high or low dimensional, and he indicates that speech does appear low dimensional. Again, it would appear that continuous (non–stationary) voiced speech was used in these analyses. However, McLaughlin and Lowry analyse stationary vowel sounds with similar aims [84], and find a correlation dimension of three to five. Using the false nearest neighbours technique [85], they also deduce a required embedding dimension of seven. Additionally, the largest Lyapunov exponent is calculated (by the Wolf algorithm [70]) to be zero, indicating the absence of chaos.

Work using Fraser's mutual information algorithm contradicts this result [67, 86]. This technique, based on information theory, estimates the optimal embedding delay time and also indicates a suitable dimension in which to embed the phase space reconstruction. The results show that an embedding dimension of three is sufficient, and indicate that the extended vowels are chaotic through a non–integer value of correlation dimension (between one and two) and the calculation of a positive entropy rate. This is in agreement with a study by Kumar and Mullick [87], who find both a positive largest Lyapunov exponent and positive entropy, over a large range of phonemes, including vowels.

Tokuda *et al.* have an on–going study of Japanese vowels. Their initial work [88, 89] focussed solely on the Japanese vowel /a/, and led to the tentative conclusion that this vowel does display chaotic behaviour. They make use of principle component analysis (PCA) to decide on a suitable embedding dimension, which they found to be three. They claim to have ob-

served stretching, folding and compression of the state space reconstruction by examination of Poincaré sections, which would indicate chaos. This is further quantified by their finding of a positive–zero–negative Lyapunov spectrum. Finally, they calculate the fractal dimension of speech (using the Lyapunov dimension), which is found to be between two and three. However, as they indicate, the Lyapunov results should be treated with caution, due to possible difficulties in the calculation. In particular, it would seem that the use of the Sano and Sawada algorithm [71] may give misleading results on such real world data, as it will be adversely affected by noise. Further work on more vowel sounds [90] using the method of surrogate data again shows a high degree of nonlinearity, but does not prove chaos is present. They have also searched for deterministic nonlinear structure in the pitch–to–pitch variations of vowels [91]. This comes to a similar conclusion to the work of Aoki and Ifukube [92], who indicated that these variations (called *jitter*) may be fractal.

A separate study of fricatives found these sounds to be higher dimensional, and by calculating the Lyapunov spectra that chaotic behaviour was present [93]. Interestingly, they used vowels as a comparison to their results on fricatives, and here the largest Lyapunov exponent was close to zero, indicating non–chaotic behaviour.

3.4.2 Lyapunov exponent estimation algorithm

The above summary indicates that there is still no complete conclusion to the subject of chaos within vowel sounds. A recent study in this department by Banbrook [5] attempted to resolve this issue, by developing a robust algorithm for the calculation of Lyapunov exponents, and then applying this to a carefully collected database of stationary vowel sounds.

Algorithms capable of calculating the entire Lyapunov spectrum from a time series were developed in 1985 by both Eckmann and Ruelle [55] and Sano and Sawada [71]. These use an average of the Jacobians of the local dynamics around the phase space reconstruction, but are very susceptible to noise. A significant advance was made by Darbyshire and Broomhead [73], which reduces noise problems by using singular value decomposition. The technique developed in this department by Banbrook, Ushaw and McLaughlin [94], uses the Darbyshire and Broomhead method as a basis.

The algorithm

In this section, the algorithm proposed by Banbrook, Ushaw and McLaughlin is described in some detail, since it forms the basis of a synthesis technique to be discussed later. A complete description is to be found in [5].

As with all Lyapunov exponent estimation techniques, the algorithm is complex. To produce correct results, it relies on the appropriate choice of a number of parameters, which must be determined experimentally. Figure 3.10 shows an overview of the functionality of the algorithm. As can be seen, the process can be broken down into a number of distinct steps, which are described below.

1. *Embedding* The speech signal is embedded into d–dimensional state space using SVD embedding.

2. *Choose neighbourhood* The practical method to calculate Lyapunov exponents from experimental data involves constructing a d–dimensional hyper–sphere containing a set number of points in state space, and then observing the evolution of this hyper–sphere. The hyper–sphere is constructed by finding a sufficient number of points within the locally linear space around a chosen point, $\mathbf{x}_i$, and this forms the neighbourhood vector set Γ_i. A check is made that points within Γ_i are not false nearest neighbours, by ensuring that they evolve along suitable trajectories. Now the neighbourhood matrix $\mathbf{B}_i$ can be formed as

$$\mathbf{B}_i = \begin{pmatrix} \gamma_1 - \mathbf{x}_i \\ \gamma_2 - \mathbf{x}_i \\ \vdots \\ \gamma_M - \mathbf{x}_i \end{pmatrix} \tag{3.22}$$

   where γ_n is the nth entry of Γ_i. It is necessary to reform this matrix after a number of steps, a, since the points will become so far separated as to break the assumption of local linearity. The choice of this parameter, called the number of evolve steps, as well as the number of points in the hyper–sphere, must be made experimentally.

3. *Calculate tangent map* If the step between $\mathbf{B}_i$ and the evolved neighbourhood matrix, $\mathbf{B}_{i+a}$, is small enough to be considered locally linear, then the eigenvalues of the tangent map between the two matrices will give the Lyapunov exponents. The tangent map, $\mathbf{T}_i$,

defines the linear transformation between $\mathbf{B}_i$ and $\mathbf{B}_{i+a}$:

$$\mathbf{B}_{i+a}^{\mathrm{T}} = \mathbf{T}_i \mathbf{B}_i^{\mathrm{T}} \tag{3.23}$$

The evolved neighbourhood matrix $\mathbf{B}_{i+a}$ is constructed as per $\mathbf{B}_i$, but the neighbourhood vector Γ_{i+a} contains points a steps on from Γ_i. Equation 3.23 can be solved for $\mathbf{T}_i$ by writing

$$\mathbf{B}_i^{+} \mathbf{B}_{i+a} = \mathbf{T}_i^{\mathrm{T}} \tag{3.24}$$

where $\mathbf{B}_i^{+}$ is the pseudo–inverse of $\mathbf{B}_i$, calculated by singular value decomposition (see Appendix A).

4. *Average exponents* The tangent map $\mathbf{T}_i$ gives a description of the local system dynamics. In order to calculate the global dynamics, a series of these tangent maps across the whole phase space structure needs to be combined together. This is achieved using the QR–factorisation technique, which allows any matrix $\mathbf{A}$ to be written as

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \tag{3.25}$$

where $\mathbf{Q}$ has orthogonal columns and $\mathbf{R}$ is a square upper–right triangular matrix with positive values on the diagonal. This allows the global Lyapunov exponents to be calculated as [55]:

$$\lambda_i = \lim_{n \to \infty} \frac{1}{n} \sum_{j=1}^{n} \log(\mathbf{R}_j)_{ii} \qquad 1 \leq i \leq d \tag{3.26}$$

where $\mathbf{R}_k$ is the kth $\mathbf{R}$ matrix, corresponding to the kth tangent map, and n is the total number of tangent maps.

As mentioned above, there are a number of parameters that must be selected in order to obtain correct results from the algorithm. These are:

- number of points in hyper–sphere, which should be as large as possible without breaking the assumption of local linearity;
- embedding dimension. This is indicated by other (*e.g.* correlation dimension) methods,

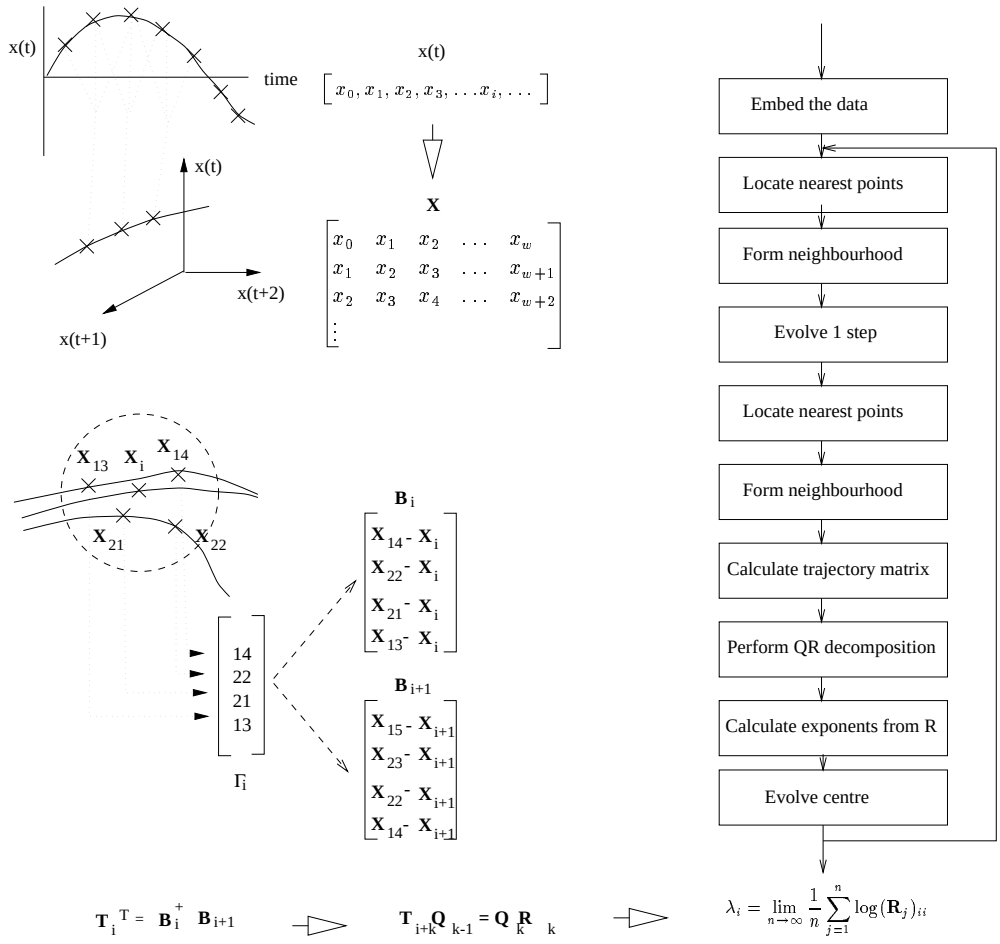


Figure 3.10: *Overview of the Lyapunov exponent estimation algorithm by Banbrook* et al. *(after [5]).*

but must be experimented with;

- size of SVD window, in order to remove any noise present;
- re–initialisation delay. This is a trade–off between a large number of steps which will show greater change, against the fact that the neighbourhood will become very dispersed and no longer be locally linear;
- total number of tangent maps, which should be as large as possible, but involve at least one revolution of the phase space reconstruction;
- number of data points, which also should be large so as to adequately fill the areas of state space covered by the reconstruction.

The problem of how to select appropriate values is discussed later.

The algorithm has an additional noise–reduction feature. This involves a different formulation of the $\mathbf{B}_i$ matrix, which allows an averaging operation to be performed [94]. Experiments on noisy Lorenz data show a considerable improvement in the estimation of the positive Lyapunov exponent using this technique [5].

The data

In normal connected speech, the time that an individual vowel is sustained is typically very short. For an accurate calculation of the Lyapunov exponents, long data lengths are required, which means that normal spoken speech is unsuitable. Therefore a database of extended vowel sounds was collected. This involved words being spoken in 'consonant–vowel–consonant' (CVC) format, with the vowel being sustained for approximately two seconds. During this period both pitch and amplitude were kept as constant as possible.

Fifteen subjects were recorded, ten male and five female. Each subject read sixty utterances, which consisted of twelve different words each repeated five times. The subjects were requested to extend the vowel portion of each utterance, which was the only part recorded. The recordings were made at 22.05 kHz, with 16 bit accuracy. Manual checking was carried out to ensure that the signals were indeed stationary. Following this, the position of the vowels on a formant chart was examined, and any ambiguous cases were discarded. Table 3.1 shows the CVC utterances that were recorded.

1 heat (/i/)	11 head (/ɛ/)	21 hut (/ʌ/)	31 caught (/ɔ/)	41 hart (/ɑ/)	51 hot (/ɒ/)
2 hart (/ɑ/)	12 hot (/ɒ/)	22 heat (/i/)	32 head (/ɛ/)	42 hut (/ʌ/)	52 caught (/ɔ/)
3 hut (/ʌ/)	13 caught (/ɔ/)	23 hart (/ɑ/)	33 hot (/ɒ/)	43 heat (/i/)	53 head (/ɛ/)
4 heat (/i/)	14 head (/ɛ/)	24 hut (/ʌ/)	34 caught (/ɔ/)	44 hart (/ɑ/)	54 hot (/ɒ/)
5 hart (/ɑ/)	15 hot (/ɒ/)	25 heat (/i/)	35 head (/ɛ/)	45 hut (/ʌ/)	55 caught (/ɔ/)
6 hit (/ɪ/)	16 hat (/æ/)	26 hurt (/ʊə/)	36 hoot (/u/)	46 hood (/ʊ/)	56 hate (/eɪ/)
7 hood (/ʊɸ/)	17 hate (/eɪ/)	27 hit (/ɪ/)	37 hat (/æ/)	47 hurt (/ʊə/)	57 hoot (/u/)
8 hurt (/ʊə/)	18 hoot (/u/)	28 hood (/ʊ/)	38 hate (/eɪ/)	48 hit (/ɪ/)	58 hat (/æ/)
9 hit (/ɪ/)	19 hat (/æ/)	29 hurt (/ʊə/)	39 hoot (/u/)	49 hood (/ʊ/)	59 hate (/eɪ/)
10 hood (/ʊ/)	20 hate (/eɪ/)	30 hit (/ɪ/)	40 hat (/æ/)	50 hurt (/ʊə/)	60 hoot (/u/)

Table 3.1: *The CVC utterances recorded, after [7]. The underlined portions indicate the vowel sounds that form the stationary vowel database (with corresponding IPA phonetic symbols).*

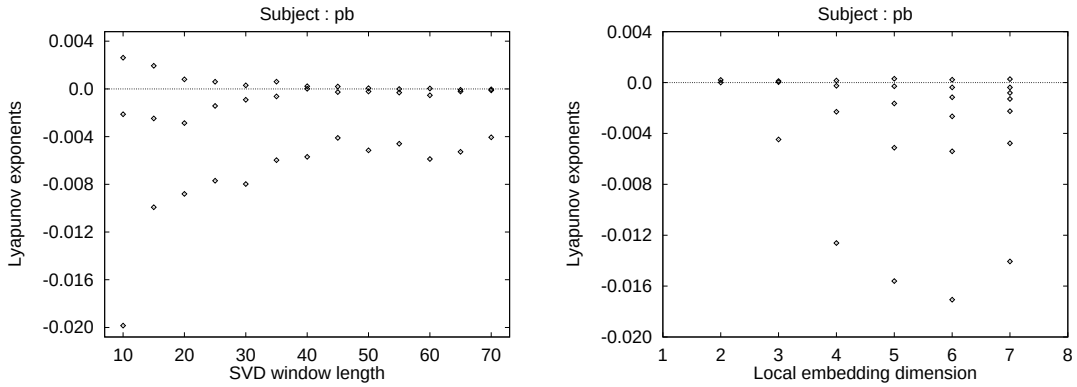


Figure 3.11: *Choosing parameters for Lyapunov algorithm: Exponents for variable SVD window length (vowel /ɒ/) and variable local embedding dimension (vowel /ʊ/). Other parameters are* 200 *neighbours;* 20 *vectors in each neighbourhood set;* 2000 *iterations of* 4 *evolve steps each (after [5]). The exponents are expressed in bits/sample.*

Results

The Lyapunov exponents were calculated for two of the subjects in the database over a wide range of values of the parameters described above in order to arrive at the best estimates for use in the full analysis. Figure 3.11 shows how the exponents vary for varying SVD window length and local embedding dimension for the male speaker PB. Here the stability of the results for SVD window lengths above approximately 40 samples led to the choice of 50 samples for this parameter. The local embedding dimension was chosen as three, since the first two exponents show little change for dimensions above this. This analysis was applied to all parameters in order to arrive at the final set of values. Using these values, the entire database was analysed. Typical results are presented in Figure 3.12, here for the vowel /ɪ/. These show the Lyapunov

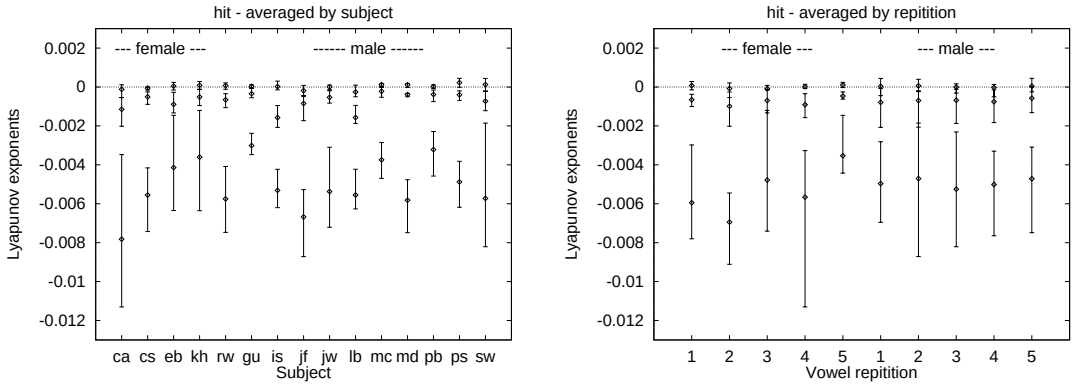


Figure 3.12: *Lyapunov exponents (in bits/sample) for the vowel /ɪ/ using the following parameters: SVD window length* 50*; global embedding dimension* 7*; local embedding dimension* 3*;* 200 *neighbours forming* 20 *vectors in each neighbourhood set;* 2000 *iterations of* 10 *evolve steps each (after [5]).*

spectra averaged by speaker and by repetition through the database. The average value in all cases was found to be one zero and two negative Lyapunov exponents, leading to the conclusion that the data is not chaotic [5]. This result is unchanged by any increase in local embedding dimension, which was increased from three through to seven without any other significant exponents appearing.

3.4.3 Further Lyapunov exponent analysis

The results described above which find that vowels are not chaotic directly contradict the findings from a number of other papers, who maintain that these signals do exhibit chaotic behaviour. In a novel, subsequent analysis, the above technique was applied to extended vowels from a different database[1], which has previously been studied using nonlinear analysis techniques based on the mutual information, which measures the entropy rate of the nonlinear system [67]. This study had found chaotic behaviour in these vowels.

Figure 3.13 shows the results obtained on an example of the vowel /i/ for variable SVD window length and local embedding dimension, respectively. Identical trends to those found by Banbrook are observed, indicating that his results are not dependent on a specific database. For SVD window lengths of above 40 samples, the exponents show little change, indicating stable results. This is in keeping with the value of 50 samples used in the previous analysis. There is

[1]Thanks to G. Kubin and H. P. Bernhard of the Vienna University Of Technology, Austria, for allowing us the use of their database.

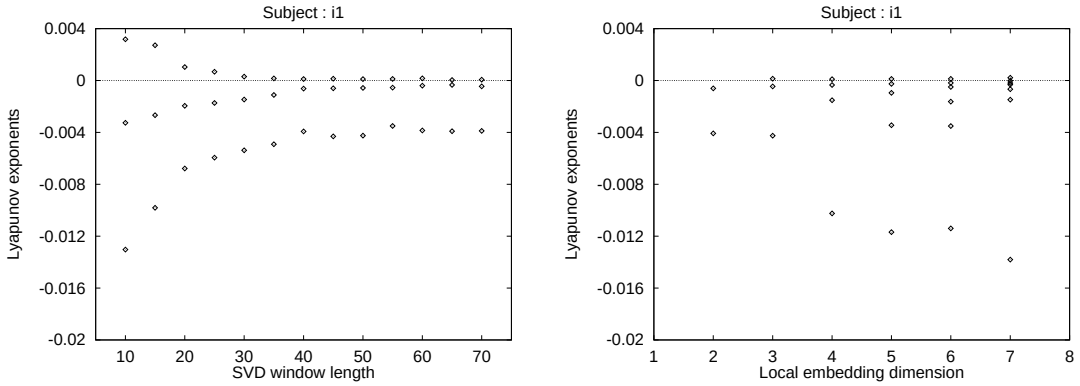


Figure 3.13: *Lyapunov exponents (in bits/sample) for the vowel /i/ from the University of Vienna database: variable SVD window length and variable local embedding dimension. Other parameters are* 220 *neighbours;* 20 *vectors in each neighbourhood set;* 2000 *iterations of* 4 *evolve steps each.*

also no significant change in the first exponent beyond a local embedding dimension of three, which again correlates well with the previous results. Applying the logic used by Banbrook [5], this would lead to the conclusion that the Vienna database does not exhibit any chaotic behaviour, since there is no positive exponent found for the parameter set used (SVD window length of 50; local embedding dimension of 3). However, for shorter SVD window lengths, implying less noise reduction, a positive exponent is seen. Hence it is important to consider what effect the process of singular value decomposition may have on the results.

3.4.4 Effects of singular value decomposition

The Lyapunov exponent estimation algorithm as developed by Banbrook *et al.* relies on the use of SVD to help reduce noise present on the signal. While it has been demonstrated that this gives good results on artificial data such as Lorenz [95], some concern has been expressed that valid parts of the signal may also be removed by this process when applied to speech.[2] This would then alter the values of the Lyapunov exponents that are calculated subsequently, and so could mask the presence of chaos. In an effort to resolve this issue, some novel work was carried out to investigate the effects of SVD on voiced speech signal characteristics.

As explained previously (Section 3.2.4), given the singular value decomposition of the embed-

[2] Private communication with G. Kubin of Vienna University of Technology.

ding matrix $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}} \tag{3.27}$$

the reduced trajectory matrix corresponding to the d most significant values of $\mathbf{W}$ can be written as:

$$\mathbf{X}' = \mathbf{X}\mathbf{V}_d \tag{3.28}$$

It is this matrix $\mathbf{X}'$ that is used in the subsequent analysis, so it is now necessary to question whether any important dynamical information has been lost when discarding the other less significant values. Figure 3.14 shows a comparison of time delay and SVD embedding for the vowel /ɔ/. Some noise reduction can be seen in the SVD case, resulting in a smoother phase space reconstruction, but it is very difficult to assess the quality of the signal.

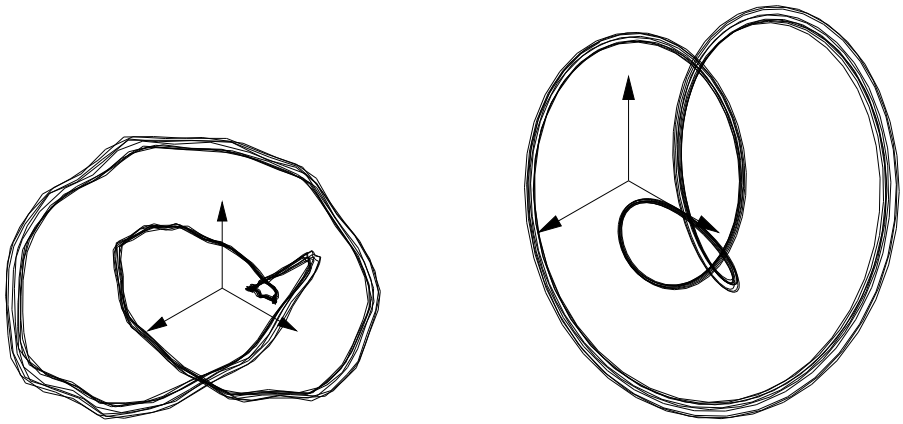


Figure 3.14: *Comparison of time delay ($\tau = 10$ samples) and SVD embedding ($w = 50$ samples) for the vowel /ɔ/.*

For this reason, it is necessary to return back to the time domain where speech–specific characteristics, such as pitch and formant structure can be examined.

As can be seen from Equation 3.27, the elements of $\mathbf{X}$ can be expressed as:

$$X_{ij} = \sum_{k=0}^{w-1} U_{ik} W_k V_{jk} \quad (0 \leq i < N,\ 0 \leq j < w) \tag{3.29}$$

where N is the data length. In forming $\mathbf{X}'$, all w_n, $n >= d$, were discarded, so the SVD

processed trajectory matrix can be written as:

$$\hat{X}_{ij} = \sum_{k=0}^{d-1} U_{ik} W_k V_{jk} \quad (0 \leq i < N,\ 0 \leq j < w) \tag{3.30}$$

Note that the dimensions of $\hat{\mathbf{X}}$ are the same as that of $\mathbf{X}$ and not $\mathbf{X}'$. Since the time delay τ between subsequent values in $\mathbf{X}$ is one sample, the SVD processed one dimensional time series $\hat{\mathbf{x}}$ can be formed from the column vector $\hat{X}_{i0}$ $(0 \leq i < N)$. It is now possible to carry out a direct comparison between the original time series $\mathbf{x}$ and the SVD processed time series $\hat{\mathbf{x}}$.

Twenty–four signals were extracted from the stationary vowel database collected for the Lyapunov analysis [5], consisting of two male and two female speakers pronouncing two examples each of the three cardinal vowels /i/, /ɑ/ and /u/. The above SVD process was then applied to the data to produce a set of SVD processed signals suitable for comparison with the originals. An SVD window length of $w = 50$ samples was used. Across all the signals, it was found that the time structure was not significantly altered, and that the pitch was unchanged. However, examining the frequency spectra shows that the higher frequencies do undergo some attenuation. Figure 3.15 shows an example of an original time domain signal and the resulting SVD processed signal for the vowel /i/ for a male speaker. It is evident that the general structure is preserved, although some fine detail has been removed. The vertical lines are pitch markers, indicating the moment of glottal closure, so giving the instantaneous pitch. These are identical in both signals, indicating that the pitch has not been altered. In Figure 3.16, the attenuation of the higher frequencies is clearly demonstrated over the three vowels, /i/, /ɑ/ and /u/. In this figure the spectral envelope, rather than the full spectrum, for each example is shown. This is for reasons of clarity, since it allows the general trend to be easily observed; in contrast a comparison of the Fourier spectra is difficult, since it is very cluttered. The spectral envelopes were calculated by the autocorrelation linear prediction method, using the estimate $(F_s + 4)$ [96] to give a model with twenty–six poles. It is immediately apparent that the spectral structure has been retained, but that it is attenuated, and that this attenuation increases with frequency. Table 3.2 shows the average attenuation of the first four formants for each of the three vowels studied in the database, and demonstrates that the trends seen in Figure 3.16 are consistent across the whole database.

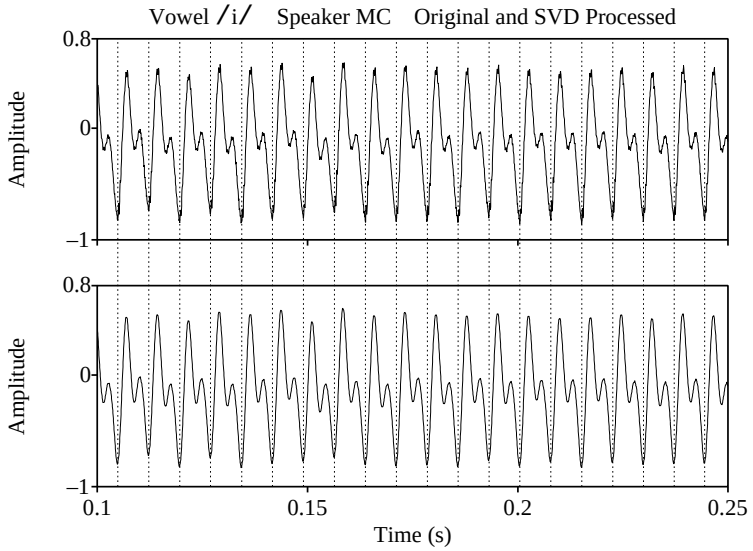


Figure 3.15: *Comparison of original (top) and SVD processed (bottom) time domain signals with pitch markers for the vowel /i/ for male speaker MC, showing the preservation of time and pitch structure.*

Vowel	F1 Attn (dB)	F2 Attn (dB)	F3 Attn (dB)	F4 Attn (dB)
i	1.0	14.1	16.5	17.8
ɑ	0.1	15.3	23.4	24.0
u	0.8	12.5	15.9	20.2

Table 3.2: *Average attenuation of the formants F1–F4 for the vowels /i/, /ɑ/ and /u/ over the database.*

Given that this attenuation of the higher frequencies takes place, it must now be considered how this could affect any subsequent dynamical analysis, such as the calculation of Lyapunov exponents. If SVD is not used then noise may remain on the signal, and this may produce the artifact wherein a positive Lyapunov exponent is found which is in fact zero [73]. The SVD process has been shown to alleviate this problem, thanks to its noise reduction properties, for theoretical (*e.g.* Lorenz) systems [95]. It would appear that it is extremely difficult to find a middle–ground between these two in real world applications: with a Lyapunov analysis of speech not using SVD, the system can never be guaranteed to be noise–free. However, employing SVD does attenuate the higher frequencies, and this may imply the loss of some

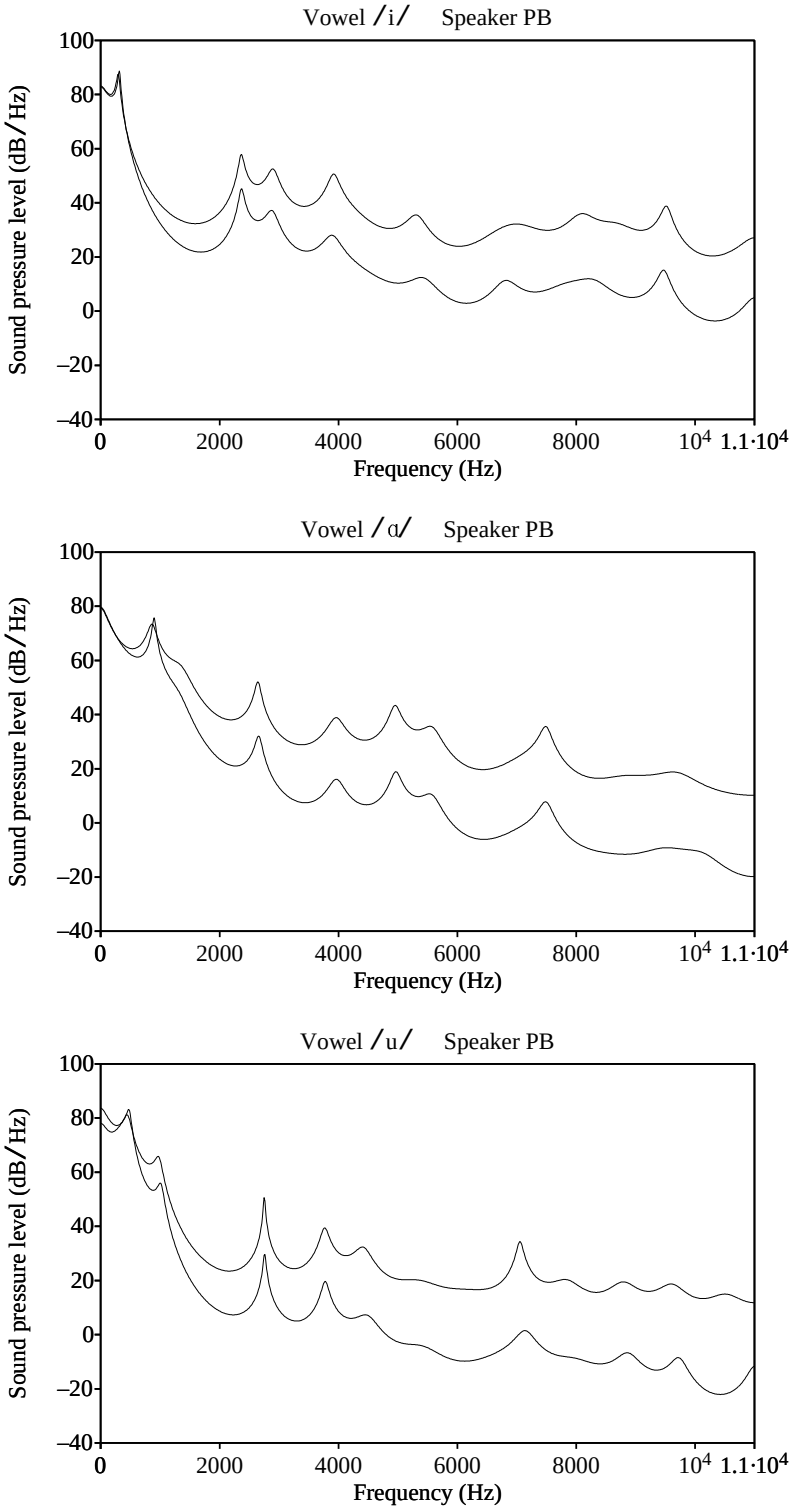


Figure 3.16: *Comparison of LP spectra for the vowels /i/, /ɑ/ and /u/ for male speaker PB. In all cases, the upper curve is the spectral envelope of the original signal and the lower the SVD processed signal.*

important higher frequency information. It is argued that the best currently available Lyapunov exponent calculation techniques have been used. Neither method can give a definite answer to whether the largest Lyapunov exponent is zero, or slightly positive, but taking into account their known limitations, it would seem clear that if voiced speech is at all chaotic, then it is extremely weakly so.

3.5 Conclusion

This chapter has introduced the field of nonlinear dynamical theory, including the concepts of chaos and how to measure it. The traditional model of speech production has been shown to have a number of short–comings and a nonlinear system has been proposed as an alternative. The problem of whether speech (especially vowel sounds) is chaotic has been examined through discussion of previous studies, and some new work has been presented which may help to explain why contradicting results have been found. In terms of speech synthesis, it does not seem that the presence or otherwise of chaos in speech is particularly relevant. The fact that speech can be reconstructed on to a low–dimensional attractor implies that a nonlinear dynamical system can be built to model the dynamics, and thus produce high quality synthesised sounds.

Chapter 4
Poincaré pitch marker

4.1 Introduction

The nonlinear techniques presented up until this point have been mainly concerned with speech analysis, in order to discover possible chaotic behaviour in vowel sounds. In this chapter nonlinear dynamical theory is turned towards the use of Poincaré sections to mark the glottal closure instants (GCI) in voiced speech. This new technique demonstrates the practical use of nonlinear methods in solving real speech processing problems, and also provides a stepping stone towards speech synthesis, since many synthesis systems require these pitch markers in order to carry out pitch modification.

4.2 Epoch marking

Pitch measurement has been a pervasive issue in speech processing for a number of years, as knowledge of the pitch finds use in many applications. In particular, instantaneous pitch detection, also called epoch detection, is very useful in speech synthesis since it provides pitch markers every period which are used for pitch modification. There are many techniques which can calculate an average value of pitch, as documented by Hess [97], but epoch detection is a more demanding task. The principle of epoch marking is seen in Figure 4.1, which shows a short section of voiced speech, with the GCIs marked, and the instantaneous pitch period, T_0. The requirement is to locate the instants in the time domain speech signal at which the glottis is closed.

A variety of existing methods can be employed to locate the epochs. These include:

- Abrupt change detection [98]. Here two linear prediction models are compared. The first is a long–term model which aims to model the long time spectral characteristics of the speech signal. The second is a short–term model which estimates the short time charac-

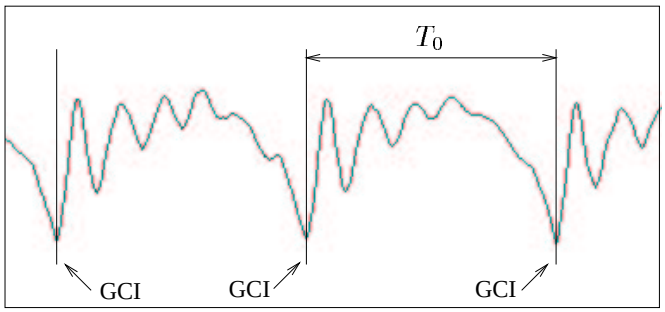


Figure 4.1: *The principle of epoch detection, showing a voiced speech segment with epoch markers and instantaneous pitch,* T_0.

teristics. Just after a glottal closure occurs, the short–term model deviates significantly from the long–term model, thus indicating an epoch.

- Maximum–likelihood epoch detection [99], which assumes that the difference between the observed speech signal and that generated by an all–pole linear model of speech production is a Gaussian process. It then searches for the maximum of the conditional probability density function of this Gaussian process to locate the epoch.

- Dynamic programming [100], in which dynamic programming methods are used to select waveform maxima from a short–time energy–normalised LP residual. Local cost functions are based on peak amplitude and quality, and the output is a set of epochs that globally satisfy the cost constraints over the whole voiced region.

All of these techniques are robust and generally provide good epoch detection. However none are perfect and errors do sometimes occur, for example in difficult areas of the speech signal such as voiced fricatives. The novel technique presented in this chapter should not be viewed as a direct competitor to the methods outlined above. Rather it is an attempt to show the practical application of ideas from nonlinear dynamical theory to a real speech processing problem.

4.3 Application of Poincaré map

A Poincaré map is often used in the analysis of dynamical systems. It replaces the flow of an n–th order continuous system with an $(n-1)$–th order discrete time map [101]. Considering a three dimensional attractor, as shown in Figure 4.2, a Poincaré section slices through the flow of trajectories and the resulting crossings form the Poincaré map. In the example shown,

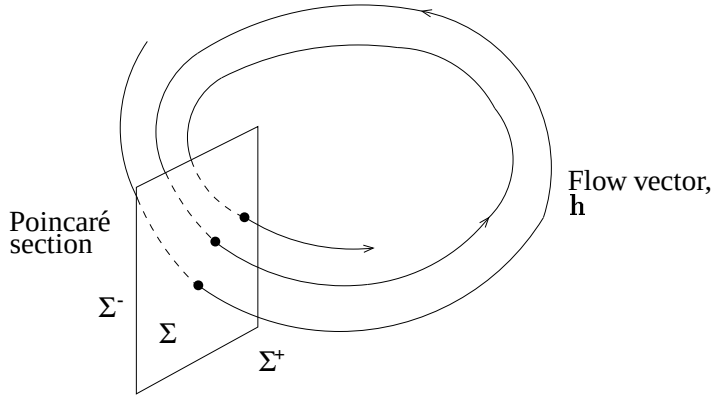


Figure 4.2: *An example of a Poincaré section, producing a (one–sided) Poincaré map.*

the Poincaré section only cuts through the flow once, so that there are only crossings in one direction (from Σ^- to Σ^+). This results in a one–sided map, whereas if the section cuts the flow in both directions a two–sided map is produced.

Re–examining the phase space reconstructions of voiced speech discussed in Chapter 3, it is evident that these three dimensional structures can also be reduced to two dimensional maps.[1] Additionally, these reconstructions are pitch–synchronous, in that one revolution of the phase space reconstruction is equivalent to one pitch period. Clearly this fact can be exploited to mark points in time separated by multiples of the pitch period using the Poincaré section. This has previously been used for cyclostationary analysis and synchronisation [102]; here it is examined for use in epoch marking.

4.4 New epoch–marking algorithm

The algorithm uses the principle outlined above to mark successive epochs. The basic processing steps required for a waveform of N points are as follows:

1. Define the marker x_{GCI} as the position in the time domain signal of a known GCI.
2. Perform an SVD embedding on the signal to generate the phase space reconstruction in 3D state space.

[1] Strictly these phase space reconstructions are discrete time maps and not continuous flows. However it is possible to construct a flow vector between points and use this for the Poincaré section calculation.

3. Calculate the flow vector, $\mathbf{h}$, at the marked point $\mathbf{x}_{\mathrm{GCI}}$ on the reconstruction.
4. Detect crossings of the Poincaré section, Σ, at this point in state space by sign changes of the scalar product between $\mathbf{h}$ and the vector $\mathbf{x}_i - \mathbf{x}_{\mathrm{GCI}}$ for all $1 \leq i \leq N$ points.
5. Points on Σ which are within the same part of the manifold as $\mathbf{x}_{\mathrm{GCI}}$ are the epochs.

When dealing with real speech signals there are a number of practical issues to contend with. Because speech is nonstationary, the input signal must be treated on a frame–by–frame basis, within which the speech is assumed stationary. Finding the correct intersection points on the Poincaré section is also a difficult task due to the complicated structure of the phase space reconstruction. Because of this, additional measures are used to help locate the epoch points. The flow chart shown in Figure 4.3 illustrates the entire process.

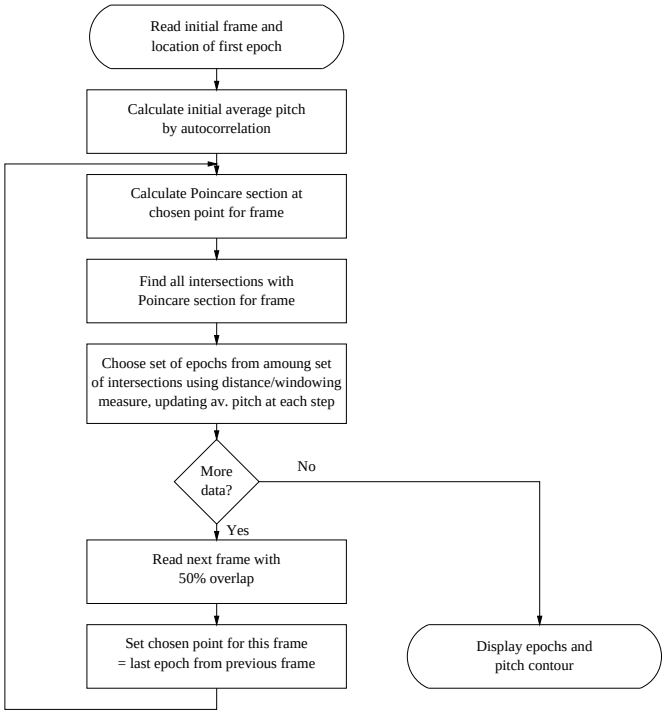


Figure 4.3: *Schematic of the epoch marking algorithm.*

The input waveform is assumed to be a voiced sound (a voiced/unvoiced detector would be required to pre–process the signal in order to remove unwanted unvoiced sounds in practice), and is blocked into overlapping frames with a 50% overlap between adjacent frames. A frame length of 35 msec is used, since voiced speech is often assumed to be approximately stationary for 30 to 45 msec.

4.4.1 Initialisation

The average pitch is used as an extra indicator of which points should be selected as GCIs, as will be explained subsequently. In order to achieve this, an accurate estimate of the average pitch is required. This is initially provided by the autocorrelation method [97]. A 30 msec window forward from x_{GCI} is firstly low–pass filtered and then centre–clipped to emphasise the periodic peaks. The autocorrelation function is then calculated, and the maximum peak within the permissible range of values (corresponding to a pitch value of between 50 Hz and 500 Hz) is picked to give the average pitch period $\overline{T_0}$.

4.4.2 Nonlinear processing

Each frame is embedded into 3D state space using an SVD window length of 50 samples, these parameters being chosen from the previous Lyapunov analysis. Using SVD embedding is an advantage in this case, since it removes any noise, making the reconstructed manifold smoother. This in turn means that the points on the Poincaré map will be easier to process. Conversely, the loss of any higher frequency information is not important for pitch detection. Assuming that the location of x_{GCI}, the very first epoch in the waveform, is known, a Poincaré section Σ is positioned normally to this point in state space and all of the P points, $\mathbf{x}_{\Sigma(j)}$ ($1 \leq j \leq P$), that traverse Σ are detected. In practice, these points can be found by looking for sign changes in $H(x)$:

$$H(x) = \langle \mathbf{h} \cdot (\mathbf{x}_i - \mathbf{x}_{\mathrm{GCI}}) \rangle \qquad 1 \leq i \leq N \tag{4.1}$$

where $\mathbf{h}$ is the approximated flow vector at $\mathbf{x}_{\mathrm{GCI}}$, $\mathbf{x}_k$ is the k–th reconstruction vector in the frame of length N and $\langle \mathbf{a} \cdot \mathbf{b} \rangle$ is the scalar product between $\mathbf{a}$ and $\mathbf{b}$. $\mathbf{x}_\Sigma$ will contain all of the epochs, since they are pitch synchronous with x_{GCI}, but other arbitrary points will also be present, dependent upon the shape and complexity of the phase space reconstruction.

Figure 4.4 shows a snapshot of the algorithm during processing on the constant pitch, stationary vowel /ɔ/ spoken by a female speaker. The epochs are marked as calculated on the time domain waveform. The left–hand–most epoch is that which was used as x_{GCI}. The phase space reconstruction shows how the Poincaré section, which intersects the flow at this point, crosses through several other parts of the manifold. This is seen more clearly on the two–sided Poincaré map. In this case the phase space structure is relatively simple resulting in four sets of

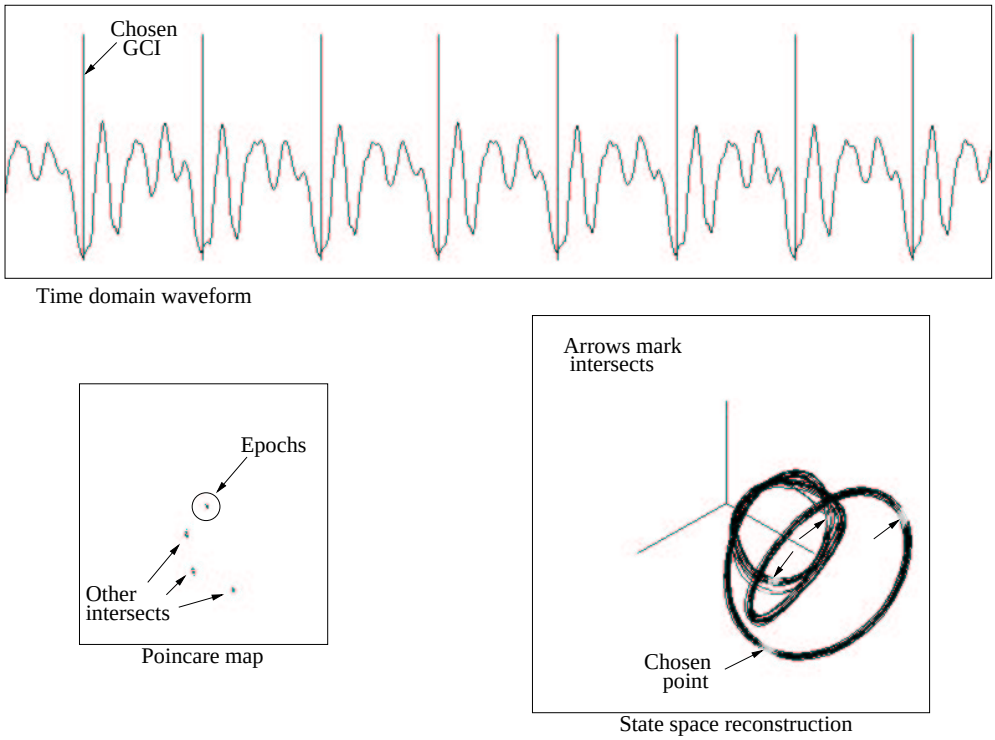


Figure 4.4: *Snapshot of processing of the stationary vowel /ɔ/, showing a frame of the signal with the calculated epoch markers (top); the two–sided Poincaré map of intersections with the phase space reconstruction (bottom left); the phase space reconstruction with intersects indicated (bottom right).*

intersection points.

4.4.3 Distance measure

The next step is to choose the correct points from $\mathbf{x}_{\Sigma}$. To do this, a distance measure, $\overline{d_m}$ is employed, which locates the points closest to $\mathbf{x}_{\mathrm{GCI}}$ in state space. This distance measure tracks an intersect point $\mathbf{x}_{\Sigma(j)}$ as it moves around one revolution of the phase space reconstruction, and finds the average distance in state space of that point from the corresponding movement of $\mathbf{x}_{\mathrm{GCI}}$:

$$\overline{d_m} = \frac{1}{R} \sum_{p=0}^{R-1} \mathrm{D}\left(\mathbf{x}_{(\mathrm{GCI}+pr)}, \mathbf{x}_{\Sigma(j+pr)}\right) \tag{4.2}$$

where $\mathrm{D}(\mathbf{a}, \mathbf{b})$ is the Euclidean distance between points $\mathbf{a}$ and $\mathbf{b}$ in 3D state space, and $r = (T_0 F_s / R)$ is $1/R$–th of a revolution around the phase space reconstruction (in samples) at the

local point, with local pitch period T_0 seconds at a sampling rate of F_s. R is typically 8 or 10 to give a good coverage of the phase space structure. In Figure 4.4, the four sets of intersect points can be easily separated by this technique, and the epochs correspond to the points enclosed within the circle on the Poincaré map, since they are within the same part of the reconstruction manifold as $\mathbf{x}_{\mathrm{GCI}}$. These are then marked onto the time domain waveform.

4.4.4 Windowing

In the above example, the incorrect points in $\mathbf{x}_\Sigma$ are well separated from the actual epoch points, and the state space distance measure can easily distinguish them. However, due to the complexity of the reconstruction for certain voiced sounds, other parts of the reconstructed manifold containing intersections with Σ will also pass close to $\mathbf{x}_{\mathrm{GCI}}$ in some cases. To prevent these points accidentally being chosen, a speech–specific windowing measure similar to that in [103] is used. Figure 4.5 shows how this operates. The average pitch period $\overline{T_0}$ is calculated at the initialisation stage using the autocorrelation method as described previously, and is then updated as each new epoch is marked. A search is then performed in a window between $0.6\overline{T_0}$ and $1.4\overline{T_0}$ onwards from the previously marked epoch for the closest point in $\mathbf{x}_\Sigma$ to $\mathbf{x}_{\mathrm{GCI}}$ in state space, since the pitch is not expected to vary by more than $\pm 40\%$ within a voiced section [97]. The new point found is marked as an epoch and the process continues through the frame. Once a frame has been processed, the last epoch marked is taken as x_{GCI} for the next frame, and so on through the data.

4.5 Experiments on voiced speech

4.5.1 Data set

Four different data sets were used to test the performance of the algorithm, giving varying degrees of realistic speech and hence difficulty.

1. Edinburgh constant pitch extended vowel sounds. This data set, consisting of 8 vowels spoken by four speakers, was extracted from the database collected for the Lyapunov exponent experiments [5]. It provides the simplest test, since the utterances are stationary, although in some cases the phase space structure is still relatively complex. Laryngo-

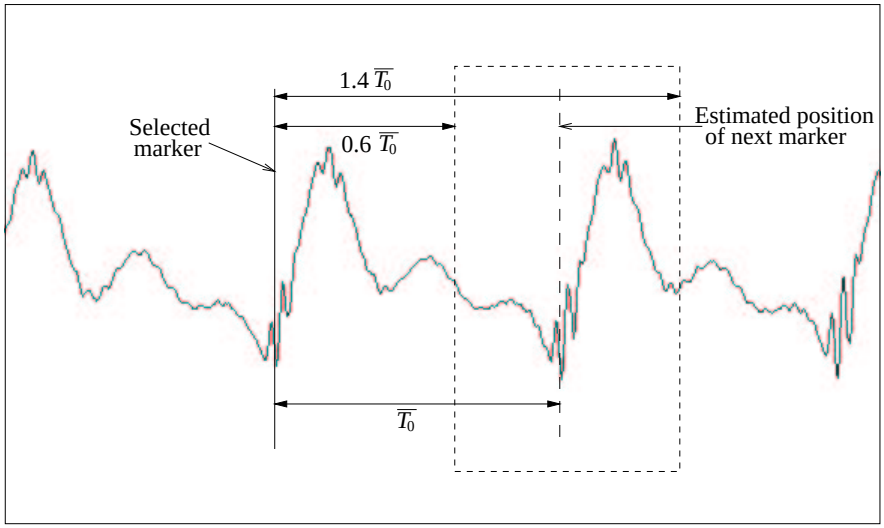


Figure 4.5: *Speech specific window to select epochs.*

graph data is also available for comparative purposes.[2]

2. BT Labs rising pitch vowel sounds.[3] A set of 9 vowels spoken by one male speaker. The actual vowel remains constant, but the pitch varies, increasing by approximately 100 Hz over the approximately 300 msec duration of the utterance. Laryngograph data was not available in this case.

3. Keele University pitch extraction database [104]. This database provides speech and laryngograph data from 15 speakers reading phonetically balanced sentences. From this several examples of continuous voiced speech were manually extracted.

4. BT Labs continuous speech. 2 phrases, spoken by 4 speakers, were processed manually to extract a data set of continuous voiced speech. Laryngograph data was also available.

In all cases the sampling rate used was 22.05 kHz so as to adequately fill the state space reconstruction. The Edinburgh database was recorded at this rate; the other signals were up–sampled, the BT data originally being at 12 kHz and the Keele signals at 20 kHz. All the signals have 16 bit resolution.

[2] A Laryngograph is a device attached to the throat that measures the impedance across the larynx, and hence gives precise GCI information.

[3] Thanks to A. Lowry of BT Labs for providing this data.

4.5.2 Results

4.5.2.1 Stationary vowels

The algorithm was found to correctly mark all of the stationary vowels considered. This is a relatively easy exercise, since the phase space structure does not change from frame to frame. The combination of the two measures (state space average Euclidean distance and time domain windowing) provides a robust performance.

At this point it is worthwhile discussing a problem that arises when using the Poincaré section technique with sampled data. Initially, even with the use of the distance and windowing measures, a loss of synchronisation occurred during the marking of the extended vowels, manifesting itself as a forward shift at each epoch. This was unexpected, since the vowel sounds were known to be stationary. The reason was found to be due to the calculation of the scalar product when finding the Poincaré section. Referring to Figure 4.6, the scalar product detects a sign change between $\mathbf{x}_a$ and $\mathbf{x}_{a+1}$, due to the crossing of Σ. Then, by default, the algorithm enters $\mathbf{x}_{a+1}$ into $\mathbf{x}_\Sigma$ and it would be chosen as an epoch. If, as in the figure, $\mathbf{x}_{a+1}$ is further away from Σ than $\mathbf{x}_a$ then a forward shift will occur, creating an accumulative error. The solution, as used in all results reported here, is to select the point closest to the Poincaré section.

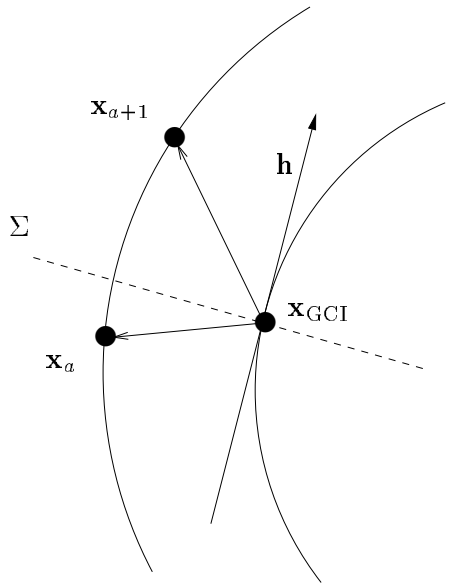


Figure 4.6: *Loss of synchronisation caused by a forward shift around the phase space reconstruction at each epoch.*

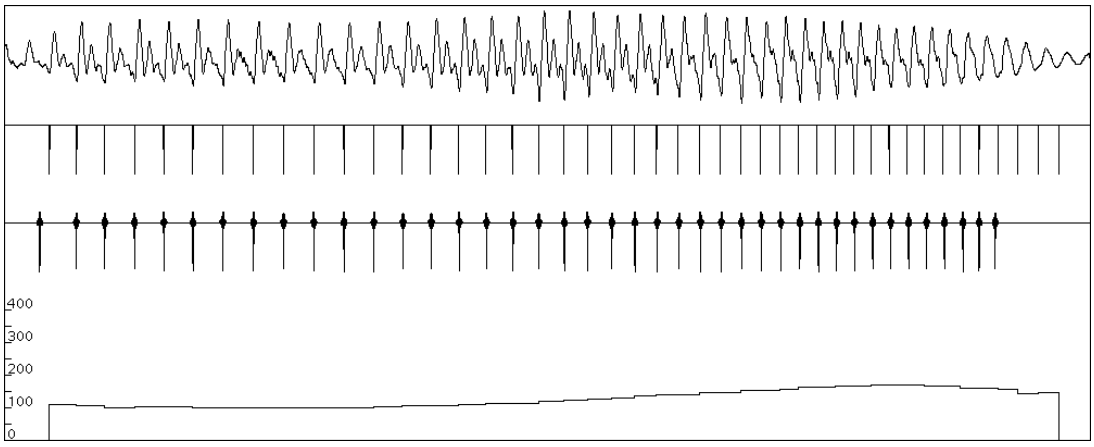


Figure 4.7: *Results for the rising pitch vowel /u/. From top to bottom: the signal; the epochs as calculated by the Poincaré algorithm; the epochs as calculated by the dynamic programming approach; the pitch contour (Hz) resulting from the Poincaré algorithm.*

4.5.2.2 Rising pitch vowels

As a next step, the simple non–stationary case of rising pitch vowels was considered. Figure 4.7 shows the results of the algorithm compared to those produced by a dynamic programming–based approach [100][4] for the vowel /u/, pronounced by a male speaker. The left–hand–most epoch was marked as x_{GCI}. It can be seen that the performance of the new algorithm is similar to that of this established approach, demonstrating that the frame–by–frame operation allows changes in the phase space structure to be tracked, caused in this case by the changes in pitch. The algorithm was found to perform correctly for all 9 vowels in the data set.

4.5.2.3 Continuous voiced speech

Finally the algorithm was tested with various voiced segments from the Keele University and BT Labs data sets. Figure 4.8 shows the algorithm's performance on a voiced section taken from the phrase "a traveller came along wrapped in a warm cloak", spoken by a female speaker. There is considerable change in the signal, and hence in the phase space structure, in this example, yet the epochs are still mostly well located when compared against the laryngograph signal. In Figure 4.9, which is a voiced section from the phrase "see if it's raining" spoken by a male speaker, the epochs are well located for the first part of the signal, but some slight loss of

[4] Available commercially as Entropic Research Laboratory's ESPS epoch function.

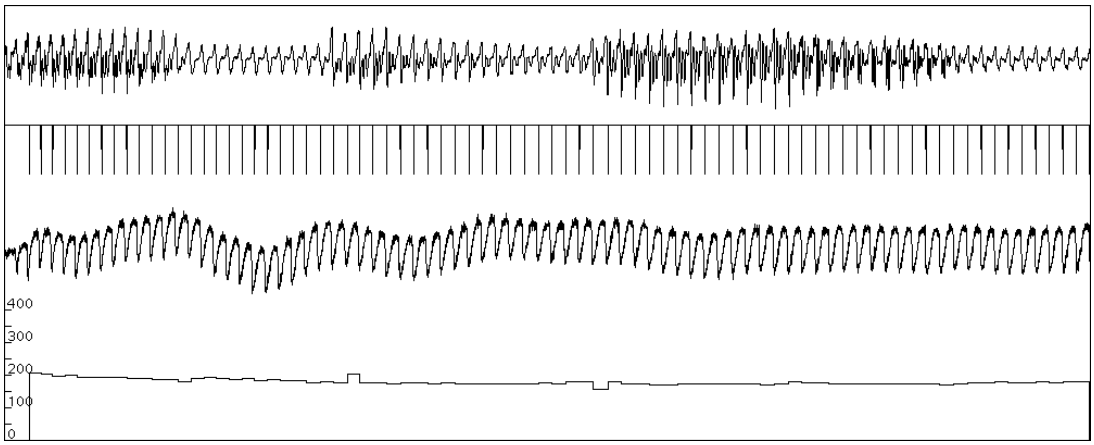


Figure 4.8: *Results for the voiced section of "came along" from the Keele database for a female speaker. From top to bottom: the signal; the epochs as calculated by the algorithm; the laryngograph signal; the pitch contour (Hz) resulting from the algorithm.*

synchronisation can be seen in the latter part.

Although the performance of the algorithm on continuous voiced speech is generally good, more severe problems have been encountered with some of the other signals tested, resulting in serious glitches or even a complete loss of synchronisation. This is usually caused by the state space reconstruction being very complicated, thus making the selection of the correct intersect points difficult. In general, the more complicated state space structures occur at low pitch values where there are many oscillations within one pitch period, and also with voiced fricatives which tend to fill the state space.

4.5.3 Discussion

It is clear that a major drawback to this algorithm is the need to know the location of the first epoch. Due to the nature of the technique, it is only possible to mark points which are pitch synchronous; it cannot be deduced *a priori* if a data point is a glottal closure instant, but once one is known all subsequent epochs in that voiced section can, in theory, be located.

This aside, the technique appears useful. It works very well on the simple cases of stationary vowels and rising pitch vowels, accurately marking all the epochs. Application to real speech signals has met with moderate success. The algorithm is often able to track consid-

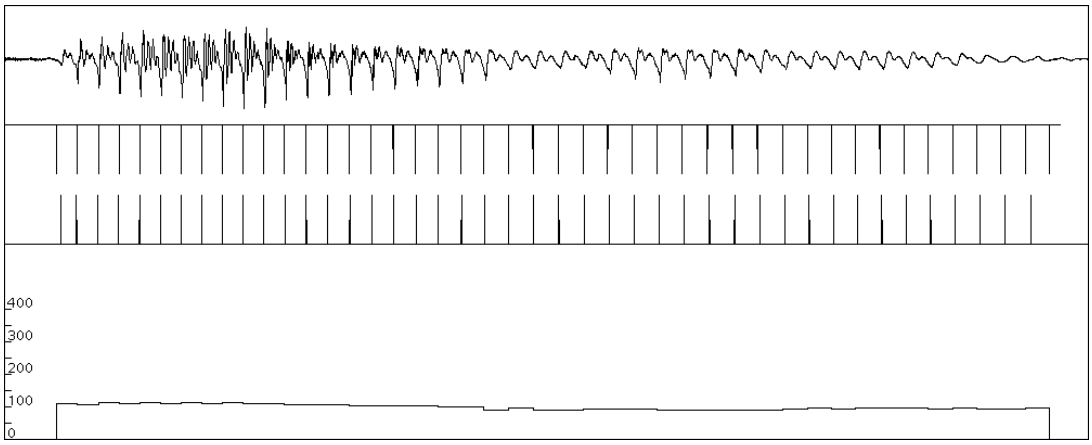


Figure 4.9: *Results for the voiced section of "raining" from the BT Labs database for a male speaker. From top to bottom: the signal; the epochs as calculated by the algorithm; the processed laryngograph signal; the pitch contour (Hz) resulting from the algorithm.*

erable changes in the phase space structure, caused by changes in vowel sound and/or pitch, however should an error occur it is unable to recover. Complicated phase space structures, such as the voiced fricative shown in Figure 4.10, are the most common cause of failure, due to the difficulty in correctly aligning the Poincaré section. Once an error has occurred, all of the points forward from the error point will be incorrectly marked. This is again due to the fact that the epoch pulses are not specifically being located, only points which are pitch synchronous. Therefore when an error occurs, causing a loss of synchronisation, it propagates through the remainder of the signal.

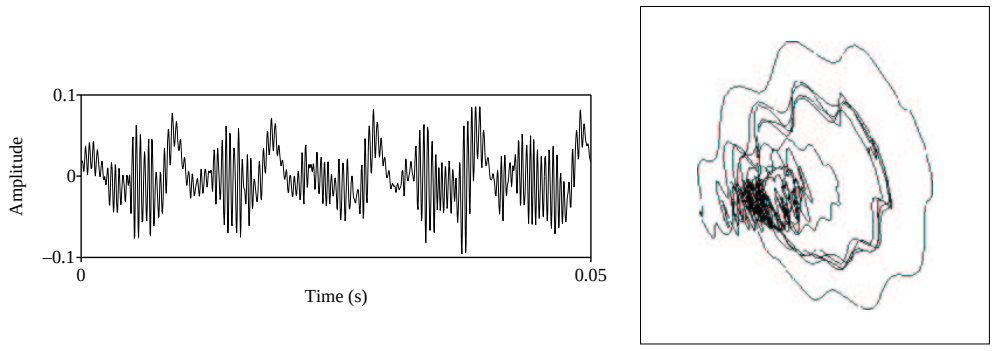


Figure 4.10: *An example of the voiced fricative /ʃ/, showing the time domain (left) and complex state space structure (right). SVD embedding is used in the state space reconstruction.*

4.6 Conclusion

In this chapter the potential for using nonlinear techniques, in the form of a Poincaré map, to solve the problem of epoch marking in voiced speech has been demonstrated. Again it should be emphasised that the algorithm presented here is not to be seen as a direct competitor to the existing methods that have been described. It does however show that nonlinear techniques can be applied in other areas beyond analysis, and also provides a useful insight into working in the state space domain.

Further work is required to address the problem of identifying the first epoch point. As such, the algorithm is currently only capable of marking pitch–synchronous points. Additionally, improvements are needed so that the algorithm is better able to cope with sudden changes in phase space structure, so as to avoid loss of synchronisation. Taking these two short–comings into account, the algorithm does perform well on simple vowels and gives promising results when applied to real continuous voiced sounds.

Chapter 5
Synthesis by locally linear model

5.1 Introduction

Now the main objective of this work can be examined: the use of nonlinear dynamical theory for the synthesis of voiced speech. There are several key objectives. Firstly the technique used must generate natural–sounding speech. This means capturing the full dynamics of the speech signal, including the natural variability. However, this alone is not sufficient. It must also be possible to impose prosody on the synthesised signal, which mainly concerns the ability to change the duration and pitch of the sound.

Following from the previous work on estimating the Lyapunov exponents of vowel sounds, the use of the local dynamics is an obvious choice for synthesising speech. A method using this locally linear model that has been implemented is described, and then its performance according to the above aims is discussed. Difficulties with pitch modification are shown to be the major short–coming of this technique, and a possible method for performing pitch modification in state space is presented, motivated by an examination of the variations of the state space reconstructions for naturally varying pitch vowel sounds. This is shown to work in principle, but fails when used in conjunction with the locally linear model.

5.2 Locally linear synthesis model

The locally linear model of attractor dynamics in state space has been shown previously to be useful for the estimation of Lyapunov exponents for vowel sounds. This involved the formation of small local neighbourhoods which are then allowed to evolve around the phase space reconstruction. The function that linearly maps between evolved neighbourhoods is called a tangent map, and this has also been applied to the synthesis of extended vowel sounds. In principle, any voiced sound that can be reconstructed on a low–dimensional attractor can be synthesised by this technique. The synthesis of unvoiced sounds will not be discussed because, as previously

mentioned, there is no perceptual difference between real unvoiced sounds and those generated by a linear prediction model [33].

5.2.1 Local dynamics for synthesis

Given the previous results that voiced speech is a nonlinear, low–dimensional system with low sensitivity to initial conditions, Banbrook [5] implemented a nonlinear synthesiser based on the local dynamics of the state space reconstruction. Following the same principle as the Lyapunov exponent analysis, a local neighbourhood is formed about the point on the phase space reconstruction nearest to the synthesis start point. The synthesis start point is chosen to be within the reconstruction manifold, but not on the existing data. Also, time delay rather than SVD embedding is used. As shown in Figure 5.1, the neighbourhood matrix is evolved one time step, and this is used to estimate the tangent map, $\mathbf{T}_i$. This dynamics estimate is applied to the synthesised start point to produce the next synthesised point and the process is then repeated. Using this method, any duration of synthesised speech can be generated. As long as the chosen

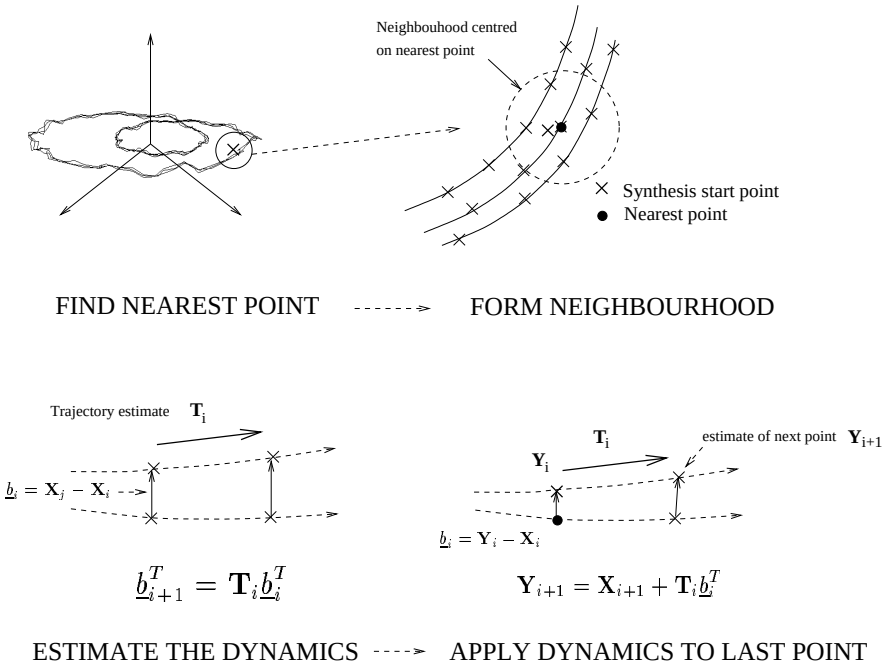


Figure 5.1: *Locally linear synthesis (after [6]).*

synthesised start point is not on the existing stored dataset, then the synthesised trajectory will be unique and will be dependent only on the initial starting point and the local dynamics around the attractor. The result of this is to produce a speech waveform that has no repetition in it, and

contains much of the natural variability of real speech. This is particularly noticeable when synthesising extended vowel sounds. Currently, using up–to–date concatenation techniques, extended vowels sound buzzy and unnatural. This is because the waveform has to be repeated in order to produce the longer sound. Techniques such as inverting the phase of the signal are employed in an attempt to reduce this effect, but this does not entirely remove it. In contrast, the nonlinear synthesiser does not perform a copying operation and so does not suffer from this problem [6].

5.2.2 Analysis of synthesised vowel sounds

To quantify the performance of the nonlinear synthesiser, several measures can be made. A natural–sounding vowel must contain some variability, since when this is not included the generated speech sounds extremely robotic. Additionally, the whole spectrum must be well reproduced. If the higher frequencies are not adequately modelled, then this will also reduce the naturalness of the synthesised sound. This latter effect is particularly evident when considering wideband (7 kHz bandwidth) and above, which is necessary for high quality synthesis.

Using the vowels /i/, /ɑ/ and /u/ from the Edinburgh extended vowel database as templates, with examples from four speakers (two male, two female), a set of synthesised speech waveforms was generated. The parameters used were 20 vectors in the neighbourhood matrix, 2000 data points from the original data in the phase space reconstruction, synthesising 500 msec of speech. The synthesis start point was calculated as a 1% shift away from the first original data vector. This ensures that the synthesised trajectory is not on the original data trajectory, but still remains within the manifold.

Jitter

Jitter is the variation in length of individual pitch periods. For normal, healthy speech it should be between 0.1 and 1% of the average fundamental frequency for sustained vowels [105]. Obviously the higher the sampling rate used, the greater the accuracy of the jitter measurement. In the experiments reported here, the original data sampling rate of 22.05 kHz is used. More complex algorithms, involving resampling to higher rates and then evaluating the quality of the interpolation, do exist (*e.g.* [106]), but a simpler measure is sufficient to give a general indication of the jitter present in the original and synthesised waveforms. This involves finding

Data type	MC (male)	PB (male)	CA (female)	KH (female)	Average
Synthesis jitter (%)	0.451	0.520	1.14	0.727	0.710
Original jitter (%)	0.690	0.686	0.685	0.906	0.742
Synthesis shimmer (%)	4.94	3.22	2.44	3.40	3.50
Original shimmer (%)	4.21	4.60	7.06	4.81	5.17

Table 5.1: *Percentage jitter and shimmer in original and synthesised waveforms, averaged over the vowels /i/, /ɑ/ and /u/ for each speaker, and as an average over the database.*

the epoch locations[1] over a 500 msec window of speech data, then calculating the average pitch period and the individual deviations from this.

The average variation, expressed as a percentage of the average pitch period length, is shown in Table 5.1 for the original and synthesised data for each speaker. This shows that the amount of jitter in the synthesised waveforms is comparable to that present in the originals. However, examining the actual cycle–to–cycle variations as a time series, as shown in Figure 5.2, reveals a trend not seen in the percentage statistics. The period length variations of the original do not follow any obvious pattern, but the variations in the synthesised data period lengths have a definite periodic structure. In this example, for the vowel /ɑ/, speaker KH, the period length variation repeats itself every 17 pitch cycles. A similar periodic structure was found in all of the other synthesised vowels, with repetition rates of between 2 and 22 cycles. Further study would be necessary to determine the cause of this repetition in the synthesised waveform. However it does seem certain, at least for the shorter repetition rates, that this will detract from the naturalness of the generated sound.

Shimmer

Shimmer is defined as the variations in energy of each pitch cycle. It is calculated by finding the variations during each cycle from the average power, over the analysis window (using the epoch markers found during the jitter analysis). These are then expressed as a percentage of the average power. Using the same 500 msec segments as in the jitter measurements, the shimmer for the three vowels over the four speakers was found. The results, averaged by speaker, are also shown in Table 5.1. As seen for the case of the jitter, the average values of the synthesised data compare well with the original. Figure 5.3 shows the power per cycle variations for the vowel /i/, speaker MC, as a time series. Again it is evident that the original data variations

[1] Using Entropic Research Laboratory's ESPS epoch function.

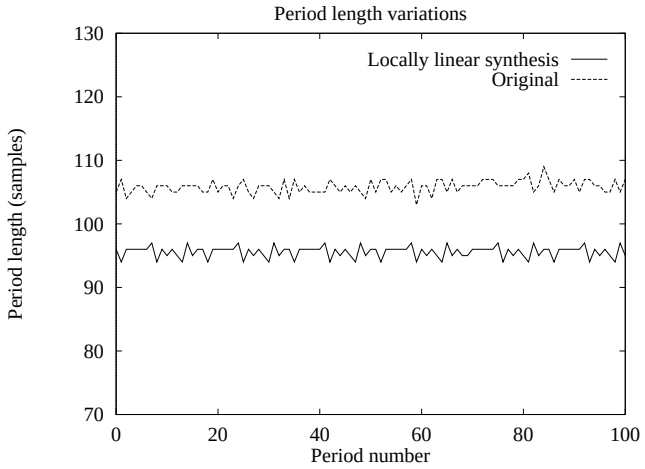


Figure 5.2: *Comparison of period length variations between the original data and the locally linear synthesiser results, for the vowel /ɑ/ pronounced by a female speaker. The original data contour has been shifted upwards by* 10 *samples to allow for easier comparison.*

have no obvious structure, whereas the synthesised data variations are definitely periodic. The repetition rate is the same as found in the jitter analysis, and is again present in all vowels examined.

Spectral characteristics

Figure 5.4 shows wide–band spectrograms of original and synthesised speech for the vowel /u/, speaker PB. It can be seen that the spectral characteristics of the original have been well captured by the synthesiser, including the higher frequencies. Correctly modelling all frequencies will make a significant impact on the naturalness of the synthesised speech, in a similar way to the improvement found in speech coding when moving from telephone band (4 kHz) to wide-band (7 kHz). In this respect, the locally linear synthesiser gives excellent results. The reason it is able to do this is due to the local technique, which is able to follow very fine variations in the attractor dynamics. Further spectrogram examples of different vowels and speakers are presented in Appendix B, all showing good spectral modelling.

The jitter, shimmer and spectrogram results as a whole show that this nonlinear technique does appear to synthesise vowel sounds in a reasonably natural manner, with the frequencies well modelled across the whole range. Jitter and shimmer are still present in the synthesised signal, but some naturalness will be lost because of the periodic structure imposed upon them.

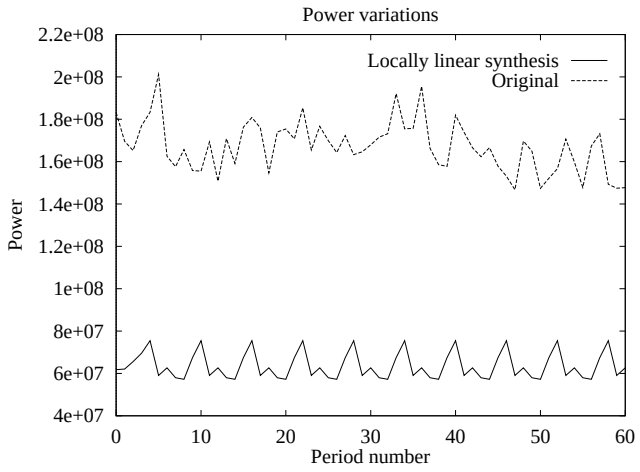


Figure 5.3: *Comparison of raw power per pitch cycle variations between the original data and the locally linear synthesiser results, for the vowel /i/ pronounced by a male speaker.*

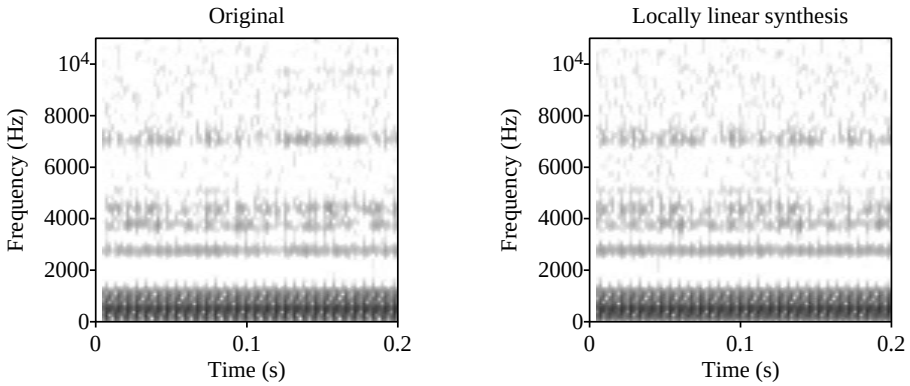


Figure 5.4: *Wide–band spectrograms of original (left) and synthesised (right) speech, for the vowel /u/, speaker PB.*

Obviously this can not be a complete substitute for the subjective test of actually listening to the generated sound. However, in limited informal tests, listeners found that the synthesised waveforms were reasonably natural–sounding.[2]

5.2.3 Pitch and phoneme modification

Given that a synthesiser can produce natural–sounding vowels, it is also necessary to be able to alter the pitch, duration and volume so that prosody can be imposed onto the synthesised phrase. Duration modification is inherently built into the nonlinear synthesiser architecture, and any required waveform length can be synthesised. Volume is comparatively easy to impose, and can be done by a post–processing stage. Pitch modification is more difficult, since it required altering the fundamental frequency of the waveform without affecting the other spectral characteristics.

The method proposed by Banbrook for pitch modification is tied to the technique for moving from one vowel phoneme to another [6]. As an example of this phoneme modification, consider moving from /i/ to /ɪ/, as shown in the formant chart of phase space reconstructions in Figure 5.5.

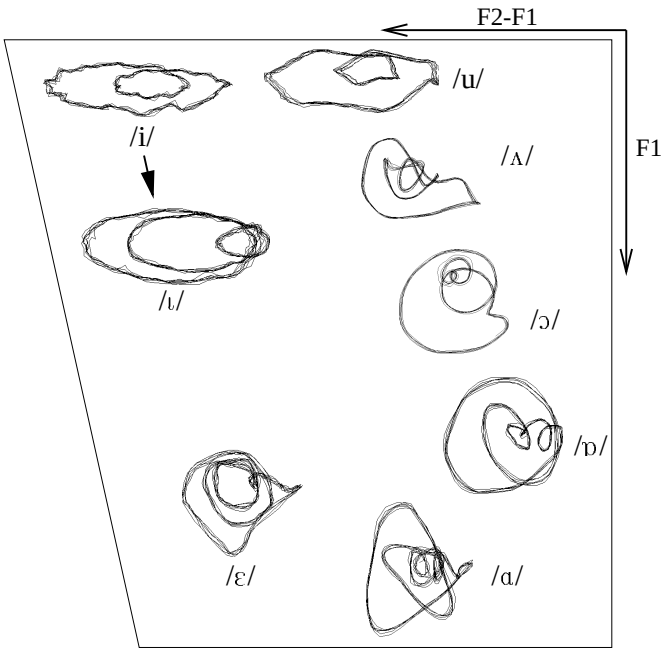


Figure 5.5: *Formant chart of phase space reconstructions (after [5]).*

[2] Example audio files are provided on CD, as described in Appendix D.

Jumping straight from one phoneme to another causes a discontinuity and audible artifacts, but this can be eliminated by defining a series of intermediate steps from

$$\mathbf{t}_i = (\mathbf{a}_i - \mathbf{e}_i)\rho + \mathbf{e}_i \tag{5.1}$$

where $\mathbf{t}$ is the set of points on the intermediate reconstruction that is a fraction ρ of the Euclidean distance between corresponding points on the main phase space reconstructions $\mathbf{a}$ and $\mathbf{e}$. ρ is incremented gradually in order to avoid discontinuities. Moving more than one position in the formant chart is achieved by stepping via the other reconstructions, using the same principle as outlined above.

In order to locate corresponding points on the phase space reconstructions so that this linear interpolation can be performed, the period cycle of each reconstruction, and hence the pitch, must be held constant. This is achieved by normalising each pitch period to exactly the same number of points, using waveform interpolation and resampling.

Pitch modification is approached in exactly the same manner. The original waveform is resampled to produce phase space reconstructions that contain the required number of samples per pitch period. Intermediate reconstructions can then be generated between these. Obviously it is not possible to locate exact "corresponding points" between the reconstructions, as the number of points per cycle changes gradually with each one. However, as long as the step size between the phase space reconstructions is never too great, there should not be any discontinuities.

5.3 Pitch variations

The method proposed by Banbrook, as described above, is clearly somewhat *ad–hoc*. Forcing the waveforms to have exactly equal pitch periods removes natural pitch jitter, although this requirement should only be necessary at transitions and can be relaxed when generating constant sounds. Putting this aside, the actual method used to vary the pitch is also unrealistic. Resampling the waveform to include more or less points into each pitch period is not the same as changing the fundamental frequency, because it does not actually change the shape of the waveform. Banbrook does note that it would be possible to use other existing pitch modification techniques (*e.g.* PSOLA) to operate on the original time domain signal to produce a set of time domain signals at different pitch values, and then generate a series of phase space reconstructions from these. However this appears excessively complex, still requires the use of

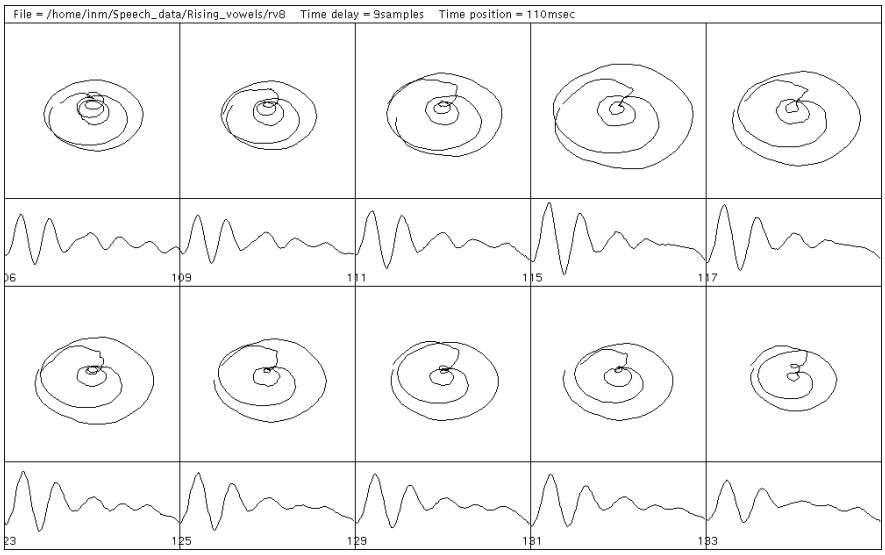


Figure 5.6: *Variations of the phase space reconstructions over changing pitch for the rising–pitch vowel /ɒ/. Each frame holds one pitch period, with at the top the phase space reconstruction and at the bottom the time domain signal. The local pitch for each frame is marked in Hz.*

linear interpolation between the different reconstructions, and would combine the problems of the two techniques.

5.3.1 Natural pitch variations

Figure 5.6 shows the vowel /ɒ/ as a series of snapshots. Each snapshot contains the time domain waveform and time delay embedding reconstruction for one pitch period, with the fundamental frequency rising from 106 Hz to 133 Hz. The time axis is variable, so that each frame contains exactly one period, defined by the position of the GCIs. The time delay τ was calculated from the mutual information[3] over the whole waveform, and was set at 9 samples. This vowel is one of the nine in the BT Labs rising pitch database, as described in Section 4.5.1.

Examination of this clearly shows that the time domain waveform changes shape with pitch. It is also evident that the changes in the shape of the phase space reconstructions do not follow a linear pattern, even over the small range of fundamental frequency shown here. Instead changes such as the the number and size of inner spirals can be observed. Therefore the use of linear interpolation between reconstructions at different pitches is unlikely to be realistic. If pitch

[3]Thanks to C. Unsworth for providing the mutual information software.

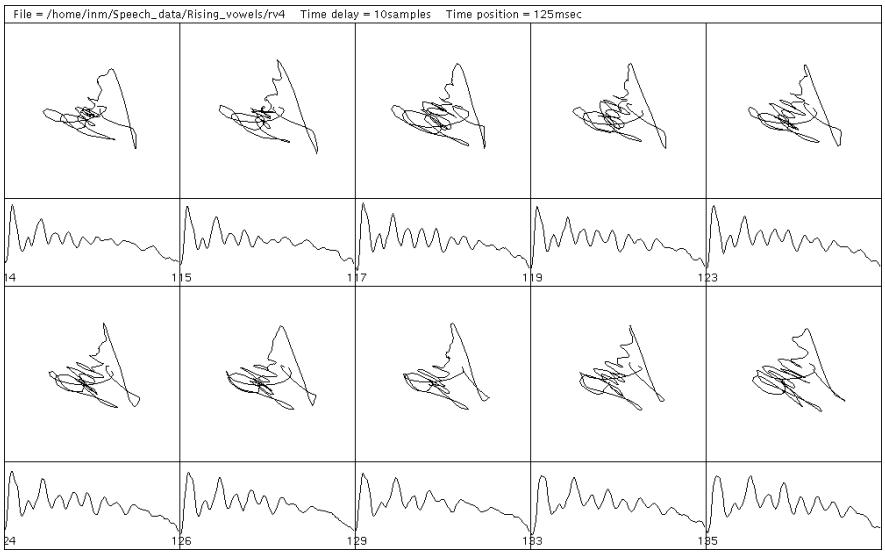


Figure 5.7: *Variations of the phase space reconstructions over changing pitch for the rising–pitch vowel /æ/, as in Figure 5.6. τ is set at 10 samples, as calculated by the mutual information over the whole waveform.*

modification is to be performed in the state space domain, which is desirable given the nature of the synthesis technique, then an alternative method will be required to give realistic results.

In the above example, the vowel phase space structure is relatively simple, although this is not the case for all other vowels. Figure 5.7 shows an example of the vowel /æ/, with the pitch varying between 114 Hz and 135 Hz. In this series of snapshots, the phase space structures are more complicated and the variations between the structures are considerably less evident. This is true of many of the vowels examined from the rising pitch database, and presents us with a problem when searching for a solution to state space–based pitch modification. Essentially, any algorithm which is to be able to modify the pitch in a realistic manner must be capable of reproducing the phase space variations observed in real pitch–varying speech.

5.3.2 Pitch variations in LP synthesised vowels

In an effort to simplify the problem, vowels generated by a linear prediction synthesiser were examined. The purpose of this exercise, which appears counter–intuitive when dealing with nonlinear synthesis, is to start with a simpler waveform, which can also be generated at any required fundamental frequency. If an analysis of this simple waveform leads to the development of a suitable algorithm for pitch modification, then this is a step towards implementing an

algorithm for real speech signals.

Figure 5.8 shows the time domain waveforms and corresponding 2D projections of the 3D phase space structures for the linear prediction synthesised vowel /u/. The LP coefficients were calculated using the constant pitch /u/ vowel sound of the male speaker PB, taken from the Edinburgh 1996 database. The LP filter was then excited by a Dirac pulse train of appropriate period. This generated the stationary synthesised vowels shown, with fundamental frequencies of 70 Hz, 100 Hz, 130 Hz and 160 Hz. These values of pitch would be typical for a male speaker.

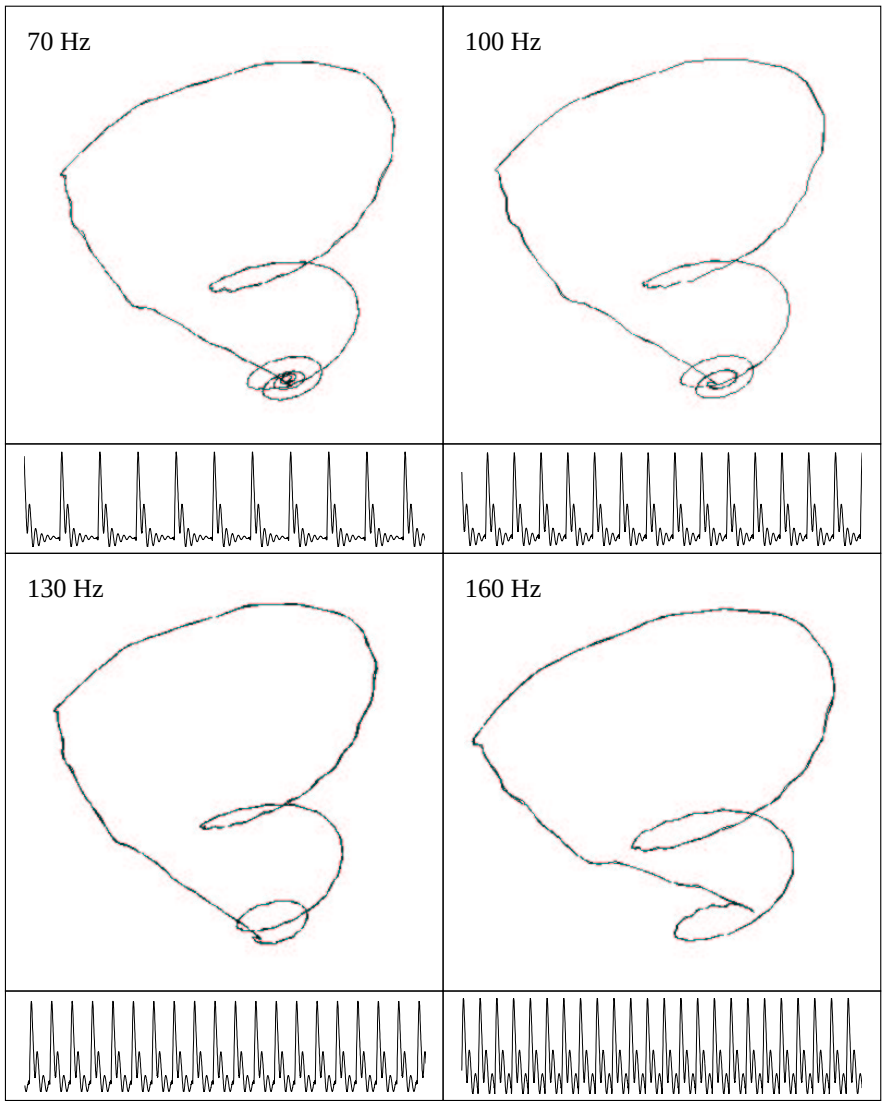


Figure 5.8: *LP synthesised vowel /u/ at pitch values of* 70 *Hz*, 100 *Hz*, 130 *Hz and* 200 *Hz.*

All of these phase space reconstructions are characterised by a trajectory rising steeply out from a centre point, which then returns towards this point in a series of downwards spirals. This equates to the excitation by the glottal pulse, followed by a slow decay of the waveform until the next impulse. At lower fundamental frequencies, corresponding to longer pitch periods, the number of spirals is greater, since there is a greater delay between epoch pulses. It is possible to divide each of the phase space reconstructions into two parts. The outer part consists of the outward pulse and the wide initial inward spiral, and is almost constant across all four structures. The inner part consists of the tight inner spirals close to the centre point, and it is here that the variation between structures due to the pitch change can be observed. The number of inner spirals appears to be due solely to the length of the pitch period. This topological description, although applied to LP synthesised vowels, also has some similarities with the real speech signals shown previously. By a careful examination of both Figure 5.6 and Figure 5.7, a form of outward trajectory followed by inwards spiralling can be seen.

Examining the LP phase space reconstructions from a nonlinear dynamical theory viewpoint, it would appear that there is a fixed point around which the trajectories evolve. Fixed points may take a number of forms. In Chapter 3, point and limit cycle attractors were presented. A more complex form is a saddle point, which has trajectories approaching the fixed point close to the in–set (stable manifold) and diverging away near the out–set (unstable manifold). Index 1 saddle points have an in–set that is a surface, and an out–set that is a curve, whereas the opposite is true of index 2 saddle points. Spiral saddle points have trajectories spiralling around the fixed point near the surface, towards or away from the saddle, for index 1 and index 2 respectively. Figure 5.9 shows an example of an index 1 spiral saddle point. Looking at the inner part of the LP phase space reconstruction, particularly at low pitches, an index 1 spiral saddle point appears to be a good description.

A homoclinic trajectory occurs when a trajectory on the unstable manifold joins another on the stable manifold, and thus forms a single trajectory connecting the saddle to itself [101]. For spiral saddle points, this type of trajectory is also called a Šilnikov orbit, after Šilnikov's Theorem [107], as shown in Figure 5.10.

Trajectories that come near the saddle point will approach the saddle close to the stable manifold and then be forced away from it near the unstable manifold. However, as they are pushed away by the out–set, they will be re–captured by the in–set and pulled back in towards the saddle point. This description captures very closely the behaviour seen in all parts of the LP vowel

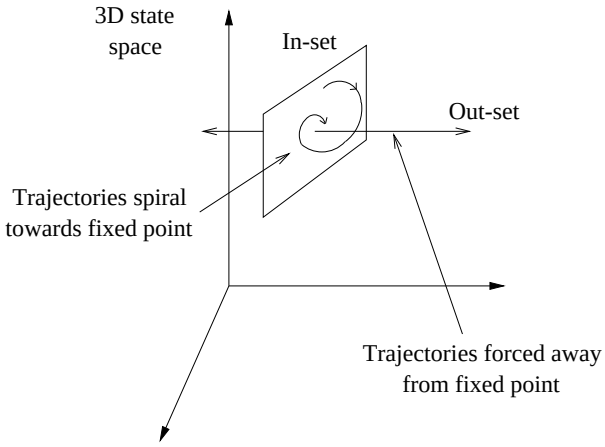


Figure 5.9: *A spiral saddle point (index 1), with trajectories spiralling towards the fixed point near a surface and diverging away along a curve.*

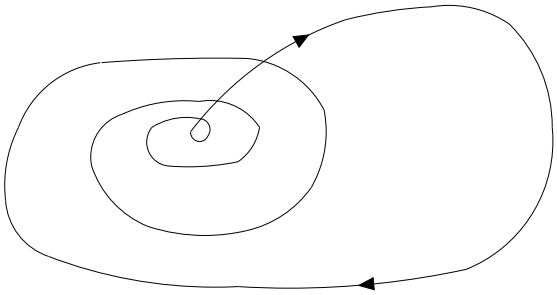


Figure 5.10: *A Šilnikov-type orbit for an index 1 spiral saddle point.*

state space reconstructions. The similarity between vowel phase space reconstructions and Šilnikov orbits has also been noted by Tokuda *et al.* [89], in their nonlinear analysis of the Japanese vowel /a/.

5.4 Attempted application of controlling chaos ideas

This analysis inspires a possible alternative approach to pitch modification, which operates entirely in the state space domain. It is based on concepts from the controlling chaos literature, and involves perturbing the trajectory of the speech signal in state space in order to affect a change in its orbit.

Examining once again the phase space reconstructions for different pitches of the vowel sound as produced by a linear prediction synthesiser (Figure 5.8), it appears that almost all of the information about the higher pitch sounds is contained in the lowest pitch vowel reconstruction. The effect of decreasing pitch, *i.e.* increasing pitch period, is an increase in the number of spirals in towards the centre of the reconstruction, while the remainder of the phase space structure is approximately constant. Therefore it should be possible to modify the lowest pitch phase space reconstruction in some way, so that a higher pitch version can be produced. This may not be entirely realistic for real vowel sounds, but an algorithm capable of pitch modification of LP synthesised sounds would provide a stepping–stone to pitch modification of real voiced speech.

5.4.1 Controlling chaos

In the field of nonlinear dynamics, there has been a large amount of interest in the possibility of controlling systems which exhibit chaotic behaviour, so as to improve their performance. The basic principle is to locate low–period unstable periodic orbits within the attractor, which mainly comprises a large number of uncontrolled chaotic orbits. Then, using small perturbations of some control parameter, the system is moved and stabilised onto one of the low–period orbits, which is chosen so that performance is optimised. This was first proposed by Ott *et al.* [108], and then further refined, allowing the technique to be used with time delay embedding, by Dressler and Nitsche [109].

5.4.2 Principle of pitch modification by small changes

Some of the concepts of this technique can be applied to the low–pitch vowel phase space reconstruction in order to modify the fundamental frequency. Assuming that the phase space reconstruction does have a Šilnikov–type orbit, with an index 1 saddle point at the centre, then the trajectory spirals in towards the fixed point near the stable manifold (which is a plane), before being ejected out close to the unstable manifold. The process then repeats with the re–injection back towards the fixed point. The idea is to perturb the trajectory when it is close to the saddle point. Moving the trajectory closer to the plane of the stable manifold will cause it to spend longer in the region of the fixed point, thus increasing the pitch period and decreasing the pitch. Conversely, moving the trajectory away from the stable manifold, in the direction of the unstable manifold, will cause a faster ejection and therefore higher pitch.

An algorithm capable of perturbing the speech trajectories as described would need to perform the following operations:

- Embed the time series in 3D state space.
- Locate the fixed point. The centre of the phase space reconstruction will be close to the index 1 saddle point, where the two–dimensional stable manifold intersects with the one–dimensional unstable manifold. In practice, it will not be possible to locate the saddle point exactly, only the closest data point to it.
- Find the direction of the stable and unstable manifolds. The stable manifold is expected to be a plane and the unstable manifold a curve.
- Perturb the trajectory. Figure 5.11 shows the trajectory approaching the saddle point and entering a sphere of radius ϵ. At the point of entry, it is a distance δ away from the stable manifold in the direction of the unstable manifold, whose magnitude is Λ. By perturbing the trajectory towards the stable manifold (*i.e.* decreasing δ), the trajectory will spend longer near the fixed point, whereas moving the trajectory away from the manifold (an increase in δ) will cause a faster ejection.
- Calculate the relationship between the size of the perturbation and the change in pitch, so that arbitrary pitch modifications can be made.

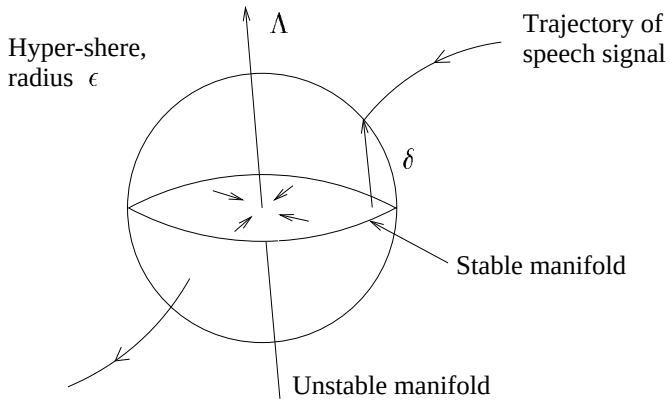


Figure 5.11: *Principle of perturbing trajectory to modify pitch.*

5.4.3 Period modification in a Šilnikov flow

Before attempting to apply the above algorithm to the LP speech signal, modifying the period of a system which is completely specified, via a set of equations, will be examined. Consider the three coupled differential equations:

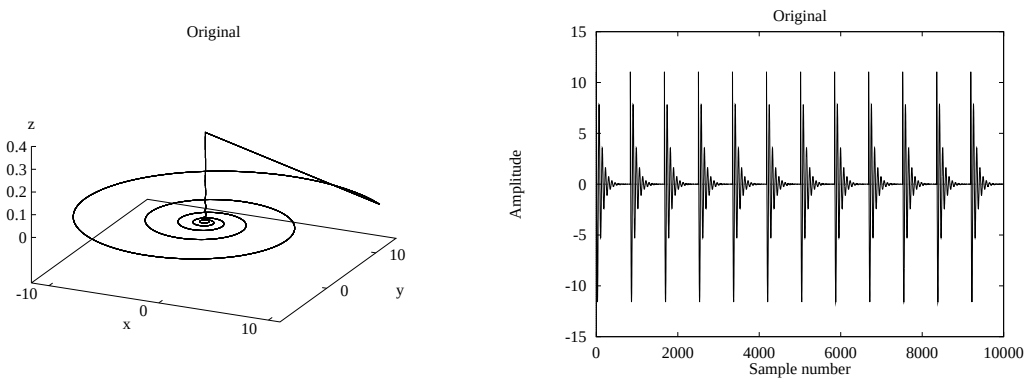
$$\begin{aligned} \dot{x} &= \alpha x - \beta y \\ \dot{y} &= \beta x + \alpha y \\ \dot{z} &= \gamma z \end{aligned} \tag{5.2}$$

These define a Šilnikov flow in the region around the fixed point, although they do not model the re–injection that is characteristic of a homoclinic orbit. The system can be seen to have a fixed point at $(0, 0, 0)$, since the time derivatives go to zero at this point. Performing an eigenanalysis, the eigenvalues are found to be at

$$\begin{aligned} \lambda_1 &= \gamma \\ \lambda_{2,3} &= \alpha \pm j\beta \end{aligned}$$

The eigenvalue $\lambda_1 = \gamma$ has a corresponding eigenvector of $(0, 0, 1)$, and the eigenvectors of the complex conjugate eigenvalues are both in the x–y plane. If α is negative and γ is positive, then this defines an index 1 spiral saddle. Trajectories will spiral in towards the fixed point near the x–y stable manifold and then will be ejected out near the z unstable manifold.

Choosing the values $\alpha = -0.1$, $\beta = 1.0$ and $\gamma = 0.08$, the equations were iterated using the fourth order Runge–Kutta method [110], with a step size of 0.1. In order to simulate a



(a) **(b)**

Figure 5.12: *Šilnikov orbit with re–injection from Equation 5.2:* (a) *3D state space and* (b) x *against time.*

homoclinic orbit, the trajectory was reset back to its start point after it had been ejected out along the unstable manifold a considerable distance from the fixed point (the threshold was set at $z > 0.4$). The resulting state space plot is shown in Figure 5.12(a). Plotting the variable x against time, as seen in Figure 5.12(b), results in a periodic waveform. The re–injection to complete the orbit is clearly seen. Evidently this is not very realistic, but this is not relevant as it is only the inward spiral followed by ejection that is of interest. In these terms, Equation 5.2 with re–injection generates a realistic homoclinic orbit.

Now the principle of perturbing the trajectory is applied. To reiterate, moving the trajectory towards the stable manifold should cause an increase in the period, whereas moving away from the stable manifold (in the direction of the unstable eigenvector) will decrease the period. When the trajectory enters a sphere of radius ϵ during the Runge–Kutta iteration, it is then modified. In the experiments presented here, ϵ was set at 0.1, which is approximately 1% of the spiral radius in the x–y plane. In order to find the relationship between the period length and the modification factor, the distance from the stable manifold in the direction of the unstable manifold, δ, and the corresponding period length, n, were recorded over a range of values. Plotting n against $\log \delta$ results in a straight line, as seen in Figure 5.13. Therefore the period length n can be expressed as:

$$n = a \ln \delta + b \tag{5.3}$$

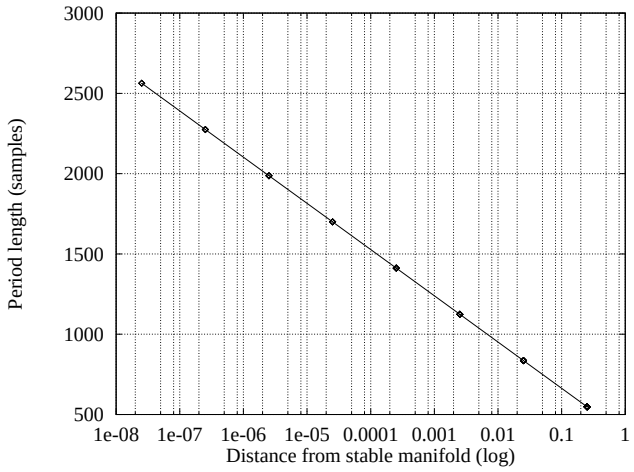


Figure 5.13: *Log relationship between period length and distance from stable manifold.*

where a and b are constants, which are easily calculated from simultaneous equations. Denoting the distance from the stable manifold as the trajectory enters the sphere before modification as δ_0, then the multiplier, R, required to change the period length to n_d samples is:

$$R = \frac{e^{\frac{n_d - b}{a}}}{\delta_0} \tag{5.4}$$

Upon entering the sphere, the position vector (x, y, z) is modified to (x, y, Rz), causing a move towards or away from the stable manifold in the direction of the unstable eigenvector.

The original orbit has a period length of 836 samples. In the following examples, the period is modified by 0.6 ($n_d = 502$ samples) and 1.4 ($n_d = 1170$ samples). a and b are calculated as -124.99 and 375.92 respectively. The values of R are then found as 14.5 and 0.069. The resulting modified waveforms are shown in Figure 5.14. Figure 5.15 shows a magnified view of the state space plot around the fixed point demonstrating the modification taking place, for the case of period modification by a factor of 1.4.

This demonstrates the validity of the trajectory perturbation approach. A Šilnikov–type orbit, which has a periodic time domain structure quite reminiscent of a vowel, has been modified, allowing both extension and shortening of the period length. In theory, it should be possible to extend the period length by any required factor by moving close to the stable manifold. The limit on period length shortening, on the other hand, is governed by the size of the sphere, since no modification occurs until the trajectory has entered it.

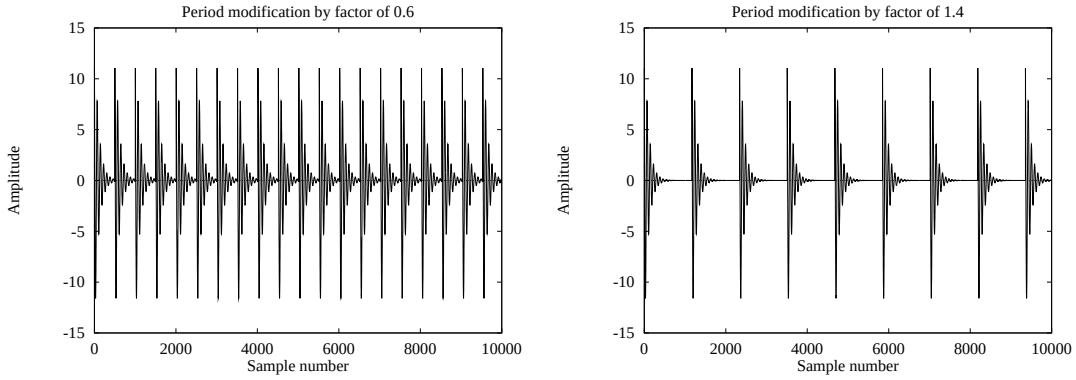


Figure 5.14: *Period modification of the Šilnikov orbit by a factor of* (a) 0.6 *and* (b) 1.4.

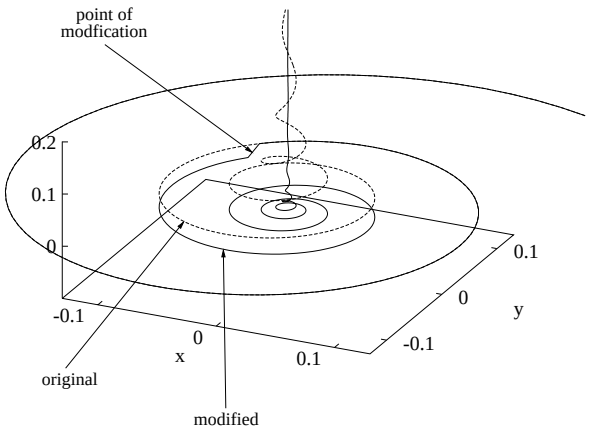


Figure 5.15: *Zoom in on the fixed point, showing a comparison of the original and the modified trajectories.*

5.4.4 Application to LP speech

Performing trajectory modifications when the system model is derived from a dataset, rather than a set of equations, will evidently be a more complex task. The stages of the algorithm outlined in Section 5.4.2 and the problems found are now discussed for the LP synthesised vowel /u/.

Embedding

The LP data first has a small amount of Gaussian white noise added, to give a signal–to–noise ratio of 20 dB. This adds some variation to the signal, thus making the manifold more than a single trajectory wide. The formation of local neighbourhoods can then be made by selecting near points from adjacent trajectories. The data is then embedded in three dimensions using time delay embedding with τ set at 12 samples, equal to the first minimum of the mutual information.

Fixed point location

Because of the asymptotic nature of the fixed point, the closer the trajectory comes to it, the greater the amount of time that will be spent in the region around it. Therefore, for sampled data, the Euclidean distance between subsequent samples will decrease as the trajectory moves towards the fixed point. The data sample which has the minimum distance between adjacent samples will be the closest data point to the fixed point. This is then used as the best known position of the fixed point, $\mathbf{x}_f$, in subsequent calculations.

Direction of manifolds

An index 1 saddle point has a two–dimensional stable manifold and a one–dimensional unstable manifold. The approximate directions of these manifolds are found by an eigenanalysis of the data trajectory about the fixed point. To do this, a tangent map, similar to that discussed in Section 3.4.2, can be formed. Taking $\mathbf{x}_f$, the closest data point to the fixed point, as the centre, a neighbourhood matrix is constructed from the M points within a hyper–sphere around $\mathbf{x}_f$. The neighbourhood matrix is then evolved forward a samples and recalculated. The tangent map then defines the linear transformation between the two local neighbourhood matrices. Its eigenvalues and eigenvectors are found by SVD. Figure 5.16 shows the eigenvectors found

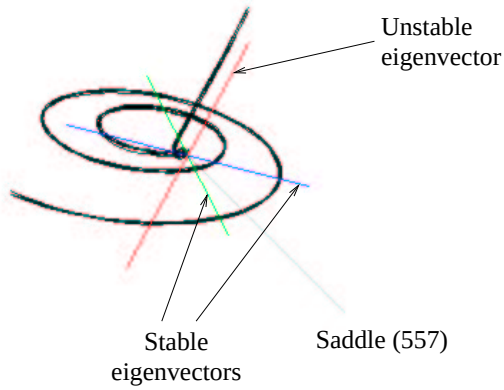


Figure 5.16: *Eigenvectors around the saddle point for the LP vowel /u/.*

by this method for the LP vowel /u/. The saddle point is marked and the three eigenvectors are shown. The largest eigenvector corresponds to the unstable manifold, and the two smaller eigenvectors are in the plane of the stable manifold. The parameters used in this analysis were 25 local neighbours, with an evolve length of 10 samples.

Perturbing trajectory

It is now possible to consider perturbing the trajectory, as shown in Figure 5.11, by moving the trajectory either towards or away from the stable manifold in the direction of the unstable manifold, as it enters the sphere of radius ϵ. However, an insurmountable problem is immediately encountered. Perturbing the trajectory implies moving it away from its existing course (defined by the locally linear tangent map calculated at each synthesis step) and into some other area of state space. This part of state space will not contain any data, and hence is not covered by the locally linear model. Therefore perturbing the trajectory moves it into a region of state space that is completely unmodelled, as shown in Figure 5.17. There is no information available to indicate how it should continue to evolve (contrary to the previous example with the set of equations that defined all points in state space), and so continuing the synthesis process after perturbing the trajectory is impossible.

This means that pitch modification by perturbing the trajectory and locally linear synthesis are not compatible, and leaves the problem of realistic pitch modification unresolved.

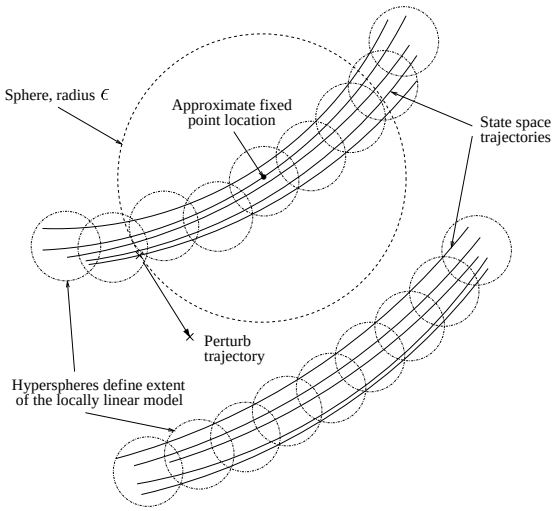


Figure 5.17: *Stylised diagram of state space around the saddle point, showing two sets of data trajectories. Perturbing trajectory implies moving into an area of state space not covered by the locally linear model.*

5.5 Conclusion

This chapter has considered the use of a locally linear model for nonlinear synthesis as implemented by Banbrook [5], which is based on ideas drawn from the Lyapunov exponent estimation algorithm. The technique has been shown to be capable of synthesising good quality speech with an 11 kHz bandwidth, although some reservations have been expressed about the periodic nature of the jitter and shimmer that the algorithm creates. However, this should be taken in the context of synthesis by the linear model of speech, which generally does not include any jitter or shimmer.

The method proposed by Banbrook to modify pitch has been shown to be unrealistic, since it does not take into account changes in the shape of the waveform with changes in pitch. Analysing pitch variations in real vowels, and then the simplified case of linear prediction synthesised vowels, led to the idea of modelling the phase space reconstructions as Šilnikov orbits. Then, taking the principle of perturbing state space trajectories by small changes from the controlling chaos literature, an algorithm for changing the period length of a Šilnikov orbit has been proposed. This was shown to be viable for period length modifications in a simple system modelled by a set of coupled differential equations, but failed when applied to speech data due to the type of modelling used. In essence, the locally linear model by definition can only model the small areas containing data. Perturbing the trajectory implies moving away

from the data, and hence outside the model. For this technique to work, a model of the whole of state space (or at least a complete model near the fixed point) is required.

Therefore, in summary, the locally linear synthesis model is limited in its application because of the inability to realistically modify pitch. Another factor that must be considered is its complexity. For every new synthesised point, it is necessary to calculate the neighbourhood and evolved neighbourhood matrices, and then calculate the tangent map between the two. Additionally, being an extension of a concatenation technique, the original data, from which the phase space reconstructions are made, must be stored. However, the technique would be useful as an add–on in a concatenative synthesiser when extended vowel sounds are required, as it can synthesise these sounds well for any required duration. It would then be necessary to use a technique such as PSOLA to impose the required pitch contour.

Chapter 6
Synthesis by global model

6.1 Introduction

In this chapter a global parametric model for speech synthesis is introduced.[1] It is based on a radial basis function neural network with a feedback loop, which operates as a free–running nonlinear oscillator to produce synthesised vowel sounds. After appropriate training, the network is expected to model the dynamics of the generating system, and as such should be capable of reproducing the natural variability of the original signal.

In the first part of the chapter some basic background on RBFs is presented. The idea of a global feedback loop in a neural network for speech synthesis has been used previously by several researchers, and their work is reviewed in the following section. The remainder of the chapter is devoted to the efforts made to implement a stable RBF synthesiser. The parameters of the network are examined in turn, in order to estimate suitable values for the resynthesis model, with performance measured in terms of mean square error (MSE). Next, resynthesis is attempted, but with little success. Some of the resynthesised waveforms have no characteristics of the original at all, indicating that the network has failed to learn the dynamics of the vowel sound completely. However a number of signals are similar to the original input training data, but with large spikes on the waveform. The cause of these spikes is investigated, and it is found that the function approximated by the network is more nonlinear than the original data, making it sensitive to small differences present in the feedback loop.

6.2 Radial basis functions

A radial basis function neural network is composed of three layers, made up of an input layer of source nodes, a nonlinear hidden layer and an output layer giving the network response. It can be used for interpolation and approximation, since it finds a surface in multi–dimensional space

[1] The term global is used in the sense “not locally–linear”.

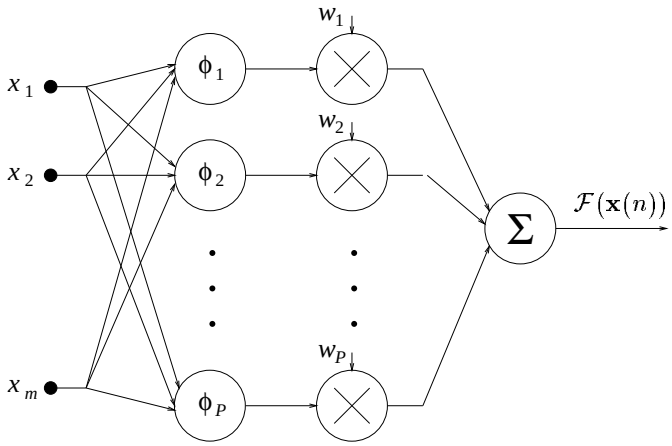


Figure 6.1: *Schematic of a RBF network.*

that is the closest fit to the input training data [13]. The closest fit will be measured by a statistic, such as the mean square error. The hidden layer performs a nonlinear transformation mapping the input space to a new space [111], in which the problem can be better solved. The output is the result of linearly combining the hidden space, multiplying each hidden layer output by a weight whose value is determined during the training process.

The general equation of an RBF network with an input vector $\mathbf{x}$ and a single output is

$$\mathcal{F}(\mathbf{x}(n)) = \sum_{j=1}^{P} w_j \phi(\|\mathbf{x} - \mathbf{c}_j\|) \tag{6.1}$$

where there are P hidden units, each of which is weighted by w_j. A schematic showing the general RBF network structure for an input vector of length m is presented in Figure 6.1. The hidden units, $\phi(\|\mathbf{x} - \mathbf{c}_j\|)$, are radially symmetric functions about the point $\mathbf{c}_j$, called a centre, in the hidden space, with $\|.\|$ being the Euclidean vector norm [112]. There are a number of possible choices for the radial basis function, such as multi–quadratic, cubic, or thin–plate spline, but the most common is the Gaussian [113]:

$$\phi(\gamma) = \exp\left(\frac{-\gamma^2}{2\sigma^2}\right) \tag{6.2}$$

σ^2 is the bandwidth of the function and controls the spread of each centre. The actual choice of nonlinearity does not appear to be crucial to the performance of the network [111, 114], and the use of a Gaussian function provides some useful mathematical properties which are described later. Therefore Gaussian centres are used in all of the RBF networks studied in this thesis.

There are two distinct strategies for training the network. The most common approach divides the problem into two steps. Firstly the centre positions and bandwidths are fixed using an unsupervised approach, not dependent on the network output. Then the weights are trained in a supervised manner so as to minimise an error function. The centres may be fixed in number and positioned according to the input data, either as a random subset of the input data points, or by the use of a clustering algorithm such as K–means [115]. Alternatively they can be fixed on a hyper–lattice structure which uniformly samples the state space and is data independent [116]. If the number of centres is not fixed, then the orthogonal least squares method can be used, which chooses centres one at a time until an adequate network has been formed [111]. The second approach is entirely supervised, with the centre positions and widths included with the weights in the error function. However minimising this global error function then becomes a nonlinear optimisation problem, which will be computationally intense, and the error function may have local minima [113]. Hence only the first approach is followed in this study.

The rationale behind the choice of RBFs, as opposed to other nonlinear models, is motivated by several factors. Papers by Broomhead and Lowe [117] and Moody and Darken [118] extended the use of RBFs from exact interpolation to the more general problem of approximation. Subsequent work by Park and Sandberg [119] proved that RBFs can have universal approximation capabilities. This makes RBFs appealing for modelling the dynamics of voiced speech. However, other neural network (NN) structures also have good approximation abilities, for example the multi–layer perceptron (MLP) [113]. The advantage of the RBF network over other NN architectures is the ease of training: with the centre positions fixed, training the weights is a linear in the parameters problem and so can be solved by least squares techniques. The amount of training data required is also less. Additionally, as will be discussed subsequently, if the centre positions are fixed on a data independent hyper–lattice, then the complexity is further reduced, and the constant structure may offer possibilities for pitch modification in the state space domain. This is not possible with the MLP because the hidden layer activation functions are not localised in the hidden space.

6.3 Other nonlinear synthesisers

Work carried out by Kubin and Birgmeier is the only other reported attempt made to use a RBF network approach to speech synthesis. They propose the use of a nonlinear oscillator, with no

external input and global feedback in order to perform the mapping

$$x(n) = \mathcal{A}(\mathbf{x}(n-1)) \tag{6.3}$$

where $\mathbf{x}(n-1)$ is the delay vector with non–unit delays, and $\mathcal{A}$ is the nonlinear mapping function [120].

The initial approach taken ([121, 122]) used a Kalman–based RBF network, which has all of the network parameters trained by the extended Kalman filter algorithm. The only parameter that must be specified is the number of centres to use. This gives good prediction results, but there are many problems with resynthesis. In particular, they report that extensive manual fine–tuning of the parameters such as dimension, embedding delay and number and initial positions of the centres are required. Even with this tuning, synthesis of some sounds with complicated phase space reconstructions does not work [120].

In order to overcome this problem, Kubin resorted to a technique that uses all of the data points in the training data frame as centres [120]. Although this gives correct resynthesis, even allowing the resynthesis of continuous speech using a frame–adaptive approach, it is unsatisfactory due to the very large number of varying parameters, and cannot be seen as actually learning the dynamics of the speech generating system.

In a related piece of work, Kubin and Kleijn implemented a time–scale modification algorithm, which can be applied to continuous speech [123]. This again uses a frame–adaptive approach, but is based on a nearest–neighbours lookup table, rather than a RBF network. Again this is a very computationally costly approach, not dissimilar to the locally linear model described in Chapter 5.

Following their dynamical analysis of the Japanese vowel /a/, Tokuda *et al.* constructed a feed–forward neural network to perform synthesis. Their structure has three layers, with five neurons in the input layer, forty neurons in the hidden layer, and one in the output layer. The time delay in the input delay vector is set at $\tau = 3$. The excitation function of the hidden nodes is of the form

$$\Phi(\zeta) = \frac{-2\zeta}{(1+\zeta^2)^2}$$

and the weights are learnt by back propagation. Using global feedback, they report successful

resynthesis of the Japanese vowel /a/. The signal is noisy, but preserves natural human speech qualities. No further results in terms of speech quality or resynthesis of other vowels are given. From this limited information, it would seem that this technique is successful, although the resulting speech may not be of a high quality. In addition, the use of back propagation makes the learning process more complex than a standard RBF technique.

Recently an alternative neural network approach has been proposed by Narashimhan *et al*. This involves separating the voiced source from the vocal tract contribution, and then creating a nonlinear dynamical model of the source [124]. This is achieved by first inverse filtering the speech signal using a LP model to obtain the LP residue. Next the residue waveform is low–pass filtered at 1 kHz, then normalised to give a unit amplitude envelope. This processed signal is used as the training data in a time delay neural network with global feedback. The NN structure reported is extremely complex, consisting of a 30 tap delay line input and two hidden layers of 15 and 10 sigmoid activation functions. The network is then trained using back propagation through time, requiring 200 passes through the 1500 samples of input data. Finally, the NN model is used in free–running synthesis mode to recreate the voiced source. This is applied to a LP filter in order to synthesise speech. Although the complexity seems excessive, and the idea of linearly separating the excitation from the vocal tract contribution before then applying a nonlinear technique is questionable, the interest of this study is their results on pitch jitter. They show that the NN model successfully preserves the jitter of the original excitation signal, indicating that dynamic modelling can correctly model the natural variations present in speech.

6.4 Free–running RBF for synthesis

Because of its good approximation capabilities, ease of training (compared to other NNs), and the possibilities for pitch modification (as described in Chapter 7), a RBF network was chosen as the architecture to use for nonlinear modelling of the voiced speech sounds. Following from the work of Kubin *et al.*, a nonlinear oscillator structure is used, as shown in Figure 6.2.

The RBF network is used to approximate the underlying nonlinear dynamics of a particular stationary voiced sound, by training it to perform the prediction

$$x_{i+1} = \mathcal{F}(\mathbf{x}_i) \tag{6.4}$$

where $\mathbf{x}_i = \{x_i, x_{(i-\tau)}, \ldots, x_{(i-(m-1)\tau)}\}$ is a vector of previous inputs spaced by some delay

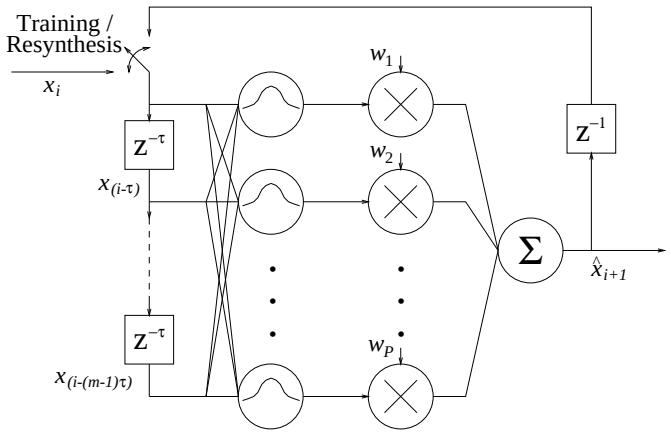


Figure 6.2: *RBF network with global feedback for free–running synthesis.*

τ samples, and $\mathcal{F}$ is a nonlinear mapping function. Considering the problem from a nonlinear dynamical theory perspective, this can be viewed as a time delay embedding of the speech signal into an m–dimensional state space to produce a state space reconstruction of the original d–dimensional system attractor. It is therefore logical to choose τ as the first minimum of the mutual information function. The other parameters that must be chosen are the bandwidth, the number and position of the centres, the embedding dimension, and the length of training data to be used. With these set, the determination of the weights is linear in the parameters and may be solved by standard matrix inversion techniques.

Once the weights have been determined, the network can be switched from training to synthesis mode: the input signal is removed and the delayed network output is fed back into the delay line to form a free–running RBF network, allowing any duration of signal to be synthesised. If the dynamics have been correctly learnt, then the synthesised signal should have the same invariant properties (*e.g.* Lyapunov exponents) as the original training data.

It is not obvious how the network parameters can be jointly optimally chosen. Birgmeier's extended Kalman filter method was an attempt at joint optimisation, but it was not found to be successful when applied to synthesis. The approach adopted in this work is to consider each parameter separately, holding the others constant, and assess the network performance. However, in assessing the performance a difficulty arises. The ideal criteria would be "at this parameter value synthesis will work", but it is not feasible to check this, since another of the parameters may be incorrect and this will give incorrect synthesis results. Additionally, it is difficult to quantify how well the synthesis has been performed, so it is not possible to tell

whether we are approaching or moving away from a good result. In fact, the only criteria actually available is the mean square error for forward prediction. Generally, the one step ahead prediction MSE will be measured, but it should be noted that this can only optimise the predictor, not indicate whether resynthesis will be correct.

Three types of centre positioning strategy were considered. Having specified the number of centres to use, they are:

1. Data subset. Centres are picked as points from around the state space reconstruction. They are chosen pseudo–randomly, so as to give an approximately uniform spacing of centres about the state space reconstruction.

2. K–means. Using the adaptive K–means clustering algorithm [115], the state space embedding vectors are partitioned into K regions (corresponding to the number of centres) and then the best vector to represent each region is chosen as a centre.

3. Hyper–lattice. An alternative, data independent approach is to spread the centres uniformly over an m–dimensional hyper–lattice. This can be viewed as a uniform sampling of the state space, and considerably reduces the complexity of the system since the same centre positions are used regardless of the topological structure of the state space reconstruction.

6.4.1 Bandwidth

For the data subset and K–means case, a heuristic measure for the bandwidth is [13]:

$$\sigma = \frac{d}{\sqrt{2P}} \tag{6.5}$$

where P is the number of centres and d is the maximum distance between the chosen centres. This is meant to ensure that the Gaussian functions are neither too peaky nor too flat. To verify this measure for voiced speech signals, the bandwidth was varied over the range 0.01 to 1.5 for 12 different signals (the vowels /i/, /ɑ/, /u/, for the female speakers CA, KH and the male speakers MC and PB). The MSE for one step ahead prediction on an unseen validation set for each case was then obtained.[2] The results of this experiment are shown in Figure 6.3. There is

[2] All MSE values reported in this thesis are relative to the input data normalised to an absolute maximum amplitude of one, rather than the more conventional normalised MSE which refers to unit variance input data. This

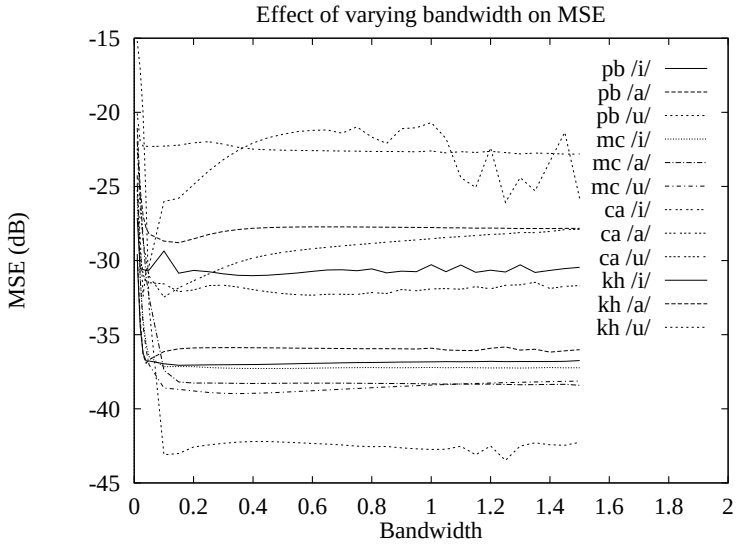


Figure 6.3: *Effect on MSE of varying bandwidth for* 100 *centres chosen as a subset of the training data. The other parameters were a training length of* 4000*, embedding dimension of* 3*, and embedding delay set at first minimum of mutual information function.*

some variation in the results between different signals, but a good general choice for σ^2 would be around 0.2. The maximum distance bandwidth measure only corresponds to the minimum MSE for two signals (MC /i/ and KH /u/), giving values in the range $0.01 \leq \sigma^2 \leq 0.06$. Therefore in general, best MSE performance is gained using a fixed bandwidth of around 0.2.

Bandwidth selection for the hyper–lattice was made in a similar way, by training the network weights over the data base and measuring the validation MSE. The results of the experiment, with the bandwidth varied between 0.05 and 1.5 are seen in Figure 6.4. A value of above 0.6 appears to be suitable, since the curves all flatten out around this point.

6.4.2 Centre location strategy

The validation MSE for the three centre location techniques (data subset, K–means, hyper-lattice) was calculated over the data base. The results, averaged by speaker and using the above bandwidth results, are shown in Figure 6.5. The techniques all give very similar performance. Averaged over the whole database, the K–means (at -32.55 dB) and subset method (at -32.59 dB) are slightly superior to the hyper–lattice (at -32.09 dB). Given the much greater com-

change is necessary because of the constant size hyper–lattice.

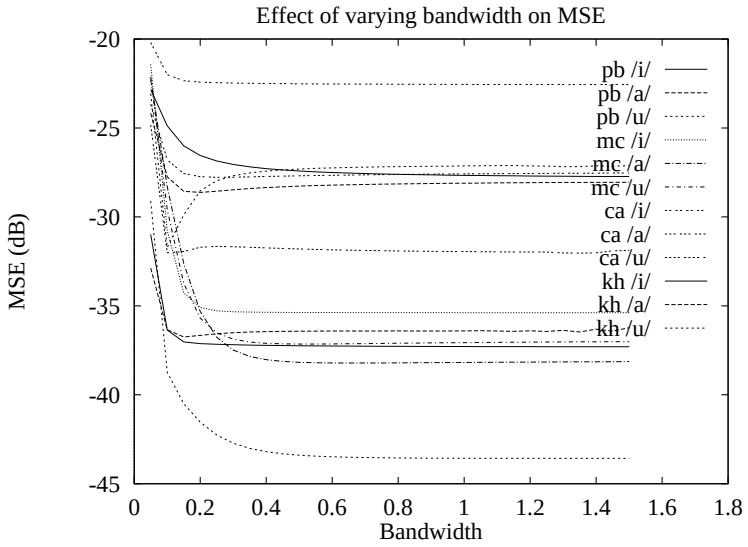


Figure 6.4: *Effect on MSE of varying bandwidth for* 64 *centres on a 3D hyper–lattice. The other parameters were a training length of* 4000 *samples and embedding delay set at first minimum of mutual information function.*

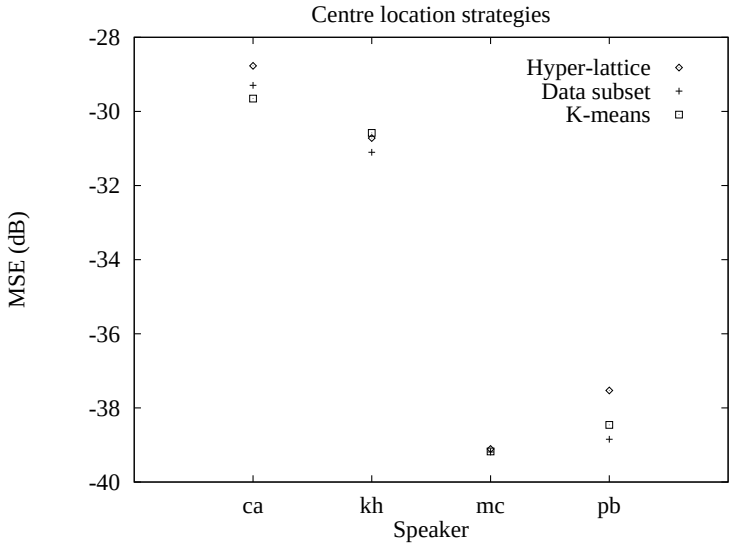


Figure 6.5: *Comparison of hyper-lattice, data subset and K–means centre location strategies. The parameters in the experiment were* 64 *centres, training data length of* 2000*, embedding dimension of* 3*, embedding delay set at first minimum of mutual information function, bandwidth of* 0.2 *for data subset and K–means, and bandwidth of* 0.8 *for hyper–lattice.*

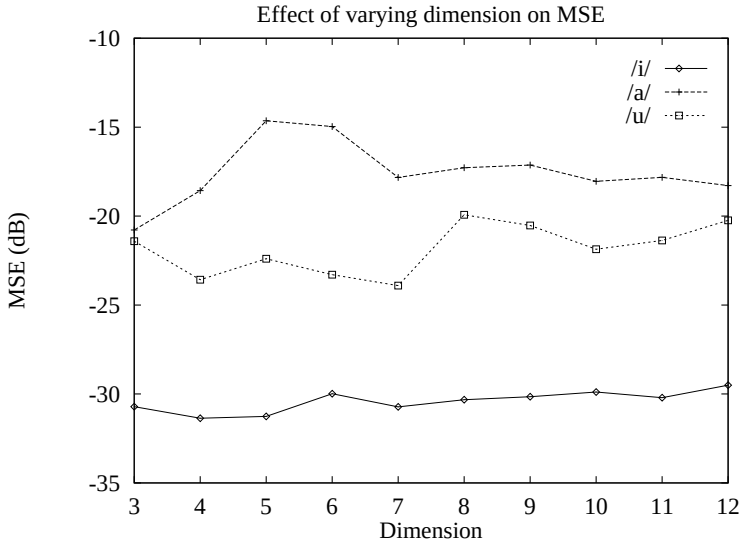


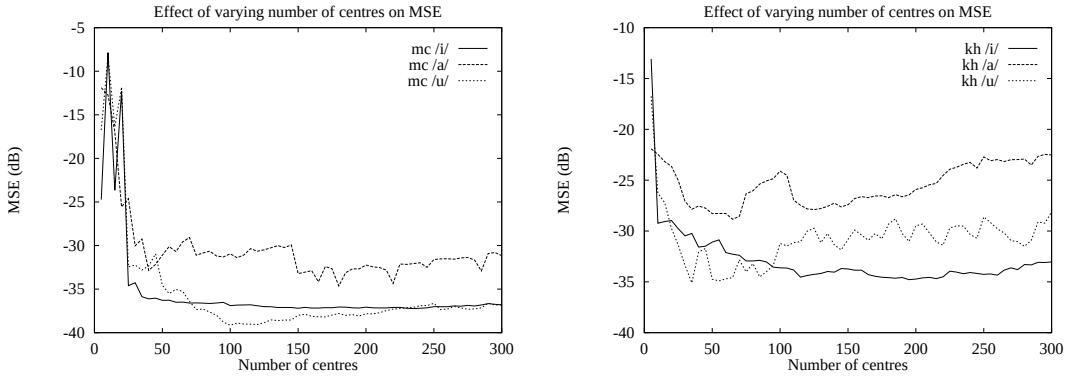
Figure 6.6: *Effect on MSE of varying dimension for* 100 *centres selected as a subset of the training data. The other parameters were a training length of* 2000 *samples, bandwidth set with the maximum distance measure, and embedding delay set at first minimum of mutual information function.*

plexity of the K–means technique, it was decided to only use the data subset and hyper–lattice methods.

6.4.3 Dimension

Previous results for the correlation dimension of vowels have given values of between two and three [5, 67, 82, 89]. Takens' theorem therefore advises an embedding dimension of seven. However, it has already been seen in earlier chapters that the vowel state space reconstructions are fully unfolded (*i.e.* no crossings) in three dimensions, indicating that this is an adequate value. An experiment was performed to examine the effect of changing the embedding dimension over the data base. The results, averaged by vowel, are shown in Figure 6.6, for $3 \leq m \leq 12$.

These results do not provide a definite answer to the embedding question, but the use of three dimensions does not seem unreasonable from a MSE point of view. Therefore in the following experiments a value of $m = 3$ is used.



(a) **(b)**

Figure 6.7: *Effect of varying the number of centres on MSE for* (a) *speaker MC and* (b) *speaker KH. The parameters in the experiment were an embedding dimension of* 3*, a training length of* 2000*, with centres chosen as a subset of the training data and bandwidth by the maximum distance method.*

6.4.4 Number of centres

A good choice of the number of centres to use in the network is extremely important. An insufficient number of centres will not adequately model the data, but choosing too many will result in over–fitting and a loss of generalisation [113]. To check for over–fitting, a standard method is to train the network with an increasing number of centres and examine how the MSE on an unseen validation data set changes. While an increasing number of centres will give a closer and closer approximation to the training data, there will come a point when the network starts to lose its generalisation properties, and the validation MSE will increase.

This principle was applied to the vowel database, as seen in Figure 6.7. This shows evidence of over–fitting in speakers MC and KH, particularly clearly in the examples MC /u/, KH /ɑ/ and KH /u/. Similar results were found for the two other speakers, showing that a good choice for the number of centres is between 50 and 150.

6.4.5 Lattice size

The hyper–lattice approach is data independent, but the actual size of the lattice in comparison to the size of the data state space reconstruction may be important. In all of the experiments, the data is normalised to have a maximum absolute amplitude of one. The width of an edge of

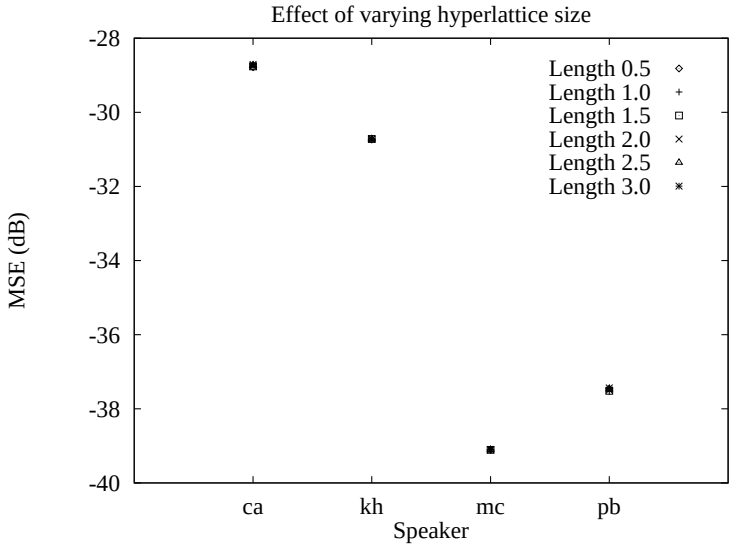
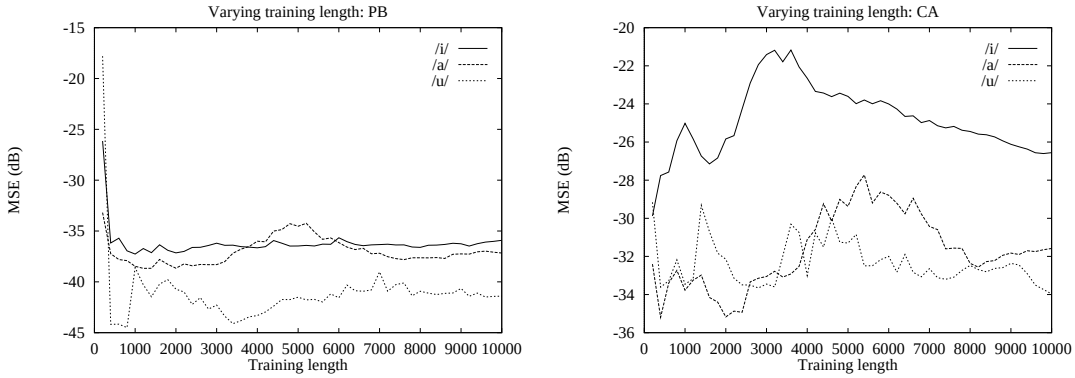


Figure 6.8: *Effect on MSE of varying hyper–lattice size for* 64 *centres on a 3D hyper–lattice. The other parameters were a training length of* 2000 *samples, bandwidth of* 0.8 *and embedding delay set at first minimum of mutual information function. Note that the MSE results for different lattice sizes for each speaker are almost identical, leading to superimposed points in the figure.*

the hyper–lattice will then define approximately where the centres will be located with respect to the state space data vectors. Using a 3D hyper–lattice with 64 centres, the width of the lattice was varied from 0.5 to 3.0. The results of this experiment are shown in Figure 6.8, averaged by speaker, and show that lattice size does not effect MSE performance. Therefore a lattice size of 1.0, corresponding to a lower "corner" at $(-0.5, \ldots, -0.5)$ is chosen, since this gives a hyper–lattice that is comparable in size to the data state space reconstruction.

6.4.6 Training length

The length of training data presented to the network is also very important, since the aim is to learn the dynamics of the underlying system, including the natural inter–pitch variability (*i.e.* jitter and shimmer). This means that a large length of data, and hence number of pitch periods, is advantageous. However, long training lengths imply long learning times which are undesirable. Training the network for different lengths, varying between 200 samples and 10000 samples does not resolve this from a MSE point of view. Figure 6.9 shows the results for speakers PB and CA. The results for speakers MC and KH are equally unclear, but it does seem reasonable to use a length of approximately 1000 to 2000 samples. This will include 10



(a) **(b)**

Figure 6.9: *Effect of varying the training length on MSE for* (a) *speaker PB and* (b) *speaker CA. The parameters in the experiment were an embedding dimension of* 3*,* 64 *centres chosen as a subset of the training data and a bandwidth of* 0.2*.*

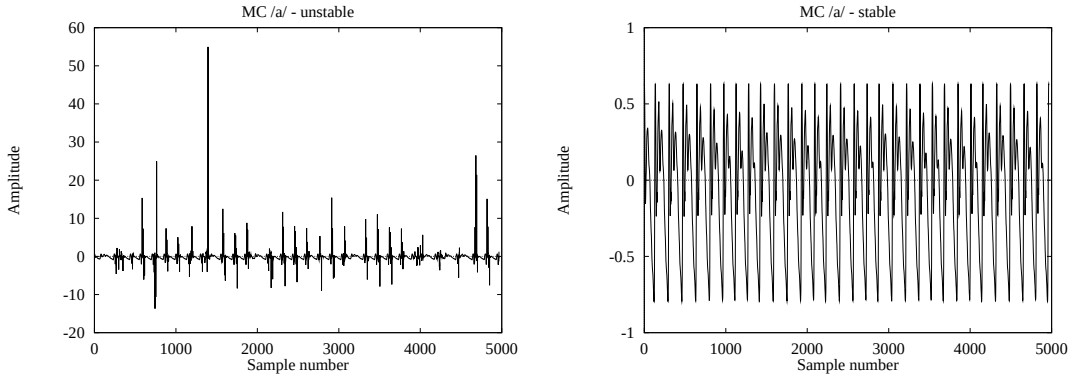
to 20 pitch periods, but does not lead to excessive training times.

6.4.7 Resynthesis

From this analysis, an initial set of parameters with which to attempt resynthesis were chosen. The parameters were set at the following values:

Bandwidth = 0.8 for hyper–lattice, 0.2 for data subset; Dimension = 3; Number of centres = 64; Hyper–lattice size = 1.0; Training length = 1000;

For each vowel in the database, the weights were learnt, with the centres either on the 3D hyper–lattice, or chosen as a subset of the training data. The global feedback loop was then put in place to allow free–running synthesis. The synthesis results were extremely variable. In a few cases, successful resynthesis was achieved. Other results gave varying degrees of success, from constant (sometimes zero) outputs, through periodic cycles not resembling the original speech signal and noise–like signals, to extremely large spikes at irregular intervals on otherwise correct waveforms. The data subset method correctly resynthesised the signals PB /i/, MC /ɑ/ and KH /u/. The hyper–lattice method correctly resynthesised PB /i/, MC /u/ and CA /u/. In all cases, success or failure was judged by visual inspection of the synthesised waveform over 5000 samples.



(a) **(b)**

Figure 6.10: *Example of sensitivity to small changes in network parameters, for the vowel /ɑ/, speaker MC.* (a) *incorrect resynthesis with a training length of* 1800 *and* (b) *correct resynthesis with a training length of* 2000. *Note the change in y–axis scale.*

6.5 Stability issues

The initial synthesis results are disappointing, since it is unclear how the parameters of the RBF network can be chosen in any satisfactory manner. Birgmeier [122] reports similar difficulties, and also notes that manually experimenting with different parameter values can improve the resynthesis. This was also found to be the case with our network. Changes in embedding dimension, bandwidth and training length were found to allow successful resynthesis for a number of signals where it had previously failed. However it should be noted that increasing embedding dimension does not automatically result in improved synthesis. It was also found that very small changes in the parameters could make the difference between success and failure of the synthesis. An example of this is the vowel /ɑ/, speaker MC. Using the hyper–lattice method, a change of 200 samples in the training length from 1800 to 2000 is enough to allow correct resynthesis, as seen in Figure 6.10 (note that a training length of 1000 also fails). The example at a training length of 1800 has a large number of spikes on an otherwise correct waveform, but moving to the 2000 training length example these disappear.

The fact that these small changes can make such an impact on the resynthesis implies that the RBF network must be very sensitive. Two possible reasons for this sensitivity and the subsequent difficulties with resynthesis may be i) poor numerical stability of the weights or ii) problems with the actual approximated mapping function $\mathcal{F}$.

6.5.1 Analysis of weights

Condition number

With the centre positions fixed, either on the hyper–lattice or chosen from the training data, the weights are determined by minimising a sum of squares error function, $E_s(\hat{\mathcal{F}})$, over the N samples of training data:

$$E_s(\hat{\mathcal{F}}) = \frac{1}{2}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2 \tag{6.6}$$

where $\hat{x}_i$ is the network approximation of the actual speech signal x_i. Incorporating Equation 6.1 into the above and differentiating with respect to the weights, then setting the derivative equal to zero gives the least-squares problem [113]. This can be written in matrix form as

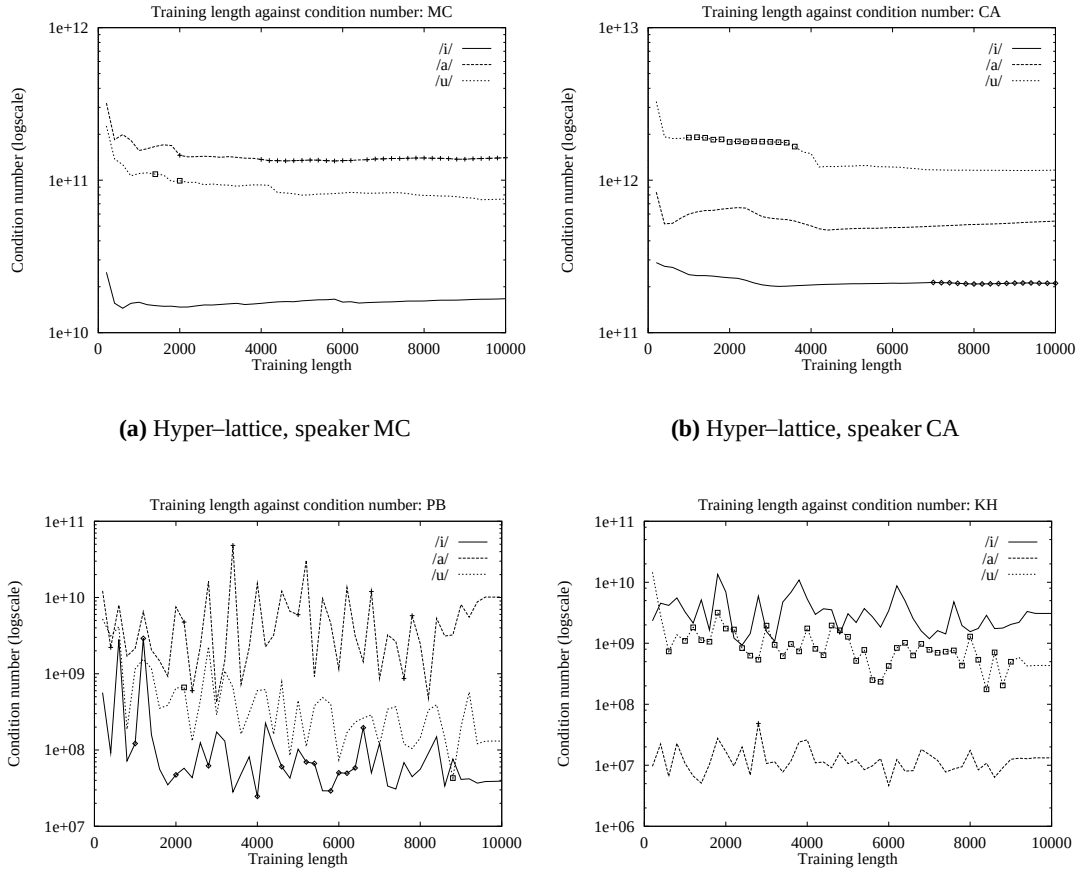
$$(\Phi^T\Phi)\,\mathbf{w}^T = \Phi^T\mathbf{x} \tag{6.7}$$

where Φ is an N x P matrix of the outputs of the centres (with P centres in total); $\mathbf{x}$ is the target vector of length N, made up of the training data; and $\mathbf{w}$ is the P length vector of weights. Setting $\theta = \Phi^T\Phi$, then the weights are found as

$$\mathbf{w}^T = \theta^{\dagger}\Phi^T\mathbf{x} \tag{6.8}$$

where $\theta^{\dagger}$ is the pseudo–inverse of θ. SVD (as described in Appendix A) is used in this calculation to ensure good numerical stability, even though θ is not actually singular. Thus the numerical properties of the matrix θ will influence the solution found for the weights. Examining the condition number of this matrix (defined as the ratio of the largest singular value to the smallest) may provide some information as to why certain parameter sets give correct re-synthesis, whereas others do not. The training data length appears to have a significant impact on correct synthesis, so it was decided to repeat the varying training length experiment while recording the condition number of θ. All the other network parameters were held constant as previously described.

The condition number results are shown in Figure 6.11, with examples for the hyper–lattice for speakers MC and CA, and for the data subset for speakers PB and KH. In each case the training length was incremented in steps of 200, from 200 samples to 10000 samples. The condition number against training length result is plotted as a continuous curve. Superimposed onto each



(a) Hyper–lattice, speaker MC

(b) Hyper–lattice, speaker CA

(c) Data subset, speaker PB

(d) Data subset, speaker KH

Figure 6.11: *Condition number against training length, calculated at* 200 *sample intervals (continuous curves), with correct resynthesis results superimposed (points).*

curve is a set of points which indicate successful synthesis at a particular training length.

It can be seen that the condition number in the hyper–lattice case is significantly greater than in the data subset case. However this does not appear to have any bearing on whether correct resynthesis is obtained or not. In all cases the condition number is large, indicating ill–conditioning, but some examples with very high condition number still give good resynthesis (*e.g.* hyper–lattice, speaker CA, vowel /u/).

It is noted that Equation 6.7 may be solved in two other ways. It can be re–written as $\Phi\mathbf{w}^T = \mathbf{x}$, thus requiring the pseudo–inverse of the $N\mathrm{x}P$ matrix Φ to be found. This is considerably more computationally complex than the pseudo–inverse calculation of the $P\mathrm{x}P$ matrix θ, but

the numerical results are not significantly different. Alternatively, the matrix inversion can be avoided by using the Householder transformation to reduce the full matrix to an upper triangular one [125]. The weights are then found by back substitution. However this does not improve the synthesis results either, and so is not considered further.

Magnitude of weights

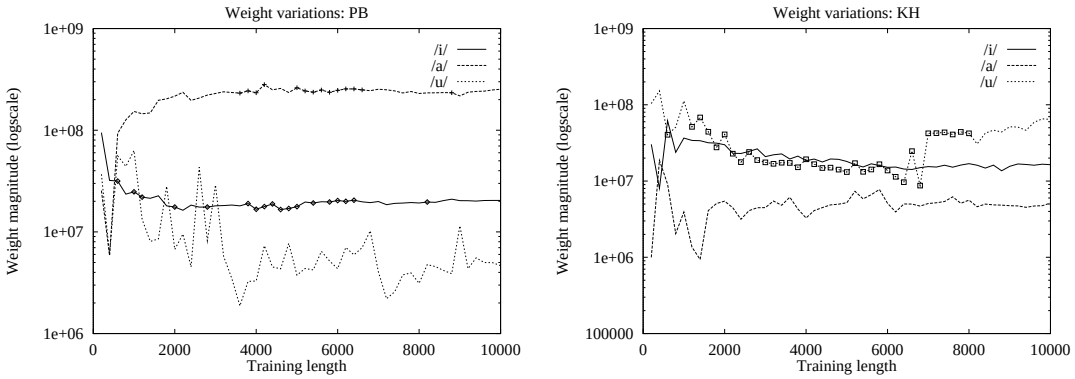
An alternative source of error may be the actual size of the weights once calculated. Numerical problems in the network calculation may occur if the weights are very large [126]. Therefore the average of the weight vector was calculated for each training data length, again varying from 200 samples to 10000 samples in steps of 200. These average weight variation results are shown in Figure 6.12, with correct resynthesis points also marked. Examples are shown for speakers PB, KH using the hyper–lattice, and speakers MC, CA using the data subset. Once again the results are counter–intuitive, with correct resynthesis occurring for cases with large weight values, and failing at some of the lower values. Once again the hyper–lattice results are several magnitudes greater than the data subset values, but this does not appear to effect which signals are correctly resynthesised.

Similar results are also obtained when plotting the magnitude of the largest single weight at each training length, indicating that the resynthesis problem is not caused by a large 'stray' weight value.

6.5.2 Analysis of mapping function

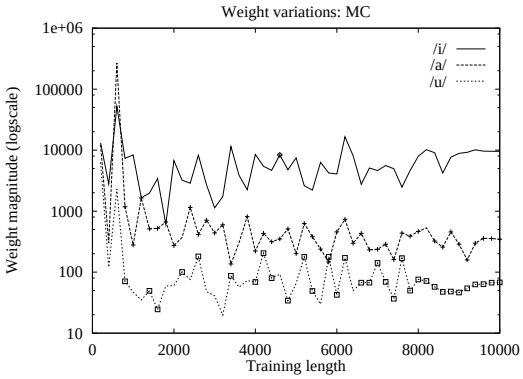
After the network has been trained, it will operate according to the mapping function $\hat{\mathcal{F}}$, which is an approximation to the underlying speech system function $\mathcal{F}$. Another possible source of instability is the stability of this approximated function. When the free–running synthesis output is either constant, a limit–cycle, or noise, it can be assumed that the training has failed entirely to teach the network the dynamics of the underlying system. However, the majority of erroneous outputs are spikes on otherwise correct waveforms, as shown previously in Figure 6.10. Here the system dynamics have been correctly learnt to some extent, and it is possible that instabilities in parts of the mapping function are responsible for the unstable waveform seen at the network output.

Therefore it is of interest to examine the properties of the approximated function $\hat{\mathcal{F}}$. One way

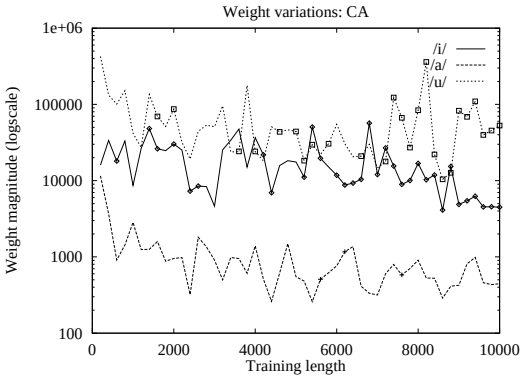


(a) Hyper–lattice, speaker PB

(b) Hyper–lattice, speaker KH



(c) Data subset, speaker MC



(d) Data subset, speaker CA

Figure 6.12: *Average weight magnitude against training length, calculated at* 200 *sample intervals (continuous curves), with correct resynthesis results superimposed (points).*

to do this is to consider the local Lyapunov exponents of $\hat{\mathcal{F}}$. Lyapunov exponents are generally calculated in the form of a global average over a whole system in order to assess the presence of chaos, as described in Chapter 3. However, it is also possible to look at *local* Lyapunov exponents at individual time steps, which allows local instabilities to be gauged. In contrast to the Lyapunov exponent calculations performed in Chapter 3, which used the speech time series, the equations of the approximated mapping function in terms of the RBF network parameters can be written down, and therefore the local exponents can be calculated analytically at each step.

In order to formulate the problem, it is assumed that the predictor is sufficiently well trained so that the following approximation can be made:

$$x(i+1) = \hat{\mathcal{F}}(\mathbf{x}(i)) = \sum_j w_j \phi_{ij} \tag{6.9}$$

i.e. it is a perfect predictor. The Lyapunov exponents are found from the Jacobian of the map along the orbit in state space, so the state space mapping must be considered:

$$\mathbf{x}(i+1) = \mathcal{G}(\mathbf{x}(i)) \tag{6.10}$$

If the orbit is perturbed by some small amount $\delta\mathbf{w}(i)$, then the linearised evolution of the perturbations is governed by the Jacobian of the map $\mathcal{G}(\mathbf{x}(i))$ [74]:

$$\delta\mathbf{w}(i+1) = \mathbf{DG}(\mathbf{x}(i))\,\delta\mathbf{w}(i) \tag{6.11}$$

where $\mathbf{DG}(\mathbf{x}(i)) = \partial\mathcal{G}(\mathbf{x})/\partial\mathbf{x}$ is the Jacobian. When finding the global, average Lyapunov exponents, a series of Jacobians are multiplied together to average over the whole orbit. However the local exponents are found from a smaller number of Jacobians. Here the stability at each time step is required, so just one Jacobian matrix is used. The local Lyapunov exponents are then the logarithms of the eigenvalues of the matrix [74]:

$$[\mathbf{DG}(\mathbf{x})(\mathbf{DG}(\mathbf{x}))^T]^{\frac{1}{2}} \tag{6.12}$$

Referring to Equation 6.10, $\mathbf{x}(i+1)$ can be written as

$$\mathbf{x}(i+1) = \begin{bmatrix} x_1(i+1) \\ x_2(i+1) \\ x_3(i+1) \end{bmatrix} = \begin{bmatrix} x(i+\tau) \\ x(i) \\ x(i-\tau) \end{bmatrix} \tag{6.13}$$

and

$$\mathbf{x}(i) = \begin{bmatrix} x_1(i) \\ x_2(i) \\ x_3(i) \end{bmatrix} = \begin{bmatrix} x(i) \\ x(i-\tau) \\ x(i-2\tau) \end{bmatrix} \tag{6.14}$$

Thus $\mathcal{G}$ can be written as

$$\begin{aligned} x_1(i+1) &= \hat{\mathcal{F}}(\mathbf{x}(i)) \\ x_2(i+1) &= x_1(i) \\ x_3(i+1) &= x_2(i) \end{aligned} \tag{6.15}$$

Considering the first element of the first row of the Jacobian, $\partial \mathcal{G}_1 / \partial x_1$:

$$\frac{\partial \mathcal{G}_1}{\partial x_1} = \frac{\partial x_1(i+1)}{\partial x_1} \tag{6.16}$$

$$= \frac{\partial \hat{\mathcal{F}}(\mathbf{x}(i))}{\partial x_1} \tag{6.17}$$

$$= \frac{\partial}{\partial x_1} \left[\sum_j w_j \exp\left(-\alpha \|\mathbf{x}(i) - \mathbf{c}_j\|^2\right) \right] \tag{6.18}$$

$$= -2\alpha \sum_j w_j (x(i) - c_{1j}) \exp\left(-\alpha \|\mathbf{x}(i) - \mathbf{c}_j\|^2\right) \tag{6.19}$$

Similarly, the other elements of the first row are found as

$$\frac{\partial \mathcal{G}_2}{\partial x_1} = -2\alpha \sum_j w_j (x(i-\tau) - c_{2j}) \exp\left(-\alpha \|\mathbf{x}(i) - \mathbf{c}_j\|^2\right) \tag{6.20}$$

and

$$\frac{\partial \mathcal{G}_3}{\partial x_1} = -2\alpha \sum_j w_j (x(i-2\tau) - c_{3j}) \exp\left(-\alpha \|\mathbf{x}(i) - \mathbf{c}_j\|^2\right) \tag{6.21}$$

Since the second and third rows are simply time delays of the first, the partial derivatives are

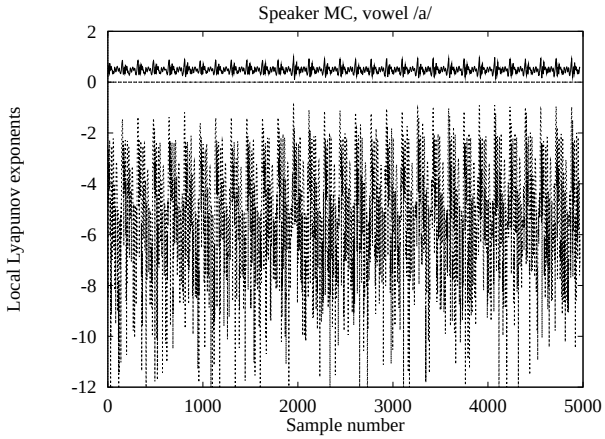


Figure 6.13: *Local Lyapunov exponents for speaker MC, vowel /a/, for a training length of* 2000 *samples, using the original signal samples in the Jacobian calculation. The exponents are expressed in bits/sample.*

easily found as

$$\frac{\partial \mathcal{G}_2}{\partial x_1} = 1;\ \frac{\partial \mathcal{G}_2}{\partial x_2} = 0;\ \frac{\partial \mathcal{G}_2}{\partial x_3} = 0 \tag{6.22}$$

and

$$\frac{\partial \mathcal{G}_3}{\partial x_1} = 0;\ \frac{\partial \mathcal{G}_3}{\partial x_2} = 1;\ \frac{\partial \mathcal{G}_3}{\partial x_3} = 0 \tag{6.23}$$

Therefore the Jacobian is found as

$$\mathbf{DG}(\mathbf{x}) = \begin{bmatrix} \partial \mathcal{G}_1/\partial x_1 & \partial \mathcal{G}_1/\partial x_2 & \partial \mathcal{G}_1/\partial x_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.24}$$

The eigenvalues of Equation 6.12 can be found by SVD. They will all be positive, since $\mathbf{DG}(\mathbf{x})(\mathbf{DG}(\mathbf{x}))^T$ is real and symmetric, and can thus be scaled according to the $1/2$ power. The local Lyapunov exponents for time step i are then found from the logarithms of the eigenvalues.

As an example, the local Lyapunov exponents from the network for speaker MC, vowel /a/, training length 2000 (synthesised signal shown in Figure 6.10(b)) are presented in Figure 6.13. The partial derivatives in Equations 6.19 to 6.21 were evaluated using the original speech data signal, not the feedback network output. The results show a positive–zero–negative set of expo-

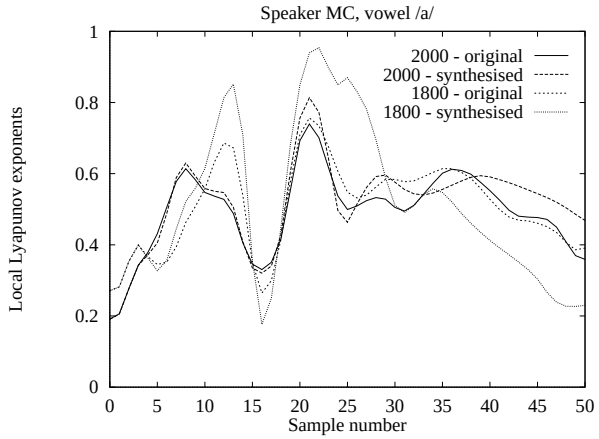


Figure 6.14: *Zoom on local Lyapunov exponents for speaker MC, vowel /a/, for training lengths* 1800 *and* 2000 *samples. Examples using both the original signal and network feedback signal in the Jacobian calculation are presented, with the exponents expressed in bits/sample.*

nents across the whole portion analysed. Although a positive exponent in the global exponents would indicate the presence of chaos, this interpretation should not necessarily be applied here. Rather, the local exponents provide a measure of the local sensitivity of the network to changes in the input. The network trained with the same vowel, but with 1800 samples of training data, has been shown to suffer instabilities. Therefore to further study this problem, the largest Lyapunov exponent for both cases was examined in detail. Figure 6.14 shows four largest local Lyapunov exponent curves over the first 50 samples of data, for both the 1800 and 2000 sample training lengths, using both the original speech data and also the network feedback signal in the calculation of the partial derivatives.

Both the original signal and feedback signal in the 2000 sample training length case settle onto a stable (non–identical) pattern, whereas the differences between the two 1800 sample training length cases are more marked. The slight glitches at around 12 and 20 samples present in the original signal exponents are amplified when the feedback signal is used. Because $\hat{\mathcal{F}}$ is trained using only the signal data, which occupies a compact manifold, the behaviour of trajectories outside this manifold is not well defined. Hence, once the trajectory leaves the manifold, the network will output erroneous data until some factor brings the trajectory back into the grasp of the model. Since the RBF network uses kernels with Gaussian responses, where the kernel output decreases with increasing distance from the centre, then stray trajectories will be pulled back towards the centres. This, coupled with the fact that the network may have greater

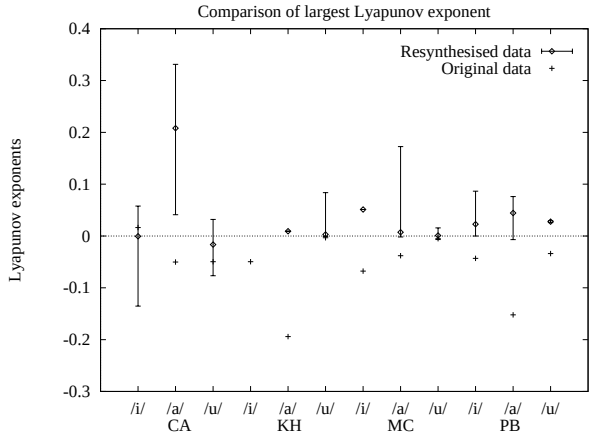


Figure 6.15: *Comparison of largest global Lyapunov exponents from synthesised and original data time series, across the database. The error bars indicate minimum and maximum values about the mean value for the resynthesised data results. Note there is no resynthesised data Lyapunov exponent for KH /i/, since no successful synthesis was achieved. The exponents are expressed in bits/sample.*

sensitivity to small changes in conditions, may explain why the large spikes occur and how the trajectory then temporarily returns to the correct orbit. It is conjectured that a slight increase in the positive local Lyapunov exponent at certain points on the orbit, coupled with the fact that the predictor is not perfect and therefore feeds back samples slightly different from the original signal, is sufficient to push the trajectory outside the manifold and cause unstable behaviour.

Similar results to those presented for MC, /ɑ/ are found across the whole database, with an increased positive exponent for networks which output signals with spikes compared to those that perform resynthesis correctly. This was further investigated by measuring the global Lyapunov exponents from the synthetic time signals, using the Lyapunov exponent estimation algorithm as implemented by Banbrook *et al.* [6] (see Chapter 3 for further explanation). Results from all of the correctly synthesised signals for each vowel example were averaged and then compared to the result from the original speech signal. A comparison of the largest global Lyapunov exponent, λ_1, between the original and synthesised data is shown in Figure 6.15. The parameters for the experiment were: SVD window length of 50, embedding dimension of 3, 2000 iterations of 5 evolve steps each, 11000 samples, 20 vectors in the neighbourhood matrix. With the exception of speaker CA, vowel /i/, the averaged largest Lyapunov exponent is greater for the synthesised data compared to the original signal. Additionally, excepting CA, /i/ and/u/, the synthesised data has $\lambda_1 \geq 0$ where as the original data has $\lambda_1 \leq 0$. The results from this algorithm must be treated with caution, as has been mentioned in Chapter 3. However the com-

parison between original and resynthesised signal results clearly shows that the RBF network causes a significant increase in nonlinearity.

6.6 Conclusion

In this chapter a neural network model for speech synthesis using radial basis functions has been introduced. Having considered the basic theory of RBFs, as well as other neural network approaches to speech synthesis, the main focus has been on attempts to resynthesise speech in a stable manner.

A thorough analysis of the various parameters used in the RBF network has been carried out, from which an appropriate set of values was deduced. The network was then trained on the database of twelve signals using these parameters and the results of free–running synthesis observed. It was found that a number of the signals were not correctly synthesised. A number of different types of erroneous output were observed, the most common being a correct waveform with numerous large spikes present on it. While the other problem signals are most likely a failure of the network to approximate the dynamics of the underlying speech system, this type of problem would appear to be a form of instability.

Numerical instability in either the inversion of the covariance matrix by SVD, or the size of the weights, was considered. Although the weight values are large, and the covariance matrix is somewhat ill–conditioned, no firm evidence was found to link either of these to incorrect resynthesis.

The stability of the mapping function was then considered. Equations for the calculation of the local Lyapunov exponents directly from the network equations were formulated, and these showed greater sensitivity in the networks which produced unstable results. Subsequent calculation of the global Lyapunov exponents showed a positive largest Lyapunov exponent for the resynthesised signals, compared to a negative largest exponent in all but one case for the original signals. It is therefore concluded that the mapping function $\hat{\mathcal{F}}$ is more nonlinear than the original speech generating system (and may even be chaotic), and as such will be sensitive to small changes, such as those generated by the feedback loop.

Chapter 7
Regularisation and pitch modification

7.1 Introduction

In Chapter 6 an RBF model was presented that, from a theoretical perspective, appears useful for vowel synthesis. However it was demonstrated to give poor synthesis performance for the majority of speech signals applied to it. In this chapter those problems are solved by applying a technique known as regularisation during the learning of the weights. This is shown to allow the stable resynthesis of all twelve vowels from the Edinburgh constant–pitch database, and additionally to model another complex system, the Lorenz system of equations.

Results are presented for the RBF synthesised speech, with comparisons made to the original speech signal and to the synthetic signal generated by a linear prediction synthesiser. These include jitter and shimmer measurements which demonstrate that the RBF network is capable of modelling the natural variability found in speech.

In the final part of the chapter, the pitch modification issue is examined, and a possible technique based on the interpolation of the weights vector is proposed. A special database was collected for this purpose, and the analysis of this for pitch modification is described.

7.2 Application of regularisation

The results from Chapter 6 imply that a large number of the mapping functions learnt by the network suffer from some form of instability. This may be due to a lack of smoothness in the function, in which case regularisation theory is the ideal solution. Regularisation theory was first applied to RBFs by Poggio and Girosi [127] and essentially means applying some smoothing factor to the approximated function $\hat{\mathcal{F}}$. More concretely, regularisation can be seen to change the ill–posed problem of learning a smooth mapping function from data examples into a well–posed one. Approximating the mapping function is ill–posed because the data does not contain the information required to uniquely reconstruct the mapping in regions where the

data is unavailable [127]. Regularisation theory applies some prior knowledge, or constraints, to the mapping function to make a well–posed problem [128]. Mathematically, a new cost function can be defined as

$$E(\hat{\mathcal{F}}) = E_s(\hat{\mathcal{F}}) + \lambda E_r(\hat{\mathcal{F}}) = \frac{1}{2}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2 + \frac{1}{2}\lambda\|\mathbf{P}\hat{\mathcal{F}}\|^2 \tag{7.1}$$

where $E_s(\hat{\mathcal{F}})$ is the sum of squares error as previously defined in Equation 6.6, $E_r(\hat{\mathcal{F}})$ is the regularising term and λ is the real, positive regularisation parameter. The regularising term contains the differential operator $\mathbf{P}$, which contains prior information about the form of the solution and whose purpose is to stabilise the map $\hat{\mathcal{F}}$ by making it smooth and continuous [13]. $\|.\|$ is the L^2 norm on the function space $\mathbf{P}\hat{\mathcal{F}}$. Minimising $E(\hat{\mathcal{F}})$ leads to the associated Euler–Lagrange equation:

$$\mathbf{P}^*\mathbf{P}\hat{\mathcal{F}}(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}\left(x_i - \hat{\mathcal{F}}(\mathbf{x})\right)\delta(\mathbf{x} - \mathbf{x}_i) \tag{7.2}$$

where $\mathbf{P}^*$ is the adjoint of $\mathbf{P}$. Using the Green's function, $G(\mathbf{x};\mathbf{x}_i)$, which satisfies

$$\mathbf{P}^*\mathbf{P}G(\mathbf{x};\mathbf{x}_i) = \delta(\mathbf{x} - \mathbf{x}_i) \tag{7.3}$$

then Equation 7.2 can be written as

$$\hat{\mathcal{F}} = \frac{1}{\lambda}\sum_{i=1}^{N}\left(x_i - \hat{\mathcal{F}}(\mathbf{x})\right)G(\mathbf{x};\mathbf{x}_i) \tag{7.4}$$

Here the points $\mathbf{x}_i$ are the centres of the Green's functions. Setting

$$w_i = \frac{1}{\lambda}(x_i - \hat{\mathcal{F}}(\mathbf{x})) \qquad i = 1, 2, \ldots, N \tag{7.5}$$

allows Equation 7.4 to be resolved as

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{w} = \mathbf{x} \tag{7.6}$$

where $\mathbf{I}$ is the $N\mathrm{x}N$ identity matrix, $\mathbf{w}$ is the length N weights vector, and $\mathbf{G}$ is the $N\mathrm{x}N$ Green's matrix. If $\mathbf{P}$ is chosen appropriately, and is invariant under both rotations and transla-

tions, then the Green's function $G(\mathbf{x};\mathbf{x}_i)$ becomes [13]:

$$G(\mathbf{x};\mathbf{x}_i) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right) \tag{7.7}$$

which can be recognised as being a Gaussian radial basis function. Thus the regularised solution becomes

$$\hat{\mathcal{F}}(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right) \tag{7.8}$$

with the centres $\mathbf{x}_i$ located at the data points. This solution is only of limited interest, since it is necessary to use all of the training data as centres. However, it can be extended to the approximate regularised solution, where there are fewer centres than training data and the centres do not have to be a subset of the training data, by redefining the mapping function as

$$\mathcal{F}' = \sum_{i=1}^{P} w_i G(\|\mathbf{x}-\mathbf{c}_i\|) \tag{7.9}$$

where $\mathbf{c}_i$ are the centres of the Green's functions. This leads to a new cost function

$$E(\mathcal{F}') = \sum_{i=1}^{N}\left(x_i - \sum_{j=1}^{P} w_j G(\|\mathbf{x}_i-\mathbf{c}_j\|)\right)^2 + \lambda\|\mathbf{P}\mathcal{F}'\|^2 \tag{7.10}$$

Minimising this gives the result [13]

$$(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0)\mathbf{w} = \mathbf{G}^T\mathbf{x} \tag{7.11}$$

where $\mathbf{G}$ is the $N\text{x}P$ Green's matrix, equivalent to the $N\text{x}P$ radial basis kernel output matrix Φ, and $\mathbf{G}_0$ is a symmetric $P\text{x}P$ matrix composed of elements $G_0(\mathbf{c}_i;\mathbf{c}_j)$:

$$G_0(\mathbf{c}_i;\mathbf{c}_j) = \exp\left(\frac{-\|\mathbf{c}_i-\mathbf{c}_j\|^2}{2\sigma^2}\right) \tag{7.12}$$

Equation 7.11 is then solved by SVD to find the least–squares solution for the weights vector $\mathbf{w}$. When $\lambda = 0$ there is no regularisation, and as λ is increased so greater smoothness is imposed onto $\mathcal{F}'$.

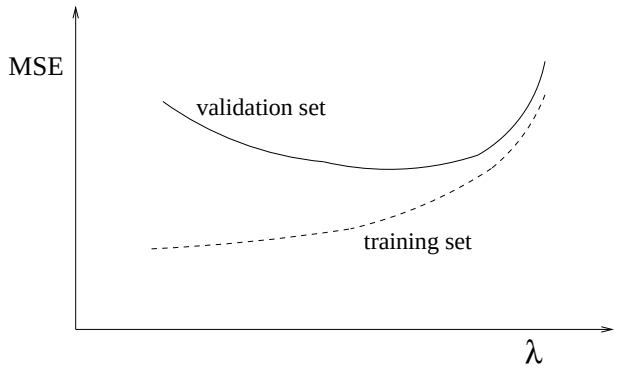


Figure 7.1: *Principle of cross–validation to find λ. While decreasing λ leads to ever closer approximations to the training data, the minimum in the validation error curve indicates the best value for generalisation.*

7.2.1 Cross–validation

Choosing an appropriate value for λ is generally solved by cross-validation [113]. After choosing all the other network parameters, these are held constant and λ is varied. For each value of λ, the MSE on an unseen validation set is calculated. The MSE curve should have a minimum indicating the best value of λ for generalisation, as shown in Figure 7.1. Again it is noted that this MSE measurement may not be the optimum way to choose parameters for successful resynthesis.

7.2.2 Learning complex systems

The regularised RBF technique has recently been applied by Haykin and Principe to modelling the invariant properties of chaotic systems, such as the Lorenz equations [128]. Their network has an input delay made up of unit delays, of length $d_E\,\tau$, where d_E is the dimension and τ is the embedding delay. When modelling the Lorenz system, d_E is chosen as 5 and τ as 4. This leads to an effective embedding dimension of 20. The network, using 400 centres chosen as a subset of the training data, is then trained with the Lorenz data which has noise added to give a signal–to–noise ratio (SNR) of 25 dB. By this method the Lorenz system is successfully modelled, with the Lyapunov exponents found as 2.566, -0.628 and -15.03 nats/sec, compared to 1.570, -0.031 and -22.31 nats/sec in the original signal (before noise was added).

A similar experiment was performed with the RBF network as described in Chapter 6, but with the weights learnt using the regularisation technique. The Lorenz equations (see Section 3.2.2)

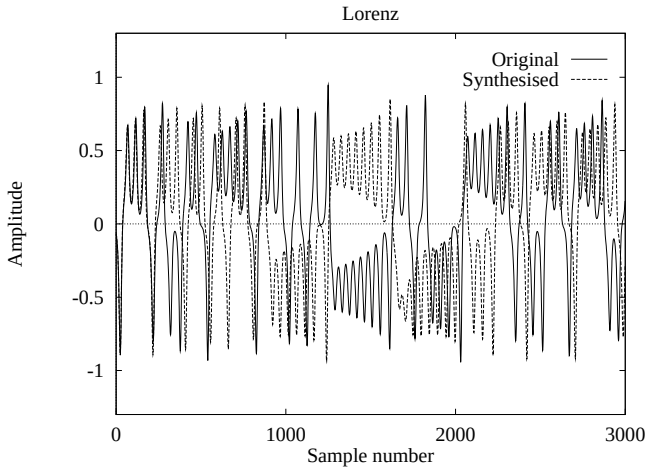


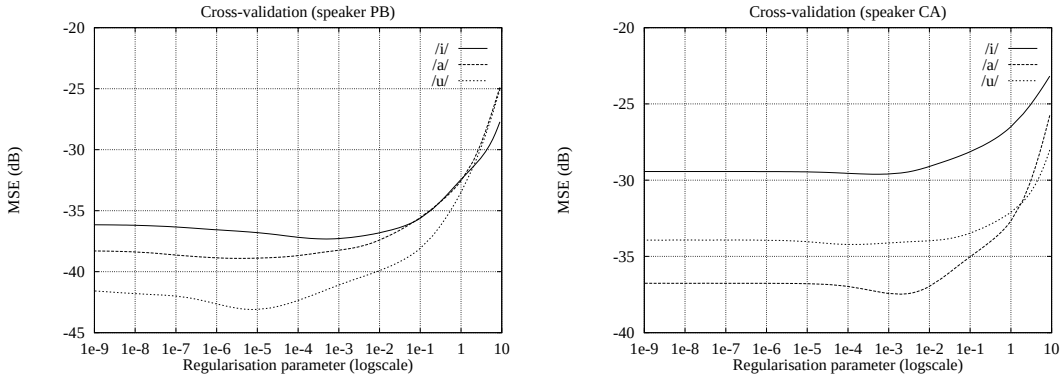
Figure 7.2: *Original and resynthesised Lorenz signals, demonstrating that the RBF network is capable of learning complex dynamical systems.*

were iterated by the 4th order Runge–Kutta method, and the time series from the variable x was recorded as training data. A step size of 0.01 was used, with the Lorenz parameters set at $\sigma = 16$, $r = 40$ and $b = 4$. It was found that a dimension of 3, coupled with the use of 125 centres and 1000 samples of training data, was sufficient to learn the dynamics of the system. Both the data subset and the hyper–lattice method gave good results, with $\tau = 11$ samples. It was not found to be necessary to add noise to the training data.

Figure 7.2 shows the original Lorenz signal, and the signal synthesised by the RBF network using the data subset method. The synthesised signal follows the original closely for an initial period, then diverges. However the form of the signal is clearly that of the Lorenz system (verified by examining the state space reconstruction), showing that the dynamics have been correctly learnt. This is confirmed by examining the Lyapunov exponents of the signals. Using the algorithm of Banbrook *et al.*, the Lyapunov exponents were calculated to be 1.161, -0.259 and -24.55 nats/sec for the original, 1.524, -0.523 and -17.52 nats/sec for the data subset, and 1.787, -0.523 and -18.05 nats/sec for the hyper–lattice. This demonstrates that the RBF network is capable of learning complex and, in this case, even chaotic systems.

7.2.3 Stable vowel synthesis

The issue of vowel sound synthesis can now be reconsidered. Some of the previous synthesis results, without regularisation, bore no resemblance at all to the original speech signal. It is



(a) **(b)**

Figure 7.3: *Cross–validation for the vowels /i/, /a/, /u/ for* (a) *speaker PB using hyper–lattice and* (b) *speaker CA using data subset method.*

therefore unlikely that simply applying regularisation will solve this problem. Considering that Haykin [128] uses a very high embedding dimension to achieve good results, it appears worthwhile increasing the embedding dimension in the network to be at least in line with Takens' theorem. Hence the embedding dimension was increased from 3 to 7. Re–running the experiments for the bandwidth, number of centres and training length parameters shows only minor differences to the 3D case in terms of MSE results. Taking these into account, the network parameters for resynthesis were adjusted to:

Bandwidth = 0.8 for hyper–lattice, 0.5 for data subset; Dimension = 7; Number of centres = 128 (*i.e.* 2^7); Hyper–lattice size = 1.0; Training length = 1000.

Selecting the regularisation parameter

The next step is to select an appropriate value for λ. Using the cross–validation technique, λ was varied between 9 and $1\text{x}10^{-9}$, and the MSE on an unseen validation set for each signal in the database was recorded. This gave curves of the expected form, as shown in Figure 7.3. This shows examples of the cross–validation for speaker PB using the hyper–lattice, and speaker CA using the data subset method. All of the curves show a minimum for λ, which was the value selected for use in the resynthesis. With the regularisation parameter chosen by this method, the 7D resynthesis gave correct results for all of the signals except KH /i/ and KH /u/ when using the data subset method of centre selection. However, only two signals (CA /i/ and MC

/i/) were correctly resynthesised by the hyper–lattice method.

It was found that λ needed to be increased significantly to ensure correct resynthesis for all the signals when the hyper–lattice was used. The actual value required was determined experimentally, but once found all of the vowels in the database were correctly synthesised for both centre selection methods.

Trade–offs

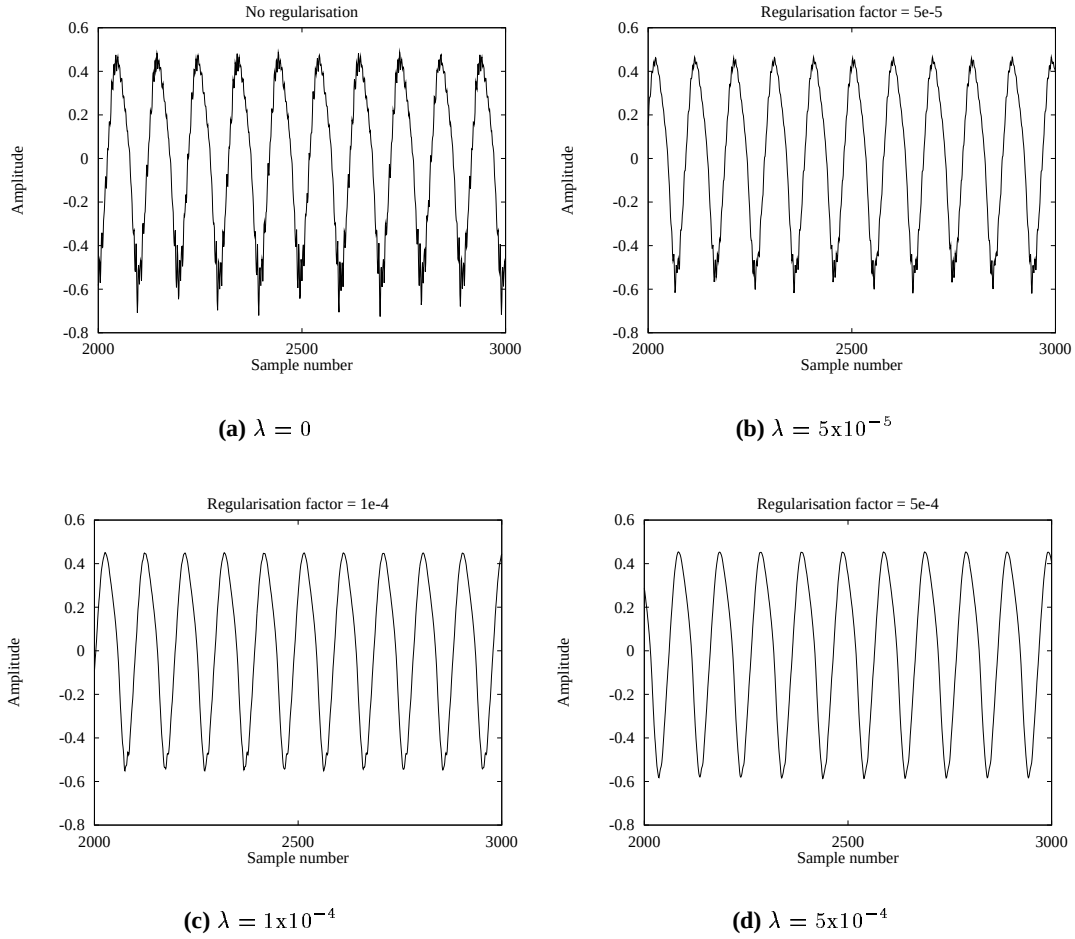
Achieving stable resynthesis inevitably comes at some cost. By forcing smoothness onto the approximated function there is the risk that some of the finer detail of the state space reconstruction will be lost. Examining the signal CA /i/, which was one of only three signals to be successfully resynthesised at an embedding dimension of seven without regularisation, it is possible to observe the effects of increasing the size of the regularisation parameter. Figure 7.4 shows the changes in the time domain signal as λ is increased from zero to $5\text{x}10^{-4}$. The fine details of the original signal (equating to high frequency information) are gradually smoothed away as the regularisation parameter is increased. Therefore, for best results, λ should be set at the smallest possible value that allows stable resynthesis.

7.2.4 Normalised RBFs

At this point it is appropriate to mention normalised RBFs. In this type of RBF network, the weights are trained without any regularisation, and the kernel function is redefined as a *softmax* function where the sum of all the kernel outputs is one [113]:

$$\phi_{ij}(\|\mathbf{x}_i - \mathbf{c}_j\|) = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{c}_j\|^2 \,/\, 2\sigma\right)}{\sum_{k=1}^{P} \exp\left(-\|\mathbf{x}_i - \mathbf{c}_k\|^2 \,/\, 2\sigma\right)} \tag{7.13}$$

This has been shown to lead to improved approximation capabilities [116], and can in itself be seen as a form of regularisation since the normalised kernels provide a smooth output over the whole hidden space (similar to an MLP sigmoid activation function). However it was found that, although comparable resynthesis results to the regularised network were obtained using the data subset centre selection method, this method did not allow any correct resynthesis when the hyper–lattice method was used. For this reason, normalised RBFs are not discussed further.



(a) $\lambda = 0$

(b) $\lambda = 5\text{x}10^{-5}$

(c) $\lambda = 1\text{x}10^{-4}$

(d) $\lambda = 5\text{x}10^{-4}$

Figure 7.4: *Effect of changes in the regularisation parameter on the time domain signal of the vowel /i/, speaker CA.*

7.3 Analysis of synthesised vowel sounds

The performance of the regularised RBF network as a nonlinear speech synthesiser is now measured, in a similar way to the examination of the locally linear synthesiser in Chapter 5. In addition to comparing the output of the nonlinear synthesiser to the original speech signal, the synthetic speech from a traditional linear prediction synthesiser is also considered. In this case, the LP filter coefficients were found from the original vowel sound (analogous to the training stage of the RBF network). The estimate $(F_s + 4)$ [96] was used to set the number of filter taps to 26. Then, using the source–filter model, the LP filter was excited by a Dirac pulse train to produce the desired length LP synthesised signal. The distance between Dirac pulses was set to be equal to the average pitch period of the original signal. In this way, the three vowel sounds for each of the four speakers in the database were synthesised.

7.3.1 Temporal and spectral characteristics

Figure 7.5 shows the time domain waveforms for the original signal, the LP synthesised signal and the two RBF synthesised signals, for the vowel /u/, speaker MC. Figure 7.6 shows the corresponding frequency domain plots of the signals, and the spectrograms are shown in Figure 7.7. In these examples, the regularisation parameter λ was set at 0.01 for the hyper–lattice, and 0.005 for the data subset.

These figures clearly demonstrate the different modelling strategies between the LP and RBF approaches. In the linear prediction case, the technique attempts to model the spectral features of the original, which is done by assuming that the excitation signal can be separated out. Hence the reasonable match seen in the spectrum (although the high frequencies have been over–emphasised), but the lack of resemblance in the time domain. The RBF techniques, on the other hand, attempt to model the complete underlying dynamical system. Thus the RBF synthesised signals resemble the original in the time domain, since it is from this that the state space reconstruction is formed. Both RBF networks successfully capture many of the original signal characteristics, although the spectral plots show that the higher frequencies have not been well modelled by this method. This is because the networks have missed some of the very fine variations of the original time domain waveform, which may be due to the imposition of the regularisation parameter λ.

Further spectrogram examples for different vowels and speakers are presented in Appendix B.

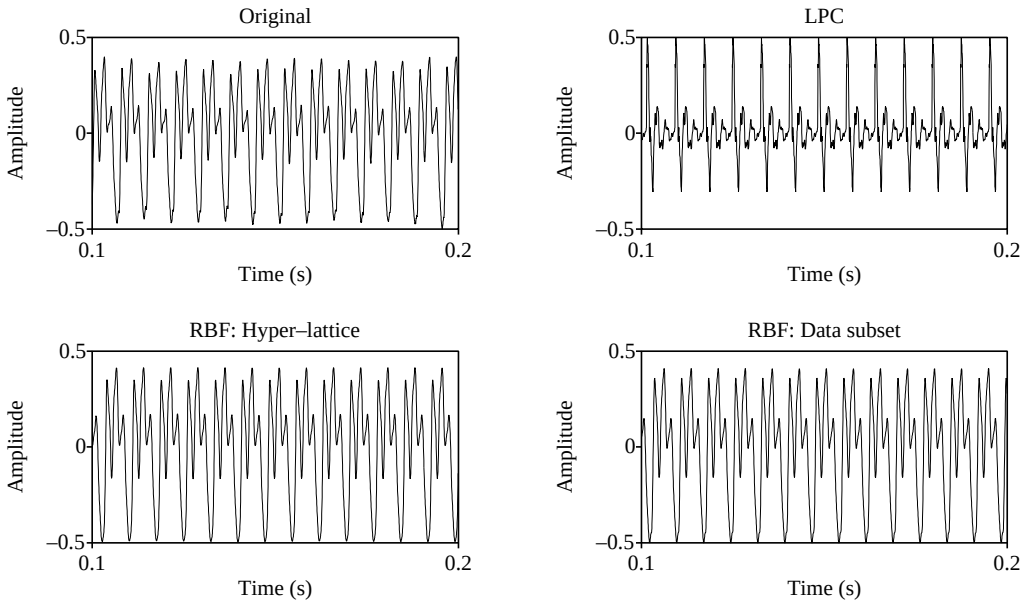


Figure 7.5: *Time domain examples of the vowel /u/, speaker MC. Top row: original signal (left) and linear prediction synthesised signal (right); Bottom row: RBF network synthesised signal, hyper–lattice (left) and data subset (right).*

The size of λ is seen to influence the quality of the signal at high frequencies. Limited informal listening tests indicated that listeners generally preferred the RBF synthesised waveforms against the LP synthesis results. Although the poor high frequency modelling is very evident, listeners found that a synthetic signal with some natural variability was preferable to the LP signal, which has no variability.[1]

Comparison of centre positioning strategies

The choice of either data subset or hyper–lattice centre strategies does not greatly effect the quality of the synthesised speech in terms of time domain and spectral characteristics. In general, the method which requires the smallest amount of regularisation will give superior results since less detail is smoothed out, but this is very signal dependent. The average performance over the database between the two techniques is approximately equivalent.

[1] Example audio files are provided on CD, as described in Appendix D.

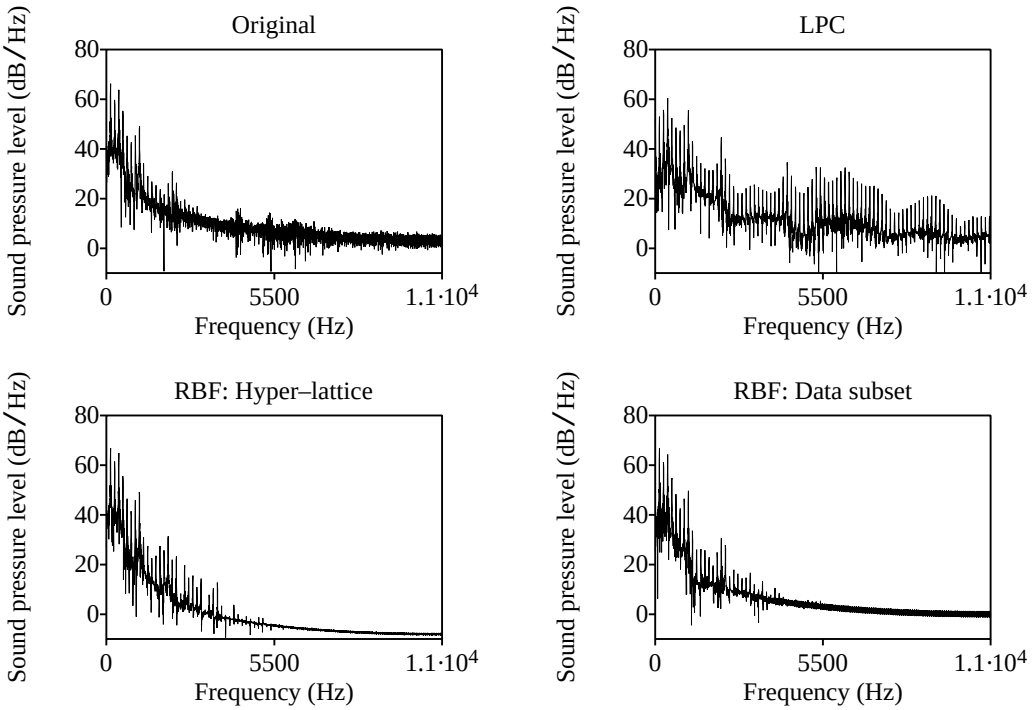


Figure 7.6: *Spectrums for examples of the vowel /u/, corresponding to the signals in Figure 7.5.*

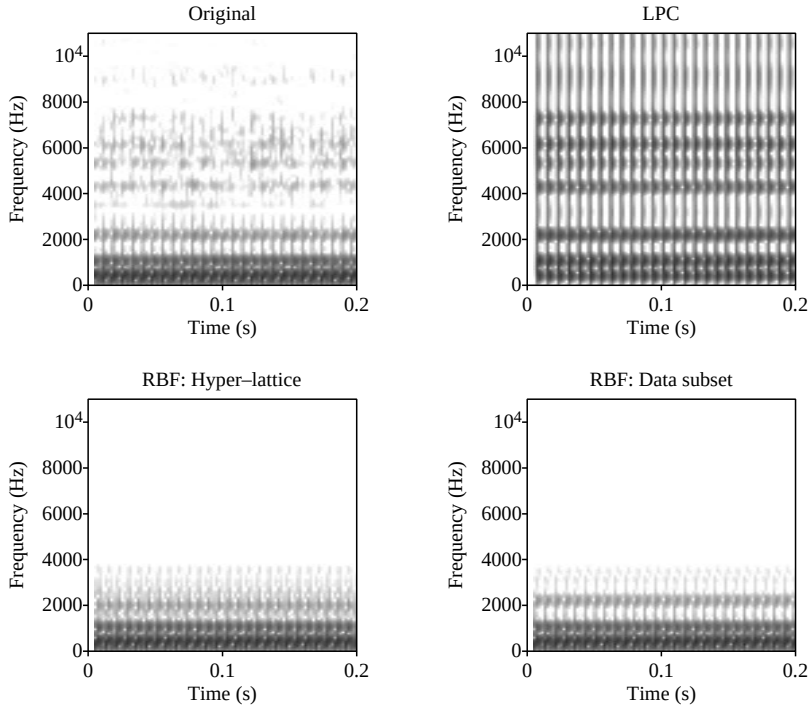


Figure 7.7: *Wide–band spectrograms for examples of the vowel /u/, corresponding to the signals in Figure 7.5.*

Data type	MC (male)	PB (male)	CA (female)	KH (female)	Average
Hyper–lattice jitter (%)	0.470	0.459	1.14	0.717	0.697
Data subset jitter (%)	0.482	0.394	0.663	0.546	0.521
Original jitter (%)	0.690	0.686	0.685	0.906	0.742
Hyper–lattice shimmer (%)	1.00	0.352	1.33	1.00	0.922
Data subset shimmer (%)	0.694	0.344	7.65	0.699	2.34
Original shimmer (%)	4.21	4.60	7.06	4.81	5.17

Table 7.1: *Percentage jitter and shimmer in original and synthesised waveforms (hyper–lattice and data subset), averaged over the vowels /i/, /ɑ/ and /u/ for each speaker, and as an average over the database.*

7.3.2 Jitter and shimmer

Jitter and shimmer measurements were made on all of the RBF synthesised waveforms in the same manner as in Chapter 5, using epoch detection over a 500 msec window of data. For jitter measurements in sustained vowels, an expected value is a 0.1 to 1% variation around the average fundamental frequency. Table 7.1 shows the results of the average pitch length variation, expressed as a percentage of the average pitch period length. Results for both the hyper–lattice and data subset techniques are presented, along side the jitter measurements of the original speech data. The hyper–lattice synthesised waveforms contain more jitter than the data subset signals, and both values are reasonable compared to the original. Additionally, the cycle–to–cycle variation time series do not have a periodic structure, which is the opposite to the results found from the locally linear synthesis model. Instead the time series follow a seemingly random pattern in a similar way to the original, as shown by the example vowel /i/ in Figure 7.8.[2]

Shimmer results (the variations in energy each pitch cycle) for the original and synthesised waveforms are also displayed in Table 7.1. It can be seen that in general there is considerably less shimmer on the synthesised waveforms as compared to the original, which will detract from the quality of the synthetic speech. However there is again no evidence of periodicity in the power–per–cycle variations when they are plotted as a time series, as seen in Figure 7.9 which is an example for the vowel /ɑ/. This is an improvement over the locally linear synthesiser, which had periodic structures present in all of the synthesised shimmer time series.

[2] Recent research suggests that these cycle–to–cycle variations may actually be fractal in nature [92].

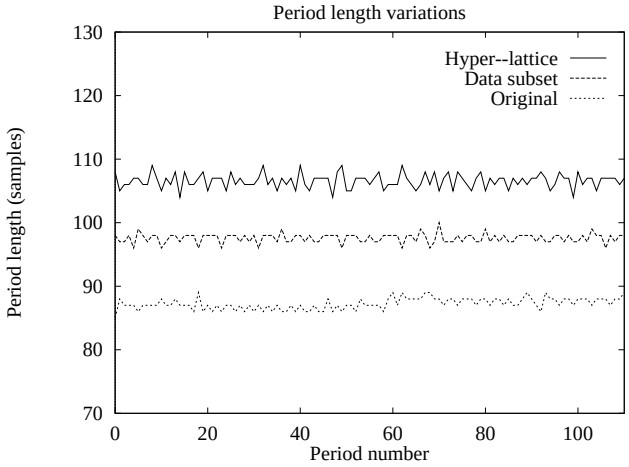


Figure 7.8: *Comparison of period length variations between the original data and the synthesis results (hyper–lattice and data subset) for the vowel /i/, speaker KH. The hyper–lattice results have been shifted upwards by* 20 *samples, and the data subset results by* 10 *samples to allow for easier comparison.*

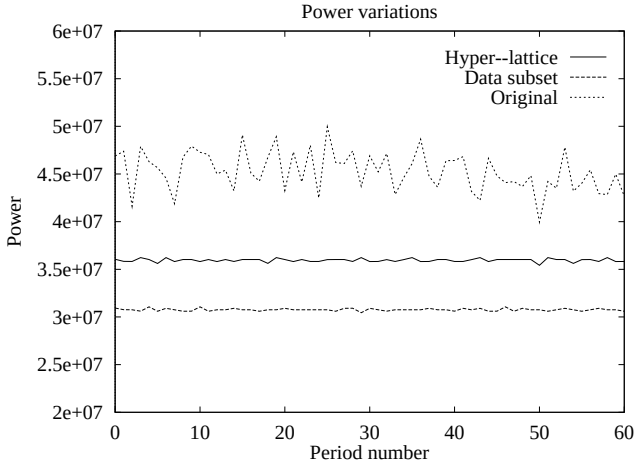


Figure 7.9: *Comparison of raw power per pitch cycle variations between the original data and the synthesis results (hyper–lattice and data subset) for the vowel /*a*/, speaker PB.*

7.3.3 Lyapunov exponents

The previous results suggest that the RBF network has successfully captured the dynamics of the original speech signal. Although the higher frequencies are not well represented, the low frequencies and time domain signal are well modelled in the synthetic waveforms. Additionally, the natural variability of the original signals is preserved. As another measure of how well the dynamics have been learnt, the dynamical invariants of the system are examined using the global Lyapunov exponents.

Figure 7.10 shows a comparison of Lyapunov exponents between the original signal and the average RBF network performance (*i.e.* averaged hyper–lattice and data subset exponents). These results show that the signals produced by the network do not have exactly the same dynamical properties as the original speech signal. However, they are very similar in having an almost–zero, negative, negative spectrum, suggesting that many of the original dynamics are preserved. Also the values are closer to those of the original exponents, compared to the unregularised results presented in Chapter 6. Therefore the regularised solution can be seen to provide a more robust and closer approximation to the original system than an unregularised network (although the improved approximation will also be due to the increased number of centres when moving from 3D to 7D).

7.4 Pitch Modification

Now that a synthesis model capable of producing natural–sounding speech in a stable manner has been implemented, the next step is to consider pitch modification. Ideally, it would be possible to be able to specify an exact pitch value at which the network will synthesise a particular vowel, even if the training data used to teach the network was at a different pitch. Unfortunately, in common with the locally linear synthesiser, pitch is not separated out by the RBF technique. This is the paradox of the nonlinear techniques presented in this thesis: by not making the assumptions used in the linear source–filter model, the implicit ability to control pitch is lost. However, pitch information is never–the–less present in the network, and therefore some suitable technique must be employed to exploit this information.

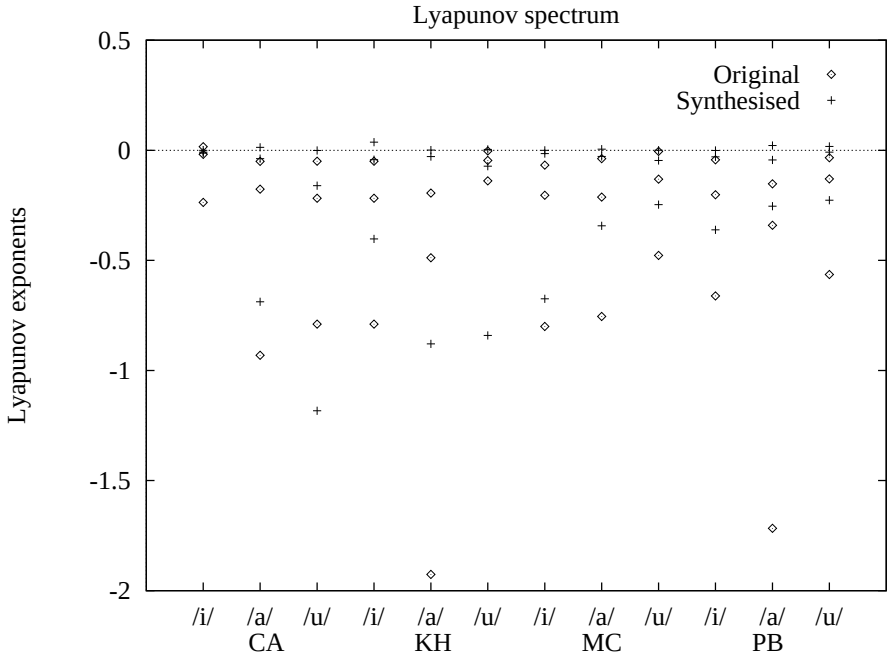


Figure 7.10: *Comparison of Lyapunov spectra across each vowel of the database. The synthesised results are the average of both hyper–lattice and data subset methods. All Lyapunov exponents are expressed in units of bits/sample.*

7.4.1 Principle

The method considered here is based on interpolation in the weight vector space. The basic principle is as follows:

1. Extract examples of the same vowel, but at different pitch values, from a speech database.
2. Model each example with the regularised RBF network, using the hyper–lattice method of centre selection.
3. The only parameters in the free–running synthesis network that are different for different input examples are the P weights, since the centres are fixed on the hyper–lattice. Therefore each vowel example is characterised by a P length vector $\mathbf{w}$.
4. Each weight vector will be a point in a P–dimensional space. An analysis must then be performed to search for the relationship that links these vectors.
5. Finally, to obtain the vowel at the desired pitch, some form of interpolation in the weight vector space would be used. This would interpolate between existing vector points in

CVC word	IPA symbol
hart	/ɑ/
heat	/i/
hood	/u/
hurt	/ɜ/
hat	/a/
head	/ɛ/
hot	/ɒ/

Table 7.2: *CVC utterances with corresponding IPA symbols as recorded for pitch analysis.*

order to estimate a new set of weights for the network. Running the network with the new weight vector would produce the vowel at the required pitch value.

The use of the hyper–lattice structure is the key to this technique, since all other centre selection strategies assign centres according to the input data. Thus the centre positions change with each input example and there is no obvious way to examine the differences between networks. The hyper–lattice structure can resynthesise data with the same quality as the data subset method, but also allows each input signal to be modelled solely by the network weights.

7.4.2 Database

To investigate the interpolation method, a special database of vowel sounds at different pitch values is required. The use of a standard continuous speech database could be envisaged, which would be searched to locate all examples of certain vowels. It is likely that in this way a selection of different pitch levels for each vowel type could be found. However searching a database in this manner is complex, and the resulting vowel examples may suffer from co–articulation effects as well as being of rather short duration. Instead a more controlled database was assembled.[3] This consists of three speakers (AG, DA, ME) pronouncing six CVC words, corresponding to six different vowels. The vowel portion of each word was extended by the speaker. Table 7.2 shows the CVC words pronounced and the corresponding IPA symbol for the extended vowel. Each word was repeated several times for each of the five pitch levels, defined as normal, highest, medium high, lowest and medium low. The sound pressure wave was recorded with a sampling rate of 22.05 kHz using 16 bit pulse code modulation (PCM). Additionally, the laryngograph signal was recorded, also at 22.05 kHz, 16 bits PCM.

[3]Thanks to Mike Edgington of BT Labs for collecting this data set.

Resynthesis of the vowels was then attempted, using the 7D regularised network. The results of this resynthesis were extremely variable, showing that the RBF network, despite the use of regularisation, is still not able to learn some signals. In particular, very little success was achieved for speakers DA and ME. Both these speakers are male with low average fundamental frequencies. This means that the state space reconstructions of these vowels, especially the low pitch level examples, will be very complex. More success was achieved with the male speaker AG, who has a higher average fundamental frequency. It was possible to correctly resynthesise examples at all pitch levels for the vowels /i/ and /u/, although problems were encountered with other vowels. The /i/ and /u/ have simpler state space reconstructions compared to the other vowels in the database, indicating why they are easier to resynthesise.

The reasons behind the poor synthesis results are not completely clear. However there are two definite contributing factors. Firstly, the state space reconstructions are more complex than those of the Edinburgh 1996 constant pitch database. This is because the speakers have low average fundamental frequencies, and a number of the examples are pronounced at "low" and "medium low" pitch levels. Secondly, the requirement for the speakers to pronounce extended vowels at pitch levels other than their normal voice pitch level appears to have resulted in nonstationary vowel waveforms in a number of cases. The speakers indicated that pronouncing sustained sounds at low and high pitch levels was difficult, and this is reflected in the poor stationarity of the speech waveforms. These two factors make for a difficult approximation task for the RBF network, and may explain at least in part the poor results seen.

7.4.3 Analysis of pitch variations

For the examples where the network correctly synthesises the vowel sound at the five different pitch levels, an analysis of the weights can be carried out. The following concentrates on the examples from speaker AG, vowel /u/. Here there are 24 examples, with the pitch varying from 102 Hz to 210 Hz.

This gives 24 points in the P–dimensional space. However, since $P = 128$, analysing this data is not easy. As a first step, the norm of each vector, $||\mathbf{w}_i||$, is calculated and plotted against pitch, as seen in Figure 7.11. This does not show any obvious correlations that might be expected, such as the magnitude of the weight vectors for similar pitch values being close in size. Next the variance of each weight within the weight vectors was calculated over the 24 examples. The plot of weight size variance against weight number is presented in Figure 7.12. It was hoped

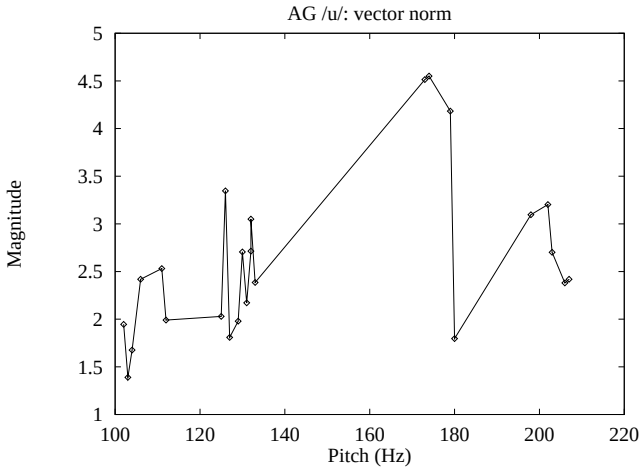


Figure 7.11: *Vector norm for each weight vector, plotted as a function of the network synthesis signal pitch. Note that the lines joining each vector norm are for clarity and do not imply a linear relationship connecting the points.*

that a small number of weights within each vector might have significantly greater variations than the remainder, indicating these parameters to be controlling factors and allowing some form of data reduction. Setting an arbitrary threshold of a variance of 0.2, the seven weight taps with variances above this were selected and used in a new vector norm against pitch plot, this time in a 7 dimensional reduced weight vector space. However this plot is not significantly different from Figure 7.11, and does not provide any further information.

Finally, even though the previous analyses indicate that the relationship, if any, between the weight vectors is nonlinear, linear interpolation between weight vectors was attempted. Such an experiment might indicate if interpolating in the P–dimensional weight space is at all useful. Selecting $\mathbf{w}_{133}$, the weights for the 133 Hz pitch network, and $\mathbf{w}_{173}$, the weights for the 173 Hz pitch network, linear interpolation was performed as:

$$\mathbf{w}_{\mu} = \mathbf{w}_{133} + \beta(\mathbf{w}_{173} - \mathbf{w}_{133}) \tag{7.14}$$

where β is a scaling factor between 0 and 1. The interpolated weights, $\mathbf{w}_{\mu}$, were then used in the RBF network for free–running synthesis. Resynthesis with the weights obtained by setting $\beta = 0.5$ resulted in a speech–like waveform with a pitch of 166 Hz. The synthesised time domain waveforms for the 133 Hz and 173 Hz examples, as well as the interpolated 166 Hz signal, are shown in Figure 7.13. A comparison of LP spectral envelopes is presented in Figure 7.14.

It can be seen that both the interpolated time domain waveform and its spectral envelope

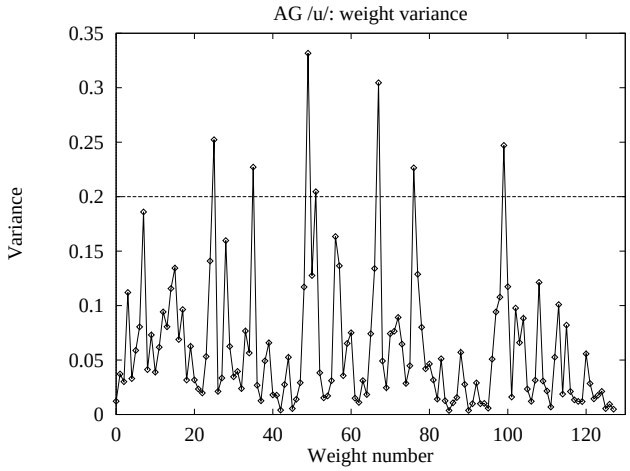


Figure 7.12: *Variance of each weight, w_i for $0 \leq i < P$, over the 24 examples plotted against weight number. Again note that the lines joining each point are for clarity only, and do not imply any linear relationship.*

are similar in form to the originals. Other values of β from 0.1 to 0.9 gave results in the range 158 Hz to 170 Hz, and all of the waveforms were found to be stable and to resemble the original speech signals in form. This demonstrates that the points along $(\mathbf{w}_{173} - \mathbf{w}_{133})$ in the weight vector space are themselves valid network weight vectors.

7.5 Conclusion

In this chapter regularisation theory has been applied to the RBF network in order to generate stable resynthesised speech. Using this technique all of the 12 example signals from the 1996 Edinburgh constant pitch database were correctly synthesised. The high frequency components were found to be poorly modelled, but the network does learn the speech dynamics sufficiently to capture the natural inter–pitch variability. Further, it was demonstrated that the regularised RBF network is capable of learning the dynamics of the chaotic Lorenz system.

Using the data–independent hyper–lattice method of centre selection, a technique for pitch modification using interpolation in the weight vector space was proposed. A special database of vowels at varying pitch levels was collected for this purpose. However it was found that a number of the signals in the database could not be correctly synthesised, despite the use of regularisation. This is thought to be due to the state space reconstruction complexity and non–stationarity of the signals. Having analysed a set of network weights from successful synthesis

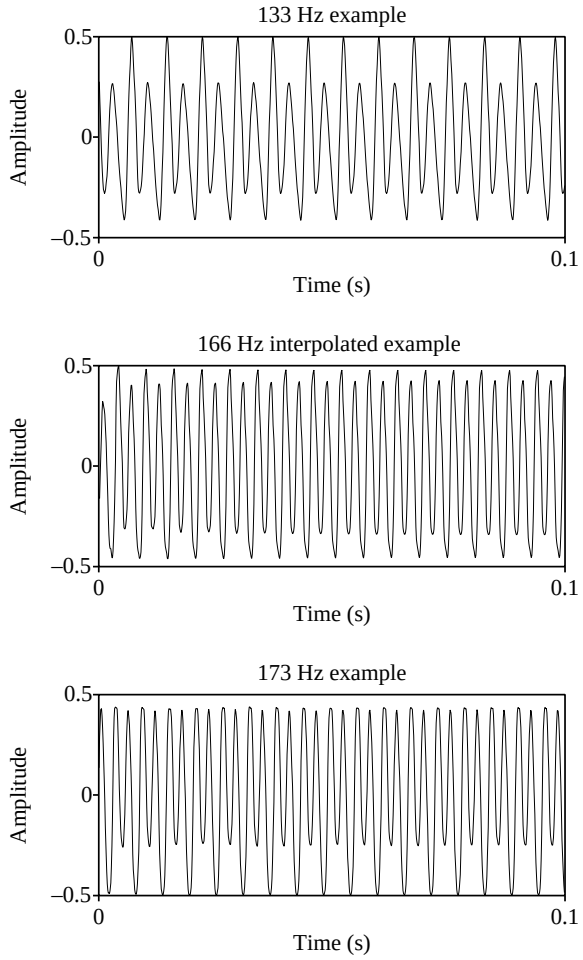


Figure 7.13: *Time domain waveforms for resynthesis. From top to bottom: resynthesised signal at 133 Hz; interpolated signal at 166 Hz; resynthesised signal at 173 Hz.*

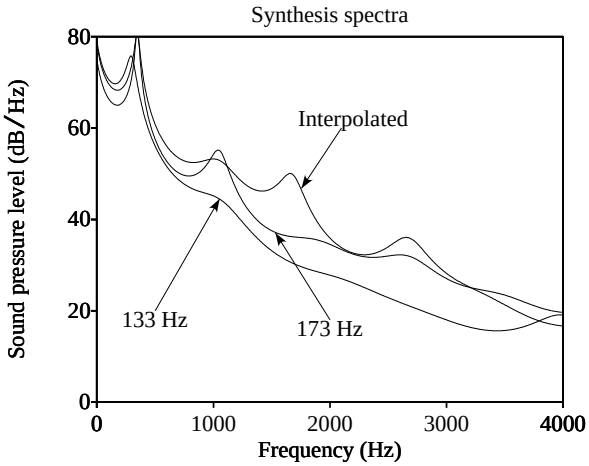


Figure 7.14: *LP spectral envelopes for the 133 Hz and 173 Hz resynthesis examples, compared to the 166 Hz interpolated example. The frequency range is limited to 4 kHz for clarity.*

networks, no obvious method to interpolate between weight vectors was found. The limited experiments with linear interpolation do however indicate this to be a promising area for further investigation.

Chapter 8
Summary and conclusions

The primary aim of the work described in this thesis is to develop nonlinear techniques for voiced speech synthesis, which allow natural–sounding synthesis coupled with the ability to modify pitch. Two nonlinear models have been investigated from this perspective and new results have been presented concerning both natural–sounding speech signal generation and nonlinear pitch modification techniques. In this chapter the main conclusions of the work are presented, and some suggestions for future work are proposed.

8.1 Achievements of the work

A large volume of work has been devoted to justifying the use of nonlinear modelling techniques for use in speech processing, but some of the results are still contradictory. In particular, the presence of chaos in vowel sounds is still a matter of debate [5, 67, 89]. The main contribution of Chapter 3 is an examination of the results from the Lyapunov exponent estimation algorithm implemented by Banbrook *et al.* [94]. It is shown that similar values for the Lyapunov exponents are obtained from another database than the one used by Banbrook *et al.*, and the effect of SVD on the results is considered. The conclusion from this analysis is that without SVD processing the exponents may be influenced by noise on the signal, but equally use of SVD may filter out parts of the signal that would otherwise give a positive exponent. However, from the point of view of constructing a nonlinear synthesis model, the fact that the voiced speech signal can be reconstructed on a low–dimensional attractor is the important factor, and the presence or otherwise of a small amount of chaos is not particularly relevant.

This low–dimensional state space reconstruction is exploited in Chapter 4 in a technique to mark epoch points in voiced speech. This work shows how dynamical theory can be applied in areas beyond analysis, in this case through the use of a Poincaré map. The novel algorithm operates by slicing through the state space reconstruction of a frame of voiced speech with a Poincaré section. All of the points in the resulting Poincaré map from the same part of the

manifold will be pitch synchronous. Hence positioning a Poincaré section at the location of one of the epoch points in the frame will yield the positions of all the others. The major drawback of the algorithm is that no method of locating the first epoch in the state space domain was found. Thus, as a stand–alone technique, the algorithm is only capable of marking pitch synchronous points. This aside, results on voiced speech show that the algorithm gives reasonable results, although further work is required to improve robustness, particularly when the state space reconstruction structure is complex.

In Chapter 5 the locally linear model for nonlinear synthesis as implemented by Banbrook [5] is considered. The vowel sounds generated by this technique are analysed in detail, since the quality of the synthetic speech is not clear from [5]. It is found that the temporal and spectral characteristics of the synthetic signals are excellent. However analysis of the inter–pitch variations shows that the algorithm modifies the jitter and shimmer by imposing a periodic structure onto the variations. Thus, although the average amount of jitter and shimmer is similar to the original, their characteristics are altered. A novel analysis of the state space structure of pitch changes in real and linear prediction synthesised vowel sounds was then carried out, in order to find a realistic method to modify pitch in the state space domain. Modelling the state space reconstructions as Šilnikov orbits, an algorithm for pitch modification by perturbing state space trajectories was proposed. This was shown to be viable when used in a simple system modelled by a set of differential equations, but was found to be incompatible with the locally linear modelling technique. Therefore, although good quality synthetic speech can be generated, the issue of realistic pitch modification is not resolved.

A parametric model is intuitively more appealing than the essentially concatenative approach used in the locally linear model. The RBF network with global feedback is such a model, and was first used in speech synthesis by Birgmeier [121]. His network, although trained in a different manner, displayed the same problems as the network described in Chapter 6. Despite an in–depth analysis of all of the network parameters, the output signal was found to be incorrect for the majority of training signals presented to the network. In an effort to discover the reason for the poor performance, the numerical stability of the learning process was examined. Even though the kernel output matrix, which must be inverted to learn the weights, is some–what ill–conditioned, and the magnitude of the weights is also large, these factors could not be linked to the incorrect resynthesis. Next, equations for the local Lyapunov exponents of the RBF network were derived, and these indicated that networks which produce unstable results show greater

sensitivity. Calculation of the global Lyapunov exponents from correctly synthesised signals showed that the network output has a larger first Lyapunov exponent compared to the original, indicating that the network increases the nonlinearity in the signal.

The main contribution of Chapter 7 is to solve the RBF stability problem by the application of regularisation. Combined with the hyper–lattice structure for centre positioning, this produces a compact model that can regenerate the dynamics of complex systems in a stable manner. As well as learning the constant pitch vowel sounds, the Lorenz system of equations is also modelled. This may be of interest to researchers in fields other than speech processing, since it indicates that signals other than speech, which can be reconstructed on low–dimensional attractors, can be learnt. Another interesting result is that the network preserves the jitter of the original speech signal. This, coupled with Lyapunov exponent results of the same order as the originals, confirms that the dynamics of the speech signals have been well modelled. However pitch modification again proves to be a sticking point. Results on the linear interpolation of the weight vectors in the weight vector space show that an interpolation technique may be appropriate, but finding the relationship which would allow the weight vector for a specific pitch to be found is not resolved.

8.2 Future work

The contributions made by this study have shown that nonlinear techniques offer the potential for high quality synthesis, including modelling of the natural variability of the speech signal. However applying these techniques is difficult, and in particular the problem of pitch modification is unresolved. Hence there are a number of areas where further investigation would be worthwhile.

Firstly, the RBF network warrants further study for both signal generation and pitch modification. To achieve a higher quality synthesised signal, better modelling of the high frequencies and shimmer is needed. This requires an improved understanding of how the network learns the dynamics of the signals, and why better modelling is achieved in certain vowels. It has been stressed throughout this thesis that a speech synthesiser must also be able to generate sounds at any required pitch, so the pitch modification issue must be addressed. The fact that sounds can be modelled by a single set of weights makes performing the pitch modification using this weight vector an attractive option, but it is not clear how to proceed beyond the linear

interpolation results already presented.

Examining other RBF structures may provide some solutions, at least in terms of improved modelling. For example a recurrent network could be implemented. This type of network has an internal memory structure and so may give improved performance, since more past information is stored [13]. Better high frequency approximation might be obtained by adopting some form of sub–band approach, perhaps with different nonlinear models for different frequency ranges [129]. Another option would be to combine a linear filter with the RBF network [130]. Given the success of linear prediction within the speech processing field, there is evidently a linear element to the speech signal and this may be better approximated by a linear filter. All of these options could be combined with a change in the choice of kernel function. Gaussian functions have been used throughout this thesis, but no study has yet been made on the effect of kernel type on resynthesis in speech. A kernel function with a less localised response could be expected to give different results.

The Poincaré pitch marking algorithm is also worthy of further investigation. If a robust technique for locating an initial epoch point can be found, then the algorithm becomes a viable epoch detector. Additionally it would be interesting to carry out a direct comparison with other epoch detection methods, to examine whether the state space representation can be used to correctly locate epochs in types of speech signals where other detectors fail.

Finally it would be interesting to turn the network to other applications. One possibility would be as part of a speech recognition system (see *e.g.* [131] for a review of speech recognition). Here the RBF structure may be able to perform the tasks of front–end feature extractor and low–level classifier simultaneously. A simple recognition system can be envisaged where each class (which could be of phones, diphones, *etc*) can be characterised by a set of weights. This would be possible through the use of the data–independent hyper–lattice structure. Then, given an input frame of speech, it would be possible to use the sum of the error residual from the RBF network over the frame to decide which class the input speech belongs to.

Appendix A
Calculation of the matrix pseudo–inverse

The pseudo–inverse of a matrix, due to Moore and Penrose [132], is useful when the exact inverse of the matrix does not exist. Given the matrix $\mathbf{A}$, the pseudo–inverse $\mathbf{A}^\dagger$ has the following properties if $\mathbf{A}$ is full rank:

$$\mathbf{A}^\dagger\mathbf{A} = \mathbf{I} \qquad\qquad \mathbf{A}\mathbf{A}^\dagger \neq \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix.

$\mathbf{A}^\dagger$ is calculated from the SVD of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T$$

and is defined as

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{W}^\dagger\mathbf{U}^T$$

where $\mathbf{W}^\dagger$ is the inverse of the diagonal matrix $\mathbf{W}$, obtained by replacing each non–zero element by its reciprocal.

Appendix B
Spectrograms

This appendix presents further spectrogram results for the extended vowel synthesis, by the locally linear synthesis (Chapter 5) and regularised RBF network (Chapter 7) techniques.

B.1 Locally linear

The spectrograms show that the locally linear technique is capable of capturing the spectral characteristics of the original across the whole range of frequencies, up to 11 kHz. An example for each speaker in the database is presented.

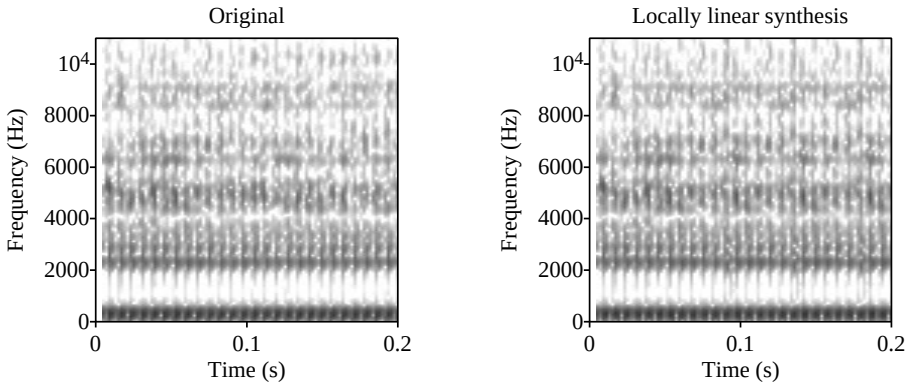


Figure B.1: *Wide–band spectrograms of original (left) and synthesised (right) speech, for the vowel /i/, male speaker MC.*

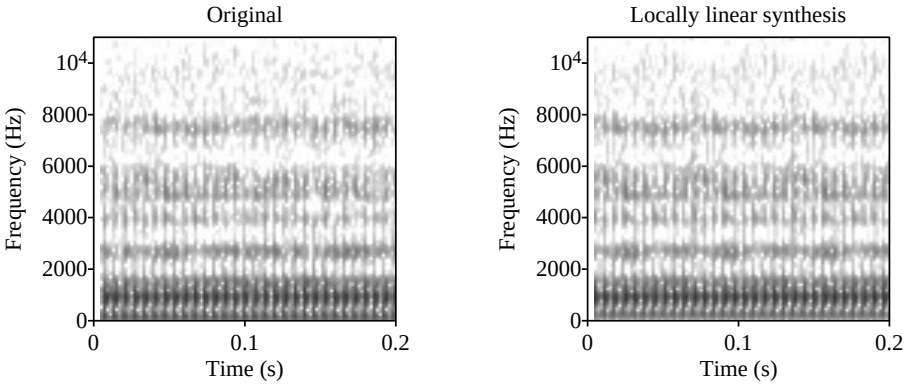


Figure B.2: *Wide–band spectrograms of original (left) and synthesised (right) speech, for the vowel /ɑ/, male speaker PB.*

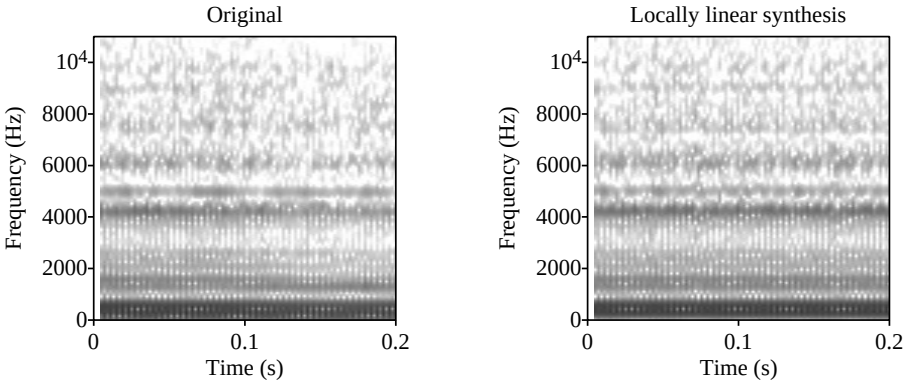


Figure B.3: *Wide–band spectrograms of original (left) and synthesised (right) speech, for the vowel /u/, female speaker KH.*

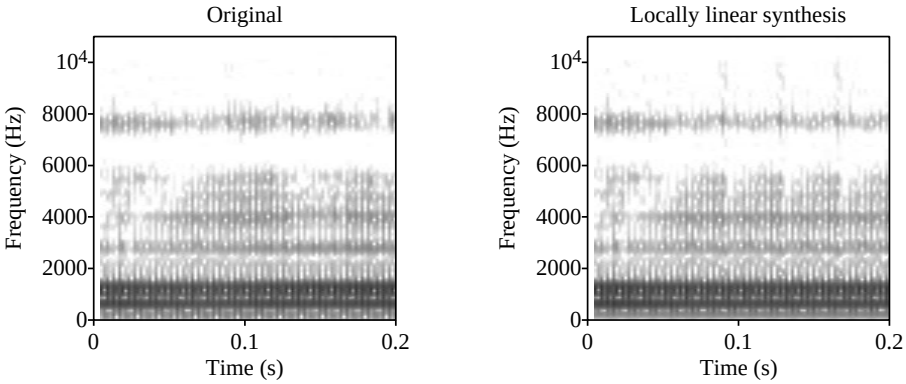


Figure B.4: *Wide–band spectrograms of original (left) and synthesised (right) speech, for the vowel /ɑ/, female speaker CA.*

B.2 Radial basis function network

These spectrograms compare the performance of the regularised RBF network against the original signal and the results from an LP synthesiser. Both the hyper–lattice and the data subset methods of centre selection are included. The value of the regularisation parameter in also given. It can be clearly seen that the RBF network does not model the higher frequencies adequately, whereas the LP model tends to over–emphasise them. However, at lower frequencies the RBF model performs well.

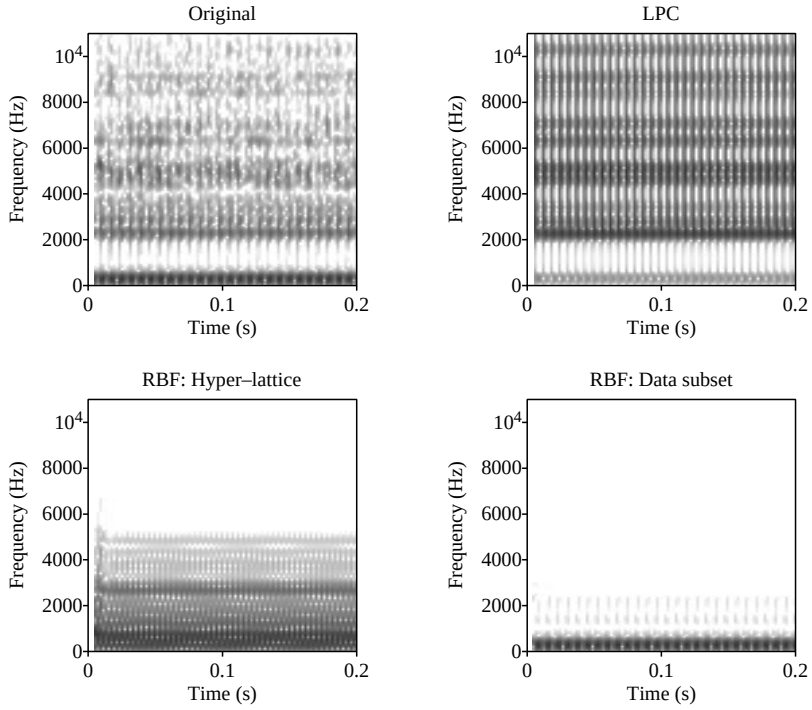


Figure B.5: *Wide–band spectrograms for the vowel /i/, male speaker MC. Top row: original signal (left) and linear prediction synthesised signal (right); Bottom row: RBF network synthesised signal, hyper–lattice,* $\lambda = 0.001$ *(left) and data subset,* $\lambda = 0.01$ *(right).*

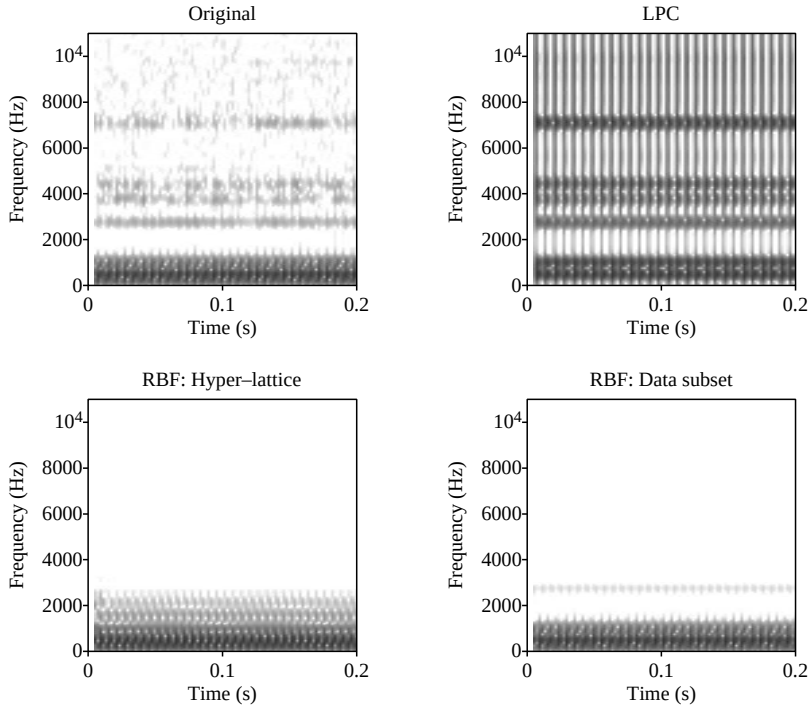


Figure B.6: *Wide–band spectrograms for the vowel /u/, male speaker PB. Top row: original signal (left) and linear prediction synthesised signal (right); Bottom row: RBF network synthesised signal, hyper–lattice,* $\lambda = 0.1$ *(left) and data subset,* $\lambda = 0.001$ *(right).*

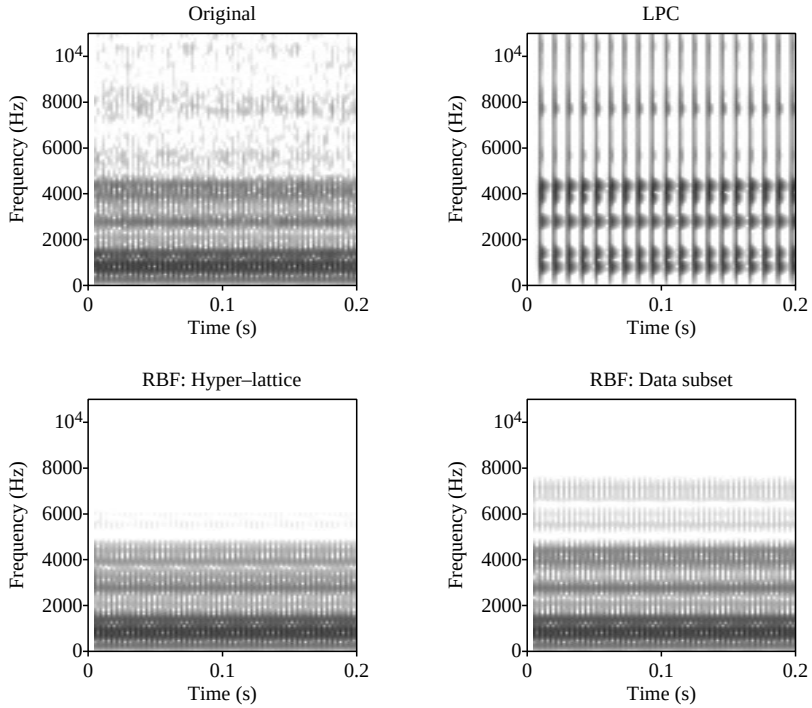


Figure B.7: *Wide–band spectrograms for the vowel /ɑ/, female speaker KH. Top row: original signal (left) and linear prediction synthesised signal (right); Bottom row: RBF network synthesised signal, hyper–lattice,* $\lambda = 0.005$ *(left) and data subset,* $\lambda = 0.01$ *(right).*

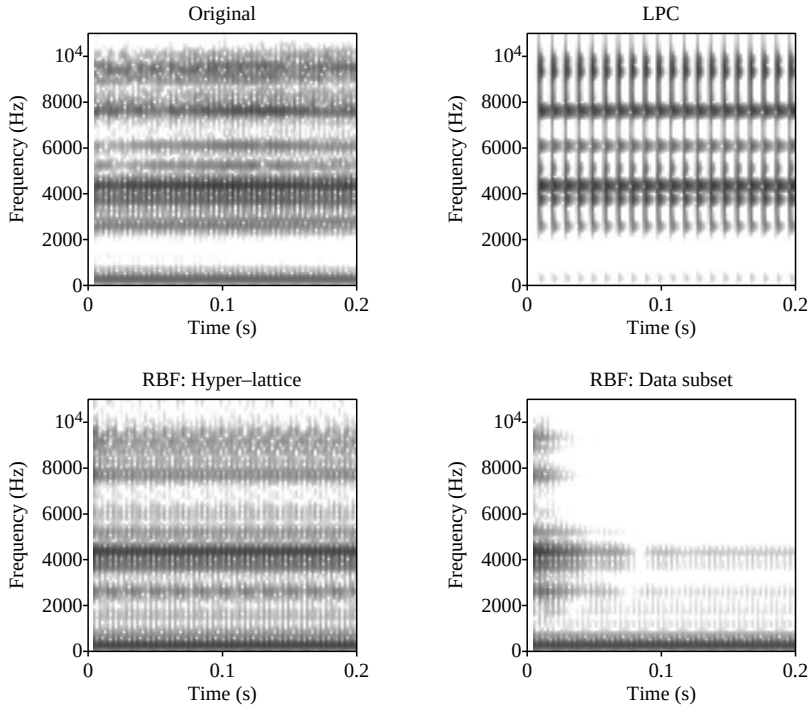


Figure B.8: *Wide–band spectrograms for the vowel /i/, female speaker CA. Top row: original signal (left) and linear prediction synthesised signal (right); Bottom row: RBF network synthesised signal, hyper–lattice,* $\lambda = 1\mathrm{e}^{-5}$ *(left) and data subset,* $\lambda = 0.001$ *(right).*

Appendix C
Publications

The following papers have been either published or submitted to conferences or journals. Those marked by † are reproduced in this appendix.

I. Mann and S. McLaughlin, "Poincaré maps and pitch detection in speech", *Proceedings of the IEE Colloquium on Signals, Systems and Chaos*, pp. 5/1 – 5/5, December 1997.

I. Mann and S. McLaughlin, "A Nonlinear Algorithm for Epoch Marking in Speech Signals using Poincaré Maps", *Proceedings of the 9th European Signal Processing Conference*, vol. 2, pp. 701 – 704, September 1998. †

M. Banbrook, S. McLaughlin and I. Mann, "Speech Characterization and Synthesis by Nonlinear Methods", *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 1, pp. 1 – 17, January 1999.

I. Mann and S. McLaughlin, "Stable Speech Synthesis using Recurrent Radial Basis Functions", *Proceedings of the 6th European Conference on Speech Communication and Technology*, vol. 5, pp. 2315 – 2318, September 1999. †

I. Mann and S. McLaughlin, "A Nonlinear Algorithm for Epoch Marking in Speech Signals", submitted to the *Japanese Journal of Signal Processing* (special issue on nonlinear processing), 1999.

I. Mann and S. McLaughlin, "Nonlinear Dynamical Modelling for Speech Synthesis, using Radial Basis Functions", submitted to *IEEE Transactions on Speech and Audio Processing*, 1999.

A NONLINEAR ALGORITHM FOR EPOCH MARKING IN SPEECH SIGNALS USING POINCARÉ MAPS

Iain Mann and Steve McLaughlin
Dept. of Electrical Engineering, University of Edinburgh,
King's Buildings, Mayfield Road,
Edinburgh. EH9 3JL. UK
Tel: +44 131 6505655; fax: +44 131 6506554
e-mail: inm@ee.ed.ac.uk, sml@ee.ed.ac.uk

ABSTRACT

A novel nonlinear epoch marking algorithm is proposed for use with voiced speech signals. Epoch detection is useful for speech coding, synthesis and recognition purposes, as it provides both the moment of glottal closure and the instantaneous pitch. Our technique functions entirely in state space, by operating on a three dimensional reconstruction of the speech signal which is formed by embedding. By using the fact that one revolution of this reconstructed attractor is equal to one pitch period, we are able to find points which are pitch synchronous by the use of a Poincaré section. Evidently the epoch pulses are pitch synchronous and therefore can be marked. Results using real speech signals are presented to illustrate the performance of the technique.

1 INTRODUCTION

In this paper we describe a novel nonlinear epoch marking algorithm for use with voiced speech. The algorithm makes use of nonlinear dynamical theory by reconstructing the system in state space and then using Poincaré sections to mark pitch synchronous points in the speech signal (*i.e.* the epochs).

Epoch detection has been a pervasive issue in speech processing for many years, since knowledge about the instantaneous pitch or moment of glottal closure is extremely important in speech coding, synthesis and recognition. A number of algorithms for epoch determination exist, most of which operate on the time domain speech signal. Various measures are employed to locate the epoch pulses, such as maximum–likelihood detection [1], discontinuities in the LPC residual [2], similarity models [3] and dynamic programming [4].

The new algorithm reported here should not be taken as a competitor to these existing techniques. Rather, it is a demonstration of the practical possibilities that nonlinear signal processing has to offer in the field of speech processing. Speech has been recognised as being a nonlinear process for a number of years [5], but this offers little advantage if we are unable to exploit this new knowledge. Researchers have already shown the possibilities available for speech coding (*e.g.* [6]); here we propose another direction to pursue. We will firstly review state space reconstruction and the theory of Poincaré sections. Next the algorithm is presented, followed by some results of epoch marking on real speech signals. We conclude with a discussion of the problems and possibilities found with this new technique.

This work was supported by an EPSRC CASE award with BT Labs, Martlesham. SML also acknowledges the support of the Royal Society.

2 STATE SPACE AND THE POINCARÉ SECTION

In nonlinear processing a d–dimensional system can be reconstructed in an m–dimensional state space from a single dimension time series by a process called embedding. Takens' theorem states that $m \geq 2d + 1$ for an adequate reconstruction [7], although in practice it is often possible to reduce m. Time delay embedding involves forming a state space trajectory matrix $\mathbf{X}$ by passing a window of length m through the time series $\mathbf{x}$:

$$\mathbf{X} = \begin{pmatrix} x_0 & x_{\tau_d} & \cdots & x_{(m-1)\tau_d} \\ x_1 & x_{(1+\tau_d)} & \cdots & x_{(1+(m-1)\tau_d)} \\ x_2 & x_{(2+\tau_d)} & \cdots & x_{(2+(m-1)\tau_d)} \\ \vdots & & & \vdots \end{pmatrix} \quad (1)$$

where τ_d is a delay time chosen so as to optimally open up the attractor. An alternative is singular value decomposition (SVD) embedding [8], which may be more attractive in real systems where noise is an issue. This technique partitions the state space into two subspaces, one containing the signal and the other the noise. The singular value decomposition of $\mathbf{X}$ is given by:

$$\mathbf{X} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}} \quad (2)$$

where $\mathbf{W}$ is diagonal, containing the singular values $w_0 > w_1 > w_2 > \ldots > w_{p-1} \geq 0$ (p being the SVD window length) and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and contain the associated singular vectors. A reduced trajectory matrix can be formed by:

$$\mathbf{X}' = \mathbf{X}\mathbf{V}_d \quad (3)$$

where $\mathbf{V}_d$ only contains the columns of $\mathbf{V}$ corresponding to the significant values of $\mathbf{W}$. This process reduces the dimension and removes noise effects [9], leaving a low dimensional, noise–free, attractor reconstruction.

It has been found that vowel sounds are low dimensional, and can be modelled in a 3 dimensional state space [10]. Figure 1 shows an example of time delay and SVD embedding of

the vowel /*i*/ in 3D state space, illustrating the smoothing capabilities of the SVD process. A time delay of 10 samples or SVD window length of 50 samples has been found to be adequate to ensure that the attractor is opened up at a sampling rate of 22kHz. This reconstruction is pitch synchronous in

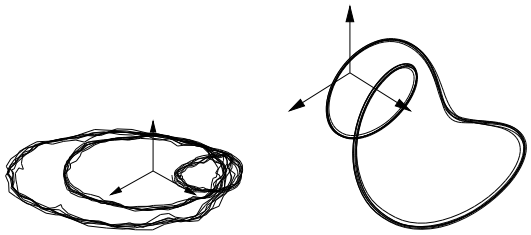


Figure 1: *(a) Time delay ($\tau_d = 10$ samples) and (b) SVD ($p = 50$ samples) embedding for the vowel /i/. $F_s = 22$kHz in the original signal.*

that one revolution of the attractor is equivalent to one pitch period. Clearly this fact can be exploited to mark points in time separated by multiples of the pitch period. The key to this in practice is the use of a Poincaré map. A Poincaré map replaces the flow of an n–th order continuous system with an $(n-1)$–th order discrete time system [11]. Thus in the case of voiced speech, the 3 dimensional attractor is replaced by a 2 dimensional cross–section. Figure 2 shows this principle, where the hyper–plane Σ cuts the flow, $\mathbf{h}$, of the attractor normally at a chosen point. Crossings in one direction only (*e.g.* from Σ^- to Σ^+) result in a one–sided map, whereas the entire set of crossings produces a two–sided map.

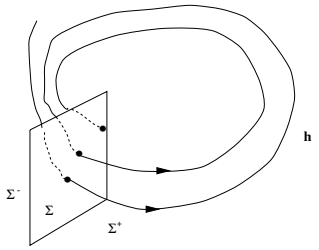


Figure 2: *An example of a (one–sided) Poincaré map, showing the hyper–plane Σ cutting the flow $\mathbf{h}$ normally.*

3 THE NEW EPOCH–MARKING ALGORITHM

Our algorithm uses the principle outlined above to mark successive epochs. To cope with the inherent non-stationarity of the speech signal, it operates on a frame–by–frame basis. Given that the input waveform is a voiced sound (no voiced/unvoiced decision module is included in the prototype), frames of length N samples are read and processed, with a 50% overlap between adjacent frames. In our experiments we have used a frame length of 35msecs ($N = 770$ at 22kHz), since speech is often assumed to be stationary for 30 to 45msec. Each frame is then embedded into 3D state space using an SVD window length of 50 samples. Assuming that the location of the very first epoch in the waveform is known, and denoting that point as x_{GCI_0}, a Poincaré section Σ is positioned normally to this point and all the points $\mathbf{x}_{CROSS}$ ($0 \leq x < N$) that traverse Σ are found. $\mathbf{x}_{CROSS}$ will contain all of the epochs, since they are pitch synchronous with x_{GCI_0}, but other arbitrary points will also be present, dependent upon the shape and complexity of the attractor.

To chose the correct points from $\mathbf{x}_{CROSS}$, we employ a distance measure, $\langle d_m \rangle$, which locates the points closest to x_{GCI_0} in state space. This distance measure tracks an intersect point x_{CROSS} as it moves around one revolution of the attractor, and finds the average distance in state space of that point from the corresponding movement of x_{GCI_0}:

$$\langle d_m \rangle = \frac{1}{R} \sum_{i=0}^{R-1} \mathrm{D}\left(x_{(GCI_0+ti)}, x_{(CROSS+ti)}\right) \qquad (4)$$

where $D(a,b)$ is the Euclidean distance between points a and b in 3D state space, and $t = (T_0 F_s / R)$ is $1/R$–th of a revolution around the attractor (in samples) at the local point, with local pitch period T_0 seconds. R is typically 8 or 10.

These points should be those within the same part of the attractor manifold as x_{GCI_0}, and hence will be the epoch points. However, due to the complexity of the attractor for certain voiced sounds, other parts of the attractor containing intersections with Σ may also pass close to x_{GCI_0}. To prevent these points accidentally being chosen we use a speech–specific windowing measure similar to that used in [12]. The average pitch period $\langle T_0 \rangle$ is initially calculated using the autocorrelation method with centre clipping over at least 30msec of the frame, and is then updated as each new epoch is marked. We then search in a window between $0.6\langle T_0 \rangle$ and $1.4\langle T_0 \rangle$ onwards from the previously marked epoch for the closest point to x_{GCI_0} in state space, since the pitch is not expected to vary by more than $\pm 40\%$ within a voiced section [13]. The new point found is marked as an epoch and the process continues through the frame. Once a frame has been processed, the last epoch marked is taken as x_{GCI_0} for the next frame, and so on through the data. The flow chart shown in Figure 3 illustrates this process.

4 RESULTS

Initially the algorithm was tested using constant pitch vowels from our own database, which also includes laryngograph data for comparative purposes. Further tests where made using a series of rising pitch vowels provided by BT Labs[1], and then signals from the Keele University Pitch Extraction Database [14], which provides speech and laryngograph data from 15 speakers reading phonetically balanced sentences. In all cases the sampling rate used was 22kHz so as to adequately fill the state space reconstruction (our database was recorded at 22kHz; the other signals were up–sampled, the BT vowels originally being at 12kHz and the Keele signals at 20kHz). All the signals have 16 bit resolution.

[1] Thanks to A. Lowry for providing this data.

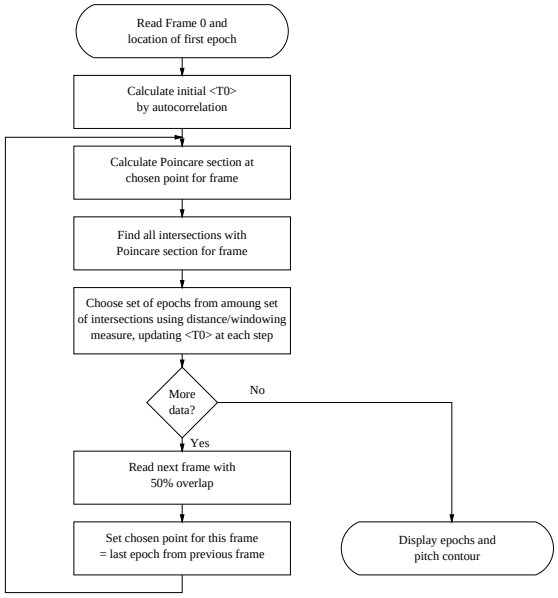


Figure 3: *Schematic of the epoch marking algorithm.*

Figure 4 shows a snapshot of the algorithm during processing on the constant pitch, stationary vowel /a/ spoken by a female speaker. The epochs are marked as calculated on the time domain waveform. The left–hand–most epoch is that which was used as x_{GCI_0}. The attractor reconstruction shows how the Poincaré section, which intersects the flow at this point, crosses through several other parts of the manifold. This is seen more clearly on the 2–sided Poincaré map. In this case the attractor structure is relatively simple resulting in four sets of intersection points, all of which are well separated making classification of the points easy. The points corresponding to the epochs are those enclosed within the circle.

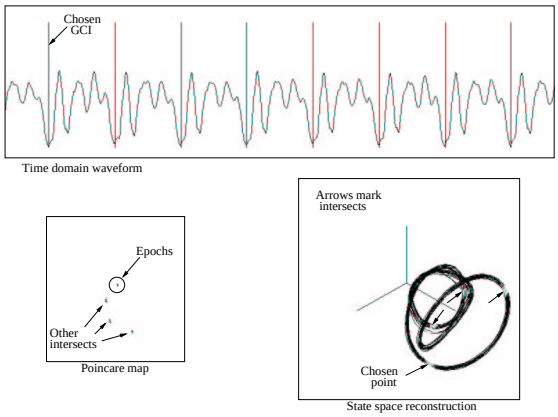


Figure 4: *Snapshot of processing of the stationary vowel /a/, showing a frame of the signal with the calculated epoch markers (top); the two–sided Poincaré map of intersections with the attractor (bottom left); the attractor reconstruction with intersects indicated (bottom right).*

The algorithm was found to correctly mark all of the stationary vowels considered.

As a next step we considered a simple non–stationary case, that of rising pitch vowels. Figure 5 shows the results of our algorithm compared to those produced by a dynamic programming–based approach [4][2] for the vowel /u/, pronounced by a male speaker. Again the left–hand–most epoch is that which was marked as x_{GCI_0}. It can be seen that the performance of our new algorithm is equal to that of this established approach, demonstrating that the frame–by–frame operation allows us to track changes in the attractor structure, caused in this case by the changes in pitch.

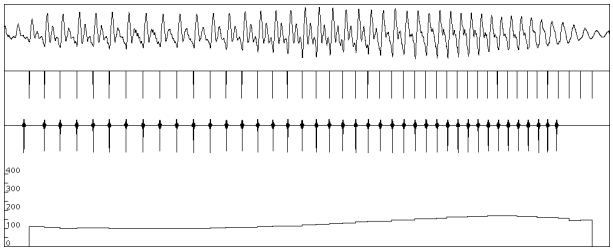


Figure 5: *Results for the rising pitch vowel /u/. From top to bottom: the signal; the epochs as calculated by our algorithm; the epochs as calculated by the dynamic programming approach; the pitch contour (Hz) resulting from our algorithm.*

Finally we tested the algorithm with various voiced segments from the Keele University database. Figure 6 shows the algorithm's performance on a section of the phrase "the northwind and the sun were disputing", spoken by a male speaker. There is a considerable change in the signal, and hence in the attractor structure, in this example, yet the epochs are still mostly well located when compared against the laryngograph signal. However problems have been encountered with other signals tested, usually caused by the state space reconstruction being very complicated, thus making the selection of the correct intersect points difficult. In general, the more complicated state space structures occur at low pitch values (many oscillations within one pitch period) and with voiced sounds where fricative noise also occurs (which tend to fill the state space).

5 DISCUSSION

Clearly a major drawback to this algorithm is the need to know the location of the first epoch. Due to the nature of the technique, it is only possible to mark points which are pitch synchronous; we cannot tell *a priori* if a data point is a glottal closure instant, but once one is known all subsequent epochs in that voiced section can, in theory, be found. Unfortunately the attractor's geometric structure does not provide any information about the epoch locations either, so at present this problem is unresolved.

[2] Available commercially as Entropic Research Laboratory's ESPS epoch function.

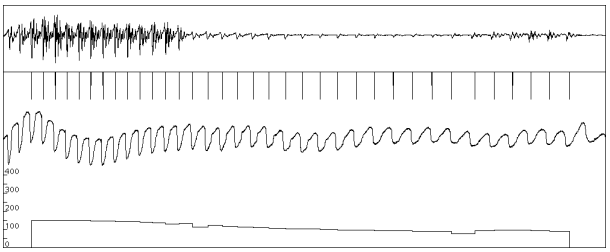


Figure 6: *Results for the voiced section of "sun were" from the Keele database. From top to bottom: the signal; the epochs as calculated by our algorithm; the laryngograph signal; the pitch contour (Hz) resulting from our algorithm.*

This aside, the technique appears useful. It works very well on the simple cases of stationary vowels and rising pitch vowels, accurately marking all the epochs. When applied to real speech signals we have met with moderate success. The algorithm is often able to track quite considerable changes in the attractor structure, caused by changes in vowel sound and/or pitch, but should an error occur (usually caused by a misalignment of the Poincaré section, often when the attractor structure is very complicated) it is unable to recover leading to incorrect marking at all points forward from the error point. This is again due to the fact that we are not actually locating the epoch pulses specifically, only points which are pitch synchronous. Therefore when an error occurs, causing a loss of synchronisation, it propagates through the remainder of the signal. Further work will need to address this problem, as well as improving the algorithm so it is better able to cope with sudden changes in attractor structure.

6 CONCLUSIONS

We have shown how nonlinear signal processing theory can be applied to the practical problem of epoch marking. The proposed algorithm, which operates in state space using a Poincaré section to mark pitch synchronous points, has been shown to perform very well on simple vowel sounds and to give promising results on real voiced speech signals. Therefore this novel technique demonstrates the potential of nonlinear theory in speech processing.

7 ACKNOWLEDGEMENTS

The authors would like to thank G. Kubin of the University of Vienna and M. Edgington of BT Labs for useful discussions about this work.

References

[1] Y. M. Cheng and D. O'Shaughnessy, "Automatic and reliable estimation of glottal closure instant and period," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 37, pp. 1805 – 1815, December 1989.

[2] R. J. DiFrancesco and E. Moulines, "Detection of glottal closure by jumps in the statistical properties of the speech signal," *Speech Communication*, vol. 9, pp. 401 – 418, December 1990.

[3] Y. Medan, E. Yair, and D. Chazan, "Super resolution pitch determination of speech signals," *IEEE Transactions on Signal Processing*, vol. 39, pp. 40 – 48, January 1991.

[4] D. Talkin, "Voicing epoch determination with dynamic programming," *Journal of the Acoustical Society of America*, vol. 85, Supplement 1, p. S149, 1989.

[5] G. Kubin, *Speech Coding and Synthesis*, ch. Nonlinear Processing of Speech, pp. 557 – 610. Elsevier, 1995.

[6] B. Townshend, "Nonlinear prediction of speech," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 425 – 428, 1991.

[7] F. Takens, "Detecting strange attractors in turbulence," in *Proceedings of Symposium on Dynamical Systems and Turbulence* (A. Dold and B. Eckmann, eds.), pp. 366 – 381, Lecture Notes in Mathematics, 1980.

[8] D. S. Broomhead and G. P. King, *Nonlinear Phenonema and Chaos*, ch. On the Qualitative Analysis of Experimental Dynamical Systems, pp. 113 – 144. Bristol: Adam Hilger, 1986.

[9] A. I. Mees, P. E. Rapp, and L. S. Jennings, "Singular–value decomposition and embedding dimension," *Physical Review A*, vol. 36, pp. 340 – 346, July 1987.

[10] A. Kumar and S. K. Mullick, "Nonlinear dynamical aspects of speech," *Journal of the Acoustical Society of America*, vol. 100, pp. 737 – 793, September 1996.

[11] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. New York: Springer–Verlag, 1989.

[12] C. Murgia, I. Mann, and G. Feng, "An algorithm for the estimation of glottal closure instants using the sequential detection of abrupt changes in speech signals," in *European Signal Processing Conference*, (Edinburgh), pp. 1685 – 1688, 1994.

[13] W. Hess, *Pitch Determination of Speech Signals*. Springer–Verlag, 1983.

[14] F. Plante, G. F. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *EUROSPEECH'95*, vol. 1, pp. 837 – 840, September 1995.

STABLE SPEECH SYNTHESIS USING RECURRENT RADIAL BASIS FUNCTIONS

Iain Mann and Steve McLaughlin
Dept. of Electronics and Electrical Engineering,
University of Edinburgh, King's Buildings,
Mayfield Road, Edinburgh. EH9 3JL. UK
Tel: +44 131 6505655; fax: +44 131 6506554
e-mail: inm@ee.ed.ac.uk, sml@ee.ed.ac.uk

ABSTRACT

The problem of stable vowel sound synthesis using a nonlinear free-running recurrent radial basis function (RBF) neural network is addressed. Voiced speech production is modelled as the output of a nonlinear dynamical system, rather than the conventional linear source–filter approach, which, given the nonlinear nature of speech, is expected to produce more natural–sounding synthetic speech. Our RBF network has the centre positions fixed on a hyper–lattice, so only the linear–in–the–parameters weights need to be learnt for each vowel realisation. This leads to greatly reduced complexity without degrading performance. The proposed structure, in which regularisation theory is used in learning the weights, is demonstrated to be stable when functioning in recurrent mode with no external input, correctly producing the desired vowel sound.

1 INTRODUCTION

In this paper we describe a neural network approach to the synthesis of voiced speech sounds. Our new technique uses a recurrent radial basis function with only a relatively small number of varying parameters and, once trained, is able to function as a stable nonlinear oscillator with no external input.

Nonlinear techniques, generally in the form of neural networks, have been applied to speech synthesis previously by several researchers. The interest in using a nonlinear method, as opposed to conventional linear ones, stems from the fact that speech has been demonstrated to be a nonlinear process [1]. Therefore a nonlinear technique should be better able to model the signal.

An RBF network for generating vowel sounds was proposed in [2]. It uses an extended Kalman filter approach to learn the weights, with centres that are chosen as a subset of the input data. This structure, although sometimes able to resynthesise the desired sound, is generally very unstable. Further work [3] solved this problem by blocking the input signal into frames and using all of the data within each frame as centres. This method is able to resynthesise on a frame–by–frame basis, but is unsatisfactory due to the very large number of varying parameters and the fact that it is not free–running synthesis *per se*. A different approach, using multi–layer perceptrons (MLP's), rather than RBF's, is taken in [4]. Only the Japanese vowel /*a*/ is examined, but it appears that stable resynthesis is achieved. However the learning process in MLP's is difficult and costly, and there will be a large number of nonlinear parameters varying between different vowel realisations. At the same time, work on modelling radar back scatter from the sea has produced free–running RBF synthesis structures [5] which make use of regularisation theory to ensure stability. However it is not clear if this technique uses all of the training data as centres. Our new method draws on this application of regularisation theory to produce a stable free–running synthesiser of vowel sounds. It also uses only a small number of centres, whose positions are fixed, thus greatly reducing complexity.

We first present the structure of the recurrent RBF network and describe its operation. We then consider issues relating to the stability of the network, and show how regularisation theory allows us to create a completely stable structure. Following this some results of synthesised speech are presented, and we conclude with a discussion of the problems and possibilities found with our new technique.

2 RADIAL BASIS FUNCTION NETWORK

The radial basis function network is used to approximate the underlying nonlinear system producing a particular stationary voiced sound. Specifically, it is trained to perform the prediction

$$y_{i+1} = \mathcal{F}\{\mathbf{y}_i\} \tag{1}$$

where $\mathbf{y}_i = \{y_i, y_{(i-\tau)}, \ldots, y_{(i-(m-1)\tau)}\}$ is a vector of previous inputs spaced by some delay τ samples, and $\mathcal{F}$ is a nonlinear mapping function. Considering the problem from a nonlinear dynamical theory perspective, we can view this as a time delay embedding of the one dimensional speech signal into an m–dimensional state space, producing an attractor reconstruction of the underlying d–dimensional system. Takens theorem states that $m \geq 2d + 1$ for an adequate embedding [6], although in practise it is often possible to reduce m. Appropriate choice of the non–trivial delay τ opens up this reconstruction. It

This work was supported by an EPSRC CASE award with BT Labs, Martlesham. SML also acknowledges the support of the Royal Society.

has been shown that vowels are low dimensional, having a dimension of around three [7], and that a value of $\tau = 10$ samples is suitable for a sampling rate of 22.05 kHz [8]. In our RBF, $m = 6$ taps are used, which is sufficient to give good results. The general structure is shown in Figure 1, for both training and synthesis modes of operation. The centres are spread uniformly over an m–dimensional

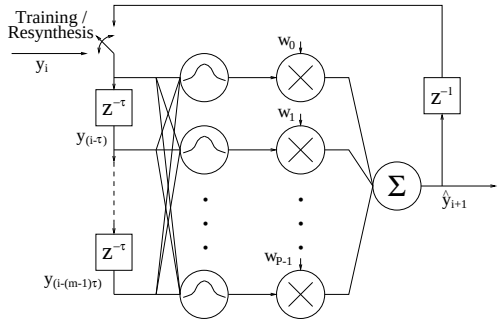


Figure 1: *Recurrent RBF network structure.*

hyper–lattice, rather than being chosen as a subset of the input data as is more usually the case. The network output (using P Gaussian centres) is given by

$$\hat{y}_{i+1} = \sum_{j=0}^{P-1} \phi_{ij} w_j \tag{2}$$

where the output of each centre is

$$\phi_{ij} = \exp\left(\frac{-\|\mathbf{y}_i - \mathbf{c}_j\|}{2\sigma^2}\right) \tag{3}$$

$\mathbf{c}_k$ being the position of the k–th centre on the hyper–lattice, and all the centres having a bandwidth σ^2 (determined experimentally). Each k–th centre output is multiplied by a weight w_k. With the centre positions known, the determination of the weights is linear in the parameters and may be solved by standard matrix inversion techniques, subject to some stability issues which are discussed in the following section.

Once the weights have been determined, we can switch from training to synthesis mode: the input signal is removed and the delayed network output is fed back into the delay line to form a free–running recurrent RBF network, allowing any duration of signal to be synthesised. Since the centre positions are fixed on the hyper–lattice identically for all signals, this means that there are only P varying parameters in our system (*i.e.* the elements of $\mathbf{w}$). This greatly reduces the complexity of the system without reducing performance. Figure 2 shows a comparison of mean square prediction error for a regularised RBF network trained with centres on a hyper–lattice and for one where the centres are chosen as a subset of the training data, over a database of four speakers (two male and two female). Very similar performance is observed for both methods, although in all cases our method gives on average approximately 1 dB better performance, clearly justifying the use of the hyper–lattice.

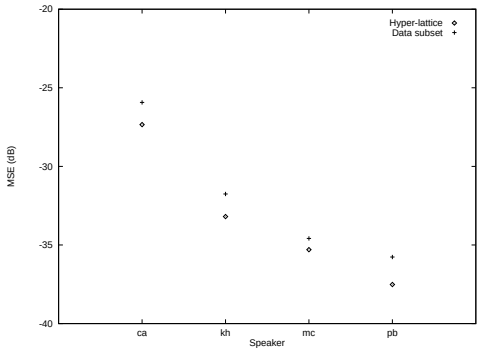


Figure 2: *MSE comparison of centre positioning strategies for 4 speakers, averaged over the 3 vowels /i/, /a/ and /u/. The RBF network was trained with 800 samples, using 64 centres (either on a 6–dimensional hyper–lattice, or as a subset of the training data). The one–step–ahead MSE was then calculated on an unseen 800 sample validation set.*

3 STABILITY ISSUES

In this section we will consider how the training technique for learning the weights must be modified in order to achieve stable resynthesis. Conventionally the weights are determined to minimise a sum of squares error function, $E_s(\hat{\mathcal{F}})$, over N samples of training data:

$$E_s(\hat{\mathcal{F}}) = \frac{1}{2}\sum_{i=0}^{N-1}(\hat{y}_i - y_i)^2 \tag{4}$$

where $\hat{y}_i$ is the network approximation of the actual speech signal y_i. Incorporating Equation 2 into the above and differentiating with respect to the weights, then setting the derivative equal to zero gives the least-squares problem [9]. This can be written in matrix form as

$$(\mathbf{\Phi}^T\mathbf{\Phi})\mathbf{w}^T = \mathbf{\Phi}^T\mathbf{y} \tag{5}$$

where $\mathbf{\Phi}$ is an NxP matrix of the outputs of the centres; $\mathbf{y}$ is the target vector of length N; and $\mathbf{w}$ is the P length vector of weights. This is normally solved by finding the pseudo–inverse by singular value decomposition.

However, our experiments have shown that if the weights are determined in this manner it is not possible to guarantee stable resynthesis when the RBF network is operated in the free–running recurrent mode. In fact it is generally the case that the output is erroneous, with a correct output found only at certain training lengths for certain input signals. More over it does not seem possible to predict which signals will produce a correct output and which will not. Very similar observations are made by Birgmeier [2, 10]. Types of erroneous output encountered include periodic limit cycles not resembling the original speech signal; constant (sometimes zero) output; extremely large spikes at irregular intervals on otherwise correct waveforms. We note here that these observations

hold true irrespective of centre positioning (*i.e.* as a subset of the training data or as a hyper–lattice), bandwidth and variations of other network parameters.

This problem can be solved by the application of regularisation theory, first described as applied to RBF's in [11]. In essence this means the application of some smoothing factor to the approximated function $\hat{\mathcal{F}}$. Mathematically, we define a new cost function

$$E(\hat{\mathcal{F}}) = E_s(\hat{\mathcal{F}}) + \lambda E_r(\hat{\mathcal{F}}) = E_s(\hat{\mathcal{F}}) + \frac{1}{2}\lambda \|\mathbf{P}\hat{\mathcal{F}}\|^2 \quad (6)$$

where $E_s(\hat{\mathcal{F}})$ is the sum of squares error defined previously, $E_r(\hat{\mathcal{F}})$ is the regularising term and λ is the real, positive regularisation parameter. A full derivation is beyond the scope of this paper (see the chapter on regularisation theory in [12]), but given the appropriate choice of the differential operator $\mathbf{P}$, it is possible to write the solution for the weights for the specific case of all training data being used as centres ($P = N$) as

$$\mathbf{w} = (\mathbf{G} + \lambda \mathbf{I})^{-1}\mathbf{y} \quad (7)$$

where $\mathbf{I}$ is the $N\mathrm{x}N$ identity matrix and $\mathbf{G}$ is an $N\mathrm{x}P$ matrix of Green's functions, which is equal to the $N\mathrm{x}P$ matrix $\mathbf{\Phi}$.

This solution is only of limited interest, since we have to use all of the training data as centres, but it can be extended to the approximate regularised solution, where $P < N$ and the centres are not necessarily on the training data. In this case we obtain

$$\mathbf{w} = (\mathbf{G}^T\mathbf{G} + \lambda \mathbf{G}_0)^{-1}\mathbf{G}^T\mathbf{y} \quad (8)$$

where $\mathbf{G}_0$ is a symmetric $P\mathrm{x}P$ matrix composed of elements $G(\mathbf{c}_i; \mathbf{c}_j)$:

$$G(\mathbf{c}_i; \mathbf{c}_j) = \exp\left(\frac{-\|\mathbf{c}_i - \mathbf{c}_j\|}{2\sigma^2}\right); \quad 0 \leq i, j < P - 1 \quad (9)$$

When $\lambda = 0$ there is no regularisation, and as λ is increased so greater smoothness is imposed onto $\hat{\mathcal{F}}$. Our results have shown that by using this method, stable realistic resynthesis is possible for all the signals in our database.

4 VOWEL RESYNTHESIS

In order to test the ability of our RBF structure to synthesise vowels in a stable and realistic manner, we examined its performance on our specially recorded database of stationary vowels. From the 16 speakers in the database, we randomly chose two male and two female speakers. Their utterances of the three cardinal vowels /*i*/, /*a*/ and /*u*/ form the basis of our test. All signals were sampled at 22.05 kHz and have 16 bit resolution. The RBF network was trained using 800 samples of the original waveform, before being set into free–run synthesis mode to generate the desired length signal. The 6–dimensional hyper–lattice used has 64 centres, corresponding to a centre at each 'corner' of the hyper–lattice. The value of the regularisation parameter, λ, was set at 10^{-1}. In all cases we were able to synthesise stable, speech–like waveforms. As an assessment of the quality of these synthesised waveforms, they were compared against the signal generated by the classic linear prediction synthesiser. In this case, the linear prediction (LP) filter coefficients were found from the original vowel sound (analogous to the training stage of the RBF network). Using the estimate $(F_s + 4)$ [13], we calculated a model with 26 poles. Then, using the source–filter model, the linear prediction filter was excited by a dirac pulse train to produce the desired length LP synthesised signal. The distance between dirac pulses was set to be equal to the pitch period of the original signal. Figure 3 shows the time domain signals generated by the two techniques for an example of the vowel /*u*/ (speaker *mc*). The original vowel sound is also included for comparative purposes. Figure 4 shows the corresponding frequency

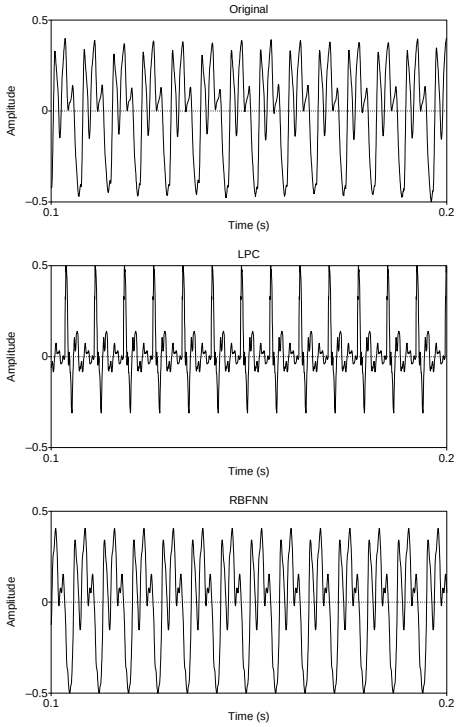


Figure 3: *Time domain examples of the vowel* /u/ *(speaker* mc*). From top to bottom: original signal; linear prediction synthesised signal; RBF network synthesised signal.*

domain plots of the three signals. For the sake of clarity the spectral envelope (calculated via the LP coefficients), rather than the full Fourier spectrum, is shown.

5 DISCUSSION

Figures 3 and 4 clearly demonstrate the different modelling strategies between the LP and RBF approaches. In the linear prediction case, the technique attempts to model the

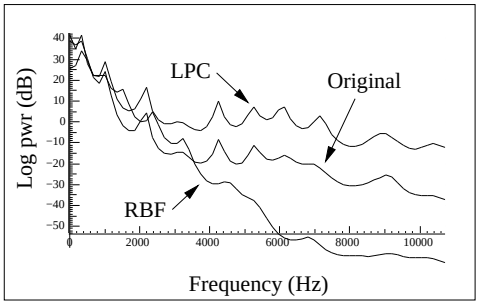


Figure 4: *Spectral envelopes for examples of the vowel* /u/*, corresponding to the signals in Figure 3.*

spectral features of the original, which is done by assuming that the excitation signal can be separated out. Hence the reasonable match seen in the spectral envelope (although the high frequencies have been over–emphasised), but the lack of resemblance in the time domain. The RBF network, on the other hand, attempts to model the complete underlying dynamical system (Equation 1). Thus the RBF synthesised signal resembles the original in the time domain: the RBF network has successfully captured many of the original signal characteristics. However the spectral plots show that the higher frequencies have not been well modelled by this method. This is because the RBF network has missed some of the very fine variations of the original time domain waveform, due to the imposition of the regularisation parameter λ. This, as described previously, adds smoothness to the approximated function $\hat{\mathcal{F}}$ to ensure stability. When listening to the signals in an informal listening test, most listeners preferred the RBF generated vowel against the LP synthesis result. The RBF signal was generally thought to be perceptually closer to the original in terms of naturalness, although it was still noticeably different.

6 CONCLUSION

We have presented a recurrent radial basis function network for vowel synthesis. By applying regularisation theory we can ensure the stability of the network, which has previously been found to be a problem with this technique. In addition, through the use of a hyper–lattice, the centre positions can be fixed in state space, thus reducing the number of varying parameters to the set of linear weights. This structure has been compared to the well–known source–filter linear model, with an example of the vowel /u/ presented. Our technique produces a more accurate version of the time–domain waveform, but does not adequately model the higher frequencies. This is due to the use of regularisation: by forcing the approximated function to be smooth to give stability, we implicitly remove some higher frequency information. However the vowel generated by the RBF network is more natural–sounding than the LP synthesised sound, justifying the use of a nonlinear method.

7 ACKNOWLEDGEMENTS

The authors would like to thank B. Mulgrew of Edinburgh University for useful discussions about this work.

References

[1] G. Kubin, *Speech Coding and Synthesis*, ch. Nonlinear Processing of Speech, pp. 557 – 610. Elsevier, 1995.

[2] M. Birgmeier, "A fully kalman–trained radial basis function network for nonlinear speech modelling," in *Proceedings of IEEE International Conference on Neural Networks*, (Perth, Australia), pp. 259 – 264, November 1995.

[3] G. Kubin, "Synthesis and coding of continuous speech with the nonlinear oscillator model," in *International Conference on Acoustics, Speech and Signal Processing*, (Atlanta, Georgia), pp. 267 – 270, May 1996.

[4] I. Tokuda, R. Tokunaga, and K. Aihara, "A simple geometrical structure underlying speech signals of the japanese vowel /a/," *International Journal of Bifurcation and Chaos*, vol. 6, no. 1, pp. 149 – 160, 1996.

[5] S. Haykin and J. Principe, "Making sense of a complex world," *IEEE Signal Processing Magazine*, vol. 15, pp. 66 – 81, May 1998.

[6] F. Takens, "Detecting strange attractors in turbulence," in *Proceedings of Symposium on Dynamical Systems and Turbulence* (A. Dold and B. Eckmann, eds.), pp. 366 – 381, Lecture Notes in Mathematics, 1980.

[7] A. Kumar and S. K. Mullick, "Nonlinear dynamical aspects of speech," *Journal of the Acoustical Society of America*, vol. 100, pp. 737 – 793, September 1996.

[8] M. Banbrook, *Nonlinear Analysis of Speech from a Synthesis Perspective*. PhD thesis, University of Edinburgh, Edinburgh, September 1996.

[9] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[10] M. Birgmeier, *Kalman–trained Neural Networks for Signal Processing Applications*. PhD thesis, Technical University of Vienna, Vienna, 1996.

[11] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481 – 1497, September 1990.

[12] S. Haykin, *Neural Networks: A comprehensive foundation*. Prentice Hall, 1994.

[13] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice–Hall, 1978.

Appendix D
CD

This appendix describes the contents of the CD which accompanies the thesis. It is divided into two sections, one containing audio files and the other example source code. The files are also accessible via an html page on the CD called `index.html`.

D.1 Audio examples

In the `Audio/` directory, a set of example synthesised vowel sounds, with the original speech for comparison, is provided. Both SUN and Microsoft formats are given. The file name labels are used to indicate speaker, vowel and type of signal (type of synthesis / original).

Files are labelled as `ss_v_ttt.ext` where:

- `ss` is the speaker's initials: mc = MC (male), pb = PB (male), ca = CA (female), kh = KH (female).
- `v` is the vowel: `i` = /i/, `a` = /ɑ/, `u` = /u/.
- `ttt` is the speech type: `org` = original, `lls` = locally linear synthesis, `lpc` = linear prediction synthesis, `lat` = RBF synthesis with hyper–lattice, `sub` = RBF synthesis with data subset.
- `ext` is the file type: `au` = SUN Sparc format, `wav` = Microsoft format.

D.2 Source code

The `Source/` directory contains source code for the Poincaré pitch marker as described in Chapter 4 and the RBF synthesiser described in Chapters 6 and 7. The Poincaré pitch marker uses the xview graphical user interface and, as such, can only be used without modification on UNIX systems which support this. No user interface is required for the RBF software, although it should be noted that it has only been tested on UNIX systems with a SUN operating system.

Reference should be made to the `readme.txt` file on the CD for further details on compiling and running the software.

References

[1] J. D. O'Connor, *Phonetics*. Penguin Books, 1991.

[2] G. E. Peterson and H. L. Barney, "Control methods used in a study of the vowels," *Journal of the Acoustical Society of America*, vol. 24, pp. 175 – 184, March 1952.

[3] E. Moulines and W. Verhelst, *Speech Coding and Synthesis*, ch. Time–domain and Frequency–domain Techniques for Prosodic Modification of Speech, pp. 519 – 555. Elsevier, 1995.

[4] T. Quatieri and R. McAulay, "Speech transformations based on a sinusoidal representation," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 34, pp. 1449 – 1464, December 1986.

[5] M. Banbrook, *Nonlinear Analysis of Speech from a Synthesis Perspective*. PhD thesis, University of Edinburgh, Edinburgh, September 1996.

[6] M. Banbrook and S. McLaughlin, "Dynamical modelling of vowel sounds as a synthesis tool," in *4th International Conference on Spoken Language Processing*, vol. 3, pp. 1981 – 1984, October 1996.

[7] M. Banbrook, S. McLaughlin, and I. Mann, "Speech characterization and synthesis by nonlinear methods," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 7, pp. 1 – 17, January 1999.

[8] A. Breen, "Speech synthesis models: a review," *Electronics and Communication Engineering Journal*, vol. 4, pp. 19 – 31, February 1992.

[9] T. DuToit, *An Introduction to Text–To–Speech Synthesis*. Kluwer, 1996.

[10] B. Townshend, "Nonlinear prediction of speech signals," in *Nonlinear Modeling and Forcasting* (M. Casdagli and S. Eubank, eds.), vol. 12, pp. 433 – 453, Addison–Wesley, 1992.

[11] N. Morgan and H. Bourlard, "Continuous speech recognition," *IEEE Signal Processing Magazine*, vol. 12, pp. 24 – 42, May 1995.

[12] G. Kubin, *Speech Coding and Synthesis*, ch. Nonlinear Processing of Speech, pp. 557 – 610. Elsevier, 1995.

[13] S. Haykin, *Neural Networks: A comprehensive foundation*. Prentice Hall, 1994.

[14] S. A. Gelfand, *Hearing: An Introduction to Psychological and Physiological Acoustics*. New York: Marcel Dekker, 1998.

[15] R. Sproat and J. Olive, *Speech Coding and Synthesis*, ch. An Approach to Text–to–Speech Synthesis, pp. 611 – 633. Elsevier, 1995.

[16] D. H. Klatt, "Review of text–to–speech conversion for English," *Journal of the Acoustical Society of America*, vol. 82, pp. 737 – 793, September 1987.

[17] R. W. Sproat and J. P. Olive, "Text–to–speech synthesis," *AT&T Technical Journal*, vol. 74, no. 2, pp. 35 – 44, 1995.

[18] M. Edgington, A. Lowry, P. Jackson, A. Breen, and S. Minnis, "Overview of current text–to–speech techniques: Part I – text and linguistic analysis," *BT Technical Journal*, vol. 14, pp. 68 – 83, January 1996.

[19] J. Terkin and R. Collier, *Speech Coding and Synthesis*, ch. The Generation of Prosodic Structure and Intonation in Speech Synthesis, pp. 635 – 662. Elsevier, 1995.

[20] P. Howell, "Cue trading in the production and perception of vowel stress," *Journal of the Acoustical Society of America*, vol. 94, pp. 2063 – 2073, October 1993.

[21] C. W. Wightman, S. Shattuck-Hufnagel, M. Ostendorf, and P. J. Price, "Segmental durations in the vicinity of prosodic phrase boundaries," *Journal of the Acoustical Society of America*, vol. 91, pp. 1707 – 1717, March 1992.

[22] T. H. Crystal and A. S. House, "Segmental durations in connected speech: Current results," *Journal of the Acoustical Society of America*, vol. 83, pp. 1553 – 1573, April 1988.

[23] W. N. Campbell and S. D. Isard, "Segment durations in a syllable frame," *Journal of Phonetics*, vol. 19, pp. 37 – 47, 1991.

[24] B. Gabioud, *Fundamentals of Speech Synthesis and Speech Recognition*, ch. Articulatory Models in Speech Synthesis, pp. 215 – 230. John Wiley & Sons, 1994.

[25] K. Ishizaka and J. L. Flanagan, "Synthesis of voiced sounds from a two–mass model of the vocal chords," *Bell System Technical Journal*, vol. 51, pp. 1233 – 1268, July–August 1972.

[26] T. Koizumi, S. Taniguchi, and S. Hiromitsu, "Two–mass models of the vocal cords for natural sounding voice synthesis," *Journal of the Acoustical Society of America*, vol. 82, pp. 1179 – 1192, October 1987.

[27] G. C. Hergerl and H. Hoge, "Numerical simulation of the glottal flow by a model based on the compressible Navier–Stokes equations," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 477 – 480, IEEE, 1991.

[28] S. Maeda, "Improved articulatory model," *Journal of the Acoustical Society of America*, vol. 84, no. S1, p. S146, 1988.

[29] L.-J. Boe, P. Perrier, and G. Bailly, "The geometric vocal tract variables controlled for vowel production: Proposals for constraining acoustic–to–articulatory inversion," *Journal of Phonetics*, vol. 20, pp. 27 – 38, 1992.

[30] G. Fant, *Acoustic Theory of Speech Production*. Mouton, 1960.

[31] J. Markel and A. Gray, *Linear Prediction of Speech*. Berlin: Springer–Verlag, 1976.

[32] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *Journal of the Acoustical Society of America*, vol. 50, no. 2, pp. 637 – 655, 1971.

[33] G. Kubin and B. S. Atal, "Linear autoregressive modeling of unvoiced speech," *Journal of the Acoustical Society of America*, vol. 93, p. 2354, April 1993.

[34] D. H. Klatt, "Software for a cascade/parallel formant synthesiser," *Journal of the Acoustical Society of America*, vol. 67, pp. 971 – 995, 1980.

[35] M. Edgington, A. Lowry, P. Jackson, A. Breen, and S. Minnis, "Overview of current text–to–speech techniques: Part II – prosody and speech generation," *BT Technical Journal*, vol. 14, pp. 84 – 99, January 1996.

[36] Y. Sagisaka and N. Iwahashi, *Speech Coding and Synthesis*, ch. Objective Optimization in Algorithms for Text–to–speech Synthesis, pp. 685 – 706. Elsevier, 1995.

[37] J. Page and A. Breen, "The laureate text–to–speech system, architecture and applications," *BT Technical Journal*, vol. 14, pp. 57 – 67, January 1996.

[38] M. Beutnagel, A. Conkie, J. Schroeter, and A. Syrdal, "The at&t next–gen tts system," in *Joint Meeting of ASA, EAA, and DAGA*, (Berlin, Germany), March 1999.

[39] D. W. Griffin and J. S. Lim, "Signal estimation from modified short–time Fourier transform," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 32, no. 2, pp. 236 – 243, 1984.

[40] E. Moulines and F. Charpentier, "Pitch synchronous waveform processing techniques for text–to–speech synthesis using diphones," *Speech Communication*, vol. 9, pp. 453 – 467, 1990.

[41] R. Veldhuis and H. He, "Time–scale and pitch modifications of speech signals and resynthesis from the discrete short–time Fourier transform," *Speech Communication*, vol. 18, pp. 257 – 279, 1996.

[42] E. Moulines and J. Laroche, "Non–parametric techniques for pitch–scale and time–scale modification of speech," *Speech Communication*, vol. 16, pp. 175 – 205, 1995.

[43] F. M. G. de los Galanes, M. H. Savoji, and J. M. Pardo, "New algorithm for spectral smoothing and envelope modification for LP–PSOLA synthesis," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 573 – 576, 1994.

[44] M. Edgington and A. Lowry, "Residual based speech modification algorithms for text–to–speech synthesis," in *International Conference on Spoken Language Processing*, 1996.

[45] T. DuToit and H. Leich, "MBR–PSOLA: Text–to–speech synthesis based on an MBE re–synthesis of the segments database," *Speech Communication*, vol. 13, pp. 435 – 440, 1993.

[46] D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 36, pp. 1223 – 1235, August 1988.

[47] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 34, pp. 744 – 754, August 1986.

[48] T. Quatieri and R. McAulay, "Shape invariant time–scale and pitch modification of speech," *IEEE Transactions on Signal Processing*, vol. 40, pp. 497 – 510, March 1992.

[49] J. Laroche, Y. Stylianou, and E. Moulines, "HNS: Speech modification based on a harmonic + noise model," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 550 – 553, 1993.

[50] K. Lam and C. Chan, "Interpolating V/UV mixture functions of a harmonic model for concatenative speech synthesis," in *International Conference on Acoustics, Speech and Signal Processing*, 1996.

[51] A. J. Abrantes, J. S. Marques, and I. M. Trancoso, "Hybrid sinusoidal modelling of speech without voicing decision," in *Eurospeech*, pp. 231 – 234, 1991.

[52] M. J. Kearney and J. Stark, "An introduction to chaotic signal processing," *GEC Journal of Research*, vol. 10, no. 1, pp. 52 – 58, 1992.

[53] T. S. Parker and L. O. Chua, "Chaos; a tutorial for engineers," *Proceedings of the IEEE*, vol. 75, pp. 982 – 1008, August 1987.

[54] J. Crutchfield, J. Farmer, N. Packard, and R. Shaw, "Chaos," *Scientific American*, vol. 255, pp. 38 – 49, December 1986.

[55] J. P. Eckmann and D. Ruelle, "Ergodic theory of chaos and strange attractors," *Review of Modern Physics*, vol. 57, pp. 617 – 656, July 1985.

[56] K. Alligood, T. Sauer, and J. Yorke, *Chaos: An Introduction to Dynamical Systems*. New York: Spinger–Verlag, 1997.

[57] R. Hilborn, *Chaos and Nonlinear Dynamics*. Oxford: Oxford University Press, 1994.

[58] E. Ott, T. Sauer, and J. Y. (Eds), *Coping with Chaos*. New York: Wiley, 1994.

[59] P. Grassberger and I. Procaccia, "Characterization of strange attractors," *Physical Review Letters*, vol. 50, pp. 346 – 349, January 1983.

[60] J. D. Farmer, E. Ott, and J. A. Yorke, "The dimension of chaotic attractors," *Physica D*, vol. 7, no. 1–3, pp. 153 – 180, 1983.

[61] F. Takens, "Detecting strange attractors in turbulence," in *Proceedings of Symposium on Dynamical Systems and Turbulence* (A. Dold and B. Eckmann, eds.), pp. 366 – 381, Lecture Notes in Mathematics, 1980.

[62] T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," *Journal of Statistical Physics*, vol. 65, no. 3-4, pp. 579 – 616, 1991.

[63] D. S. Broomhead and G. P. King, *Nonlinear Phenomena and Chaos*, ch. On the Qualitative Analysis of Experimental Dynamical Systems, pp. 113 – 144. Bristol: Adam Hilger, 1986.

[64] A. I. Mees, P. E. Rapp, and L. S. Jennings, "Singular–value decomposition and embedding dimension," *Physical Review A*, vol. 36, pp. 340 – 346, July 1987.

[65] H. D. I. Abarbanel, *Analysis of Observed Chaotic Data*. Spinger–Verlag, 1996.

[66] A. M. Fraser and H. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical Review A*, vol. 33, pp. 1134 – 1140, 1986.

[67] H. P. Bernhard and G. Kubin, "Detection of chaotic behaviour in speech signals using Fraser's mutual information algorithm," in *Proceedings of the 13th GRETSI Symposium on Signal and Image Processing*, (Juan–les–pins, France), pp. 1301 – 1311, 1991.

[68] A. M. Fraser, "Information and entropy in strange attractors," *IEEE Transactions on Information Theory*, vol. 35, pp. 245 – 262, March 1989.

[69] H. P. Bernhard and G. Kubin, "A fast mutual information calculation algorithm," in *Eusipco'94*, pp. 50 – 53, 1994.

[70] A. Wolf, J. Swift, H. Swinney, and J. Vastano, "Determining lyapunov exponents from a time series," *Physica D*, vol. 16, pp. 285 – 317, 1985.

[71] M. Sano and Y. Sawada, "Measurement of the lyapunov spectrum from a chaotic time series," *Physical Review Letters*, vol. 55, pp. 1082 – 1085, September 1985.

[72] P. Bryant, R. Brown, and H. D. Abarbanel, "Lyapunov exponents from observed time series," *Physical Review A*, vol. 65, pp. 1523 – 1526, 1990.

[73] A. G. Darbyshire and D. S. Broomhead, "Robust estimation of tangent maps and liapunov spectra," *Physica D*, vol. 89, no. 3-4, pp. 287 – 305, 1996.

[74] H. D. I. Abarbanel, R. Brown, and M. B. Kennel, "Variation of lyapunov exponents on a strange attractor," *Journal of Nonlinear Science*, vol. 1, pp. 175 – 199, 1991.

[75] Calliope, *La Parole et son Traitement Automatique*. Masson, 1989.

[76] T. Koizumi, S. Taniguchi, and S. Hiromitsu, "Glottal source–vocal tract interaction," *Journal of the Acoustical Society of America*, vol. 78, pp. 1541 – 1547, November 1985.

[77] D. M. Brookes and P. A. Naylor, "Speech production modelling with variable glottal reflection coefficient," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 671 – 674, 1988.

[78] H. M. Teager and S. M. Teager, "Evidence of nonlinear sound production mechanisms in the vocal tract," in *Proceedings of the NATO Advanced Study Institute on Speech Production and Modelling*, (Bonas, France), pp. 241 – 261, July 1989.

[79] I. Steinecke and H. Herzel, "Bifurcations in an asymmetric vocal–fold model," *Journal of the Acoustical Society of America*, vol. 97, pp. 1874 – 1884, March 1995.

[80] J. Schoentgen, "Non–linear signal representation and its application to the modelling of the glottal waveform," *Speech Communication*, vol. 9, pp. 189 – 201, 1990.

[81] N. Tishby, "A dynamical systems approach to speech processing," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 365 – 368, 1990.

[82] B. Townshend, "Nonlinear prediction of speech," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 425 – 428, 1991.

[83] M. Casdagli, "Chaos and deterministic versus stochastic non–linear modelling," *Journal of the Royal Statistical Society B*, vol. 54, no. 2, pp. 303 – 328, 1991.

[84] S. McLaughlin and A. Lowry, "Nonlinear dynamical systems concepts in speech analysis," in *Eurospeech'93*, pp. 377 – 380, 1993.

[85] M. Kennel, R. Brown, and H. I. Abarbanel, "Determining embedding dimension for phase space reconstruction using a geometrical construction," *Physical Review A*, vol. 45, no. 6, pp. 3403 – 3411, 1992.

[86] H. P. Bernhard and G. Kubin, "Speech production and chaos," in *Proceedings of the 12th International Congress of Phonetic Sciences*, (Aix–en–Provence, France), pp. 394 – 397, August 1991.

[87] A. Kumar and S. K. Mullick, "Nonlinear dynamical aspects of speech," *Journal of the Acoustical Society of America*, vol. 100, pp. 737 – 793, September 1996.

[88] I. Tokuda, R. Tokunaga, and K. Aihara, "Laryngeal vibrations: Towards understanding causes of irregularities in speech signals of vowel sounds," in *Towards the Harnessing of Chaos* (M. Yamaguti, ed.), pp. 413 – 418, Elsevier, 1994.

[89] I. Tokuda, R. Tokunaga, and K. Aihara, "A simple geometrical structure underlying speech signals of the Japanese vowel /a/," *International Journal of Bifurcation and Chaos*, vol. 6, no. 1, pp. 149 – 160, 1996.

[90] I. Tokuda, T. Miyano, and K. Aihara, "Are normal vowels really chaotic," in *1997 International Symposium on Nonlinear Theory and its Applications*, (Honolulu, U.S.A.), pp. 1173 – 1176, November 1997.

[91] I. Tokuda, T. Miyano, and K. Aihara, "Spike–and–wave surrogate analysis of Japanese vowels," in *1998 International Symposium on Nonlinear Theory and its Applications*, (Crans–Montana, Switzerland), pp. 719 – 722, September 1998.

[92] N. Aoki and T. Ifukube, "Fractal modeling of fluctuations in the steady part of sustained vowels for high quality speech synthesis," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E81A, pp. 1803 – 1810, September 1998.

[93] S. S. Narayanan and A. A. Alwan, "A nonlinear dynamical systems analysis of fricative consonants," *Journal of the Acoustical Society of America*, vol. 97, pp. 2511 – 2524, April 1995.

[94] M. Banbrook, G. Ushaw, and S. McLaughlin, "How to extract lyapunov exponents from short and noisy time series," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1378 – 1382, 1997.

[95] M. Banbrook, G. Ushaw, and S. McLaughlin, "Lyapunov exponents from a time series: A noise–robust extraction algorithm," *Chaos, Solitons and Fractals*, vol. 7, no. 7, pp. 973 – 976, 1996.

[96] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice–Hall, 1978.

[97] W. Hess, *Pitch Determination of Speech Signals*. Springer–Verlag, 1983.

[98] R. J. DiFrancesco and E. Moulines, "Detection of glottal closure by jumps in the statistical properties of the speech signal," *Speech Communication*, vol. 9, pp. 401 – 418, December 1990.

[99] Y. M. Cheng and D. O'Shaughnessy, "Automatic and reliable estimation of glottal closure instant and period," *IEEE Transactions on Audio, Speech and Signal Processing*, vol. 37, pp. 1805 – 1815, December 1989.

[100] D. Talkin, "Voicing epoch determination with dynamic programming," *Journal of the Acoustical Society of America*, vol. 85, Supplement 1, p. S149, 1989.

[101] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. New York: Springer–Verlag, 1989.

[102] G. Kubin, "Poincaré sections for speech," in *Proceedings of the 1997 IEEE Workshop on Speech Coding*, (Pocono Manor, USA), September 1997.

[103] C. Murgia, I. Mann, and G. Feng, "An algorithm for the estimation of glottal closure instants using the sequential detection of abrupt changes in speech signals," in *European Signal Processing Conference*, (Edinburgh), pp. 1685 – 1688, 1994.

[104] F. Plante, G. F. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *EUROSPEECH'95*, vol. 1, pp. 837 – 840, September 1995.

[105] Y. Horii, "Fundamental frequency perturbation observed in sustained phonation," *Journal of Speech and Hearing Research*, vol. 22, pp. 5 – 19, March 1979.

[106] J. Schoentgen and R. de Guchteneere, "An algorithm for the measurement of jitter," *Speech Communication*, vol. 10, pp. 533 – 538, 1991.

[107] L. P. Šilnikov, "A case of the existence of a denumerable set of periodic motions," *Soviet Math. Dokl.*, vol. 6, p. 163, 1965.

[108] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Physical Review Letters*, vol. 64, no. 11, pp. 1196 – 1199, 1990.

[109] U. Dressler and G. Nitsche, "Controlling chaos using time delay coordinates," *Physical Review Letters*, vol. 68, no. 1, pp. 1 – 4, 1992.

[110] W. H. Press *et al.*, *Numerical Recipes in C*. Cambridge University Press, 1992.

[111] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 302 – 309, March 1991.

[112] B. Mulgrew, "Applying radial basis functions," *IEEE Signal Processing Magazine*, vol. 13, pp. 50 – 65, March 1996.

[113] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[114] M. J. D. Powell, *Algorithms for Approximation*, ch. Radial Basis Functions for Multivariable Interpolation: A Review, pp. 143 – 167. Oxford University Press, 1987.

[115] C. Chinrungrueng and C. H. Sequin, "Optimal adaptive k–means algorithm with dynamic adjustment of learning rate," *IEEE Transactions on Neural Networks*, vol. 6, pp. 157 – 169, January 1995.

[116] P. E. Strauch, *Nonlinear Noise Cancellation*. PhD thesis, University of Edinburgh, 1997.

[117] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321 – 355, 1988.

[118] J. Moody and C. J. Darken, "Fast learning in networks of locally–tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281 – 294, 1989.

[119] J. Park and I. W. Sandberg, "Approximation and radial basis function networks," *Neural Computation*, vol. 5, pp. 305 – 316, 1993.

[120] G. Kubin, "Synthesis and coding of continuous speech with the nonlinear oscillator model," in *International Conference on Acoustics, Speech and Signal Processing*, (Atlanta, Georgia), pp. 267 – 270, May 1996.

[121] M. Birgmeier, "A fully kalman–trained radial basis function network for nonlinear speech modelling," in *Proceedings of IEEE International Conference on Neural Networks*, (Perth, Australia), pp. 259 – 264, November 1995.

[122] M. Birgmeier, *Kalman–trained Neural Networks for Signal Processing Applications*. PhD thesis, Technical University of Vienna, Vienna, 1996.

[123] G. Kubin and W. B. Kleijn, "Time–scale modification of speech based on a nonlinear oscillator model," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 453 – 456, 1994.

[124] K. Narashimhan, J. C. Principe, and D. Childers, "Nonlinear dynamic modeling of the voiced excitation for improved speech synthesis," in *International Conference on Acoustics, Speech and Signal Processing*, (Phoenix, Arizona), pp. 389 – 392, March 1999.

[125] S. Haykin, *Adaptive Filter Theory*. Prentice Hall, 1996.

[126] G. Feng, "Stable adaptive predictor for nonlinear systems using neural networks," *Computers and Electrical Engineering*, vol. 20, no. 5, pp. 383 – 390, 1994.

[127] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481 – 1497, September 1990.

[128] S. Haykin and J. Principe, "Making sense of a complex world," *IEEE Signal Processing Magazine*, vol. 15, pp. 66 – 81, May 1998.

[129] H. Haas and G. Kubin, "Multi–band nonlinear oscillator model for speech," in *32nd Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 338 – 342, 1998.

[130] F. M. A. Acosta, "Radial basis function and related models: an overview," *Signal Processing*, vol. 45, pp. 37 – 58, 1995.

[131] S. Young, "A review of large–vocabulary continuous–speech recognition," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45 – 57, 1996.

[132] R. Penrose, "A generalised inverse for matrices," *Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406 – 413, July 1955.