INNOVATIVE METHODS IN THE PREDICTION AND ANALYSIS OF SOLAR-TERRESTRIAL TIME SERIES

by

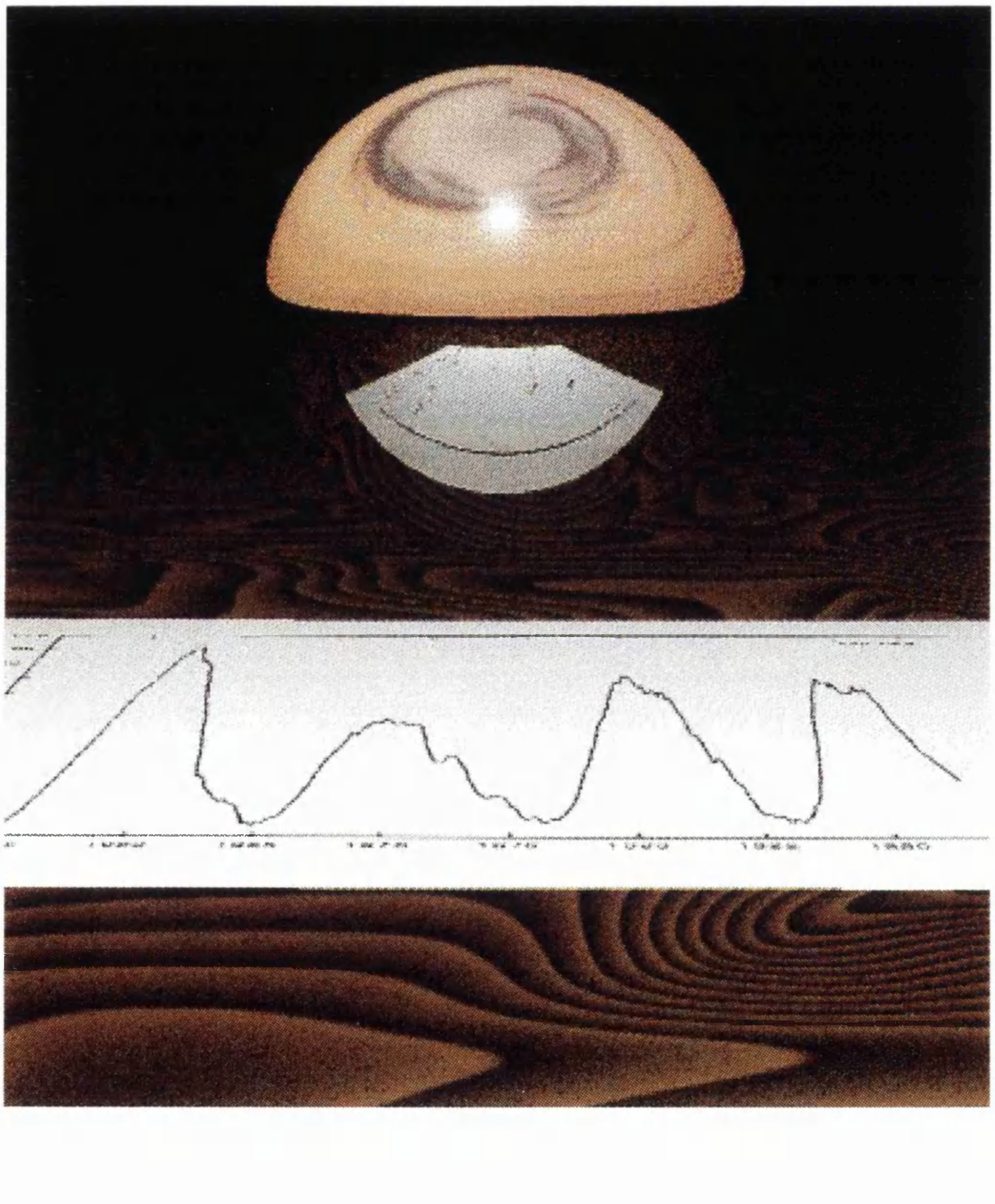
ANDREW JOHN CONWAY B.Sc. Hons.

Thesis
submitted to the
University of Glasgow
for the degree of
Ph.D.

Department of Physics
and Astronomy,
University of Glasgow,
Glasgow G12 8QQ.

October 1995

Thesis
10386
Copy 1



Acknowledgements

Although I have to admit that it was I who performed all the work that went into the creation of this thesis, the debt that I owe the following people is immeasurable. Whether it be for their advice on matters scientific, their general encouragement or their mere presence, I thank them all.

First and foremost I thank my parents Dr. Anthony P. Conway and Dr. Rafat Abdarabbani for all their love and encouragement. From my Mother I have learned the skill of self-discipline and inherited a fire of determination which has enabled me to work hard over the last seven years and meet many of my goals. From my father, I have gained my thirst for knowledge, my curiosity, as well as a sense of humour that serves to keep me sane (I theenok). To Elaine Rowan, who possesses the eerie ability to drag my mind away from work (even though she is unaware of it), I offer my deepest gratitude for all the support and love she has given me in the last two and a half years. I also thank Natasha Conway, not only for being my little sister (something in which she had no choice), but also for being a very good friend. Now that she has moved to Cambridge to do her own Ph.D. I will really miss her (but not her music).

In the department, my first thanks go out to my supervisor, Prof. John C. Brown, who has given me just the right blend of encouragement, guidance and freedom. I also thank him for his keen eye (the left one I think) and respect him for his amazing ability to identify the key issues and questions in almost any problem. At work and in bed, Dr. Keith Macpherson is a valued friend and colleague. The meticulous work that went into his thesis, as well as his reasoned voice were profound influences on my thinking and working throughout the last three years, and for that I am in debt to him. Over the last three years (soon-to-be) Dr. Sarah Matthews has had to put up with me more than most. I thank her for her friendship and also for listening to my endless stories and strange ideas. Life in the office would not have been as enjoyable as it was for me without the presence of Aidan Keane. His sense of humour and his unswerving ability to be an "all round nice guy" have brightened some of my gloomiest days - thank you Aidan. If someone asked me for an example of a true astronomer, I would lead them to the door of Dr. Shashi Kanbur. Shashi receives my profound gratitude for helping me in matters statistical and computational, as well as for his friendship over the last couple of years. A very special thank you is reserved for Miss Daphne Davidson, who has saved me from being swamped by practicalities on many occasions. Also, as any Glasgow graduate will tell you, Daphne's pleasant nature and slight insanities make her an integral part of the Astronomy & Astrophysics group. I also wish to thank Dr. John Simmons, Dr. Norman Gray, Dr. Guy Janin of ESOC, Mr. Andrew Wood, Dr. Martin Hendry, Mr. Iain Coleman, Dr. Dave Clarke, Dr. Robin Green, (soon-to-be) Dr. Giota Petkaki, Mr. David Keston, Prof. Rex Whitehead, Dr. Alec MacKinnon, Dr. Moray Anderson, Dr. Kenny Wood, (soon-to-be) Dr. Allan Wilson and (soon-to-be) Dr. Stephen Gowdy and all the others who I have had the pleasure of working with.

Outside the office, I would like to thank my old friend (real) Dr. Arun Bhatt for reminding me how to be a good honest human being. To my good friend and long time partner in music, Mr. Alex Huntly, I give my gratitude for the inspiration he has given me over the last ten years. Another font of inspiration and bizarre ideas is Mr. Peter Clive, who must rank as one of the most remarkable people that I have the good fortune to know. Together with him and the aforementioned Mr. Keane, I thank the entity known as Ooooaargraneeesems for many enjoyble evenings. I also thank Gavin Starks, who has been my friend since the day I came to this University, for the time we have spent talking and playing music together over the years. I also thank Sîan "Scarey Mum" Rowan, Fraser "wee bruv" Rowan and Lucy "Oh Tummies" Ralston for being my second family.

Living in the real and present world is not always pleasant, especially when writing a Ph.D. thesis. For this reason I thank the makers of the following T.V. programs: Babylon 5, Star Trek TOS/TNG/DS9/Voyager, The Bill, The (American) Civil War, The Sweeney, A Very Peculiar Practice and Between the Lines.

I reserve my final bout of gratitude for three creatures outwith the human race - my cats: "The Beast of Stripos" *aka* Stripey (who died in 1994), "Solomutron" *aka* Solomon and "The Sheboratron" *aka* Sheba.

Summary

The aim of this thesis is to explore the application of feed forward neural networks, and other numerical methods, to the prediction and analysis of solar terrestrial time series. The three time series under scrutiny are the sunspot number, the 10.7cm solar flux and the geomagnetic K_p index. Each time series will be predicted and examined on time scales of days, months and years. As the work of the thesis unfolds, new perspectives on the time series of interest will be afforded, fueling the prediction intiatives of the later Chapters. New techniques for analysing time series are proposed and applied, as well as some new methods of using neural networks to make predictions.

Chapter 1 reviews the three main fields of interest. The first field is that of the statistical theory of time series modelling. The basic concepts and terminology are introduced, followed by a review of various time series models and prediction schemes. The more recent topic of neural networks is the second reviewed field. Again the basic ideas are introduced, and the defining equations of feed forward neural networks are stated along with a complete description of the training algorithm known as back propagation. To link these first two fields I suggest how the neural network can be viewed as a statistical time series model. Next, the current understanding of the solar terrestrial environment is reviewed, starting with an overview of solar activity, with particular attention paid to the phenomena associated with the solar cycle. The terrestrial environment is then discussed, focussing on how the Sun and its activity affects the Earth's magnetic field. Finally, a selection of past attempts at predicting solar terrestrial time series are described and discussed.

Chapter 2 is where the analysis of the three time series is documented. The work of this chapter is concerned with providing an impression of matters such as: the accumulation and formatting of the data; the search for periodicities; the nature of any periodicities; the non-stationarity of sunspot number; the stationary aspects of sunspot number; the auto-correlation of the time series; the cross-correlation of the time series, especially in relation to the Sun's influence on the Earth; and the use of wavelet transform in analysing time series. Apart from being of intrinsic interest in itself,

this work provides a familiarity with the data that will directly and indirectly fuel the prediction initiatives of the following chapters.

Chapter 3 is an exploration of feed forward neural networks and back propagation. In this chapter some simple FFNNs performing some simple problems are investigated, as well as the (not simple) training algorithm, back propagation. For several cases it is shown what networks can, and cannot, be expected to do because of limitations of numbers of neurons or the activation function used. It is also shown that back-propagation is not always reliable as a training algorithm, as it sometimes completely fails in training networks to perform tasks that they should be able to perform in theory. An important new method, that of analytic training, is also introduced. This method shows how to "train" a neural network to perform any analytic function, by way of constructing and solving a set of linear equations.

Chapter 4 further bridges the gap between neural network methods and the statistical models of time series. In this chapter, various artificial time series are predicted using neural networks, and in a few cases, analytic training is used to prescribe what the minimum requirements are for a network to be able to predict a given class and order of statistical time series model. The ability to compare theory and practice makes the results of this chapter very interesting. At the end of the Chapter, the problem of delayed prediction is highlighted. Delayed predictions are predictions in which events in the time series (such as peaks or troughs) are predicted late.

Chapter 5 documents the results of using networks to predict the solar-terrestrial time series. First of all, a search of network architectures is performed for each of the fifteen time series: sunspot number, solar flux and K_p index, each in five formats: daily, smoothed daily, monthly, smoothed monthly and yearly. Then the most successful networks from this search are trained further to see if any improvement is possible. After this the possibilities of probing deeper into the future are investigated by iterating networks on their own predictions and by training networks to predict 6 steps ahead. For all the best predictions, uncertainties of prediction and levels of bias are estimated from residual histograms and tabulated in Tables 5.20, 5.21 and 5.22. Finally some attention is paid to the choice of training set.

Chapter 6 explores three new ways of using feed forward to predict time series. The first is the wavelet filtered prediction method, in which a FFNN is trained to predict wavelet filtered time series, given inputs from the original time series. The idea underlying this approach is that, by separating the various time scales of the time series, it is easier for the network to identify the scales on which its behaviour is predictable. The second new method involves using a genetic algorithm to train networks to avoid the problem of delayed prediction. Finally the prospect of using networks that take inputs from two time series simultaneously is investigated.

Chapter 7 draws together the results of the preceeding chapters to provide a more general, and digestible, summary of what has been achieved by the work of this thesis. It also proposes new avenues of research and makes suggestions of how the various methods used in this thesis could be improved upon. To end the thesis a prediction of the maximum of cycle 23 (and 24!) is made.

Appendix A deals with various results which do not belong anywhere else in the thesis. In particular a more general description of analytic training is given, showing how to train *I-H-O* feed forward neural networks.

Appendix B lists and discusses the C computer program used to train the networks.

Pre-Amble

Finally, I wish to give the reader some understanding of the thoughts that have determined the style in which I present this thesis.

I have chosen to write the bulk of this thesis in the first person for two reasons: to highlight ideas and work that are my own; and to remind the reader that, sometimes at least, what is being stated is subjective. The latter is not a "get-out clause" for any mistakes on my part, but has arisen from my frustation in distinguishing fact from opinion in the many scientific documents that I have read to date.

In writing this thesis, I have tried to repeat the most important ideas as often as possible, so that the reader is not required to perform great feats of memory. For similar reasons, I have tried to include references and cross-references where-ever possible. On the odd occassion, I might introduce ideas or concepts before fully defining all the terms involved. This is primarily to put things in context, without the burden of excessive detail. As with many texts, this might require a "two pass" reading of some sections: one pass to obtain a feeling for the work and another to gather the details.

The quotes at the head of each Chapter all have relevance to the material within. In some cases the link may be apparent only to me, but I assure the reader that a link *does* exist.

Commonly Used Abbreviations

NN	Neural Network
FFNN	Feed Forward Neural Network
BP	Back Propagation
GA	Genetic Algorithm
WFP	Wavelet Filtered Prediction
AR	Auto Regressive
MA	Moving Average
ARMA	Auto Regressive-Moving Average

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Nasty" *Jack Regan*

The aim of this thesis is to investigate the prediction of solar-terrestrial time series. Examples displayed in Fig. 1.1 are Sunspot number, 10.7 Solar Flux and K_p index of Geomagnetic Activity. In each case the complexity of the underlying physical system prohibits predictions of the time series by constructing a physical model. Instead predictions of the future of one of these time series must rely solely on the history of that time series and perhaps the histories of other related time series. This approach has been the subject of a whole branch of statistics which first emerged in the 1920s, with the bulk of the ground-breaking work occurring in the 1960s and 1970s e.g. Box and Jenkins (1970). Until the mid 1980s most studies were confined to *linear* models of *stationary* time series (these terms are defined in Section 1.2). This was justified to some extent because many apparently non-stationary time series could be transformed in some way so that they became stationary. However the time series mentioned above are manifestly non-stationary and are not easily reduced to stationarity (see 1.2.3) so most of the "classic" work on time series is not applicable. In the last ten years these statistical ideas have been extended to analyzing non-stationary time series directly, e.g. Priestley (1988), though much of this research is still in progress and so is scattered throughout various journals and proceedings. I hope that this thesis will provide an accessible interpretation of at least some of these ideas, demonstrating their application.

The explosion of interest in *Neurql Networks* in the last decade following the invention of the *Back Propagation* training algorithm by Rumelhart et al. (1986) has provided another method for predicting time series. That is, a Neural Network maybe trained on a history of data to make

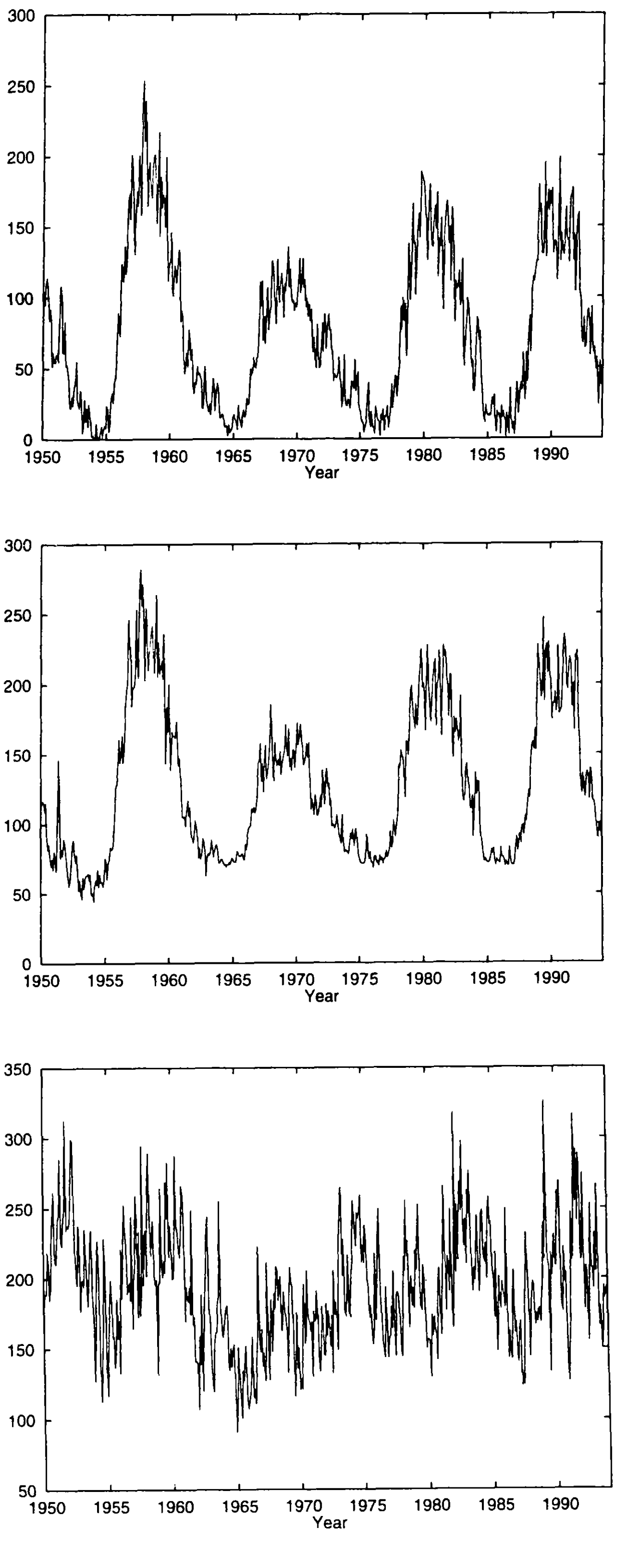


Figure 1.1: Three important solar-terrestrial time series in monthly format over the latter half of the twentieth century: a) Sunspot Number b) Solar Flux c) K_p Geomagnetic Index.

predictions of the future. This method is initially very appealing, not just because of the romantic notion that you are dealing with an electronic brain, but also because it *appears* so much simpler than the theory behind statistical time series modelling. However, these early illusions quickly dissolve when one realizes that a NN can be soberly called generalized non-linear regression and that the supposedly magical action of a network learning is little more than a least squares minimization using gradient descent. The simplicity of the method too is an illusion because there are so many free parameters to be determined, e.g. the number of units or neurons in the network and how they should be connected. If one then accepts a popular class of NNs such as a two-layer feed forward NN trained by standard back-propagation using a momentum term, then there are still equally perplexing decisions to be made: e.g. how many units should be in each layer, how should the data be presented during training and what values of learning rate and momentum parameter are best? Even once a NN has been trained to predict a given time series with some accuracy nagging questions remain unanswered: how is it making its predictions and what do the values of weight connections mean in relation to the time series? In this study I hope to answer these questions, though in the main the answers will be of numerical derivation. I also wish to investigate new ways of making time series predictions with NNs, several new methods are discussed in Chapter 6 which uses a NN to predict components of a time series on different time scales, the components being obtained from a wavelet transformation of the time series. In Chapter 6 I describe a genetic algorithm training scheme and how its flexible nature allows it to cure a problem which can occur when using back-propagation to train NNs to predict time series.

Below I present the list of questions that I shall address in this thesis:

- Which designs of Neural Network make the best prediction of the following time series: Sunspot Number, 10.7cm Solar Flux, K_p Geomagnetic Index?
- On what time scales is it feasible to make predictions of these series and can cross-correlation between them and other time series prove beneficial?
- How should prediction accuracy be judged? What is the criteria for success?
- For multi-layer feed forward NNs, how many layers are needed and how many units are needed in each layer for best prediction accuracy?
- If using back propagation, what are the best values for the learning rate ϵ and the momentum α? Is an adaptive parameter scheme useful?

- What is the best method of data presentation e.g. sequential or non-sequential presentation of patterns? Should the time series be pre-processed in any way, e.g. log or exp transforms?
- Is Back Propagation the best training algorithm or could some other method, e.g. a genetic algorithm, be advantageous?
- How is the NN making the predictions? More specifically, can anything be learned from the state of the weight connections after training is finished?
- How do NNs compare with other methods in terms of prediction accuracy, e.g. nearest neighbour, linear prediction filters, other non-neural, non-linear methods?
- How do NNs relate to the classes of statistical methods in existence?

Whilst I have stated above that the initial buzz of Neural Networks may quickly wear off, over time I have come to realise that the subject is both stimulating and interesting, especially when fused with challenges of time series prediction. Unlike others, I have decided to dispense with the terminology that is intended to separate the mathematical networks from the biological. For example, I will use the term neuron, not unit, and I shall not prefix the words "neural network" with "artificial". I do this for two reasons. Firstly because there is no fear of confusion in this thesis, I shall be exclusively dealing with the mathematical variety of network. Secondly because, in some small way at least, terminology can serve to enthuse and even inspire.

1.1 General Considerations

A time series is defined to be a sequence of values representing the state of some system at a number of different times. In this thesis I shall only consider time series that are recorded at equal time intervals.

Let x_t represent the time series in question and let it extend from x_1, the first observation, to x_T, the most recent observation. By causality x_t can only depend on the values at previous times, $x_1 \ldots x_{t-1}$ and of course previous unobserved values; such a series is sometimes called *non-anticipative*. It is therefore natural to make a prediction of x_t, represented by $\hat{x}_t$, by forming a function of some of the series' history $x_{t-I} \ldots x_{t-1}$ i.e.

$$\hat{x}_t = P(x_{t-I}, \ldots x_{t-1}, t)$$

The time parameter is included explicitly so that, for example, a deterministic mean can be expressed. The residuals, r_t, are then

$$r_t = x_t - P(x_{t-I}, \ldots x_{t-1}, t)$$

Suppose that $P(x_{t-I}, \ldots x_{t-1}, t)$ is a bad predictor of x_t, then the residuals will retain the unexplained structure of x_t. This means the r_t's cannot be independent, so by this reasoning the ideal predictor function, P_0 is such that

$$a_t = x_t - P_0(x_{t-I}, \ldots x_{t-1}, t) \tag{1.1}$$

where $\{a_t\}$ is an unpredictable time series with zero mean and with all the a_t's independently distributed (see Section 1.2). Note that this says that the time series has an "explainable" deterministic component, described by P_0 and an "unpredictable" stochastic component represented by a_t. A further and more aesthetic requirement for P_0 is that in accordance with *Occam's Razor* it should be as simple as possible which amongst other things implies that it should use as small a number of inputs, I, as possible.

The predictor function as defined above assumes that x_t may be written as a function of the previous values of the time series. A more general definition would be

$$P_0(x_{t-I}, \ldots x_t, a_{t-J}, \ldots a_t, t) = 0 \tag{1.2}$$

where the development of x is influenced by the input of the history of another time series a. So that even if the form of P_0 is discovered, extracting a prediction from it is not a trivial task because the values of this other time series might not be readily available. To simplify matters, almost all proposed time series models consider a to be white noise with constant mean and variance.

From here on the *predictor function* is abbreviated to $P(I, t)$.

The problem is now to find $P(I, t)$ and to determine if it satisfies Eqn. (1.1) (or more generally Eqn. 1.2) on all sequences of the observed data $x_1 \ldots x_T$. The larger $T - I$, the easier it becomes to verify that the residuals are independent. Naively one might assume some parametrisation of the predictor function i.e. $P(I, t, \alpha_1, \ldots, \alpha_N)$ and treat the problem as the solution of the set of nonlinear equations:

$$\begin{aligned} P(I, \quad & I+1 \quad ,\alpha_1, \ldots, \alpha_N) &= x_I + 1 \\ P(I, \quad & I+2 \quad ,\alpha_1, \ldots, \alpha_N) &= x_I + 2 \\ & \vdots & \\ P(I, \quad & T \quad ,\alpha_1, \ldots, \alpha_N) &= x_T \end{aligned} \tag{1.3}$$

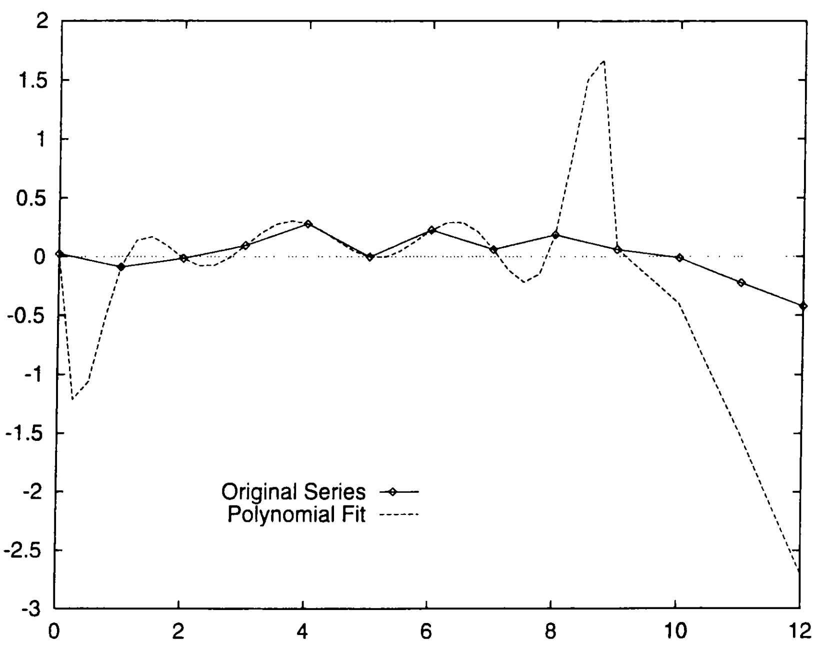


Figure 1.2: A 10th order polynomial in time is fitted to a time series from $t = 0 \ldots 9$. Its predictions are of course meaningless but this simple example illustrates the point that using a predictor function with too many free parameters can fit the observed data perfectly without having any generalization to newly acquired data.

which is a set of $T - I$ equations in N unknowns. By having $N \geq T - I$ *the implicit function theorem* says that it may be possible to solve (1.3) and thus make perfect predictions inside the realm of the known data. This of course will almost certainly fail as a predictor when applied to new data. A simple example using only explicit time dependence illustrates this in Figure 1.2. This is a fundamental problem when using Neural Networks and is referred to as *Generalization*, see Section 1.4.1.

To re-emphasize the most important point:

when searching for $P_0(I,t)$ given the observed data, one must bear in mind that the predictor function is not required to predict the time series x_t but to predict only the deterministic component.

1.2 The Statistical Description of Time Series

In this section I wish to review the established statistical concepts and methods involved in the analysis of Time Series. Since I lacked any real knowledge of statistics at the outset of my Ph.D. study I will start my discussion with a brief review of some basic statistical concepts.

1.2.1 Basic Statistical Concepts

Let a random variable, X, represent some measure of a stochastic process, and let x denote a realization of X or measurement of that process[1] X has a distribution described by its *probability density function* or *p.d.f.*, $p(x)$, which is defined so that

$$\text{Pr. of X in } (x, x+dx) = p(x)\,dx$$

The *expectation* of a function of the random variable X, $E[f(X)]$ is then defined as

$$E[f(X)] = \int_0^\infty f(x)\,p(x)\,dx$$

also the mean of X, μ_X is defined as

$$\mu_X = E[X]$$

and the *variance* of X, σ_X^2 is defined as

$$\sigma_X^2 = Var[E] = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

Quantities involving higher order terms in X, referred to as higher order moments of the distribution, can also be defined but usually a partial description of a stochastic process in terms of its mean and variance is sufficient. If a process is a *Gaussian* or *normal* process then all the moments can be expressed in terms of the mean and variance and therefore the mean and variance are a complete description of the process.

If dealing with two random variables X and Y then further concepts are needed to describe their inter-dependence. The *joint distribution* of X, Y is then given by the *joint p.d.f.* p(x,y) so that:

$$\text{Pr. of } X \text{ in } (x, x+dx) \text{ and } Y \text{ in } (y, y+dy) = p(x,y)\,dx\,dy$$

[1] In this thesis I shall use upper case letters to denote a random variable, and lower cases letters to denote a realization of a random variable. The (unfortunate) exception to this rule is the a_t which is a random variable representing a white noise time series value at time t.

The covariance of X and Y, $Cov[X,Y]$ is defined as

$$Cov[X,Y] = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

and the correlation of X and Y, $Corr[X,Y]$ is defined as

$$Corr[X,Y] = \frac{Cov[X,Y]}{\sqrt{Var[X]\,Var[Y]}}$$

Note that $Cov[X, \pm X] = \pm Var[X]$ so that $Corr[X,Y]$ is between -1 and 1.

Generalizing to N random variables, one can talk about the joint distribution of $X_1, X_2, \ldots, X_N$ and its joint p.d.f. $p(x_1, x_2, \ldots, x_N)$. Each of these variables has its own mean and variance and the notion of covariance can be extended to describe the inter-dependence of several variables at once.

1.2.2 Time Series

Statistically speaking, an observed sequence of a time series, $x_1, \ldots, x_T$, is viewed as as being the realization of T random variables, $X_1, \ldots, X_T$. I shall refer to the random variables, X_t, as being the time series and the observed values, x_t, as being a realization of the time series. The auto-covariance function of the time series is defined to be $Cov[X_t, X_{t+k}]$ and likewise the auto-correlation function is defined as $Corr[X_t, X_{t+k}]$. Each member of the time series is a random variable X_t, which has a mean and variance and these together with the inter-dependences of a sequence of N time series values are completely described by the joint distribution of $X_t, \ldots, X_{N+t-1}$. If the joint distributions of $X_t, \ldots, X_{N+t-1}$ for all possible t's are the same and if this is true for any choice of N then the time series is said to be *strictly stationary*. A natural consequence of this is that the means and variances of all the elements of the time series are the same, i.e.

$$\begin{aligned} E[X_t] &= \mu \\ Var[X_t] &= \sigma^2 \end{aligned} \tag{1.4}$$

where μ and σ are independent of time. Also a strictly stationary time series must have

$$Cov[X_t, X_{t+k}] = Cov[X_s, X_{s+k}]$$

because $X_t, \ldots, X_{t+k}$ and $X_s, \ldots, X_{s+k}$ have the same joint distribution. So that

$$Cov[X_t, X_{t+k}] = \gamma_k \tag{1.5}$$

that is the auto-covariance only depends on the lag k. Notice that $\gamma_0 = \sigma^2$, the covariance at zero lag, is just the variance of the time series.

A less stringent and more practical definition of stationarity is that of *weak stationarity*, where only equations (1.4) and (1.5) are satisfied. In terms of moments of the joint distributions, this can be viewed as stationarity of order 2. If the joint distributions of sequences of a weakly stationary time series are Gaussian then the time series is actually strictly stationary. In general, though,

$$\text{Strict Stationarity} \Rightarrow \text{Weak Stationarity}$$

In what follows, unless stated otherwise, I use the term stationarity to refer to weak stationarity.

A very important but simple time series is the independent *white noise* time series that I shall denote by a_t. It is defined so that each a_t is independently distributed which means that

$$Cov[a_t, a_{t+k}] = 0$$

for all non-zero k. If $E[a_t]$ and $Var[a_t]$ are independent of time then a_t is a stationary white noise time series. It is important to realize that zero covariance does not imply independence because independence is concerned with more than just the second order properties of a joint distribution. Unless otherwise stated I shall use a_t to denote a zero mean, stationary and independent white noise time series.

1.2.3 Practical Implications of Stationarity

The use of the preceding ideas is somewhat limited in the real world because in many cases only one realization of the time series is available. For example, measuring the mean value of the monthly Sunspot number for July 1995 would require the inconvenience of travelling to many parallel universes to obtain a statistical sample. Instead all we have is one "realization" of the Sunspot number time series that extends back to the 18th Century. However, if a time series is assumed to be stationary then one realization of it becomes more akin to a repeated experiment, where the conditions of the experiment remain the same but past outcomes of the experiment may affect future outcomes (this idea re-appears in mathematical form in Section 1.2.6). In this sense, the assumption of stationarity is a great convenience because it means that many statistical quantities relating to a time series, e.g. the mean and variance, can be estimated by time averaging.

Consider three examples of time series:

$$X_t = \sin(\omega t + \phi)$$

$$Y_t = \sin(\omega t + \phi) + \sum_{m=1}^{M} a_{t-m}$$

$$Z_t = \sin(\omega t + \phi) + Z_{t-1} + a_t$$

Clearly none of these time series is stationary because their means depend on time. The question is: can we deduce whether they are stationary or not if given only one realized sequence? X is completely deterministic and when enough observations become available it can be readily identified as being periodic, indicating that a Fourier transform would reveal its precise form. With this complete understanding X can be trivially reduced to a zero and therefore stationary signal. Likewise Y can be reduced to a stationary signal once the deterministic mean component is identified, however if $Var[a_t] \gg 1$, then it may require many more observations over time to identify the form of the deterministic component. Z is random walk with drift and as such is non-stationary even if the deterministic mean is identified and removed. However the time series $Z_t - Z_{t-1}$ is similar in form to Y and so in the same way it can be reduced to a stationary time series.

The above demonstrates the assumptions underpinning the application of what are called "Box-Jenkins" models for time series, which are described in the Section 1.2.4. A Box-Jenkins model is applicable if the given time series is reducible to a stationary time series and if this stationary time series can be described by a linear model. All three of the above time series were contrived so that they could be easily described by a Box-Jenkins type model. However in general there is no reason to suspect that nature will produce time series that conform to such models. In the context of making predictions assuming a Box-Jenkins model can prove to be a serious handicap. To quote an example from Priestley (1988), consider the time series U_t:

$$U_t = a_t + \alpha a_{t-1} a_{t-2} \tag{1.6}$$

This time series has zero mean:

$$E[U_t] = E[a_t] + \alpha E[a_{t-1}] E[a_{t-2}] = 0$$

also

$$Cov[U_t, U_{t+k}] = E[U_t U_{t+k}]$$

$$
\begin{aligned}
&= E[a_t\, a_{t+k} + \alpha a_t\, a_{t+k-1}\, a_{t+k-2} + \alpha a_{t-1}\, a_{t-2}\, a_{t+k} + \\
&\quad \alpha^2 a_{t-1}\, a_{t-2}\, a_{t+k-1}\, a_{t+k-2}] \\
&= E[a_t\, a_{t+k}] + \alpha^2 E[a_{t-1}\, a_{t-2}\, a_{t+k-1}\, a_{t+k-2}]
\end{aligned}
$$

which for $k = 0$ gives the variance:

$$Var[U_t] = \sigma^2(1 + \alpha^2\sigma^2)$$

with zero auto-covariance for all non-zero k. So this time series is not only stationary but indistinguishable from white noise as far as its second order properties are concerned and as such the best prediction that can be made is $E[a_t]$, which is just zero. However, it is clear from the definition of U that the time series has more structure than white noise and therefore a better prediction must be possible. So in this case the Box-Jenkins procedure would be inadequate because firstly it would be easy to mistake U for white noise and more significantly, even if it were realized that some structure did exist, U cannot be described by a linear model. If linear models are unable to describe some stationary time series, such as U, it is clear that more sophisticated non-linear models are needed, especially when dealing with non-stationary time series.

At the beginning of this section I noted a conceptual and practical problem when dealing with non-stationary time series in that only one realization of the time series is available for building a model. It is interesting to phrase this problem as a question in the context of Neural Networks. Consider a network that takes the last 12 months of monthly sunspot number as inputs and is required to predict the sunspot number for the next month. Also imagine that two training sets are available, one containing a complete monthly record from 1850 to 1900 and the other containing a complete monthly record from 1940 to 1990. The question is then, will a NN trained on the more recent data set provide significantly more accurate predictions of monthly sunspot number in 1995? If the answer is yes, then this says that the statistical properties of sunspot number have changed in some way over the last hundred years. Does this mean that the sun has undergone some global physical change in the last hundred years? Dynamo models applied to the sun certainly indicate that this is a physical possibility, as does the (rather uncertain) observational evidence for the Maunder minimum. In Chapter 2 I investigate this question in more detail.

1.2.4 Classical Models for Time Series

Without making some assumptions about the form of the predictor function in Eqn. (1.1) little progress can be made in trying to model or predict time series. In this section I wish to discuss well understood "classical" statistical models that assume that the predictor function is linear and that the time series being modelled is stationary. Due to the second assumption the predictor function must be exactly the same for all sequences of the time series, which means that the form of the predictor function must be independent of time:

$$P(\infty, t) = \sum_{i=1}^{\infty} \alpha_i X_{t-i}$$

where the α_i's are constants. Notice that for convenience here and in what follows I have allowed the predictor function to use the infinite history of the time series, i.e. $P(I,t) \rightarrow P(\infty, t)$.

The time series that this predictor function is to predict is assumed to be of the form

$$X_t = P(\infty, t) + a_t$$

that is, the noise is just added onto the deterministic component.

By introducing the back-shift operator defined so that

$$B\,X_t = X_{t-1}$$

the time series can be written as

$$a_t = \sum_{i=0}^{\infty} \alpha_i B^i\, X_t = H(B)\,X_t$$

where H(z) is a power series defined as

$$H(z) = \sum_{i=0}^{\infty} \alpha_i z^i$$

If $H(z)$ has no zeroes in or on the unit circle in the complex plane then the inverse of $H(z)$ can also be expressed as a convergent power series in z. Assuming that $H(z)$ is invertible, X_t can now be represented in terms of a white noise time series

$$X_t = \sum_{i=1}^{\infty} \beta_i B^i\, a_t = G(B)\,a_t$$

where $G(z)$ is the inverse of $H(z)$.

To summarize, there are two equivalent linear representations of a time series X_t, one in terms of the past values of the time series itself and another in terms of a white noise time series where the white noise time series may be thought of as the residuals from using the linear predictor function to predict previous values of the time series i.e.

$$a_t = X_t - P(I, t)$$

So it is natural to create two classes of practical time series models to accommodate these representations. Firstly, if it is believed that the future values of a time series are heavily dependent on values in the recent past an *Auto Regressive* (AR) model of order p might be appropriate. An AR(p) model is of the form

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + a_t$$

Otherwise it is possible that a *Moving Average* (MA) model of order q might be suitable. An MA(q) is of the form

$$X_t = \sum_{i=0}^{q} \beta_i a_{t-i}$$

These correspond to the truncation of the power series of $H(z)$ and $G(z)$ respectively, which may be acceptable approximations in some cases.

A third class of model, called the *mixed Auto Regressive Moving Average* model, is also widely used, where the ARMA(p,q) model is defined as

$$\sum_{i=1}^{p} \alpha_i X_{t-i} = \sum_{i=0}^{q} \beta_i a_{t-i}$$

The ARMA(p,q) model can be thought of as being equivalent to finding a linear predictor function which is required not to produce a residual time series which is white noise, as above, but to produce a residual time series which is an MA(p) time series.

In the limit of infinite order these three models become three equivalent representations of the same time series (assuming $H(z)$ has a well-behaved inverse). The difference between the three models in practice, where the orders must obviously be finite, is that one model may provide a better approximation to its power series when truncated than the other models. To illustrate this consider a time series of the form

$$Y_t = \phi Y_{t-i} + a_t$$

where $\phi < 1$. Y_t can obviously be described exactly by an AR(1) model. By successive substitution the equivalent MA representation of Y can be shown to be

$$Y_t = \sum_{i=1}^{\infty} \phi^i a_{t-i}$$

which is of infinite order.

The so-called "Box-Jenkins" method of modelling time series can now be broken down into three stages

1. Reducing the time series to stationarity

2. Deciding the most appropriate model and order

3. Finding the parameters of the model

Stage 1 involves transforming the time series using techniques like the ones described in Section 1.2.3 and can justifiably be called the most difficult stage. Stage 2 usually involves some amount of trial and error in determining the best model and as such it can be quite tedious. The final stage is comparatively easy and usually relies on techniques such as least squares fitting or maximum likelihood estimates.

Although these methods rely on a body of solid theoretical results built up by many eminent statisticians throughout this century, the model building process is far from ideal in the sense that it requires guess work, intuition and experience to yield good models. And even then, the two assumptions of stationarity and linearity greatly restrict its applicability.

1.2.5 Non-linear Statistical Models

An obvious way to improve upon these classical models is to use a non-linear predictor function. The need for non-linear models has already been highlighted by the time series defined in Eqn. 1.6, which although (weakly) stationary requires a non-linear model. In this section I briefly review three types of non-linear model, a more complete discussion of each model can be found in Priestley (1988).

The **Bilinear model** is of the form

$$X_t + \sum_{i=1}^{p} \alpha_i X_{t-i} = \sum_{i=0}^{r} \beta_i a_{t-i} + \sum_{i=1}^{m} \sum_{j=1}^{k} \gamma_{ij} X_{t-i} a_{t-j}$$

and is basically an ARMA(p,q) model with cross-talk terms between the history of X and the history of a. Although the bilinear model might appear to be only a slight generalization of an ARMA model, because no powers of X_t or a_t are explicitly included, the work of Brockett (1976) reveals that the bilinear model can actually represent a wide range of non-linear predictor functions. However there are some characteristically non-linear phenomena that it cannot explain, such as limit cycle behaviour.

The **Threshold Autoregressive model** escapes from linearity in quite a different way. It uses the past behaviour of a time series to decide which of a set of linear AR models is most relevant to the time series at the present time. A typical first order model would be

$$\begin{aligned} X_t &= \alpha^- X_{t-1} + a_t^- \quad , X_{t-1} < \theta \\ X_t &= \alpha^+ X_{t-1} + a_t^+ \quad , X_{t-1} > \theta \end{aligned}$$

where θ is the threshold of the model. In this case the predictor function is actually discontinuous i.e.

$$P(1,t) = \alpha^- X_{t-1} - (\alpha^- - \alpha^+) X_{t-1} H(X_{t-1} - \theta)$$

where $H(x)$ is the Heaviside step function. The general form of the threshold model may be written as

$$X_t = \sum_{i=1}^{I^{[j]}} \alpha_i^{[j]} X_{t-i} + a_t^{[j]}$$

where the index j is determined by the recent behaviour of X. Unlike bilinear models, threshold autoregressive models can represent limit cycle behaviour and therefore might be of particular relevance to predicting time series which exhibit cyclic trends. A thorough investigation of these type of models can be found in Tong (1983).

An **Exponential Autoregressive model** is an AR model where the regression coefficients are exponential functions of a previous value of the time series. A typical example of a first order model of this kind would be

$$X_t = (\alpha + \beta e^{-\gamma X_{t-1}^2}) X_{t-1} + a_t$$

so that the magnitude of the last value determines the strength of the autoregression in a more continuous manner than was the case for the threshold AR models.

1.2.6 State Dependent Models

The idea of the *State Dependent Model* or SDM was introduced by Priestley (1980) and provides a generalization which contains all three of the non-linear models in the last section. A key motivation behind the invention of the SDM is that it should not assume an explicit parametrisation from the outset, as was the case with all previous models. An important assumption underlying the SDM is that the predictor function can be written in the form

$$X_t = P(X_{t-I}, \ldots X_{t-1}, a_{t-J}, \ldots a_{t-1}) + a_t$$

which is the same as Eqn. (1.2) except that the explicit time dependence has been removed and now it is assumed that X_t can be written as a sum of the predictor function and a white noise time series. This latter assumption is not trivial but it is a tremendous simplification when attempting to make predictions. It is then further assumed that the predictor function is analytic so that a Taylor expansion may be used to first order about time t:

$$X_{t'} \quad = \quad P(X_{t-I}, \ldots X_{t-1}, a_{t-J}, \ldots a_{t-1}) + a_t$$

$$+\sum_{i=1}^{I} \frac{\partial P}{\partial X_{t-i}}(X_{t'-i} - X_{t-i})$$
$$+\sum_{j=1}^{J} \frac{\partial P}{\partial a_{t-j}}(a_{t'-j} - a_{t-j})$$

by collecting terms of the same time index the above expression can be more concisely written as

$$X_t + \sum_{i=1}^{I} \alpha_i(\mathbf{x}_{t-1})X_{t-i} = \mu(\mathbf{x}_{t-1}) + \sum_{j=1}^{J} \beta_i(\mathbf{x}_{t-1})a_{t-j}$$

where

$$\mathbf{x}_t = (X_{t-I+1}, \ldots X_t, a_{t-J+1}, \ldots a_t)$$

is called the state vector as it determines how a time series governed by an SDM will evolve from time t. Its form is identical to that of an ARMA(I,J) model but with the coefficients depending on the current "state" of the time series, hence its name. It is important to realize that the time dependence arises only via the state vector and *not* explicitly, so that an SDM can be regarded as a stationary model.

1.3 Other Time Series Models

My partitioning between statistical mode'ↄ and other models is rather arbitrary and is only decided by the fact that the former models were invented by statisticians and so have naturally been built around a statistical frame-work. The other models, described in this section, can all be phrased in statistical terms but have arisen from different backgrounds and motivations. They are all designed to predict time series and as such tend to assume a predictor function of the form

$$P(I, t) = \mu(t) + f(X_{t-I}, \ldots, X_{t-1})$$

where f can be non-linear. Notice the difference between this and Eqn. (1.1) in that the explicit time dependence is contained solely within the function $\mu(t)$.

1.3.1 Linear Prediction

Linear prediction is really just prediction using a statistical AR model but I include it here because the following description shows how one might calculate the coefficients of such a model. It also descends on the problem of time series prediction from a slightly different direction than the statistical models, and so is also interesting in that respect.

Given a set of values, $\{x_1, \ldots, x_T\}$, sampled evenly in time, how can some future value x_u be predicted as a linear combination of these values ? Denoting this prediction by $\hat{x}_u$, it is given by

$$\hat{x}_u = \boldsymbol{\alpha}.\mathbf{x}_t = \sum_{i=1}^{I} \alpha_i x_{u-i}$$

where $\mathbf{x}_t$ is the state vector defined previously and I shall refer to $\boldsymbol{\alpha}$ as being the prediction vector. The error of this prediction scheme on the available data can be expressed as

$$\begin{aligned} E &= \sum_{t=I+1}^{T} (\hat{x}_t - x_t)^2 \\ &= \sum_{t=I+1}^{T} (\boldsymbol{\alpha}.\mathbf{x}_t - x_t)^2 \end{aligned}$$

To find the minimum error the derivative of this must be set to zero for all the α_i's i.e.

$$\begin{aligned} \frac{\partial E}{\partial \alpha_j} &= 2 \sum_{t=I+1}^{T} \left(\sum_{i=1}^{I} \alpha_i x_{t-i} - x_t \right) x_{t-j} \\ &= 2 \sum_{i=1}^{I} \sum_{t=I+1}^{T} \alpha_i x_{u-i} x_{u-j} - \sum_{t=I+1}^{T} x_t x_{t-j} \\ &= 0 \end{aligned} \tag{1.7}$$

At this point it is useful to assume that the time series x_t is stationary and that it has had its (constant) mean subtracted beforehand. Armed with this knowledge, the autocovariance, R_{ij} becomes a function of "lag" $k = |j - i|$ only:

$$R_k = \frac{1}{N-k} \sum_{t=k+1}^{T} x_t x_{t-k} \tag{1.8}$$

so that (1.7) becomes

$$\sum_{i=1}^{I} \alpha_i R_{|i-k|} = R_k \tag{1.9}$$

which is a set of I linear equations for I unknowns. The R's can be estimated from (1.8) and then (1.9) can be solved for the α's. However, there is a serious problem to be addressed because the R's are merely sample estimates of the auto-covariance function. For an AR(I) time series to be stationary the (possibly complex) roots of

$$1 - \sum_{i=1}^{I} \alpha_i z^i = 0$$

must all exceed unity in modulus. If for a given AR time series one of these roots lies close to, but outside, the unit circle then it is conceivable that the estimation of the α's might erroneously place this root inside the unit circle which will then cause the prediction method to be unstable. To solve this problem there are many suggested methods of "massaging" the α's, for example see Press et al. (1994).

1.3.2 The McNish-Lincoln Algorithm

The McNish-Lincoln algorithm is based on the assumption that the time series in question is cyclic with additional superimposed variations that are governed only by the recent behaviour of the cycle. Formally such a time series is not stationary, but can be thought of as stationary in the sense that it is becomes stationary once the cyclic trend is removed. Given a stretch of data containing N cycles of I points per cycle the algorithm proceeds as follows:

1. Define the start point of all the cycles e.g. the minima
2. Average over all of these cycles to form a *mean cycle*
3. Subtract the mean cycle from each of the actual cycles, leaving the "residuals" for each cycle
4. The prediction of a point in the cycle n is then constructed using the value of the mean cycle for that point, corrected by a linear combination of the residuals from the last few points in the present cycle, $n-1, n-2, n-3, \ldots$

To clarify this last step, the prediction of the i^{th} value of the n^{th} cycle is given by

$$P_{i,n} = \mu_i + \sum_{m=1}^{M} k_m r_{i-m,n} \qquad i = 1, \ldots, I$$

where the μ_i's are the I values that define the mean cycle and $r_{i,n}$'s are the residuals for the n^{th} cycle. The k's are determined in a least square fashion, in much the same way as the α's were obtained for the linear prediction method. In fact the McNish-Lincoln algorithm is just a form of AR model that performs its regression on the residuals.

The McNish-Lincoln algorithm appears to be restrictive in several ways. Firstly the value of the mean cycle does not enter into the prediction of the residuals directly. This means that the longer time scale variations of the time series, represented by the mean cycle, seem to be de-coupled from the shorter time scales accounted for by the residuals. This might be thought of as a severe problem when dealing with sunspot number because the residuals are noticeably larger near the peak of the cycle. This also means that the residuals for SSN are *not* stationary. Another problem encountered when using this technique is that the cycles must be of the same length. The solar cycle has an average period of about 11 years but the actual length of a cycle can be 10, 11 or 12 years. So to apply the McNish-Lincoln technique some kind of "fix" is required, which may be done, for example, by stretching and shrinking cycles so that they are all of the same length. This method, though, results in further complications when attempting to make predictions. Also, it is difficult to identify the length of a cycle from its rising phase, which would be a necessary pre-requisite for prediction in practice.

Predictions of yearly and three monthly sunspot number made with this algorithm were first documented by McNish and Lincoln (1949) in which estimates of the method's accuracy were given together with a comparison against the earlier prediction method of Yule (1926)[2]. Much more recently Kerridge et al. (1989) used the method to predict various solar-terrestrial data on time scales of months. Also a comparison of this longstanding scheme with NNs has been performed by *Macpherson (1993) with further details in Macpherson (1994). This study found that neural networks could provide a slightly better prediction accuracy that the McNish-Lincoln method.

1.3.3 Nearest Neighbour Techniques

The nearest neighbour method is closely related to the State Dependent Models reviewed earlier. Here the notion of state space is re-employed representing the recent history or state of the time series by its state vector,

$$\mathbf{x}_t = (X_{t-I+1}, \ldots X_t)$$

The prediction algorithm then searches the entire history of the time series, in the form of state vectors, to construct its prediction of X_{u+T} beyond the present state $\mathbf{x}_u$. Once the $k(> I)$ states closest to $\mathbf{x}_u$ have been selected they are then used to form the prediction vector $\mathbf{b}$. Let the closest states be represented by $\mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_k}$ then the prediction vector is

$$\mathbf{b} = (b_1, b_2, \ldots, b_I)$$

such that

$$\mathbf{b}.\mathbf{x}_{t_i} = X_{t_i+T}$$

to as good as an approximation as possible. Notice that this stage can be approached as a linear prediction problem. The prediction vector $\mathbf{b}$ may then be used to to predict X_{u+T}. This technique was suggested and demonstrated by Farmer and Sidorowich (1987) with particular attention paid to the prediction of chaotic time series.

[2]Yule provided much of the early work upon which the classic time series models were based

In the steps outlined above, several issues were swept underneath the carpet. Firstly what determines the "nearest" vector? This is decided by imposing a metric on the state space, naturally providing the concept of distance between vectors required to judge "nearness". But what should this metric be? After all, knowledge of the metric would require intimate knowledge of the state space, which is clearly not available. However, since the vectors of interest are all near each other and, assuming the metric of the state space is locally flat, the use of a Euclidean metric would seem to be an acceptable approximation.[3] Also ignored above was the choice of the number of nearest neighbours, k. There is no easy way to decide this a priori but (drawing some guidance from Takens (1981)) k must be greater than I to obtain the prediction vector and $I \geq D$ where D is the dimension of the strange attractor in state space. Also, why use a linear function to provide the prediction? This issue was addressed by Farmer and Sidorowich (1987) where they considered higher order polynomials for the task and found that this added complexity was of no great advantage, which again may be a reflection of the local flatness of the state space. Finally, what if the time series wanders into some uncharted region of state space? In this case no prediction can be made and perhaps this is more honest than other models which might rashly make a prediction based on the established predictor function. However, this should not be too great a problem for chaotic time series as they will tend to remain near their attractor.

1.4 Neural Networks

A Neural Network is a very general concept embracing a plethora of algorithms and functions, some continuous, some discrete and some binary. For predicting time series I shall only be concerned with NNs that use continuous functions. Rather than attempt to describe Neural Networks in general terms, I describe only one class of NN in detail in this section, the Feed-Forward Neural Network, which I use for the work in this thesis. Recurrent Networks and Radial Basis Function (RBF) Networks are two other popular classes of NN which, although not used in the work to come, shall be described for the sake of completeness. In this section I concentrate on NN properties alone, addressing the issue of predicting time series with NNs in Section 1.5.

[3] I find it strange that Farmer and Sidorowich (1987) do not specify their choice of metric

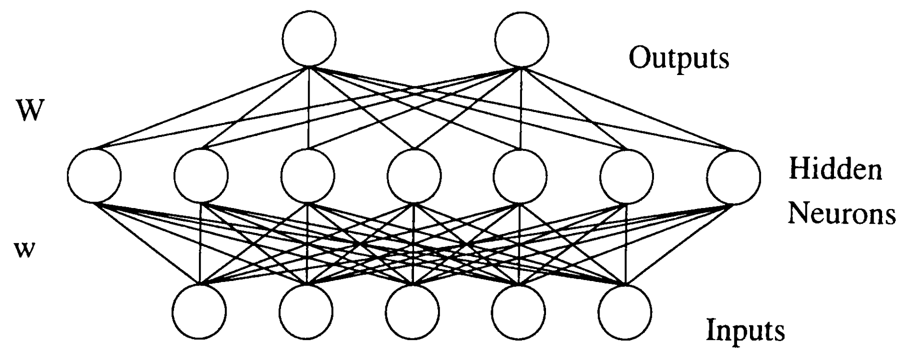


Figure 1.3: A schematic picture of a one hidden layer feed forward neural network.

1.4.1 Feed-Forward Networks

Figure 1.3 illustrates a feed-forward neural network with one hidden layer. The inputs or input neurons, are values $\{A_i\}$ given to the NN from the outside world. The values of the hidden neurons, $\{B_j\}$ are then set according to the equation

$$B_j = g\left(\sum_{i=1}^{I} w_{ji} A_i\right) \qquad j = 1, \ldots, H \tag{1.10}$$

where w_{ij} are parameters called the input weight connections and $g(x)$ is called the input activation function. The outputs or output neurons have their values, $\{C_k\}$, set in much the same way:

$$C_k = G\left(\sum_{j=1}^{H} W_{kj} B_j\right) \qquad k = 1, \ldots, O \tag{1.11}$$

where $G(x)$ is the output activation function and W_{kj} are the output weight connections. The input activation function is usually taken to be sigmoidal (so-called because it looks like a deformed S shape e.g. $\int$), saturating to finite values as $x \to \pm\infty$. This avoids hidden neurons taking on large values which can make training (described later) more difficult. A commonly used activation function is

$$g(x) = \frac{1}{1 + \exp(-2\beta x)} \tag{1.12}$$

where β is a constant, which saturates to 0 as $x \to -\infty$ and 1 as $x \to \infty$. Another common choice is the hyperbolic tangent function:

$$g(x) = \tanh 2\beta x$$

which saturates to -1 and 1. As a matter of convention I will exclusively use the former activation function and for convenience I will take $\beta = \frac{1}{2}$. The output activation function can be a sigmoidal activation function as well but sometimes it is convenient to use a linear output activation function

$$G(x) = x$$

For all the work in this thesis I shall use a linear output activation function. Note that if both input and output activation functions are linear then each output will be a linear combination of the inputs. One reason that NNs are so useful arises from the fact that they are composed of non-linear functions which allows them to be separated into layers[4]. In fact, it can be shown that a one hidden layer feed-forward network can represent any continuous function to an arbitrary accuracy given enough hidden neurons. Further, a feed-forward network with two hidden layers can represent any function with finite discontinuities, again to an arbitrary accuracy if given enough hidden neurons. These last two dramatic statements need some further qualification which can be found in the paper that first proposed them, namely Cybenko (1989). As already indicated, all of the above description can be extended to NNs with an arbitrary number of hidden layers by using updating rules like (1.10) and (1.11) for the extra hidden layers. However, for the reasons given above the use of more than 2 hidden layers seems unwarranted.

The next problem is to set the network's weight parameters so that it performs a certain task, this process is called *training*. To illustrate the training of a network, consider a simple NN designed to learn to add two numbers together. Firstly, a training set is built up, consisting of input-output patterns. Such a pattern is just a pair of numbers (the inputs) and their sum (the desired output). To begin with the NN's weights are set to be small and random, providing a neutral starting point for training. Then a pattern is chosen from the training set and the network's inputs are set to the pair of numbers in that pattern. The difference between the output and the desired output for that pattern is then used to slightly alter all of the NN's weights, by a scheme to be discussed later. This process is then repeated for all the patterns in the training set. I shall refer to one complete presentation of the training set as being one *training iteration*. To complete the training of this summing network typically several hundred or even thousand iterations may be necessary. Training would be stopped once a sufficiently low error is achieved on all patterns in the training set. In general though, for reasons discussed later, this is not a good stopping criterion to use, especially when training NNs to predict time series.

[4] With linear activation functions, an N layer network effectively applies N successive linear transforms to the inputs. N linear transforms are of course equivalent to single linear transform, hence the use of multiple layers serves no purpose.

I have left blank the precise method by which the weights are adjusted in the above illustration because there a number of methods available. I shall describe the most famous method, that of *Back Propagation of Errors*, in some depth and then comment how it can be altered and perhaps improved.

For a general $I - H - O$ feed-forward network[5] each training pattern consists of I inputs and O desired outputs, which I denote by an input vector $\boldsymbol{\xi}^{\mu}$ and a desired output vector $\boldsymbol{\zeta}^{\mu}$, where μ is a label for a particular training pattern. Next an error measure must be constructed that quantifies the accuracy of the NN's outputs with respect to the desired outputs. The simplest such measure is just the square error summed over all the outputs on training pattern μ,

$$E^{\mu} = \frac{1}{2}(\boldsymbol{\zeta}^{\mu} - \mathbf{C}(\mathbf{w}, \boldsymbol{\xi}^{\mu}))^2 \tag{1.13}$$

where $\mathbf{C}(\mathbf{w}, \boldsymbol{\xi}^{\mu})$ is the vector composed of the NN's outputs when presented with input vector $\boldsymbol{\xi}^{\mu}$. The object of training is to try and minimize E^{μ} for all μ. In order to accomplish this it is necessary to calculate the gradient of E^{μ} with respect to each weight parameter. Differentiating (1.13) with respect to an output weight connection yields

$$\begin{aligned} \frac{\partial E}{\partial W_{kl}} &= (C_k - \zeta_k)\, G'\left(\sum_{j=1}^{H} W_{kj} B_j\right) B_l \\ &= \Delta_k\, B_l \end{aligned}$$

where

$$\Delta_k = (C_k - \zeta_k)\, G'\left(\sum_{j=1}^{H} W_{kj}\, B_j\right) \tag{1.14}$$

and similarly for the input weight connections

$$\begin{aligned} \frac{\partial E}{\partial w_{mn}} &= \sum_{k=1}^{O}(C_k - \zeta_k)\, G'\left(\sum_{j=1}^{H} W_{kj} B_j\right) W_{km}\, g'\left(\sum_{i=1}^{I} w_{mi}\xi_i\right) \xi_n \\ &= \delta_m\, \xi_n \end{aligned}$$

where

$$\delta_m = g'\left(\sum_{i=1}^{I} w_{mi}\xi_i\right) \sum_{k=1}^{O} \Delta_k\, W_{km} \tag{1.15}$$

Note that, in the above expressions for the derivatives of E^{μ}, the μ superscript has been dropped for the sake of clarity. back propagation then uses gradient descent to minimize the errors E^{μ}. If

[5] $I - H - O$ means that the NN has I inputs, H hidden inputs and O outputs

Δw^{μ}_{mn} and ΔW^{μ}_{kl} represent the alterations to the corresponding weights after pattern μ has been presented, then they are given by the learning rules

$$\Delta W^{\mu}_{kl} = -\epsilon\, \Delta^{\mu}_{k}\, B^{\mu}_{l} \tag{1.16}$$

and

$$\Delta w^{\mu}_{mn} = -\epsilon\, \delta^{\mu}_{m}\, \xi^{\mu}_{n} \tag{1.17}$$

where ϵ is called the learning rate and in effect determines the step-size of each iteration. The back propagation algorithm is basically the calculation of the delta's using (1.14) and (1.15) followed by the application of the learning rules (1.16) and (1.17). It is the fact that the δ's can be computed in terms of the Δ's that gives rise to the name of the algorithm. The extension of back propagation to a feed forward NN with many hidden layers is easy, as it only involves calculating the delta's for each extra layer in terms of the delta's obtained for the layer above.

There are many possible refinements to the back propagation algorithm and almost all of them aim to replace or improve the gradient descent minimization. These refinements are intended either to improve the stability of the algorithm (i.e. ensure that it does actually reduce the error) or speed up its progress to save on valuable computer time. A useful concept in discussing the minimization involved in training NNs is that of the *error surface* in *weight space*. This can be imagined as the error plotted as a function of the NNs weights (since there are usually many weight connections it is in fact a hypersurface). In reality training has to deal with many error surfaces simultaneously because each training pattern has its own error surface given by $E^{\mu}(\mathbf{w})$. It is implicitly assumed in the back propagation algorithm that it is possible to converge to a minimum on most, if not all, of these surfaces simultaneously. Since a minimization on all these surfaces leads to a minimum of the summed error, $\sum_{\mu} E_{\mu}$, a single *summed error surface* can be defined. At this point I would like to point out that in much of the neural network literature the distinction between an error surface and the summed error surface becomes quite blurred. When discussing training it is quite common for authors to talk about minimization on a single error surface, which presumably means what I have called the summed error surface. However, this is not really correct because all training methods [6] perform minimization on the individual error surfaces. This distinction may seem subtle but to ignore it will overlook the fact that a particular task may be beyond the reach of a NN because its patterns are so arranged that minimization is impossible on all error surfaces at once.

The most common improvement on raw back propagation is the inclusion of a *momentum* term

[6] With the possible exception of a genetic algorithm scheme

in the learning rules (1.16) and (1.17). They then become

$$\Delta W^{\mu}_{kl} = -\epsilon\, \Delta^{\mu}_{k}\, B^{\mu}_{l} + \alpha \Delta W^{\mu}_{kl}[\text{old}]$$

and

$$\Delta w^{\mu}_{mn} = -\epsilon\, \delta^{\mu}_{m}\, \xi^{\mu}_{n} + \alpha \Delta w^{\mu}_{mn}[\text{old}]$$

where α is called the momentum parameter and the [old] label indicates that the weight change is from the last iteration. Momentum effectively smooths the weight adjustments, helping to quash any large fluctuations. Also if the weight changes due to a specific pattern stay roughly constant from one iteration to the next, perhaps because of a constant slope or plateau on the cost surface, then the momentum term leads to a higher effective learning rate. For any weight connection, w, the learning rule takes the general form

$$\Delta w = -\epsilon\, \frac{\partial E}{\partial w} + \alpha\, \Delta w[\text{old}]$$

By assuming that $\frac{\partial E}{\partial w}$ remains approximately constant from one training iteration to the next, recursive substitution using the learning rule yields

$$\begin{aligned} \Delta w &= -\epsilon\, \frac{\partial E}{\partial w}\, (\alpha + \alpha^2 + \alpha^3 + \alpha^4 + \ldots) \\ &= -\frac{\epsilon}{1-\alpha}\, \frac{\partial E}{\partial w} \end{aligned}$$

so that the effective learning rate will be large if $\alpha < 1$. Of course the real reason momentum is used is not because of these back-of-the-envelope arguments, rather it is used because it is found pragmatically to work, in that it does seem to stabilize training and make it more efficient. I demonstrate this in the next chapter, in Section 3.2.

Gradient descent is a reliable minimization method, in the sense that it does eventually converge on a minimum, but it can be painfully slow, especially if the learning rate is too small. Also, as training progresses, the learning rate may need to be reduced to allow for more delicate weight adjustment. So, to speed up training on even ground and to slow it down on rough terrain, an *adaptive parameter* scheme can be used to make the learning rate a dynamic variable - that is it is determined by the current state of training. A typical example of an adaptive parameter scheme might be as follows.

- If the summed error is only gradually decreasing over n iterations then increase ϵ by a factor of k.

- If the summed error increases after any iteration then reduce ϵ by a factor k and repeat the iteration

The obvious problem with this scheme is the selection of n and k and the definition of "gradually". An implementation of such a scheme is given in the next chapter in Section 3.2. A rather involved scheme for determining ϵ and α dynamically is given in Yu et al. (1995), where the authors claim that training times can be reduced by a factor of 10 to 50 over fixed parameter back propagation.

Rather than just tweak gradient descent, more radical improvements have been proposed that replace gradient descent with more advanced minimization algorithms. A concise review of many such schemes is given in Hertz et al. (1991). One which has recently gained popularity is *Conjugate Gradient descent* mainly because it is faster than standard gradient descent. Conjugate gradient descent is based upon a steepest descent line search method, which for a particular pattern changes the weight vector $\mathbf{w}$ according to

$$\mathbf{w}[\text{new}] = \mathbf{w}[\text{old}] - \lambda\mathbf{d}[\text{new}]$$

where

$$\mathbf{d}[\text{new}] = \nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots\right)$$

where λ is effectively determined by taking many steps in the steepest gradient direction until the error (on that particular cost surface) can no longer be decreased in that direction. Conjugate gradient descent improves on this line search by compromising the new search direction, with the old direction:

$$\mathbf{d}[\text{new}] = -\nabla E + \beta\mathbf{d}[\text{old}]$$

where β can be specifically chosen so as to prevent the new direction "spoiling" the effect of the last iteration using the old direction.

Conjugate gradient descent replaces standard gradient descent but still requires the first order gradient information that the back propagation algorithm computes so efficiently. For this reason the term back propagation sometimes only refers to the part of the algorithm that calculates the derivatives, so that training algorithms are often described with phrases such as "Back propagation using gradient descent with a momentum term" or "Back propagation using conjugate gradient descent". There are also some minimization methods that use second derivative information but they can be very expensive in computer time and so I do not discuss them here.

Another problem with training that can affect all of the above methods is the possible presence of other local minima on the cost surfaces. This means that the training algorithm could converge on a minimum which does not correspond to the task it is required to perform. However, any survey of the literature on NN applications, and indeed the results that I present in this thesis, indicate that local minima are rarely encountered. In fact some researchers have voiced the opinion that for some (as yet unexplained) reason local minima just don't exist in the majority of NN applications. It can even be envisaged that the task the NN is expected to perform does not correspond to the global minimum at all, but instead to a local minimum. Such a bizarre situation can be caused by an unfortunate choice of error measure[7]. One other training method which provides a more global search of the error surface uses a genetic algorithm, which is of interest for several reasons, if not only for the fact that it does not rely on the derivatives for minimization. I will discuss the genetic algorithm in general terms in Section 1.6 and again in Section 6.2 when I use it to construct a training scheme.

The whole problem of optimizing the training algorithm is over-shadowed, and ironically partially removed, by the need for generalization in NN learning. In short, it is no longer so crucial to hunt for the exact minima of the training patterns' cost surfaces (if that was at all possible anyway) because the stopping criterion demanded by generalization forbids training from approaching these minima too closely.

In order to elucidate the last rather surprising statement, consider a NN that is designed to perform the rather mundane task of determining the gradient of a line, where the components of the input vector $\boldsymbol{\xi}$ contain the y values of the line at $x = 0$ to $x = 1$. This NN will need one output and only two inputs to perform this task and in effect needs to learn the function $f(\xi_1, \xi_2) = \xi_2 - \xi_1$. Once the NN is trained it will successfully reply with the difference of its inputs. However, it will only do this inside the range of inputs represented by the training set, e.g. it cannot be expected to return the gradient of the pattern $\boldsymbol{\xi} = (0, 100)$ if it was only trained on a range of inputs that went as far as $\boldsymbol{\xi} = (0, 10)$ (this is because of the NN being non-linear and the problem being linear). This is a trivial example of generalization in that the NN cannot be expected to generalize itself to input patterns that were not represented in the training set.

If, instead, the NN is required to determine the gradient of data composed of y values from a line with added noise, i.e. perform a linear regression, then many new problems arise. Firstly two inputs are inadequate. Let's say that the NN is given many inputs and that the input vector now

[7]An example of precisely this problem has become part of the lore of NNs. It is said that were was once a U.S. military project that used neural networks to identify Russian and U.S. tanks from photographs. A network was trained, and its success rate was extremely high. Then a fresh set of photos were introduced, and the network's performance on these was terrible. It turned out that the network had merely learned to distinguish between light and dark - because in the original set of photos, the U.S. tanks were photographed in higher light levels than the Russian tanks.

contains the y values of the line at evenly spaced points from $x = 0$ to $x = 1$. Now imagine training this NN for a fixed number of iterations N, ensuring the error on the training set, E_{Training}, actually decreases with each step. Now if some new examples not contained within the training set (but within the range represented in the training set) are presented to the NN and an error, E_{Test}, is calculated using this "test" set then it will almost certainly be found that

$$E_{\text{Test}} > E_{\text{Training}}$$

and that after some $N = N_0$ the gulf between the two errors will increase gradually as N is increased. At first sight this seems bizarre: the NN is becoming better at the task required of it by one criterion, the training set error, while at the same time its performance is deteriorating according to another, the test set error. This is a classic example of the problems of attaining good generalization. After some point in training the NN starts to learn the noise of the training set, crippling its ability to generalize to examples outside the training set. A simple analogy can be drawn with a child learning French at school. If the child only learns "parrot fashion" certain phrases that the teacher has taught as part of the course, he or she may well achieve a good score on tests but will not fare well if taken to France on a school trip. One way of avoiding this problem is to use a larger training set but often the amount of data is quite limited as is computer time. A more practical alternative is to halt training before the error on the test set begins to increase at $N = N_0$ in the above example. This is why exact convergence on the minima is not needed and is in fact wholly undesirable. In cases involving "noisy" patterns arrival at the exact minima of the cost surfaces corresponds to a precise fit of the training patterns *including* the noise. This is sometimes referred to as *overfitting* and is a factor that influences the optimum choice for the number of hidden neurons. Methods such as this are sometimes called *internal validation* or *cross validation* An alternative method, preferable if training data is in short supply and the luxury of a test set is unaffordable is that which I shall call *incestuous validation*. In this case all the data is to be used as the training set, but training proceeds for one or a few iterations at a time with a subset of the training set removed. This provides a temporary test set on which the generalization ability of the NN can be judged. Then this subset it returned and another subset is chosen and the whole process is repeated until the errors on these test sets begin to rise. In this way training can be "validated" with least sacrifice for a test set. A more abstract description of generalization is given in Hertz et al. (1991) and references therein.

1.5 Predicting Time Series with Feed-Forward NNs

Given a time series of data, x_t, NN prediction proceeds by presenting I time series values as inputs to the NN and expecting the NN to reply with prediction of future values at its outputs. Therefore to construct the training set requires assembling the vectors

$$\xi^{\mu} = (x_{\mu+1}, x_{\mu+2}, \ldots, x_{\mu+I}) \qquad \mu = 0, 1, 2, \ldots$$

and their corresponding outputs

$$\zeta^{\mu} = (x_{\mu+I+p_1}, x_{\mu+I+p_2}, \ldots, x_{\mu+I+p_O}) \qquad \mu = 0, 1, 2, \ldots$$

where p_i is the *predict ahead time* of the i^{th} output, where $i = 1, 2, \ldots, O$. So, given the input vector ξ^{μ} the NN is required to respond with the output vector ζ^{μ}. Of course a sizeable number of such input-output patterns are also needed to form a test set. The NN can then be trained as described before with the training and test set error both being monitored to ensure that training is stable and that overfitting is not occurring.

1.5.1 Choices

Before training can commence the parameters of the training algorithm need to be chosen. If back propagation using gradient descent with a momentum term is employed then the learning rate ϵ and the momentum α need to be chosen. Also one has to decide whether the patterns should be presented sequentially or randomly. These choices are not easy to make, and really have to be made using experience and trial and error. As indicated above the choice of learning rate can be avoided by introducing an adaptive scheme but this can introduce even more parameters that also need to be chosen. If such adaptive schemes are to be pursued justification must sought in that the scheme is fairly insensitive to the choice of these new parameters. In any case training is a means to an end and as long as this end is reached within a reasonable amount of time it is not too important that the training algorithm was not optimal[8].

In addition to all the choices that need to be made concerning the the training algorithm, there are many more important questions that need to be answered with relevance to the actual prediction of the time series,

1. How many inputs are required to make a prediction of p time steps into the future?

[8]This is not a valid excuse when the NN needs to be trained "on-line" using new data as it becomes available.

2. Should a NN be devoted to predicting with only one output?

3. If predicting several time steps ahead, is it better to do so with a NN performing that prediction directly or is it better to do so in an iterative manner using a NN with an output predicting at one time step ahead? In the second case, the NN's predictions are used as subsequent inputs.

4. Is there a sampling or smoothing of the data that would favour the predict ahead time required? After all the daily variations of sunspot number might be more of a hindrance when predicting a year ahead.

5. Is there some other transform of the data, e.g. logarithmic or exponential, that might help in predicting the time series?

The answers to the above questions obviously depend on the nature of the time series under scrutiny. The answers are also dependent on one another, so that for example the answer to question 1 depends on how questions 4 and 5 were answered. This means that it is very hard to come to definitive answers on any of these questions by numerical experimentation, even more so because each "experiment" might require the training of many networks, which is very costly in computer time. However, the reward for exploring these possibilities is two-fold: firstly it will show how to improve prediction accuracy and secondly it can shed some light on the nature of the time series. For example, in attempting to find the best way to predict sunspot number one year ahead, it may be discovered after many trials that a NN taking 18 inputs of monthly sunspot number with one output predicting 12 months ahead is best for the task. In doing these trials the prediction method has been improved and it is seen that only time scales greater than a month are relevant to predicting the solar cycle on the time scale of one year.

Much of the effort behind this thesis has therefore been devoted to just this kind of work - optimizing and exploring the NN algorithm. In this context, I remain sober in the sense that I do not view the NN as being a magical device that is superior to all others in predicting time series. Rather, I view a neural network as a very flexible tool with which I can explore the predictability and nature of solar-terrestrial and other time series.

1.5.2 Neural Networks as Time Series Models

NNs can be classified along with the time series models that I have reviewed at the start of this chapter. The predictor function for the k^{th} output of a single hidden layer feed forward neural network can be written as

$$C_k = G\left(\sum_{j=1}^{H} W_{kj}\, g.\left(\sum_{i=1}^{I} w_{ji}A_i\right)\right) \qquad k = 1,\ldots,O$$

by combining (1.10) and (1.11). Obviously this is a non-linear function parameterized by constants which do not depend explicitly on time. In this sense a feed forward NN is a *non-linear* but *stationary* auto-regressive time series model. Note that this does not mean that NNs cannot represent non-stationary time series, in fact even an AR(1) model can represent some very specific non-stationary time series. Stationarity with reference to a model only requires that its form has no explicit time dependence. However, there are undoubtedly many types of non-stationary time series that certain classes of NN cannot hope to describe, another issue which I explore in this thesis.

1.5.3 Other Classes of Networks

The feed-forward neural network is probably the most commonly used network in current applications. However there are several other designs of network gaining popularity at present which may have some advantages in certain applications. An extension of the feed-forward network is the recurrent network which allows weight connections to connect *any two* neurons together. The training of these NNs can be performed by an amended form of back propagation given in Hertz et al. (1991). One reason why this might be an advantage in predicting time series was suggested by Connor et al. (1994). Allowing connections from the output layer to the input layer can allow the errors on the outputs or some function of the errors on the outputs to be effectively fed back into the network as inputs. This is analogous to extending the AR model to the ARMA model, in that the feed forward network has a predictor function of the form $P(x_{t-I},\ldots x_{t-1})$ which is the same as in (1.1) while the recurrent network has a predictor function of the form $P(x_{t-I},\ldots x_{t-1}, a_{t-J},\ldots a_{t-1})$ which is closer to the more general form in (1.2).

Another quite different class of NN is the Radial Basis Function network which uses a different style of learning. Until now I have only been referring to one kind of learning, that is *supervised* learning where the NN learns by attempting examples and being corrected for its mistakes. *Unsupervised learning* only involves a NN learning differences in its inputs without any external judgement of what is right or wrong. In this sense a NN becomes a kind of non-linear statistical discriminator. The Kohohen Map, described in Hertz et al. (1991) is one such example of an unsupervised NN. The radial basis function NN has one hidden layer and feed forward connections but differs from a conventional feed forward NN in two respects. Firstly the input activation functions are Gaussians

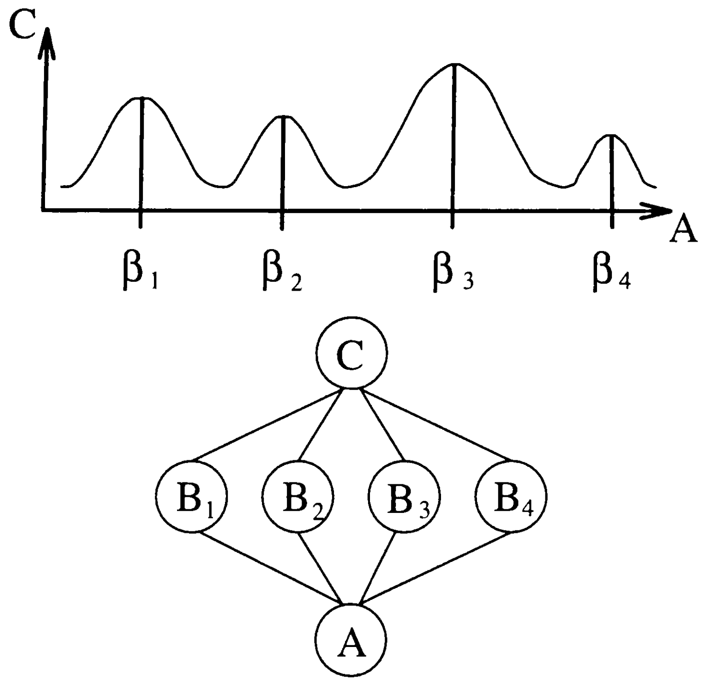


Figure 1.4: A simple radial basis function NN. The graph shows how the output C is a summation of Gaussian functions of A, where the four Gaussians correspond to the hidden neurons B_j.

and secondly the input weight connections are trained using an unsupervised method. Figure 1.4 shows a radial basis function network with only one input and one output. The hidden neuron values are given by

$$B_j = g(-\alpha_j\,(A - \beta_j))$$

where the α's and β's are effectively the input weight connections and the activation function g(x) is a Gaussian i.e.

$$g(x) = \exp(-x^2)$$

By using an unsupervised learning scheme it is possible to arrange the widths and positions of the Gaussians so that they do not overlap appreciably. Once this is the case an input will tend to only "activate" one of the hidden neurons. The hidden neurons can then be connected to the output neuron as follows

$$C = \sum_{j=1}^{O} B_j \zeta_j$$

where ζ_j is the desired output for input $A = B_j$. This will obviously produce correct outputs for certain input values corresponding to the peaks of the Gaussians but to produce correct outputs in the overlapping regions between the Gaussians requires both unsupervised training on the α's and β's and supervised training on the output weight connections. This description can easily be extended to apply to a radial basis function network with several inputs and outputs. Since the radial basis function use Gaussian input activation functions, with only one hidden layer they can actually represent functions with discontinuities to an arbitrary accuracy given enough hidden neurons. As stated before a feed-forward NN needs 2 hidden layers to achieve this, so that in this respect radial basis functions offer significant reductions in complexity.

1.6 The Genetic Algorithm

The genetic algorithm is a general technique inspired by the *survival of the fittest* philosophy suggested by Darwin. Again I will illustrate its use with a simple example and then describe it in more general terms. Imagine a safe that can only be opened by typing in a four digit number. Also imagine that you have a thief's tool that allows you to rate how close a particular number is to the correct number by giving you an integer number from 0 (completely wrong) to 10 (the safe opens). A genetic algorithm will open the safe for you as follows:

1. Try 20 random four digit numbers

2. Record the rating for each one
3. Keep the five best numbers and discard the rest - these are the parents
4. Create fifteen new numbers from the parents by taking the first two digits from one parent and the last digits from another e.g. parents 2157 and 8923 will produce 2123. These new numbers are the children.
5. Randomly change some of the children's digits
6. Try these 20 numbers and return to 2 until safe is opened

Of course the thief would need to be quite patient and brave as there is no guarantee of how fast the algorithm will converge on the answer or even if it will converge on the answer. In fact by performing tests on the safe-cracking scheme above, I found that, on one occasion the thief had to wait for 3,300 generations before he could plunder the contents. On another occasion he had to wait for only 29 generations.

The more general approach to the genetic algorithm introduces the notion of a *string* represented by $\mathbf{S}$ and a problem whose solution can be coded as a string, $\mathbf{S}_0$. The purpose of the genetic algorithm is to seek this solution string. For the algorithm to be effective there must be some rating of a particular string $R[\mathbf{S}]$ so that one string can be "nearer" to $\mathbf{S}_0$ than another string. Once this is done the genetic algorithm can be generally described as follows

1. Create a population of N strings
2. Rate each one
3. Sort the strings and keep P *parent* strings, discarding the rest
4. Create $N - P$ *child* strings by *breeding* the parents
5. *Mutate* the child strings
6. Return to 2

In this more general phrasing of a genetic algorithm higher rated strings are only more *likely* to become parents. Also the process of *breeding* is generalized allowing for other ways of combining parents to produce children or even allowing for the mating of more than two parents. The process of mutation can take on many forms but it is usually a random operation included in order to maintain diversity in the population. Although there is no definite proof of convergence for the genetic algorithm, it has shown itself to be remarkably robust and successful both in applications and nature. Through experience in using genetic algorithms in many problems I have found that its success is not too sensitive to the choice of N, P and the other details of the algorithm. This fact combined with the freedom in choosing the rating of the strings makes the genetic algorithm both easy to implement and very flexible as I shall demonstrate later when I use it to train NNs.

1.7 The Sun and its Influence on the Earth

It was established by Heinrich Schwabe in 1826 that the number of spots on the Sun varied with a cycle of about 10 years. In 1848 Rudolf Wolf realized the importance of Schwabe's discovery and set about reconstructing sunspot number records back to 1700. Daily numbers could only be found back to 1818, monthly means to 1749 and only annual means existed as far back as 1700. For this reason Sunspot number records before 1850 are viewed as being somewhat unreliable. Wolf also began a program of observation at Zurich which developed into a world wide effort which is still maintained to this day. Wolf's system for expressing sunspot number, R, sometimes called *Wolf's Sunspot Number* or the *Zurich Sunspot Number* is as follows:

$$R = k\,(10\,g + f)$$

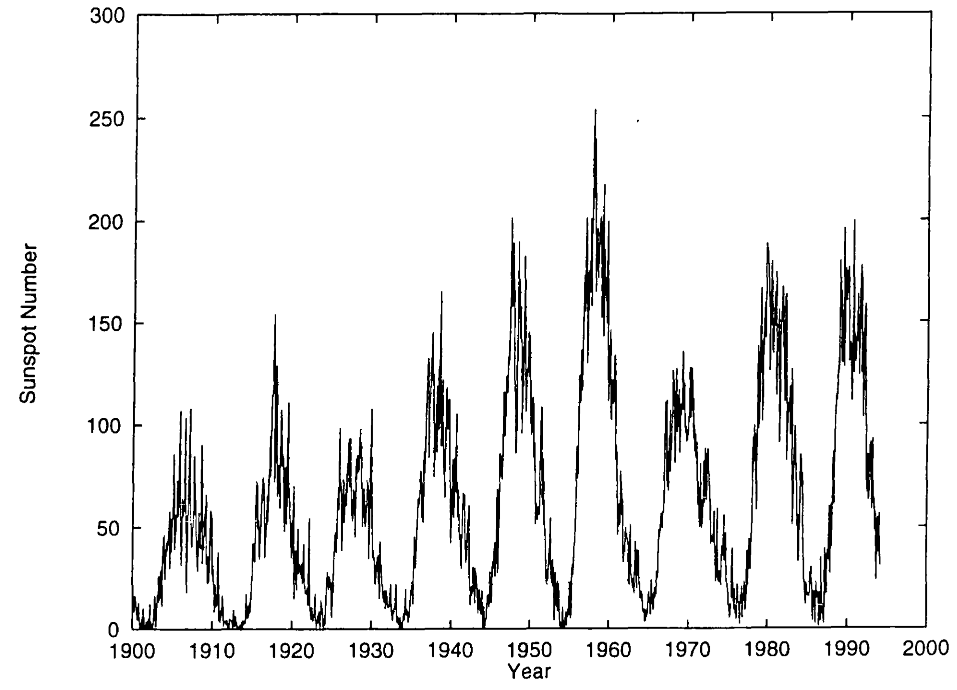
where g counts the number of Sunspot groups and f is the count of the number of individual sunspots. The factor k was introduced to allow a standardization of R for the different opinions on what constitutes a group or even a spot. The Zurich Sunspot Number (SSN) is a daily index and is often expressed as a (calender) monthly mean or as a yearly mean. Figure 1.5 shows unsmoothed and smoothed SSN, where the form of smoothing that I have used is the 13 month running mean defined as:

$$R_i^{\text{smooth}} = \frac{1}{13}\sum_{j=-6}^{6} R_{i-j}$$

Note that sometimes the 13 month running mean is defined as

$$R_i^{\text{smooth}} = \frac{1}{12}\left(\sum_{j=-5}^{5} R_{i-j} + \frac{1}{2}(R_{-6} + R_6)\right)$$

because in this way annual variations in the data are properly accounted for. Since the Earth's orbital period has little impact on the number of sunspots it is the former formula that should be used to smooth SSN, whereas if K_p index (see below) is to be smoothed the latter formula should be employed. Prediction of Sunspot Number in any of its forms has been a "Sword in the Stone"



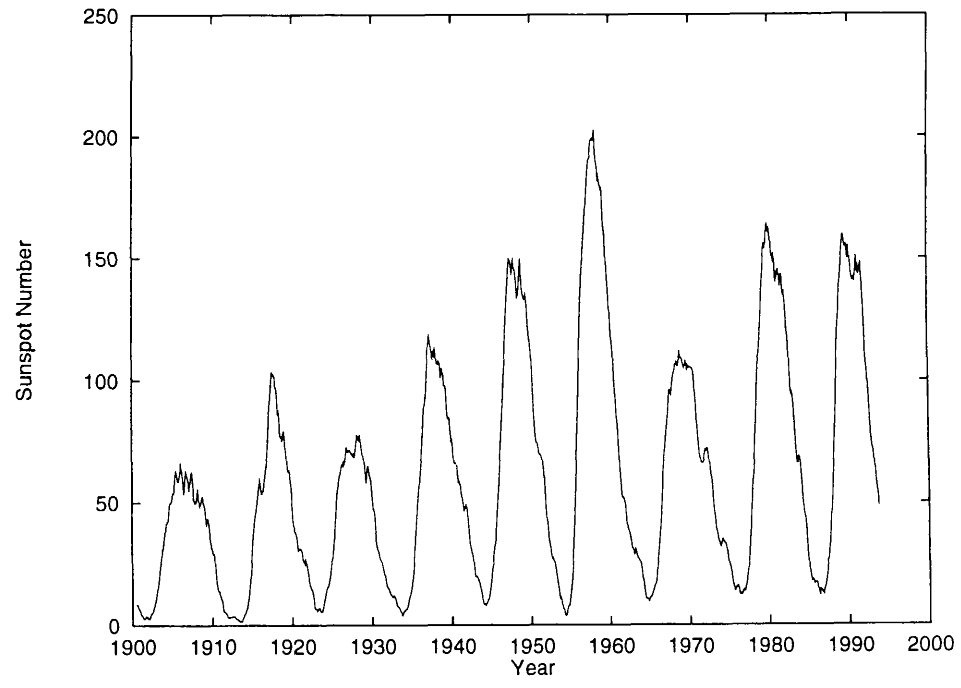


Figure 1.5: a) Unsmoothed Monthly Sunspot Number b) Smoothed Monthly Sunspot Number.

problem for many years now, often being used as a test-bed for new prediction schemes[9].

At the time of writing this thesis the Sun is entering its 23 cycle. By definition a cycle is defined to begin at Sunspot minimum, with cycle 1 commencing in 1761. Observations in the last two hundred years show that the cycle period varies between 8 and 14 years. It also has a variable amplitude, with a maximum recorded SSN of 217 occurring during the solar maximum of 1959.

Towards the end of the 19th century geomagnetic observations, initiated by Gauss, indicated that the magnetic field of the Earth also showed a variation with a similar period but a slightly different phase to that of the SSN cycle. It was an inescapable conclusion that the sunspots were related to some influence over the Earth's magnetic field; but quite how the number of dark patches on the sun related to Earth-bound phenomena was a mystery. The result of this was the birth of many myths which used the "magical" influence of sunspots as explanations for all manner of things Earthly, from the crop yields to the physical and mental well-being of individuals. Even today scientific links are made between solar activity and many terrestrial phenomena. The most famous example is of course the aurorae providing beautiful displays at Northern Latitudes as the Earth's magnetic field interacts with streams in the solar wind. More tenuous links have also been made with the climate in Friis-Christensen and Lassen (1991), with fish populations in Davydov (1986) and even with the sexual behaviour of Crangon Crangon (shrimps).

At present there are almost 200 magnetic observatories around the world, each one equipped with a magnetometer that measures the Horizontal field strength, H, the Vertical Field Strength, V and the direction of compass North with respect to true North, D. H and V are normally quoted in units of μT (micro Tesla) and D is usually quoted in degrees. During periods of little geomagnetic activity the values of H, V and D measure the Earth's dipole field, which is generated by electric currents in the Earth's fluid core. But the Earth's dipole field cannot account for the rapid changes in magnetic field that are observed during more active periods. These must be due to electric currents in the ionosphere and magnetosphere. One such current that results in a gentle, regular diurnal variation is induced by the atmospheric tides which are caused by solar heating and to a lesser extent by the gravitational affect of the moon. These are called the solar quiet (S_q) and lunar quiet (L_q) variations. There are also 27-day periods which are connected to the solar rotation period, which are attributed to the presence of streams in the solar wind that form from material leaving certain regions of the Sun. There is also an unexplained 6 month oscillation which reaches its maximum at the two equinoxes.

[3] which is surprising in some ways as the physical mechanism which governs the solar cycle is still poorly understood in quantitative terms.

In order to summarize the magnetic variations at a particular observatory over a three hour period, the geomagnetic K index is used. This takes the greatest disturbance in H or V over the last three hours and rates it on a quasi-logarithmic scale from 0 (no disturbance) to 9 (very disturbed). This scale is also sub-divided into 3 bands per number by suffixation with a $+$ or $-$. The planetary index, K_p, is obtained by averaging the K values from each station where each station will receive some sort of weighting, like the k parameter for SSN. The shortest time scale of interest in this thesis is one day, so from here on, unless otherwise stated, the K_p index should be understood to mean the daily *sum* of the eight three-hourly values. Note that the monthly and yearly K_p indices are formed by *averaging* the daily K_p values for each month and year. Figure 1.6 shows a sequence of K_p index together with the smoothed and unsmoothed monthly means back to when records began in 1932. A variation of the K_p index, is the A_p which is measured in the same fashion but scaled linearly rather than logarithmically. Notice the presence of a strong 27 day period in the daily data, corresponding to the solar rotation period. There is also another geomagnetic activity index called the *antipodal* or *aa* index. This is a combination of values from only two observatories, one in England and one in Australia. The *aa* index has the advantage that it has been continuously measured since the 1880s.

Highly correlated with SSN but only available since 1947 is the *10.7 Solar Flux* (SF). The correlation can be well described by a linear equation:

$$SF = 0.915 * SSN + 59.916$$

The SF is plotted in Figure 1.7, in smoothed and unsmoothed monthly form. It is noticeably "smoother" than SSN and is regarded as being a superior measure of solar activity in that it can be objectively measured.

1.7.1 Solar Activity

This section is only intended to give a brief description of the various phenomena that are collectively known as *solar activity*. They are all associated with the release or transport of energy in the photosphere, chromosphere and corona. In some regions convective energy flow in the photosphere is suppressed, causing cool regions, which are seen as sunspots. In other regions intense brightening is observed at many wavelengths indicating the release of large quantities of energy, these are termed *flares*. Further out in the corona bright convex arcs of expanding material are seen, these being symptomatic of a *Coronal Mass Ejection* (CME), where the bright leading arc is interpreted as a shock front formed by the fast moving ejected material. These are just three of the most striking

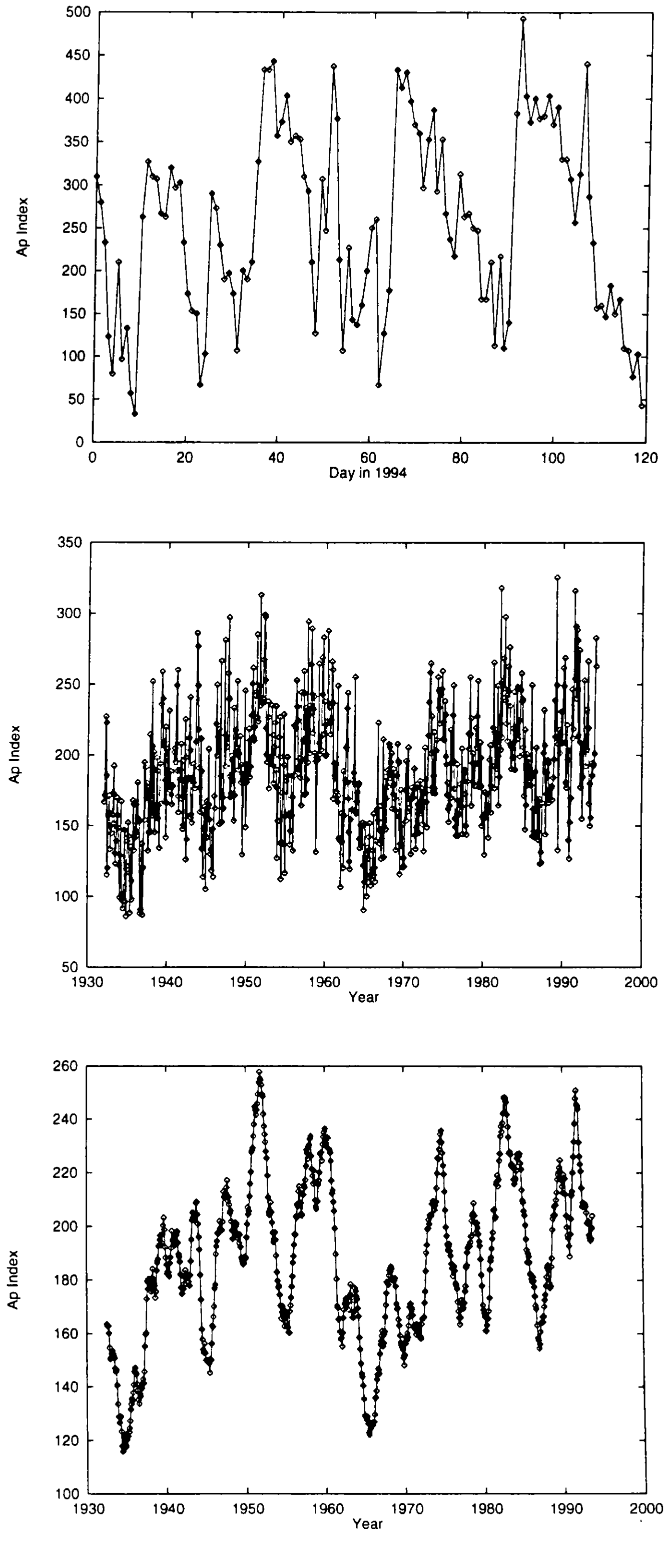


Figure 1.6: a) Daily K_p index during 1994 b) Unsmoothed Monthly K_p c) Smoothed Monthly K_p.

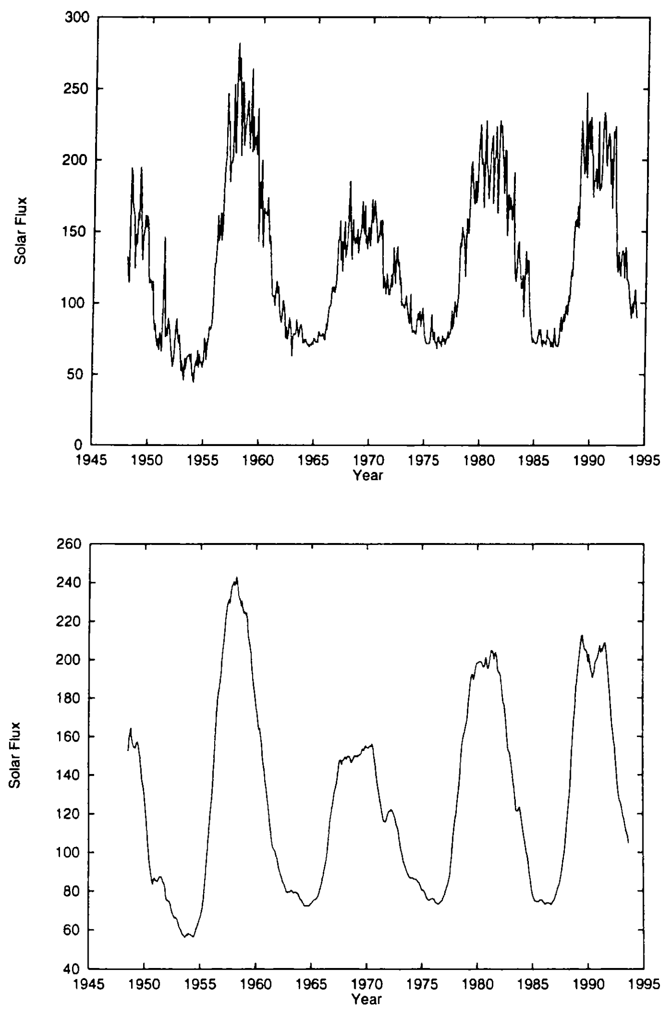


Figure 1.7: a) Unsmoothed Monthly Solar Flux b) Smoothed Monthly Solar Flux.

and immediate manifestations of the exchanges of energy, whether it be kinematic or magnetic, that take place on the sun.

Sunspots usually appear in groups or pairs and are not evenly distributed over the solar globe. At any given time certain regions of the Sun are more likely to develop sunspots that others. In Liggett and Zirin (1985) it is found that sunspots are 10 times more likely to appear in a region of given area near an active region than in an equal sized area of little activity. Also sunspots are always found in a band near the solar equator, the size of this band depending on the phase of the cycle (this is called Sporer's Law). The band is as wide as $\pm 40^{\circ}$ just after SSN minimum, at the start of the cycle, and as narrow as $\pm 10^{\circ}$ at the end of the cycle. The transition from wide to narrow is gradual throughout the cycle but the return from narrow to wide is sudden, producing the famous butterflies on the Maunder Sunspot diagram, which can be found in many books about the Sun. e.g. Zirin (1987). The farting pencil dithered on the spot for a moment until it gradually began to realize that it was actually safe to proceed. Spots often form in pairs, with the two spots being referred to as the leading spot, because it leads in the sense of the solar rotation, and the following spot. The leading spot appears first, with the following spot almost always forming closer to the equator. As the spots evolve they separate in distance, both moving away from the equator. As this proceeds, the leading spot catches up in latitude so that the pair gradually align themselves with the parallels of latitude as they grow older. During the separation smaller spots can appear in between the pair so that a spot group is formed. The leading spot can usually be identified in the group because apart from leading, it is usually quite circular in shape, whereas the other spots including the following spot can be quite irregular. These spot groups evolve in a matter of days and persist for up to a month, with the leading spot being the last to disappear. Having said this, a spot does not usually disappear immediately. Rather it disintegrates, breaking up into many smaller spots that gradually fade away.

The amount of spots on the sun can also be quantified by the area of the Sun that is covered with spots. At any one time no more than 1% of the disk is covered with spots. As might be expected there is a correlation with SSN:

$$\text{Spot Area (Millionths of the Solar Disk)} \sim 16.7\,R$$

However, the correlation is not very good especially in view of the fact that Sunspot Area maximum occurs about 2 or 3 years after Sunspot Number maximum. Since flare activity peaks at about the same time as Sunspot Area, it can be argued that sunspot area is a better indicator of solar activity.

Sunspots are not just black disks, but are often quite irregular in shape and are not uniformly dark. The darkest region is called the *umbra* and is at the centre of the spot with a less dark region surrounding the umbra called the *penumbra*. There is finer structure than this, especially when the spot is viewed in $H\alpha$, where bright flecks, radial filaments and flows can be seen in and around the penumbra. Sunspots also have strong magnetic fields, first reported in Hale (1912). In Hale et al. (1919) it was found that the following and leading spots had opposite polarities. Also if the leading spots had positive polarity in the northern hemisphere of the sun then the leading spots of the southern hemisphere would have negative polarity (the Hale-Nicholson Law). It was subsequently established that the situation would be reversed for spots born at the start of the next solar cycle. The spot's magnetic field is strongest in the middle of the umbra and is also normal to the surface of the Sun there. Further out from the centre, the field becomes weaker and is more inclined to the vertical.

Of course the above description is very general and often exceptions to the rules are observed, sometimes there is no following spot or sometimes groups evolve without any clear leading or following spots.

Active regions are comprised of a whole cast of phenomena, sunspots being just one player. Any region that shows any combination of flares, sunspots, plages (bright patches), filaments or loops is referred to as an active region. All of these features can be ascribed to magnetic field formations, either suppressing or encouraging the transfer of energy in some way. For example a suggested explanation of sunspots is that strong vertical magnetic fields below them in the photosphere suppress the convection of heat from the bottom of photosphere to the surface. This is because convection requires a horizontal flow at the base of the photosphere which, in the case of sunspots, is difficult because the plasma would have to flow at right angles to the vertical magnetic field. Flares are attributed to releases of energy due to sudden changes in the magnetic field, e.g. in magnetic reconnection. The strong magnetic fields which are present during reconnection can accelerate electrons and ions, which can go on to produce enhanced thermal emission in Soft X-Rays and impulsive, non-thermal emission in the form of Hard X-Rays and Gamma Rays, resulting in the release of energy into light and particle kinetic energy that is known as a flare. Flares are associated with active regions and are therefore most common and also most violent just after SSN maximum.

In the **Corona** many phenomena are observed that can be linked with activity at the Earth. This is not surprising because the corona is really the start of the solar wind, so that any material disturbance there will propagate out with the wind. The most dramatic of these is the *Coronal Mass Ejection* (CME) where large "bubbles" of plasma are thrown out of the corona, expanding as they do so. At the leading edge of CMEs is a shock front caused by compression of the overlying and relatively slowly moving coronal plasma. So photographs and movies show a very bright arc, indicating the outer edge of the CME. Time sequence photographs (or coronagraphs) can be found in Zirin (1987) from the ill-fated P78-1 spacecraft that was shot down by the U.S. Air Force shortly after making the observations included in Zirin's book. The material in a CME will take two days to reach the Earth's orbit and if it actually hits the Earth it can be blamed for heightened geomagnetic activity. The frequency of CMEs is quite surprising as they can occur as often as once every 24 hours expelling as much as $10^{13}\,kg$ of solar material each time. This means that they can carry away as much mass per unit time as the solar wind, on average.

In Soft X-Ray images of the corona there are dark regions where coronal material appears to be absent. It has been inferred that in these regions the magnetic field lines open out, allowing plasma to leave the sun. *Coronal holes*, as they are called, are often observed at the solar poles hinting at a poloidal field that may have existed at some point in the cycle (see below). Bartels suggested in Chapman and Bartels (1940) that there existed "M-regions" on the Sun that caused the cyclic 27-day disturbances apparent in stretches of geomagnetic activity. After Soft X-Ray images were obtained by *Skylab* in the 1970s these mysterious M-regions were identified as being coronal holes. Their effect on the Earth is attributed to streams of material flowing along the open field lines which are "anchored" to the surface of the Sun, causing the streams to sweep by the Earth with a roughly 27 day period.

The Sun does not have an obvious **Global Magnetic Field** throughout its cycle. Though the polar coronal holes provide tantalizing evidence of a dipole field, the magnetic structure between the poles is usually extremely complicated and not dipole-like at all. I say "usually" because at Sunspot minimum there are suggestions of a global poloidal field in the inner corona, evidenced by the presence of streams of material following along poloidal paths. However, there is certainly suggestion of global structure in the field throughout the entire cycle, evidenced for example in the Hale-Nicholson law concerning sunspot polarities. Also at the end of each cycle (at the minimum) the polarity of the Sun's magnetic field suddenly flips so that polarity of the open magnetic field lines at the poles reverse. This behaviour appears to be consistent with a *Solar Dynamo* model, where the flows of the plasma due to solar rotation and local effects both contribute to a complicated re-generation of the solar magnetic field. This picture starts with a poloidal field just below the photosphere at the beginning of the cycle. Since the conductivity there is so high, the field lines are frozen into the plasma and carried around with the solar rotation. Differential rotation causes a twisting of the field, with field lines near the equator being pulled further ahead, until an almost toroidal field is formed (i.e. the field lines are parallel to lines of constant latitude). The vertical and horizontal convective motions cause large pockets of plasma to undergo twisting due to the Coriolis force. This sets up toroidal currents which re-generate the global poloidal field but with opposite polarity to that of the previous cycle. In this way the 22 year magnetic cycle is explained. Also explanations for active regions are offered, in that, during the twisting of the toroidal field (which occurs at solar maximum), bundles of twisted magnetic "flux rope" become buoyant and float to the surface of the photosphere causing the phenomena associated with active regions. Other effects such as the Hale-Nicholson law, Sporer's Law and sunspot polarity reversal have all found explanations stemming from the basic dynamo picture. However the picture is rather qualitative in that an MHD theory has not yet been developed that can convincingly describe the re-generation of the global magnetic field and the complicated local phenomena simultaneously.

The **Solar Wind** which begins at the corona has a dramatic effect upon the Earth's magnetic field and can be detected as far out as 40 AU. The Sun emits the wind in geometrically the same way a rotating garden sprinkler emits water, with the material travelling radially once it has left the sun. However, if expelled material is traced from one region of the Sun it will be found to be moving outward in a spiral fashion. Since the plasma leaving the Sun is highly conductive the field is frozen into the expanding wind and consequently the field is wound into a spiral too. There are of course variations in the speed, density and magnetic field due to activity on the Sun. For example the supersonic speeds of CMEs will cause shock waves to form, compressing the wind ahead and causing rarefied cavities behind as shown in Figure 1.8. Because of these compressions/rarefactions the magnetic field will be strengthened/weakened accordingly.

Spacecraft measurements show that the magnetic field of the solar wind is subject to sudden reversals of direction, dividing the interplanetary medium into sectors in the plane of the ecliptic where the magnetic field is pointing either towards or away from the Sun. This is explained as the spacecraft regularly passing through a surface which divides the Northern and Southern heliomagnetic hemispheres, the shape of the surface being like that of a ballerina's skirt. This interpretation is fortified by the disappearance of the sector effect when spacecraft are placed well above or below the ecliptic plane. There are also sudden changes in field polarity due to the presence of streams of material that have emerged from coronal holes; the magnetic field of the stream will have the same polarity as that of the hole.

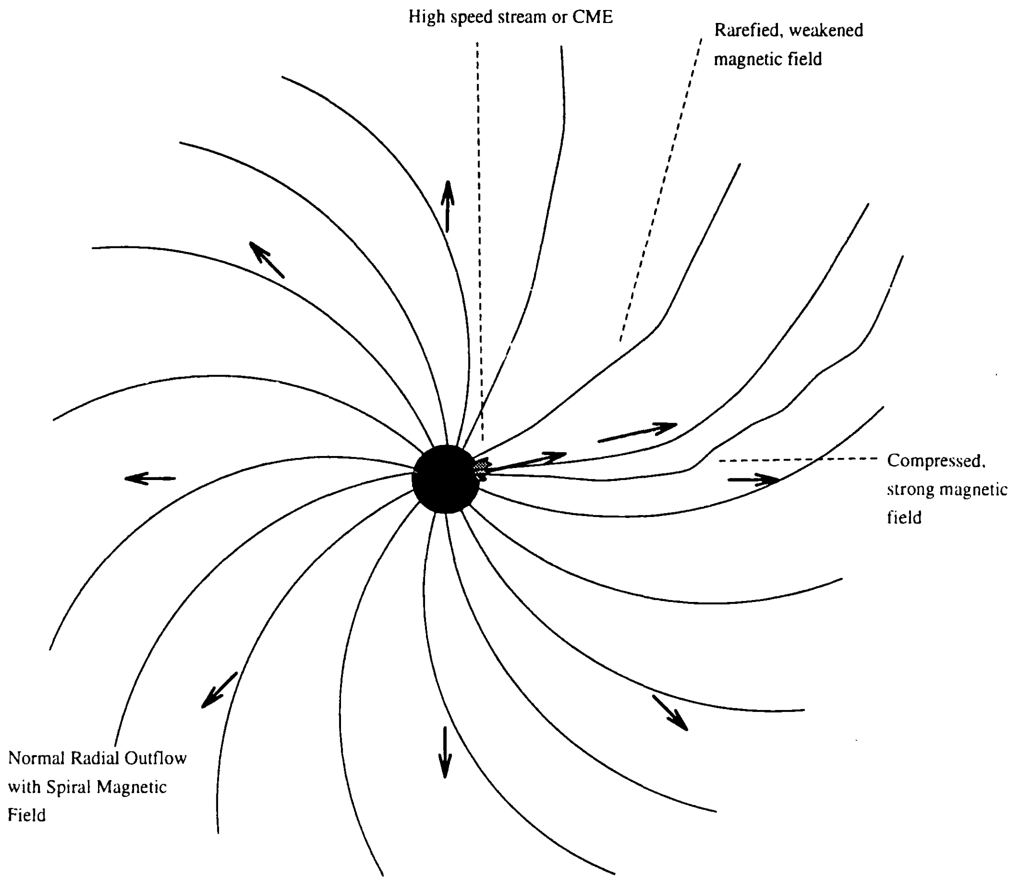

High speed stream or CME
Rarefied, weakened magnetic field
Compressed, strong magnetic field
Normal Radial Outflow with Spiral Magnetic Field

Figure 1.8: Schematic picture of the Solar Wind disturbed in one region by outflow of material from a coronal hole or by a CME.

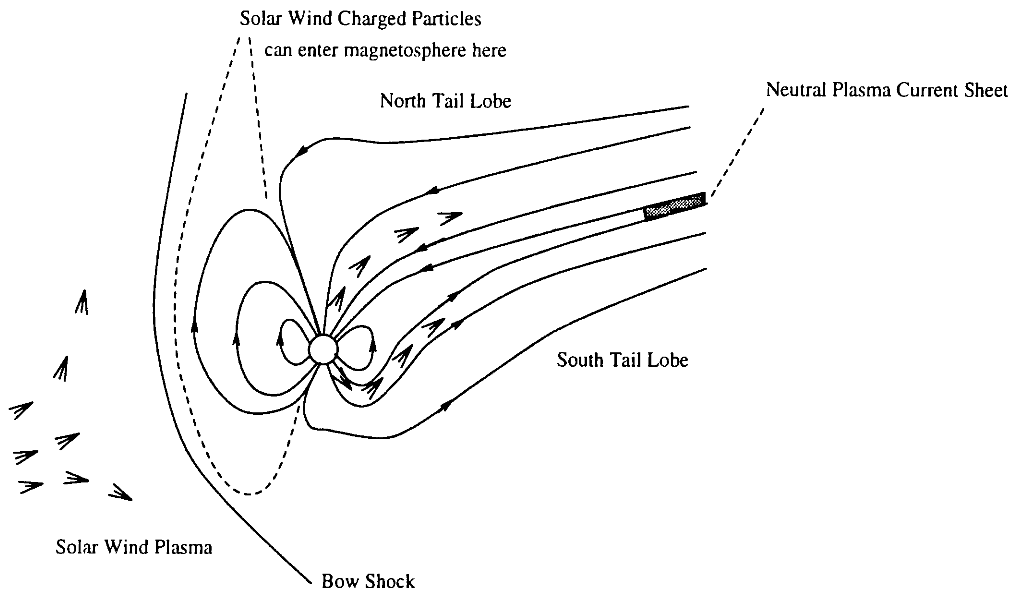

Solar Wind Charged Particles
can enter magnetosphere here
North Tail Lobe
Neutral Plasma Current Sheet
South Tail Lobe
Solar Wind Plasma
Bow Shock

Figure 1.9: Schematic picture of the Earth's magnetic field.

1.7.2 The Earth's Magnetic Field

The atmospheric gases of the Earth start to become ionized at about 40 km above the surface, the region from this level up to about 500 km is referred to as the *ionosphere*. The region above the ionosphere, where the geomagnetic field of the Earth almost entirely determines the motions of charged particles, is termed the magnetosphere and it extends to about 10 Earth radii towards the Sun and to over 100 Earth radii on the night side of the Earth. Figure 1.9 illustrates the basic shape of the Earth's field. The inner field is dipolar with the North pole at a geographic latitude of $77°$ North. On the sunward side the field is compressed by the incident solar wind with some field lines being "peeled off" and swept into what are termed the tail lobes of the field. At the polar cusps, which divide the closed sunward field lines from the open tail lobe field lines, particles are able to enter the magneto-sphere either being trapped along the closed field lines or being released to flow in the tail lobes. Apart from this the solar wind plasma is segregated from the magnetospheric plasma because of the "freezing out" of the wind's field, causing the wind to be diverted around the magnetosphere. Over 40 radii behind the Earth there is a magnetically neutral current sheet, where the oppositely directed open field lines can meet and reconnect during substorms (see below).

1.7.3 Solar Activity and the Earth

Geomagnetic activity is almost entirely due to concentrations of material in the solar wind. The variations due to the internal electric currents of the Earth are almost negligible in comparison. The presence of streams or speed, density and magnetic field enhancements in the stellar wind can sometimes be traced back to the Sun indirectly and directly. The most obvious indirect method is the recognition of a 27 day cycle in geomagnetic activity, as in Figure 1.6. A direct association between individual solar and geomagnetic events is difficult because the solar material takes several days to reach 1 AU and in doing so travels along a curved path which may or may not coincide with the Earth. Even when it does, its effect can be quite complicated depending on the orientations of the magnetic field of the Earth and the wind material. Until the 1970s, it was thought that geomagnetic storms were induced by streams of material from Bartel's mysterious M-regions and by the release of material in solar flares. Since M-regions turned out to be coronal holes and material has been observed to be ejected from the Sun in the form of CMEs it is now believed that these coronal phenomena are responsible for much of the geomagnetic activity. It has been argued by Gibson (1993) that all geomagnetic effects can be linked to CMEs and that in comparison solar flares are unimportant. However there is mounting evidence (e.g. Bravo (1994)) that CMEs, erupting prominences, flares and active regions are all connected, occurring in or near coronal holes. So to say that material and energy released in a flare has an inconsequential effect on the solar wind may be true but to say that solar flares are unrelated to disturbances in the wind, and therefore at the Earth, is false.

The monitoring of geomagnetic activity in terms of the disturbance of magnetic fields at various stations across the Earth shows three basic kinds of behaviour: *Crochets*, *Storms* and *Substorms*.

The regular S_q and L_q variations that are due to electric currents in the lower ionosphere can be enhanced by heating and ionization from increased levels of soft X-Rays and Extreme Ultra-Violet (EUV), which can result from a solar flare. These enhancements are called *crochets* and are quite small (about 0.1% of the Earth's normal field) and for obvious reasons occur only during the day. Magnetic *storms* are more violent and spread world wide often commencing with a rapid rise in magnetic field strength over few hours followed by strong fluctuations that last for a couple of days. If measured at low latitudes a marked drop in overall field strength is also observed. The initial rise is due to the compression of the day side magnetic field by the incident wind disturbance. The following drop is caused by a *ring current* of charged particles around the equator effectively generating a magnetic field that opposes that of the Earth. A *substorm* lasts only for half an hour and can occur during storms or in quiet periods. The size and frequency of the storms depend on the orientation of the solar wind's magnetic field, which can switch suddenly due to sector effects and high speed wind streams. If the component of the wind magnetic field parallel to the Earth's magnetic axis is opposite to the direction of the Earth's field then reconnection can occur, resulting in more open field lines being swept behind the Earth into the tail lobes. This then causes reconnection on the nightside of the Earth which has been observed to send plasma moving towards and away from the Earth. For these reasons such a configuration of wind-Earth magnetic field make substorms strong and frequent, whereas substorms become infrequent and less severe if the magnetic field components match. Associated with these substorms are the auroræ whose stunning visual manifestations are caused by the excitation of neutral atmospheric atoms by energetic electrons in the polar regions of the magnetic field. Auroræ are most common in an oval band which encircles the magnetic poles and which can be seen very clearly from images recorded by satellites.

1.8 Predictions of Solar-Terrestrial Activity

This section reviews the application of the prediction schemes outlined at the start of this chapter to the problem of forecasting the Sunspot Number time series. A serious problem in comparing predictions is that different authors have used different measures of prediction accuracy. Most authors provide a root mean square type error but as seen earlier the errors should be distributed as white noise for the predictions to be "completely predicted". This requires further information on the authors' work which is simply not available in most cases. A further problem is that the errors are of course based on each author's test set, so that comparison of like with like is not possible unless I repeat the entire methods of each author exactly. Despite this, comparison of errors must have some meaning and so, where appropriate and possible, I include RMS errors either quoted from the papers directly or inferred from the authors' results.

McNish and Lincoln (1949)

Data: SSN - 12 month smoothed and 3 month smoothed: 1834-1943

Method: McNish-Lincoln

RMS:

1 year ahead: using only the previous year 7.1

2 years ahead: using only the previous year 10.6, using the last two years 11.1 This paper provides one of the most longstanding prediction mechanisms, which is still used today to provide SSN and

SF forecasts in the monthly Solar-Geophysical Data reports published by the *World Data Centers*. The predictions above are made from the start of the cycle so I use P to mean the predict ahead time from the start of the cycle. The prediction errors increase until $P \geq 6$ when the errors appear to decrease. This is explained by the fact that the SSN residuals a larger "variance" at SSN maximum and therefore the prediction errors there will be higher too. It is also worth noting that using the mean cycle alone results in an error of 7.4. It is not stated in the paper whether the authors believe that the use of the previous year is actually a statistically significant improvement over the mean cycle used by itself. Using 3 monthly smoothed SSN, based on the SSN from eight months into the past the prediction accuracies are as follows: 8 months ahead: 4.7
16 months ahead : 9.0
24 months ahead : 10.6
Using only the mean in this case is significantly worse giving 8,16 and 24 month prediction errors of $5.7, 11.5, 17.7$

DeMeyer (1981)
Data: SSN annual means 1749-1979
Method: Empirical Mathematical
RMS:
No prediction errors quoted, see text.

The prediction methods investigated in this paper are really geared to the mathematical modelling of SSN, in doing so 3 such models are addressed. Model A recursively identifies and subtracts the strongest frequency components from the data until no significant peaks remain in the spectrum. The author then finds that the resulting residual time series is white noise. Model B proceeds in a similar fashion but multiplies every second maximum by -1 so as to reflect the 22-year magnetic cycle. Model C abandons any attempt to work with periodicities as the author feels that the sunspot cycle is not formed by the superposition of many periodic processes. Instead functions of the form

$$W_j(t) = A_j \left(\frac{t-t_j}{\tau_j}\right)^2 \exp\left[-\left(\frac{t-t_j}{\tau_j}\right)^2\right] \quad ,t \geq t_j \tag{1.18}$$
$$= 0 \quad ,t < t_j \tag{1.19}$$

are used to represent cycle j, with A_j, t_j and τ_j being fitted numerically for each cycle. W is in fact a Maxwellian function and is used because it characterizes the tendency for cycles to have shorter attack than decay phases. However, it can be seen from any plot of SSN that the smaller amplitude cycles are generally more symmetric than large amplitude cycles, something which cannot be accounted for in this mathematical model. By the author's own admission these models are neither useful nor reliable for prediction. This is because these models are fitted with no regard to the problem of generalization, discussed earlier. This might have been overcome in models A or B by using the periodicity removal algorithm on two separate sets from the data and comparing the results. Model C cannot really make a good prediction unless some use is made of the correlation properties of A_j, t_j and τ_j for successive j's.

Kapoor and Wu (1982)
Data: SSN annual means, monthly means and daily values
Method: ARMA
RMS:
Yearly, ARMA(5,4), 16.3 (1 year ahead)
Monthly, ARMA(8,7), N/A (1 month ahead)
Daily, ARMA(2,1), N/A (1 day ahead)

This paper suggests that an n^{th} order continuous stochastic time series when discretised produces not an AR(n) model but an ARMA(n,$n-1$) time series. For this reason the authors assume that SSN is well described by an ARMA(n,$n-1$) model (though they do not state why they believe the Sun conforms to a continuous n^{th} order stochastic process). The paper does not clearly state errors for each timescale of prediction, but from the quoted variance of the white noise terms for yearly numbers I inferred that the one year ahead prediction error would be 16.3. Since this error is derived from the data used to fit the model I expect that prediction error on an independent test set would be larger than 16.3. However, the results they proceed to quote using only data prior to 1958 have an absolute error that is always less than 15. This discrepancy is not accounted for in the paper.

Holland and Vaughan (1984)
Data: smoothed monthly solar flux
Method: Adapted McNish-Lincoln
RMS: Prediction errors N/A see text.

This paper's method is really just the McNish-Lincoln method with the interpolation step (required to stretch each cycle to the fixed number of points in the mean cycle) performed by a Lagrangian least squares algorithm. It is therefore somewhat surprising that the predictions turned out to be more accurate. Much of this paper is devoted to assessing the statistical meaning of the predictions, i.e. establishing confidence intervals on the predictions and using the χ^2 statistic to make comparisons with the standard McNish-Lincoln method. In so doing an RMS error is not quoted so I cannot include an error value for this method.

Schatten and Sofia (1987)

Data: Annual mean values of SSN and SF for cycle 22 max.

Method: empirical-physical dynamo model

This paper predicts the maximum SSN for a given cycle using measures of the Sun's polar field strength at solar minimum. According to the dynamo picture of the Sun (see end of Section 1.7.1) the Sun has poloidal field at solar minimum, which, through differential rotation, develops into a toroidal field and from there into a complex web at solar maximum, which emerges through the surface of the photosphere giving rise to active regions with sunspots etc. The motivation of this paper relies on the assumption that a strong poloidal field at solar minimum is indicative of heightened activity at solar maximum. Since solar dynamo models cannot provide any more quantitative rigour to the last statement the authors' turn to empirical linear relationships of the form

$$R_m = k\,B$$

where R_m is the maximum annual mean Sunspot number, B is some (direct or indirect) measure of the solar polar field strength and k is an empirically derived constant. The three methods of estimating the poloidal field strength of the sun were as follows: direct measurement of field strengths at the poles, flattening of the current sheet dividing the heliomagnetic hemispheres in the Solar System and the bending angle of the polar magnetic field inferred from streamers. In this way the authors predicted that the maximum would occur in 1990 ± 1 year and be of the value 170 ± 25. Using the correlation with SSN the SF value was inferred to be 210 ± 25. According to SSN solar maximum occurred in 1989 peaking at 158, below the prediction but well within the authors' uncertainty limits. The solar flux also peaked in 1989 at a value of 213.6, with a secondary maximum in 1991 of 208. Despite this seemingly accurate prediction it must be remembered that the predicted value of 210 came indirectly via prediction of SSN, which was not nearly as accurate, and through a good but not perfect correlation with SSN. This is suggestive that in this case two wrongs made a right.

Gonzalez and Schatten (1987)

Data: Annual mean values of SSN and SF for cycle 22 max.

Method: empirical-physical dynamo model
This method operates on the same principles as Schatten and Sofia (1987) above, this time using the minimum of the monthly smoothed *aa* and A_p geomagnetic indices to provide the connection with the solar poloidal field that can be exploited to predict the SSN maximum. In this case 10 cycles of *aa* index and 5 cycles of A_p are used to build up a roughly linear correlation between maximum sunspot number and minimum *aa* value. Using this method they predicted that smoothed monthly SSN maximum would occur in October 1990$\pm$ 9 months with a value of 166 ± 35. The smoothed monthly sunspot number maximum (as calculated in this thesis) actually occurred in November 1989, reaching a value of 156.

Zhang (1988)
Data: Annual mean SSN
Method: Threshold Autoregressive Model
RMS:
1 Year Ahead: 16.3

This paper makes a straightforward application of a Threshold Autoregressive model described in Section 1.2.5. The author uses data from 1700 to 1956 to fit the model and then checks it on a test set from the years 1957-1985. From the values quoted in the paper I have inferred that the 1 year ahead RMS prediction accuracy of the model is 16.3, this can appreciated in Figure 1.10 where the RMS errors calculated over successive nine year intervals are plotted. The maximum of cycle 22 is predicted to occur in 1990-91 and have SSN reaching 81.2 ± 16.2. This prediction is an extreme underestimate with the (rather optimistic) error quoted by the author failing to cover the actual value. In fact judging by the predictions of the years 1986-94, where the RMS error is 51.47 (the last point in Figure 1.10), it seems that this model's predictions of the future are significantly worse than those on the author's test set. Figure 1.10 shows this as a sudden change in RMS error in 1985, the end of the test set. Either the author was extremely unfortunate or else the test set was incorrectly used to gauge the success of this method. If the latter is the case then the use of a Threshold Autoregressive model seems inappropriate for describing SSN.

Wilson (1990)
Data: Smoothed SSN for cycle maximum prediction
Method: Empirical-statistical

In this paper the rate of growth of SSN between minimum and maximum is correlated to the

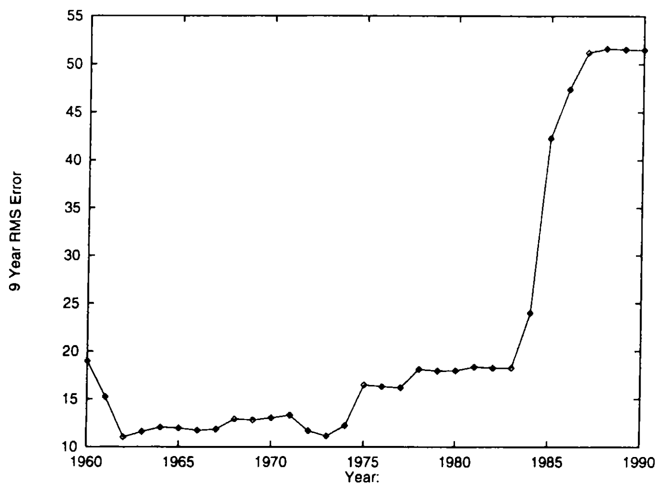


Figure 1.10: This is a plot of the RMS error of yearly SSN predictions evaluated on successive nine year intervals from 1956 to 1994. The years 1956-1985 were the test set and the last nine years 1986-1994 are the years for which the author made predictions into the future.

height of the maximum. The rate of growth is measured in a straightforward fashion taking the difference in SSN between minimum and t months after minimum and dividing by t. It is found that larger values of t provide a better correlation with maximum SSN, and so therefore provide better prediction accuracy. The author also points out that using the maximum measured rate of growth correlates best with maximum sunspot number, although it can occur only 1-2 years before the maximum allowing only relatively short term predictions. Using data up until the beginning of 1989 the author predicts that the height of cycle 22, according to smoothed SSN, will exceed 164.0 within a 97.5% confidence limit. The SSN peaked in 1989 at a value of 159.

Kurths and Ruzmaikin (1990)

Data: Annual mean SSN

Method: Nearest-neighbour method.

RMS:

1 Year Ahead: 30.5

This paper applies the method proposed by Farmer and Sidorowich (1987) (see Section 1.3.3) to the prediction of yearly SSN with the solar magnetic polarity included by multiplying every odd cycle by -1. The years 1749 to 1923 were used to fit the model, with data thereafter reserved for

Table 1.1: Prediction RMS errors for various statistical models. Note that the number of parameters includes the constant mean which is subtracted from the data before fitting.

Model	AR(9)	Subset AR(9)	TAR	Subset Bilinear
RMS	13.8	14.6	12.2	11.13
No. of Parameters	10	4	19	11

use as a test set. Only a plot of the predictions on their test set is provided in the paper and so by method of using a ruler to measure the gaps between actual and predicted SSN I have found the RMS error of their method to be 30.5, evaluated over the range 1926 to 1979. The majority of the errors were incurred at the largest maximum in history, that of cycle 19 and the maximum of cycle 18. They also predict the maximum of cycle 22 to occur in 1990-91 with a peak value of about 162.

SubbaRao and Gabr (1984)
Data: Annual mean SSN 1700-1955
Method: Bi-linear, AR, Threshold Autoregressive
RMS: See Table 1.1.

These results can be found in Chapter 6 of the book, where the three types of model are compared. Also used are ***subset*** AR and bi-linear models. When fitting a time series model such as an AR(N) model all N previous values of the time series' history are used. If some of the coefficients are found to be small, it might be worthwhile refitting the model missing out those terms; this is what is meant by subset. For example in the study in this book, the AR(9) model uses $X_{t-1}, X_{t-2}, X_{t-3}, \ldots, X_{t-9}$ but the subset AR model uses only X_{t-1}, X_{t-2} and X_{t-9}. The models were fitted using data from 1700 to 1921 and the prediction accuracy tested on 3 cycles of data from 1922 to 1955[10], the RMS errors inferred from the authors' 1 year ahead prediction variances for each model are listed in Table 1.1. Although these prediction errors are quite impressive it must be remembered the huge maximum of 1957 is not included in the test set, though it was included in the test sets of previous authors. As shall been seen with NNs later in the thesis the 1957 maximum is a surprise to almost every prediction scheme.

Since the authors are approaching the subject from a statistical background they also examine the properties of the residuals, concluding that the residuals are both independent and normally distributed for each model. Despite these statistical results the independence of the residuals appears

[10]a repeated mis-print in the authors' text states that the range is 1922 to 1935.

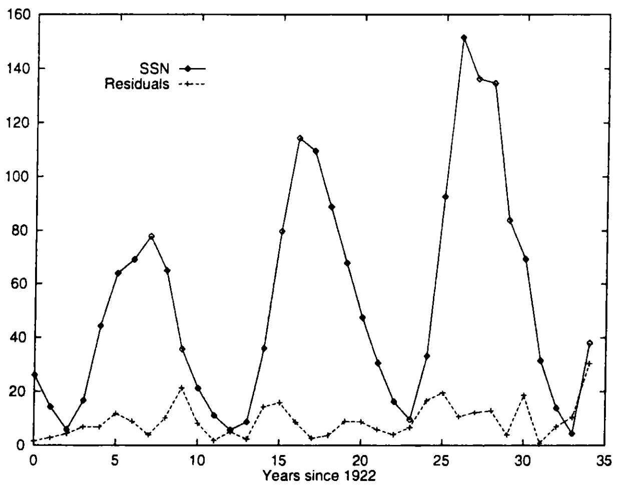


Figure 1.11: The residuals of the bi-linear model.

to be rather frail, judging by Figure 1.11, which plots the residuals of the Bi-linear model with Sunspot Number. It is immediately obvious that prediction error is larger during the rise and fall of each cycle.

Chapter 2

The Data and its Analysis

"Tell us what we want to know or we'll give you a kickin' and fit ya up good an' propa"
Jack Regan

2.1 The Gallery

Many introductory texts on time series analysis, e.g. Cryer (1986), start by stating that the most important means of analyzing a time series is just the visual inspection of a plot, i.e. just look at it. So the first step I take in this chapter is to provide a gallery of graphs that show the entire available and reliable history of the three important time series in their various formats: yearly, monthly, smoothed monthly and also extracts from the raw daily series.

As a note to these plots remember that the technology, personnel and method of observing has changed throughout the years so there are certainly subjective effects of some kind present. This is especially true for the *sunspot number* where the subjectivity is of such a level that two observers who are working with the same image can actually arrive at two different, albeit similar, values. This problem is partially removed by averaging over the opinions of many observers at many sites.

The K_p index does not suffer so greatly from this problem but still the methods for measuring the magnetic field disturbances could have changed, even just subtly, since regular records began in 1932. Again subjectivity of a single instrument is in part allowed for by averaging over the measurements at 13 stations located between the latitudes 63°S and 46°N: Lerwick (UK), Eskdalemuir (UK), Hartland (UK), Ottawa (Canada), Fredericksburg (USA), Meannook (Canada), Sitka (USA), Eyrewell (New Zealand), Canberra (Australia), Lovo (Sweden), Rude Skov (Denmark), Wingst (Germany), and

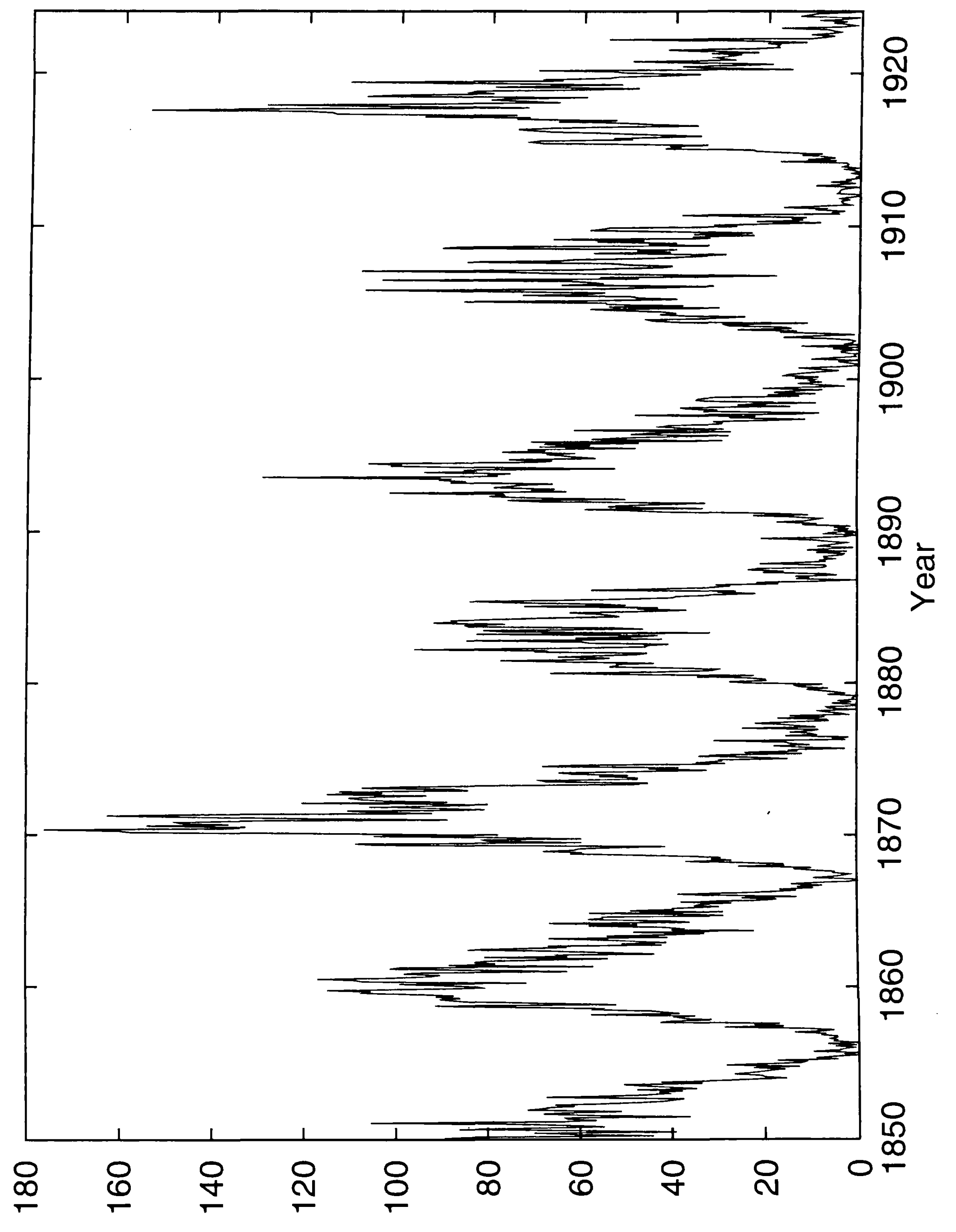


Figure 2.1: The first half of the reliable history of the Zurich (calendar) monthly average sunspot number.

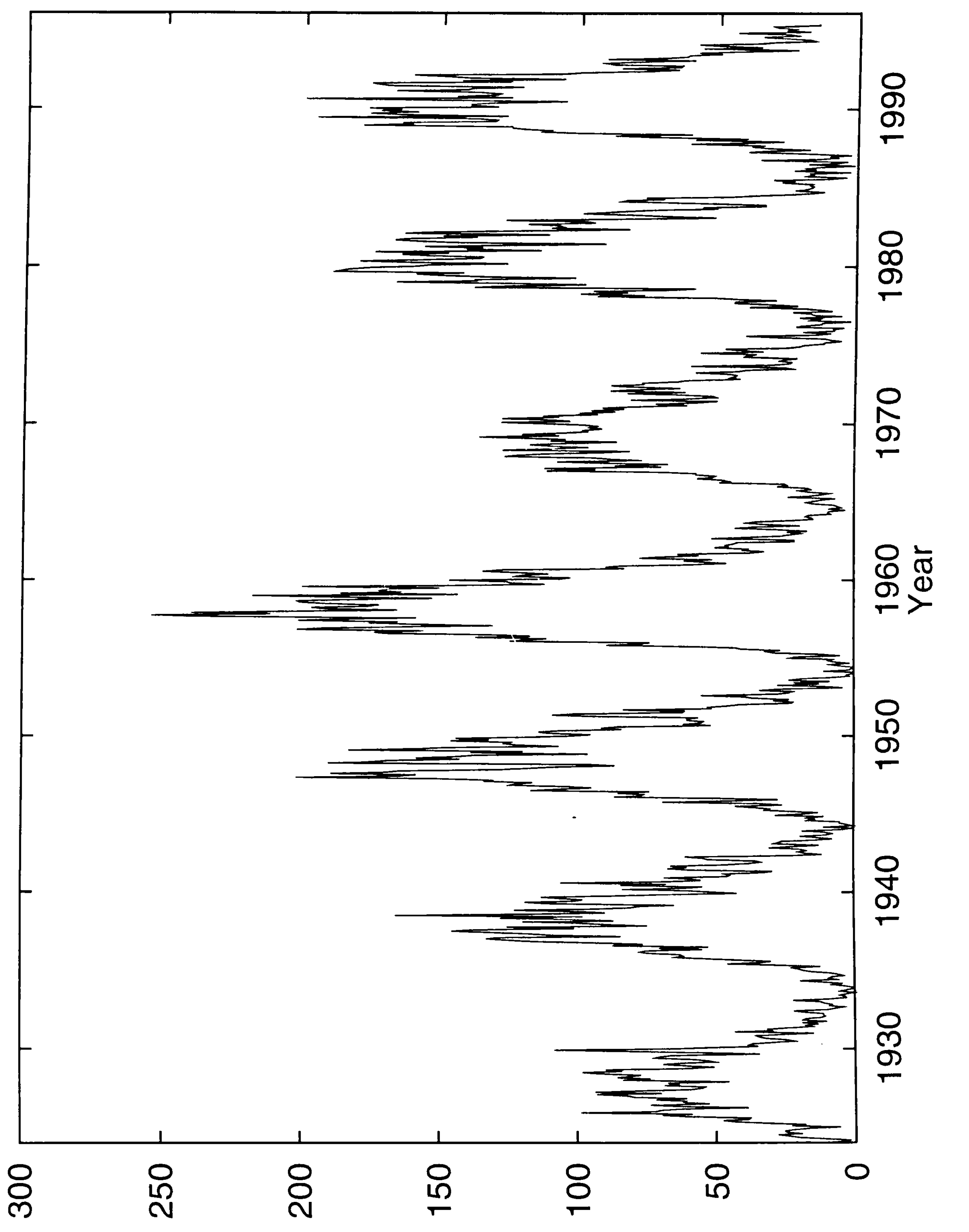


Figure 2.2: The second half of the entire reliable history of the Zurich (calendar) monthly average sunspot number. N.B. the scale is not the same as in Figure 2.1.

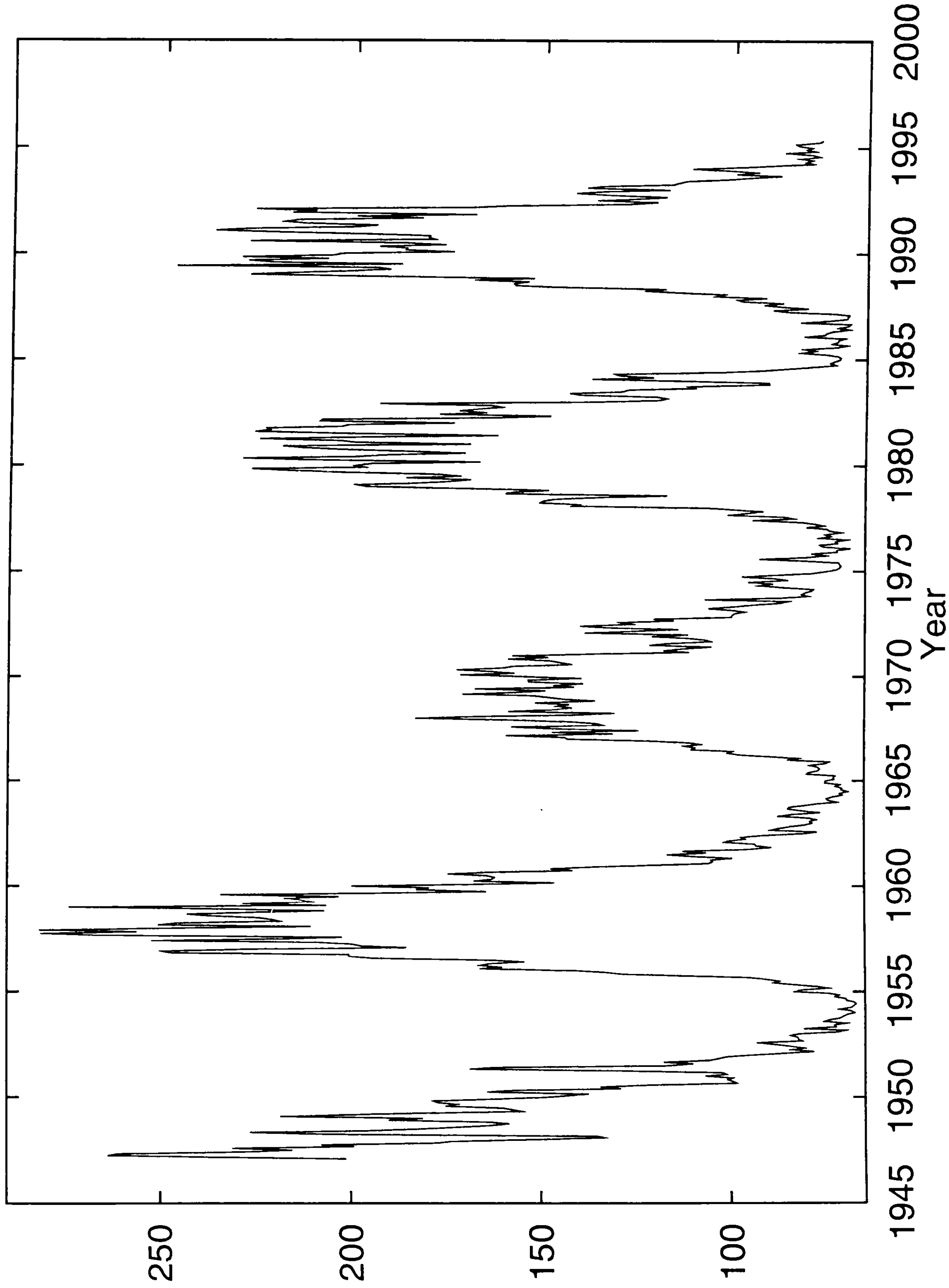


Figure 2.3: The entire history of the 10.7cm (calendar) monthly average solar flux.

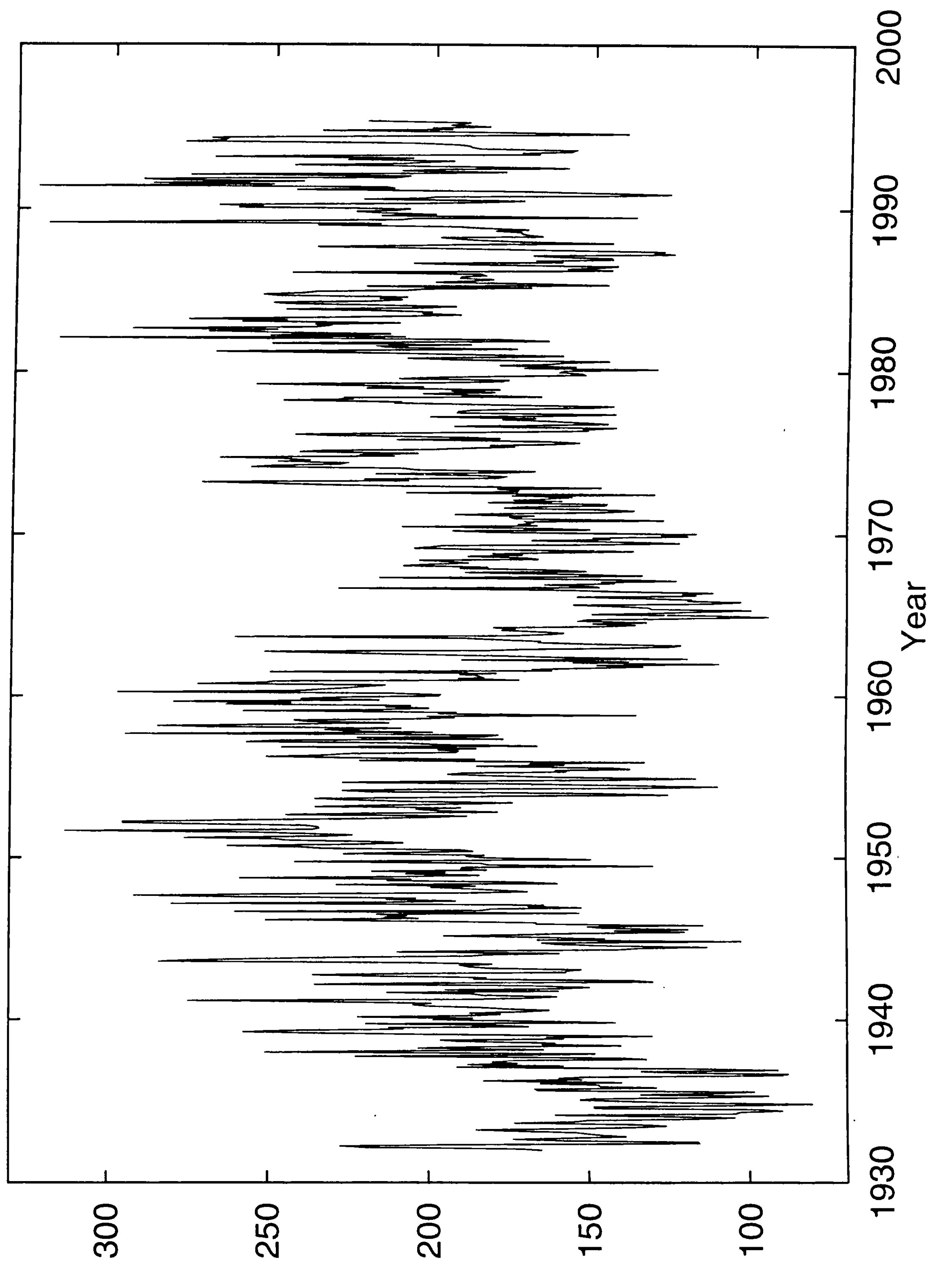


Figure 2.4: The entire history of the (calendar) monthly average K_p.

Witteveen (The Netherlands). Unlike sunspot number the geomagnetic disturbances are of course intrinsically dependent on the location of the observatory.

The *10.7 cm solar flux* has the shortest range of the three covering only four complete cycles to date. Unfortunately the data is not actually available for every day since its regular measurement began in February, 1947. Many daily values for many weekends and festive periods are missing from the data in years before 1963. Thereafter only 7 days are missing due to malfunctions and maintenance. For months that have days missing I have reduced the number of days in the month accordingly in the calculation of the monthly average. Unfortunately another version of the daily solar flux time series is available that seems to have these missing days "filled in" using linear interpolation. While this probably has little impact when working with monthly averages I feel it is important to treat the available data with respect and be wary of such "fixes".

2.2 Power Spectra

Figures 2.11, 2.12 and 2.13 show the *Power Spectra* of the three monthly time series: sunspot number, solar flux and K_p index respectively. For each time series a stretch of $M = 2^N$ months, where N is an integer, was selected and mean-subtracted so as to remove the zero frequency power from the spectrum. This data was then "windowed" using the *Bartlett Window*:

$$w_i = 1 - \left| \frac{i - \frac{M}{2}}{\frac{M}{2}} \right|$$

where i labels the month from the start of the stretch of M months used. The Bartlett window serves to reduce the leakage of data from neighbouring bins in the Fourier transform, improving on the usual situation where the window function is effectively just a square function of unit height. The discrete Fourier transform, $\hat{x}_j$ of the time series $x_1, x_2, \ldots, x_M$ is then:

$$\hat{x}_j = \sum_{m=0}^{M-1} w_m \, x_m \, e^{i2\pi jm/M} \qquad j = 0, \ldots, M-1$$

From the Fourier transform the power p_j in each frequency bin j can be calculated by the following formulae

$$\begin{aligned}
p_0 &= \frac{1}{W}|\hat{x}_0|^2 \\
p_j &= \frac{1}{W}\left(|\hat{x}_j|^2 + |\hat{x}_{M-j}|^2\right) \qquad j = 0, \ldots \frac{M}{2} - 1 \\
p_{\frac{M}{2}} &= \frac{1}{W}|\hat{x}_{\frac{M}{2}}|^2
\end{aligned}$$

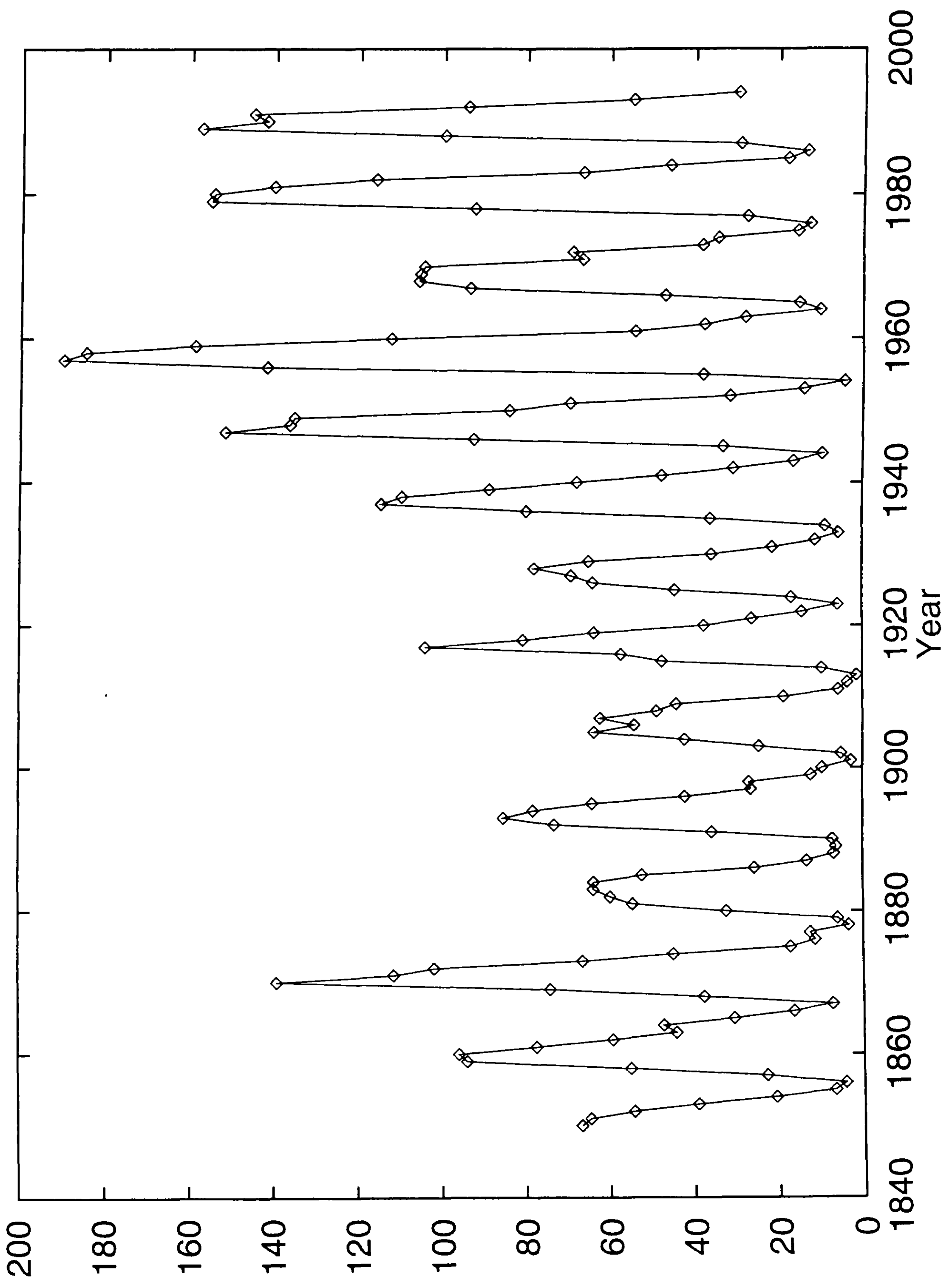


Figure 2.5: The entire history of the Zurich yearly average sunspot number.

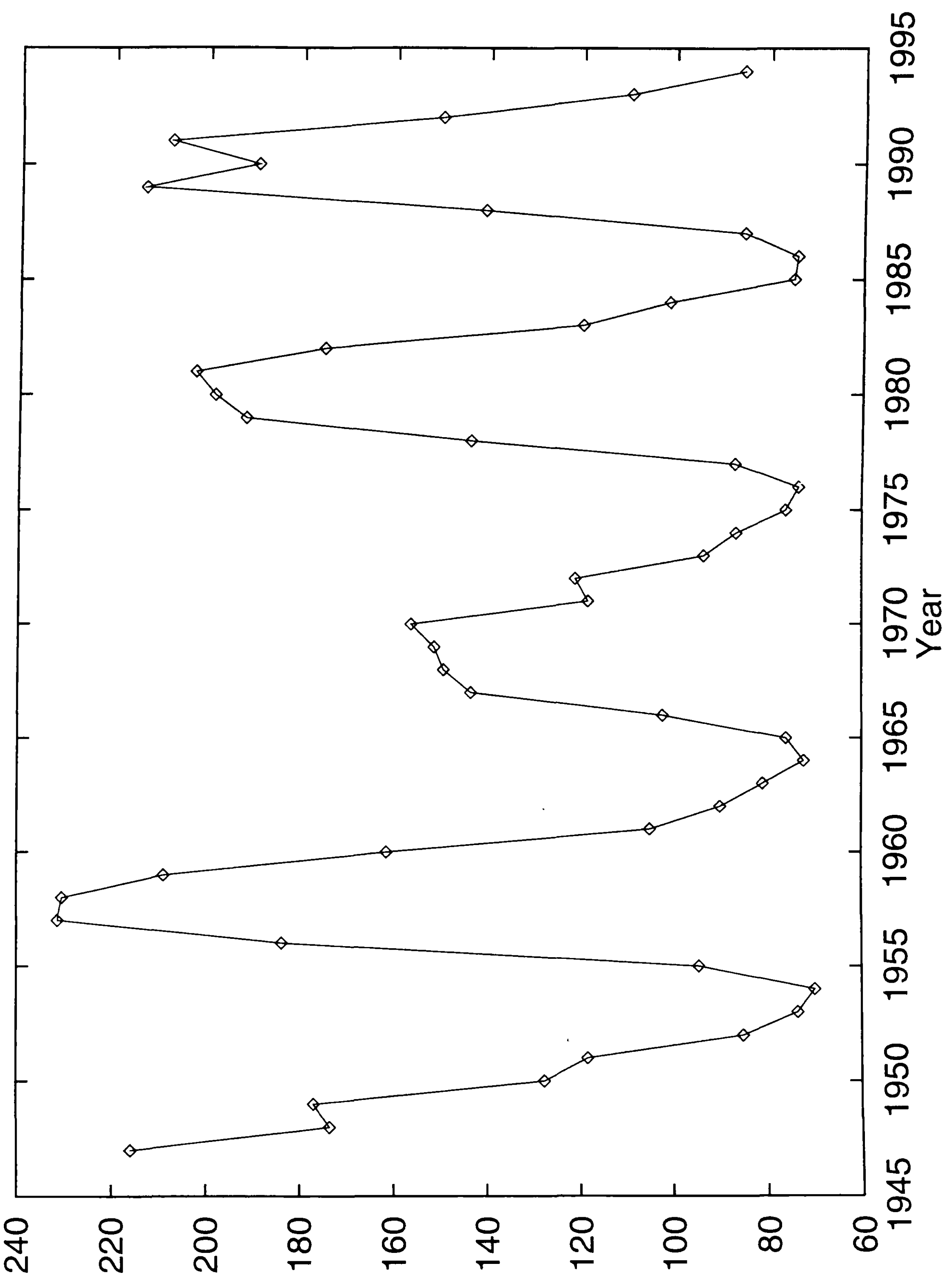


Figure 2.6: The entire history of the 10.7cm yearly average solar flux.

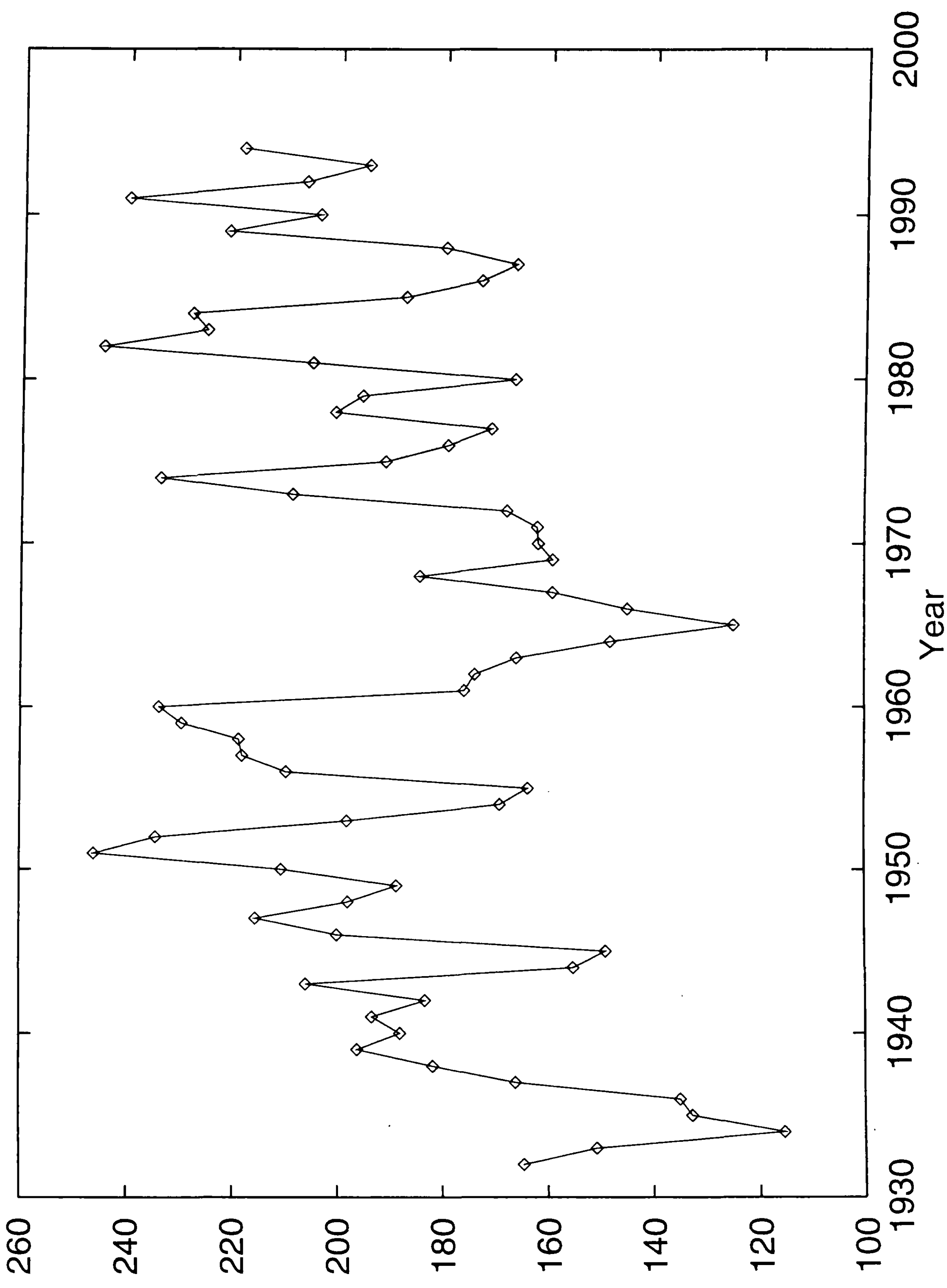


Figure 2.7: The entire history of the yearly average K_p index.

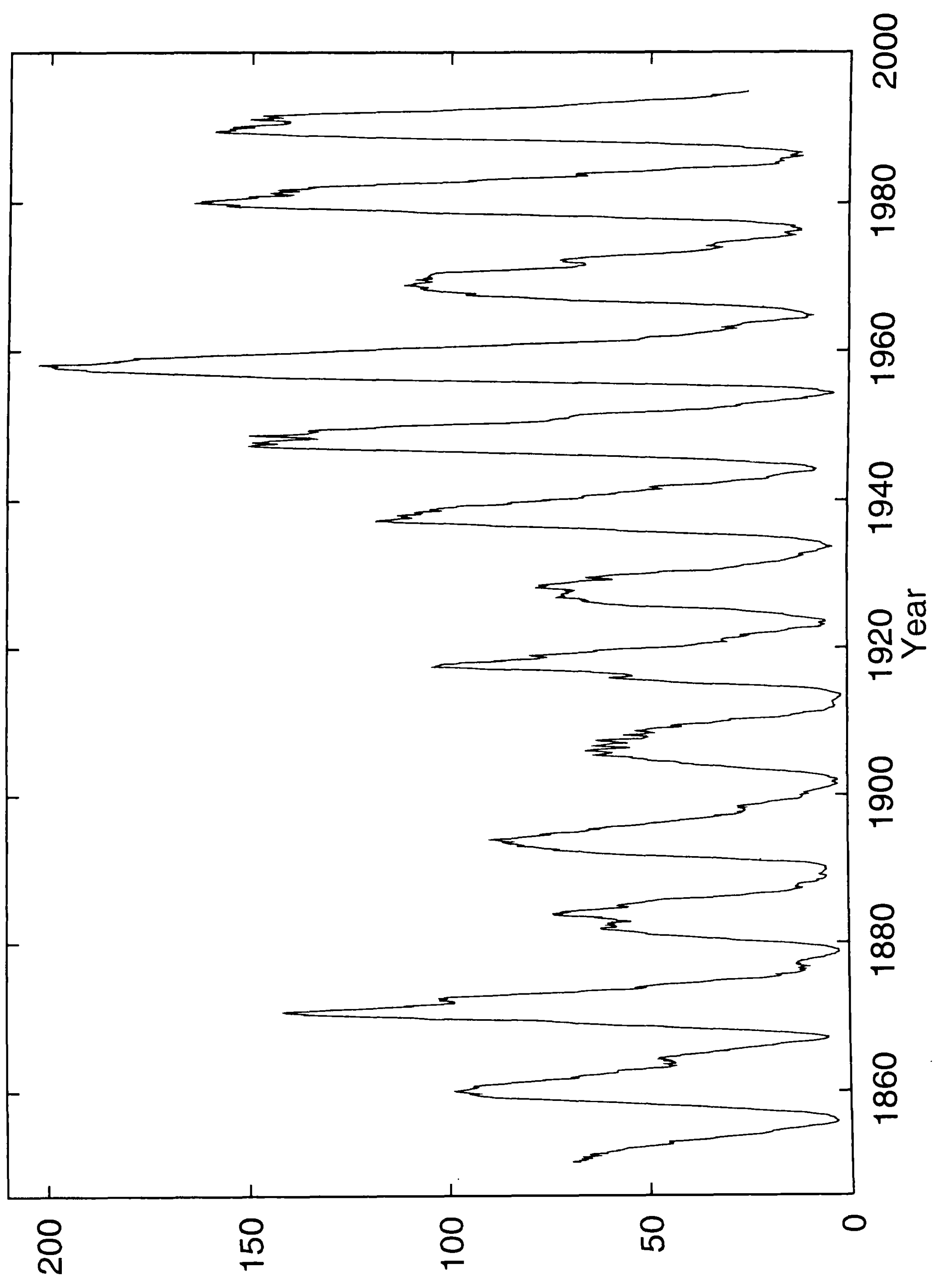


Figure 2.8: The entire reliable history of the 13 month smoothed Zurich (calender) monthly average sunspot number.

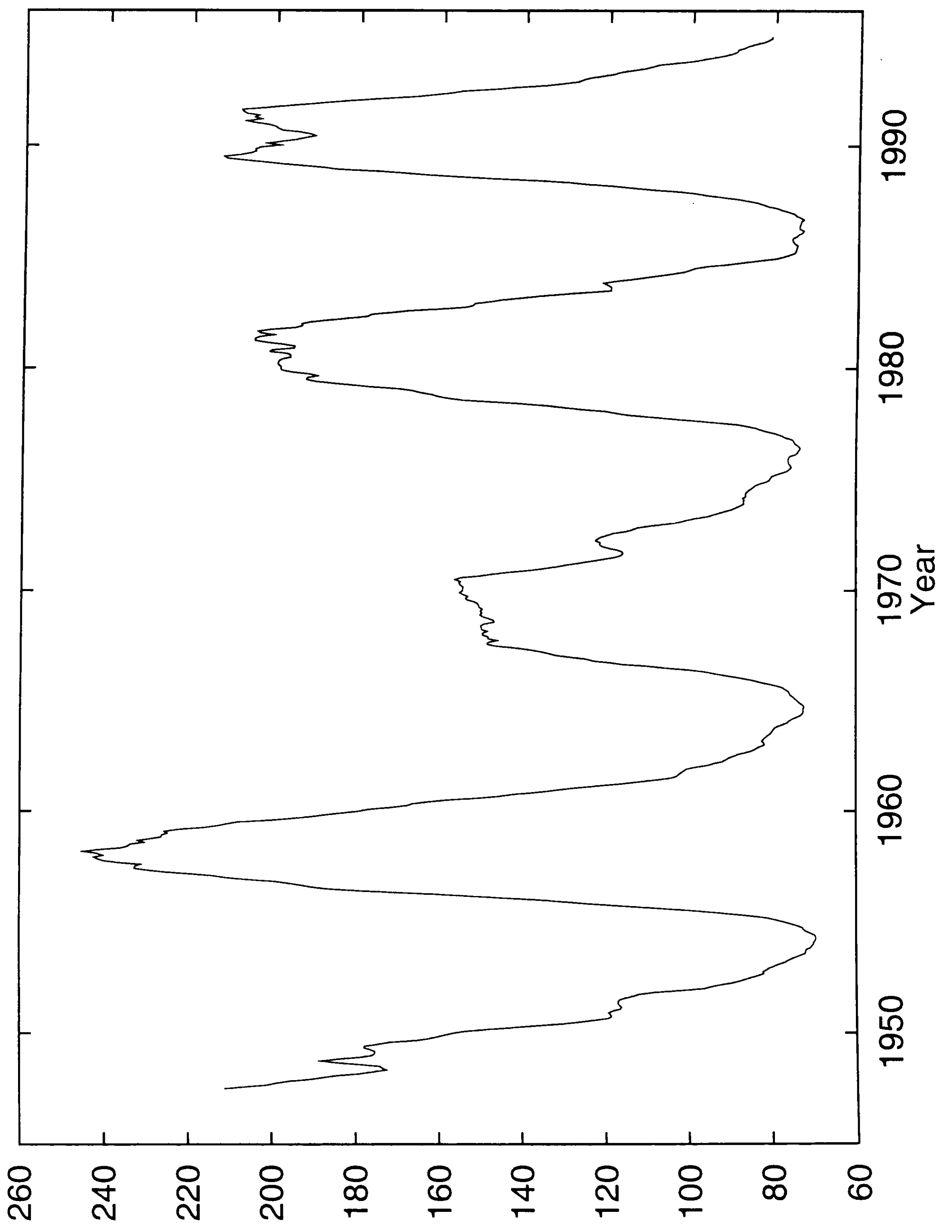


Figure 2.9: The entire history of the 13 month smoothed 10.7cm (calender) monthly average solar flux.

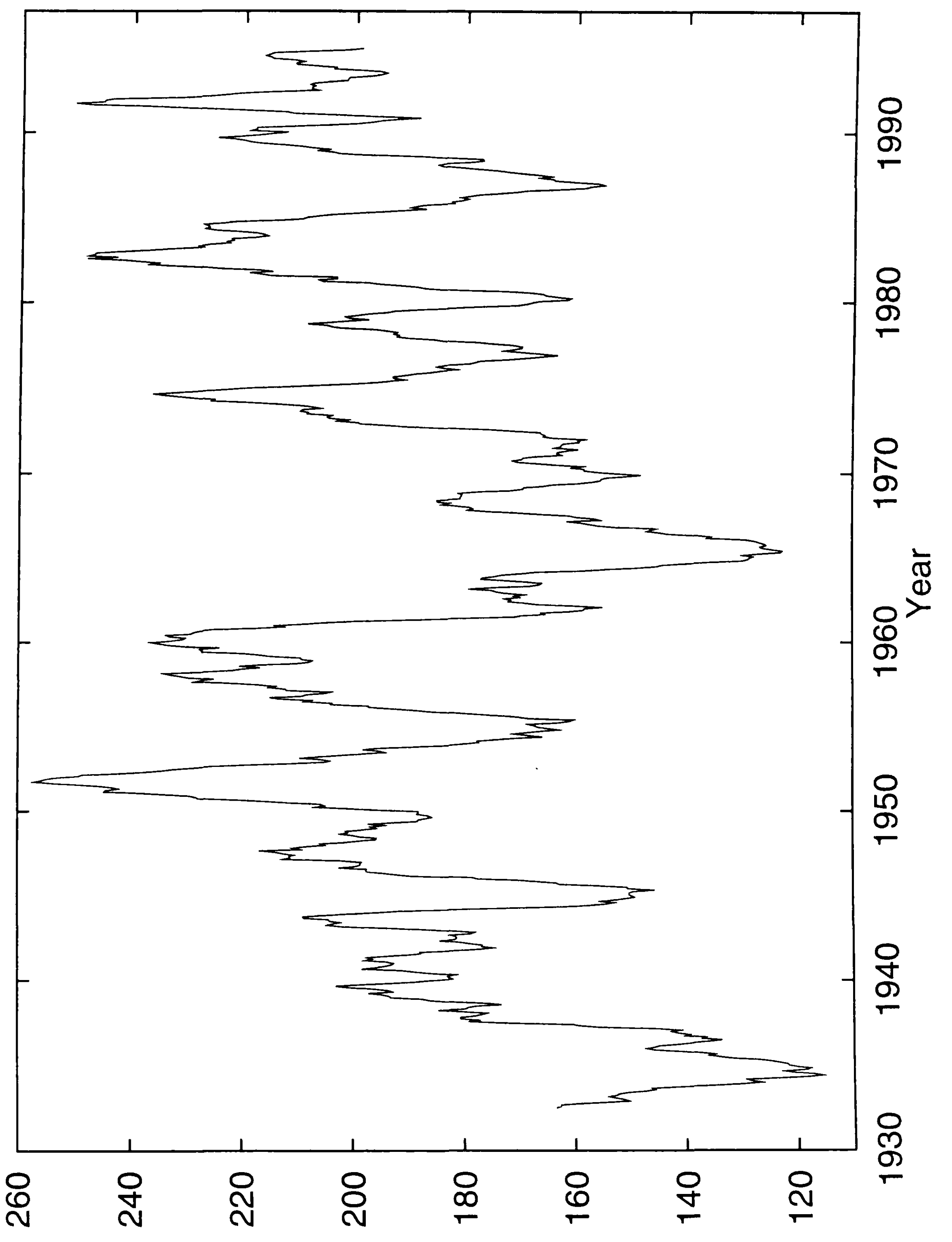


Figure 2.10: The entire history of the 13 month smoothed (calender) monthly average K_p index.

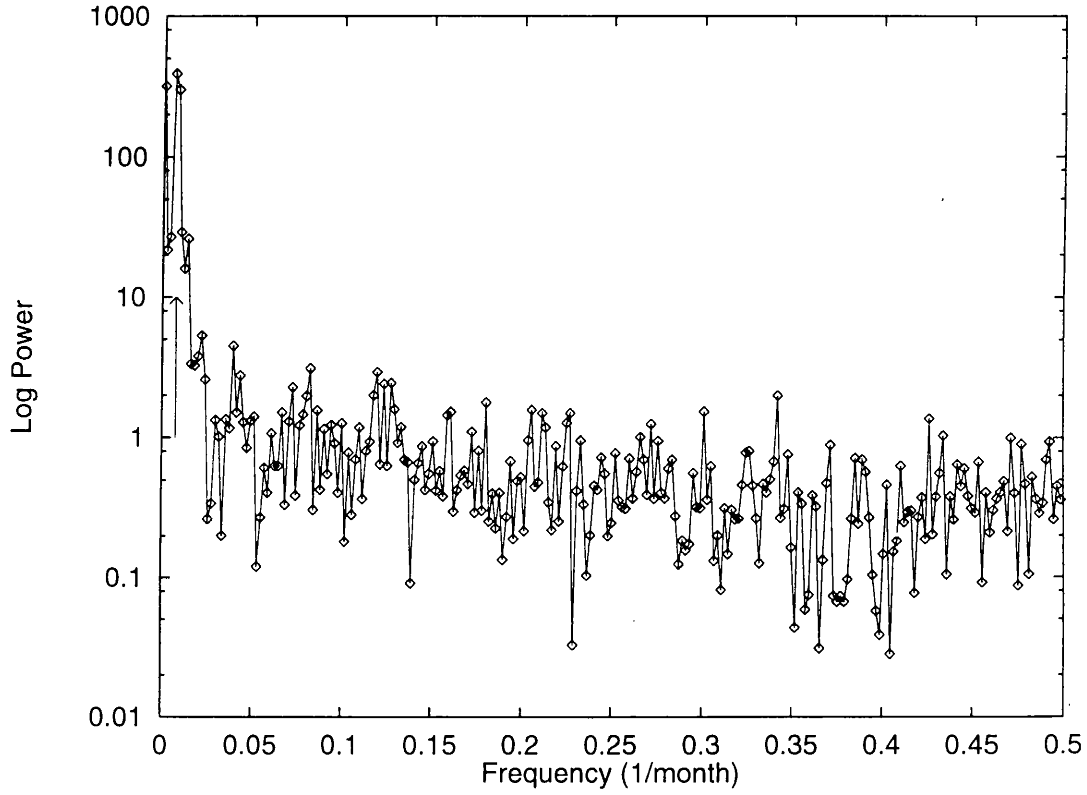


Figure 2.11: The power spectrum of (calender) monthly average sunspot number. The arrow marks the frequency corresponding to the 11 years.

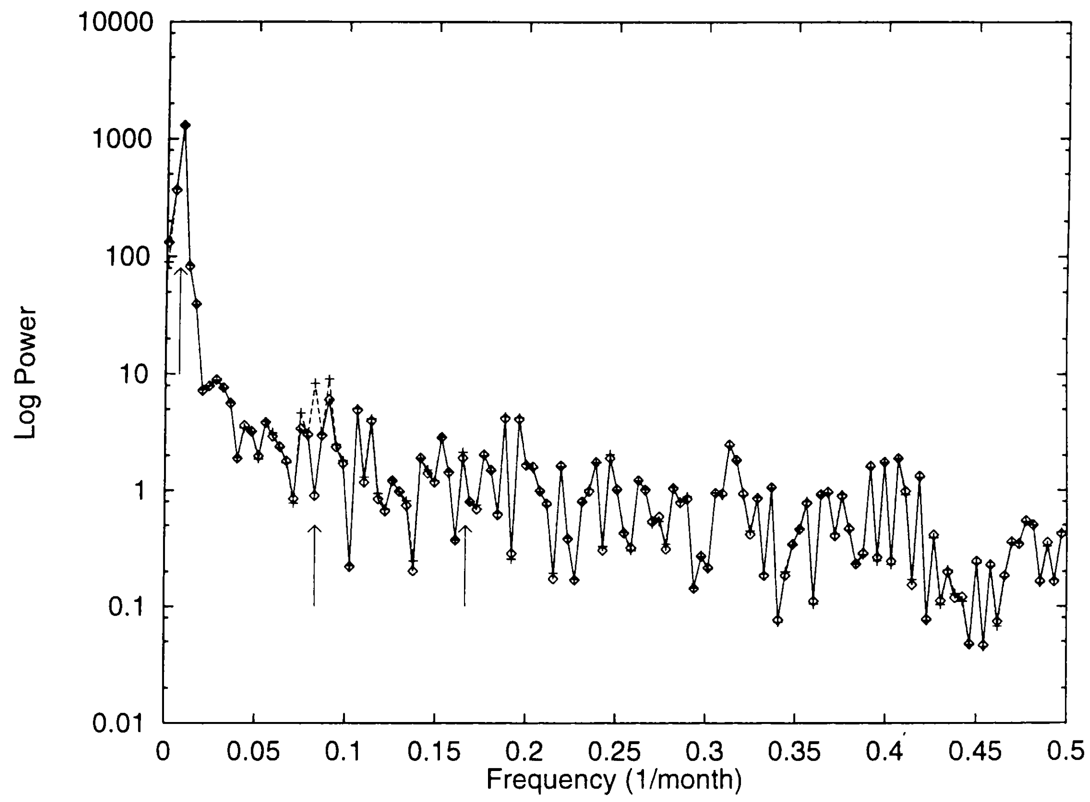


Figure 2.12: The power spectrum of (calender) monthly average solar flux. The dotted curve shows power spectrum of the adjusted solar flux time series and the solid curve shows the power spectrum of the observed solar flux time series. The arrows indicate the frequencies corresponding to periods of 11 years, 12 months and 6 months.

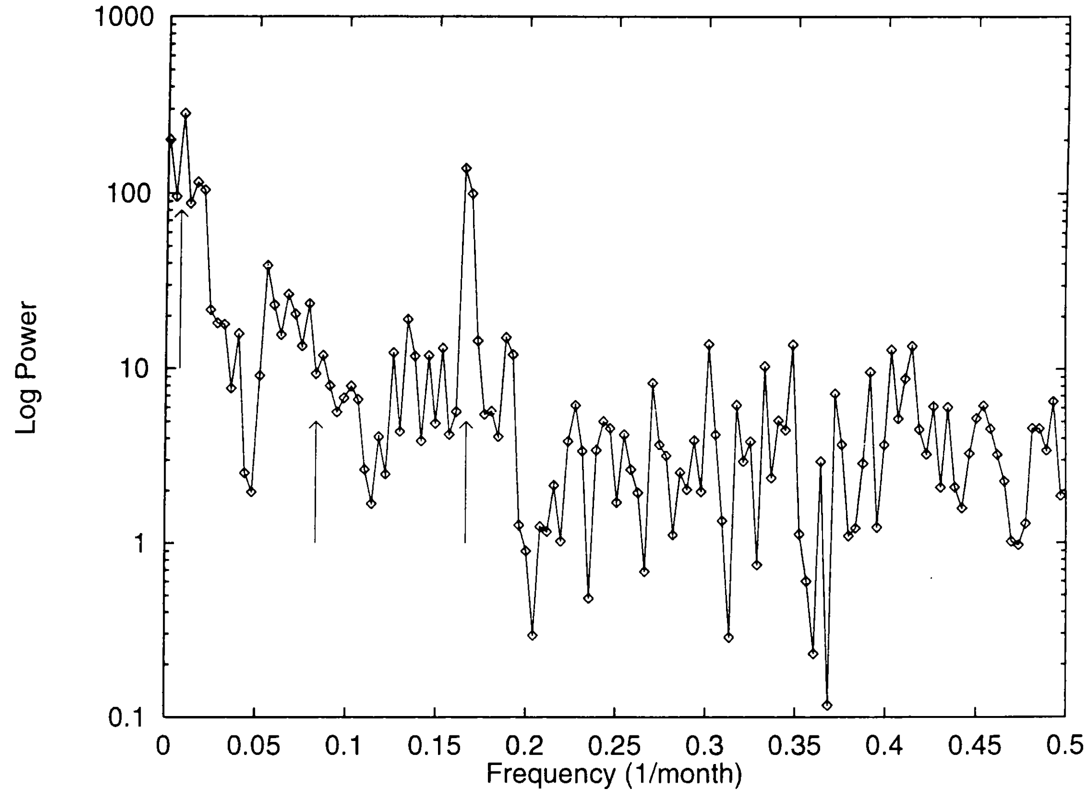


Figure 2.13: The power spectrum of (calender) monthly average K_p index. The arrows indicate the frequencies corresponding to periods of 11 years, 12 months and 6 months.

where the constant W is given by

$$W = \sum_{j=0}^{M-1} w_j^2$$

and frequencies in cycles per month, f_j, corresponding to the index of the Power Spectrum are given by

$$f_j = \frac{j}{M} \qquad j = 1, \ldots, \frac{M}{2}$$

The convention used above means that f_j for $j > \frac{M}{2}$ are really negative frequencies, i.e. f_{M-i} for $i < \frac{M}{2}$ is equivalent to $-f_i$. Since the data being used is a sample of a (supposed) stochastic process the p_j's should be regarded as estimates of the actual power spectrum. It can be shown (e.g. Jenkins and Watts (1968)) that the p_j's will be asymptotically unbiased estimates[1] as $M \to \infty$ but that these estimates have a variance of 100%, i.e.

$$Var[P_j] = (E[P_j])^2$$

where P_j is the random variable of which p_j is a realization. To improve on this I have smoothed the spectrums in a straightforward manner by re-binning the data in each plot. If the data is re-binned as

$$q_k = \sum_{j=0}^{b-1} p_{j+bk}$$

where b is the number of points in each bin, then the variance of the smoothed power spectrum estimates is given by

$$Var[Q_j] = \frac{(E[Q_j])^2}{b}$$

(The effect of re-binning on bin leakage is addressed in Appendix Section A.2, for here suffice it to state that it effectively reduces leakage). Of course the price to be paid is a loss of frequency resolution, so a balance between variance and resolution has to be made. For the plots in Figures 2.11, 2.12 and 2.13 I have only used bins of size $b = 2$ because larger bins result in the 11 year peak becoming unresolved.

The **sunspot number** power spectrum in Figure 2.11 only shows one clear peak, that of the 11 year solar cycle indicated by the arrow. By clear I mean outwith the limits dictated by the variance i.e. the variance of each p_j is equal to $\frac{E[P_j]}{\sqrt{2}}$. Since the cycle is so variable in length and height this peak is very wide, spread over the period range of about 7 to 16 years. The **solar flux** spectrum in Figure 2.12 again only shows one definite peak. In this plot there are two very similar spectra.

[1] Though the leakage between frequency bins complicates this statement.

The solid line is the spectrum obtained from the *adjusted* solar flux, which is actually the data that has been used so far in this thesis. The dashed curve is the power spectrum of the *observed* solar flux. The adjusted series corrects for the eccentricity of the Earth's orbit and in so doing removes some power from the 12 month, and to a lesser extent from the 6 month periods. This highlights two things: firstly that the variance of the spectrum is high enough to render the 12 year variation almost invisible and secondly that the difference between the adjusted and observed series is slight and confined to a very narrow frequency band corresponding to 11-13 months in period. The first point begs the further question which cannot really be answered; what other "real periodicities" lie buried in the noise of the spectrum? The $\mathbf{K_p}$ **Index** in Figure 2.13 has a somewhat weakened and broadened 11 year peak because the K_p time series shows the cycle to a much lesser extent than either solar flux or sunspot number. Also clear in this plot is the 6 month period peak (see Section 1.7), note that there is no trace of a peak at 12 months confirming that the 6 month peak is *not* a harmonic of some annual effect. By looking at the phase of this peak it can be determined that the two maxima that occur each year tend to occur at the equinoxes; this can also be confirmed from careful inspection of the data itself. This result has been previously reported in Fraser-Smith (1972).

Figures 2.14, 2.15 and 2.16 show the power spectra for the daily series. Since there is so much more daily data, large smoothing bins could be used to reduce the variance by a much larger factor than was possible for the monthly series. The **daily sunspot number** spectrum in Figure 2.14 shows two distinct peaks at periods of 27 and 13.5 days, the latter almost certainly being a harmonic. The **daily solar flux** spectrum in Figure 2.15 again shows the 27 day solar rotation period, but now a complete set of harmonics can be clearly identified. The $\mathbf{K_p}$ **Index** in Figure 2.16 is even more striking with the height of the 27 day harmonics dying off very slowly indeed. To explain these prominent harmonics I present an idealized model of a time series exhibiting a strong periodicity in the form of a series of impulses.

Imagine a time series is represented by a continuous function $T(t)$ of the form

$$T(t) = \sum_{j=0}^{J-1} S\left(\frac{t-bj}{a}\right) + \epsilon(t)$$

where S(x) is a function localized around $x = 0$ such that

$$S(x) > 0 \quad |x| < \frac{1}{2} \tag{2.1}$$

$$\sim 0 \quad |x| > \frac{1}{2} \tag{2.2}$$

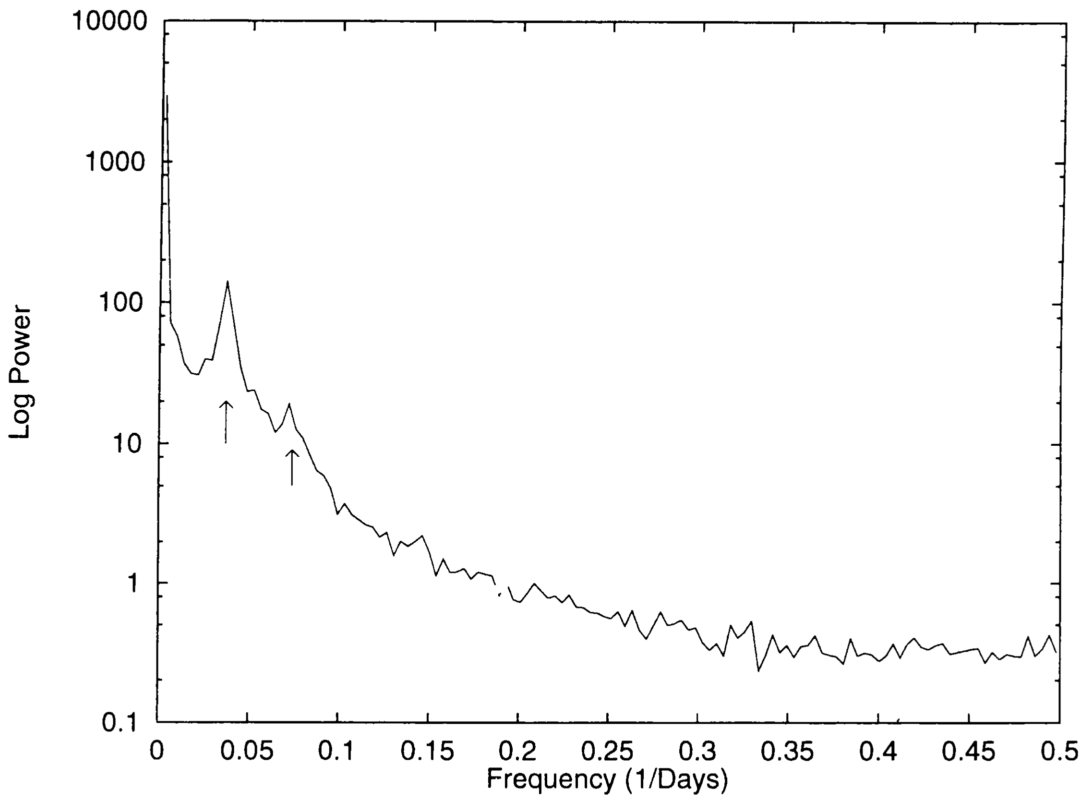


Figure 2.14: The power spectrum of daily sunspot number. The arrows mark the frequency corresponding to the 27 day solar rotation period and its first harmonic. The spectrum was smoothed using $b = 256$.

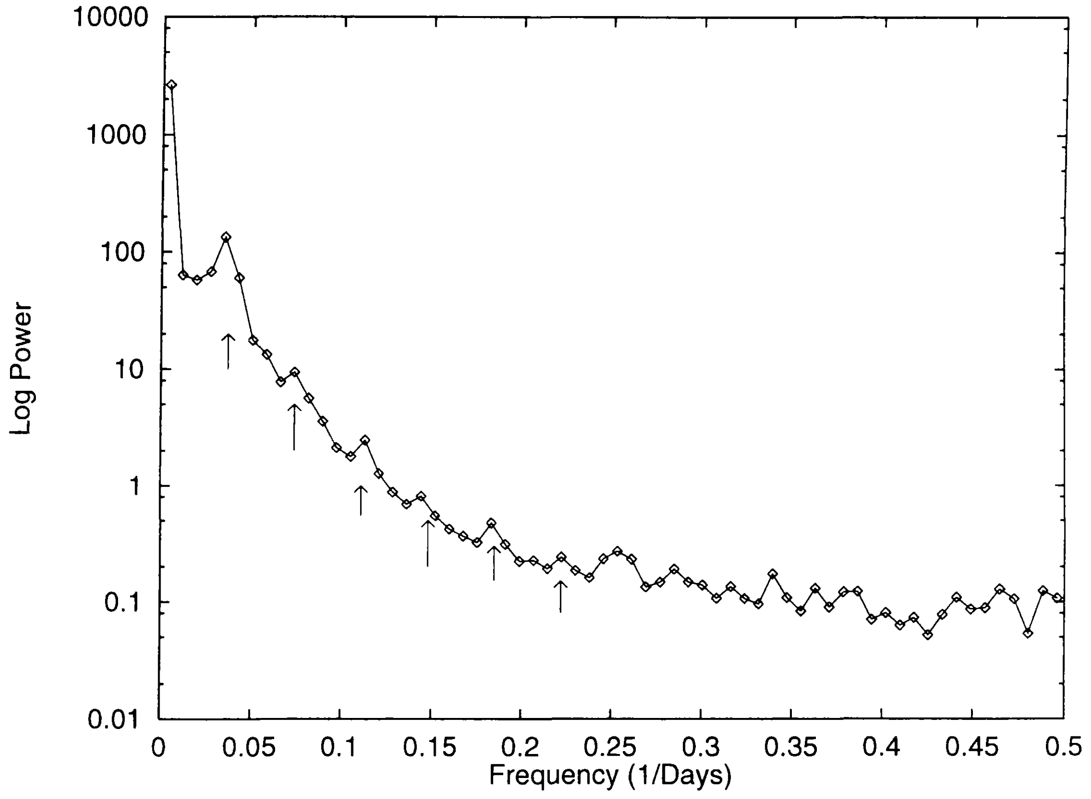


Figure 2.15: The power spectrum of daily solar flux. The arrows indicate the 27 day solar rotation period and its harmonics. The spectrum was smoothed using $b = 128$.

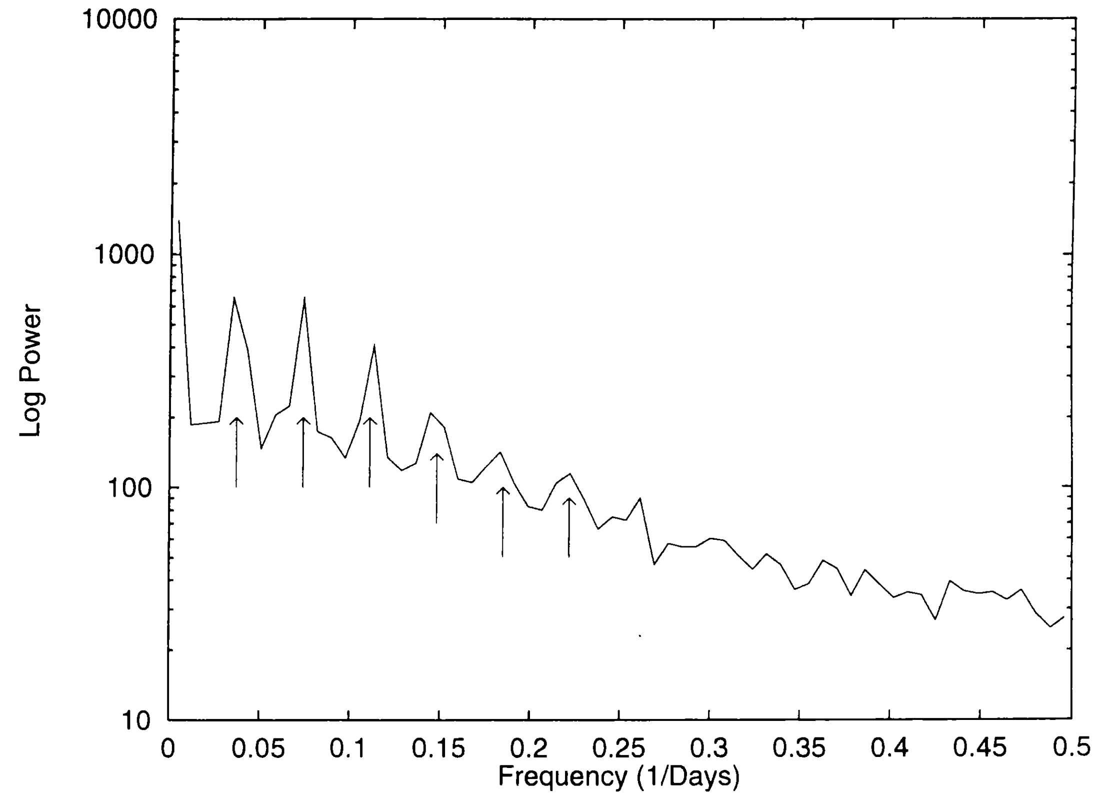


Figure 2.16: The power spectrum of daily average K_p index. The arrows indicate the 27 day solar rotation period and its harmonics. The spectrum was smoothed using $b = 256$.

and $\epsilon(t)$ is a zero-mean white noise process which is continuous in time; a continuous white noise process may seem counter intuitive but the concept can be made mathematically precise in terms of its auto-covariance function,

$$\gamma_\epsilon(u) = \int_{-\infty}^{\infty} \epsilon(t)\,\epsilon(t+u)\,dt \tag{2.3}$$

which will be a delta function, $\gamma_\epsilon(u) = \delta(u)\sigma^2$ where, by definition[2], $Var[\epsilon(t)] = \sigma^2$. The function $T(t)$ is therefore a mathematical representation of a time series of regularly spaced identical pulses with superimposed white noise. To find the power spectrum of $T(t)$, its Fourier Transform must first be calculated.

$$\begin{aligned}\mathcal{F}\left[T(t)\right](k) &= \mathcal{F}\left[\sum_{j=0}^{J-1} S\left(\frac{t-bj}{a}\right)\right](k) + \mathcal{F}\left[\epsilon(t)\right](k) \\ &= \mathcal{F}_S(k) + \mathcal{F}_N(k)\end{aligned}$$

where $\mathcal{F}_S(k)$ is the Fourier transform of the signal and $\mathcal{F}_N$ is the Fourier transform of the noise. The power spectrum $P(k)$ can now be calculated as

$$\begin{aligned}P(k) &= \left(\mathcal{F}_S(k) + \mathcal{F}_N(k)\right)^* \left(\mathcal{F}_S(k) + \mathcal{F}_N(k)\right) &(2.4)\\ &= |\mathcal{F}_S(k)|^2 + |\mathcal{F}_N(k)|^2 + \mathcal{F}_S^*(k)\mathcal{F}_N(k) + \mathcal{F}_S(k)\mathcal{F}_N^*(k) &(2.5)\end{aligned}$$

where * denotes complex conjugation. Since $P(k)$ is actually a continuous random variable its expectation must now be taken. $\mathcal{F}_S(k)$ is the Fourier transform of the deterministic signal so $E[\mathcal{F}_S(k)] = \mathcal{F}_S(k)$; with the same property holding for its complex conjugate and modulus squared. The expectation of the Fourier transform of the noise is given by

$$\begin{aligned}E[\mathcal{F}_N(k)] &= E\left[\int_{-\infty}^{\infty} \epsilon(t)\,e^{ikt}\,dt\right] \\ &= \int_{-\infty}^{\infty} E[\epsilon(t)]\,e^{ikt}\,dt \\ &= 0\end{aligned}$$

because $E[\epsilon(t)] = 0$ since $\epsilon(t)$ has zero mean. Since the same result holds for $\mathcal{F}_N^*(k)$ the last two terms of (2.5) disappear in the expectation. The expectation of the remaining term, $|\mathcal{F}_N(k)|^2$ can be found as follows

$$E[|\mathcal{F}_N(k)|^2] = E\left[\int_{-\infty}^{\infty} \epsilon(x)\,e^{-ikx}\,dx \quad \int_{-\infty}^{\infty} \epsilon(y)\,e^{iky}\,dy\right]$$

[2]Strictly speaking, the variance of a continuous time white noise series is infinite. However, if the variance is defined in terms of the power spectrum, that is in terms of the Fourier transform of $\gamma_\epsilon(u)$, then the variance can be defined as a finite value.

$$\begin{aligned} &= E\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \epsilon(x)\epsilon(x+u)e^{iku}\,dx\,du\right] \qquad , y = x+u \\ &= E\left[\int_{-\infty}^{\infty} \sigma^2\,\delta(u)\,e^{iku}du\right] \\ &= \sigma^2 \end{aligned}$$

using the mathematical definition of the white noise process (2.3). So the expected power spectrum is just

$$E[P(k)] = |\mathcal{F}_S(k)|^2 + \sigma^2$$

that is the noise component simply adds its variance onto the power spectrum of the signal. Turning now to the the signal and using the linear, shifting and scaling properties of the Fourier transform

$$\begin{aligned} \mathcal{F}_S(k) &= \sum_{j=0}^{J-1} \mathcal{F}\left[S\left(\frac{t-bj}{a}\right)\right](k) \\ &= a\hat{S}(ka)\sum_{j=0}^{J-1} e^{ikbj} \\ &= a\hat{S}(ka)\frac{1-e^{ikbJ}}{1-e^{ikb}} \qquad \text{summation of a geometric series} \end{aligned}$$

where $\hat{S}(t) = \mathcal{F}[S(t)](k)$. Finally taking the squared modulus yields

$$|\mathcal{F}_S(k)|^2 = a^2|\hat{S}(ka)|^2\frac{1-\cos kbJ}{1-\cos kb}$$

So that the expectation of the power spectrum is given by

$$E[P(k)] = a^2|\hat{S}(ka)|^2\frac{1-\cos kbJ}{1-\cos kb} + \sigma^2$$

Using the defining properties of the pulse function in (2.2) it can be shown, e.g. Brigham (1988), that the power spectrum of $S(\frac{t}{a})$, that is $|\hat{S}(ka)|^2$, is a decreasing, though not necessarily monotonic, function in k over the range $(0, \infty)$. It is then clear that the smaller the pulse width a becomes, the slower its power spectrum decays. The cosine term reaches maximum when $k = \frac{n2\pi}{b}$, n being a non-negative integer, so that $E[P(k)]$ will show a global maximum at $k = \frac{2\pi}{b}$ with subsequent smaller peaks decreasing in size due to the modulation of $|\hat{S}(ka)|^2$.

To summarize then, this ideal time series $T(t)$, consisting of J pulses each of width a, spaced at a regular interval of b with superimposed zero-mean white noise, has a typical power spectrum as shown in Figure 2.19. The spectrum is shifted up the power axis by the noise to σ^2. There is a fundamental frequency peak at $k = \frac{2\pi}{b}$ with harmonics at multiples of this frequency, decaying

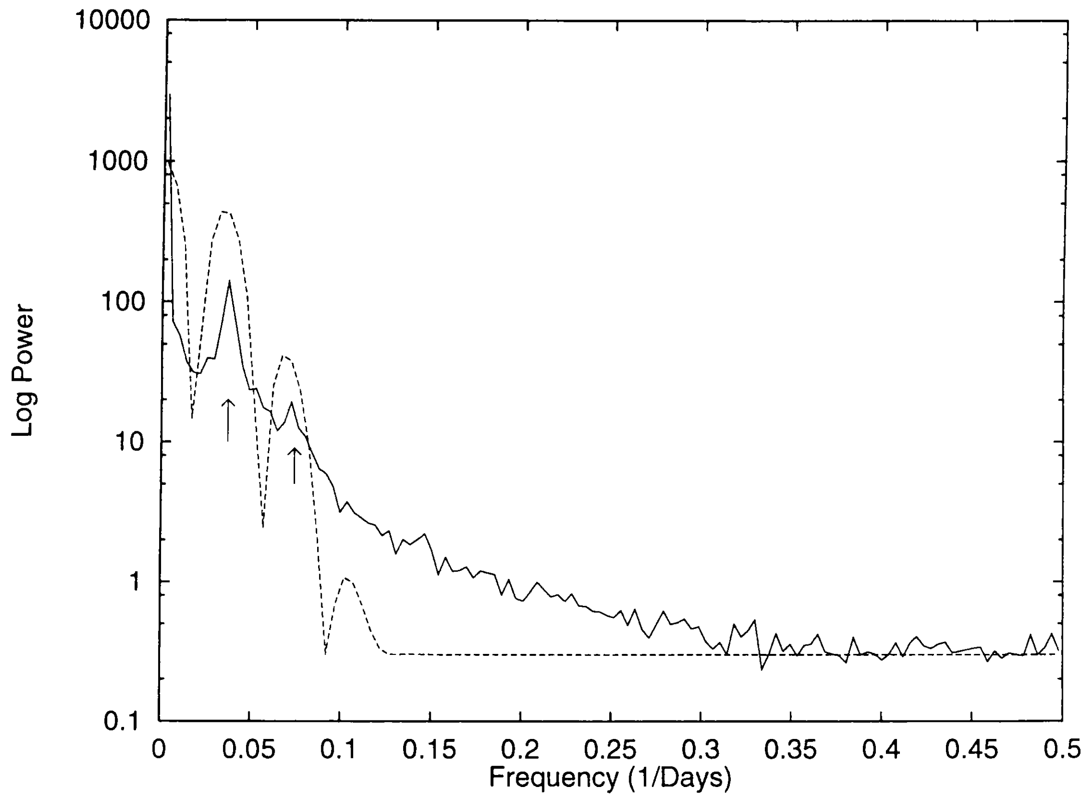


Figure 2.17: The power spectrum of daily sunspot number plotted with the theoretical spectrum with pulse width a=25.

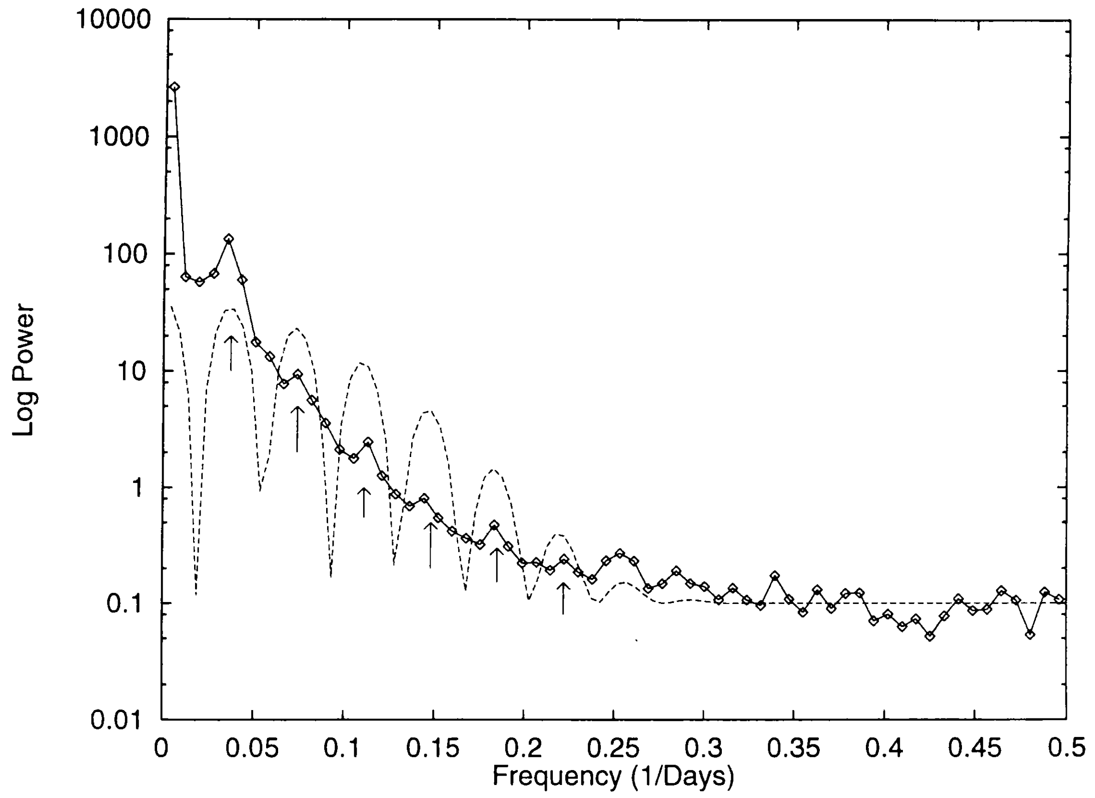


Figure 2.18: The power spectrum of daily solar flux plotted with the theoretical spectrum with pulse width a=10.

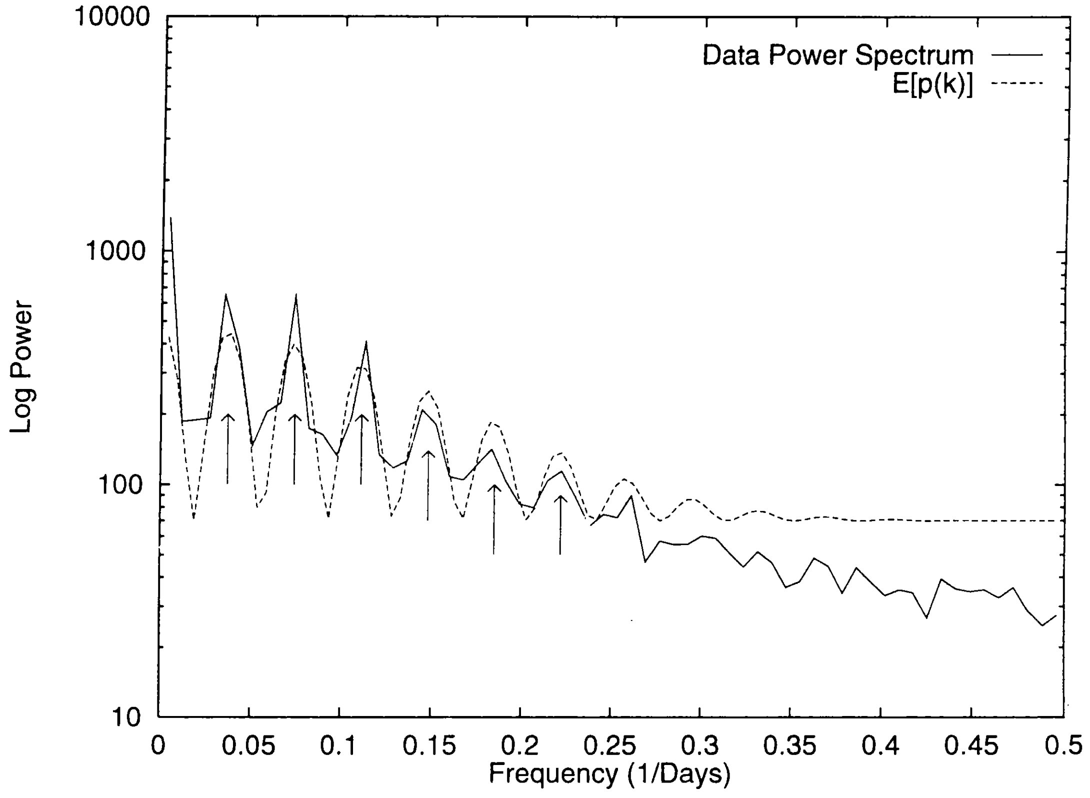


Figure 2.19: The power spectrum of daily average K_p index plotted with the theoretical spectrum with pulse width a=6. The spectrum was smoothed using $b = 256$.

in size at a rate according to the pulse width a; narrow pulses resulting in a slower decay of the harmonic peaks.

To relate this to the observed daily data and its power spectrum consider first the strong harmonic peaks for the K_p spectrum in Figure 2.16. The first peak is at 27 days, the solar rotation period, so $b = 27$. However, the size of the harmonic peaks indicate that the width of the pulses a must be relatively small compared to b. In an attempt to quantify the size of a, albeit rather roughly, I shall assume that the pulses are Gaussian i.e.

$$S(x) = e^{-x^2}$$

so that

$$\left|\hat{S}(k)\right|^2 = \pi e^{\frac{-x^2}{2}}$$

Setting $b = 27$, Figure 2.19 shows that the plot of $E[P(k)]$ with $a = 6$ is consistent with the data. Figure 2.18 shows this for the solar flux, where now $a = 10$. For sunspot number in Figure 2.17, $a = 25$. The plots are not in close agreement for several reasons: the data spectrum is smoothed, the assumption of white noise may not be valid and the assumption of a Gaussian pulse may not be accurate. Also in reality the pulses are neither of constant height nor equally spaced. However these results are sufficient to establish the relative lengths of pulses and are consistent with the nature of the three time series for the following reasons:

- **K_p Index** The presence of 27 day fluctuations in geomagnetic activity has been well-known for many years. As stated in Section 1.7 there are good reasons to believe that they correspond to streams in the Solar Wind that emanate from coronal holes. These would sweep past the Earth every 27 days, on average, but would presumably cause relatively short lived activity because the stream has emerged from a localised region on the Sun. Figure 2.22 shows the mean subtracted data over a year in 1994-1995. It is clear that the negative excursions are longer than the positive.

- **Solar flux** This time, the 27 day rotation period enters directly because the solar flux time series betrays the presence of localised and enhanced radio emission on the visible side of the Sun. These localised regions are of course linked to active regions which survive for many solar rotation periods. However, it is not quite an on-off effect because the contribution from a localised radio source will diminish smoothly as it moves onto the limb and increase smoothly as it moves from the limb to the centre. Thus the positive excursions will be slightly shorter

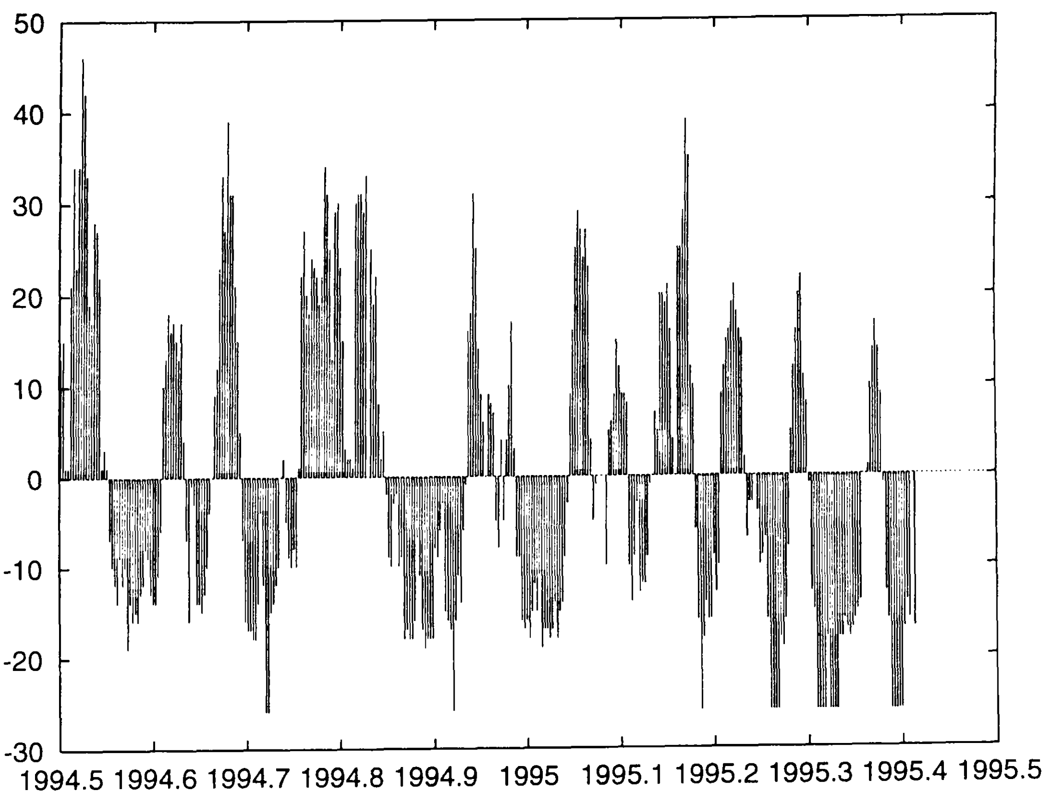


Figure 2.20: An extract of daily sunspot number with the mean subtracted for the range of the plot.

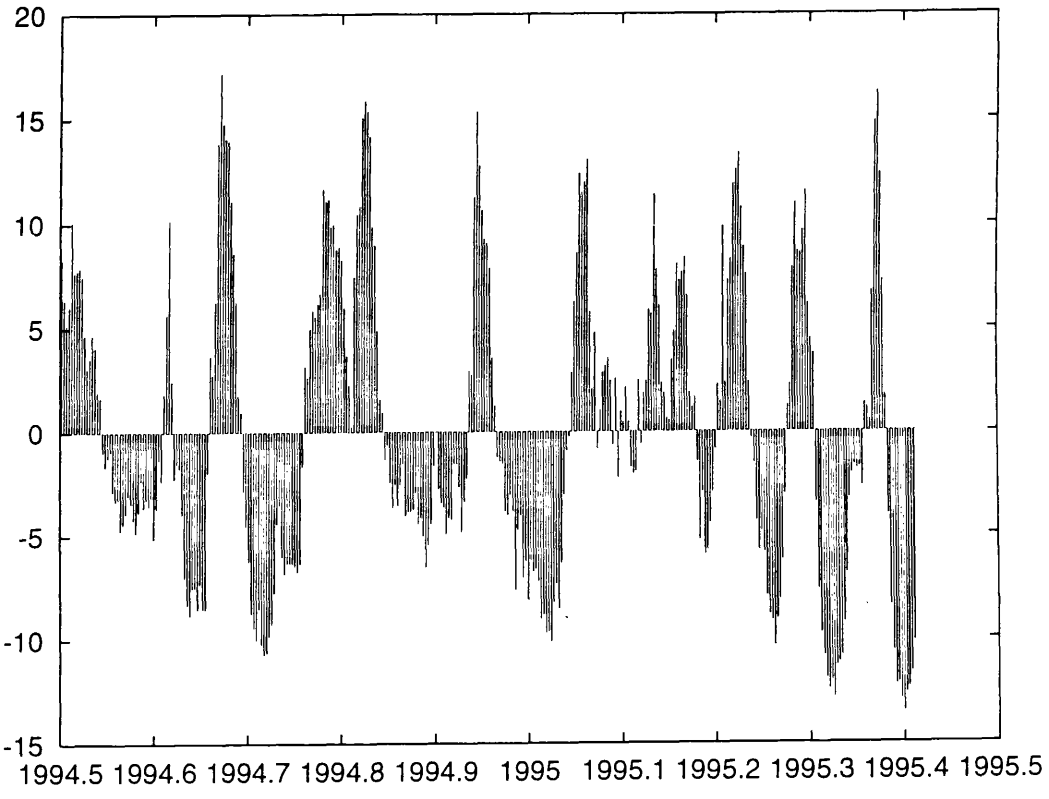
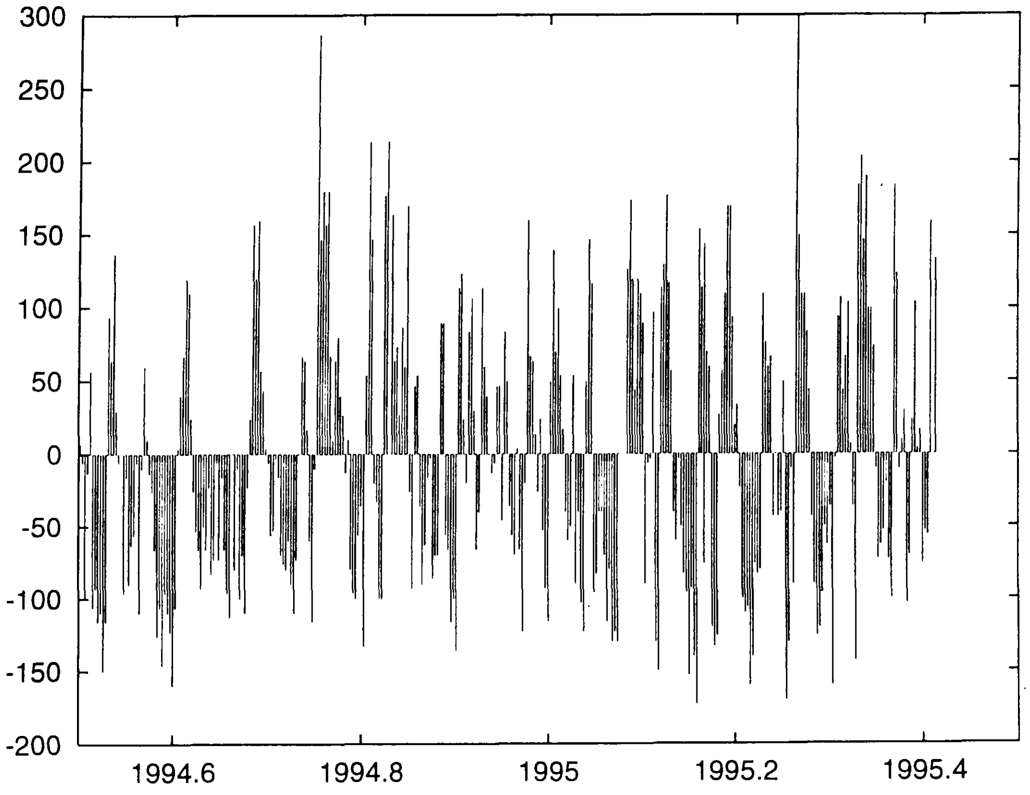


Figure 2.21: An extract of daily solar flux with the mean subtracted for the range of the plot.


300
250
200
150
100
50
0
-50
-100
-150
-200
1994.6
1994.8
1995
1995.2
1995.4

than the negative excursions, on average. This again is (perhaps only suggestively) visible in Figure 2.21.

- **sunspot number** Again like the solar flux there is an on-off type effect, only with sunspot number it is more pronounced because sunspot groups can just "appear" near the limb - either a spot is visible or it is not visible. This means that the sunspot number will, on average, have a larger pulse width than the other two series as evidenced above in the rapid decay of the harmonic peaks. This means that perhaps square rather than Gaussian pulses should have be used in the above comparisons for sunspot number. Since sunspots have lifetimes which are comparable with the solar rotation period the situation is more complicated and the "on average" qualification needs to be borne in mind when looking at data. However, Figure 2.20 does show that the sunspot number variation is more "square" than either K_p or solar flux.

The use of the power spectrum in analysing the 27 day fluctuations for the above time series certainly does show features that are consistent with the nature of the time series as discussed above. What has been achieved that is beyond the ability of, say, the human eye on a daily time series plot, is the provision of some kind of average information about the cyclic variation, specifically in suggesting the average shape of the cyclic variations.

2.3 Stationarity and the Solar Cycle

As indicated in Section 1.2.3 the sunspot number time series is generally regarded as being a *Non-Stationary* time series. However, there is no explicit justification of this in the literature that I have surveyed. This is presumably because there is no conclusive way of showing that a time series is non-stationary. I also find it surprising that I have been unable to find documented attempts at reducing sunspot number to stationarity. This is presumably because no researcher has had much success in the attempt and has consequently thought the results were unpublishable. There is a fairly general non-stationarity test described in Priestley (1988), though no reference is made to the use of this test on the sunspot number. Instead of applying such a test I wish to present some empirical and intuitively persuasive arguments as to why sunspot number is a non-stationary time series. Remember that (weak) stationarity requires three things: constant expectation, constant variance and an auto-covariance function that depends only on lag. I shall defer consideration of the auto-covariance until the next section and concentrate for the moment on whether or not the expectation of sunspot numberis in any way constant.

2.3.1 Why sunspot number is a non-stationary time series

The issue of non-stationarity is really two-fold. First it has to be established that the sunspot number is really non-stationary and secondly it is necessary to investigate the feasibility of reducing it to stationarity. There is no guarantee that the latter is possible, which was one of the main problems with the Box-Jenkin's "classical" approach to fitting linear time series models (see Section 1.2.4). In many ways, even down to the mathematics of it, the prospect of reducing an arbitrary non-stationary time series to stationarity is about as realistic as believing that an arbitrary set of non-linear equations can be reduced to linearity. So in the following arguments I stress not only what makes the sunspot number non-stationary but also what *is* stationary about it.

Let S_n denote yearly sunspot number. If sunspot number was stationary then $E[S_n]$ would be a constant, independent of n. That is

$$E[S_n] = E[S_{n+m}] \tag{2.6}$$

for any value of m. However, if n is the year of a cycle minimum and $m = 4$, say, then it is intuitively obvious that (2.6) is not true. Perhaps then a statement concerning the shape of the cycle is possible; such a statement might be

$$E[S_n] = E[S_{n+\tau}] \tag{2.7}$$

where τ is the mean length of the solar cycle. There are two reasons to suspect that this statement is not true. The first one is that (2.7) seems untrue for cycie maxima, different cycles have different sized maximum. Looking at a cycle with a large maximum, e.g. cycle 18 or cycle 19, it is clear that it is not just the maximum value which is "above average" but almost all the values of the cycle. This could be interpreted as each cycle reflecting some underlying physical state of the sun and its global magnetic field, in which case $E[S_n] \neq E[S_{n+\tau}]$ and therefore the series is non-stationary. However, it could be argued instead that we do not see the expectation value, we see the expectation value plus (supposed) stochastic fluctuations. Explaining this effect as a stochastic variation would require the concept of a mean cycle, given by $E[S_n]$ for $n = 1, \ldots, \tau$ plus some time series which his highly autocorrelated, to ensure that if one point departs greatly from its expectation its neighbours will also. This idea is the basis for the McNish-Lincoln time series model (see Section 1.3.2). The second reason why (2.7) seems unreasonable is the very fact that cycles vary in length. Just how much the length of the solar cycle has varied over its measured history is not certain in my opinion. If the pre-1850 sunspot numbers (reconstructed by Wolf) are to be accepted then the variation in cycle length is very large: from 8 years to 15 years, which correspond respectively to the first and second cycles on record at the start of the 17th Century. Excluding the pre-1850 data, the minimum cycle

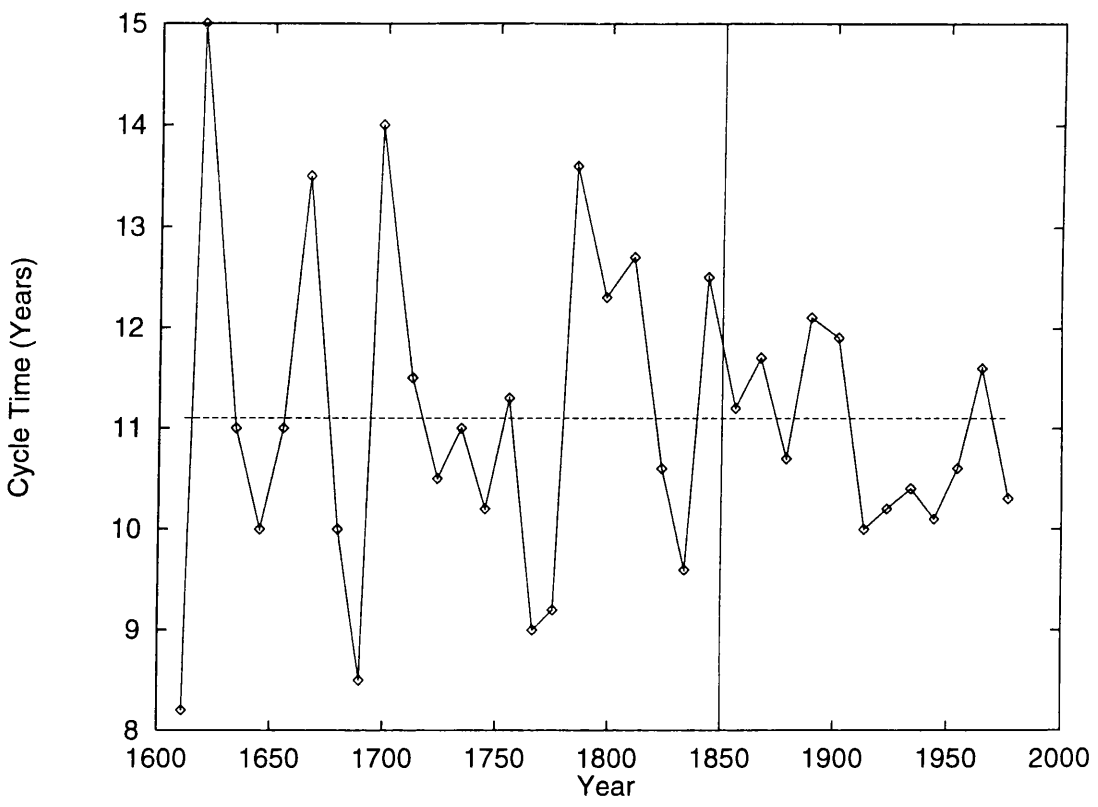


Figure 2.23: A plot of Sunspot Cycle lengths against the start time for each cycle. The horizontal line indicates the mean period, the vertical line divides the modern data from Wolf's reconstructions of older data.

length is 10 years (cycle 15) and the maximum is 12 (cycle 13). So if the pre-1850 reconstructions are to be trusted then the evidence for long-term non-stationarity is strong. However, the transition from large variations in cycle length to small variations in cycle length is actually around or before 1850, making the assertion of the previous sentence somewhat suspect. Figure 2.23 plots the cycle length against cycle start date.

In order to quantify the statistical significance of pre- and post-1850 cycle length variations the F-statistic may be used with the null hypothesis that the pre- and post- data sets both come from distributions with the same variance. Let σ_-^2 be the sample variance of pre-1850 cycle lengths and let σ_+^2 be the sample variance of post-1850 cycle lengths, the F-statistic is defined as

$$F = \frac{\sigma_+^2}{\sigma_-^2}$$

(Note that the pre- and post-1850 sets are of unequal sizes, namely 22 and 12 respectively.) Since the distribution of the F-statistic is well-known, the significance of F can be estimated (accounting for the unequally sized data sets) and used to quantify the meaning of the differing variances. The means and variances for the two sets are:

$$\mu_- = 11.15 \quad \sigma_-^2 = 3.38$$

$$\mu_+ = 10.90 \quad \sigma_+^2 = 0.58$$

The F-statistic obtained was 5.871 with a significance level of 0.004. This significance level can be interpreted as the probability of obtaining an F-statistic larger than 5.871 under the assumption of the null hypothesis that both pre- and post-1850 sets actually have the same underlying variance. The next question is: did the variance really abruptly change at 1850? Figure 2.23 seems to give the human eye this impression, but there is also another way to check this. By sliding the vertical line dividing the pre- and post- data sets in Figure 2.23 back and forth by a number of cycles and recalculating the F-statistic, it is possible to see which dividing date, if any, gives the most significant difference between variances. Figure 2.24 plots the F-statistic against cycle offset. Given that the total number of cycles in the sample is only 34 I have not used offsets which would reduce the number of cycles in either set below 8, hence the range -14 to 4. The minimum value on this plot corresponds to the divide which causes the most significant difference in variances between pre- and post- sets. In fact there are two minima, one at an offset of -14 (the year 1700), with significance level 0.0034, and the other at offset 0 (the year 1850), with significance level 0.0042. The one at -14 might correspond to the fact that Wolf's reconstruction of yearly data only extended back to about 1700 but it is quite possible that at this large offset, with only 9 points in the pre-1700 group, that the F-Statistic is becoming unreliable. The other minimum occurs at 1850 but there are many small values of significance down to offset -4, so the change in variance is not necessarily abrupt at 1850.

The conclusion is that a change in behaviour of cycle length is certainly present in the data and that it occurred sometime between 1800 and 1850. It is therefore entirely possible that this is due to the unreliability in Wolf's sunspot number reconstruction, though it cannot be ruled out on the basis of the above results that the Sunspot cycle went through a period of intrinsic change. Such a change would be consistent with a chaotic dynamo picture (see Section 1.7.1) and the conjecture that there might have been no cycle at all before the Maunder minimum, i.e. that a stable cycle has only been in existence since some time in the 1600s.

It is indisputable that the cycle length does vary to some degree, but is this unquestionably a symptom of non-stationarity? Not necessarily, for precisely the same reasons as the varying size

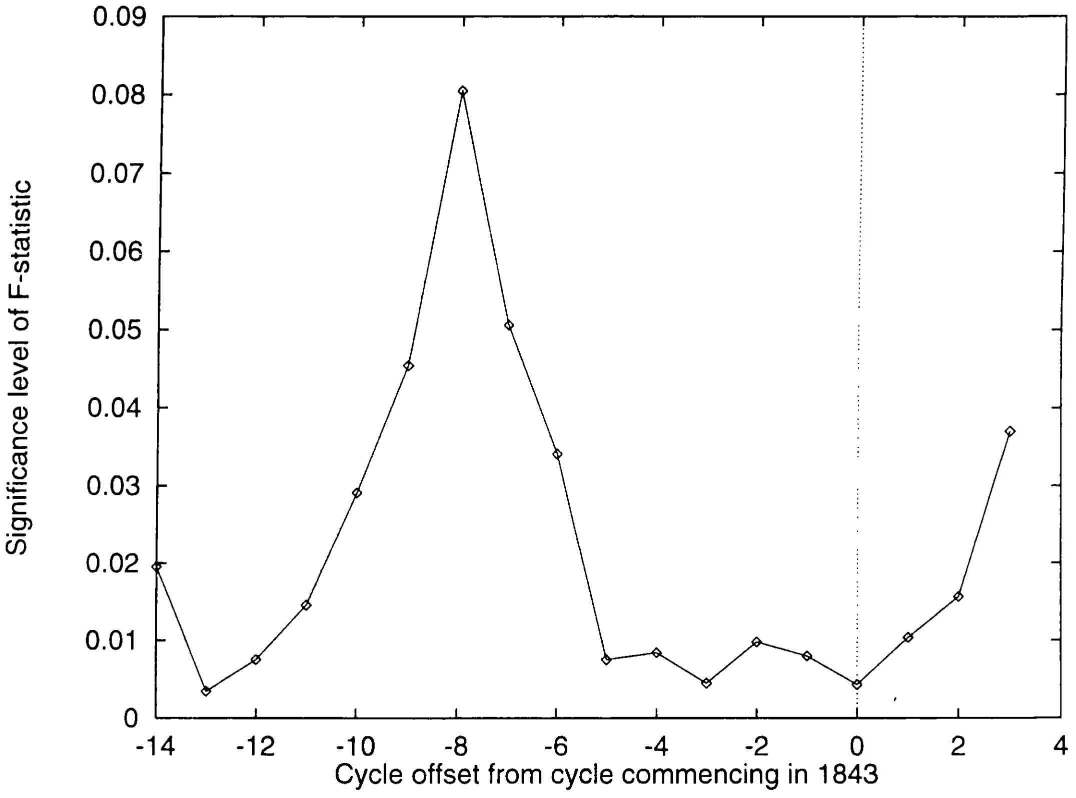


Figure 2.24: The F-statistic plotted for various offsets of the line dividing pre- and post- data in Figure 2.23.

of maxima cannot be viewed as proof of Sunspot Number being non-stationary. Imagine for the moment that the sunspot number has a mean cycle of length τ, so that $E[S_n] = E[S_{n+\tau}]$, then the highly correlated fluctuations proposed above that give rise to the variable cycle size could quite conceivably also cause a minimum to apparently arrive early or late. So the crucial issue that needs to be resolved if sunspot number is to be predicted on the timescale of years is whether or not the existence of a mean cycle is justifiable. If not, then there are non-stationary effects at work, in effect "moving the goal posts" of prediction. The predictability of such non-stationary effects would then become the main issue and it is quite conceivable that some consideration of the global magnetic field structure of the Sun and possibly a dynamo model may be required.

2.3.2 Evidence for a mean cycle

It has already been shown that there was a statistically significant change in the recorded length of the solar cycle in the years surrounding 1850. Since it cannot be established whether this change is intrinsic to the solar cycle or due to distorted reconstruction of data, I shall only use data from 1850 onwards. This means that only 12 complete cycles are available for use in the following test. The hypothesis that I wish to investigate is:

The sunspot number can be regarded as a time series S_t with a mean cycle of constant length τ, such that $E[S_t] = E[S_{t+\tau}]$

However with only 12 cycles of data it is impossible to verify this with any certainty. Instead a weaker, but more testable, hypothesis seems more appropriate:

If $S_{n,k}$ is the sunspot number for the kth point of the nth cycle and there are N cycles then

$$E\left[\sum_{i=1}^{N/2} S_{n_i,k}\right] = E\left[\sum_{j=N/2}^{N} S_{n_j,k}\right]$$

for $k = 0 \ldots \tau - 1$ and for $n_i \in \{1, 2, \ldots, N\}$ such that $n_i \neq n_j$ for $i \neq j$. Basically this says that if the 12 cycles are split into any two groups, the two mean values of each group for each phase point should be the same. The second hypothesis is an implied consequence of the first hypothesis but not the other way around. Therefore the test that is described below can only effectively disprove the first hypothesis:

- Let T_i represent the time between the minima of cycle $i-1$ and i, $i = 1, \ldots, 12$. (I subjectively decide the date of each minimum from the monthly sunspot number time series).

Table 2.1: Probabilities averaged over all possible groups at each of the 10 cycle phase points

1	0.501579
2	0.503997
3	0.500690
4	0.503060
5	0.504791
6	0.506950
7	0.509000
8	0.506109
9	0.503458
10	0.505551
Av.	0.504518

- The estimated length of the mean cycle is then

$$\hat{\tau} = \frac{1}{12}\sum_{i=1}^{12} T_i$$

  and was found to be 130.6 months.

- A start date for the first mean cycle is chosen (I subjectively decide this).

- The minima of the mean cycles now defines where each cycle is supposed to begin. From the above reasoning the apparent minima may not occur at the same time as the mean cycle minima.

- The monthly time series is used to form a new time series such that the new time series gives the average sunspot number at 10 phase points per cycle i.e. each point of the new time series is the average over 13.06 months.

- The 12 cycles are split into two groups of 6, group X and group Y

- The two mean values, one from group X and the other from group Y, for each phase point k are calculated and compared using Student's t-distribution. This yields the probability p_k of obtaining a difference in means greater than the one observed.

- Another different pair of groups are chosen and the last two steps are repeated until there are no groups left.

The probabilities averaged over all possible group pairings, of which there are

$$\frac{1}{2}\left(\frac{12!}{(6!)^2}\right) = 462$$

are given in Table 2.1. A value near one indicates that the means of that phase point are significantly similar in most of the groups. In each case the value is very close to 0.5, and in each case it was found that the spread of values that went to form each average was from 0.01 to 0.99. The conclusion is therefore that the data contains no evidence to verify the second hypothesis above. It is also reasonable to further conclude that the assumption of a mean cycle, as defined above, is not justified. However, as with all statistical methods this result could be the product of an "unlikely" set of data. Also the test that I have made is not objective because it depends on some very subjective decisions, i.e. the definition of the cycle minima and the starting point for the first mean cycle. The latter was checked to some extent by repeating the test for different starting points, but the results of the test were unchanged in each case. Even if the hypothesis is rejected there are many more that could rise in its place, for example a mean cycle that actually varied in length but was constant in shape or vice-versa. Tests like the above could be constructed for these hypotheses but restrictions of time and the fact that there are only 12 "reliable" cycles do not permit me to investigate these further possibilities.

2.3.3 What is stationary about the solar cycle?

There are several qualitative statements that can be made about features of the solar cycle that appear to stay the same, or at least nearly the same, over time. The obvious statement being that "there is a cycle and that it has a period of 10-12 years" (ignoring pre-1850 data). A statement such as "the cycle attack time is always less than the cycle decay time" would also appear to borne out by the data, for example see Table 2.2 which shows that the average attack time for a cycle is 4.8 but the average decay time is 6.2 years. Also for all the post-1850 data the attack time is always shorter than the decay time. Note that in the pre-1850 data there are many exceptions to this rule. Once again this statistically significant change is consistent with either the solar cycle undergoing an intrinsic period of change or distortion due to unreliable reconstruction. I now proceed to quantify some of these properties and examine the significance of them using the *Spearman Rank-Order Linear Correlation Coefficient* described in Press et al. (1994).

The Cycle Shape may be described by the attack time T_a, decay time T_d and the height of the maximum $S_{\max}$. Each one of these quantities varies considerably from cycle to cycle but the question is: is there some aspect of the cycle shape that is preserved? First of all I turn to the relationship between T_a and T_b, where the rule $T_a < T_b$ holds without exception for the post-1850 data. From the results of the linear correlation it appears that there is a significant though not necessarily linear

Table 2.2: Miscellaneous information about sunspot cycles as published by N.O.A.A. The headings refer to the data below them and reflect the conclusions made in Eddy (1977)

Sunspot Cycle Number	Year of Min[1]	Smallest Smoothed Monthly Mean[2]	Year of Max[1]	Largest Smoothed Monthly Mean[2]	Rise to Max (Yrs)	Fall to Min (Yrs)	Cycle Length (Yrs)
		RECONSTRUCTED DATA - UNRELIABLE					
-	1610.8	-	1615.5	-	4.7	3.5	8.2
-	1619.0	-	1626.0	-	7.0	8.0	15.0
-	1634.0	-	1639.5	-	5.5	5.5	11.0
-	1645.0	-	1649.0	-	4.0	6.0	10.0
-	1655.0	-	1660.0	-	5.0	6.0	11.0
-	1666.0	-	1675.0	-	9.0	4.5	13.5
-	1679.5	-	1685.0	-	5.5	4.5	10.0
-	1689.5	-	1693.0	-	3.5	5.0	8.5
-	1698.0	-	1705.5	-	7.5	6.5	14.0
		RECONSTRUCTED DATA - QUESTIONABLE					
-	1712.0	-	1718.2	-	6.2	5.3	11.5
-	1723.5	-	1727.5	-	4.0	6.5	10.5
-	1734.0	-	1738.7	-	4.7	6.3	11.0
-	1745.0	-	1750.3	92.6	5.3	4.9	10.2
1	1755.2	8.4	1761.5	86.5	6.3	5.0	11.3
2	1766.5	11.2	1769.7	115.8	3.2	5.8	9.0
3	1775.5	7.2	1778.4	158.5	2.9	6.3	9.2
4	1784.7	9.5	1788.1	141.2	3.4	10.2	13.6
5	1798.3	3.2	1805.2	49.2	6.9	5.4	12.3
6	1810.6	0.0	1816.4	48.7	5.8	6.9	12.7
		RECONSTRUCTED DATA - RELIABLE					
7	1823.3	0.1	1829.9	71.7	6.6	4.0	10.6
8	1833.9	7.3	1837.2	146.9	3.3	6.3	9.6
9	1843.5	10.5	1848.1	131.6	4.6	7.9	12.5
		FIRST HAND DATA - RELIABLE					
10	1856.0	3.2	1860.1	97.9	4.1	7.1	11.2
11	1867.2	5.2	1870.6	140.5	3.4	8.3	11.7
12	1878.9	2.2	1883.9	74.6	5.0	5.7	10.7
13	1889.6	5.0	1894.1	87.9	4.5	7.6	12.1
14	1901.7	2.6	1907.0	64.2	5.3	6.6	11.9
15	1913.6	1.5	1917.6	105.4	4.0	6.0	10.0
16	1923.6	5.6	1928.4	78.1	4.8	5.4	10.2
17	1933.8	3.4	1937.4	119.2	3.6	6.8	10.4
18	1944.2	7.7	1947.5	151.8	3.3	6.8	10.1
19	1954.3	9.6	1968.9	110.6	4.0	7.6	11.6
21	1976.5	12.2	1979.9	164.5	3.4	6.9	10.3
22	1986.8	12.3	1989.6	158.5	2.8		
Mean		6.0		112.9	4.8	6.2	11.1

1 When observations permit, a date selected as either a cycle minimum or maximum is based in part on an average of the times extremes are reached in the monthly mean sunspot number, in the smoothed monthly mean sunspot number, and in the monthly mean number of spot groups alone. Two more measures are used at time of sunspot minimum: the number of spotless days and the frequency of occurrence of "old" and "new" cycle spot groups.

2 The smoothed monthly mean sunspot number is defined as 13-month running means of monthly mean numbers.

Table 2.3: Results of a Rank-Order Linear correlation test and linear regression on T_a vs T_d. The table lists the range of years used for the test, the number of cycles in that range, the Spearman Rank-order Correlation Coefficient and its significance, and the parameters of a least squares fit to a straight line: $T_d = aT_a + b$, as well as the errors on a and b.

Range	No. Cycs	Corr. Coeff.	Sig. Level	$a(\sigma)$	$b(\sigma)$
$t > 1850$	12	-0.42	0.180	9.34(1.38)	-0.62(0.33)
$t > 1712$	25	-0.49	0.012	9.31(0.85)	-0.62(0.19)
$t > 1610$	34	-0.40	0.018	8.00(0.76)	-0.37(0.15)

anti-correlation between T_a and T_d; the correlation coefficient being about -0.4 with significance levels of 82% for post-1850 data and better than 98% for the older data. The value of the fitted gradient b is also negative within error bounds, again indicating a significant anti-correlation. The reason for the anti-correlation is connected to the fact that the sum of attack and decay times is by definition the cycle length. Figure 2.26 shows a plot of T_a against S_{max} where a very strong linear correlation can be seen by eye, especially in the post-1850 data. Notice that the straight line fits for the data, especially for the post-1850 data, are influenced by the outlier corresponding to the colossal maximum of cycle 19 in 1957. This plot suggests the result encountered later in this thesis that this huge maximum occurred with little or no warning and thus was essentially unpredictable. In Figure 2.27 the values of the Sunspot maxima are plotted with decay times. The Spearman linear correlation coefficient gives a value of about 0.5, but with significance levels of 97% for all available data and 85% for post-1850 data. The correlation is therefore significant but given the amount of data it is hard to determine whether it is actually linear.

The relationship between attack time and the height of the maximum is especially relevant to the prediction of the maximum, though of course the attack time cannot be used for prediction itself for obvious reasons.

Returning to the question of non-stationarity, what can be said about the change in the relationships between these quantities? A glance at Figure 2.25 shows that the pre-1712 data points are scattered far more than the post-1712 data points. However this earlier data is quite unreliable and should really be ignored. If so then the admittedly rather weak correlation appears to be well maintained before and after 1850. The relationship between T_a and S_{max} in Figure 2.26 shows the strongest anti-correlation, with the post-1850 points clearly forming a line (N.B. the fitted line is steepened by the cycle 19 outlier). The pre-1850 data is more scattered and correspondingly results

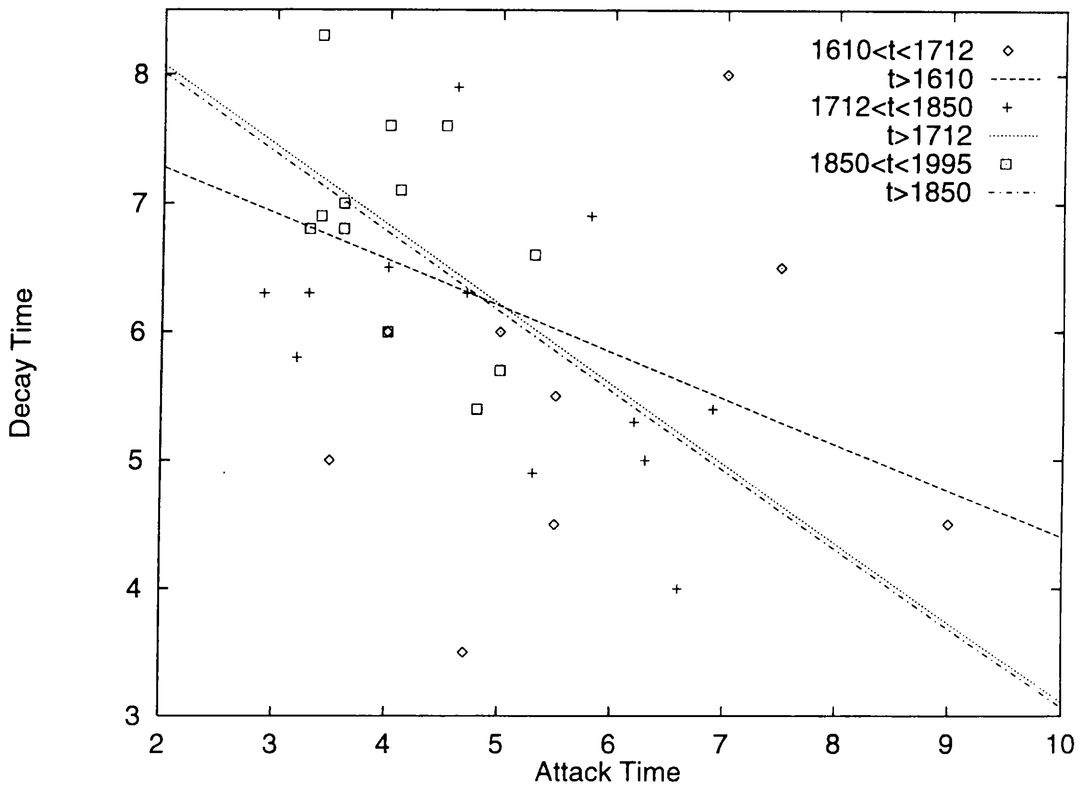


Figure 2.25: A plot of attack times against decay times for Sunspot Cycles. The points represent a plot of the data in the year ranges given in the key. The straight lines show the least square fits from Table 2.3 on all available data since 1610, on data since 1712 and on data since 1850.

Table 2.4: Results of a Rank-Order Linear correlation test and linear regression on T_a vs S_{max}. The table lists the range of years used for the test, the number of cycles in that range, the Spearman Rank-order Correlation Coefficient and its significance, and the parameters of a least squares fit to a straight line: $S_{\text{max}} = aT_a + b$, as well as the errors on a and b.

Range	No. Cycs	Corr. Coeff.	Sig. Level	$a(\sigma)$	$b(\sigma)$
$t > 1850$	13	−0.91	$1.8\,10^{-5}$	308 (36.0)	−47.2(8.84)
$t > 1750$	23	−0.86	$< 10^{-6}$	233 (18.8)	−27.6(4.17)

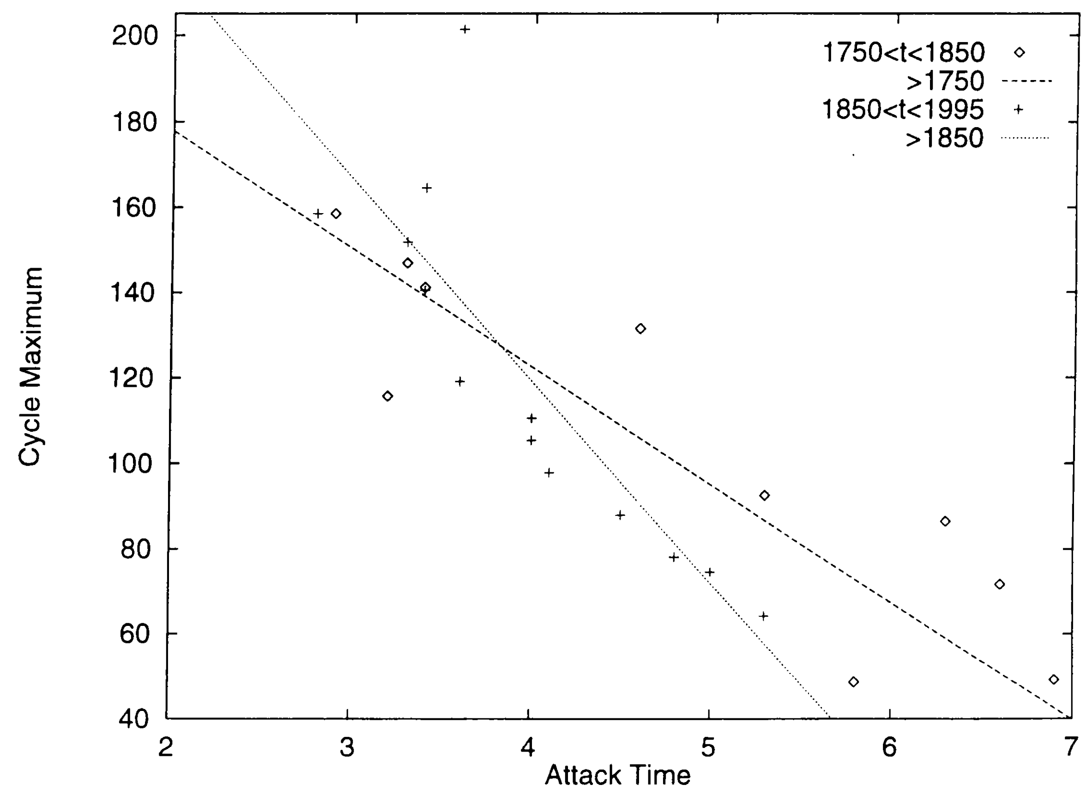


Figure 2.26: A plot of attack times against sunspot maximum for Sunspot Cycles. The points represent a plot of the data in the year ranges given in the key. The straight lines show the least square fits from Table 2.4 on all available data since 1750 and on data since 1850.

Table 2.5: Results of a Rank-Order Linear correlation test and linear regression on T_d vs $S_{\max}$. The table lists the range of years used for the test, the number of cycles in that range, the Spearman Rank-order Correlation Coefficient and its significance, and the parameters of a least squares fit to a straight line: $S_{\max} = aT_d + b$, as well as the errors on a and b.

Range	No. Cycs	Corr. Coeff.	Sig. Level	$a(\sigma)$	$b(\sigma)$
$t > 1850$	12	-0.44	0.15	-14.1 (100)	19 15)
$t > 1750$	22	-0.46	0.03	24 (41)	13(6)

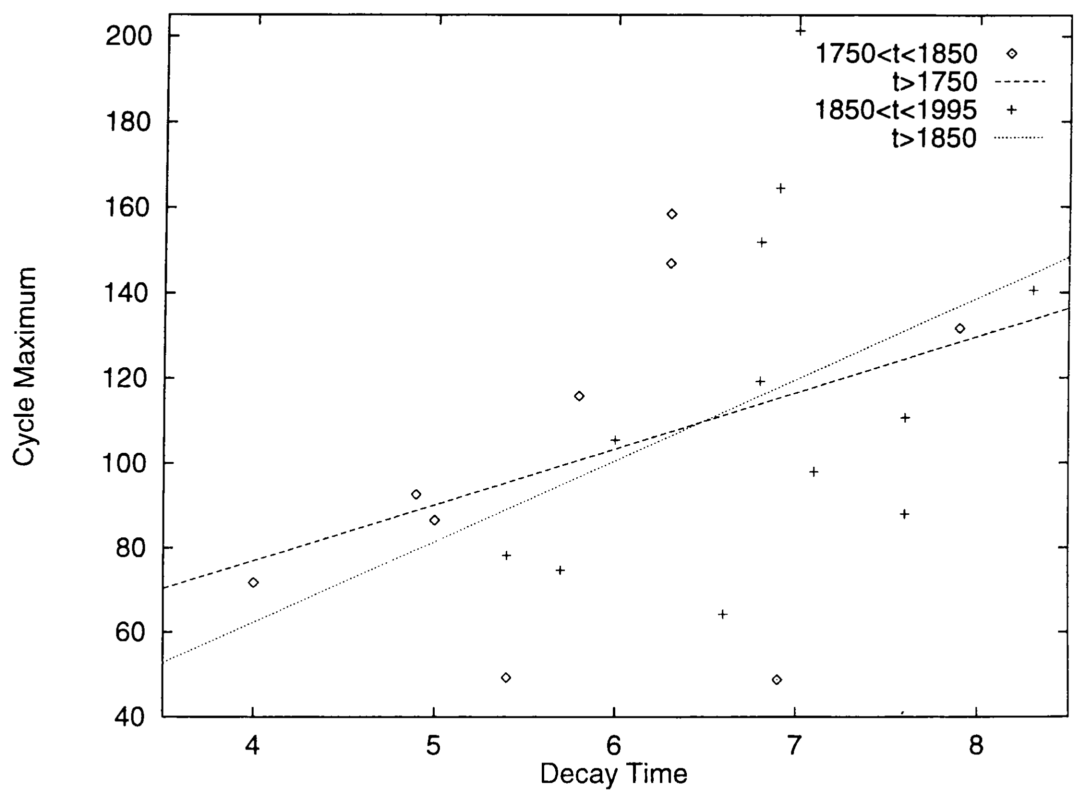


Figure 2.27: A plot of decay times against sunspot maximum for Sunspot Cycles. The points represent a plot of the data in the year ranges given in the key. The straight lines show the least square fits from Table 2.5 on all available data since 1750 and on data since 1850.

in a slightly weaker correlation coefficient when included in the correlation test. The fitted line is also markedly different when the pre-1850 data is included, with the two estimates of gradient (and intercept) failing to fall within the error bounds. There is also some evidence from the plot for believing that the linear relationship is weaker for cycles with large maxima. Finally the relationship between T_d and $S_{\max}$ is the weakest, with only a slight suggestion of a linear relationship. Being so weak, no statement can be made as to whether the relationship between these quantities has changed or stayed the same. So, of the three relationships only one shows a definite change, that of T_a vs. $S_{\max}$, probably because this is the also the strongest. As far as prediction is concerned this result bears good and bad news. It is good news in that it provides hope of predicting the maximum from the attack part of the cycle but is bad news because there seems to be a marked change in this relationship, which means that the earlier data is not really relevant in the construction of prediction schemes.

In conclusion, the following statements relate to features of the solar cycle which can be regarded as being approximately stationary during the period between 1850 and the present day, 1995. It should also be remembered that these results are drawn from Table 2.2 which is based on 13 month running mean smoothed sunspot number.

- There is a cycle that is variable in length and size. The mean length is 10.9 years, with the longest and shortest cycles being 10.0 and 12.1 years respectively. The mean height is 119.7 with the smallest and largest maxima being 64.2 and 201.3 respectively.
- $T_a < T_d$, the attack time is less than the decay time, giving the sunspot cycle its asymmetric appearance.
- T_a is highly anti-correlated with $S_{\max}$, that is large maxima have short attack times.
- T_d is relatively independent of $S_{\max}$, meaning that small cycles are likely to be more symmetric.

2.4 Shorter Timescales: Months and Days

The last section was concerned with the solar cycle and features which varied on the timescale of years. I did not consider the solar flux or the K_p index because they are only available over about five or six cycles. Also, it was not of great interest to investigate the cross-correlations of these time series on the timescale of years since the dynamical and physical effects that may bring about such cross-correlations operate on the timescale of months or days. In this section I shall comment on such effects with a mind to exploit them for prediction purposes later.

2.4.1 Auto-Correlation

The sample auto-correlation of a time series x_t, $t = 0, \ldots, N$ is defined as

$$\rho_k = \frac{1}{(N-k)s^2} \sum_{t=k}^{N} (x_t - \bar{x}_t)(x_{t-k} - \bar{x}_{t-k}) \qquad k = 1, 2, 3, \ldots, K$$

where s^2 is the sample variance given by

$$s^2 = \frac{1}{N} \sum_{t=0}^{N} (x_t - \bar{x}_t)^2$$

This definition really only applies to stationary time series where the mean and variance are constant over time and the auto-covariance and thus auto-correlation depend only on the lag k and not on the particular stretch of data used. However, to demonstrate the non-stationarity of sunspot number the auto-correlation of monthly Sunspot Number is calculated for three stretches of data: 1856-1995, 1856-1923.3 and 1923.3-1995. The dates 1856 and 1923.5 are chosen because they correspond to minima at the beginning and in the middle of the reliable portion of the Sunspot data. Figure 2.28 shows the results, the variances of the three data sets being 2240, 1177 and 2860. Immediately it is apparent from the variances that the data is non-stationary, but also that the auto-correlation for the pre- and post- 1923 data sets is very different indicating that the cycle length is perhaps shorter for the post-1923 data set. Bearing in mind that the auto-correlation depends on the data-set used, Figure 2.28 plots the auto-correlations for monthly sunspot number, solar flux and K_p index using the same range, 1947 to 1995, for each. The plot for the K_p index shows the 6 month oscillation very clearly but the presence of the 11 year solar cycle period is slight, with only a small dip at about 65 months and a gentle rise to 130 months. The reason that sunspot number and solar flux have such similar auto-correlation plots is because they are so highly cross-correlated; a subject addressed in the next section.

2.4.2 Cross-Correlations

The sample cross-correlation of two time series x_t and y_t, $t = 0, \ldots, N$ is defined as

$$\rho_k = \frac{1}{(N-k)\sqrt{s_x^2 s_y^2}} \sum_{t=k}^{N} (x_t - \bar{x}_t)(y_{t-k} - \bar{y}_{t-k}) \qquad k = -K, \ldots, -2, -1, 0, 1, 2, \ldots, K$$

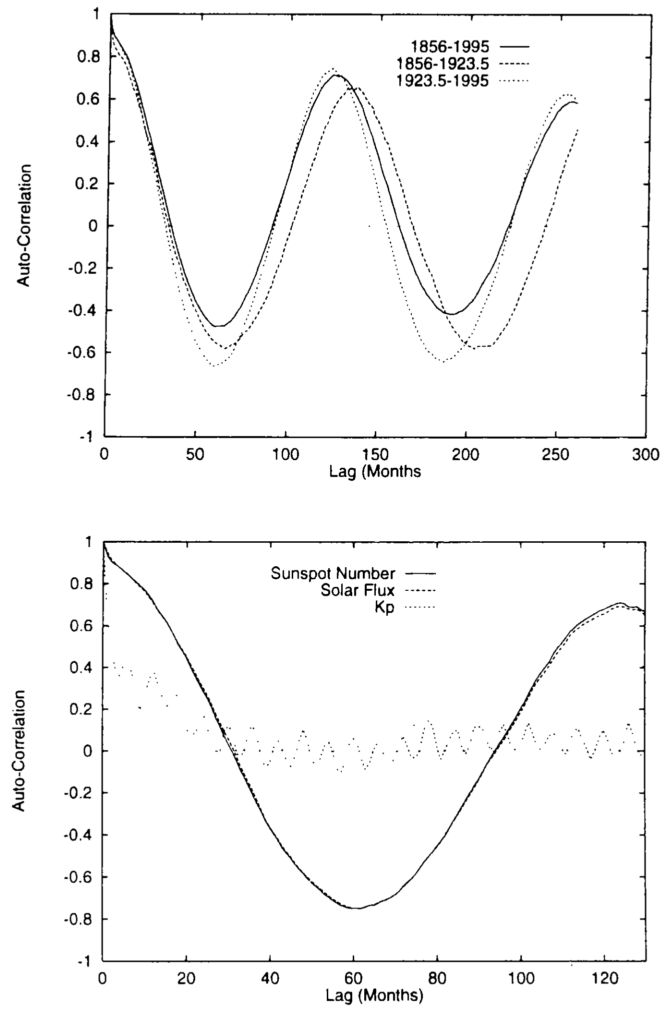


Figure 2.28: Auto-correlations of a) sunspot number over three date ranges and b) Sunspot Number, solar flux and K_p index over the range 1947 to 1995.

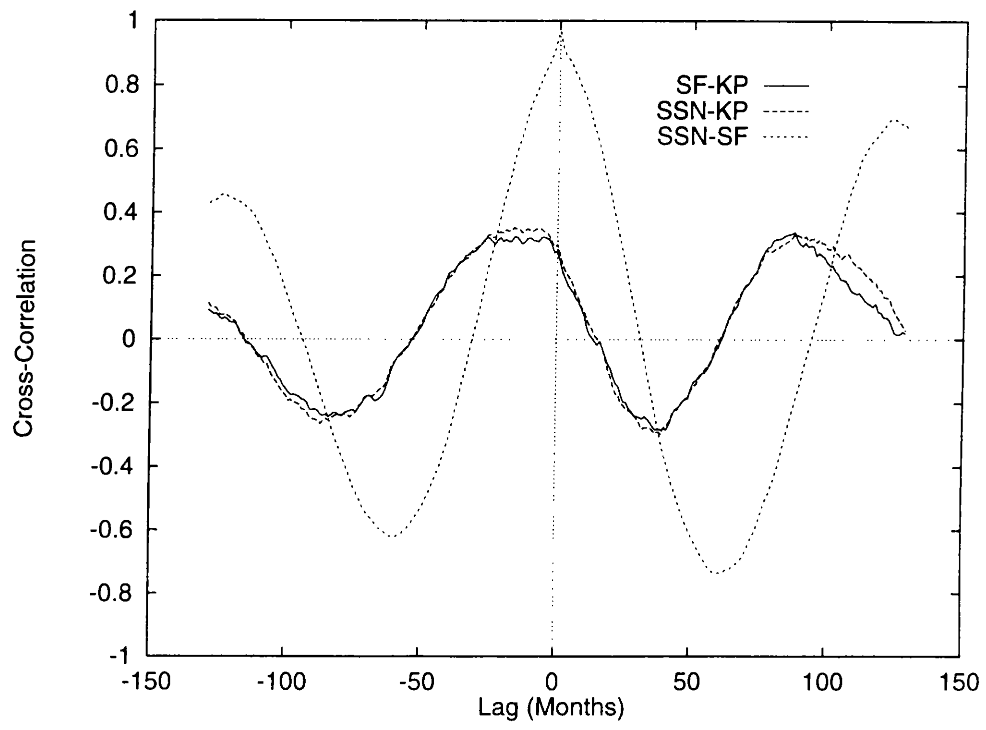


Figure 2.29: Cross-correlations of a) sunspot number with solar flux b) Sunspot Number with K_p index c) solar flux with K_p Geomagnetic Index. The peaks at negative lags in b) and c) mean that features in SSN/SF appear a number of months before they appear in K_p index.

where s_x^2 and s_y^2 are the sample variances. Notice that unlike the auto-correlation above the negative lags are now distinct from the positive lags. Like the auto-correlation, the definition of cross-correlation is really only relevant to the comparison of stationary time series, but again there is no reason why it cannot be computed as long as it is remembered that the plots in Figure 2.29 are for one particular stretch of data, i.e. from 1947 to the present day. Looking at the cross-correlation of sunspot number and solar flux it is immediately apparent that there is a strong correlation between these two series, with the strongest correlation occurring at zero lag. This result means that there is little prospect for predicting solar flux using the the recent history of sunspot number (or indeed vice versa) because neither contains any obvious forewarning of how the other is about to behave. This is confirmed later on in Section 6.3, where the two series are used in combination as inputs to a neural network predicting sunspot number. The Spearman Linear Correlation coefficient is

0.986 with a very high significance level. I have also performed a linear regression on the sunspot number/solar flux relationship finding it to be:

$$SF = 0.907(\pm 0.008) * SSN + 60.065(\pm 0.73)$$

which contains the quoted relationship in Section 1.7 within its error bounds.

The cross-correlations of either solar flux or sunspot number with K_p index show a much weaker correlation peaking at lags of between 5 and 25 months. Looking at the monthly plots (e.g. Figures 2.2 and 2.4) of the time series the reason for this lag can be attributed to the K_p index peaking months to years after the Solar cycle. The Spearman Linear Correlation coefficient is only 0.350 (but with a very high significance) at a lag of 15 months for sunspot number with K_p index, so the correlation is not a strong (linear) one. Also when interpreting the cross-correlations it is important to realise that the peaks at non-zero lags in the sunspot number or solar flux/K_p cross-correlations do not mean that the Sun's influence takes many months to "reach" the Earth, i.e. the association is not necessarily a direct causal one. In this case the conclusion must be that the solar activity just after the solar maximum is responsible for maximum geomagnetic disturbance at the Earth.

2.4.3 Daily Data and the Wavelet Transform

There is a new problem to be faced when looking at daily data: the sheer amount of it. One method that can be used to summarize the data is the Fourier transform, which was used earlier to look at the most prominent feature of the daily data: the effect of the solar rotation. However, this method was not ideal because the 27 day pulses in the data are unevenly spaced, of variable duration and of variable height. Another class of linear transformation, the *wavelet transform*, offers the ability to resolve only features of a certain time scale. Before going on to look at the daily data I shall describe the wavelet transform in more detail. Since the wavelet transform is a relatively new subject in Mathematics and since it is currently being plundered by scientists and engineers, the definitions and jargon associated with it are varied and seemingly contradictory. However, in what follows I refer to only two varieties of the wavelet transform: the continuous wavelet transform and the discrete (Daubechies) wavelet transform. I shall use the former to introduce the concept of the wavelet transform and use the latter in work with the data.

The **continuous wavelet transform** of a function $s(t)$ is defined as

$$CWT(a,b)\,[s(t)] = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t)\,\psi\left(\frac{t-b}{a}\right)\,dt$$

where $\psi(t)$ is called the wavelet and is a localised function in t. If the admissibility condition is satisfied, that is

$$c_\psi = \int_o^\infty \frac{|\Psi(\omega)|^2}{\omega} d\omega$$

is finite, where

$$\Psi(\omega) = \mathcal{F}[\psi(\sqcup)]$$

then it can be shown that, e.g. Chan (1995), the function $s(t)$ can be expressed as a linear transformation of the continuous wavelet transform

$$s(t) = \frac{1}{c_\psi} \int_{-\infty}^{\infty} \int_0^{\infty} CWT(a,b)\psi\left(\frac{t-b}{a}\right) da\, db \tag{2.8}$$

For the admissibility condition to be satisfied and to avoid the integral over ω being divergent $\Psi(0) = 0$, so that

$$\int \psi(t)\, dt = 0$$

To lend meaning to the above mathematical statements, consider (2.8). This double integral is just superposing infinitely many wavelets of different sizes a and positions b in order to re-form the function $s(t)$. If the function is a Gaussian, i.e.

$$s(t) = \exp\left(-\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

then the wavelet transform will peak around the point $(a = \sigma, b = \mu)$ and tend to zero away from that point. The exact behaviour of the transform depends of course on the wavelet function $\psi(t)$. It is also possible to perform only a partial reconstruction of the function $s(t)$. Let this partial reconstruction be called s_{a_1,a_2}, then

$$s_{a_1,a_2} = \frac{1}{c_\psi} \int_{-\infty}^{\infty} \int_{a_1}^{a_2} CWT(a,b)\psi\left(\frac{t-b}{a}\right) da\, db$$

so that only detail between the scales a_1 and a_2 is included in the reconstruction, it is common to say the other scales have been "filtered out". However, unlike the Fourier transform the basis functions of the Wavelet transform, i.e. $\psi\left(\frac{t-b}{a}\right)$, are not necessarily orthogonal which means that there is some "leakage" between neighbouring scales. A similar filtering process can be performed with the position parameter b, with similar problems. The non-orthogonality leads to some redundancy in the continuous wavelet transform, evidenced by the fact that the inverse transformation requires a double integration.

In this way the wavelet transform can be used to decompose signals into components of different timescales. What is meant by "timescale" is not clear for the reasons given above and really depends on the particular wavelet used. In fact it has been said that the wavelet transform is a mathematical microscope whose optics are determined by the wavelet used.

Initially I used the continuous wavelet transform with discrete time series by calculating the transform numerically at discrete values of a and b. It is a surprising side effect of the non-orthogonality of the wavelet transform that with sufficiently fine quantization of a and b, perfect reconstruction is still possible. However, since the selection of the range and quantization of a and b is rather arbitrary, a different approach, that of the **Discrete (Daubechies) wavelet transform** was found to be preferable. This wavelet transform is intrinsically numerical and actually has orthogonal wavelets but these Daubechies wavelets, as they are called, have no continuous counterpart. It was invented by the French Mathematician Daubechies, e.g. Daubechies (1990), and has been documented in Press et al. (1994) complete with the necessary routines to perform the algorithm. In many ways it is the analogue of the fast Fourier transform and like the fast Fourier transform it can only deal with data vectors that are an integer power of two data elements long.

As an example Figure 2.30 shows the last few Sunspot cycles and their decomposition into three scale bands using the discrete wavelet transform. Since there were $512(= 2^9)$ months of data used, there are 9 detail scales which were then summed in groups of three to give the three detail scales plotted. Since the transformation is linear the sum of the three components give a perfect reconstruction of the original signal.

Returning to the daily data, Figure 2.31 shows plots of daily sunspot number, daily solar flux and daily K_p index that have been wavelet filtered, removing any long (>2 months) and short (< 5 days) term variations. Each peak in the wavelet filtered time series corresponds to a peak visible in the original data - no spurious features are introduced. However, due to the filtering out of short timescales it must be realized that some neighbouring peaks are merged together. Notice particularly how clearly the wavelet filtering of the daily K_p index brings the 27 day fluctuations out of the large short timescale variations in the series.

Now that wavelet filtered versions of the time series have been obtained, with emphasis placed on the 27 day variations, it is possible to investigate the cross-correlation of the time series on that time scale. Before doing so it is important to ask whether the process of wavelet filtering can introduce any spurious features in a cross-correlation. On an intuitive level it appears not, since the wavelet filtering does not introduce any spurious peaks. A more formal and mathematical statement may well be possible to derive and in Chapter 7 I discuss how this might be done. If it is accepted that the process of wavelet filtering does not create spurious correlations then its advantage is obvious

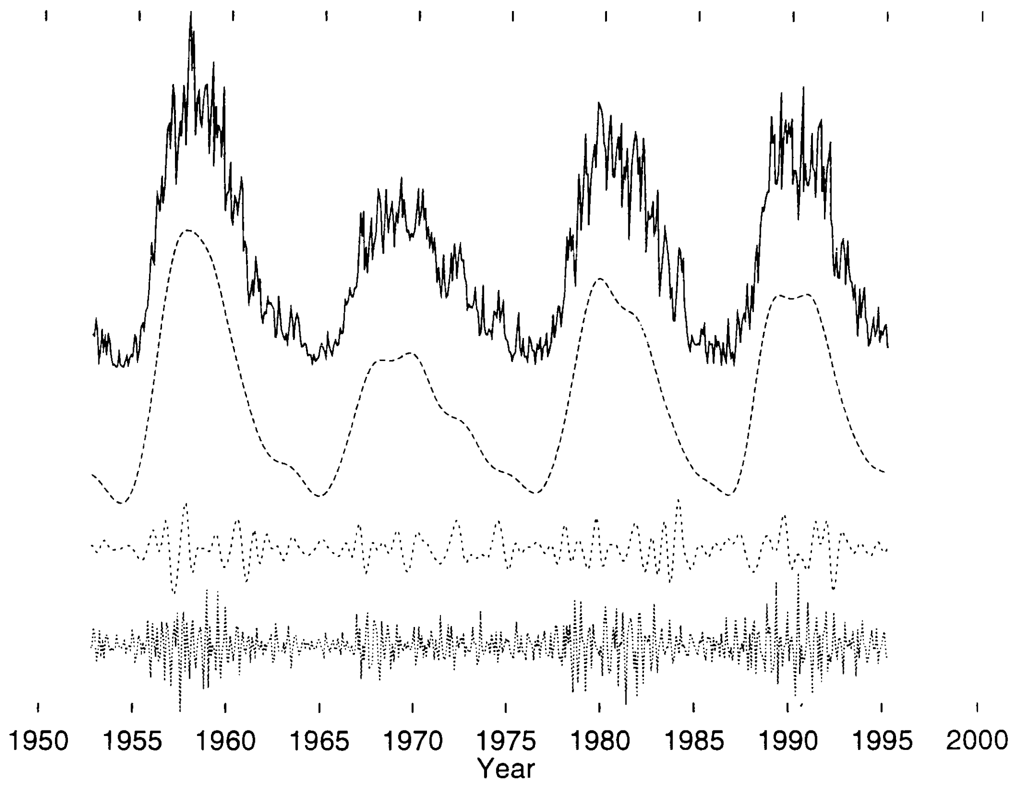


Figure 2.30: Sunspot number and its decomposition into three levels of detail using the discrete (Daubechies) wavelet transform. The three components have been offset along the y-axis of the plot for the sake of clarity.

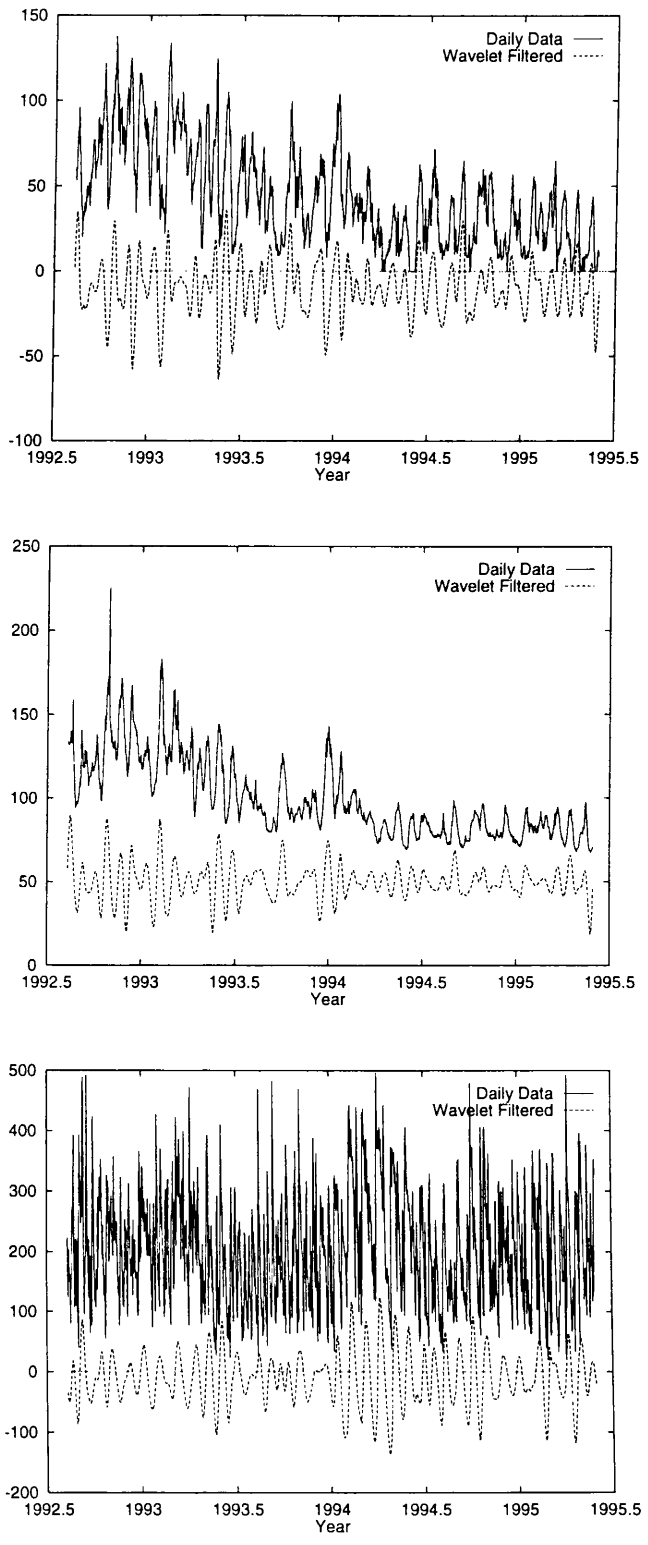


Figure 2.31: Wavelet filtering of a) sunspot number b) solar flux and c) K_p index. These time series have been filtered with the discrete Wavelet transform so that only timescales of around the 27 solar period are present.

from Figure 2.32. This plot shows the cross-correlation as estimated using the raw daily data and also wavelet filtered daily data. It is apparent that the peaks of correlation in both are in agreement but that the wavelet filtered curve is much smoother with much more prominent peaks. Essentially the effects of the solar cycle are filtered out and the long-term cross-correlation of the 27 day pulses is highlighted.

Since sunspot groups are associated with active regions and active regions are associated with enhanced radio emissions, the cross-correlation between sunspot number and solar flux should be very high on all timescales. The cross-correlation plot in Figure 2.33 confirms this both for the raw daily data and for the wavelet filtered data, also clearly showing the presence of the solar rotation period. This further tests the fact that wavelet filtering of time series does not adversely affect the cross-correlation, e.g. by creating spurious peaks. The next question to ask is if there is any detectable delay between the 27 day pulses in the sunspot number and the 27 day pulses in the geomagnetic activity. Figure 2.34 shows the calculated cross-correlations for many stretches of data. Looking at the first plot where the data range is from 1950 to the present day it is seen that K_p index seems to lag the sunspot number by about 3 or 4 days. However, is this true for all subsets of this data range, perhaps the delay is different at solar maxima and minima and this 3 or 4 day lag is only an "on average" result. The next plot shows the results of calculating the cross-correlations of data around solar minima. In this case all but two of the minima examined show a strong anti-correlation at zero lag. Turning to the cross-correlations at solar maxima, the correlation is now positive at zero lag with the cross-correlation peaking at lag times of typically about 6 days. These cross-correlations suggest that during solar minima the K_p index is anti-correlated with sunspot number whereas at solar maxima the K_p index seems correlated with the sunspot number from several days before. Looking at the typical spacing between peaks in Figure 2.32 and by simply comparing daily plots of data, as in Figure 2.35, it seems that the connection between these time series is related to the solar rotation period. In Section 1.7 the 27 day variations in the geomagnetic field were said to be caused by streams emanating from coronal holes on the Sun. It was also mentioned that such regular disturbances are most clearly seen during solar minima because at times of high solar activity the geomagnetic disturbances are so numerous and complicated. This goes some way to explaining why the cross-correlations at maxima and minima differ in character. Also, at solar minima, why is the K_p index about 14 days out of phase with the 27 day cycles in the sunspot number? Given the speed of the streams that cause the geomagnetic disturbances, which will typically be larger than 600 km/s, and even allowing for the effect of spiraling, the stream material will take at most about

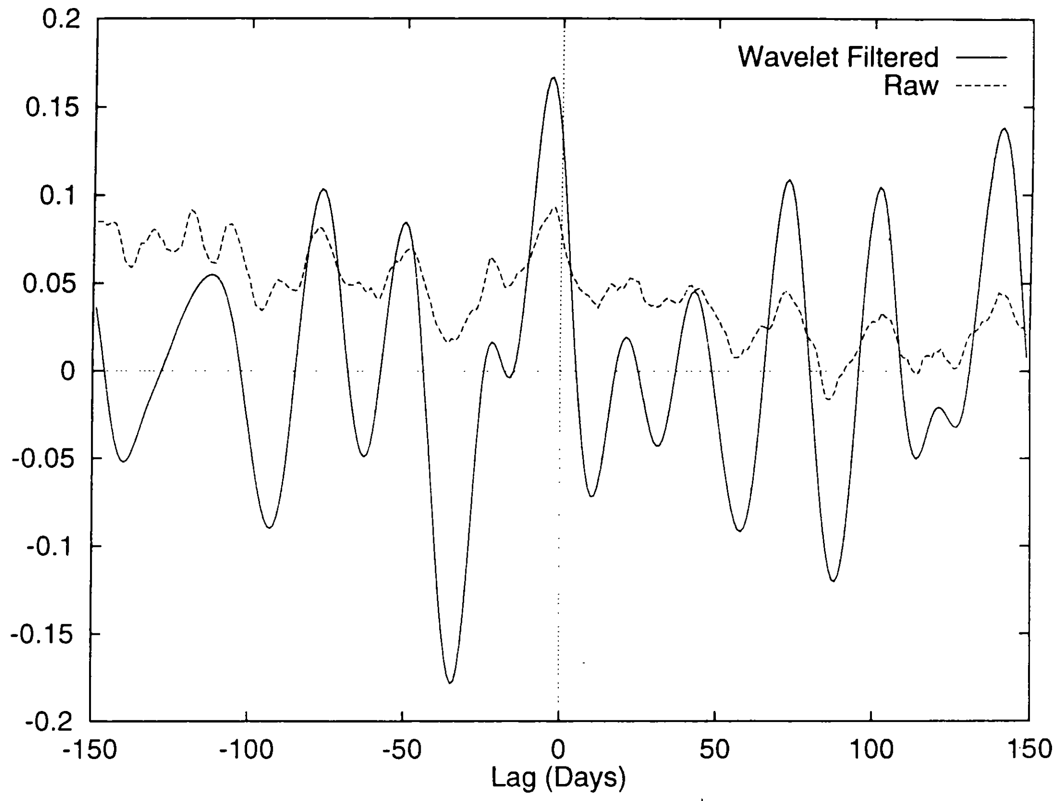


Figure 2.32: The cross-correlation of daily sunspot number and K_p index calculated using raw unprocessed values and using the wavelet filtered versions of the two time series. The wavelet filtering removes variations on timescales that are very different from the 27 day solar rotation period ($<$ 5 days and $>$ 2 months). The stretch of data is from 1976 to 1987, roughly the whole of cycle 21.

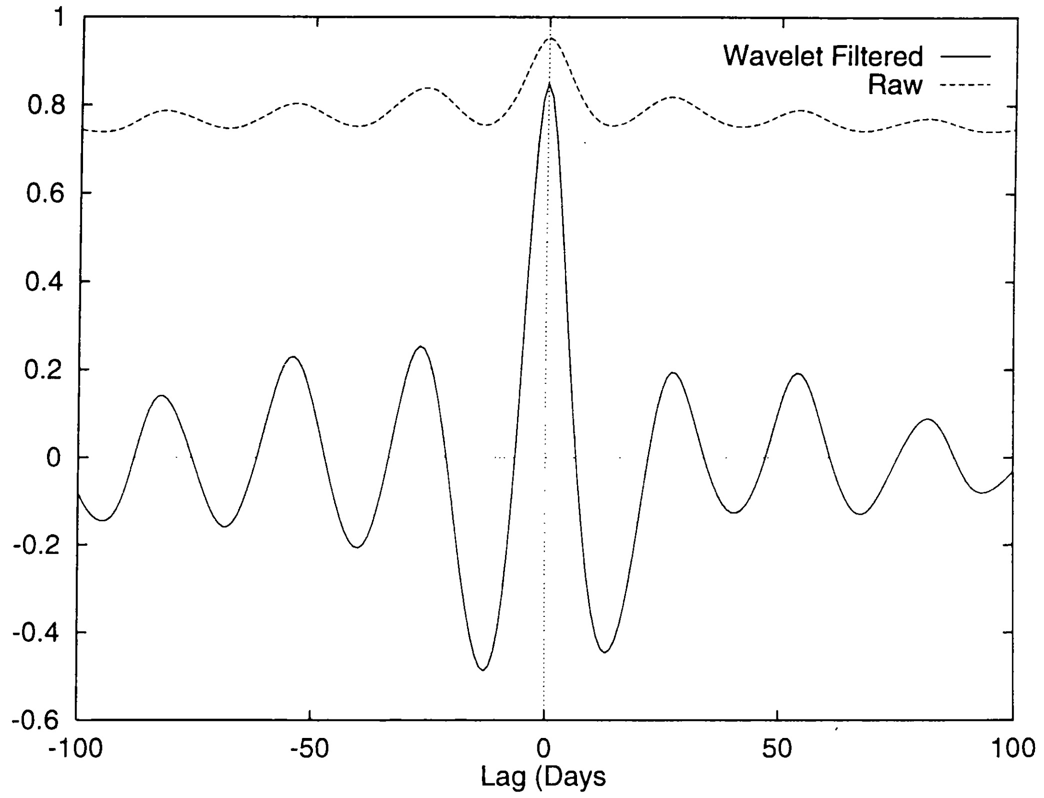
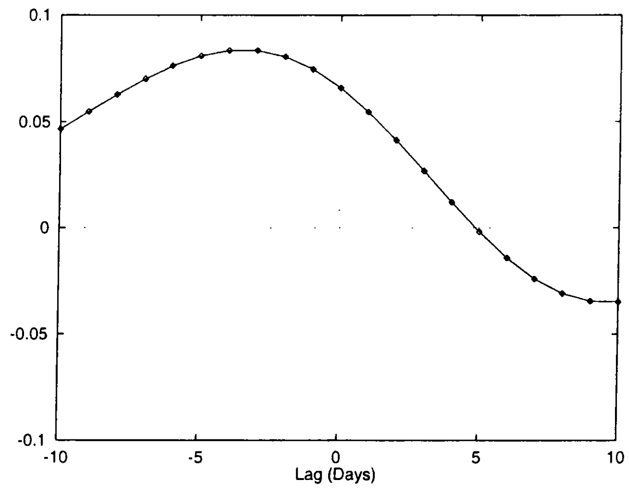
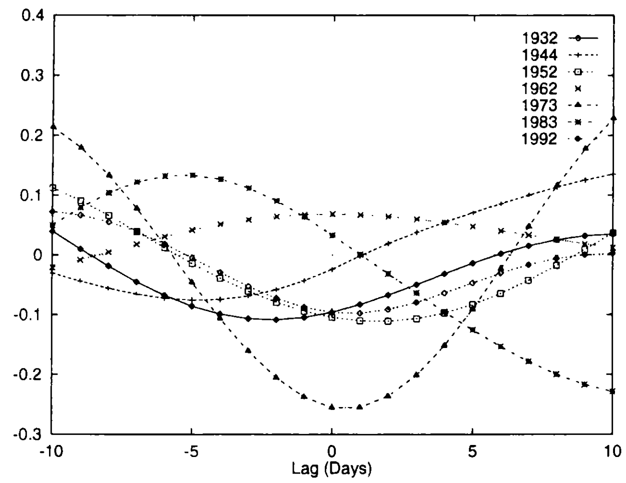


Figure 2.33: The cross-correlation of daily sunspot number and solar flux calculated using raw daily values and using the wavelet filtered versions of the two time series. The wavelet filtering removes variations on timescales that are very different from the 27 day solar rotation period ($<$ 5 days and $>$ 2 months). The stretch of data is from 1950 to 1995.





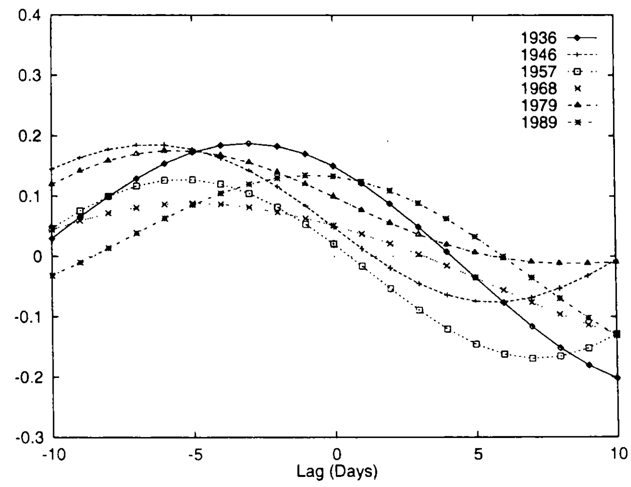


Figure 2.34: Plots of data showing the cross-correlation of sunspot number and K_p for a) all the data since 1950 b) minima c) maxima. The time series have been wavelet filtered beforehand to filter out timescales of < 5 days and $> 2 months$.

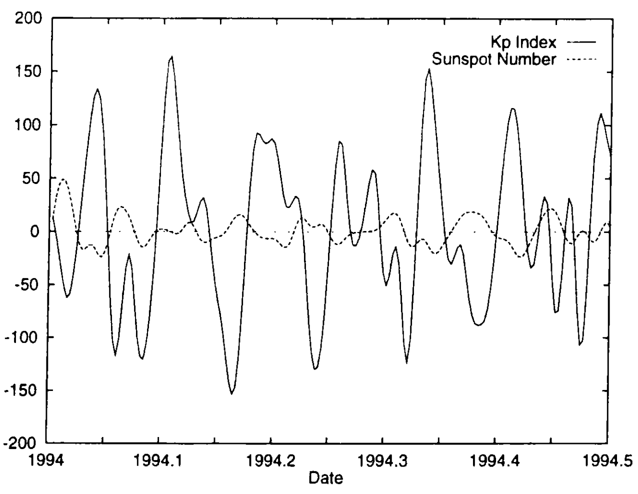


Figure 2.35: Extracts from the daily sunspot number and K_p index that have been wavelet filtered. The anti-correlation of the two series 27 day fluctuations is apparent.

four days to arrive at the Earth's orbit. It seems unlikely, therefore, that the sources of the streams, the coronal holes, and the Sunspot groups are located nearby on the Sun. At solar maximum there are many flares and coronal mass ejections, both of which are associated with active regions. Since such activity is also thought to be responsible for geomagnetic disturbances, e.g. Hargreaves (1992), and remembering that Sunspot groups are usually found in active regions, the positive but lagged cross-correlation is not unexpected.

2.5 Concluding Remarks

The work in this chapter was not intended to be an exhaustive investigation of every aspect of the time series concerned. It was concerned with providing an impression of matters such as: the accumulation and formatting of the data; the search for periodicities; the nature of any periodicities; the non-stationarity of sunspot number; the stationary aspects of sunspot number; the auto-correlation of the time series; the cross-correlation of the time series especially in relation to the Sun's influence on the Earth; and the use of wavelet transform in analysing time series. Apart from being of intrinsic interest in itself, I feel that this work provides a familiarity with the data that will directly and indirectly fuel the prediction initiatives in the following chapters.

Chapter 3

An Investigation of Neural Networks

"It's all a question of mind over matter. What is mind? Doesn't matter. And what of matter? Never mind." *Rev. Francis Bigger*

This chapter explores how a Feed Forward Neural Network (FFNN) can represent a desired input-output relationship which is described by an analytic function. Most of the work is concerned with 1-H-1 networks, as the more general case, that of I-H-O networks learning O functions of I variables, can be viewed in terms of the results obtained in this simpler case. Specifically I shall attempt to answer two questions: how FFNNs can represent functions; and how FFNN can be *taught* to represent functions. There is a crucial distinction between these two issues.

By the end of the chapter I will have shown three important results:

- How to "analytically" train FFNNs to fit an arbitrary analytic function to an arbitrarily small error of fit. This will be achieved by reducing the problem of finding the network weights to the solution of a system of linear equations.
- How the number of hidden neurons H relates to this.
- The (in)stability of Back-propagation

Table 3.1: Some applications of feed forward neural networks.

Galaxy Classification	Storrie-Lombardi et al. (1992)
Time Series Prediction	Lapedes and Farber (1987)
Gluon-Jet Detection	Lonnblad et al. (1990)
Meteorite Fragment Pairing	Conway and Bland (1995) - In progress

3.1 Fitting and Learning to Fit Functions

To recap, as shown in Figure 1.3, the FFNN consists of layers of neurons, the neurons in each layer giving their outputs only to the neurons in the layer immediately above. The bottom layer neurons are called inputs, the top layer neurons are called outputs and the layers in between are called hidden layers. The parameters that determine the strength of the connections between neurons are called the *weights* or *weight connections*. (1.10) and (1.11) give the equations for calculating the hidden and output layer neurons from the inputs. To set up the desired output response to a set of inputs the weights must be set appropriately - the process of setting these weights is called "training". At this stage a digression is appropriate in order to answer a very good question levelled at many people who use and/or study neural networks: "What is the difference between FFNNs and other data fitting techniques such as polynomial fits or generalised linear regression?" The answer, in general, is that there is no difference, a FFNN is another data-fitting technique. In fact the definition of a FFNN can even be extended to encompass many other data-fitting techniques [1] However, the FFNN according to its most popular meaning is the one where the activation functions are all sigmoidal, with the possible exception of the output activation function being linear (see Section 1.4.1). In this sense the FFNN is just a certain class of data-fitting algorithm, one for which a great deal of popular excitement and interest has been generated. The interest is mostly generated by the historical association with (vastly over-simplified) models of the brain, still evident in the jargon associated with neural networks. The other explanation for their popularity is that they actually work, in that they are capable of successfully tackling a wide variety of problems, see Table 3.1.

3.1.1 Fitting functions

An *I*-*H*-*O* FFNN can represent an O dimensional function of I variables, by definition. To start with I shall consider a 1-H-1 FFNN. If given an input A, the equations for the (values of the) hidden

[1]Consider a 1-H-1 FFNN with linear output activation functions and where the input activation function to the n^{th} hidden neuron is x^n.

neurons are

$$B_j = g(w_j A) \qquad j = 1, \ldots, H \tag{3.1}$$

and for the output neuron

$$C = G\left(\sum_{j=1}^{H} W_j B_j\right) \qquad k = 1, \ldots, O \tag{3.2}$$

If this network is required to fit a function $f(x)$ over the range $[a, b]$ then the error function

$$E(\mathbf{w}, \mathbf{W}) = \int_a^b (C(x) - f(x))^2 \, dx$$

must be minimised with respect to the weights, which parametrise $C(x)$. The situation for fitting multi-dimensional functions of many variables is similar, with the error consisting of a sum of square errors for each output and a multiple integration over the range of interest for each input. As stated in Section 1.4.1, Cybenko (1989) showed that when using a one hidden layer FFNN with sigmoidal activation functions, and for a continuous set of functions $\mathbf{f}(\mathbf{x})$, the error function can be made arbitrarily small given enough hidden neurons. There is no guarantee that any minimisation method can actually find the required weights, though. Of course when using a FFNN numerically on a computer the integral over x in the error has to be replaced by a sum over sample points in the range $[a, b]$. In this case the behaviour of the network's fit to $f(x)$ between the sample points is undetermined. It might be expected that sparse sampling and many hidden neurons (and therefore many weights) will encourage a network's fit of a function to behave wildly between sample points, as would happen in the fitting of a polynomial where the order is larger than the number of sample points. In practice, however, a FFNN trained by back propagation seems to be immune from this problem, irrespective of the sampling and the number of hidden neurons. To demonstrate this Figure 3.1 shows several 1-H-1 FFNN fits to $0.5 + 0.2 \sin 3x$ over $[0, 1]$. Back propagation was used to train the NNs for a fixed number of iterations in each case. The first thing to notice is how poor the fits are given the large number of iterations, this is because back propagation converges very slowly on the best fit. This is confirmed by the fact that by using smaller values of learning parameter, and allowing training to continue, it is possible to keep improving the fit ad nauseam. However, as training progresses the possible improvement in the fit per iteration rapidly becomes very small, as seen by the training graph in Figure 3.1d. What is very surprising is that a 1-25-1 network with a training set of 50 points failed to converge on a fit at all. Depending on the initial value of the learning rate it would either fit a straight line to the training set or produce outputs that were extremely large. The reason for this is not clear, but as I will show later it is not difficult

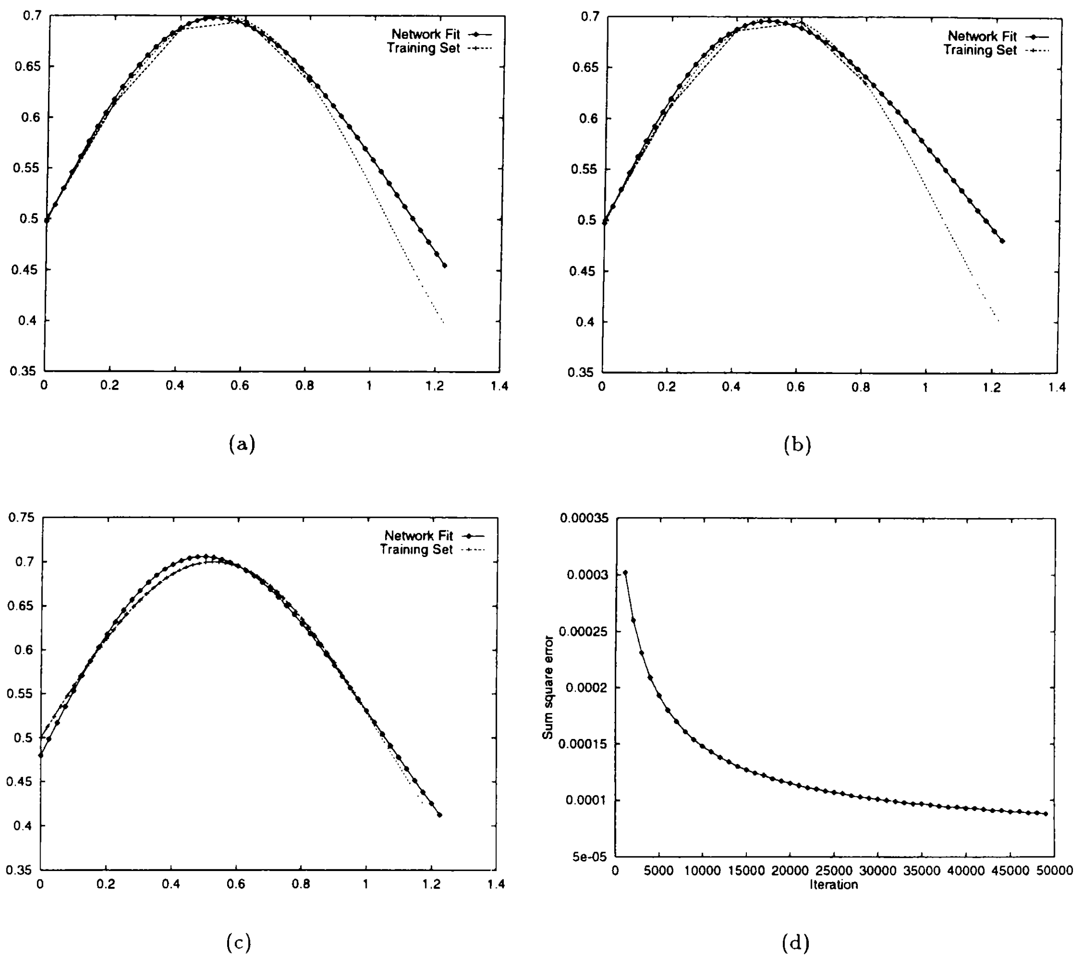


Figure 3.1: Three FFNN fits to the function $0.5 + 0.2\sin 3x$ over the range $[0, 1]$: a) 1-5-1 with 5 training points b) 1-25-1 with 5 training points c) 1-5-1 with 50 training points. Plot d) shows how the square error summed over the training points decreases per iteration for case a).

to find examples where a network is capable of fitting a function but back propagation cannot train a network to fit that function in a stable fashion. In answer to the original question that motivated these tests, the 1-25-1 network produces a surprisingly smooth fit to the five points of its training set, certainly as good as that of the 1-5-1 network. So the addition of $2 \times (25-5) = 40$ extra weights, and therefore fitted parameters, even on a sparsely sampled function does not produce the dramatic variations that can be found if, say, a 50^{th} order polynomial was fitted to the five points. The reason for this lies in the fact that polynomials are formed by adding together functions which tend to $\pm\infty$ at large x where as the sigmoids used in a standard FFNN go to 0 and 1 at $-\infty$ and $+\infty$.

Of course, the issues raised from the above tests are not all confined to the function $0.5+0.2\sin 3x$. I have simply used this function as an example. In more general terms, what can be said about a FFNN's ability to fit functions and back propagation's ability to train networks to fit functions?

Turning to the first part of the above question, consider a 1-H-1 FFNN fitting an $M-1$ order polynomial in the range $[\alpha, \beta]$, then

$$f(x) = \sum_{n=0}^{M-1} a_n\, x^n$$

Let the input activation function $g(x)$, which is not necessary sigmoidal, be analytic in the range $|x| < x_0$, so that it is possible to perform a Taylor expansion about the origin:

$$\begin{aligned} g(x) &= \sum_{n=0}^{\infty} g^{(n)}(0)\frac{x^n}{n!} \\ &\equiv \sum_{n=0}^{\infty} g_n x^n \qquad |x| < x_0 \end{aligned}$$

substituting this into (3.1) and using (3.2), with linear output activation function $G(x) = x$, gives the network's output as

$$\begin{aligned} C(x) &= \sum_{i=1}^{H} W_i \sum_{n=0}^{\infty} g_n w_i^n x^n \\ &= \sum_{n=0}^{\infty} \left(\sum_{i=1}^{H} W_i w_i^n \right) g_n x^n \end{aligned}$$

so to obtain a perfect fit the weights must be chosen to equate the polynomial coefficients of $f(x)$ with those of $C(x)$

$$g_n \sum_{i=1}^{H} W_i w_i^n = \begin{cases} a_n & n < M \\ 0 & n \geq M \end{cases} \tag{3.3}$$

On top of this, if this representation is to be valid then

$$|w_i \alpha| \quad < \quad x_0$$

$$|w_i\beta| < x_0$$

must be satisfied for $i = 1, \ldots, H$ so that the Taylor expansion power series of $g(x)$ is convergent. As things stand there appear to be an infinite number of equations in $2H$ unknowns. Consider first solving the equations for $n < M$ only,

$$\sum_{i=1}^{H} W_i w_i^n = A_n \qquad \text{where } A_n = \frac{a_n}{g_n} \tag{3.4}$$

If H is chosen so that $H \geq M$, and values for all the w_i's and $H - M$ of the W_i's are chosen, then (3.4) forms a completely determined system of M linear equations in M unknowns. Since we have complete freedom in choosing the w_i's and the $H - M$ W_i's, it seems reasonable to assume that they can be chosen to make the system of equations consistent. With this being the case, then the problem is now reduced to a solution of a set of linear equations in the remaining M W_i's. However, the fit is not complete because the network's output function $C(x)$ still contains non-zero terms of order M and above. The error in performing the fit as described thus far is therefore

$$\begin{aligned} E(x) &= C(x) - f(x) \\ &= \sum_{n=M}^{\infty} \left(\sum_{i=1}^{H} W_i w_i^n \right) g_n x^n \end{aligned}$$

Now, the aim is to reduce this error to below some tolerance for the fit to become "good enough". As shall now be shown this can be achieved by choosing the values of the w_i terms that are "small enough". Reducing w_i must reduce the $W_i w_i^n$ terms because, from the system of equations in (3.4), it can be shown that

$$W_i w_i^n \rightarrow 0 \quad \text{as} \quad w_i \rightarrow 0 \quad \text{if} \quad n \geq M$$

(A sketch of why this is plausible is given in the Appendix Section A.4).

The above has tacitly assumed that the coefficients of the Taylor expansion of the input activation function $g(x)$ are non-zero, i.e. $g^{(n)}(0) \neq 0$, for $n < M$. Also it was assumed that $H \geq M$, otherwise it would be impossible to avoid inconsistency in the set of linear equations in the output weights. Therefore it has been implicitly proved that, given the above assumption about the input activation function, a 1-H-1 network fitting an N^{th} order polynomial will need at least as many hidden neurons as there are polynomial coefficients, i.e. $H \geq N + 1$.

As an illustration of the above, I now demonstrate the "analytic" training of a 1-H-1 network on a linear function and prove that it is indeed possible to make the error of fit arbitrarily small by reducing the size of the input weights. Having demonstrated the method I will then use it to construct an example that highlights the fact that back propagation is not a stable algorithm for training FFNNs to fit certain functions.

Example: A 1-H-1 network fitting $f(x) = a_1x + a_0$. Here M=2, so there are two equations

$$\begin{aligned}\sum_{i=1}^{H} W_i &= A_0\\ \sum_{i=1}^{H} W_i w_i &= A_1\end{aligned}$$

and once the w_i's have been chosen, there are H unknowns, namely the W_i's. In order to make these equations completely determined $H-2$ of these unknowns need to be chosen, so set $W_i = \gamma$ for $i > 2$. The equations to be solved are then

$$\begin{aligned}W_1 + W_2 &= A_0 - \gamma(H-2)\\ w_1W_1 + w_2W_2 &= A_1 - \gamma\sum_{i=1} w_i\end{aligned}$$

To avoid linear dependence and inconsistency $w_1 \neq w_2$, so with this in mind solving for W_1 yields

$$W_1 = \frac{(A_0 - \gamma(H-2))\,w_2 - \left(A_1 - \gamma\sum_{i=3}^{H} w_i\right)}{w_2 - w_1}$$

with a similar expression for W_2. I now prove that it is possible to achieve arbitrarily small errors by reducing the size of the input weights. The error on the network's output will be

$$E(x) = \sum_{n=2}^{\infty}\sum_{j=1}^{H} W_j w_j^n g_n x^n$$

begging the question of how $W_j w_j^n$ behaves as all the w_i's are brought nearer to zero. Let $w_1 = \lambda w_2$, where $\lambda \neq 1$, then

$$\begin{aligned}W_1 w_1^n &= \frac{(A_0 - \gamma(H-2))\,\lambda^n w_1^{n+1} - \left(A_1 - \gamma\sum_{i=3}^{H} w_i\right) w_1^n}{(\lambda - 1)w_1}\\ &= w_1^{n-1}\frac{(A_0 - \gamma(H-2))\,\lambda^n w_1 - \left(A_1 - \gamma\sum_{i=3}^{H} w_i\right)}{(\lambda - 1)}\end{aligned}$$

which means that

$$W_1 w_1^n \rightarrow 0 \quad \text{as} \quad w_1, w_2 \rightarrow 0 \quad (n \geq 2)$$

finally since the upper bound on the absolute error reduces as the input weights are reduced, as can be seen from

$$|E(x)| < \sum_{n=2}^{\infty}\sum_{j=1}^{H} \left|W_j w_j^n\right| |g_n x^n|$$

then $|E(x)| \rightarrow 0$ as $w_1, w_2 \rightarrow 0$ because $|W_i w_i^n| \rightarrow 0$ for $i = 1, 2$. For $i > 2$ $|W_i w_i^n| = \gamma w_i^n$ can be reduced freely by either reducing γ or the w_i's for $i > 2$. Thus it has been proved that the error of the fit can be brought arbitrarily close to zero, for any non-zero A_0, A_1, any $H \geq 2$. It has also been proved that the smallest network capable of fitting $f(x) = ax + b$ is a 1-2-1 network, because it is possible to obtain a fit with the following variables set to zero: $w_i = 0$ for $i > 2$ and $\gamma = 0$.

3.1.2 Learning to fit functions

In this section I wish to study how a network might learn to fit functions. The underlying theme of this work is that a FFNN might be capable of fitting a function but back-propagation, or for that matter any other training algorithm, may be unable to realise this capability. The following will concentrate only on standard back-propagation, which can be thought of as least squares minimization by gradient descent.

In practice, the choice of learning rate, ϵ, affects the number of iterations required to complete training. If ϵ is very small, learning will be slow and the error will decrease very gradually, thus requiring many more iterations to complete training. If ϵ is very large then in some instances the error might change wildly with each iteration while in others it may still decrease albeit very slowly. The following discussion will provide some insight into how the training procedure operates and how the choice of learning rate might affect the end result.

Minimizing x^2 by gradient descent provides an illustration of the above statements and is shown in Fig. 3.2. For this simple case it is easy to show that ϵ must be less that unity for convergence and that $\epsilon = 1$ will cause oscillation between the two values $\pm x_0$, where x_0 is the starting value. If the learning rate is very much less than 1, or very slightly less than 1, then convergence to the minimum is very slow.

To draw closer to the case of training a NN consider the function:

$$f(y) = Wg(wy) - d - y \tag{3.5}$$

The problem here is to bring $f(y)$ as close to zero as possible in some range of y, say y_1 to y_2, with respect to W, w and d; where $g(x)$ is the sigmoidal activation function defined in (1.12). Minimizing $f(y)$ is equivalent to training a 1-1-1 feed forward neural network with a linear output activation function with threshold[2] to simulate the function $h(y) = y$ over the given range of y, where y is the input and $Wg(w(y - c)) - d$ is the output. This example demonstrates two things: firstly the

[2]Remember a threshold is equivalent to a weight connection to a neuron which has a fixed value

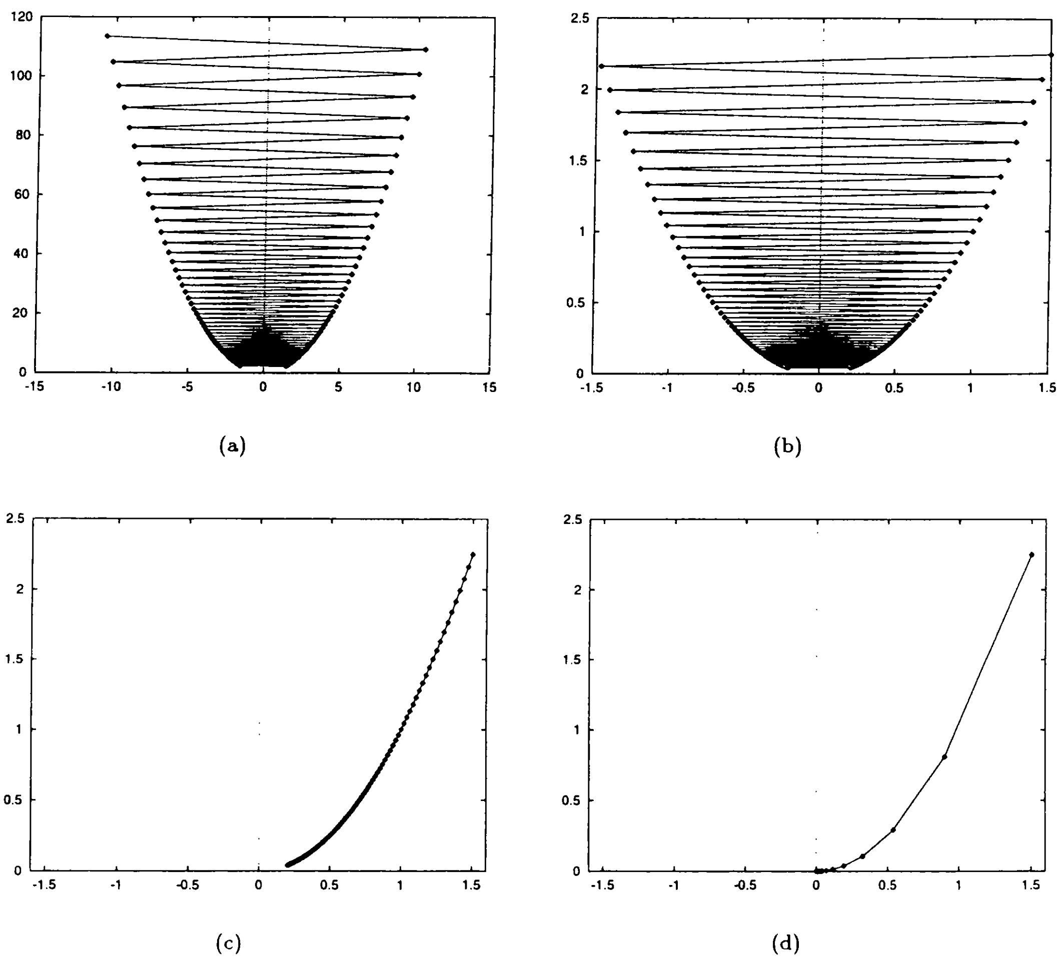


Figure 3.2: This is a simple analogy to show how the value of the learning parameter, ϵ, can affect the progress of training. The function $f(x) = x^2$ is minimized using gradient descent using a) $\epsilon = 1.1$, b) $\epsilon = 0.9$, c) $\epsilon = 0.01$ and d) $\epsilon = 0.2$. The plots show the value of f(x) versus the value of x for each iteration, the starting value in each case is $x = 1.5$.

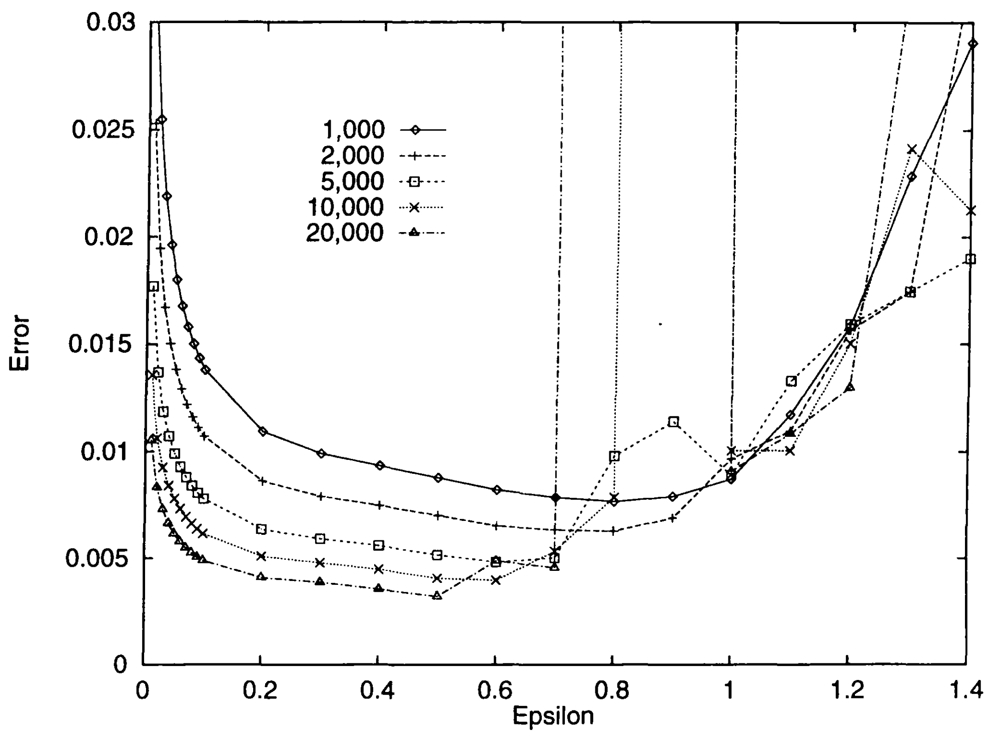
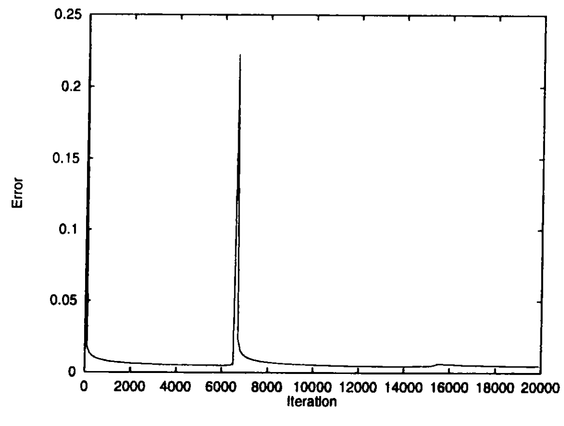


Figure 3.3: These curves show the error after a number of iterations using a range of values of learning rate. The lines leaving the top of the plot indicate peaks of very large error.

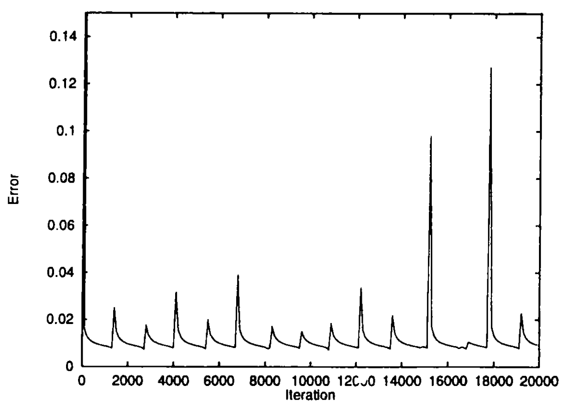
unstable somewhere between 1,000 and 2,000 iterations. By 5,000 iterations the "best" learning rate is 0.6 with higher values causing instability in the minimization. One interesting feature is that as training continues the "lump" that appeared in the curve at 2,000 iteration has exploded into a sharp spike, which by 20,000 iterations peaks at an ϵ value of about 2.0.

To see more directly what is happening when instability sets in, Figure 3.4 shows the error per iteration for $\epsilon = 0.7, 1.0, 1.1$. The error decreases smoothly until it reaches a lower bound, it then suffers a discontinuous jump and then recovers until it again reaches its lower bound which results in another jump etc. For obvious reasons these jumps occur more frequently for the larger values of ϵ.

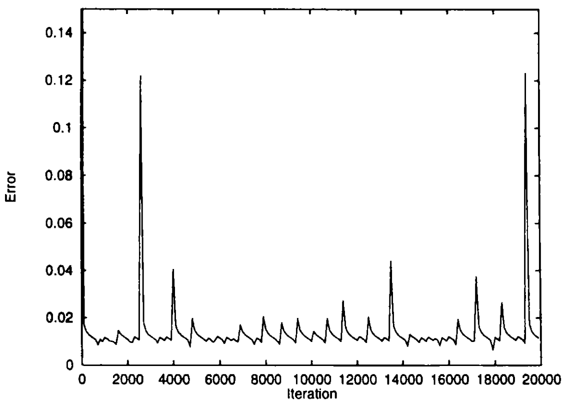
The conclusion in this simple case is that gradient descent cannot achieve an error below some lower bound if the learning rate remains fixed because training will become unstable as this lower bound is reached. But what actually happens at the instability and why does it occur? By plotting



(a)



(b)



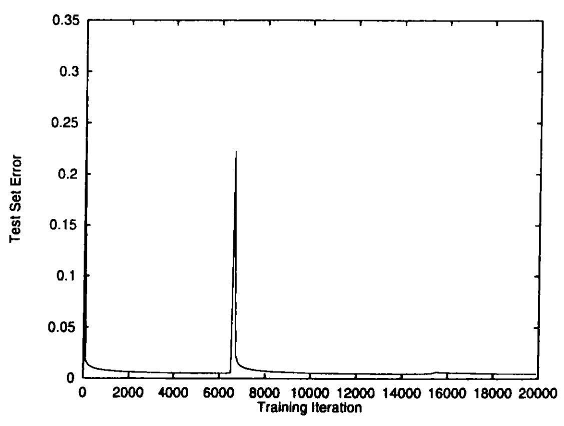
(c)

Figure 3.4: The change in error with iteration for (a) $\epsilon = 0.7$, (b) $\epsilon = 1$ and (c) $\epsilon = 1.1$.

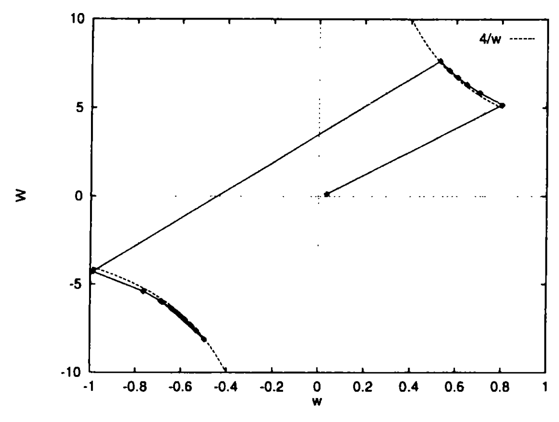
successive values of the parameters w, W and d along three orthogonal axes a graphic picture of the progress of training can be constructed. The conditions in (3.6) can now be thought of as the parametric equations of a curve in these three dimensions and the points on this curve correspond to the minimum value of $f(y)$ for a given w. I shall call this the "solution curve". In this picture the goal of training is to approach this curve and follow it as far as possible as $w \longrightarrow 0$. It is easier to see what is happening by projecting this 3D picture onto 2 2D plots, i.e. w vs W and w vs d. Fig. 3.5 shows such graphs for a total of 20,000 iterations with $\epsilon = 0.7$, plotting the parameters every 1,000 iterations. Also included in this figure is the error per iteration plot which exhibits a jump at 6,600 iterations. The corresponding jump across weight space can be seen in both the parameter plots. Interestingly, the algorithm reconverges on the negative solution for the parameters, which is analogous to gradient descent leaping over the minimum when minimizing x^2. The behaviour between 6,000 and 8,000 is displayed down to a resolution of 1 iteration in Fig 3.6. In this plot the NN is flung, seemingly at random, to some part of weight-space and then in only a couple of iterations it finds its way back to the negative part of the theoretical curve and then slowly resumes it trek towards $w = 0$. A second more gentle increase in error occurs after 15,000 iterations and this is evidenced on the parameter plots by the curve doubling back on itself. Note that this gentle rise in error *cannot* be attributed to overfitting for reasons mentioned above. It is caused by the use of a learning rate which is too large, thus making training unstable. This is an important result to remember when halting the training of more complicated networks: has the network reached its best generalization ability or would a smaller learning rate be able to reduce the error on the test set further? Figure 3.7 shows the corresponding plots for $\epsilon = 1$. Now the training is very unstable with many jumps in the error. The parameters now appear to shuttle backwards and forwards in a region near the theoretical curve, though of course the detailed behaviour will be more complicated.

Now that it is established what happens to parameters after a jump, I turn to the question of what causes a jump in the first place. The obvious explanation is that the error surface in weight space becomes very steep as $w \longrightarrow 0$. A "steep slope" in the error surface will cause a NN that has wandered onto it to suffer large weight changes. The fact that lower values of learning rate succeed in taking the training further can be taken as evidence of this. In order to examine what is happening more precisely I define the error function $E(w, W, d)$ as

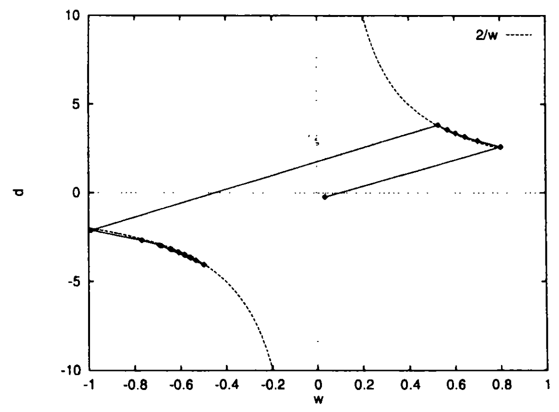
$$E(w, W, d) = \int_0^1 (f(y))^2 \, dy$$



(a)

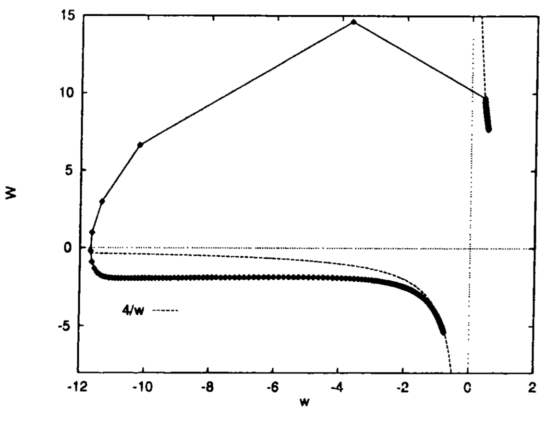


(b)

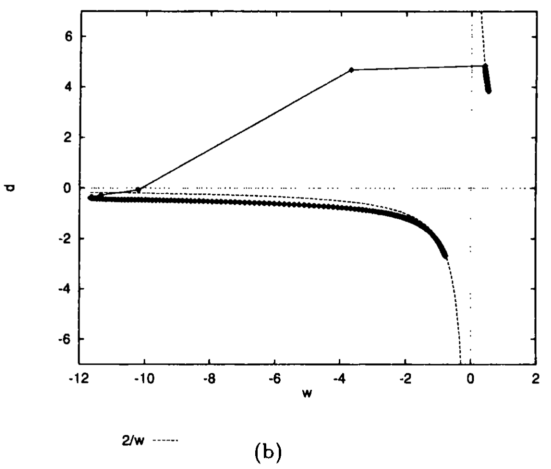


(c)

Figure 3.5: (a) Shows the change in error with iteration and (b) and (c) show how the three parameters of the 1-1-1 NN, trained with $\epsilon = 0.7$, change with iteration. The isolated point near the centre is the starting point of training with the point corresponding to the 1,000th iteration being connected to it by a line and so on for every 1,000 iterations up to 20,000.

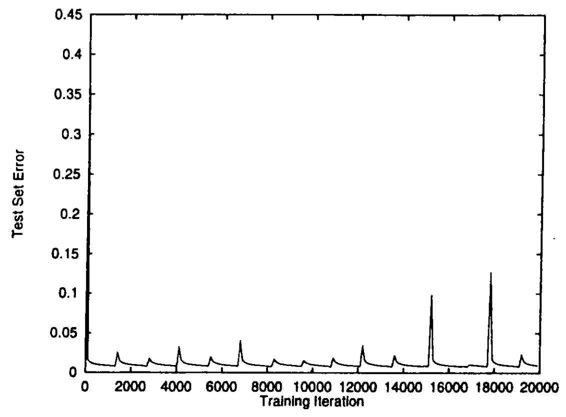


(a)

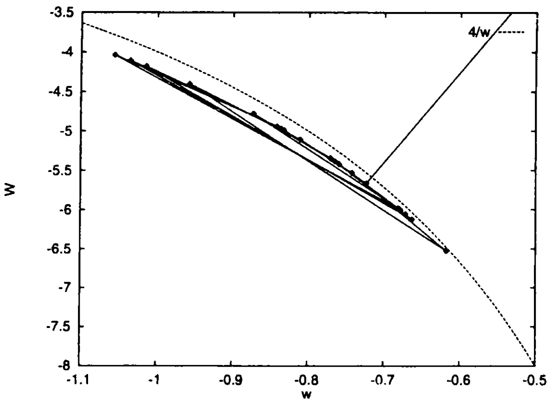


(b)

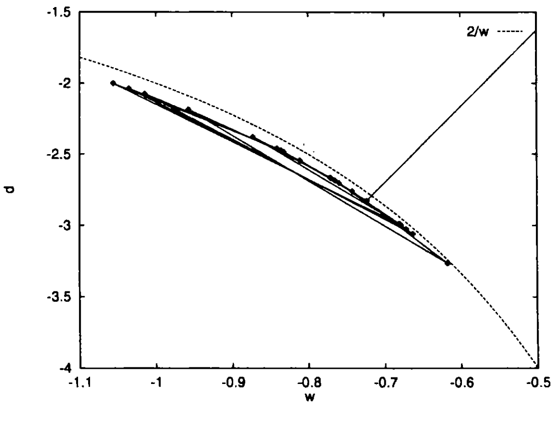
Figure 3.6: These two plots are the magnifications of the plots in Fig. 3.5 during the jump across weight space. The first point in the plot corresponds to iteration 6,000 with the parameters being plotted for every iteration until 8,000 iterations. Note that this is not the finest resolution possible because each iteration is composed of the individual adjustments for every member of the training set.



(a)



(b)



(c)

Figure 3.7: (a) Shows the change in error with iteration and (b) and (c) show how the three parameters of the 1-1-1 NN, trained with $\epsilon = 1$, change with iteration. The point (not shown) at the end of the line which leaves the plot is the starting point of training with the next point corresponding to the 1,000th iteration being connected to it by a line and so on for every 1,000 iterations up to 20,000.

$$= \int_0^1 \left(\frac{W^2}{(1+e^{-wy})^2} + (y+d)^2 - \frac{2W(y+d)}{1+e^{-wy}} \right) dy \tag{3.7}$$

Minimizing $E(w, W, d)$ is then equivalent to finding an approximate solution to $f(y) = 0$ for y in the range $[0, 1]$. Notice that the gradient descent method used above is not actually using the derivatives of Eqn. 3.7 with respect to the weight parameters, instead it uses the derivatives of $f(y_i)$, where y_i is the current "training pattern". One assumption commonly, but often tacitly, made when using gradient descent is that using derivatives based on only one training pattern at a time will still minimize $E(w, W, d)$ after the whole training set is presented - fortunately experience seems to validate this assumption. The easiest way to examine the behaviour of the derivatives of (3.7) near the solution curve to first order in w as w becomes small is as follows: For $\frac{\partial E}{\partial d}$ first differentiate (3.7) and then evaluate the integral,

$$\begin{aligned}\frac{\partial E}{\partial d} &= \int_0^1 \left(-2\frac{W}{1+e^{-wy}} + 2(y+d) \right) dy \\ &= -2\frac{W}{w} \log \frac{1+e^{w}}{2} + 1 + 2d\end{aligned}$$

expanding the log term to $O(w^2)$ (this is necessary to obtain $\frac{\partial E}{\partial d}$ to first order), gives

$$\frac{\partial E}{\partial d} = -2\frac{W}{w}\left(\frac{w}{2} + \frac{w^2}{8}\right) + 1 + 2d$$

Now to examine this derivative near the solution curve set $W = \frac{4}{w} + \eta$ and $d = \frac{2}{w} + \delta$, where η and δ are the departures of W and d from the solution curve which need *not* be small:

$$\begin{aligned}\frac{\partial E}{\partial d} &= -2\frac{\frac{4}{w}+\eta}{w}\left(\frac{w}{2} + \frac{w^2}{8}\right) + 1 + 2\left(\frac{2}{w} + \delta\right) \\ &= 2\delta - \eta - \frac{\eta w}{4}\end{aligned}$$

$\frac{\partial E}{\partial W}$ may be obtained in the same way except that it is actually more convenient to perform the series truncation before the integration. Finding $\frac{\partial E}{\partial w}$ is more complicated but maybe derived most efficiently by taking the series truncation of E, integrating it and differentiating with respect to w. To summarize the three derivatives are:

$$\begin{aligned}\frac{\partial E}{\partial w} &= \left(\frac{\eta}{2} - \delta\right)\frac{1}{w} + \left(\frac{\eta^2}{8} + \frac{\eta}{6} - \frac{\delta\eta}{4}\right) + \left(\frac{\eta^2}{24} - \frac{\eta}{8} - \frac{\delta}{8} - \frac{1}{30}\right) w + O(w^2) \\ \frac{\partial E}{\partial d} &= 2\delta - \eta - \frac{\eta w}{4} + O(w^2) \\ \frac{\partial E}{\partial W} &= \frac{\eta}{2} - \delta - (\eta - \delta)\frac{w}{4} + O(w^2)\end{aligned} \tag{3.8}$$

As w becomes increasingly small

$$\Delta w = -\epsilon\frac{\partial E}{\partial w} \sim -\epsilon\left(\frac{\eta}{2} - \delta\right)\frac{1}{w}$$

Table 3.2: Starting the 1-1-1 network at a near perfect solution and then applying back-propagation, even with a small learning rate has terrible results as seen here.

Iteration	w	W	d
0	0.001	4000	2000
1	-44.89	3999	1987
2	-3,334	3999	1968
3	-38.83	3989	1969
4	-14.82	3988	1956
5	-49.22	3982	1950

so that the change in w after a single iteration can be quite large unless either ϵ is chosen to be sufficiently small or $\delta = 0$ and $\eta = 0$. In practice it can be seen (e.g. Fig. 3.5) that as the error is decreasing, W and d never actually lie on the solution curve but merely approach it, so δ and η are not zero. As w is reduced the $\frac{1}{w}$ term will become more dominant causing Δw to become very large, so that at some point the learning rate can no longer restrain the growth of Δw giving rise to the jumps in error. This problem might be avoided if the learning rate were reduced during training to temper the growth of Δw. An alternative way might be to fix w every so often during training and train the NN so that E is minimized as far as possible with respect to W and d, forcing the NN closer to the solution curve and thus damping the effect of the $\frac{1}{w}$ term. The former method is known in Neural Network terminology as an *adaptive parameter scheme* whereas the latter method is very similar to the idea behind *conjugate gradient descent*, described in Hertz et al. (1991).

All of the above numerical results depend to some extent on the starting point of training, i.e. the initial values of w, W and d. However, as can be seen from any of the parameter plots, the parameters quickly find their way onto the solution curve and then begin their descent along it. So for this simple 1-1-1 network the behaviour during training is quite independent of the starting point, at least after the first few iterations; but what will happen if the NN is started on the solution curve at a very small value of w? If $w = 0.001$ then by (3.6) $W = 4000$ and $d = 2000$. These values result in $\sqrt{E} \sim 10^{-9}$ which is far lower than any error achieved in the previous examples. If training is now performed starting at these values then even for a learning rate as small as 10^{-4} the good solution is immediately lost. Table 3.2 reveals that w is the source of the instability because it is varying wildly while the other two parameters change comparatively slowly. This is entirely expected because of the $\frac{1}{w}$ term in the derivative $\frac{\partial E}{\partial w}$.

This simple example has demonstrated that even if a set of ideal weight values exist so that a neural network can mimic a function to arbitrary accuracy, the training algorithm used might be unable to find these weights. Here, gradient descent with a fixed learning rate could not converge on the ideal weights easily because one of the weight derivatives became very large as the ideal weights were approached. A smaller learning rate will allow closer convergence but requires many more training iterations to do so. A more sophisticated method, such as an adaptive parameter scheme, could provide some improvement but it is not at all obvious how such a scheme should operate or guarantee success/convergence.

So it appears that training using gradient descent can be quite a hazardous procedure, more so when dealing with more complicated networks where multiple local minima can plague the learning process. Ironically however, another problem in the training of neural networks, that of attaining the best generalization ability, seems to alleviate the problems caused by using gradient descent. The issue of generalisation ability was discussed in Section 1.5, and will be met again on numerous occasions.

3.2 Numerical Tests

To complete this discussion on fitting functions I shall now illustrate the training of 1-H-1 NNs on several functions and examine the effect of altering the parameters of back-propagation and the relation of the trained NNs to the theoretical results in Section 3.1.2.

The functions that shall be used for the following tests are

$$\begin{aligned}
1)\quad f(x) &= \frac{1}{2}(1+x)\\
2)\quad f(x) &= \frac{1}{2}(1+x^2)\\
3)\quad f(x) &= \frac{1}{2}(1+x^3)\\
4)\quad f(x) &= \frac{1}{45}(27+2x-56x^2+72x^3)\\
5)\quad f(x) &= \frac{1}{4}(3+\sin 2\pi x)
\end{aligned}$$

and the fit will use N_p points evenly sampled from the range $[0,1]$, i.e.

$$x_i = \frac{i}{N_p-1} \qquad i = 0,\ldots,N_p-1$$

notice that the range of these functions on this domain is $[0.5 : 1]$. These functions were chosen because they are all quite "smooth" in the given domain. Later in Section 3.3, I shall consider the effect of trying to fit more "rough" data, so that networks must effectively perform some smoothing. For now, interest is confined to: how the trained Network weights represent a given function; how the number of hidden neurons H affects the final fit and the progress of training; how the values N_p, ϵ of α affect training; and the possible (dis)advantages of using an adaptive parameter scheme. An attempt to perform some kind of grid search for the best set of parameters $(H, N_p, \epsilon, \alpha)$ would be extremely arduous. Instead I shall save some time (mine and the computer's) by making use of some hindsight drawn from personal experience in constructing the following tests.

To start with I shall examine the effect of varying the parameters of back propagation, the learning rate ϵ and the momentum α. Remember that the momentum includes a fraction α of the last weight change in making the current weight change. For this investigation: $H = 12$, which should be sufficient to fit the above five functions if the foregoing theoretical discussions are believed; and $N_p = 20$, because this is will allow for a good representation of each of the above functions. The remaining question is how to decide which parameters give the "best fit". The final RMS error would provide a convenient criterion but the meaning of "final" has to be decided in the absence of a generalisation stopping criterion. The first stopping criterion I shall adopt is that of instability, if the network error starts to increase on any iteration, after the first hundred, then training will be halted. The "after the first hundred" is necessary because the beginning of training can be quite unstable. For tiny values of learning rate it is quite conceivable that training can proceed for a very long time without any instability occurring and also without any significant progress. To avoid this I impose a maximum of 20,000 iterations on training. Remember, by definition, a training iteration is one complete presentation of the training set. So the criteria of stopping is whichever of the following occurs first

1. The error between the last two iterations has increased

2. The number of iterations has reached 20,000

The second is a practical restraint that reflects the speed of the computer. Had this been ten years ago the maximum number of iterations might have been 100, perhaps in ten years it might be 100 million. I shall construct three 2D maps of (ϵ, α), one showing the error, another showing the number of iterations, which must be at least 100 (training was never stable) and 20,000 (training is still reducing the error) with the third map showing the error change in the last stable iteration. These maps are plotted as a 16 shade grayscale with the limits of black and white as shown on each plot. Note that white indicates any value *larger* than the value shown. This provides quite complete information on the success or failure of the training. There are three plots of the results of these tests fitting function 1): Figure 3.8 shows the three maps for a coarse grid of large values of ϵ; Figure 3.9 shows a finer grid for ϵ taking on smaller values; Figure 3.10 shows a blow up of the best region from Figure 3.9. It can be concluded from these plots that the best results are achieved for $\epsilon = 0.033$ to 0.055, with $\alpha = 0.9$. In Figure 3.9 it is clearly seen that the column for $\alpha = 0.9$ seems to produce the best results, in terms of final error and stability of training. This value of α seems to be a sort "magic" number in FFNN training as many researchers find it to be the best value to use, all of the following papers independently arrive at this conclusion in the training of Neural Networks to predict time series: *Conway (1993), *Macpherson (1993) and Aso and Ogawa (1993). Referring to Section 1.4.1 it can be seen that a value of α slightly less than unity will provide considerable acceleration in "flat regions" of the weight space error surface. However, if it is too near unity it may also produce divergent behaviour during training. In this light the "magic" of $\alpha = 0.9$ is understandable. These parameter searches were only repeated for function 2) because of the long time required for execution. The results of these searches can be found in Figures 3.11, 3.12 and 3.13.

Attempts at using adaptive parameter scheme show that they usually create more problems than they solve, of the several that I have tried I shall only describe one. If after an iteration the error has increased, restore the weights to their values before the iteration was made, reduce epsilon by a factor k and repeat the iteration. Notice that in this scheme no attempt is made to increase ϵ after a series of error reducing iterations. There are three problems that arose with this scheme. First of all, there is no guidance for choosing k even after much experimentation, so the value that I chose to use, $k = 1.001$, is quite arbitrary. Also sometimes the training would get completely stuck on one iteration - that is no matter how many times the step was remade with reduced ϵs, the error would still increase. If eventually the iteration succeeded then the rest of training has to suffer from the vastly reduced ϵ. The obvious fix to this problem is raïse ϵ after many successful iterations, but I found that such a scheme often suffers from terminal indecision, where a step is re-taken many times, oscillating between high and low values of ϵ. Another problem with the above scheme is that the training time becomes uncertain, and possibly effectively infinite if ϵ is reduced too far or oscillation between two (or more) values occurs. The solution might then be to stop restoring the weights and just allow training to continue with a smaller ϵ after the positive jump in error. This can also be fatal, as it is possible, and not uncommon in practice, especially when using linear output activation functions, for weights to increase explosively. Such large weights will then cause overflow errors in the computer, effectively ending training. The construction of an adaptive parameter scheme starts to feel like the addition of endless epicycles, i.e. fixing the problems caused by previous fixes. For this reason I have avoided their use, although I will use a more manual version of an adaptive

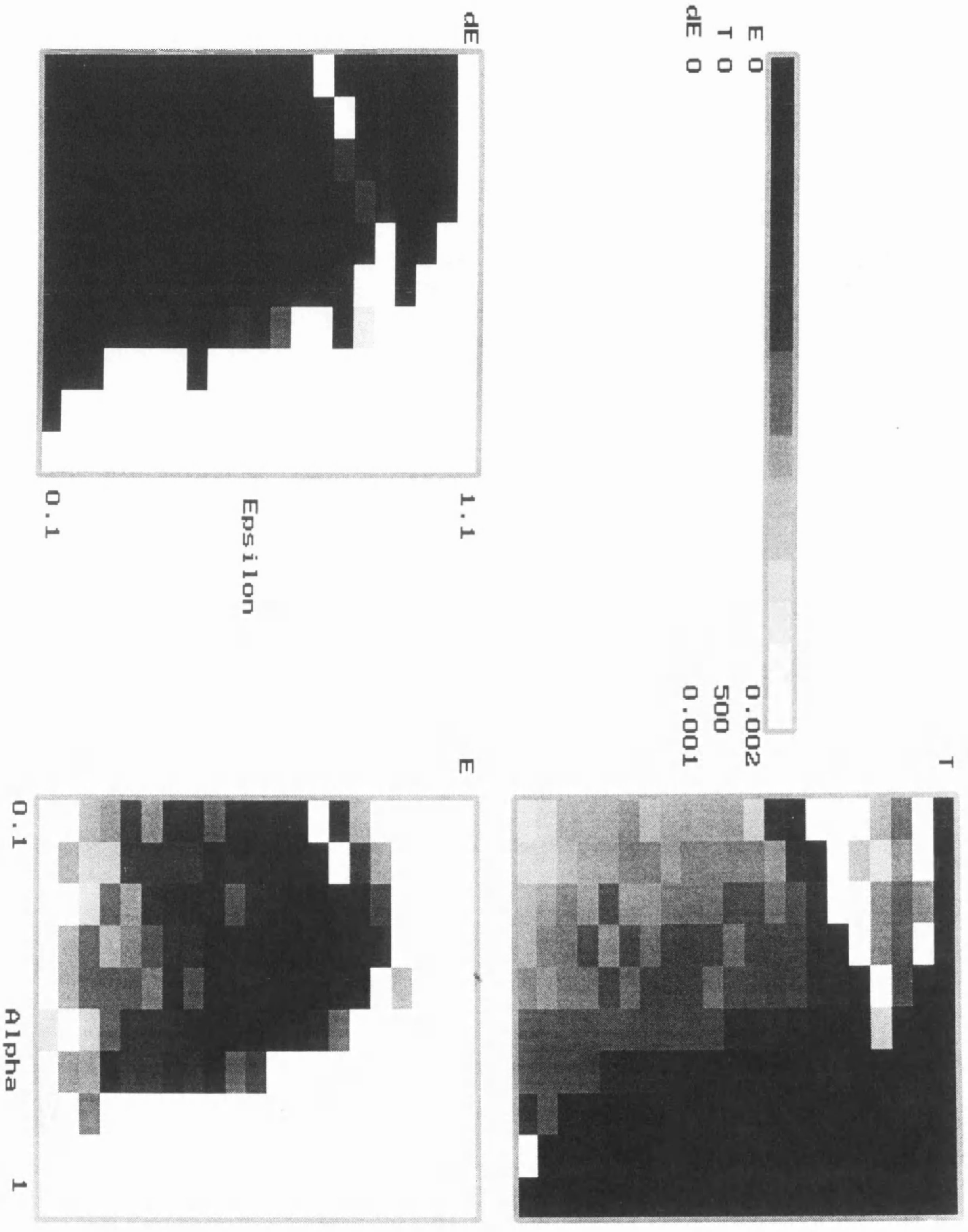


Figure 3.8: The three plots demonstrate the effect of changing ϵ and α in training a $1-12-1$ FFNN to learn the function 1) in the text. The stopping criterion for training is also detailed in the text. The gray scale plots show final error E, last stable change in error dE and the number of training iterations T. In these plots white represents that the value has exceeded that indicated in the key.

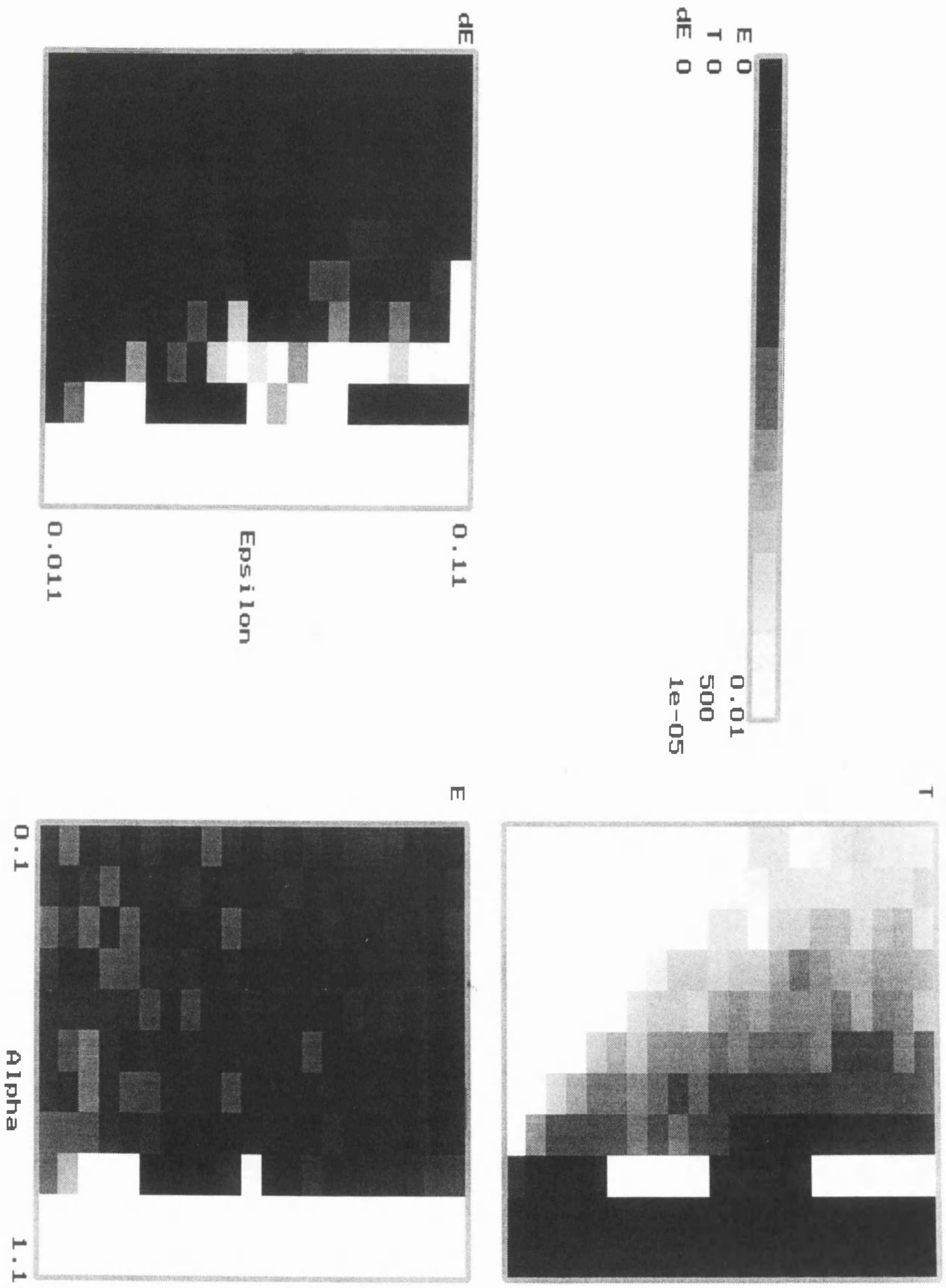


Figure 3.9: As for Figure 3.8 but with a finer grid in ϵ.

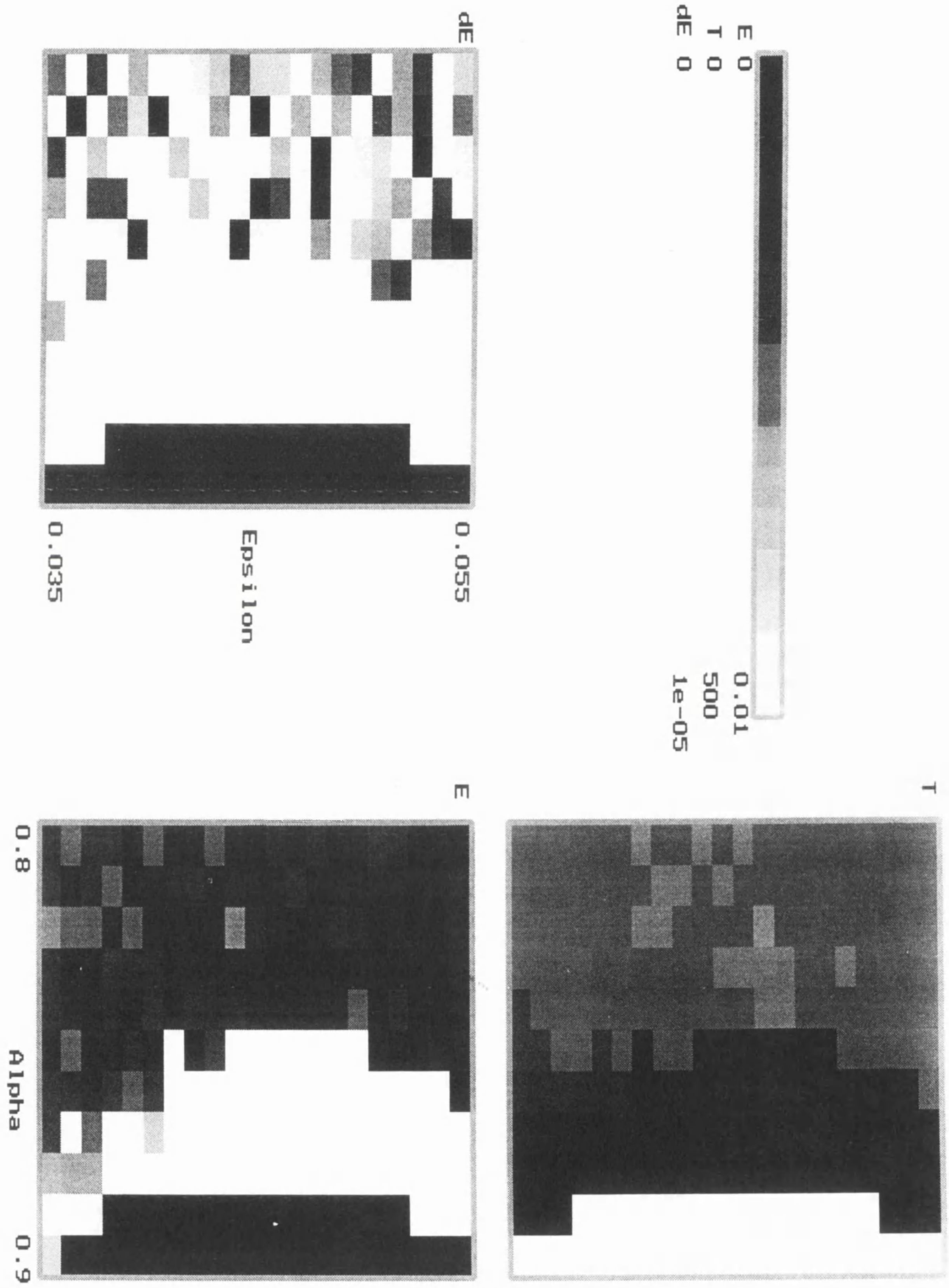


Figure 3.10: As for Figure 3.8 but with a finer grid in ϵ and α. These plots blow up the best region in Figure 3.9

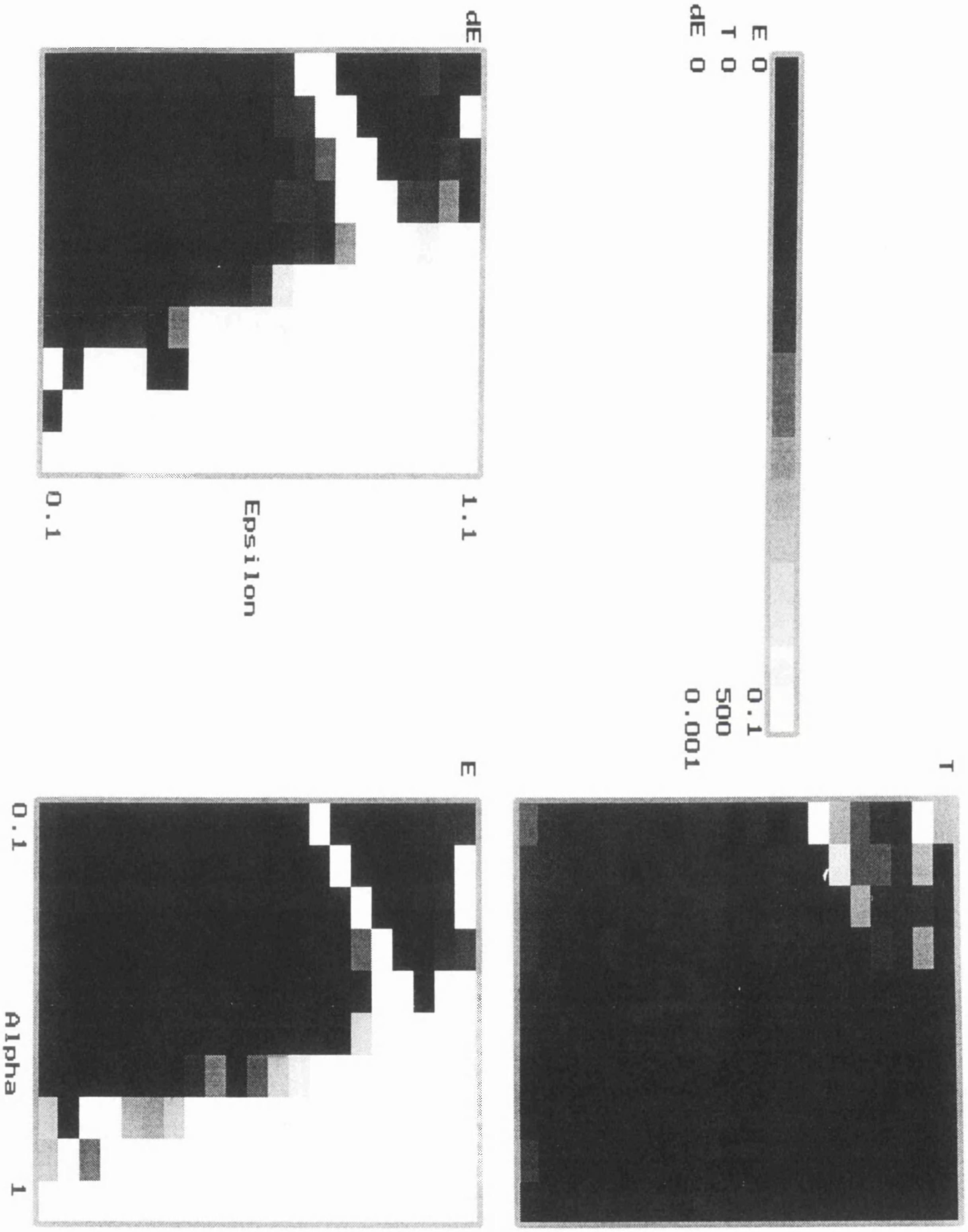


Figure 3.11: The three plots demonstrate the effect of changing ϵ and α in training a $1-12-1$ FFNN to learn the function 2) in the text. The stopping criterion for training is also detailed in the text. The gray scale plots show final error E, last stable change in error dE and the number of training iterations T. In these plots white represents that the value has exceeded that indicated in the key.

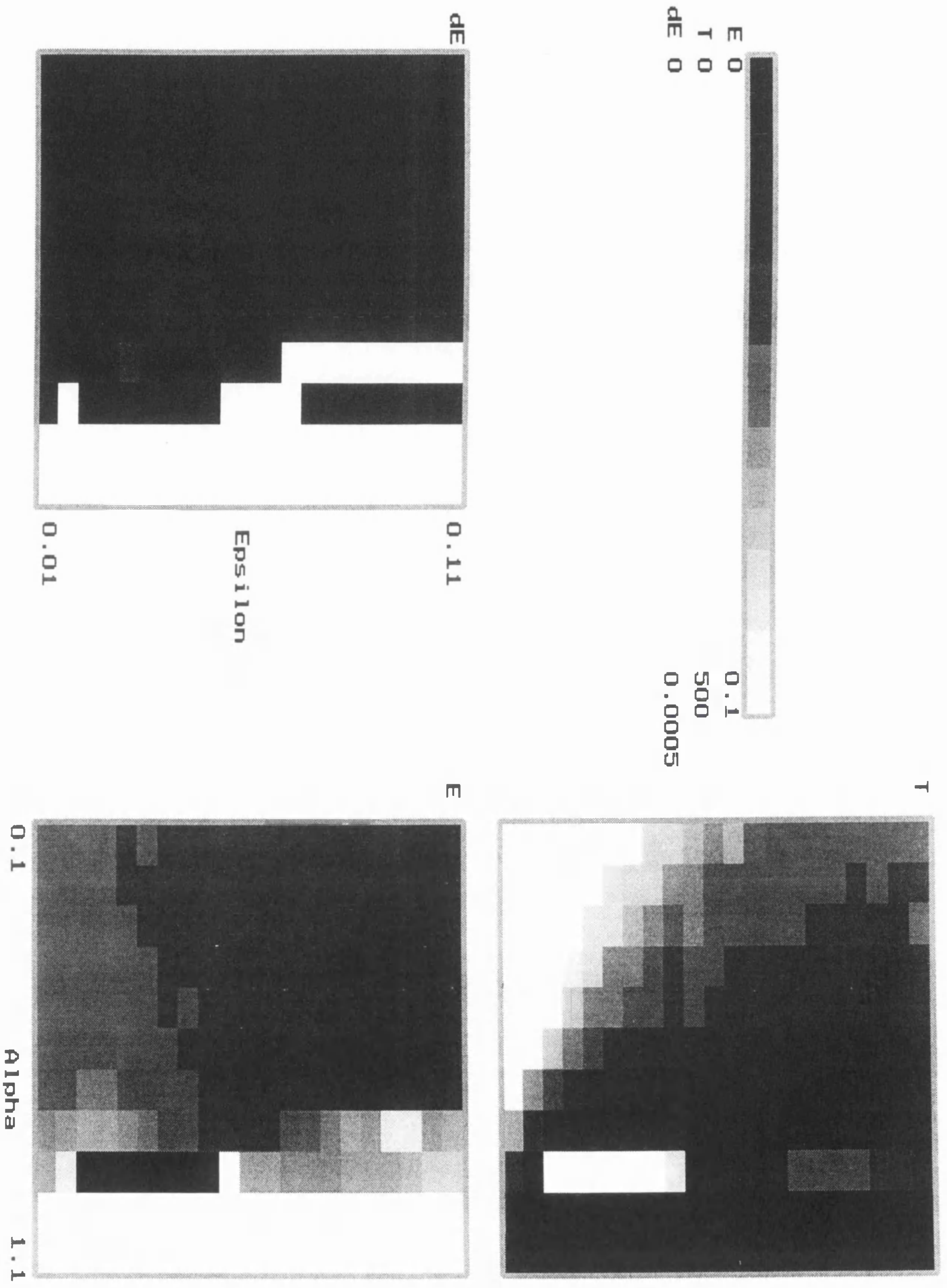


Figure 3.12: As for Figure 3.11 but with a finer grid in ϵ.

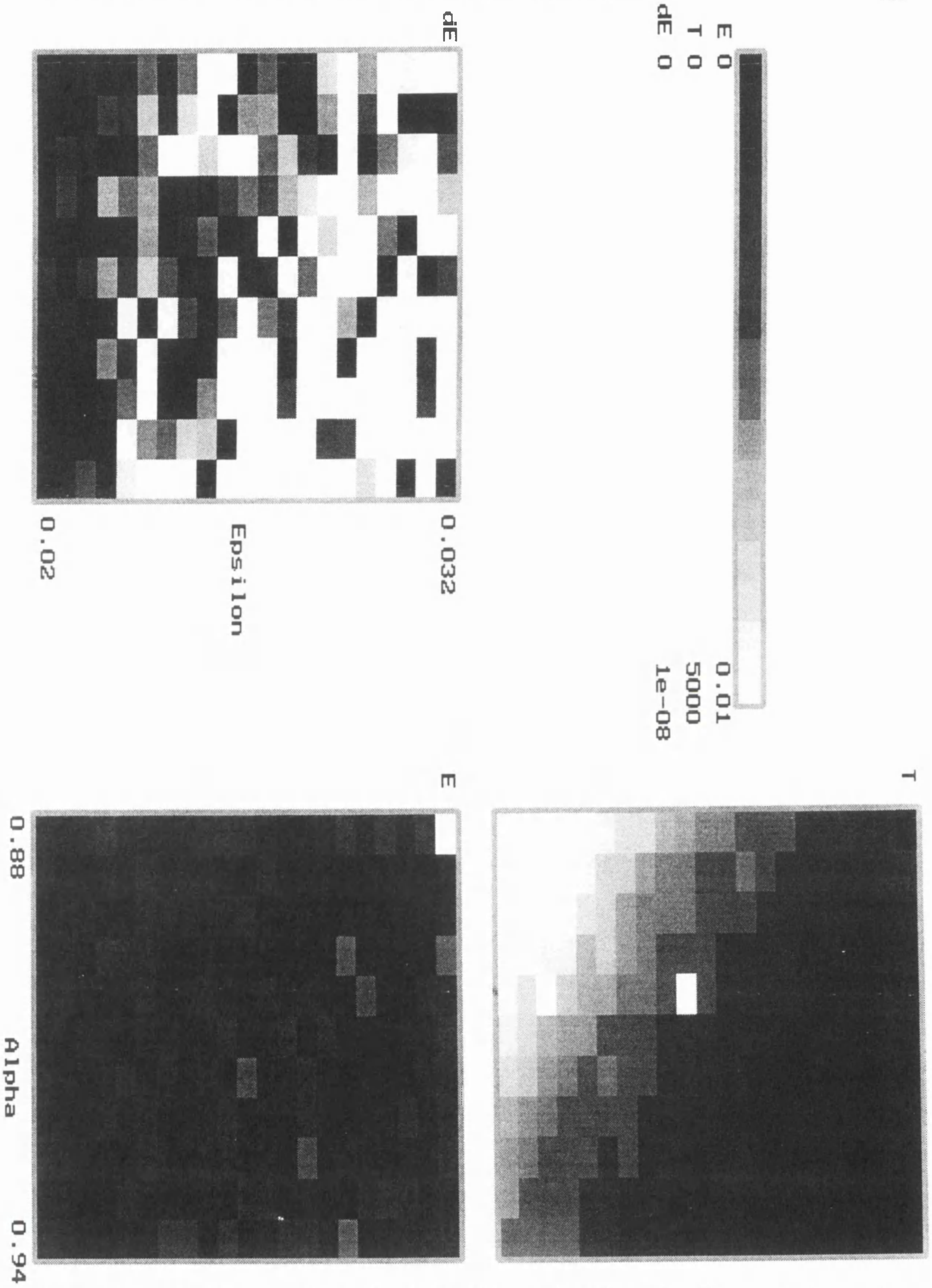


Figure 3.13: As for Figure 3.11 but with a finer grid in ϵ and α. These plots blow up the best region in Figure 3.12

parameter scheme when training some FFNNs to predict time series.

As shown above a FFNN with H hidden neurons can only hope to accurately fit an H^{th} order polynomial (that is a polynomial whose largest power of x is H), and it can only do this in some finite domain, decided by the particular activation function used. However the theoretical discussion above considered the function being fitted on a continuous interval of its domain, e.g. $[0, 1]$. In training a network the fit is really only performed at N_p discrete points in $[0, 1]$. What difference does this make? Essentially the NN is given the freedom to choose *any curve* that passes through (or near) the N_p points in the range $[0, 1]$, in much the same way as would happen in the fitting of a polynomial to a sampled function. Unlike using a polynomial to fit a function, and as shown already, the behaviour of a FFNN fit between sample points is not dramatically variable. I now test FFNNs with different number of hidden neurons on each of the six functions above. The learning parameters, chosen to be $\epsilon = 0.036$ and $\alpha = 0.9$ are just the best parameters from the above tests. Note, however, that though these were found to be best for the function 1) as learned by a 1-12-1 network, there is no guarantee that the these values are optimal for other networks learning other functions. Remembering this fact, Figure 3.15a shows the RMS errors, calculated from the $N_p = 20$ training set points in $[0, 1]$, for networks with 1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 20, 25, 30 and 40 hidden neurons. The plots for all five functions share three features. Firstly they show a flat low error region between about 5 and 10 hidden neurons, secondly they show large errors for 1 and 2 neurons and finally they show more erratic behaviour for large numbers of hidden neurons (> 10). The failure to achieve a good fit is understandable for 1 and 2 neurons, where there are just not enough trainable parameters to perform the fit, except in fitting function 1), in which case the training is actually unstable. Network s with moderate numbers of neurons, i.e. about 5 to 10, could be trained for the full 20,000 iterations without any instability occurring, hence their low errors. For networks with greater than about 10 hidden neurons the reason the error is so large is that training becomes unstable after only a few hundred iterations. Is this because large networks require smaller learning rates for stability? It seems not, as using a lower learning rate of $\epsilon = 0.01$ shows very similar results, as can be seen in Figure 3.15b. In any case the best fits achieved even by the network's with 5 to 10 hidden neurons are not that good. In many cases the network has merely learned to draw an almost straight line that approximately fits the curve over $[0, 1]$, like in Figure 3.14. In the case of function 4), the cubic polynomial, the network does not fit a line but fits a curve which is "smoother" than the one it is supposed to learn. This property of FFNNs, the fact that they err on the side of smoothness, will be of more importance later, when data with errors or noise is used.

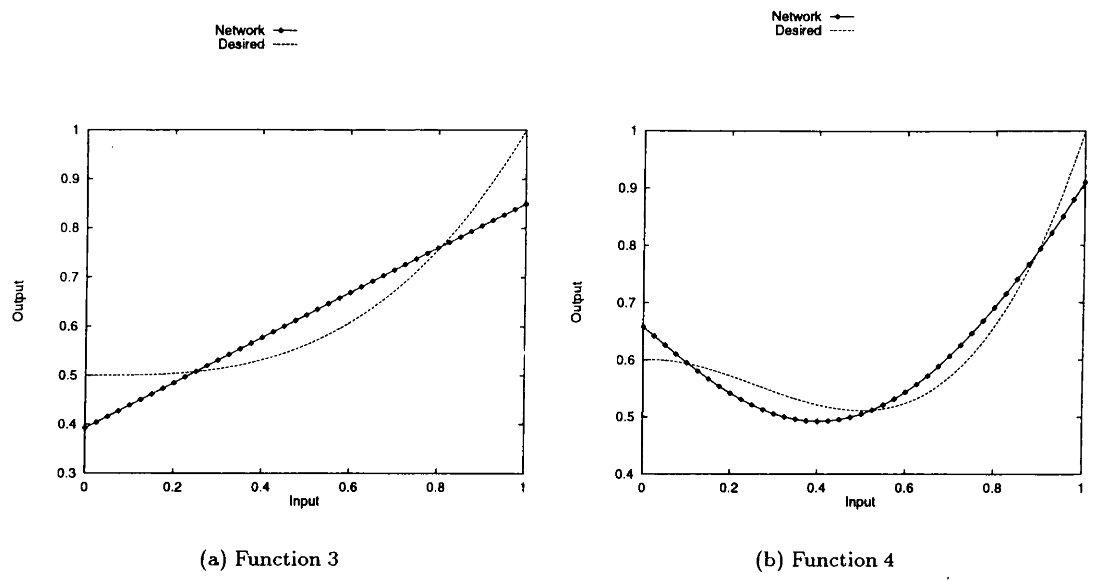
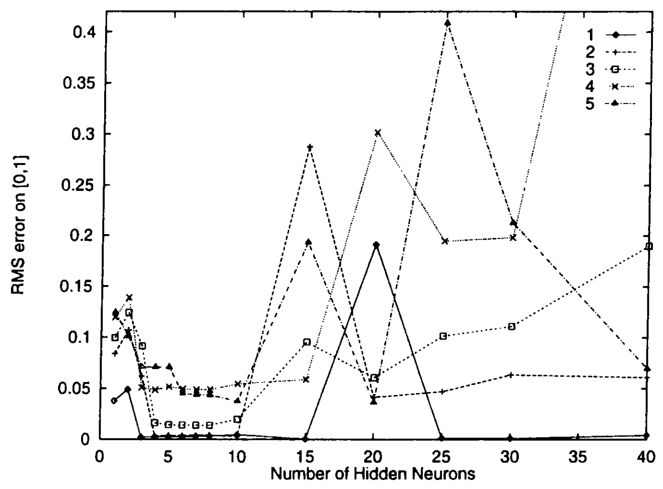


Figure 3.14: FFNN fits of functions 3) using a 1-3-1 NN and and 4) using a 1-8-1 NN. In both cases the fits are "smoother" in some sense than the desired function.

The next question is that of the number of training patterns N_p and more importantly, the method of sampling the interval and the order in which the training patterns are presented. So far the sampling has been even, that is N_p evenly spaced points from 0 to 1 (inclusive) have been used. The first pattern in an iteration is $(x = 0, F(0))$ with the following patterns presented in a sequential fashion until $(x = 1, F(1))$, after which the next iteration begins. It is important to note that the weight changes take place after each pattern is presented, not after an iteration is completed. The latter method, where the weight changes for each pattern are accumulated with their application occurring at the end of the iteration, is called *batch mode* training. It is usually found to be less effective, e.g. Hertz et al. (1991), which I have also found to be the case in my tests. Obviously there is a trade-off to be made between the time taken for training and the resolution required to accurately fit the function in question. In numerous tests I have noticed that the larger N_p, the more likely it is for instability to occur in training. There is perhaps no easy way to understand why this is so, but it does seem somehow more taxing for a $1-8-1$ network to fit $N_p = 40$ examples as opposed to $N_p = 20$ examples. Next, does oversampling encourage a neural network to learn the function underlying the sampling? The answer must really be no, as can be easily appreciated by a simple counter-example. Consider a NN fitting function 1) in the range $[0, 1]$. A very good fit will



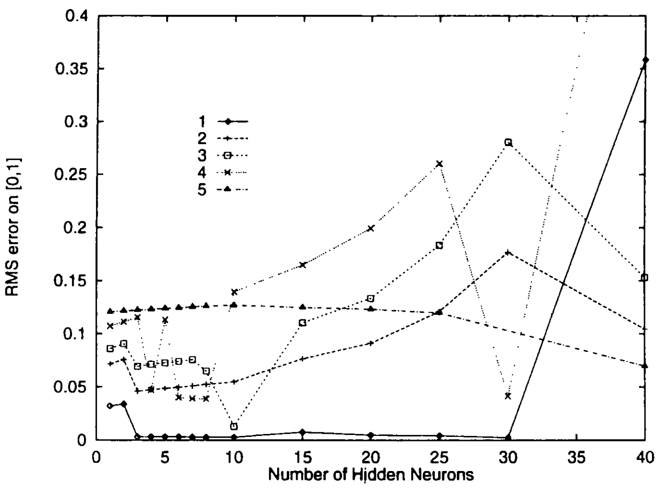


Figure 3.15: The RMS error on the training set for networks with different numbers of hidden neurons, learning the above five functions. a) uses $\epsilon = 0.036$ and b) uses $\epsilon = 0.01$, both have $\alpha = 0.9$.

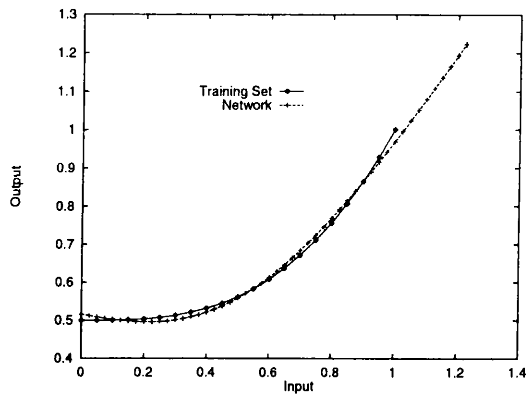


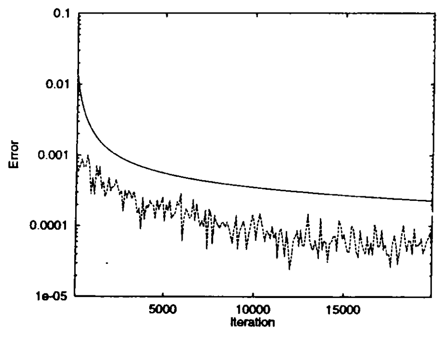
Figure 3.16: An extrapolation (and also interpolation) of the NN fit to function 3) on 20 points in the range $[0, 1]$.

achieved if the network's output, given input x, effectively fits the function

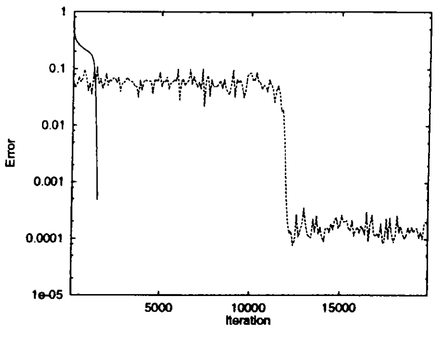
$$C(x) = \frac{1}{2} + \frac{x}{2} + \sum_{n=2}^{\infty} a_n x^n$$

where $|a_n| \ll 1$ for $n \geq 2$. As training proceeds one might hope that the a_n's are reduced so that the network approaches a good approximation but in general a NN network can only learn the function in the domain in which the function was sampled. Increasing N_p, and therefore the sampling, will not force the NN to learn that function outside $[0, 1]$. These speculations are borne out in practice and are illustrated by extrapolation of the NN fit to function 3) plotted in Figure 3.16. Notice also that this plot illustrates the stable interpolation offered by a NN in that there were $N_p = 20$ fitted points, but 50 plotted points.

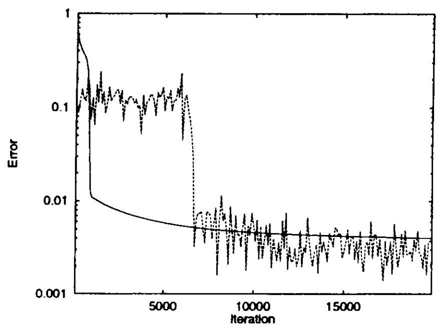
Next I investigate the effect of using non-sequential random presentation. The training set is still a set of N_p points evenly spaced on $[0, 1]$, but now the patterns are plucked at random and presented to the network. An iteration is now defined to be a presentation of N_p patterns. This means that the need to define an iteration at all is purely superficial. As can be seen from Figure 3.17 the error varies quite dramatically from iteration to iteration when using random presentation, but in the long-term it reduces the error as efficiently, if not more so, than sequential presentation does. The wild variations can actually be viewed as an advantage and, as observed in Hertz et al. (1991), can be though of as adding noise to the training procedure, which in theory at least, can provide some insurance against getting stuck in a local minimum. Notice also that the instability stopping



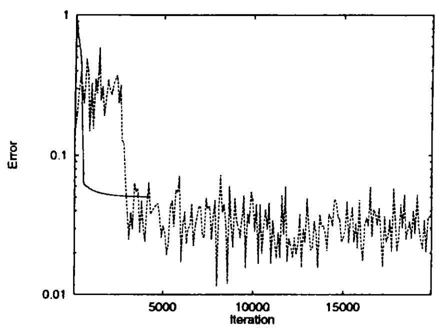
(a) 1



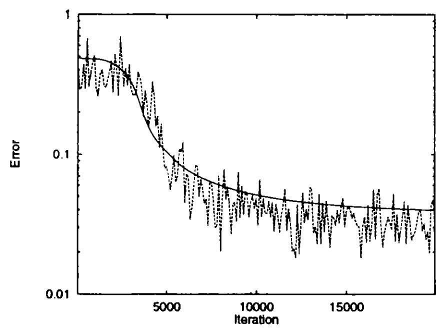
(b) 2



(c) 3



(d) 4



(e) 5

Figure 3.17: Comparison of sequential and random presentation in training a 1-8-1 FFNN on the five test functions; $\epsilon = 0.036$ and $\alpha = 0.9$. The solid line shows the error per training iteration for sequential presentation and the dashed line shows the error for random presentation. Since random presentation caused rapid fluctuation in error the instability stopping criterion was not applied.

criterion has been suspended when dealing with random presentation because the error fluctuations are so wild even over hundreds of iterations. It is also worth noting that the stopping criteria dictated above does not allow for the possibility that after instability, training could actually lead to smaller error than just prior to the instability jump (see the discussion on the $1-1-1$ network earlier). Finally, a completely random presentation scheme can be adopted, where the function is sampled randomly at each step of training. After several thousand iterations the effective sampling might be regarded as being very high. However, this is not a realistic prospect in real applications, because naturally, the input-output functional representation is not known, there is only a fixed set of examples which can form a training set.

Before moving onto the more practical concerns of fitting a finite data-set, which perhaps could contain noise or errors, I wish to point out the limitations of the preceeding work. Firstly the above work assumes that the input-output relationship to be learnt can be expressed as an analytic function in the range of interest. In fact, this is an assumption that underlies the use of FFNNs in general. More to the point it was assumed that the activation function was analytic in the region of interest and that the activation function Taylor power series coefficients were non zero - that is $g^{(n)}(0) \neq 0$ for any n. For the sigmoid $d^2g/dx^2 = 0$ at $x = 0$. In terms of the above arguments this means that FFNNs using the sigmoid activation function cannot represent function 2), or indeed any function containing a quadratic term in its power series expansion about the origin, to within arbitrary accuracy. However, at odds with this is the fact that a $1-8-1$ network can fit the function 2) quite well. The above methods showed only one possible way to arrive at a set of (linear) equations for the weights. It is equally valid to expand the sigmoid about any other point, where its second derivative, and therefore second order Taylor coefficient is non-zero. Another way to see this, in terms of expanding the activation function about the origin, is to imagine that the other coefficients in the power series arrange themselves to replicate the behaviour of x^2 in $[0,1]$. This again points out the fact that the network is under no obligation to learn the function underlying the sample patterns in the training set. In a similar vein it is possible to show that a 1-H-1 FFNN with input activation function $g(x) = \tanh \beta x$ cannot represent any function with $f(0) \neq 0$, since the zero order term in its Taylor expansion is $g(0) = 0$. It is now interesting to ask, for the functions that do not contain a quadratic term, whether back-propagation is teaching the network's the above analytic method or some other, more convoluted, method of approximating the given function. Table 3.3 compares $\sum_{i=1}^{H} Ww^n$ with a_n/g_n, where a_n is the n^{th} polynomial coefficient of the function to fitted and g_n is the n^{th} coefficient in the Taylor expansion of g(x) about the origin. This was done for

Table 3.3: Comparison of the Taylor expansion of the network's fits to functions 2) and 3) with the polynomial coefficients for those functions.

	Function 1		Function 3	
n	$\frac{a_n}{g_n}$	$\sum_{i=1}^{H} Ww^n$	$\frac{a_n}{g_n}$	$\sum_{i=1}^{H} Ww^n$
0	1	0.997	1	1.026
1	2	2.033	0	-0.408
2	?	0.561	?	18.52
3	0	0.505	-96	-45.83
4	0	0.192	0	121.686
5	0	0.136	0	-301.286

functions 1) and 3) above. It is immediately apparent that back-propagation had not taught the networks to learn what the analytic method above dictates. I have found this to be true in all investigated cases. Finally, it is also worth noting that the range of x for which the Taylor power series expansion of the sigmoid about the origin remains convergent is $(-\pi, \pi)$. (This is because at $z = i\pi$, the complex extension of the sigmoid has a singularity). In practice, if using the above analytic method to choose a network's weights, this is not a problem because the input weights can be chosen to be as small as needed.

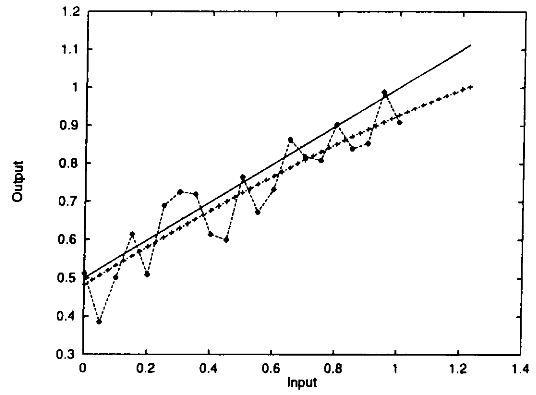
3.3 Fitting Data with Noise

In the last section the functions under scrutiny were smooth and well-behaved. Also when a network was given samples from these "nice" functions, the examples were free from error. This section attempts to bring the preceeding discussions closer to the reality of predicting time series with FFNNs by looking at data which has a stochastic component. The problem in this section is to investigate the use of an 1-H-1 FFNN in mapping sets of input values to output values. However, now the outputs are not determined only by the inputs but also by some unpredictable factor - noise. An immediate consequence of dealing with desired output values which are not entirely determined by the inputs is that two identical inputs can have different outputs. But, there is no way that any FFNN, as defined in the present context, can respond to the same input value with two different output values. As mentioned at the start of Chapter 1, there is no need to follow the data in each of its stochastic twists and turns, the goal is to predict only the predictable part of the relationship (what else?). To start with the simple case of a 1-H-1 FFNN learning a deterministic function plus noise is investigated, and in the next chapter I shall examine $I - H - 1$ networks and their success, or failure, in predicting some of the time series introduced in Chapter 1.

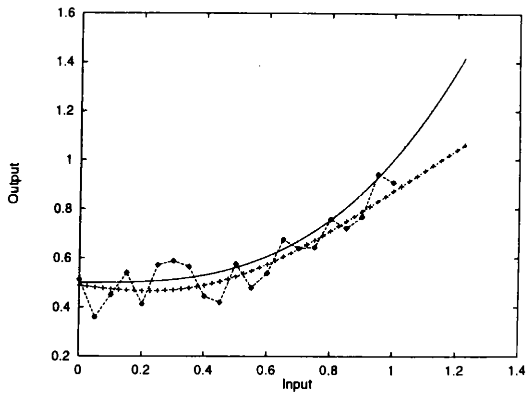
3.3.1 Noisy Function Learning

The problem is superficially similar to that of function learning above. A function $f(x)$ is chosen to be learnt over some of its domain, say $[0, 1]$ and N_p sample points are calculated in that region giving input-output pairs $(x_i, f(x_i))$, $i = 1, \dots, N_p$. However, before the training of the network begins the desired output values are doctored by adding some noise to each one, the training set pairs then become $(x_i, f(x_i) + a_i)$ where a_i is any white noise process. For the moment I assume that the x_i's are all different so that conflicting examples do not arise, and also so that hopes of reducing noise by combining patterns with the same input are dashed. The question is: how well can a FFNN learn the underlying function, and how does the variance of the noise and the number of hidden neurons affect this? Previous examples are encouraging, in that networks err on the side of a smooth fit, but equally it has been shown in theory that a FFNN with 20 hidden neurons can fit every single one of these data points (since a 20th order polynomial can). In any case, without knowledge of the underlying function, the age old problem of noisy data appears. Is it a smooth function distorted by noise or a jagged function which is free from noise? In other words, should all the data points be fitted smoothly or exactly? Of course this is impossible to answer, but FFNNs, even with an excess of hidden neurons, tend to (unknowingly of course) plump for the former interpretation. That is FFNNs, or more precisely FFNNs trained by back-propagation, always attempt to draw something smooth through the data, even if they are capable of fitting each point exactly. I demonstrate this in Figure 3.18 by showing the results of using FFNNs with different numbers of hidden neurons to fit some of the above functions with varying levels of added noise. A strange result is also demonstrated here. If more neurons are used the network seems more likely to be trained to just fit a straight line through the points in the training set. Increasing the level of noise or the number of training examples also encourages the network to make a straight line fit. It is important to stress that these results do not reflect on the network itself, they show how a FFNN, trained by back-propagation, behaves.

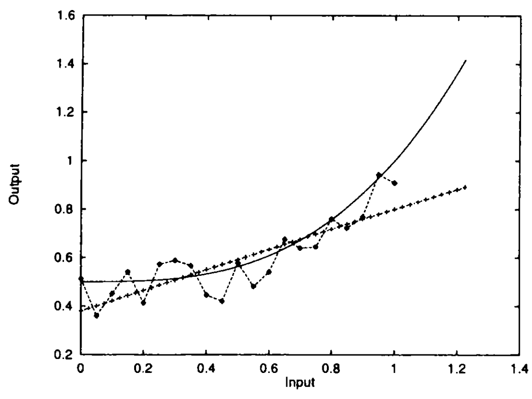
Another practical and potentially more serious concern might be noise on the inputs. That is a point x is selected to make an input-output pair that is $(x, f(x + a_i))$. With noise on the outputs the "error" on the outputs is just confined to the variance of the noise, however, even the slightest addition of noise to an input x could have drastic effects on the desired output if the derivative of f is large or undefined near x.



(a) Function 1



(b) Function 3



(c) Function 3 - 24 hidden

Figure 3.18: $1-12-1$ fits to functions 1) and 3) with added Gaussian white noise with standard deviation 0.1. The last plot shows the fit for function three using a network with 24 hidden neurons, surprisingly it fits a straight line. The solid line is the underlying function, the dotted lines with the +'s is the networks fit and the dashed lines with the diamond is the training set points.

With the addition of noise of any kind into the problem, the question of generalisation arises: is the network fitting the noise of the training set examples? The answer in the above cases must be no. The plotted points of the networks' fits in Figure 3.18 contain a test set, that is every second point is not part of the training set. It is therefore obvious that overfitting is not a problem because: the training set points are not being fitted exactly; and the test set points are no more "incorrect" than the training set points.

3.4 Concluding Remarks

In this chapter some simple FFNNs performing some simple problems have been investigated, as well as the (not simple) training algorithm, back propagation. In many cases it has been shown what the networks can, and cannot, be expected to do because of limitations of numbers of neurons or the activation function used. It has also been shown that back-propagation is not reliable as a training algorithm, sometimes completely failing to train networks to perform tasks that they should be able to perform in theory. In fact a glance at some of the ϵ-α grid searches show that white squares can lie next to black squares - that is two neighbouring sets of training parameters can lead to success and failure in training. What is more when one zooms in on one of these grid searches a similar situation is seen in many instances. The process of training these FFNNs with back propagation seems to be a chaotic one - not a desirable feature. However, when fitting a function over a part of its domain in which it is analytic and with certain provisos on the input activation function used, it has been shown in this chapter how back-propagation can be essentially replaced by the (construction and) solution of a system of linear equations.

Most of the above work was concerned only with 1-H-1 FFNNs, but many of 'he above results can be extended to apply to learning functions of I variables, i.e. using an I-H-O FFNN. In the Appendix Section A.4 the analytic training of an $I - H - O$ FFNN is explored with similar, but of course more complicated, results. In the next Chapter, instead of attempting to investigate the general problem of fitting functions if I variables numerically, I turn directly to face the prediction of time series with I-H-O networks. Armed with some detailed knowledge of the simpler cases studied in this chapter, knowledge comprised of both experimental experience and theoretical insight, the more practically minded studies of the next two chapters is given an intuitive basis from which to proceed.

Chapter 4

Prediction of Artificial Time Series

"Treason, treachery, infamy, infamy!!! Oooh, they've all got it in for me." *Julius Caesar*

Not surprisingly Neural Networks do not predict the future without error. Some configurations of Neural Networks are capable of making better predictions than others, and the best configurations for predicting one time series are not necessarily best for predicting other time series. In this Chapter I investigate how different networks perform on a variety of artificial (i.e. non-natural) time series.

The basic aim of this Chapter is to shed some light on how FFNNs make predictions of time series by using examples where theory and practice can be compared. By "theory", I mean both the theory of time series, as described in Chapter 1, and the theory behind neural networks. Regarding the latter, I shall use the method of analytic training, introduced in the preceeding Chapter, to show how many hidden neurons are needed to predict some simple time series. The most important issue to emerge out of the following results is that of the delay effect, the implications of this effect will be discussed in depth at the end of this Chapter.

4.1 Criteria of Success

Before exploring the infinite parameter space of possible FFNNs (different numbers of inputs, hidden neurons and outputs), training algorithms and parameters, it is prudent to discuss what is actually meant by "better predictions". That is, what should be the criteria of successful prediction? Two obvious criteria are the absolute and squared difference between the predicted value and the actual value. The question is: is one of these sufficient to assess the success of prediction by itself? The answer is *no*, because of a problem such as "delay", which will be discussed as and when the issue arises throughout this chapter.

Given the original time series, $\{t_i\}$, and the predictions made of a sequence of N elements within this time series, $\{p_i\}$, an *RMS error* can be simply defined as

$$E_{\mathrm{RMS}} = \sqrt{\frac{\sum_{i=1}^{N}(t_i - p_i)^2}{N}} \tag{4.1}$$

or as a percentage error

$$E_{\%\mathrm{RMS}} = 100\sqrt{\frac{\sum_{i=1}^{N}\left(\frac{(t_i-p_i)}{t_i}\right)^2}{N}} \tag{4.2}$$

Another commonly used error is the χ^2 *statistic*:

$$\chi^2 = \sum_{i=1}^{N}\frac{(t_i - p_i)^2}{p_i} \tag{4.3}$$

In the case of predicting Sunspot number, the RMS error is usually greatest near the maxima in SSN, while the % RMS error tends to be greatest at regions of low time series values, e.g. SSN minima. The χ^2 *statistic* must be treated with some caution because the prediction error is not necessarily distributed according to the normal distribution. Nonetheless, even though it may lack its usual statistical meaning in this context, it has been a commonly used measure of success in the past, for example Holland and Vaughan (1984) and *Macpherson (1993). Even from this brief summary it is quite clear that each error measure exaggerates some features of the prediction accuracy and obscures others. This suggests that one error measure is certainly insufficient in describing the quality of a set of predictions. As discussed in Chapter 1, the residuals of a good prediction scheme should form a white noise time series, that is there should be no structure left to predict. However, establishing the "whiteness" of noise, whether it is done by looking directly at the mean and variance or by analysing the spectrum is not a trivial problem and I do not perform such tests in this chapter. In addition to the above three error measures, another tool that will prove itself invaluable later is the time shift measure, T_τ, defined:

$$T_\tau^2 = \frac{1}{N-\tau}\sum_{i=1}^{N-\tau}(t_{i+\tau} - p_i)^2 \tag{4.4}$$

The time shift function at lag τ can be thought of as the re-calculated RMS error after the prediction curve have been slid along the time axis by τ time-steps. The time shift function can also be written as

$$T_\tau^2 = \bar{t^2} + \bar{p^2} - 2\frac{1}{N-\tau}\sum_{i=1}^{N-\tau} t_{i+\tau}p_i$$

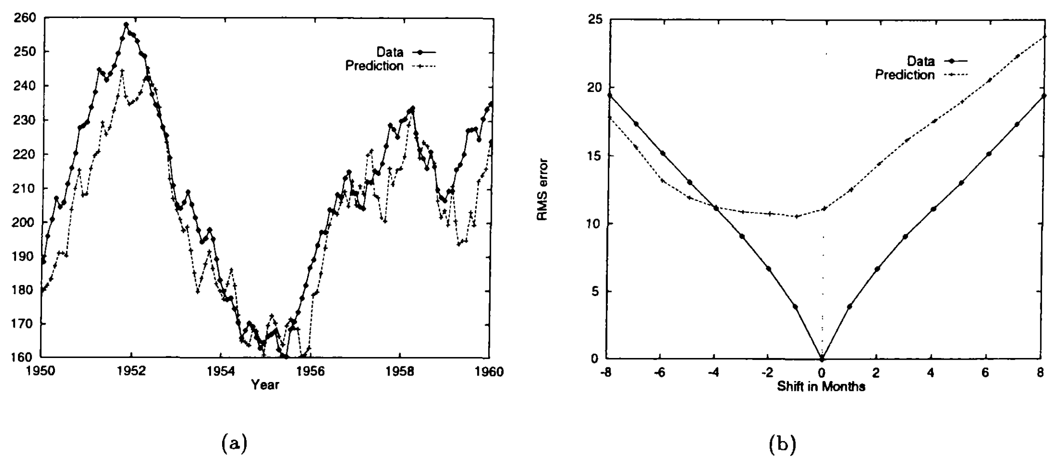


Figure 4.1: A plot of a) 6 month ahead predictions of geomagnetic K_p index made with a 12-12-1 FFNN and b) the time shift plot for these predictions.

where the last term is -2 times the sample covariance at lag τ between the predictions and the time series.

For assessing the accuracy of the results in this thesis, I have chosen to use the RMS error and the time shift plot. I use the RMS error because it is perhaps the most commonly used error by others (see the end of Chapter 1), and because, to my mind at least, it is the one that provides the most immediate meaning. The reasons why the time shift function is useful is the subject of the following section.

4.1.1 Delay and echo

In using FFNNs to predict time series, it has been my experience that many prediction plots of time series, such as those in Figure 4.1a, seemed to resemble the original time series shifted a couple of steps forward in time. This apparent "delay" can be tentatively interpreted as the network echoing its most recent input as an output. To verify if this is the case the time shift function, as defined in (4.4), was used to produce graphs such as the one in Figure 4.1b. Each point on this graph is obtained by sliding the predictions τ months along the time axis and recalculating the error. In this case the predictions are made 6 months ahead, but the time shift plot indicates that they best match the original time series if they are slid 1 months backwards. In this sense they are really predictions of 5 months ahead. However, predictions are still being made so this *Delay Effect*, as I shall call it from here on, is not just a simple echoing of an output. As shall be seen in this chapter, predictions invariably suffer from delay, sometimes it is the straight echoing of an input at an output, but most often it is seen as a minimum in the time shift curve that is not at $\tau = 0$.

4.2 Choices

The two main choices that have to made in using a NN to predict a time series are deciding its *architecture* and selecting a *training scheme.*

4.2.1 Architecture

In general, the architecture of a neural network refers both to the number of units in the network, and to how they are connected. In this chapter the only class of networks under consideration is that of the two layer feed forward neural network, so that the *architecture* of the network is completely specified by the number of neurons in each layer. Remember that a FFNN has an input layer, a hidden layer and an output layer, where the connections relay the input data to the hidden layer, and relay the hidden neuron data to the outputs. This produces a functional transformation of the inputs at the outputs, where the outputs are required to be predictions of the future. The free parameters in designing a FFNN architecture are then:

I - The number of inputs

H - The number of hidden neurons

O - The number of outputs

How should the number of inputs be chosen? Clearly, only one input wil' not be enough to make a meaningful prediction, except for some special cases where the time series is deterministically monotonic. If the time series is deterministically periodic, and has only one maximum (or minimum) per period, for example a sinusoid, then two inputs will be sufficient. In such cases the extravagance of using many inputs covering the time series' ancient history will then slow down training needlessly. It is interesting to note that although the NN does not need these earlier values to make an accurate prediction it might still make use of them (see later 4.3). Likewise, one must also be careful not to present the network with too much detail at the inputs, e.g. predicting at one year ahead with 365 daily inputs. In any case, using more units than is needed will slow training. It is therefore apparent that the minimum number of inputs needed to predict a given time series depends on both the nature of the time series in question and the format in which it is presented. These choices are really a matter of common sense. For example, 13 month smoothed sunspot number (as defined in section 1.7) shows features down to a time scale of a few months and exhibits the trend of the solar cycle on time scales of 6 months and above. This suggests that 12 monthly inputs spanning the last year of SSN will be a reasonable choice if the NN is to predict on a scale of 6 months and beyond.

How should the number of hidden neurons be chosen? As their name suggests the values of the hidden neurons have no obvious meaning and therefore the choice of the number of hidden neurons is arguably the most difficult. The last chapter showed how to determine how many neurons were required to fit analytic functions using a 1-H-1 network. Here there are two apparent complications: firstly, the precise future-past relationship of the time series data is not likely to be expressible as an analytic function, if only because of a stochastic component; secondly, as discussed above, a FFNN which is to be of use in the prediction of time series must have $I > 1$. Referring to the first problem, and harking back to Section 1.1, a general form[1] for a time series is

$$X_t = P_0(X_{t-I}, \ldots, X_{t-1}) + a_t \tag{4.5}$$

where a_t is an (unpredictable) white noise time series. To predict this time series using a FFNN with one output predicting at one-step-ahead, the I-H-1 network would need to represent the function $P_0(y_1, \ldots, y_I)$, where the y_is are the network inputs. That is, in this case, the FFNN has just to fit a function of I variables. If the function is analytic then the analytic training method of the last chapter becomes relevant, as I shall demonstrate later in Sections 4.3 and 4.4. The range of the inputs over which $P_0(y_1, \ldots, y_I)$ must be fitted can be envisaged by imagining the y_is to form a vector, this vector defining a point in an I-dimensional state space. It is then obvious that the predictor function need only be fitted in regions of the state space that the time series is likely to visit, as illustrated in Figure 4.2 - in the terminology of chaotic mechanics this region is the attractor. This point is returned to in more detail in Section 4.3 where the simple case of predicting a sinusoid is considered. So, with a known analytic predictor function, it is possible to say with some certainty how many hidden neurons are needed to obtain a perfect fit[2] of the function. Of course, the time series form given in (4.5) is not the most general, and, as pointed in Connor et al. (1994), recurrent networks may be required to handle moving average type terms. The second problem, that of extending the analytic training methods to cope with more than 1 input is dealt with in Appendix Section A.4.

[1] This is not completely general because no moving average type terms are included and no explicit time dependence is allowed - see Section 1.1

[2] Recalling the method of analytic training from before, the fit is actually asymptotically perfect

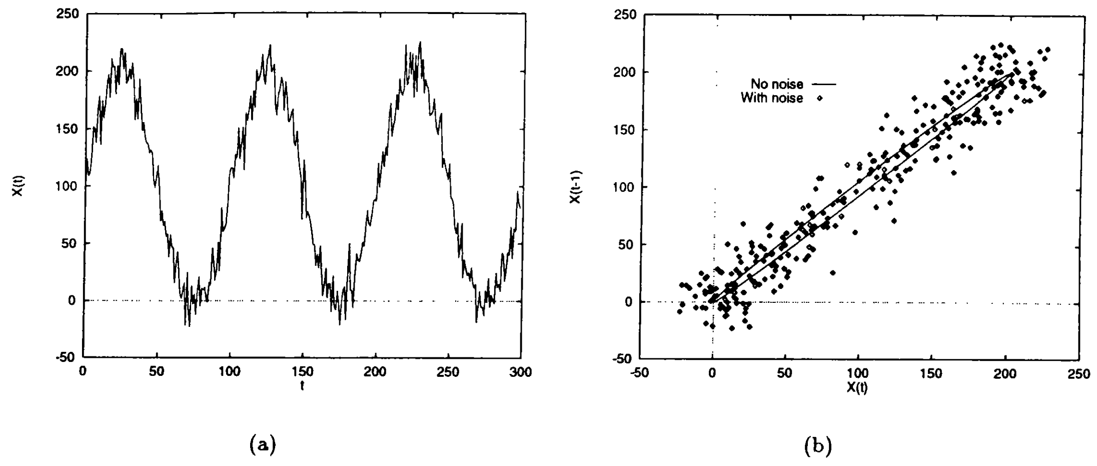


Figure 4.2: Figure a) shows a sinusoid with added Gaussian noise. Figure b) a state space plot showing, the path of both the sinusoid with added noise (the points) and a sinusoid without added noise (the curve).

How should the number of output neurons be chosen? The simplest case is to have only one output predicting at p steps ahead. For time's sake this is the only case that I shall examine. An alternative is to have O outputs predicting at $p_1, p_2, \ldots, p_O$ steps ahead. *Macpherson (1993) found the surprising result that this multiple output method, with $p_i = i$, gave slightly better performance than single output networks. Finally, as shown in Appendix Section A.4, the analytic training method for an I-H-1 network can be generalised (relatively easily) to an I-H-O network, though there will be a factor of O more sets of linear equations to solve for the output weights.

4.2.2 Training Schemes

For the work in this chapter standard back-propagation will be employed as the training algorithm. Tests will be carried out to find out what values of training parameters are best and which architectures are best. In addition I shall demonstrate the method of analytic training in finding the weights for some of the time series predicting FFNNs. Since these numerical tests are very time consuming they will not be performed exhaustively for every conceivable combination of parameters. Instead I shall use some hindsight and experience, as well as some educated guess-work, to decide which particular cases are worthy of investigation. Note that, when using back-propagation, training is stopped when one of the following occurs

1. If after the first 100 iterations the error has increased for ten iterations.

2. The number of iterations has reached 20,000.

Justification for these stopping criteria is given in Section 3.2. Unless otherwise stated the learning parameters are taken to be $\epsilon = 0.01$ and $\alpha = 0.9$. If the completed number of iterations reads as 111, then this almost certainly means that training was never stable.

4.2.3 Test Time Series

The following time series will be used throughout this chapter to provide a basis for the numerical tests. I have chosen to split them into five groups, each group represented by an acronym:

EXn	Time explicit. That is $X_t = f(t) + a_t$.
NARn	Auto-regressive type time series.
NMAn	Moving average type time series.
NAMn	Mixed Auto-regressive Moving average type time series.

Each group is given its own section, with the test time series for that group given in a numbered list at the beginning of that section. The 'N' in NMA and the other acronyms is used to distinguish these time series, which are possibly non-linear in form, from the classical and linear Box-Jenkins models. Also the use of a_t is reserved to mean a Gaussian white noise process and likewise μ and σ^2 are reserved to denote the mean and variance of a_t. It is also worth noting that, in the software that I have written, the time series data are always scaled to lie between 0.2 and 0.8 - reasons for this and a complete description of the Neural Network software can be found in Appendix B. Unless otherwise stated I use the first 105 time series elements, i.e. $t = 0, \ldots, 104$ as a training set and the next 45, i.e. $t = 105, \ldots, 149$, as a test set. The Gaussian white noise series was generated using the NAG routine g05ddf and was also used to supply the "start-up" values for NAR, NMA and NAM time series examples. Each time series used the same realization of these white noise random numbers, that is to say the random number seed for the generator was the same for each generated time series.

Finally, a note of warning about the results concerning the RMS prediction errors. The RMS errors give an indication as to the size of the errors for each network on each time series' test set. The standard deviation of these RMS errors was nearly always as large as the error itself. In practice this means that if, for example, the set has five examples removed and the error is re-calculated, then the RMS error can differ by several percent from its original value. So in many cases it is simply not meaningful to talk about one network as being more successful than another network if the difference of the errors is quite small. However, at the same time, removal of a few training patterns can affect all the networks' errors in much the same way - e.g. the removal of 5 "hard to predict" patterns will reduce all the errors. For the NAR5 time series I have quoted two sets of errors, the second set calculated on a slightly different test set. Note that the ranking of networks in terms of test set error is preserved and general conclusions like "Networks with 8 inputs perform worst" can be drawn from either set of errors. So, when a statement like "the best NN achieves a prediction error that is signficantly better an echo" is made, use of the word "significantly" has been justified by checking the prediction errors in the manner just described.

4.3 Time Explicit Time Series

These are time series which can be written as

$$X_t = f(t) + a_t$$

The only two time series I shall investigate in this section are

$$\begin{aligned} \text{EX1)} \quad X_t &= t - 1850 + a_t \\ \text{EX2)} \quad X_t &= 100 + 100 \sin \frac{2\pi}{100} t + a_t \end{aligned}$$

The results of using different networks to predict different realizations of these time series are documented in the following pages. Each page has a table showing the final test set RMS error and number of iterations for each network, a commentary on the results, a plot of the best network's predictions and the associated time shift plot. The convention from here onwards is that the data to be predicted is shown as a solid line and the predictions are shown as a broken line.

EX1: $X_t = t - 1850 + a_t$ This time series can be re-written as an ARMA(1,1) time series:

$$\begin{aligned} X_t &= (X_{t-1} - a_{t-1}) + 1 + a_t \\ &= X_{t-1} + a_t - a_{t-1} + 1 \end{aligned}$$

It is easy to see that the best prediction one step ahead, using only the last value, is just $\hat{X}_{t+1} = X_t + 1$. The expected RMS error on such a prediction is $\sqrt{2}\sigma$. Given the last I values of a time series of the form $X_t = b_0 + b_1 t + a_t$, it is possible to reduce the expected RMS error further by adopting the following prediction scheme:

$$\hat{X}_t = \frac{1}{I} \sum_{i=1}^{I} X_{t-i} + b_1 \frac{I}{2}$$

The residuals, $Y_t = \hat{X}_t - X_t$, then possess the properties

$$\begin{aligned} E[Y_t] &= 0 \\ Var[Y_t] &= \sigma^2\left(1+\frac{1}{I}\right) \end{aligned}$$

that is the predictions are unbiased and have an expected RMS error of $\sigma\sqrt{(1+\frac{1}{I})}$. In fact this prediction method is the best that can be hoped for, in the least squares sense, given access to only I inputs.

In testing network performance on the EX1 time series, four different realizations of the time series were generated, each with a different noise level. They are labelled as follows:

n No Noise

a $\sigma = 0.1$

b $\sigma = 1$

c $\sigma = 5$

At this point it is interesting to ask how the architecture of the FFNN determines its ability to predicting the EX1 time series. From above, the predictor function for this time series is

$$P_0(x_1, x_2, \ldots, x_I) = \frac{1}{I}\sum_{i=1}^{I} X_{t-i} + b_1\frac{I}{2} \tag{4.6}$$

The output of an I-H-1 network can be written to first order in the inputs by using a multi-dimensional Taylor expansion about $x_i = 0 \quad i = 1, \ldots, I$ as

$$\begin{aligned} C(x) &= \sum_{j=1}^{H} W_j g\left(\sum_{i=1}^{I} w_{ji}x_i\right) \qquad (4.7) \\ &= \sum_{j=1}^{H} W_j\left(g_0 + \sum_{i=1}^{I} g_1 w_{ji}x_i\right) \\ &= g_0\sum_{j=1}^{H} W_j + g_1\sum_{i=1}^{I}\left(\sum_{j=1}^{H} W_j w_{ji}\right)x_i \end{aligned}$$

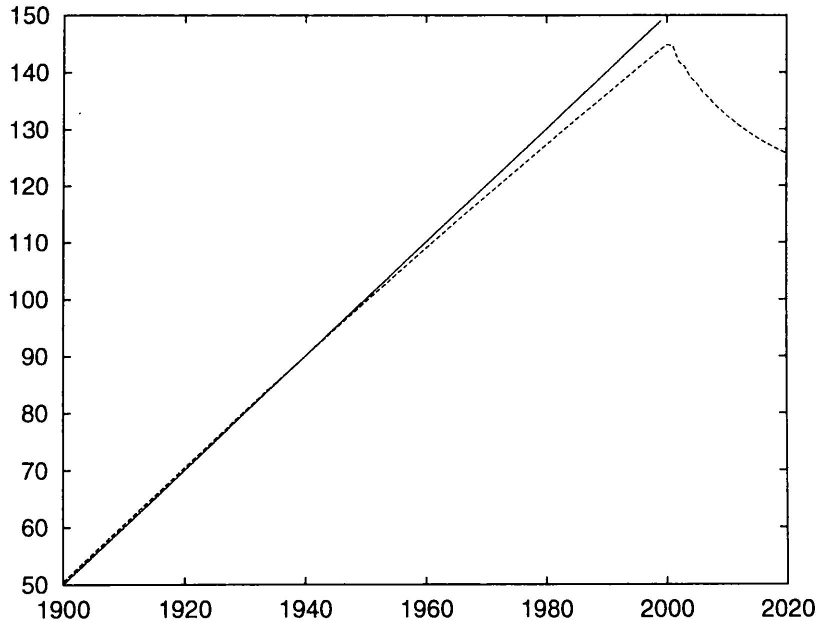
where $g_0 = g(0)$ and $g_1 = g'(0)$. Comparing (4.6) and (4.8) shows that the network weights must satisfy the following two equations

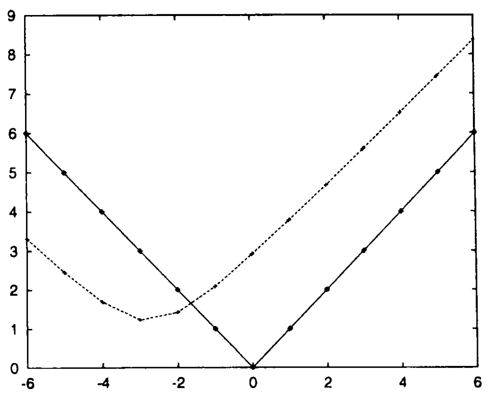
$$\begin{aligned} \sum_{j=1}^{H} W_j &= \frac{b_1 I}{2g_0} = A_0 \\ \sum_{j=1}^{H} W_j w_{ji} &= \frac{1}{Ig_1} = A_1 \end{aligned}$$

EX1n: As might be expected the network cannot learn to generalise well to the test because the test set values are outside the range of the training set values. Notice, though, that the training set values are well fitted. The time shift plot shows something interesting which serves as a warning for later use, in that a minimum of 3 months is caused not by any delay or echoing effect, but by the consistent under-estimation of the predictions on the test set.

Table 4.1: The results of predicting the EX1n time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	4.534	5.487	5.374
4	4.897	4.515	4.316
8	2.915	3.817	3.792
	Number of Iterations		
2	20000	20000	20000
4	20000	20000	20000
8	20000	20000	20000

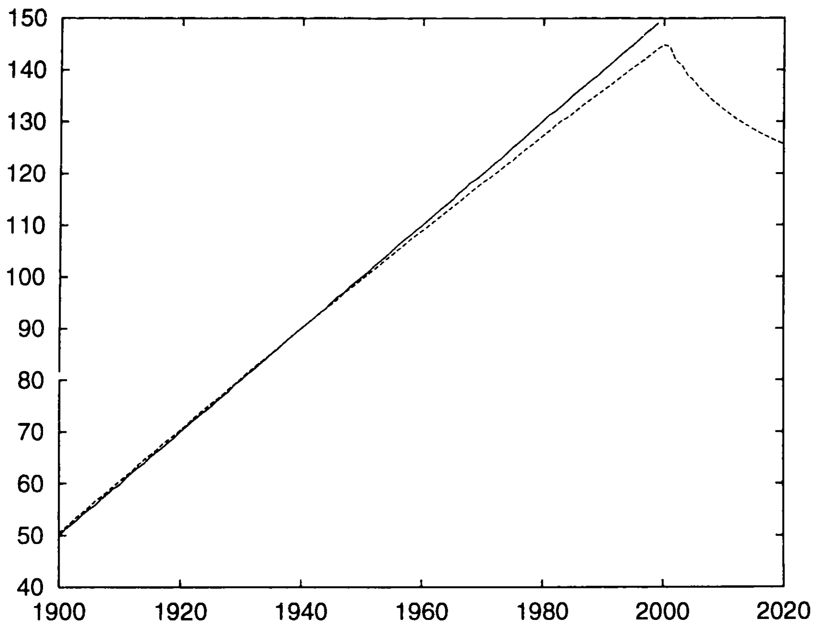


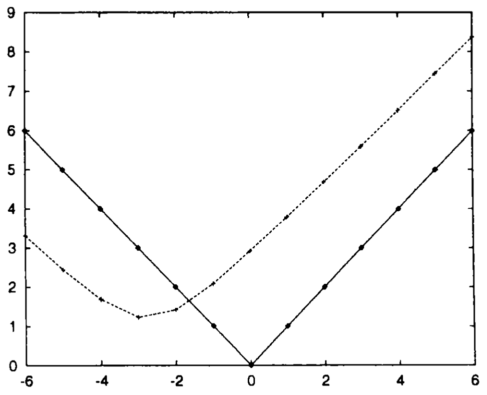


EX1a:

Table 4.2: The results of predicting the EX1a time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	4.540	5.496	5.391
4	4.903	4.521	4.325
8	2.920	3.824	3.802
	Number of Iterations		
2	20000	20000	20000
4	20000	20000	20000
8	20000	20000	20000

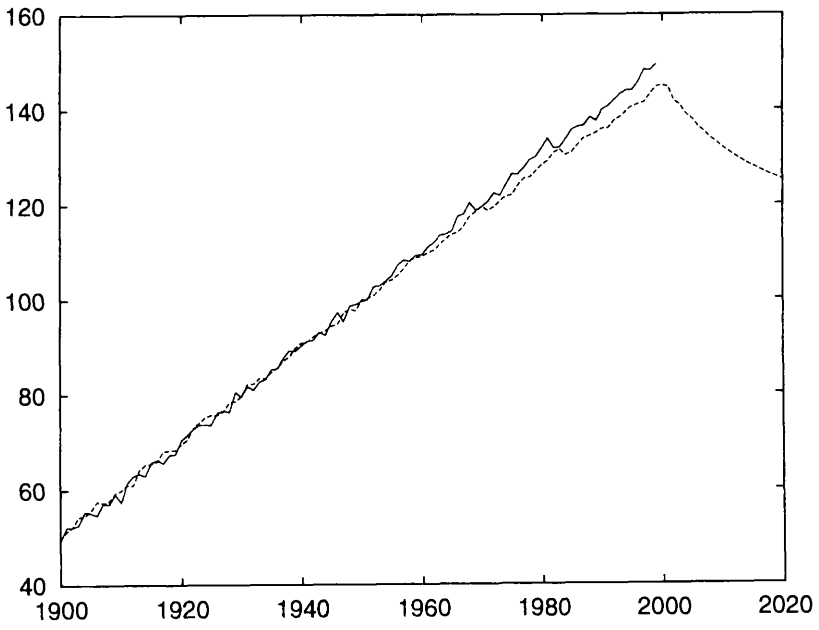


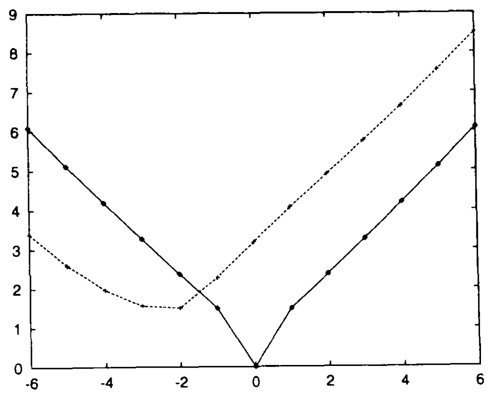


EX1b:

Table 4.3: The results of predicting the EX1b time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	4.756	5.767	5.915
4	5.087	4.750	4.739
8	3.181	4.098	4.117
	Number of Iterations		
2	20000	20000	20000
4	20000	20000	20000
8	20000	20000	20000

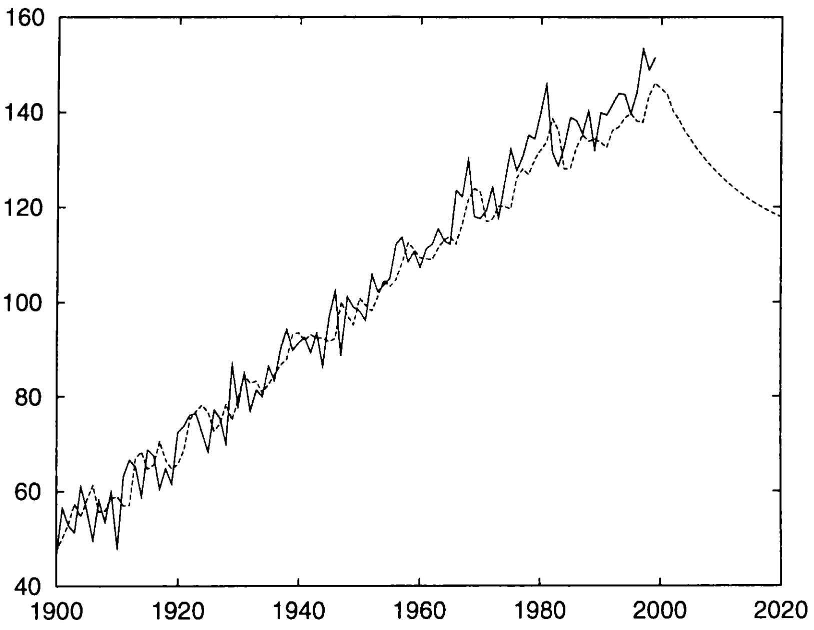


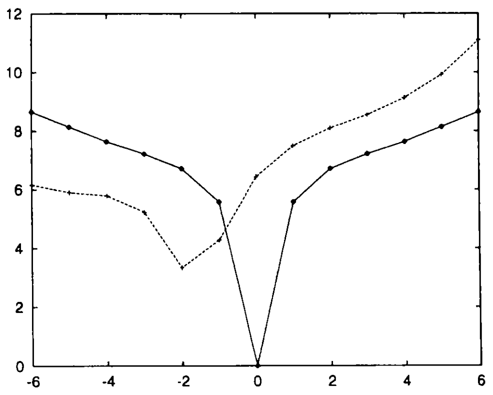


EX1c:

Table 4.4: The results of predicting the EX1c time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	7.529	8.545	9.458
4	7.771	7.435	8.052
8	6.426	6.908	7.221
	Number of Iterations		
2	20000	20000	20000
4	20000	20000	20000
8	20000	20000	20000





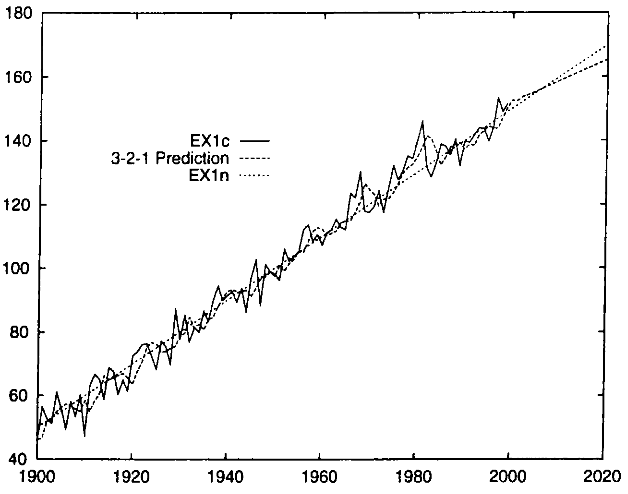


Figure 4.3: This plot shows the results of using an "analytically trained" 3-2-1 FFNN to predict the EX1c time series. The RMS error of these prediction, 5.37, is lower than any of the networks trained using back propagation. The expected RMS error for these predictions is 5.77.

As with the examples in the previous chapter the input weights can be chosen to be small enough to achieve an arbitrarily small error. A convenient solution to the above equations involves putting $v_j = w_{ji}$ and $H = 2$, which is possible because the RHS of the second equation is independent of i. Doing so allows the following prescription for finding a Network to predict the EX1 time series:

- Choose v_1 and v_2, where $v_1 \neq v_2$ to be small
- The output weights are then given by

$$W_1 = \frac{A_1 - v_2 A_0}{v_1 - v_2}$$
$$W_2 = \frac{A_1 - v_1 A_0}{v_2 - v_1}$$

Figure 4.3 shows predictions of EX1c using a 3-2-1 FFNN that had its weights specified in this way. The RMS error of these predictions, 5.37, is much better than any of the errors from networks trained by back-propagation, and is close to the expected RMS error of 5.77. To summarize, it has been shown

1. that only two hidden neurons are necessary to predict the EX1 time series as well as possible in the least square sense, for any number of inputs.

2. that the use of I inputs yields an expected RMS prediction error of $\sigma\sqrt{(1+\frac{1}{I})}$, i.e. the more inputs used the better the prediction

3. how to find the weights of such of network by the solution of a system of two equations in two unknowns

This analytically trained network will perform well on any realization of the EX1 time series where $v_j^{\max}x^{\max} \ll \pi$ is satisfied, $v_j^{\max}$ being the largest (in the absolute sense) input weight and $x^{\max}$ being the largest input. In contrast, back-propagation can only hope to train the network to perform well in the regions of the inputs that were represented in training.

EX2: $X_t = 100 + 100\sin\frac{2\pi}{100}t + a_t$. For this time series a training set of two periods was used i.e. $t = 0,\ldots,199$ with a third period serving as a test set, i.e. $t = 200,\ldots,299$. Over one time step the maximum change in the time series will be approximately

$$\Delta X_t^{\max} \sim 2\pi\cos 0 = 2\pi$$

so the variance of the noise in these tests is scaled accordingly, each of the four levels of noise receiving a label:

n No Noise

a Var[a_t]= $\frac{1}{16}\pi^2$

b Var[a_t]= π^2

c Var[a_t]= $16\pi^2$

Table 4.5 shows the test set RMS errors for FFNNs with linear *output* activation functions, with different numbers of input and hidden neurons - the output in each case only predicts one step ahead. The lower part of this table shows whether or not the time shift plot of the predictions showed any signs of the following:

Echoing	$\hat{X}_t = X_{t-1}$
Averaging	$\hat{X}_t = \frac{1}{2}(X_{t-1} + X_{t-2})$
Delay	Time shift plot minimum is not at $\tau = 0$ but neither echoing nor averaging is in evidence.

where $\hat{X}_t$ represents the network's prediction of X_t. All three appear on a time-shift plot as a minimum at $\tau < 0$, where delay and echoing have a single minimum at $\tau = -1$ and averaging

Table 4.5: The results of predicting sine plus noise with different configurations of NN. The upper part of the table shows RMS error on the test set for each configuration and noise level, while the lower part shows whether or not the predictions are affected by echo.

	No. of Inputs															
	2				4				8				12			
	Variance of Noise															
	n	a	b	c	n	a	b	c	n	a	b	c	n	a	b	c
	RMS error on Test Set															
2	1.49	2.35	6.44	17.05	0.65	1.55	5.06	17.02	1.34	1.76	5.26	17.40	0.92	3.82	6.42	17.16
6	1.21	2.12	6.44	17.05	0.85	1.44	5.10	17.32	0.71	1.56	4.76	16.88	0.83	1.47	5.09	15.25
12	0.87	2.13	6.68	17.34	0.20	1.47	5.10	17.21	0.11	1.68	4.68	17.15	3.04	2.12	6.50	17.27
	Is the Network Echoing?															
2	N*	N*	E	A	N*	N	D	A	N*	N	D*	D*	N*	N	D	D
6	N*	N	E	A	N*	N	D	D	N*	N*	D	D*	N*	N	N	D
12	N*	N*	E	E	N*	N*	D	D*	N*	N	D	D	N*	N	E	E

Key to table:

*	Test set error was still decreasing when training was stopped at 100,000 iterations
E	Predictions showed pure echoing
D	Predictions had lower RMS error when slid back one time step
A	Predictions showed pure averaging over most recent two inputs
N	Predictions showed none of the above

appears as a flat-bottomed minimum covering both $\tau = -1$ and $\tau = -2$. The distinction between echoing (or averaging) and delay is governed by the difference between the RMS error for the NN's prediction and the RMS error attained by echoing. If the two are the same, the predictions are classed as echoed, if the RMS error for prediction is significantly lower than that for echoing then the predictions are classed as delayed. In any case, inspection of the plot of NN predictions will clearly show if either pure echoing or averaging is present.

Before discussing the results of these tests it is of interest to examine the task the network is trying to learn. The FFNN must learn a function such that

$$X_t = P(X_{t-1}, X_{t-2}, \ldots, X_{T-I})$$

Table 4.6: The results of predicting sine plus noise with NNs of different configuration. In this test the NN's input activation functions were linear (by accident!), turning the network into a linear AR model. The learning parameter ϵ was initially taken to be 0.001 and altered after 100,000 training iterations as described in the text. The upper part of the table shows RMS error on the test set for each configuration and noise level, while the lower part shows whether or not the predictions are affected by echo.

	No. of Inputs															
	2				4				8				12			
	Variance of Noise															
	n	a	b	c	n	a	b	c	n	a	b	c	n	a	b	c
	RMS error on Test Set															
2	1.19	2.35	6.44	17.05	0.65	1.55	5.06	17.02	0.79	1.76	5.26	17.40	3.57	3.82	6.42	17.16
6	0.26	2.12	6.44	17.05	0.60	1.44	5.10	17.32	0.12	1.56	4.76	16.88	10^{-4}	1.47	5.09	15.25
12	0.26	2.13	6.68	17.34	0.82	1.47	5.10	17.21	0.56	1.68	4.68	17.15	3.04	2.12	6.50	17.27
	Is the Network Echoing?															
2	N*	N*	E	A	N	N	D	A	N*	N	D*	D*	N	N	D	D
6	N	N	E	A	N	N	D	D	N*	N*	D	D*	N	N	N	D
12	N	N*	E	E	N*	N*	D	D*	N	N	D	D	N*	N	E	E

expanding X_t in terms of X_{t-1} can be achieved as follows

$$\begin{aligned} X_t &= 100 + 100 \sin \frac{2\pi}{100} t \\ &= 100 + 100 \sin \frac{2\pi}{100} ((t-1)+1) \\ &= 100 + 100 \sin \frac{2\pi}{100}(t-1) \cos \frac{2\pi}{100} + 100 \cos \frac{2\pi}{100}(t-1) \sin \frac{2\pi}{100} \end{aligned}$$

now $\cos \frac{2\pi}{100} \sim 1$ and letting $s = \sin \frac{2\pi}{100}$

$$\begin{aligned} X_t &= X_{t-1} + 100s \frac{X_t - X_{t-1}}{|X_t - X_{t-1}|} \left(1 - \sin^2 \frac{2\pi}{100}(t-1)\right)^{\frac{1}{2}} \\ &= X_{t-1} + s \frac{X_{t-1} - X_{t-2}}{|X_{t-1} - X_{t-2}|} \left(2X_{t-1} - \frac{X_{t-1}^2}{100}\right)^{\frac{1}{2}} \end{aligned}$$

Another approximation has been made in using the sign of $X_{t-1} - X_{t-2}$ to correctly express the cosine in terms of the sine. The predictor function is therefore (approximately)

$$P(x,y) = x + s\frac{x-y}{|x-y|}\left(2x - \frac{x^2}{100}\right)^{\frac{1}{2}}$$

which is discontinuous along the line $y = x$. However, the network does not need to learn to fit the entire surface of $z = P(x,y)$, it only needs to learn the function for values of x and y on the curve given parametrically as

$$\begin{aligned} x &= 100 + 100 \sin \frac{2\pi}{100} t \\ y &= 100 + 100 \sin \frac{2\pi}{100}(t-1) \end{aligned}$$

So already it is obvious that the seemingly simple case of predicting a sinusoid corresponds to the learning of a complicated two dimensional function. It is also obvious that two inputs is the bare minimum and it seems likely that more than two hidden neurons will be required to represent the predictor function. Another fact that has been highlighted is that the predictor function need only be fitted in a restricted region, in this case on a curve on a surface instead of the whole surface or, as illustrated earlier in Figure 4.2, for a sinusoid with noise. All the approximations used above are improved as the time interval between time series elements is reduced, or equivalently as the period of the sinusoid to be learned is increased. The addition of noise will render the above scheme useless, essentially because the numerical gradient (implicitly used in the scheme) is rendered almost useless when noise is present. To compensate for the noise, more inputs will be needed so that the network can perform some kind of smoothing of input data. The results in Table 4.5 show two interesting features. Echoing and delay is more prevalent when the variance of the noise is large and when many hidden neurons are used. The first is expected, because as the noise is increased it seems reasonable that it will be more difficult for the network to learn the underlying predictor function. The second is somewhat surprising, but might indicate that the addition of the extra hidden neurons has again made it more difficult for the network to learn the predictor function. Note that in the case of no noise pure echoing results in an error of 2π, so that any RMS error of more than about 6.28 indicates that the network is not really producing any kind of useful prediction. If noise is present then the RMS error for the best prediction, i.e. $\hat{X}_t = 100 + 100 \sin \frac{2\pi}{100}t$, is just σ, whereas the largest RMS error incurred by echoing will be $\sqrt{2\sigma^2 + 4\pi^2}$. It is clear, therefore, that in predicting the sinusoid the error from echoed prediction is always considerably larger than for the best prediction method. So why, then, does the network echo? The question can also be applied to averaging because in some cases networks actually appear to average the most recent two inputs, rather than just echo the most recent one. The RMS error incurred by averaging is $\sqrt{2\sigma^2 + 9\pi^2}$, which is much larger than that of echoing. It can also be seen that averaging seems to occur in the presence of large variance noise when there are 2 or 4 inputs and 2 or 6 hidden neurons. This might suggest that, in the presence of high noise and, for networks with fewer neurons, training produces networks that average. Whether it is echoing or averaging that is the end result, the behaviour betrayed in these tests is consistent with training halting at a stable local minimum. These minima being termed "local" because they do not correspond to the prediction task the network was required to learn. There is, however, no easy way to be sure if this is really the case. One possible test is to take the same architecture and re-train it with different parameters starting each time from a different point in weight-space. If the global minimum is found then it seems safe to conclude that echoing is a local minimum pitfall of training. However, failure in these tests to find the "global" minimum proves nothing, though it may suggest that if a global minimum does exist a local minimum of echoing or averaging is more attractive to the network when trained by back-propagation. I do not perform these tests here.

4.4 Auto-Regressive Type Time Series

These are time series which can be written as

$$X_t = \sum_{i=1} b_i X_{t-i} + a_t$$

The time series I shall investigate in this section are

$$\text{NAR1)} \quad X_t = X_{t-1} + a_t$$

$$\begin{aligned}
\text{NAR2)}\quad X_t &= \frac{X_{t-1}+X_{t-2}}{2}+a_t \\
\text{NAR3)}\quad X_t &= \frac{\sqrt{X_{t-1}^2+X_{t-2}^2}}{2}+a_t \\
\text{NAR4)}\quad X_t &= X_{t-1}^2-X_{t-2}^2+a_t \\
\text{NAR5)}\quad X_t &= 0.5X_{t-1}-0.2X_{t-2}-0.4X_{t-3}-0.1X_{t-4}+a_t
\end{aligned}$$

In all of these time series the white noise series has $\sigma = 0.1$. The NAR1 series is a random walk time series and is actually non-stationary. The realization of it used for these tests is characteristic of a random walk in that it embarks on an excursion beneath zero, a phenomena which is referred to as a stochastic trend. Another realization might equally well show an excursion above zero or perhaps just small oscillations about zero. Since the same set of random numbers is used for all five of the NAR time series, it is not surprising to see the NAR2 and NAR3 time series behaving in a very similar manner. For linear AR series an analytic prescription for the networks can be derived as before. In the least squares sense the best predictor function is just

$$P_0(X_{t-I},\ldots,X_{t-1})=\sum_{i=1} b_i X_{t-i}$$

so equating terms with an $I-H-1$ network's output, given in (4.8), gives the set of $I+1$ equations

$$\begin{aligned}
\sum_{j=1}^{H} W_j &= 0 \\
\sum_{j=1}^{H} W_j w_{ji} &= \frac{b_i}{g_1}=A_i \qquad i=1,\ldots,I
\end{aligned}$$

that need to be satisfied. Since the above set of equations is linear in the ws as well as the W, for a FFNN to represent a general linear AR time series only 2 hidden neurons are needed. With $H=2$ and writing $\alpha=W_1=-W_2$, the above set of equations become

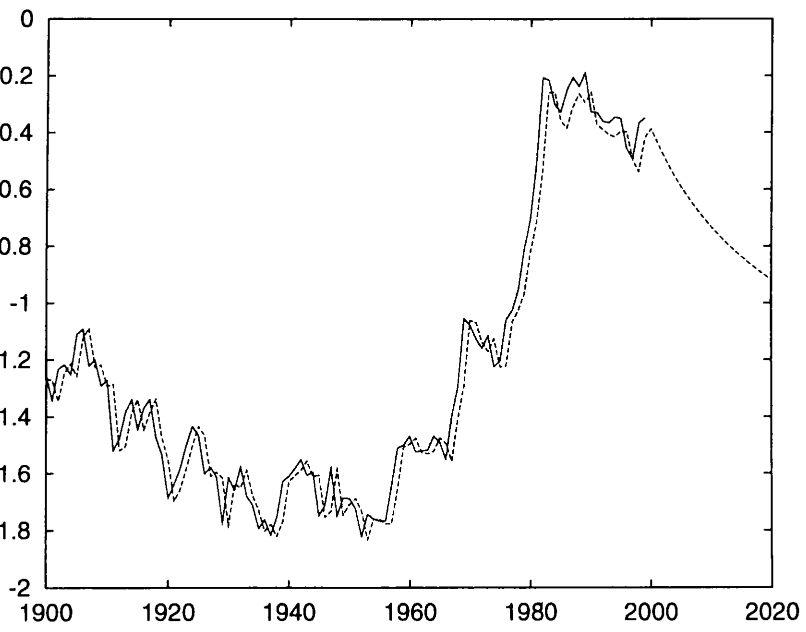
$$\begin{aligned}
W_1+W_2 &= 0 \\
\alpha(w_{1i}-w_{2i}) &= A_i \qquad i=1,\ldots,I
\end{aligned}$$

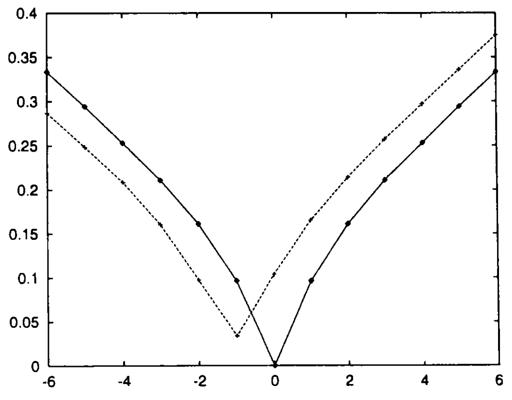
Choosing small input weights only requires that α is scaled up, so the approximation can easily be made as good as is needed. When dealing with the non-linear AR time series in this way the fact that $g_2 = 0$ for a sigmoid is once again an obstacle. This means that, in general, a FFNN with sigmoidal input activation functions and linear output activation functions cannot be expected to represent a non-linear AR time series predictor function to arbitrary accuracy. Of course, as mentioned before, a FFNN trained by back-propagation might find a close approximation to such a non-linear function over the range of inputs represented in the training set. In any case, the error due to the unpredictable noise term might well dominate any error incurred by failure to represent the predictor function exactly.

NAR1: The best prediction scheme for this time series is just an echo, i.e. $\hat{X}_t = X_{t-1}$, and it yields an error of 0.096 on the realization of the time series used here. None of the networks of Table 4.7 learned to echo particularly well, with the 4-2-1 network achieving the best error of 0.104. It is also apparent that the use of 8 inputs results in the poorer test set errors at the end of training.

Table 4.7: The results of predicting the NAR1 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.105	0.104	0.118
4	0.105	0.105	0.122
8	0.108	0.114	0.129
	Number of Iterations		
2	18475	20000	13366
4	15861	13407	18194
8	20000	10679	20000

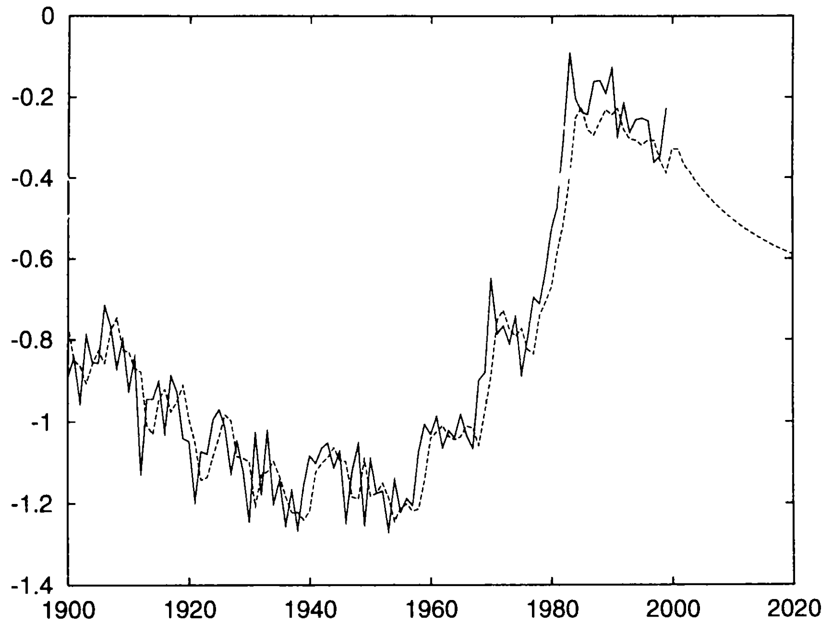


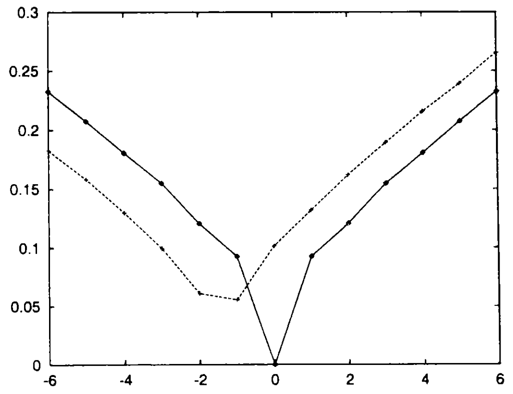


NAR2: In this case the best prediction scheme, $\hat{X}_t = \frac{X_{t-1}+X_{t-2}}{2}$, yields an error of 0.096. It would be expected that no other prediction scheme could beat this error, but it turns out that a simple echo scheme actually yields a slightly lower error of 0.092. This strange result is due to the particular realization of the time series that is used here. As can be seen from Table 4.8 none of the networks came particularly close to either of these errors. Again the larger networks, with 8 inputs or 8 hidden neurons had the worst errors at the end of training.

Table 4.8: The results of predicting the NAR2 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.104	0.102	0.112
4	0.101	0.105	0.120
8	0.108	0.112	0.128
	Number of Iterations		
2	7708	4870	3774
4	371	9290	12225
8	8677	6026	15503

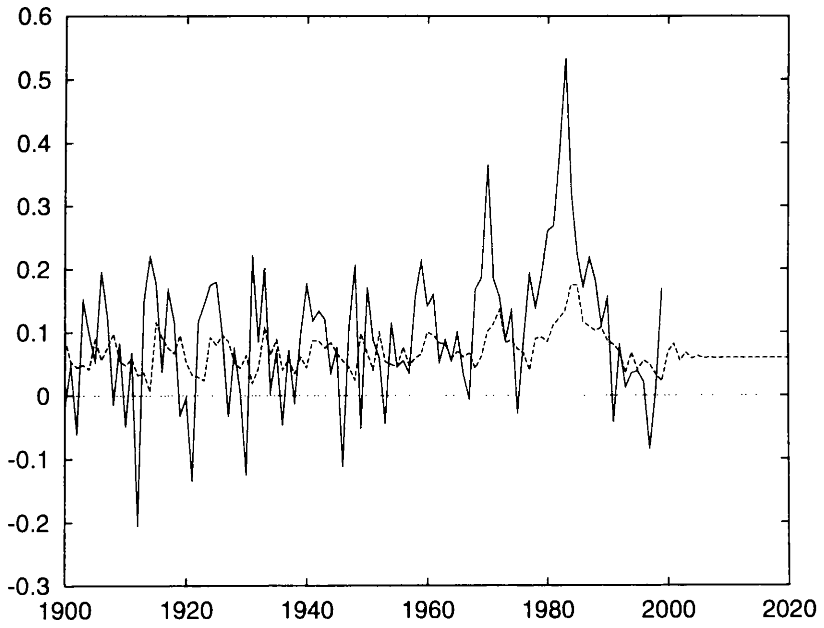


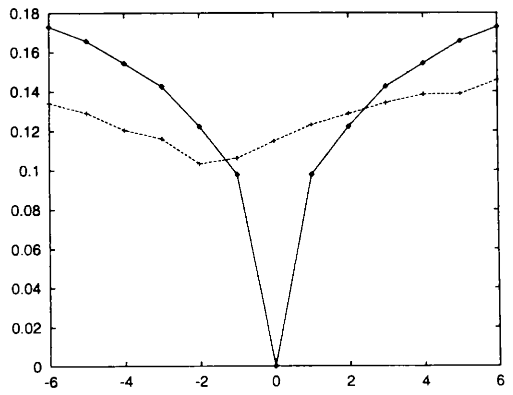


NAR3: Using the prediction scheme $\hat{X}_t = \frac{\sqrt{X_{t-1}+X_{t-2}}}{2}$, the error obtained is 0.093. Echoing gave rise to a prediction error of 0.097. The most successful networks in Table 4.9 were those fewer neurons, but still, none of networks improve upon echoing.

Table 4.9: The results of predicting the NAR3 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.117	0.115	0.143
4	0.117	0.129	0.128
8	0.116	0.130	0.122
	Number of Iterations		
2	8145	111	111
4	20000	4669	111
8	20000	6503	607

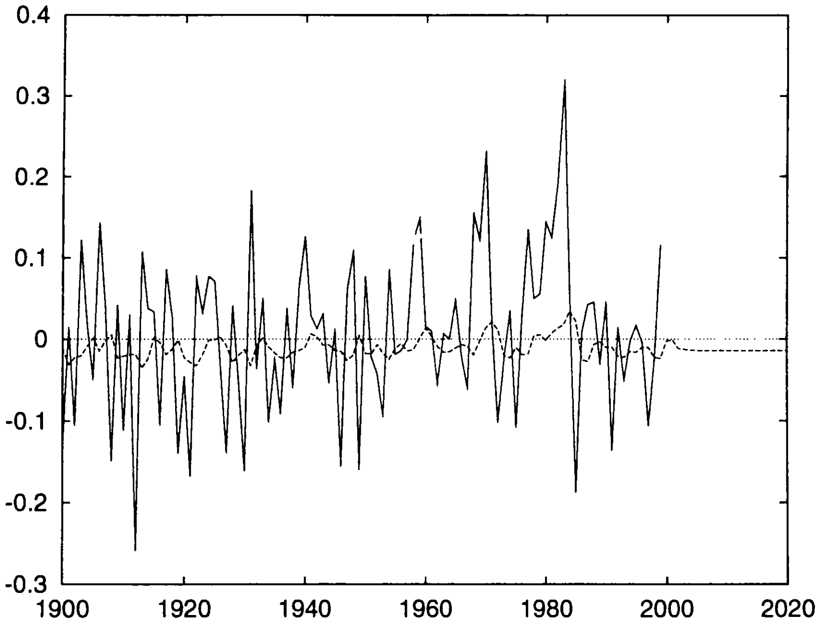


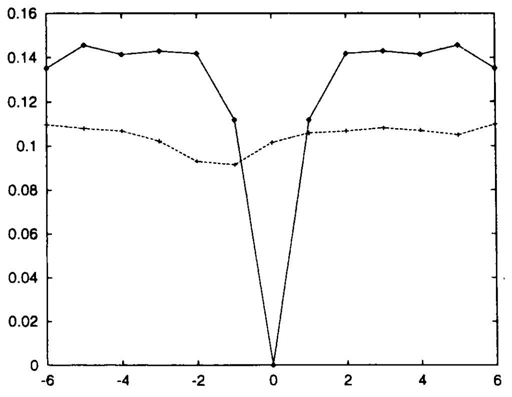


NAR4: Using the prediction scheme $\hat{X}_t = X_{t-1}^2 - X_{t-2}^2$ yields the error 0.093, with echoing giving an error of 0.111. As can be deduced from the form of the time series the predictable part, i.e. $X_{t-1}^2 - X_{t-2}^2$, is always going to be small and therefore swamped by the noise for a $\sigma = 0.1$. This explains why the predictions appear less spread around 0 than the data. In Table 4.10 it seems that all the networks have achieved reasonable success, at least equally or improving upon a simple echo. However, it is quite sobering to realise that if the absurdly simple prediction scheme, $\hat{X}_t = 0$, is used then the error is 0.099 - better than any of the networks.

Table 4.10: The results of predicting the NAR4 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.110	0.102	0.105
4	0.102	0.108	0.103
8	0.110	0.111	0.111
	Number of Iterations		
2	111	111	111
4	111	111	111
8	20000	588	111



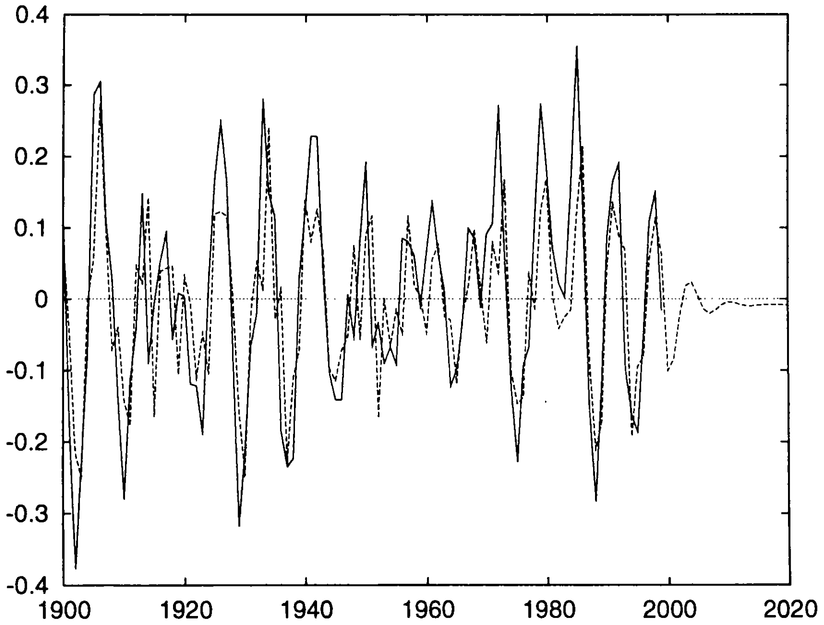


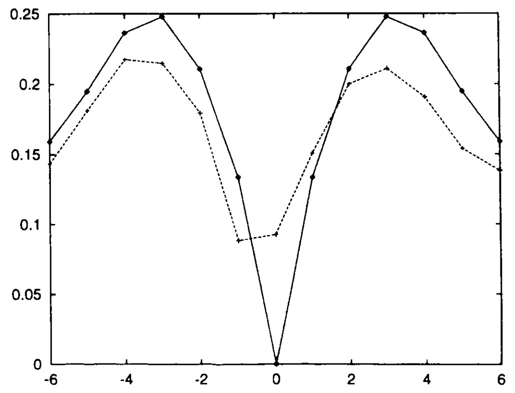
NAR5: In this example it is interesting to note how the network performs when given less than 4 inputs. The error of prediction by using $\hat{X}_t = 0.5X_{t-1} - 0.2X_{t-2} - 0.4X_{t-3} - 0.1X_{t-4}$ as a prediction should yield the best possible expected RMS error of 0.1. Using this scheme on the realization of the data used here yielded an actual error of 0.095. It comes as a surprise that the networks with only 2 inputs perform best, just beating the error of the best possible prediction scheme. If the two term prediction scheme $\hat{X}_t = 0.5X_{t-1} - 0.2X_{t-2}$ is used, then the actual error obtained is 0.1054, so one might expect that a network with only two inputs would be unable to improve on this. As with the other examples inclusion of more input neurons than necessary seems to be detrimental to the training of the network. The second test set used in Table 4.11 consists of data in the reduced range of 1960-1999. The result is that all the networks have had their errors slightly increased, so that now the 2 input networks do not beat the best prediction scheme error of 0.095. This suggests that the above surprising result can be explained as a statistical fluke.

Table 4.11: The results of predicting the NAR5 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.094	0.114	0.107
4	0.093	0.109	0.136
8	0.093	0.121	0.106
	RMS error on 1960-1999		
2	0.096	0.120	0.111
4	0.095	0.113	0.143
8	0.095	0.127	0.110

Hidden	*No. of Inputs* 2	4	8
	Number of Iterations		
2	20000	284	160
4	2301	257	220
8	1850	111	1898





4.5 Moving-Average Type Time Series

The time series used here are

$$\begin{aligned}
\text{NMA1)}\quad X_t &= a_{t-1} \\
\text{NMA2)}\quad X_t &= \frac{a_{t-1}+a_{t-2}}{2} \\
\text{NMA3)}\quad X_t &= \frac{\sqrt{a_{t-1}^2+a_{t-2}^2}}{2} \\
\text{NMA4)}\quad X_t &= a_{t-1}^2 - a_{t-2}a_{t-1} \\
\text{NMA5)}\quad X_t &= 0.5a_{t-1} - 0.2a_{t-2} - 0.4a_{t-3} - 0.1a_{t-4}
\end{aligned}$$

Here the best prediction scheme is not as obvious as it was for the NAR time series. To illustrate some of the properties of a moving average time series consider the linear case:

$$X_t = \sum_{n=1}^{N} \theta_n a_{t-n}$$

The (unconditional) expectation of a linear moving average time series is just 0, and if $\hat{X}_t = 0$ is used as a prediction scheme then the expected RMS error will be given by the root of the variance, the variance being given by

$$E[X_t^2] = \sigma^2 \sum_{n=1}^{N} \theta_n^2 \tag{4.8}$$

Echoing, on the other hand, leaves the residuals Y_t, giving an expected square error derived as follows

$$\begin{aligned}
E[Y_t^2] &= E[(X_t - X_{t-1})^2] \qquad (4.9)\\
&= \sigma^2\theta_1^2 + \theta_N^2 + \sum_{n=2}^{N}(\theta_n - \theta_{n-1})^2
\end{aligned}$$

So whether echoing or predicting using 0 (zero prediction) is superior depends on the coefficients of the series in question. Either way no information is gained about the future by using such trivial prediction schemes, so that improvement on the lesser of these two errors indicate that a particular prediction scheme is of some merit.

A FFNN with I inputs uses the last I time series elements to construct its prediction, so it is useful to re-write a MA series in terms of an auto-regressive series, if indeed it is possible to do so. For simplicity, I only examine a simple linear MA model, i.e.

$$X_t = \alpha a_{t-1} - \beta a_{t-2}$$

By successive substitution the following pseudo-AR type expression can be obtained:

$$X_t = \alpha a_{t-1} - \beta \left(\frac{\beta}{\alpha}\right)^I a_{t-I-2} - \sum_{n=1}^{I} \left(\frac{\beta}{\alpha}\right)^n X_{t-n}$$

So using the prediction scheme

$$\hat{X}_t = -\sum_{n=1}^{I} \left(\frac{\beta}{\alpha}\right)^n X_{t-n}$$

will result in an expected square error of

$$E[(X_t - \hat{X}_t)^2] = \sigma^2\alpha^2 + \beta^2 \left(\frac{\beta}{\alpha}\right)^{2I}$$

Consider the case of a second order linear MA time series formed from white noise with $\sigma = 1$, the three prediction schemes outlined above then give rise to the following expected square errors

Zero prediction	$\hat{X}_t = 0$	$E_0 = \alpha^2 + \beta^2$
Echoing	$\hat{X}_t = X_{t-1}$	$E_{\text{echo}} = \alpha^2 + \beta^2 + (\alpha + \beta)^2$
Pseudo-AR	$\hat{X}_t = -\sum_{n=1}^{I} \left(\frac{\beta}{\alpha}\right)^n X_{t-n}$	$E_I = \alpha^2 + \beta^2 \left(\frac{\beta}{\alpha}\right)^{2I}$

Firstly consider the case where $\left|\frac{\beta}{\alpha}\right| < 1$. Then

$$E_I < E_0 < E_{\text{echo}}$$

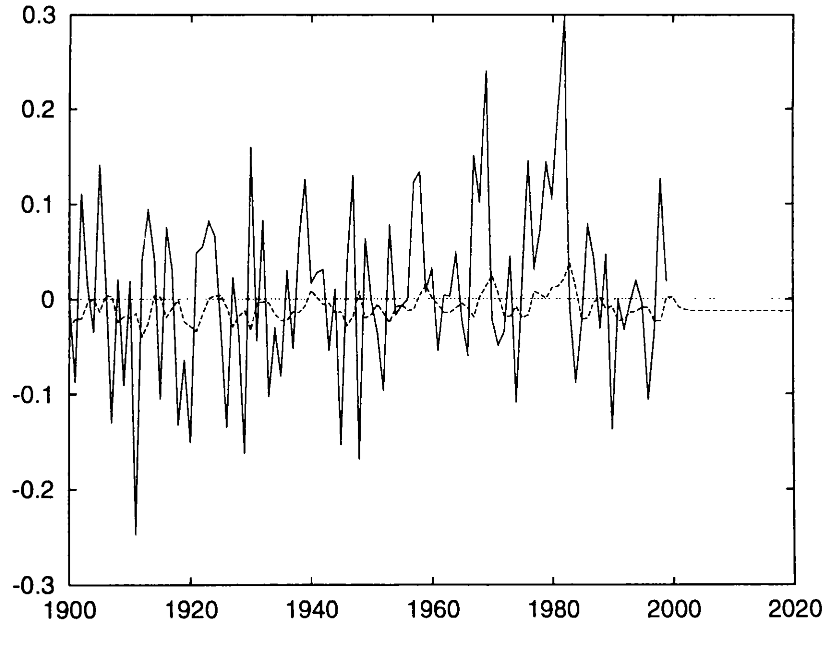
where E_I decreases as I increases, and in the limit of large I, $E_I \sim \alpha^2$, which is the smallest achievable error. If, on the other hand, $\left|\frac{\beta}{\alpha}\right| > 1$, then the pseudo-AR method becomes useless and the zero prediction method becomes the best. Finally, if $\left|\frac{\beta}{\alpha}\right| = 1$ then

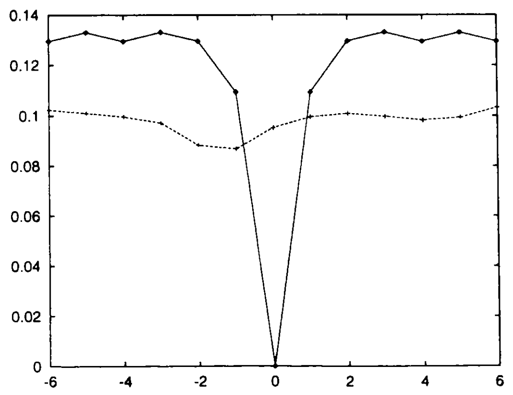
$$E_I = E_0 \leq E_{\text{echo}}$$

NMA1: There is of course no way of predicting genuine white noise, other than the trivial prediction scheme $\hat{X}_t = 0$. Doing so results in an expected RMS error of $\sigma = 0.1$. Over the test set the actual error was found to be 0.096, which is roughly what all the network's achieved. The networks, however, did not learn to predict this way, that is they did not learn to ignore their inputs and output 0 (zero output weights could achieve this, for example). As can be seen from both the prediction plot and the time-shift plot below, the network's output does still depend (weakly) on its inputs, but the variance of the predictions is certainly significantly smaller than the time series' own variance. Echoing has an expected RMS error of $\sqrt{2}\sigma = 0.141$, though on this realization of the time series the actual error was found to be 0.108.

Table 4.12: The results of predicting the NMA1 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.101	0.096	0.101
4	0.095	0.103	0.097
8	0.100	0.102	0.102
	Number of Iterations		
2	4407	111	111
4	111	111	111
8	2969	896	314

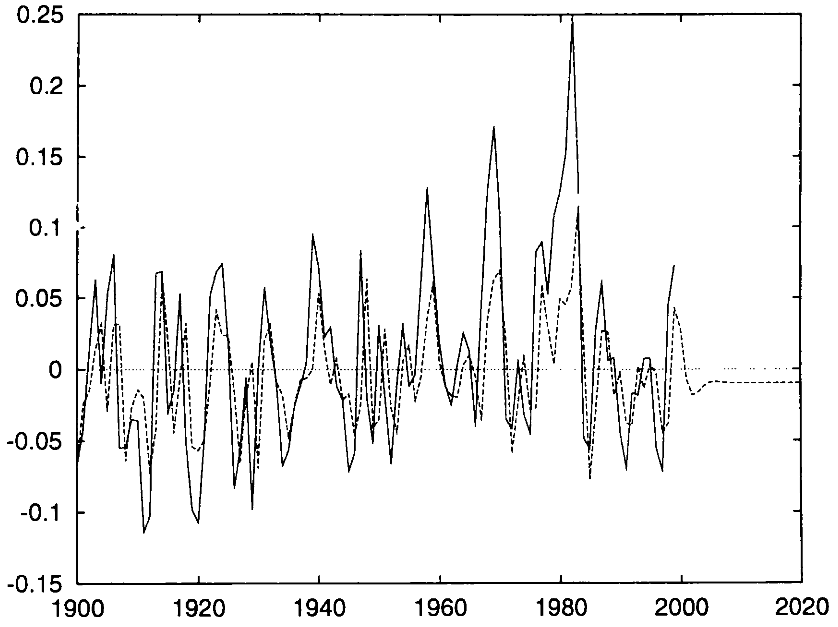


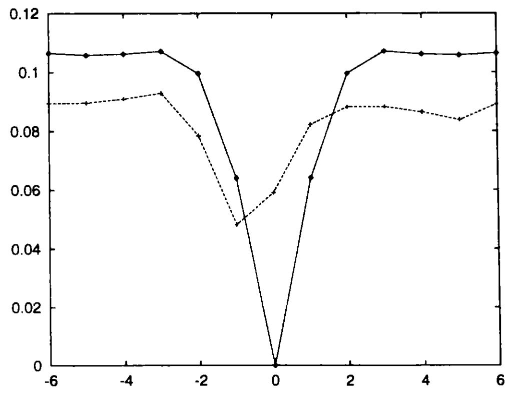


NMA2: In this case the expected RMS errors for echoing and zero prediction are the same, being $0.1/\sqrt{2} = 0.071$. For the realization of the time series used here the errors were 0.063 and 0.079 respectively. If a pseudo-AR method is used then the expected RMS error is again $0.1/\sqrt{2} = 0.071$, though the error on this realization was actually 0.068. The best network has improved on all of these prediction schemes, though as can be seen from the plot below its predictions tend to underestimate the magnitude of the time series.

Table 4.13: The results of predicting the NMA2 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.060	0.062	0.083
4	0.060	0.061	0.080
8	0.059	0.060	0.060
	Number of Iterations		
2	4772	6756	111
4	4321	4894	116
8	16806	10760	15092

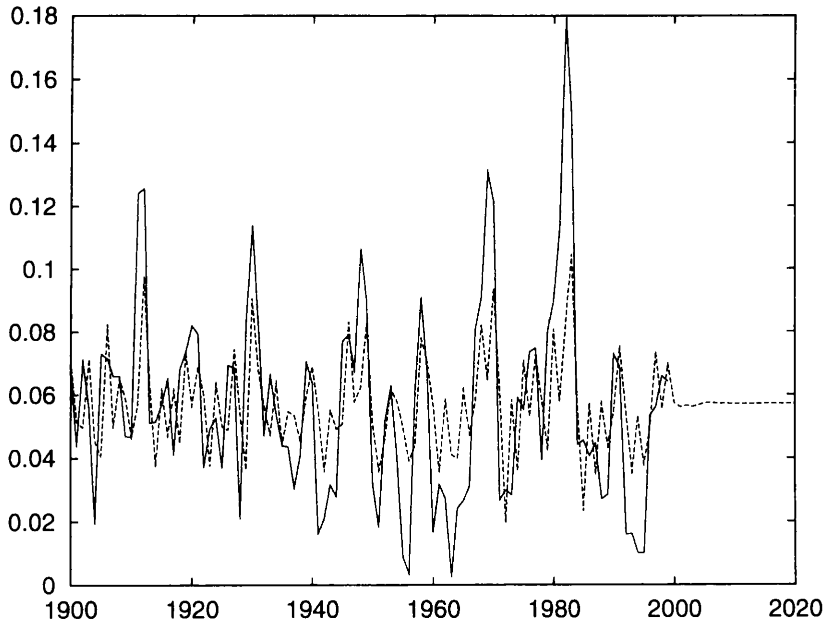


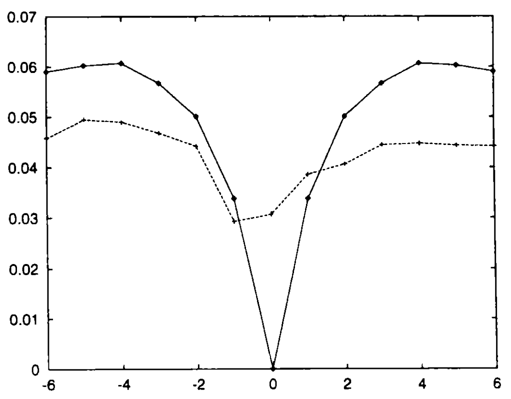


NMA3: Here the zero prediction method is not appropriate as this time series is strictly positive. Echoing yielded an error of 0.033, which was only slightly improved by all the networks with more than 2 inputs.

Table 4.14: The results of predicting the NMA3 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.035	0.031	0.032
4	0.035	0.031	0.031
8	0.035	0.032	0.031
	Number of Iterations		
2	20000	20000	20000
4	111	20000	20000
8	20000	20000	20000

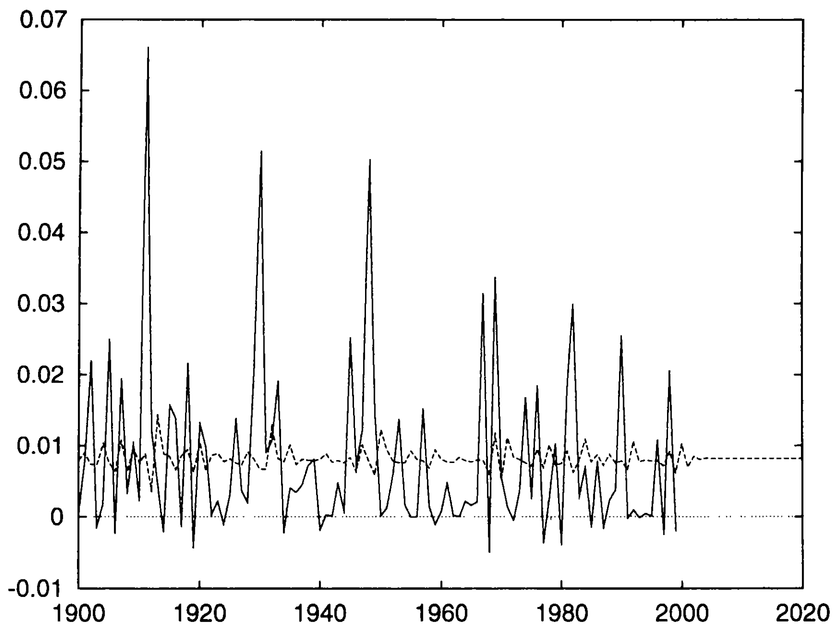


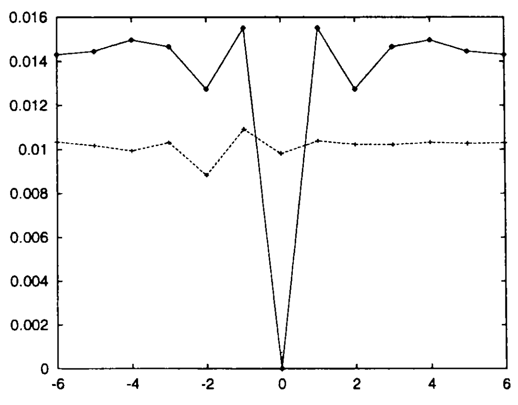


NMA4: In this case it can be shown that the (constant) expectation value of this time series is just $\sigma^2 = 0.01$. Using this as a prediction method, i.e. $\hat{X}_t = 0.01$, yields and error of 0.011, whereas echoing achieves an error of 0.015. As seen from Table 4.15, all the networks achieve an error of 0.01. The prediction plot and time shift plot both indicate that the network's prediction is rather flat, that is it is consistently outputting a value of around 0.08. Also notice that the large spikes are not predicted well at all, and that they almost always cause the network's next prediction to be an overestimate.

Table 4.15: The results of predicting the NMA4 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.010	0.010	0.010
4	0.010	0.010	0.010
8	0.010	0.010	0.010
	Number of Iterations		
2	111	7086	20000
4	20000	20000	20000
8	143	1263	1764

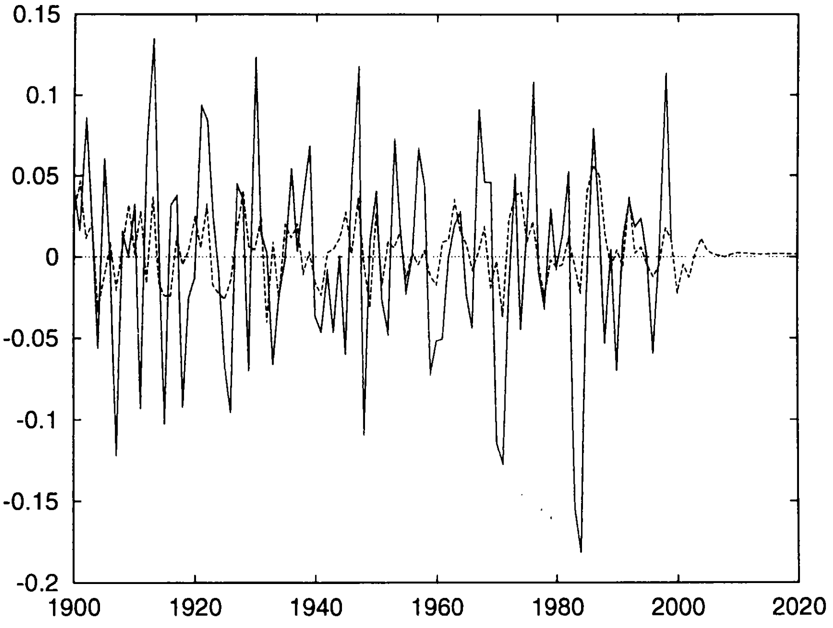


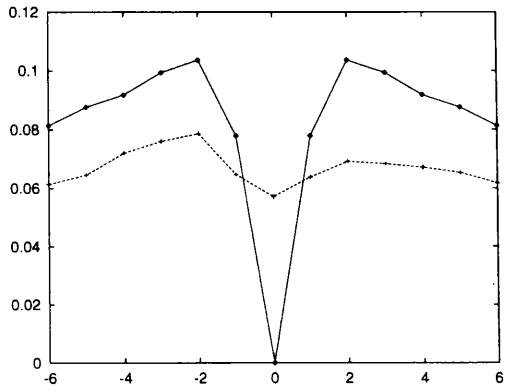


NMA5: The expectation for this linear MA model is just zero and from (4.8), the expected RMS error of zero prediction is just 0.068. On the realization of the time series used the error of zero prediction was actually 0.062. The expected RMS error for echoing from (4.10) was found to be 0.070, the error on the data used here being 0.077. All the networks performed reasonably well on the data, apart from the networks with 8 inputs. The prediction plot shows that most of the peaks in the data were were predicted, though underestimated in magnitude. This fact is confirmed in the time shift plot because the minimum is at a shift of 0.

Table 4.16: The results of predicting the NMA5 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.057	0.058	0.058
4	0.057	0.057	0.063
8	0.057	0.057	0.066
	Number of Iterations		
2	4349	7804	111
4	5103	1189	111
8	2291	871	2639





4.6 Auto-Regressive Moving-Average Type Time Series

In this section the following time series are used for prediction:

$$\begin{aligned}
\text{NAM1)}\quad X_t &= X_{t-1} + a_t + 0.4a_{t-1} - a_{t-2} + 0.6a_{t-3} \\
\text{NAM2)}\quad X_t &= \frac{X_{t-1} + X_{t-2}}{2} + a_t + 0.4a_{t-1} - a_{t-2} + 0.6a_{t-3} \\
\text{NAM3)}\quad X_t &= \frac{\sqrt{X_{t-1}^2 + X_{t-2}^2}}{2} + a_t + 0.4a_{t-1} - a_{t-2} + 0.6a_{t-3} \\
\text{NAM4)}\quad X_t &= X_{t-1}^2 - X_{t-2}^2 + a_t + 0.4a_{t-1} - a_{t-2} + 0.6a_{t-3} \\
\text{NAM5)}\quad X_t &= 0.5X_{t-1} - 0.2X_{t-2} - 0.4X_{t-3} - 0.1X_{t-4} + a_t + 0.4a_{t-1} - a_{t-2} + 0.6a_{t-3}
\end{aligned}$$

Methods of predicting linear ARMA time series have been well established and can be found in many standard texts, notably Box and Jenkins (1970) and Cryer (1986). However, I do not attempt to use them here. In the following I shall give the error obtained if predictions are made using the Auto-regressive part of the time series only. This, of course, is not the best possible method, as it is possible to exploit the moving average structure to obtain more accurate predictions.

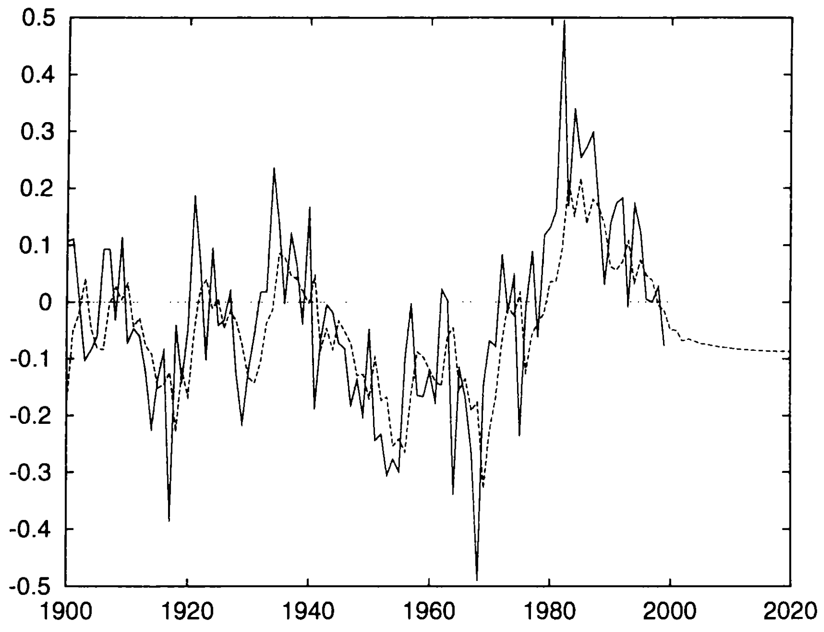
4.7 The Learning Rate

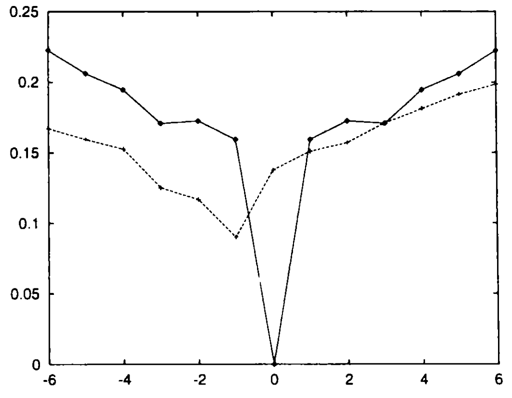
The results of the previous sections do not really allow conclusions such as "network A-B-C is best for predicting time series X" to be made. For example, it was shown earlier that the NAR1 time series (the random walk whose best prediction is just an echo) can be predicted by a 2-2-1 network. However, according to Table 4.7 the 4-4-1 network has the lowest test set error at the end of training. The only conclusion that can be stated in this case is that, for the starting set of weights used, and for a learning rate of 0.01 and momentum of 0.9, and for the training scheme and stopping criterion adopted, the best network, in terms of error on the particular test set used, for predicting the NAR1 time series was a 4-4-1 network. So it seems that the practical study of how different FFNNs predict different time series is about as elegant as the last sentence. In an attempt to remedy this situation I shall repeat some of the training runs of the last section but with different learning rates, each time starting the networks from the same starting point as before. Below are a few reasons why different learning rates should give different end results. Some evidence for the reasoning behind these ideas can be found in Section 3.1.2 of the last chapter.

NAM1: The errors for echoing, zero prediction and AR only (this is in fact echoing in this instance) are 0.158, 0.190 and 0.158. So the Table 4.17 shows that the network's are doing significantly than just an echo.

Table 4.17: The results of predicting the NAM1 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.141	0.138	0.150
4	0.142	0.140	0.146
8	0.143	0.144	0.147
	Number of Iterations		
2	1982	3673	2838
4	1126	1552	2487
8	1618	1352	279



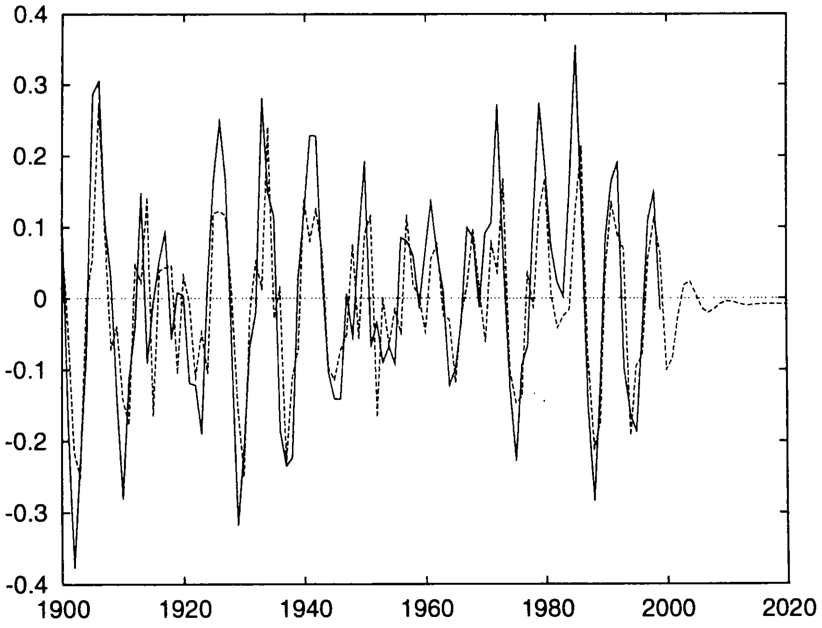


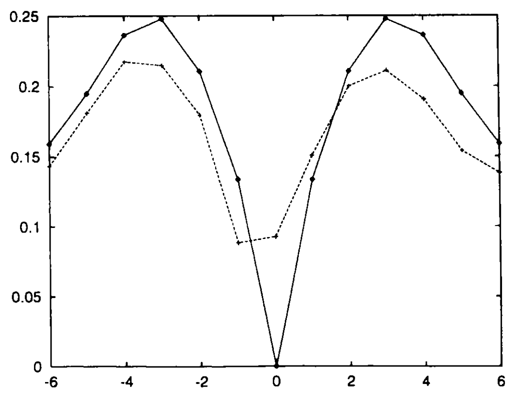
NAR5: In this example it is interesting to note how the network performs when given less than 4 inputs. The error of prediction by using $\hat{X}_t = 0.5X_{t-1} - 0.2X_{t-2} - 0.4X_{t-3} - 0.1X_{t-4}$ as a prediction should yield the best possible expected RMS error of 0.1. Using this scheme on the realization of the data used here yielded an actual error of 0.095. It comes as a surprise that the networks with only 2 inputs perform best, just beating the error of the best possible prediction scheme. If the two term prediction scheme $\hat{X}_t = 0.5X_{t-1} - 0.2X_{t-2}$ is used, then the actual error obtained is 0.1054, so one might expect that a network with only two inputs would be unable to improve on this. As with the other examples inclusion of more input neurons than necessary seems to be detrimental to the training of the network. The second test set used in Table 4.11 consists of data in the reduced range of 1960-1999. The result is that all the networks have had their errors slightly increased, so that now the 2 input networks do not beat the best prediction scheme error of 0.095. This suggests that the above surprising result can be explained as a statistical fluke.

Table 4.11: The results of predicting the NAR5 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.094	0.114	0.107
4	0.093	0.109	0.136
8	0.093	0.121	0.106
	RMS error on 1960-1999		
2	0.096	0.120	0.111
4	0.095	0.113	0.143
8	0.095	0.127	0.110

Hidden	*No. of Inputs* 2	4	8
	Number of Iterations		
2	20000	284	160
4	2301	257	220
8	1850	111	1898

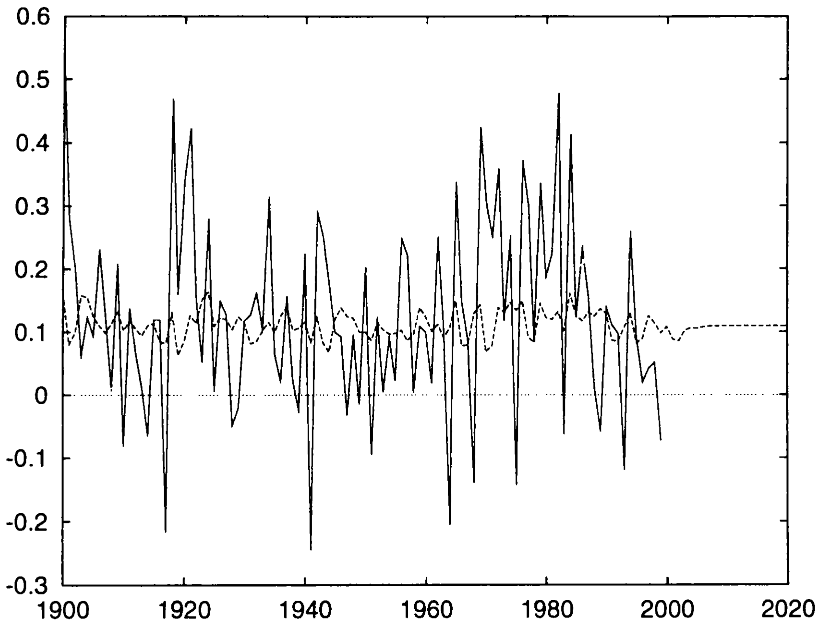


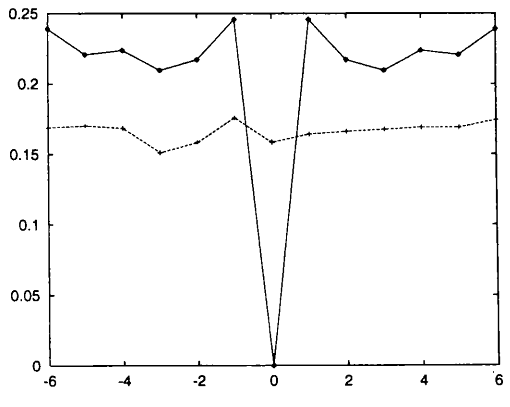


NAM3: The errors for echoing, zero prediction and AR only are 0.243, 0.214 and 0.158. Here only the 4-4-1 network has managed to equal the AR only prediction, though all the networks improve greatly on the trivial prediction methods. As can be seen from both prediction plot and the time shift plot, the network's predictions are rather flat, varying around the the value of 0.1.

Table 4.19: The results of predicting the NAM3 time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	4	8
	RMS error on Test Set		
2	0.162	0.159	0.161
4	0.161	0.158	0.166
8	0.164	0.163	0.160
	Number of Iterations		
2	111	15619	20000
4	20000	20000	436
8	224	1121	169

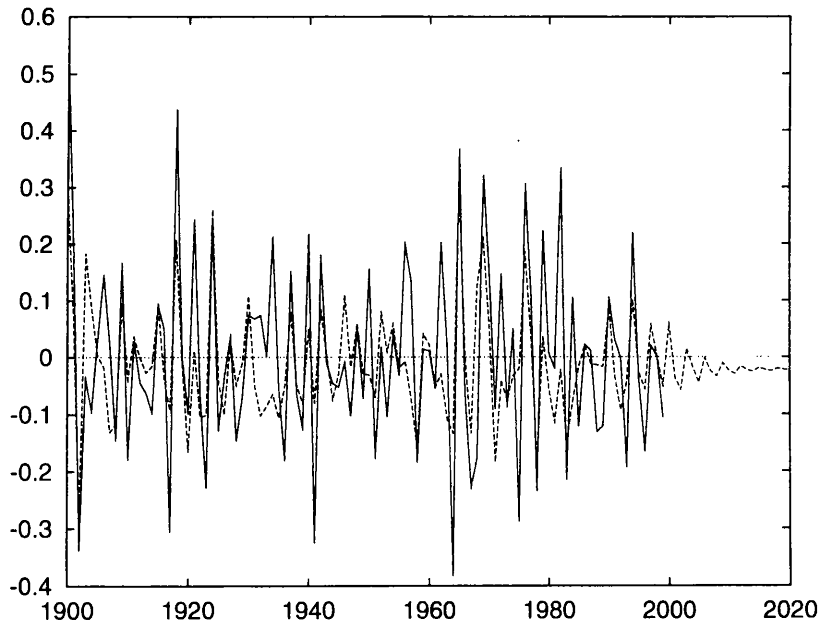


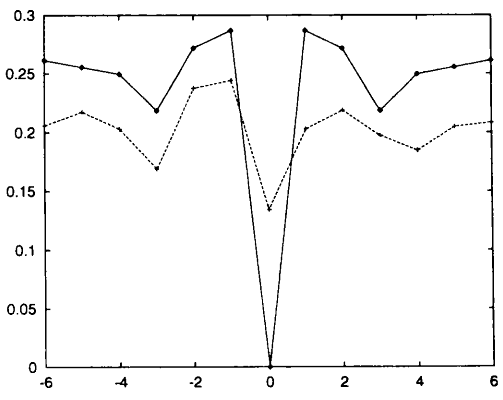


NAM4:The errors for echoing, zero prediction and AR only are 0.284, 0.172 and 0.158. In this case it is the networks with 8 inputs that perform best, though all the networks beat the AR only error. The networks' errors are all a vast improvement upon echoing but only a slight improvement upon the zero prediction. This can be understood by realizing that the linear MA terms will dominate the second order AR terms in this time series.

Table 4.20: The results of predicting the NAM4 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.143	0.146	0.138
4	0.142	0.144	0.143
8	0.143	0.141	0.134
	Number of Iterations		
2	20000	20000	20000
4	20000	20000	20000
8	20000	20000	20000

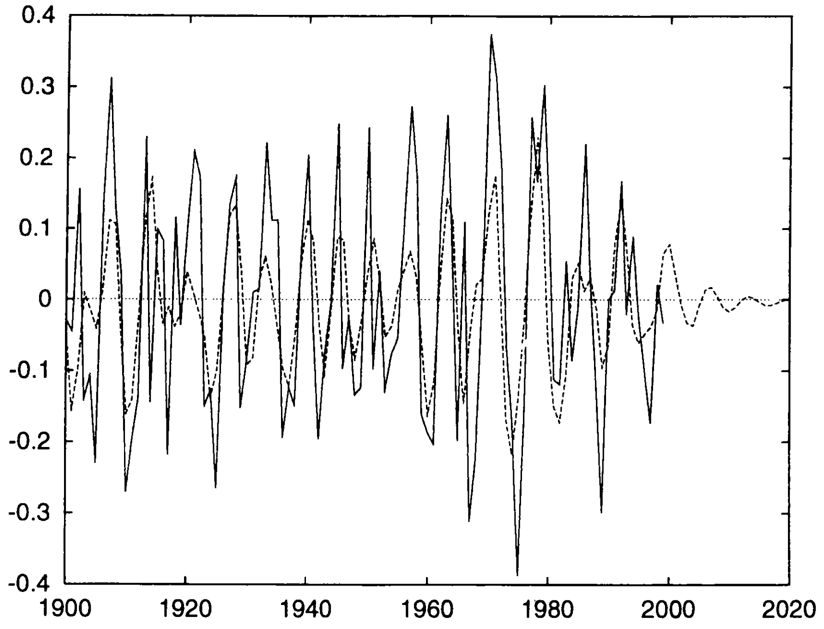


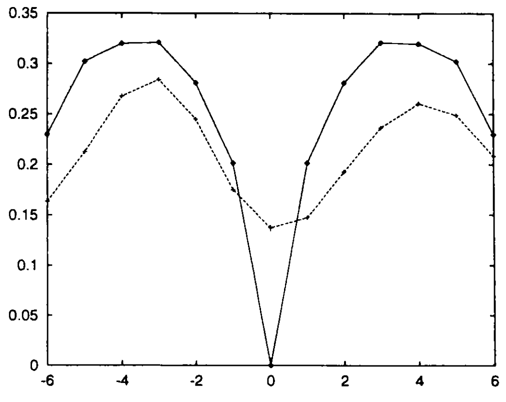


NAM5: The errors for echoing, zero prediction and AR only are 0.199, 0.180 and 0.139. In this case the networks were unable to improve significantly upon the AR only error. It is clear from the errors tabulated below that the networks with only two inputs are not able to predict this time series very well.

Table 4.21: The results of predicting the NAM5 time series with different architectures of FFNN.

	No. of Inputs		
Hidden	2	4	8
	RMS error on Test Set		
2	0.165	0.138	0.141
4	0.165	0.137	0.140
8	0.165	0.138	0.142
	Number of Iterations		
2	4871	1351	1256
4	7648	843	1672
8	20000	363	6791





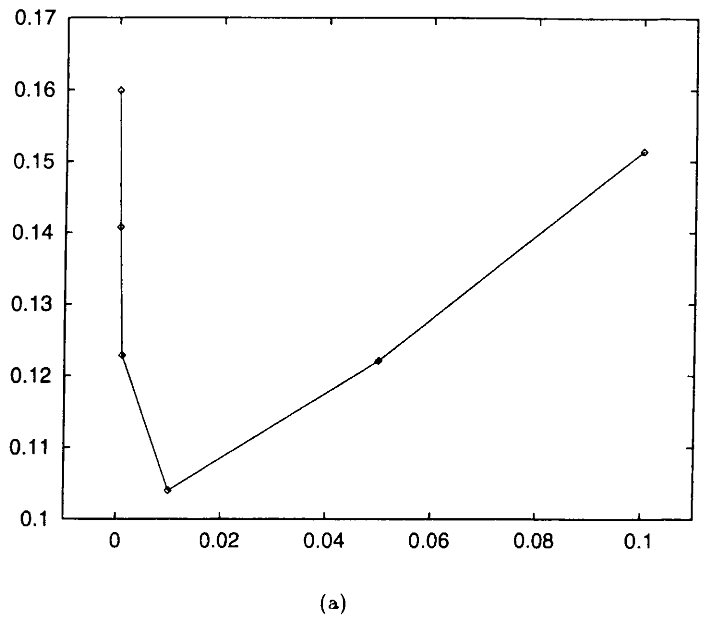


Figure 4.4: The typical shape of final test set error plotted against learning rate ϵ. The data here is for the 4-2-1 network of Table 4.22.

1. Too large a learning rate can cause training to become unstable earlier. That is, a smaller value of learning rate should allow the the network to be trained to a lower final error

2. Going to the other extreme, too small a learning rate can cause training to tip-toe over-cautiously through weight-space. This is not relevant to most of the cases in the previous sections because training was usually terminated before the 20,000 iteration limit was reached.

3. If two networks are started from the same point in weight space, but are trained with different learning rates, they might both converge on different solutions, or converge on the same solution via different paths. This being the case it is quite possible, for example, for the network trained with the lower learning rate to end up with the higher error.

The results presented in Tables 4.22, 4.23 and 4.24 illustrate that the first and second points in the above list seem to constitute the rule, whilst the third point is the exception. In fact for the networks in the tables below, the final test set error, if plotted against learning rate, would have a U-shaped profile, as shown in Figure 4.4.

Table 4.22: The results of training networks from Table 4.7, on the NAR1 time series, with different values of learning rate ϵ. Using the best possible prediction scheme, which is just an echo, the prediction error was 0.096.

	Learning Rate ϵ				
Network	0.0001	0.0005	0.001	0.05	0.1
2-2-1	0.145	0.138	0.132	0.117	0.138
4-2-1	0.160	0.141	0.123	0.122	0.151
8-8-1	0.216	0.149	0.139	0.190	0.249
	Number of Iterations				
2-2-1	20000	20000	20000	3321	1572
4-2-1	20000	20000	20000	3078	1493
8-8-1	20000	20000	20000	12076	3770

Table 4.22 shows the results of repeating the training of three of the networks used to predict the NAR1 series. The networks chosen were, 4-2-1 (previous error 0.104), 8-8-1 (previous error 0.129) and the 2-2-1 network (previous error 0.105). Remember that the 2-2-1 network is the smallest network capable of producing an echo, the best prediction scheme for the NAR1 series. None of these re-trained networks improve on either the best error of 0.104, set by the 4-2-1 network, or improve on their own previous errors. So the choice of $\epsilon = 0.01$ used before was a good one[3].

Similarly Table 4.23 shows the results for the networks predicting the NAR2 series. Here the errors achieved before using $\epsilon = 0.01$ were 2-2-1 (error 0.104), 4-2-1 (error 0.102) and 8-8-1 (error 0.128). Again no significant improvement has been gained by using different values of ϵ, though the 8-8-1 has improved to 0.121 with $\epsilon = 0.005$.

The networks trained on the NMA5 time series showed instability very early on in training, for example, in Table 4.16 2 networks stopped as soon as they got the chance, i.e. at 111 iterations.. So the question is: can these networks go on to produce smaller errors if a smaller learning rate is used. In Table 4.24, I show the results of using smaller learning rates in training the networks with 8 inputs. As can be seen the final error is fairly insensitive to the learning rate used, with training proceeding beyond 111 iterations only if a small value of learning rate is used. To see if an even smaller learning rate can provide any improvement I have repeated the training of the 8-4-1 network using learning rates as low as 10^{-7}. So the conclusion seems to be that these networks cannot be pushed any further with back propagation. It is also interesting to note that the inclusion of more hidden neurons seems to result in a higher final error. Since an 8-8-1 FFNN is capable of

[3]Note that without making any rash claims, I would like to point out that the choice of $\epsilon = 0.01$ was not plucked from a hat, nor based solely on the results of previous numerical tests, rather the choice was based on a good guess guided by experience.

Table 4.23: The results of training networks from Table 4.8, on the NAR2 time series, with different values of learning rate ϵ.

	Learning Rate ϵ				
Network	0.0001	0.0005	0.001	0.05	0.1
2-2-1	0.108	0.106	0.104	0.115	0.134
4-2-1	0.117	0.109	0.103	0.117	0.141
8-8-1	0.159	0.125	0.121	0.172	0.220
	Number of Iterations				
2-2-1	20000	20000	20000	1375	639
4-2-1	20000	20000	20000	940	472
8-8-1	20000	20000	20000	11272	2946

doing anything an 8-4-1 network can do the conclusion must be that networks with greater numbers of hidden neurons are more difficult to train by back-propagation. The results for the NAR time series in Tables 4.22 and 4.23 would seem to confirm this conclusion. As can be seen there is no improvement of error beyond 0.063, though the number of iterations required to achieve the error is increased for the lower learning rates.

Partly by chance the learning rate of 0.01 used initially in the architecture tests appeared to be the best for almost all cases. The above tables show that, had a different learning been chosen, then, within reason, the final test set RMS error would not be greatly increased. Even when training was stopped at the first opportunity, i.e. 111 iterations, training with a lower learning rate increased the number of training iterations without significantly reducing the resultant RMS error.

4.8 Different Starting Points

How does the starting point in weight space affect the end result of training? It seems from common practice, e.g. Hertz et al. (1991) and *Conway (1993), that the best initial values for the weights are those that are small and random e.g. randomly distributed around 0 with variance 0.1. Sometimes reason for this is offered by arguing that, if the network is initially given small weights so that initially only the "linear region" of the activation functions are used, then training will start in a relatively stable region. In this section I shall see if any improvement can be made by starting the networks from different sets of initial weights. Five sets of randomly generated weights were used, labelled 0 to 4. In all cases the weights were generated by a random number generator that produced random numbers uniformly in $[-r : r]$, r being the range of the generator. All previous results used

Table 4.24: The results of training networks from Table 4.16, on the NMA5 time series, with different values of learning rate ϵ. None of the network's here has improved upon the previous best error of 0.055.

	Learning Rate ϵ				
Network	0.0001	0.0005	0.01	0.05	0.1
8-2-1	0.058	0.058	0.058	0.060	0.063
8-4-1	0.063	0.063	0.063	0.064	0.064
8-8-1	0.070	0.069	0.067	0.067	0.068
	Number of Iterations				
8-2-1	314	111	479	111	111
8-4-1	405	111	111	111	111
8-8-1	111	20000	20000	322	429

	Learning Rate ϵ				
Network	10^{-7}	10^{-6}	$5\,10^{-6}$	10^{-5}	$5\,10^{-5}$
8-4-1	0.183	0.063	0.063	0.063	0.063
	Number of Iterations				
8-4-1	20000	20000	7892	405	111

an initial set of weights with $r = 1$. Set 0 was also created with $r = 1$, but a different seed was used with the generator, sets 1 and 2 both had $r = 0.1$, and sets 3 and 4 had $r = 0.01$.

Firstly in Table 4.25 I examine the effect of using the above initial weight sets on networks predicting the NAR1 time series. As can be seen there is little or no improvement in each case, the biggest gain being for the networks with 8 inputs. Note, however, that the use of weight sets 3 and 4, the sets with the range of $r = 0.01$, causes the training of the 4-2-1 network to become unstable by 247 iterations, resulting in a massive final error.

Table 4.26 shows similar results but now the use of weight set 2 also causes instability in the training of the 4-2-1 network. Also the 2-2-1 network has improved its error to 0.098, close the best possible error of 0.096. Remember that set 0 has the same range, $r = 1$, as was used previously for the results in Table 4.8.

In the case of networks predicting the MA5 time series, in Table 4.27, there is very little change in the end result of training. This is probably because these networks require a much lower learning rate than the one used, $\epsilon = 0.01$, as indicated by the results in Table 4.24.

Table 4.25: The results of training networks from Table 4.7, on the NAR1 time series, with different starting points in weight space. Using the best possible prediction scheme, which is just an echo, the prediction error was 0.096.

	Weight Set				
Network	0	1	2	3	4
2-2-1	0.104	0.105	0.105	0.105	0.105
4-2-1	0.104	0.104	0.563	0.562	0.562
8-8-1	0.128	0.125	0.125	0.125	0.126
	Number of Iterations				
2-2-1	13787	16829	17407	17933	17887
4-2-1	14234	15867	254	247	247
8-8-1	17191	20000	20000	20000	20000

Table 4.26: The results of training networks from Table 4.8, on the NAR2 time series, from different initial weights.

	Weight Set				
Network	0	1	2	3	4
2-2-1	0.098	0.102	0.103	0.103	0.103
4-2-1	0.103	0.103	0.371	0.370	0.370
8-8-1	0.126	0.125	0.124	0.125	0.126
	Number of Iterations				
2-2-1	458	860	1077	1998	1897
4-2-1	9110	6748	263	257	258
8-8-1	9301	15645	20000	20000	20000

Table 4.27: The results of training networks from Table 4.16, on the NMA5 time series, from different initial weights. None of the network's here have improved upon the previous best error of 0.055.

	Weight Set				
Network	0	1	2	3	4
8-2-1	0.063	0.063	0.063	0.063	0.063
8-4-1	0.061	0.063	0.063	0.063	0.063
8-8-1	0.066	0.063	0.063	0.063	0.063
	Number of Iterations				
8-2-1	111	132	111	111	111
8-4-1	111	216	111	127	122
8-8-1	963	423	314	416	419

4.9 Conclusions

In this section I draw together the results of this chapter and conclude what rules of thumb can be formed to help with the predictions of the next chapter and any future work with FFNNs. The motivation of this chapter was to see how well Neural Networks performed on artificial time series where alternative methods of prediction were available. For some of the time series it was even possible to find the *best possible prediction scheme.* For other time series trivial prediction schemes were constructed, so that the errors of these schemes had to be improved upon for the networks to prove their worth. From the discussions throughout this chapter I have constructed the following list of "rules", but remember that in working with neural networks every rule seems to have plenty of exceptions.

1. *Networks were almost always able to provide meaningful predictions.* In most cases the networks were able to better trivial prediction schemes such as echoing or zero-prediction. In the cases where this was not so, the reasons could usually be ascribed to the nature of the time series. For example, the best prediction in the NAR1 case was actually an echo. In some cases the networks showed that they were on par with the best possible prediction schemes, e.g. for the NAR4 and NAR5 time series.

2. *Larger networks seem to be more difficult to train.* In many of the examples above it was found that using networks with 8 inputs or 8 hidden neurons resulted in a poorer error at the end of training. The exceptions mainly being in the prediction of the highest order time series, e.g. the NAM time series, where the use of only 2 inputs or 2 hidden neurons could not adequately predict the time series.

3. *Almost all the network predictions show some delay.* This is discussed in more detail in Section 4.10.

4. *There is a best range of values for the learning rate.* As seen by the results in Tables 4.22 and 4.23 the best values of learning rate were round about 0.01. A higher learning rate would result in a higher error because of instability during training, whilst a lower learning rate would require many more training iterations to achieve the same end result.

5. *The best learning rate for one time series is not necessarily best for another.* Table 4.24 shows that the best learning rates for the MA5 time series is about 0.001 or less.

6. *Starting networks from different points in weight-space do not produce markedly different results.* The traditional notion that networks should be started with weights that are small and scattered around 0 appears to be correct. Also using different starting points for the weights does not seem to affect the end result of training too significantly, as seen in Tables 4.25,4.26 and 4.27. However, it appears that if the initial weights are too small then instability in training is more likely. In fact, as can be seen from the back-propagation equations in Section 1.4.1, if the initial weights are all zero then a network with linear output activation functions cannot progress in training because all the initial weight changes are also zero.

4.10 Concluding Remarks - Delay and Echo

Almost all the networks studied in this chapter produced some kind of delayed predictions. That is, the minimum in the time shift plot lay not at 0 but at some negative value. What does this mean? In one sense a delay of 1 time-step can be re-phrased as a statement such as "the 1 step ahead predictions bear more resemblance to the most recent input of the network than to the values they were required to predict". But this is not the same as echoing, echoing means that the predictions are actually the same as the most recent input of the network. So is delay in the predictions necessarily a bad thing, in other words as long as the actual error of prediction is less than the error incurred by echoing, does it matter that the predictions bear a greater resemblance to the network's most recent input? The answer would have to yes, it does matter, for the following simple reason. The main impetus behind predicting time series in this thesis is to provide forewarning of events in the solar-terrestrial environment that might cause adverse affect to human exploits in near-space or on the Earth, for example in managing the orbital decay of satellites. Looking at any of the delayed predictions for the artificial time series in this chapter it is clear that many of the large peaks are only evident in the predictions *after* "the event" has already begun, such is the nature of delay. Such failure to predict these large events is therefore a major problem. The question is now, whether this failure to predict these events is the fault of the network, or is because the nature of the time series does not allow a better prediction to be made. To answer this question, I turn to the NAR time series where the best prediction method is readily available. In general a NAR time series is of the form

$$X_t = f\left(X_{t-I}, \ldots, X_{t-1}\right) + a_t$$

where the best possible prediction scheme is just

$$\hat{X}_t = f\left(X_{t-I}, \ldots, X_{t-1}\right)$$

with an expected RMS error of σ. To see what the time shift function looks like, I assume that X_t is a stationary time series, and for brevity I shall write it as

$$X_t = f_t + a_t$$

where $f_t = f\left(X_{t-I}, \ldots, X_{t-1}\right)$, it is then clear that f_t must also be a stationary time series. The expectation of the (squared) time shift function, after a little manipulation can be written in the following form

$$\begin{aligned} E\left[T_\tau^2\right] &= E\left[(\hat{X}_{t-\tau} - X_t)^2\right] \\ &= \sigma^2 + 2\sigma_f^2(1 - \rho_{f\tau}) - 2E[a_t f_{t-\tau}] \end{aligned}$$

where $\sigma_f^2 = Var[f_t]$ and $\rho_{f\tau}$ is the auto-correlation function of f_t, also $E\left[T_0^2\right] = \sigma^2$. Also note that the second term involving the auto-correlation cannot be negative. The last term in the expression, i.e. $2E[a_t f_{t-\tau}]$ can give rise to the asymmetry, and therefore delay, in the time shift plot. More specifically if $\tau > 0$ then it is easy to show that $E[a_t f_{t-\tau}] = 0$, because there can be no a_t terms in $f_{t-\tau}$. So this means that an apparent delay is present if $E[a_t f_{t-\tau}] \neq 0$ for $\tau < 0$. To illustrate the above, consider the AR time series

$$X_t = \alpha X_{t-1} + a_t$$

(which is stationary if and only if $|\alpha| < 1$). In this case for $\tau = 1$

$$E[a_t f_{t-1}] = \alpha\sigma^2$$

so that $T_0^2 - T_1^2 = -\alpha\sigma^2$. It is therefore clear that a delay will be apparent in even the best possible prediction scheme.

In a similar fashion it is possible to derive the expected shape of a time shift plot for a linear MA time series of the form

$$X_t = \sum_{n=1}^{N} \theta_n a_{t-n} + a_t$$

This time the best predictor function is not obvious, so I shall assume that there is some method of estimating the a_ts given the history of the X_t time series. Let these estimates be $\hat{a}_t$ and let them differ from the true values by some other zero-mean (the estimates are unbiased) stationary white noise time series b_t, with variance σ_b^2, such that

$$\hat{a}_t = a_t + b_t$$

and let b_t be uncorrelated with a_t. It is then possible to show that the time shift function has the following form:

$$\begin{aligned}
E[T^2_{\tau>0}] &= \sigma^2 + \sigma_b^2 \sum_{n=1}^{N} \theta_n^2 + 2\sigma^2 \left[\sum_{n=1}^{N} \theta_n^2 - \sum_{n=1-\tau}^{N} \theta_n \theta_{n-\tau} \right] \\
E[T_0^2] &= \sigma^2 + \sigma_b^2 \\
E[T^2_{\tau<0}] &= \sigma^2 + \sigma_b^2 \sum_{n=1}^{N} \theta_n^2 + 2\sigma^2 \left[\sum_{n=1}^{N} \theta_n^2 - \sum_{n=1}^{N+\tau} \theta_n \theta_{n-\tau} \right] - 2\sigma^2 \theta_{-\tau}
\end{aligned}$$

Again the positive and negative side of the time shift curve are symmetrical except for one term, in this case $-2\sigma^2\theta_{-\tau}$. So again it is possible to see an apparent delay intrinsic to the best available prediction scheme.

So it seems that for NAR and linear MA type time series at least, a delay in the predictions is not necessarily a failure of the method at all. However, delay is still a problem in practice for the reasons outlined above, so a different question now begs an answer. Can one tell if a delay is symptomatic of the time series, and therefore incurable, or if it is a fault of the method? The answer to this question is actually more subtle than might first be thought, because it seems natural to assume that their can be no hope for improvement if the best prediction scheme intrinsically suffers from delay. If all one is interested in is the error then this is indeed true. However, if in practice it is of more importance to predict an event on time rather than predicting the future with lowest RMS type accuracy, then the problem of delay can be solved by redefining what is *meant by best*. In other words, new criteria of success can be crafted that rates a set of predictions as *best* if they not only achieve a low RMS error, but they also achieve "on-time" prediction of future events. The reason that all the prediction methods from before suffered from delay is that they were not required to do anything but minimize RMS error. In constructing new methods of producing non-delayed predictions it must be accepted that, in some sense, RMS accuracy is being sacrificed for on-time prediction.

In the next chapter I turn to the prediction of natural solar-terrestrial time series, using the same approach as was used in this chapter. In the chapter after next, I return to the issues on delay that have just been raised and show how a genetic algorithm training scheme can embrace the more complicated criteria of success needed to produce networks that predict the future "on time", without delay.

Chapter 5

Prediction of Solar Terrestrial Time Series

"Londo: I feel like I am being neebled to death by..., em, Veer, what are those creatures with webbed feet the Earthers talk about that go 'Quack Quack'?
Vir: Cats?
Londo: Ah, yes, that's it. I feel like I am being neebled to death by cats."

The drive behind this chapter is somewhat different from the last. The study of artificial time series allowed for some comparison between theory and practice, whereas the work of this chapter is of a much more practical nature. The basic aim sounds simple enough: to obtain the best possible predictions of Sunspot Number, 10.7cm Solar Flux and Geomagnetic K_p index using feed forward neural networks. The first complication is that of the definition of what is "best", which was discussed at the end of the last chapter. For the moment I will still use an RMS error, as this is the error that lies at the heart of standard back-propagation. I shall also comment on any delay in the predictions and in Chapter 6 I will demonstrate a method that can find networks which predict without delay. Secondly, given some criteria of success, how can one tell whether the best possible predictions have been achieved - perhaps there is a better configuration of network and/or training algorithm lurking elsewhere in parameter space, perhaps even just around the corner from the present best set of parameters? In general there is no assurance, but the results of the last chapter suggest that, if a reasonable set of parameters have been found (e.g. I,H,O,ϵ,α and initial weights), then neighbouring sets of parameters do not usually produce significantly different results[1].

One final point before proceeding further. A common criticism (mainly from physicists) of using neural networks to predict the solar-terrestrial weather is that, even if successful, it provides little or no understanding of what is actually going on. As seen in Chapter 3 it is possible to construct a FFNN to fit almost any analytic function in a given range, but it is not generally possible to take a FFNN's weights and say what (analytic) function it might represent. For example, there is no easy way to look at the weights of a network and deduce that it was trained to predict a particular non-linear AR time series. Note also that in the results of the last chapter, the number of inputs of the best network did not necessarily correspond to the order of NAR model it was trained to predict. So, training many networks to predict sunspot number and looking at the configuration and weights of the best network is unlikely to tell you very much about the time series, and certainly little or nothing about the physics behind the time series. However, I believe that such criticism, though perhaps founded in fact, is mis-directed. The aim of the work in this thesis is not to understand the solar-terrestrial environment in terms of physics, it is to find structure and exploit it in order to make predictions of the future. Although no direct physical understanding is sought here, the discovery, documentation and exploitation of structure in natural systems is usually an aid, and in many cases a precursor, to physical understanding.

5.1 The Time Series

Chapters 1 and 2 described, reviewed and explored the three time series of interest in this thesis: the sunspot number (SSN), the 10.7cm solar flux and K_p geomagnetic index. In this chapter I shall attempt to predict these time series in five basic formats: daily, smoothed daily, monthly, smoothed monthly and yearly, where I have defined the smoothed daily series $\bar{s}_t$ as being the 7 day running mean of the original series s_t, i.e.

$$\bar{s}_t = \frac{1}{7}\sum_{n=-3}^{3} s_{t+n}$$

This means that there are effectively 15 time series that are to be predicted.

[1] The reader might be aware that some of the results in the third chapter contradict this last statement, as training was seen to become unstable after some number of iterations with seemingly random consequences. However, as has been stated before, the use of a generalisation stopping criterion removes these difficulties because training is almost always stopped before these instabilities occur.

5.2 The Approach

In this chapter I will begin with a search of neural network architectures, so as to find the best network for predicting each of the 15 time series 1 time step in advance. I have restricted my attention to networks with 2,6,12 and 18 inputs and 2,6,12 and 18 hidden neurons. This is partly to prevent the required computer time required from becoming prohibitively large but also because, in preliminary tests, I found no advantage in using larger networks.

After this, the best networks for each time series will be subjected to further training, using a smaller learning rate if the original training run was halted early. These best networks will then be used to provide iterated predictions and some comparison will be made with networks that have been trained to predict 6 steps into the future. There will then be some discussion of how to assess the potential accuracy of a trained network's prediction, resulting in the provision of 80% uncertainty estimates for all networks trained in this chapter. Finally some thought is given on how the choice of training set might affect the end results.

As before, training is halted

1. if after the first 100 iterations the error has increased for ten iterations,
2. or if the number of iterations has reached a given maximum number.

The maximum number of iterations depends on the time series that the network is being trained on, because there is more daily data than monthly data, and more monthly data than yearly data. For the yearly time series the maximum limit is 20,000, for the monthly series 10,000 and for the daily time series 5,000. All training runs will use standard back-propagation with a learning rate of 0.01 and a momentum of 0.9, with sequential presentation of patterns from the training set and application of weight changes immediately after the presentation of a pattern (i.e. *not* batch mode training). I shall not experiment with different initial sets of weight parameters, for reasons of time, and because the results at the end of the last chapter suggest that the gains were not considerable, as long as the initial weights were not too small.

The following results are organised into three sections, one for each time series, with a page devoted to each of the formats. The layout of the results is much the same as the last chapter. The echo error on the network's test set is provided in each case as a bench-mark.

5.3 Sunspot Number

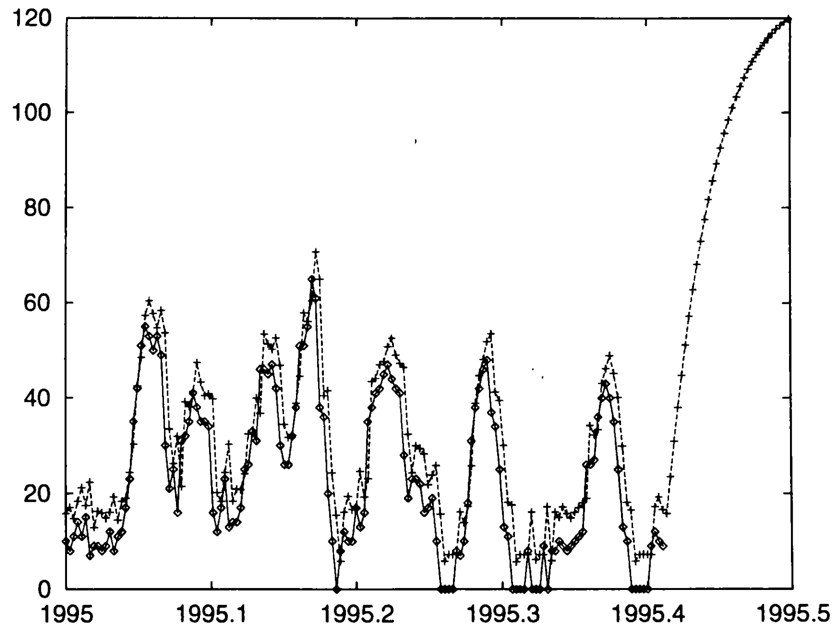
The following pages show the results of training different network architectures to predict the five formats of sunspot number. Each format is given its own page composed of: a statement of the training and test sets used; a commentary on the results; a table of RMS test set errors for each architecture; a prediction plot made using the best network; and its corresponding time shift plot.

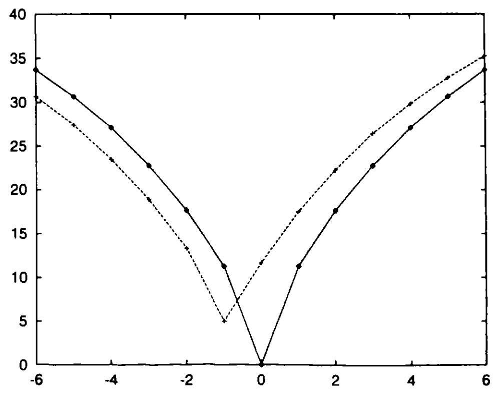
I only use the sunspot numbers after 1850 for the following work because the statistical properties of Sunspot Number seemed to change significantly at this time (see Section 2.3.1). Whether this change is because of the non-stationary nature of the series or is a side-effect of the reconstruction by Wolf, is immaterial. Either way the behaviour of the time series before 1850, seems to be different from what is seen after 1850, so its inclusion as training or test set data is inappropriate.

Daily
Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May 1995
In predicting the daily series, networks with fewer hidden neurons seem to fare best, with the lowest error of 10.7 achieved by both the 2-2-1 network and the 18-2-1 network. (The issue of what functions an I-2-1 network can be expected to represent is discussed later in Section 5.10.) It is apparent from the prediction plot that all the maxima of the 27 day cycles are predicted late. The time shift plot confirms that delay is present and that the predictions are slightly worse than an echo, which has an error of 10.3.

Table 5.1: The results of predicting the daily sunspot number time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	10.7	11.2	10.9	10.7
6	10.8	12.1	12.9	13.3
12	11.1	12.9	14.1	13.6
18	11.3	13.7	14.9	16.8
	Number of Iterations			
2	349	5000	1803	1889
6	288	2733	5000	5000
12	364	5000	5000	5000
18	272	5000	5000	5000

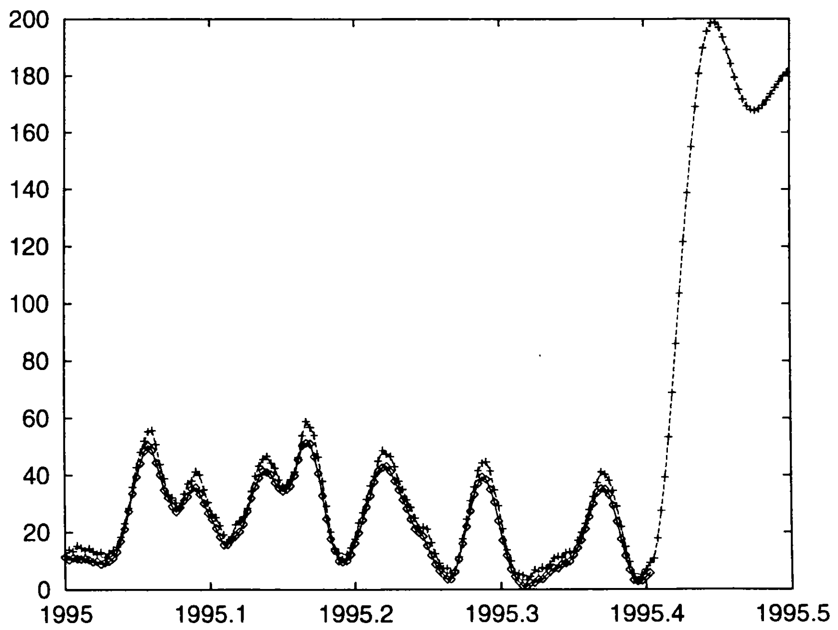


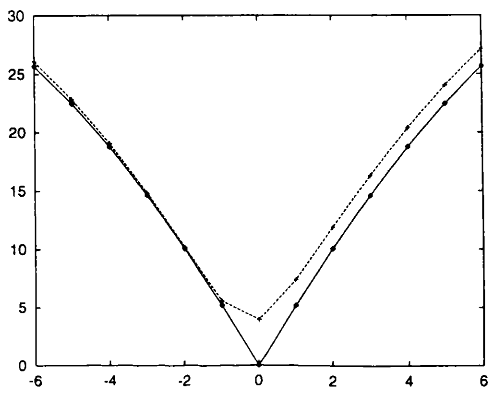


Smoothed Daily
Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May 1995
For the smoothed daily time series, the smallest networks achieve the lowest errors, with the best networks, the 2-2-1 and 6-2-1 networks, achieving an RMS error of 3.9. It is apparent that all the predictions have a positive bias. This is undoubtedly because the training set values (covering the maximum of cycle 22) were mostly larger than the 15 to 50 range of sunspot number shown in the prediction plot. The predictions do not seem to suffer from a delay and the prediction error is significantly better than an echo, which has an error of 5.2. Finally, notice the ludicrous results of the iterated predictions.

Table 5.2: The results of predicting the smoothed daily sunspot number time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	3.9	3.9	4.9	5.4
6	5.1	6.6	8.1	8.8
12	6.7	8.2	11.4	12.0
18	7.1	10.4	13.9	15.0
	Number of Iterations			
2	1343	4085	5000	5000
6	926	5000	5000	5000
12	1041	5000	5000	5000
18	745	5000	5000	5000

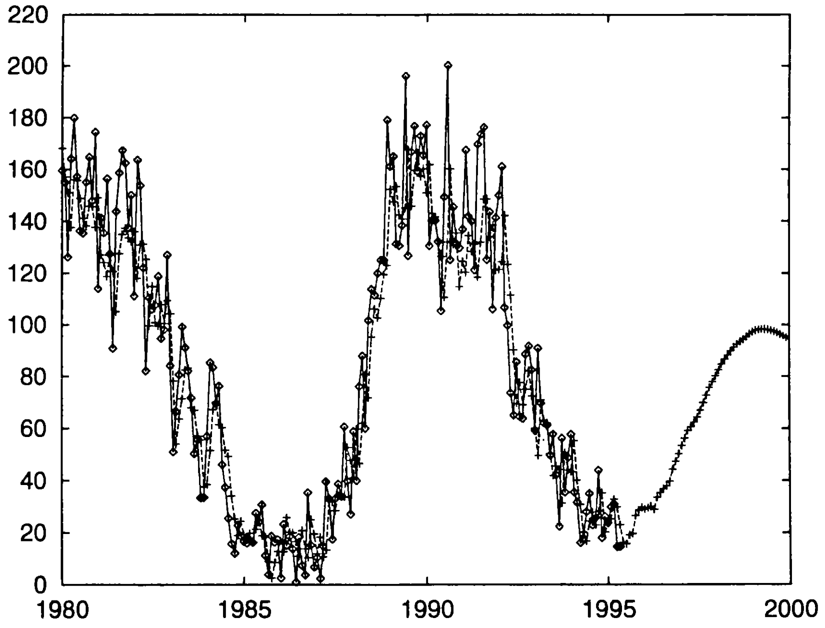


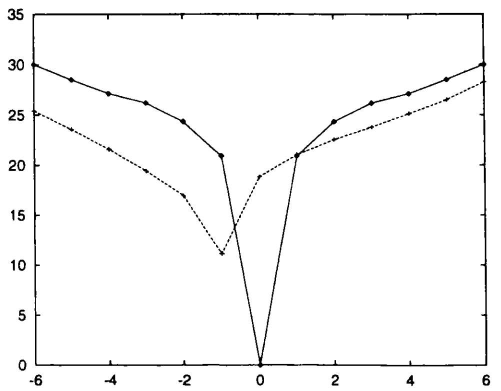


Monthly
Training set: Jan. 1900 - Dec. 1966
Test set: Jan. 1967 - May 1995
The monthly sunspot numbers appear to be predicted best by the 18 input networks with 2 or 6 hidden neurons. While the best achieved error, 18.6, is significantly lower than the echo error of 20.9, the predictions are heavily delayed. This is apparent from the prediction plot, where many peaks are predicted late, and also from the minimum of the time shift plot.

Table 5.3: The results of predicting the monthly sunspot number time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	19.3	19.0	18.8	18.6
6	19.8	20.0	19.7	18.6
12	21.6	21.4	20.4	19.1
18	25.9	24.0	20.8	19.0
	Number of Iterations			
2	704	1563	1426	10000
6	436	10000	1077	1242
12	10000	10000	10000	10000
18	10000	10000	10000	10000

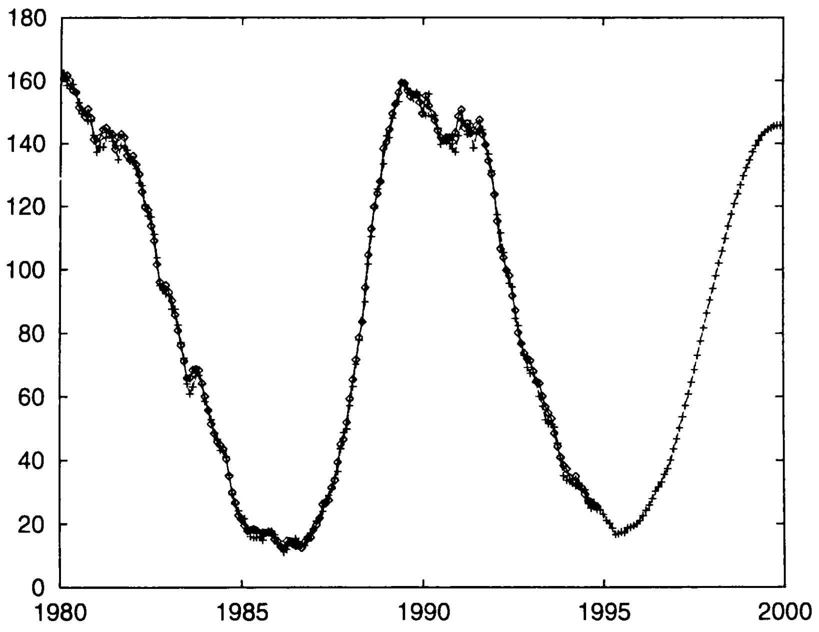


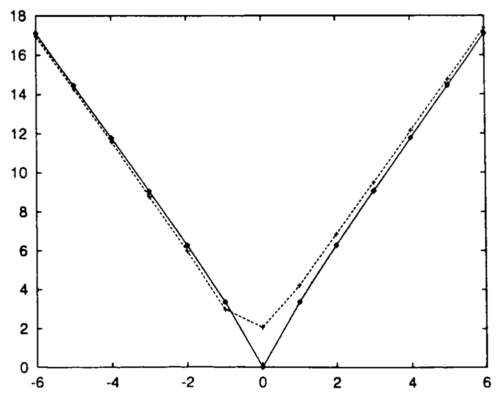


Smoothed Monthly
Training set: Jul. 1900 - Dec. 1966
Test set: Jan. 1967 - Nov. 1994
All the networks in Table 5.4 achieve comparable errors, with the largest network having the best error of 2.0. The time shift plot shows that little delay is present and that the echo error of 3.4 has been significantly improved upon. The iterated predictions predict that the next solar maximum will occur in the middle of 1999, with a max value of about 145.

Table 5.4: The results of predicting the smoothed monthly sunspot number time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	2.3	2.4	2.4	2.4
6	2.3	2.3	2.3	2.3
12	2.5	2.3	2.4	2.2
18	2.9	2.5	2.3	2.0
	Number of Iterations			
2	10000	10000	10000	10000
6	10000	10000	10000	10000
12	10000	10000	10000	10000
18	10000	10000	10000	10000

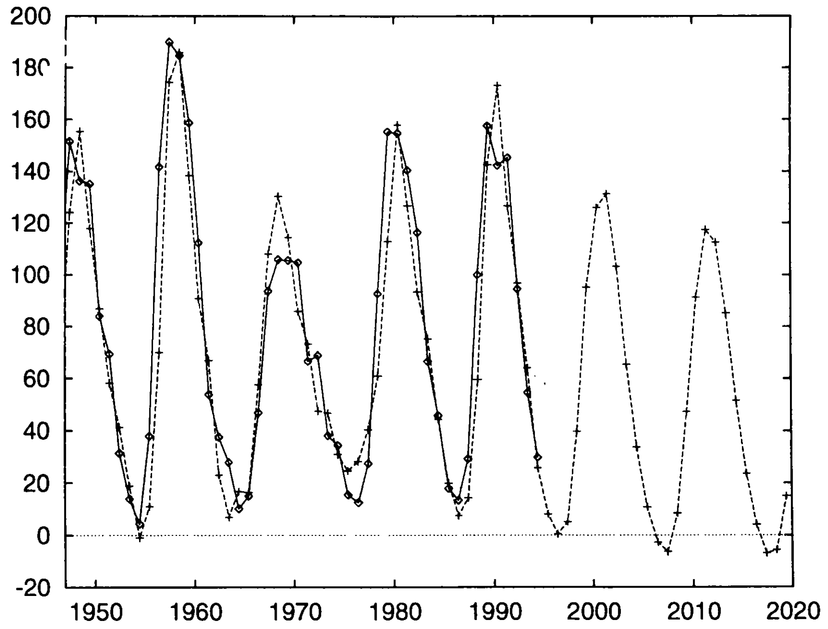


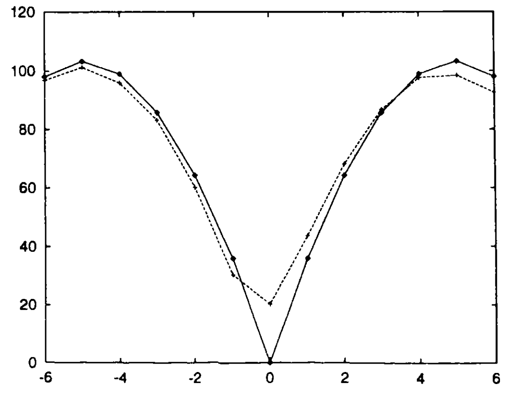


Yearly
Training set: 1850 - 1951
Test set: 1952 - 1994
The smallest networks in Table 5.5 appear to have the highest errors in predicting the yearly sunspot number. The 12 input networks fare best, with the 12-2-1 network achieving the best error of 20.2. The time shift plot indicates that there is little delay and that this best error is significantly less than the echo error of 35.9. The iterated predictions show that the network has successfully captured the 11 year cycle, but rather unphysically, negative sunspot numbers are predicted in the iterated cycles.

Table 5.5: The results of predicting the yearly Sunspot Number time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	24.0	25.8	20.2	22.0
6	23.4	43.2	21.3	21.9
12	22.5	23.5	21.4	22.3
18	21.6	22.6	22.2	22.6
	Number of Iterations			
2	12128	7406	6507	19008
6	20000	179	2906	8555
12	20000	20000	17368	6133
18	20000	19937	2481	6634





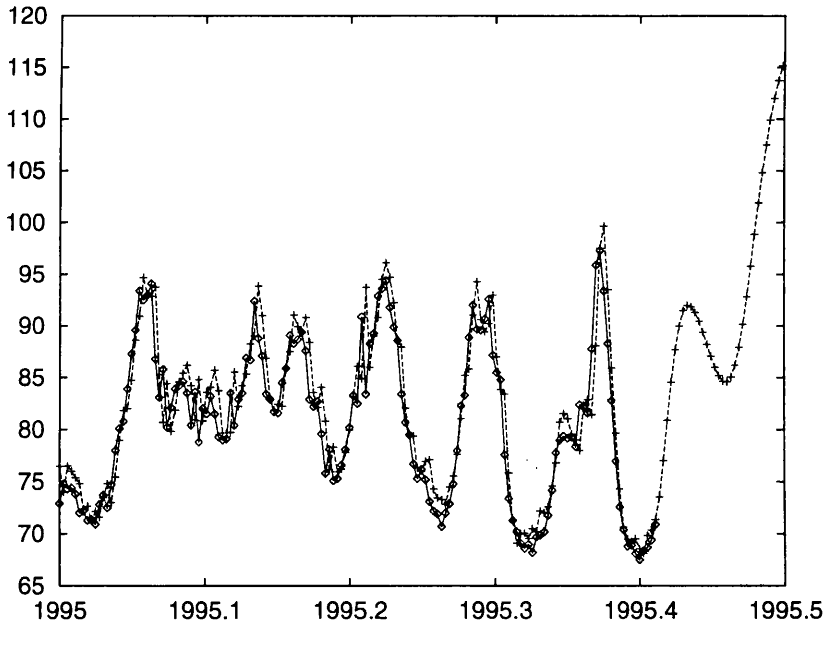
5.4 Solar Flux

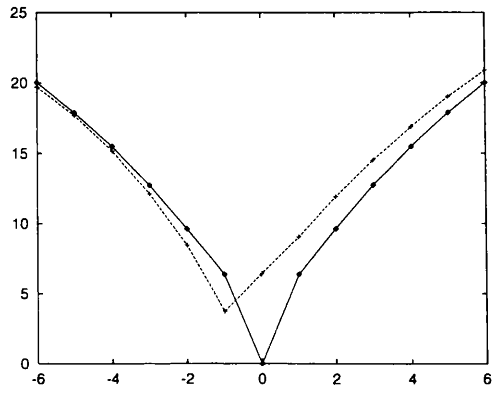
The following pages show the results of training different network architectures to predict the five formats of solar flux. Each format is given its own page composed of: a statement of the training and test sets used; a commentary on the results; a table of RMS test set errors for each architecture; a prediction plot made using the best network; and its corresponding time shift plot.

Daily
Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May 1995
The prediction of the daily time series appears to have achieved a very small error indeed, with the smaller networks faring slightly better than the larger ones. Having said this, the predictions show a strong delay and the best network prediction error is slightly larger that the echo error of 5.9.

Table 5.6: The results of predicting the daily solar flux time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	6.2	10.9	6.3	6.0
6	6.8	7.6	8.4	7.6
12	8.0	9.3	10.1	8.3
18	8.7	10.3	11.7	10.8
	Number of Iterations			
2	871	158	5000	5000
6	858	5000	5000	5000
12	5000	5000	5000	5000
18	1361	5000	5000	5000

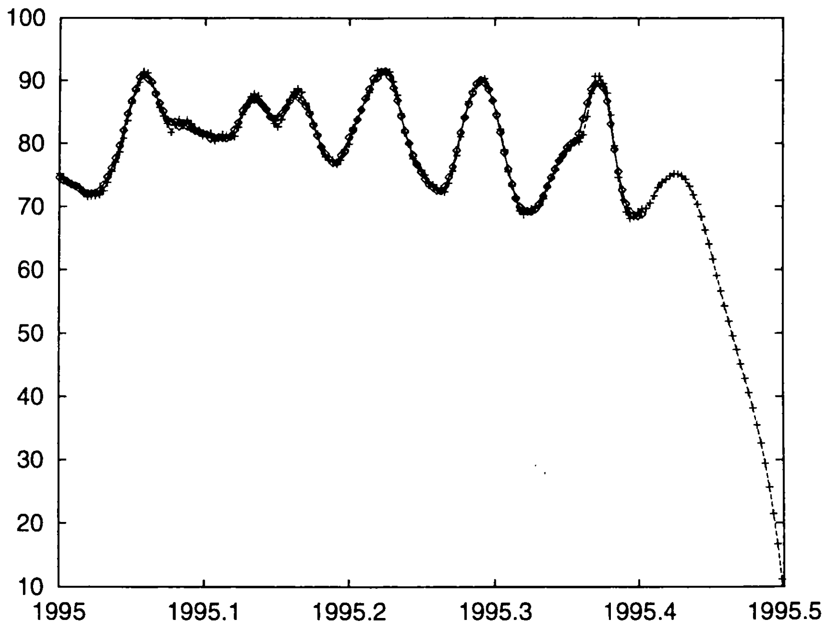


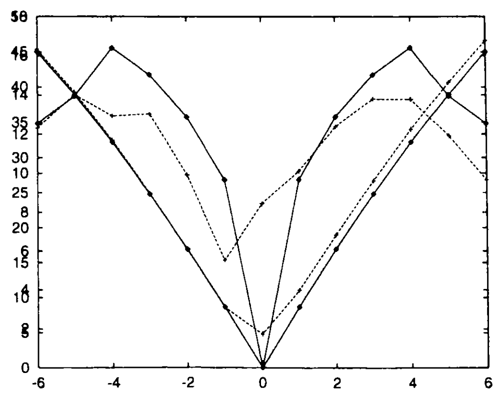


Smoothed Daily
Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May 1995
The predictions of the smoothed daily time series are undelayed, with a very low prediction error, significantly smaller than the echo error of 3.1. Notice how the network predictions appear to iterate through one (approximately) 27 day cycle before becoming unstable.

Table 5.7: The results of predicting the smoothed daily solar flux time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	6	12	18
	RMS error on Test Set			
2	1.7	7.1	1.6	1.5
6	1.5	1.7	2.4	1.7
12	2.0	1.6	3.4	1.9
18	1.5	2.8	3.2	3.0
	Number of Iterations			
2	5000	159	5000	5000
6	2473	5000	5000	5000
12	3248	5000	5000	5000
18	2211	5000	2379	5000

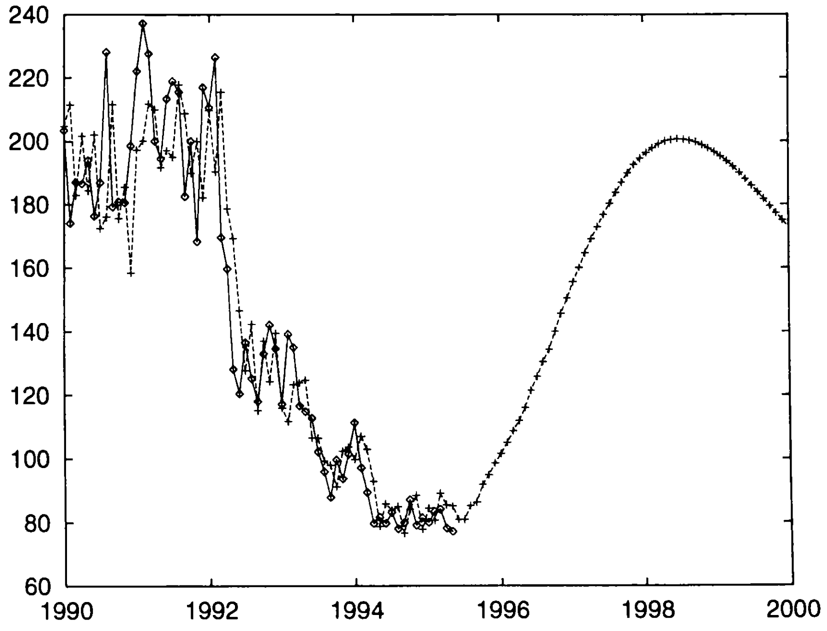


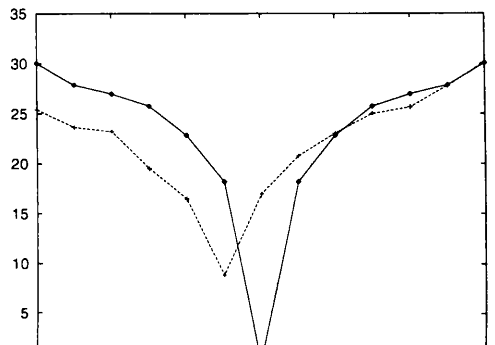


Monthly
Training set: Jan. 1947 - Jun. 1980
Test set: Jul. 1980 - May 1995
It appears from Table 5.8 that the best networks for predicting the monthly solar flux are ones with only 2 hidden neurons. Again the network seems to have learned enough about the solar cycle to iterate ahead to a future maximum. The maximum is predicted to occur somewhat earlier than might be expected, being almost three years before the maximum predicted with the yearly time series. On the other hand, at the time of writing, there is suggestion that the sunspot cycle has already passed its minimum. If this is indeed the case, then cycle 22 was only 9 years long, and remembering that the rise time of recent maxima was typically between 3 and 5 years, it seems likely that the maximum of cycle 23 would occur before the turn of the century. Although the monthly predictions are delayed, they are still slightly better than an echo, which has an error of 18.1.

Table 5.8: The results of predicting the monthly solar flux time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	6	12	18
	RMS error on Test Set			
2	17.4	17.4	17.2	17.0
6	17.8	19.1	18.0	17.0
12	19.9	21.0	20.3	17.2
18	25.7	25.3	20.8	20.9
	Number of Iterations			
2	1970	4276	2631	3840
6	1271	10000	2023	3342
12	10000	10000	1389	2578
18	10000	10000	10000	111

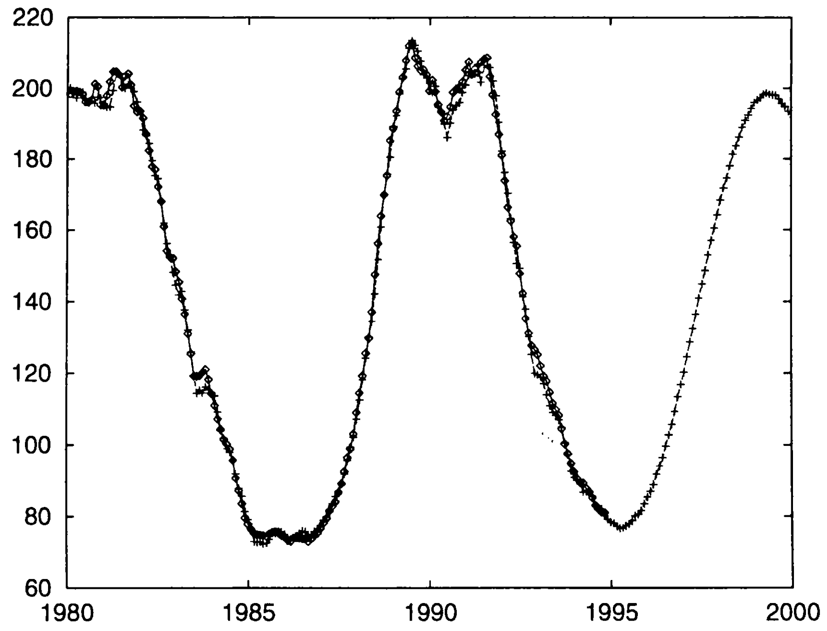


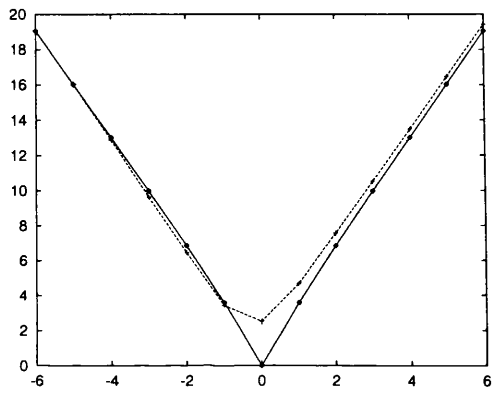


Smoothed Monthly
Training set: Jul. 1947 - Jun. 1980
Test set: Jul. 1980 - Nov. 1995
The smoothed monthly series seems to be best predicted by the networks with more than 2 inputs. The next maximum is predicted to occur before the turn of the century in 1999, about a year after the maximum predicted by the monthly time series. It appears that the predictions are undelayed, with an error significantly better than that the echo error of 3.6.

Table 5.9: The results of predicting the smoothed monthly solar flux time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	3.9	3.2	2.8	34.6
6	3.8	2.7	2.7	2.8
12	4.1	2.6	2.7	3.1
18	5.0	2.6	8.0	2.5
	Number of Iterations			
2	10000	10000	10000	282
6	10000	10000	10000	10000
12	10000	10000	10000	10000
18	10000	10000	150	10000

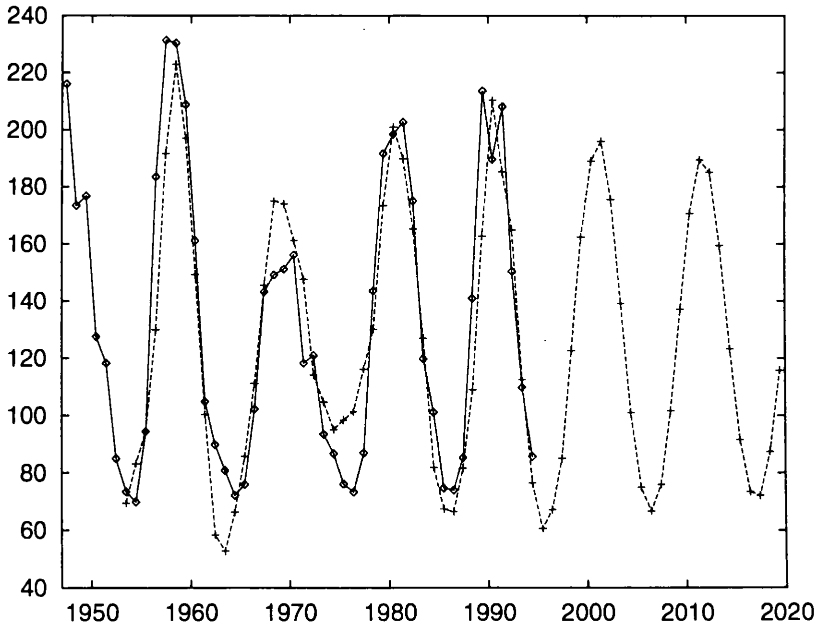


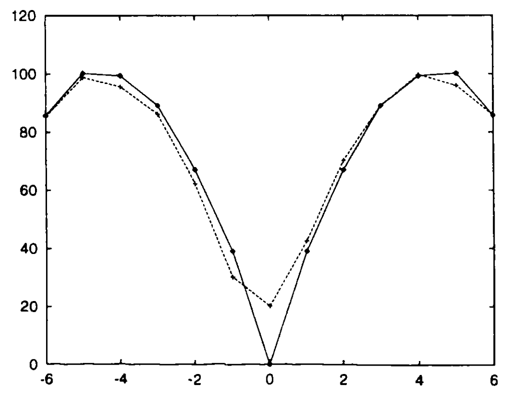


Yearly
Training set: 1947 - 1980
Test set: 1981 - 1994
Prediction of the yearly solar flux time series is difficult because there are only 47 complete years of data available at present. Nonetheless, the network performances are quite reasonable, excepting the 18-2-1 network and the networks with only 2 inputs. Notice how the best network has learned enough about the solar cycle to iterate ahead and predict two further maxima in 2001 and 2011. Both the prediction and time shift plots indicate that there is little delay present in the test set predictions, and that the predictions are significantly better than an echo, which has an error of 39.1.

Table 5.10: The results of predicting the yearly solar flux time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	27.8	20.1	22.3	42.0
6	27.8	20.9	21.6	22.1
12	27.8	22.9	23.6	24.1
18	27.7	24.1	21.0	21.2
	Number of Iterations			
2	3501	8555	20000	884
6	2375	7486	20000	7197
12	1808	2867	119	20000
18	1353	8030	20000	14268





5.5 Geomagnetic K_p Index

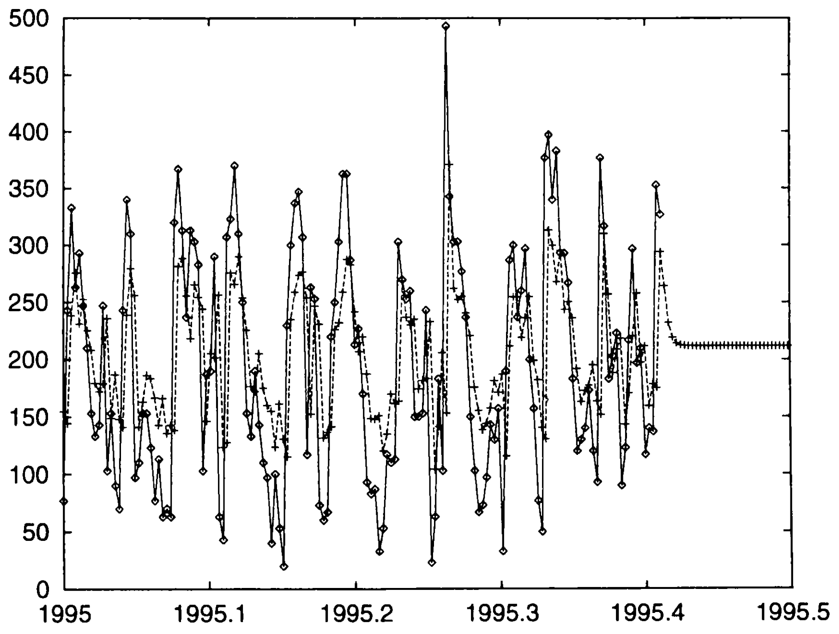
The following pages show the results of training different network architectures to predict the five formats of geomagnetic K_p index. Each format is given its own page composed of: a statement of the training and test sets used; a commentary on the results; a table of RMS test set errors for each architecture; a prediction plot made using the best network; and its corresponding time shift plot.

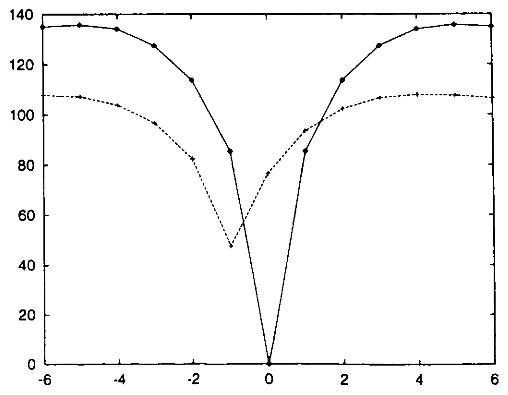
Daily

Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May. 1995
All the networks tested here bettered the echo error of 85.4. Although the 2-6-1 network achieved the lowest error with 76.9, all the networks have really performed equally well. The prediction plot clearly shows a delay in the predictions, also apparent in the time shift plot. The network's outputs did not learn to span the range of the time series very well at all, which could be an indication that the daily K_p index is quite unpredictable, at least by these methods.

Table 5.11: The results of predicting the daily K_p index time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	77.5	77.7	77.9	77.8
6	76.9	77.0	77.6	77.6
12	77.1	77.1	77.3	77.7
18	78.1	77.2	77.3	77.8
	Number of Iterations			
2	286	3172	602	5000
6	209	1802	1795	4798
12	5000	5000	5000	3362
18	5000	5000	5000	4593

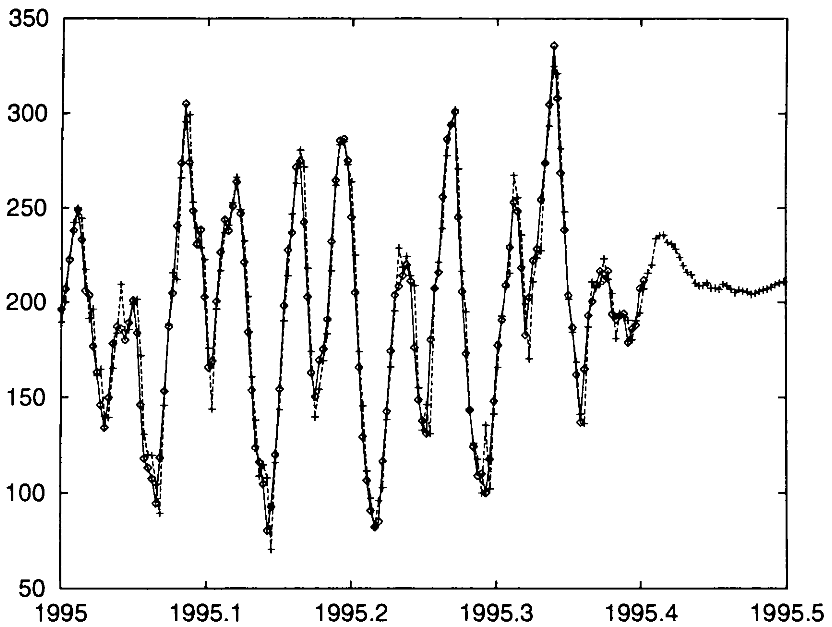


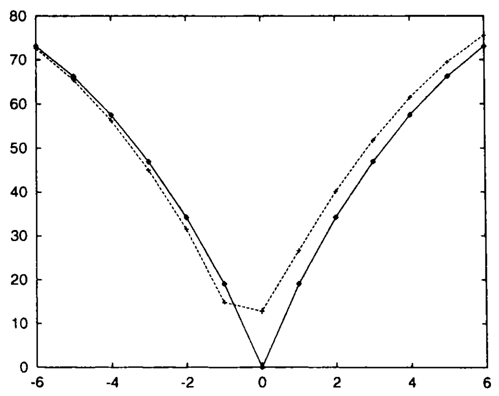


Smoothed Daily
Training set: Jan. 1985 - Dec. 1991
Test set: Jan. 1992 - May. 1995
The smoothed daily predictions appear to be very accurate indeed, with little delay. In addition the echo error of 19.1 is significantly larger than any of the network's errors. It is clear that networks with 12 or 18 inputs and 2 or 6 hidden neurons fare best.

Table 5.12: The results of predicting the smoothed daily K_p index time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	14.6	14.7	13.0	12.8
6	14.5	15.0	13.4	13.4
12	14.8	15.3	13.8	13.9
18	15.7	16.0	14.5	14.0
	Number of Iterations			
2	5000	5000	3713	5000
6	5000	5000	5000	5000
12	5000	5000	5000	5000
18	5000	5000	5000	5000

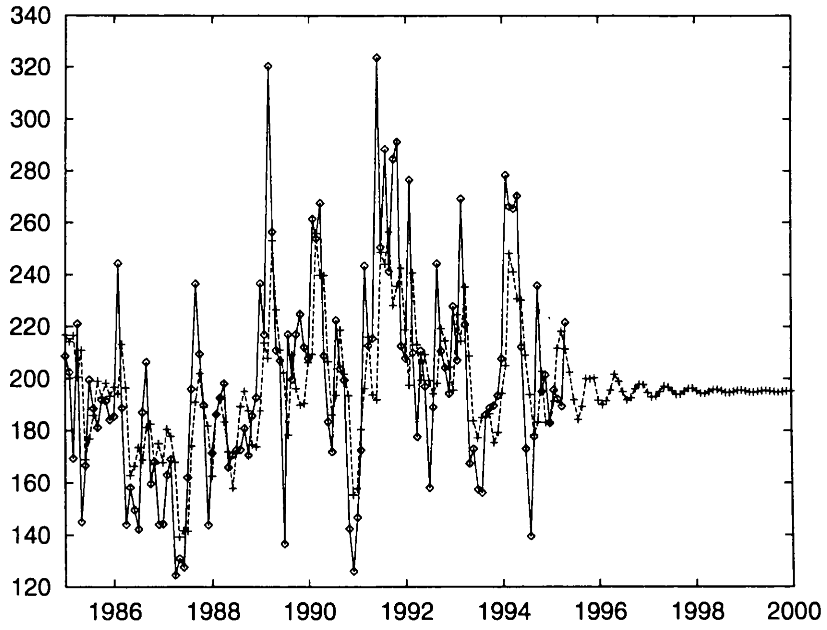


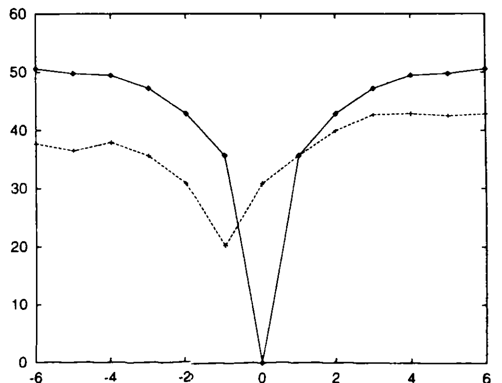


Monthly
Training set: Jan. 1933 - Jun. 1977
Test set: Jul. 1977 - May. 1995
All the networks' predictions of the monthly series improve on the echo error of 35.6. The best predictions are obtained with the 12 input networks, though all the networks have achieved comparable errors. Again, it is clear from the predictions and the time shift plot that the predictions are heavily delayed, with many peaks and troughs predicted a month late. The network's outputs have not learned to accommodate the range of the series well, with under/over-estimation occurring at the highest/lowest points. Notice that the strange oscillations in the iterated predictions have a periodicity of about six months, indicating that the network has indeed recognised the six month periodicity of the geomagnetic data, previously discussed in Section 2.2.

Table 5.13: The results of predicting the monthly K_p index time series with different architectures of FFNN.

Hidden	*No. of Inputs* 2	6	12	18
	RMS error on Test Set			
2	31.6	31.3	30.4	30.7
6	31.3	31.2	30.5	31.0
12	31.7	31.4	30.4	30.6
18	32.3	31.9	30.6	30.9
	Number of Iterations			
2	877	2534	3990	7373
6	465	10000	10000	4145
12	442	10000	10000	10000
18	10000	10000	10000	10000

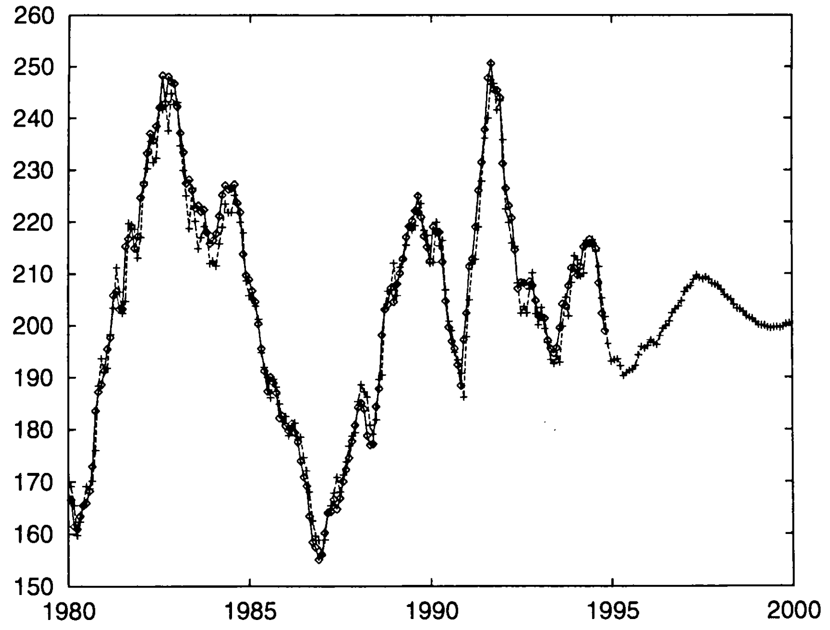


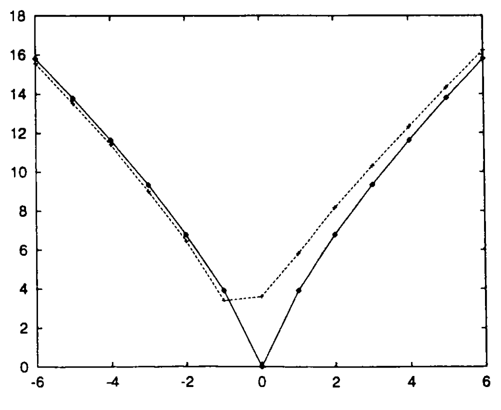


Smoothed Monthly
Training set: Jul. 1933 - Jun. 1977
Test set: Jul. 1977 - Nov. 1995
The smoothed monthly predictions appear to be very accurate, with all the networks, except for the 18-2-1 network, predicting with an RMS error of between 3.5 to 4.1. Close inspection of the prediction plot reveals that there is a slight delay. The time-shift plot confirms this and shows that the best network's error of 3.5 is only a slight improvement on the echo error of 3.9.

Table 5.14: The results of predicting the smoothed monthly K_p index time series with different architectures of FFNN.

	No. of Inputs			
Hidden	2	6	12	18
	RMS error on Test Set			
2	3.8	3.8	3.7	16.5
6	3.6	3.6	3.6	3.9
12	3.6	3.5	3.9	3.9
18	4.1	3.7	3.9	3.5
	Number of Iterations			
2	10000	10000	10000	865
6	10000	10000	10000	10000
12	10000	10000	10000	10000
18	10000	10000	10000	10000

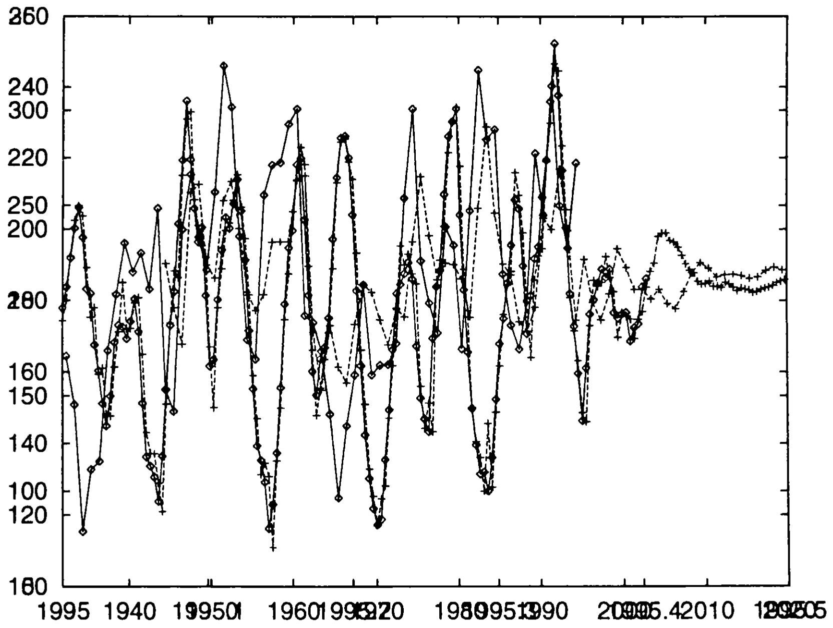




Yearly
Training set: 1933 - 1976
Test set: 1977 - 1994
The yearly predictions of K_p index are clearly rather poor. This is possibly because the training set is so small, giving the network relatively few examples from which to learn. Also, the K_p time series does not exhibit the solar cycle very strongly, making the task of prediction more difficult than was the case for either the sunspot number or solar flux. The predictions are also heavily delayed, but not to the point where an echo would be superior, the error for echoing being 26.7.

Table 5.15: The results of predicting the yearly K_p index time series with different architectures of FFNN.

Hidden	*No. of Inputs*			
	2	6	12	18
	RMS error on Test Set			
2	23.2	24.3	23.8	25.3
6	23.1	24.7	22.9	25.8
12	23.3	25.3	23.6	27.4
18	23.6	27.6	24.7	35.2
	Number of Iterations			
2	5028	13696	18480	588
6	2692	8651	2536	1609
12	1528	2663	164	955
18	1282	4441	2854	1435



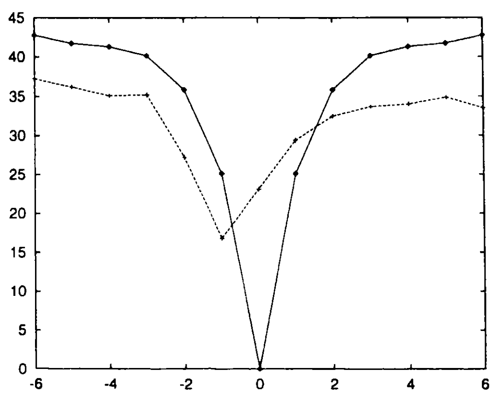


Table 5.16: These are the test set RMS errors obtained by continuing the training of the best networks from before with a smaller learning rate of 0.001. The asterisk marks networks that previously reached their maximum number of iterations, note that all these networks carried on for a maximum number of iterations in the further training.

Data	Network	Error	Last Error	Further Iter.
Daily SSN	2-2-1	9.8	10.7	1039
Smo. Day. SSN	2-2-1	2.4	3.9	5000
Monthly SSN	18-6-1	18.3	18.6	111
Smo. Mon. SSN	18-18-1	2.0	2.0*	10000
Yearly SSN	12-2-1	20.2	20.2	13529
Daily SF	18-2-1	5.8	6.0*	5000
Smo. Day. SF	18-2-1	1.5	1.5*	5000
Monthly SF	18-2-1	16.9	17.0	174
Smo. Mon. SF	18-18-1	2.4	2.5*	10000
Yearly SF	6-2-1	19.8	20.1	403
Daily K_p	2-6-1	77.2	76.9	209
Smo. Day. K_p	18-2-1	12.6	12.8*	5000
Monthly K_p	12-2-1	30.4	30.4	3283
Smo. Mon. K_p	18-18-1	3.2	3.5*	10000
Yearly K_p	12-6-1	22.7	22.9	111

5.6 Further Training

To see if any improvement in prediction accuracy is possible I have continued the training of the best network[2] for each series, using a smaller learning rate of 0.001. The results are shown in Table 5.16. The training proceeds exactly as before, until either the maximum number of iterations is reached, or instability or overfitting occurs. Since no stopping criteria is in force for the first 100 iterations, it is actually possible for a network's test set RMS error to be increased, as seen in a couple of cases. For the daily and smoothed daily time series the further training seems to have been worthwhile, though in the remaining cases any improvement is only modest and hardly significant. As far as delay is concerned the situation has not changed at all: if the original network showed delay, so does the further trained network; if the original network showed no delay then the same is true for the further trained network.

Some networks, marked with asterisks in Table 5.16, reached their maximum number of iterations in the original training runs. These networks again reached their maximum number of iterations when further trained with the lower learning rate. This means that at no time did training become

[2]If I were to be completely thorough I would continue the training of all the networks, however being bound by the limitations of computer time, this is not a luxury I could afford.

Table 5.17: These networks all reached their maximum number of iterations in the original training warranting further training with the original learning rate of 0.01.

Data	Network	Error	Last Error	Further Iter.
Smo. Mon. SSN	18-18-1	1.9	2.0*	10000
Daily SF	18-2-1	6.0	6.0*	2184
Smo. Day. SF	18-2-1	1.3	1.5*	5000
Smo. Mon. SF	18-18-1	2.0	2.5*	10000
Smo. Day. K_p	18-2-1	12.8	12.8*	760
Smo. Mon. K_p	18-18-1	3.5	3.5*	10000

unstable and that at no time did the test set error increase (a symptom of over-fitting). These networks can obviously be pushed further with the original learning rate of 0.01, and the results of so-doing can be found in Table 5.17. The improvement in most cases is rather modest, with the smoothed solar flux benefitting the most. It is interesting to note that all but one of these networks were learning to predict smoothed time series. This result has an interesting interpretation, especially when it is realised that the cause of the halting of training, in almost all of the results in this chapter, was for reasons of overfitting rather than instability. The smoothed series, almost by definition, are easier to predict than the unsmoothed series, and from the above results it is apparent that the training of a network predicting a smoothed series can continue for much longer without being troubled by overfitting. The conclusion, which might well have been expected, is that smoother time series are less prone to overfitting than more variable time series.

5.7 Predicting further ahead

Methods of predicting further than one step into the future are of course desirable. In the case of the smoothed series, predicting further than one step ahead is almost a necessity because future data is used in constructing the running mean. There are two methods available: training the network's output to predict further ahead; or iterating ahead by supplying the network with its own predictions as inputs.

5.7.1 Free Iteration

The process of iteration has been demonstrated and briefly remarked upon in the foregoing results. To recap, iteration involves the re-cycling of a network's predictions as subsequent inputs. Since

Table 5.18: These are the errors obtained by iterating the predictions of the best networks for each time series. The iteration is free running, in that the network is only exposed to actual time series data once, at the start. The final column gives the number of iterated points that were used to calculate the RMS error.

Data	Network	Error	No. of points
Daily SSN	2-2-1	97.4	151
Smo. Day. SSN	6-2-1	150.6	148
Monthly SSN	18-6-1	46.1	113
Smo. Mon. SSN	18-18-1	23.4	106
Yearly SSN	12-2-1	45.3	34
Daily SF	18-2-1	90.8	151
Smo. Day. SF	18-2-1	517.4	148
Monthly SF	18-6-1	54.5	113
Smo. Mon. SF	18-18-1	42.1	106
Yearly SF	6-2-1	35.9	34
Daily K_p	2-6-1	103.5	151
Smo. Day. K_p	18-2-1	57.1	148
Monthly K_p	12-2-1	41.7	113
Smo. Mon. K_p	18-18-1	21.6	106
Yearly K_p	12-6-1	28.1	34

iteration compounds the errors of previous predictions, in making new predictions, it is difficult to make even a rough estimate of the prediction accuracy. In fact, iteration can lead to stable predictions on one part of a time series, whilst in other parts of the same series it may lead to wild or even explosive behaviour, as seen in the case of daily sunspot number. The results of this sub-section are concerned only with free running iteration. That is the network is only given access to the original time series values at the very beginning, thereafter it must rely on its own predictions.

Figures 5.1 to 5.5 show the freely iterated predictions using the best one step ahead predicting networks from Section 5.6, with sunspot number predictions at the top and K_p index predictions at the bottom of each page. Notice that in many cases periodic structure is evident in the iterated predictions. Table 5.18 summarises the iterated prediction accuracies which vary from the surprisingly good, e.g. the smoothed monthly sunspot number, to the truly awful, e.g. smoothed daily solar flux. These results show that free iteration is not a reliable prediction method.

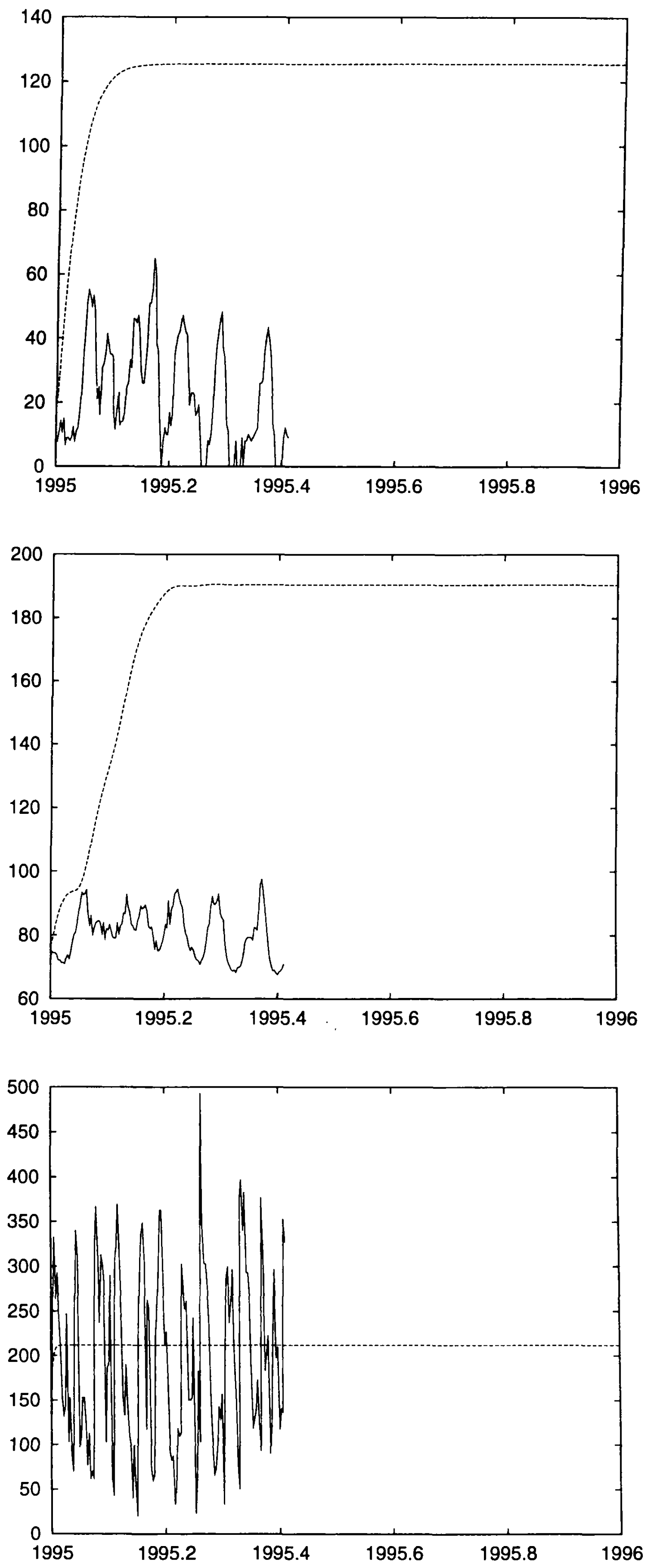


Figure 5.1: The freely iterated predictions of the daily time series.

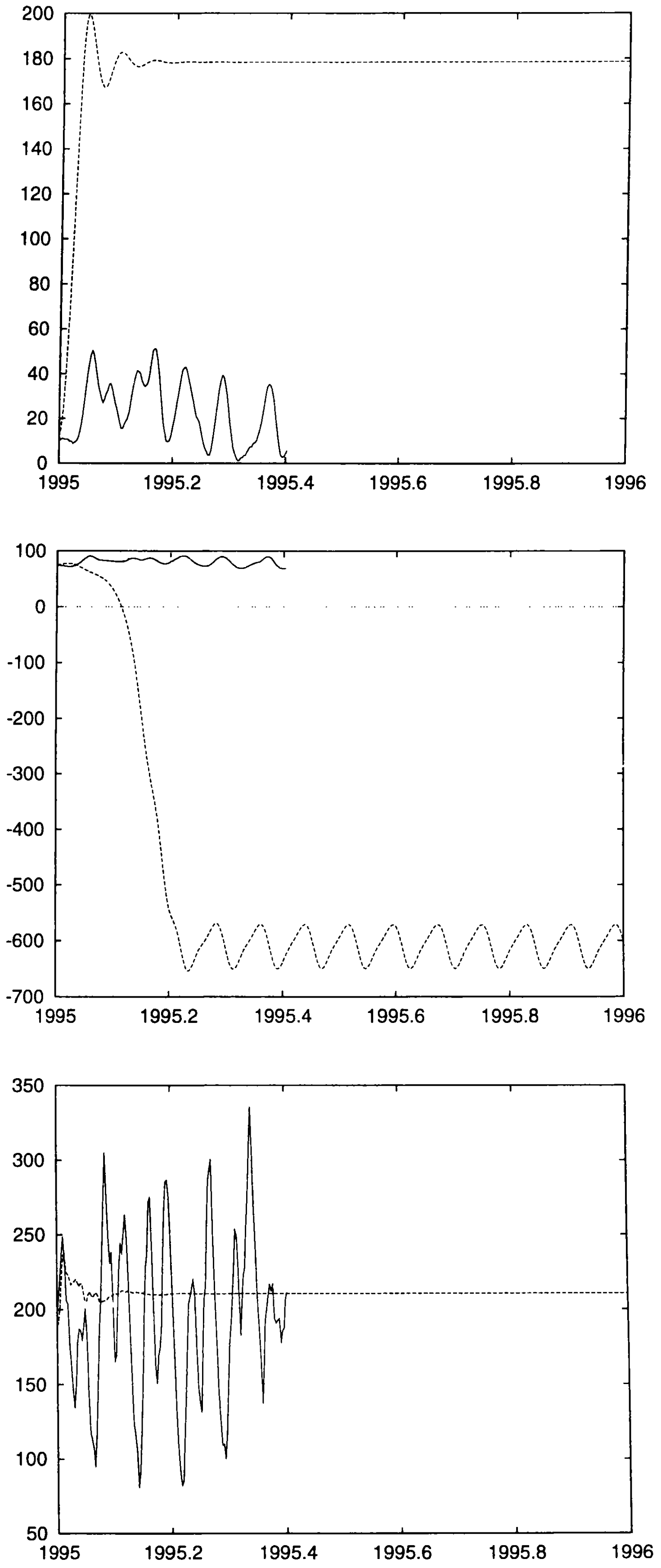


Figure 5.2: The freely iterated predictions of the smoothed daily time series.

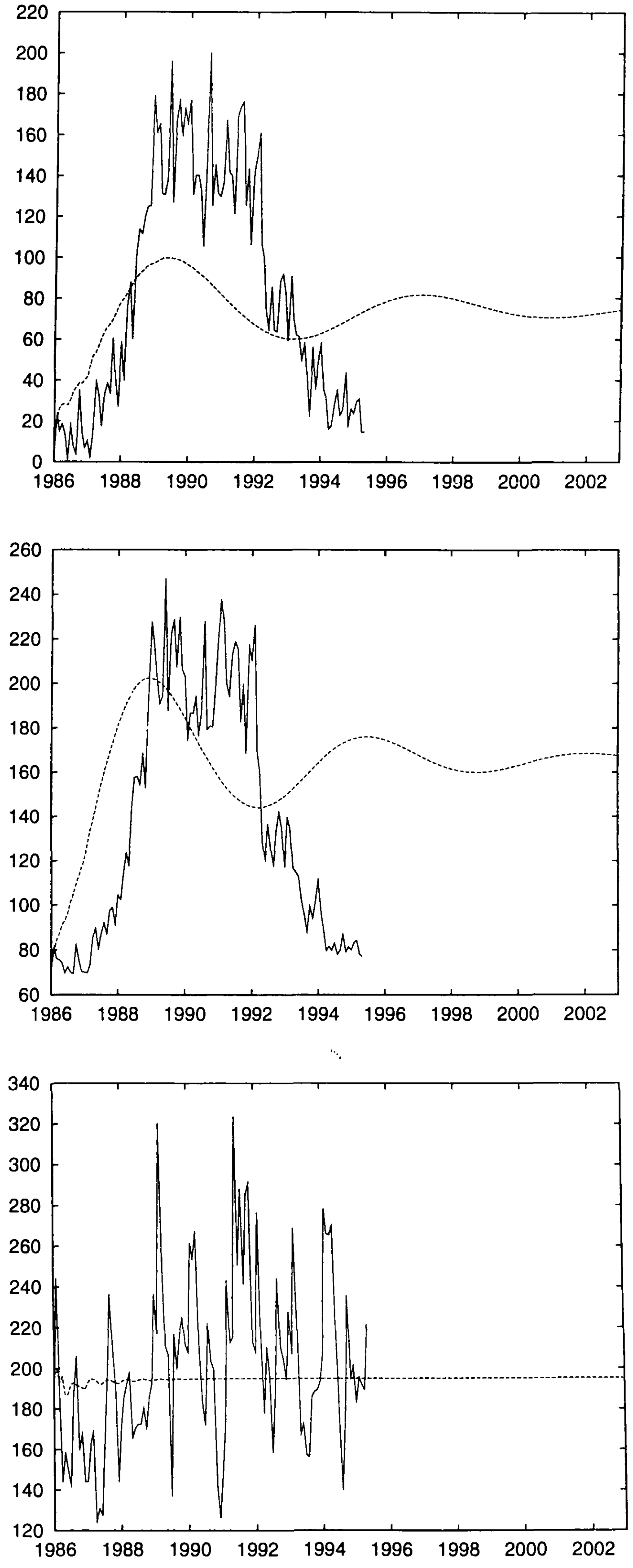


Figure 5.3: The freely iterated predictions of the monthly time series.

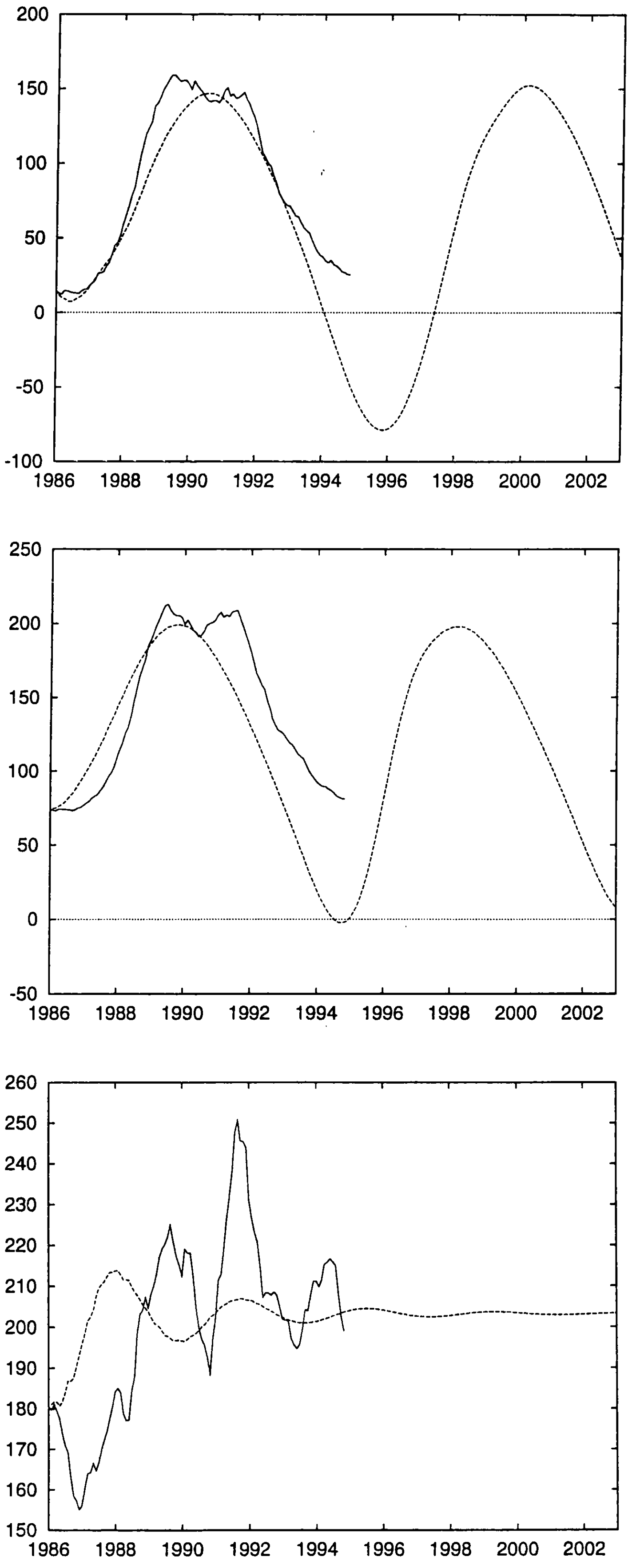
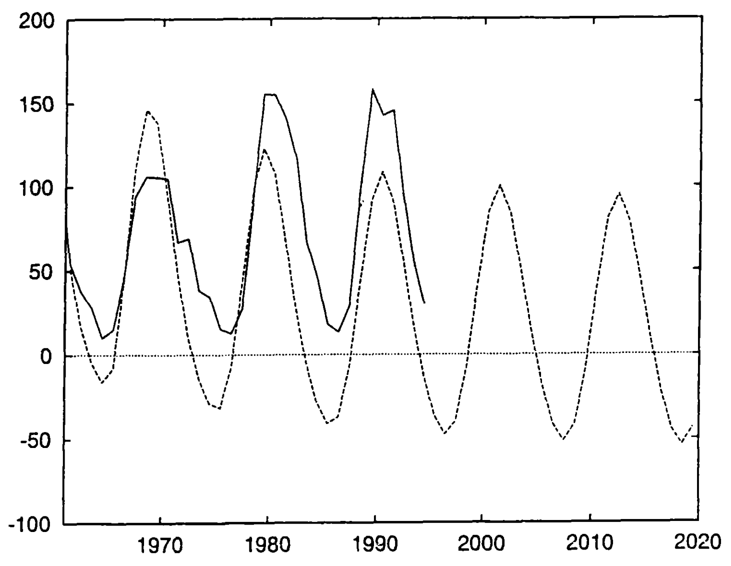
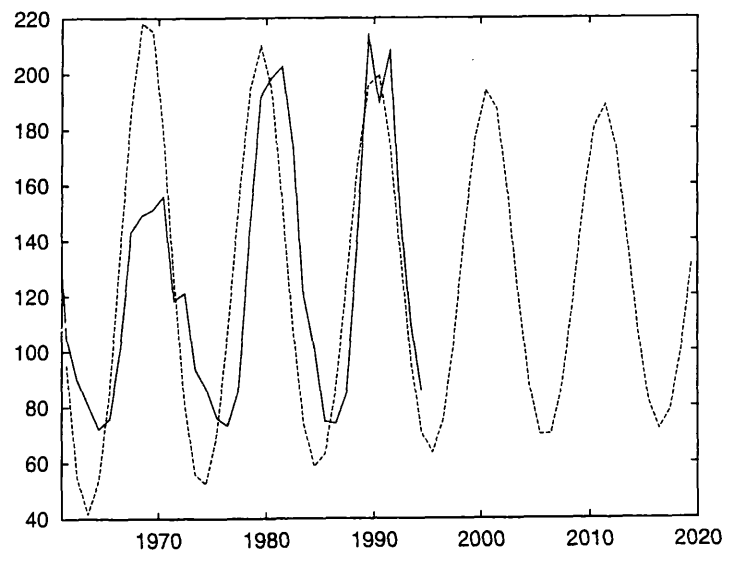


Figure 5.4: The freely iterated predictions of the smoothed monthly time series.





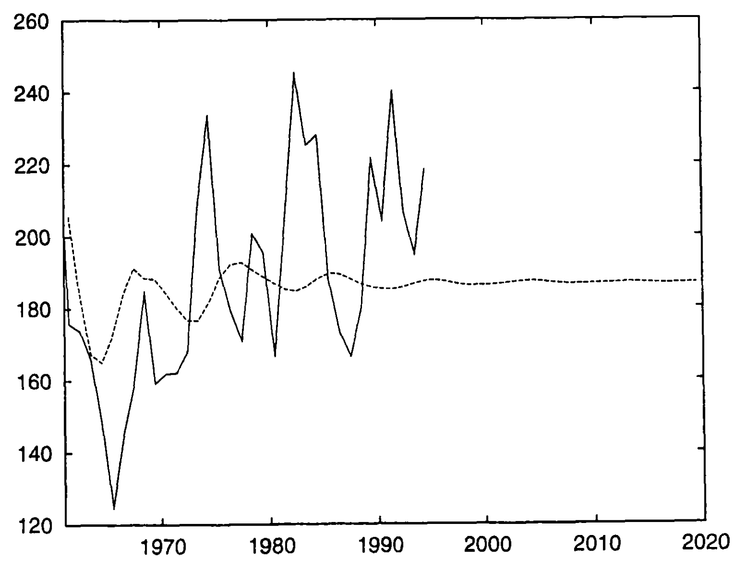


Figure 5.5: The freely iterated predictions of the yearly time series.

5.7.2 Iterating 6 Steps Ahead

Figures 5.6 and 5.7 show some examples of iterated 6 month ahead prediction on the yearly and smoothed daily time series respectively. In the case of the yearly data the predictions are most successful for the sunspot number and the solar flux, presumably because of the strong presence of the solar cycle. Although the predicted maximum values of each cycle are quite unreliable, the year of maximum is predicted quite well. Notice that the time of solar maximum is not usually predicted early, and that if predicted late, it is only so by one or two years.

The smoothed daily predictions show definite signs of a 3 to 4 day delay, and in the case of sunspot number a definite positive bias. This bias, as explained before, is because the training set is dominated by solar maximum data, whereas the plotted predictions are for solar minimum.

Table 5.19 lists the prediction errors for both the iterated and non-iterated 6 step ahead predictions. In almost every case the non-iterated predictions are superior to the iterated predictions. The exceptions are the daily K_p index (for which the difference in errors is negligible, given their huge size) and the three yearly series. This may be because a 6 year ahead prediction requires the network to leap across half a solar cycle, a task which is beyond the simple 2 and 6 hidden neuron networks found to be successful at predicting 1 year ahead. To show that this is really the case, I have trained a 12-18-1 network to predict yearly sunspot 6 steps ahead and found it to have a prediction error of 28.3.

5.7.3 Learning to predict 6 Steps Ahead

In order to have a comparison for iterated prediction I have trained networks to predict 6 steps ahead. In the absence of any other guidance I have simply used the architectures that were found to be the best for one step ahead predictions. The training proceeded as before, with an initial run of iterations with learning rate $\epsilon = 0.01$, followed by either further iterations with this value or with the smaller value of $\epsilon = 0.01$. The training and tests sets are the same as was used before. The final test set RMS errors for each network are listed in Table 5.19.

5.7.4 Iterating n steps ahead

Figure 5.8 shows the prediction errors achieved by iterating 2 to 18 steps ahead for the three smoothed monthly series. (Note that these errors are calculated on the entire history of sunspot number which is why the 6 step iteration prediction errors do not correspond with those in Table 5.19). Surprisingly, while the error increases linearly for the sunspot number and solar flux, the error actual levels out

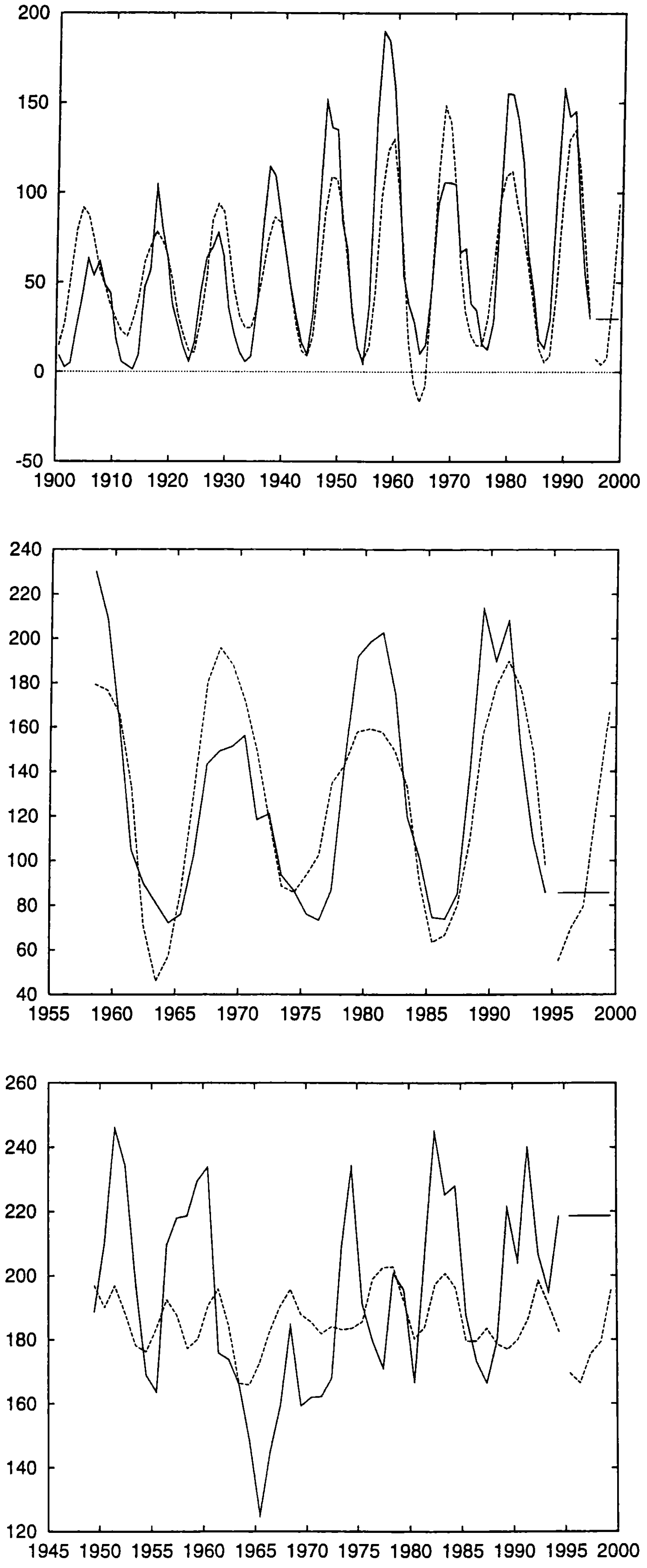
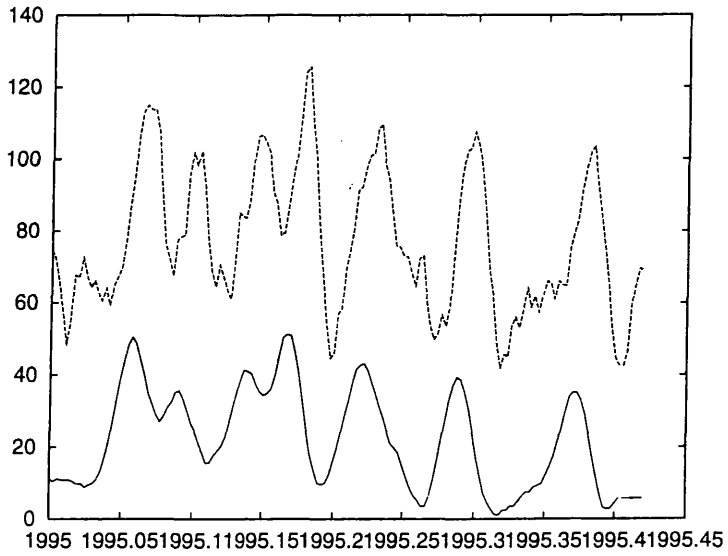
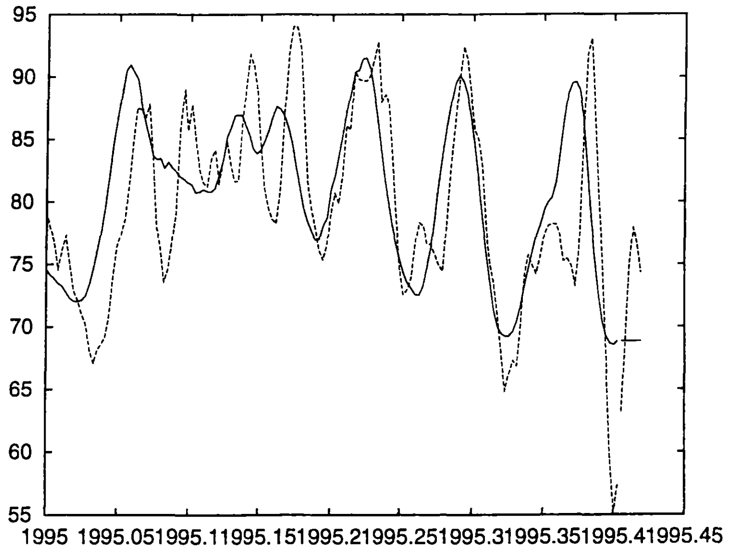


Figure 5.6: The 6 year ahead iterated predictions of the yearly time series.





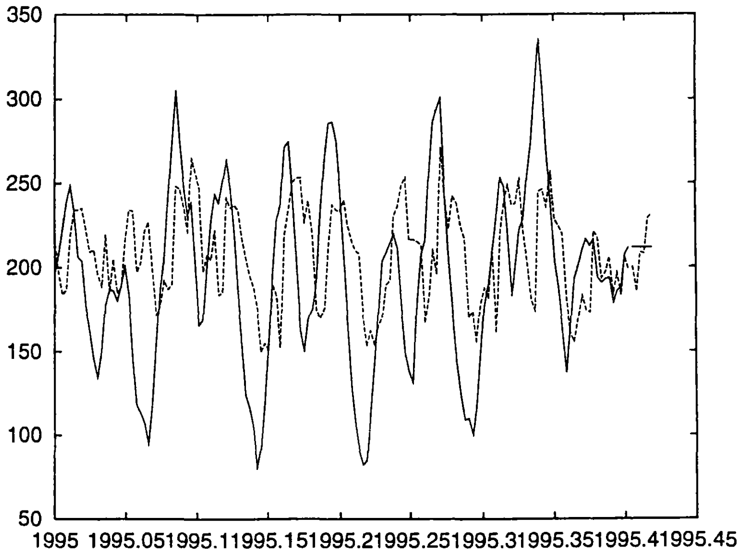


Figure 5.7: The 6 day ahead iterated predictions of the smoothed daily time series.

Table 5.19: Below are the errors of six month ahead prediction using: a network iterating 6 steps ahead; and a network that has been trained to predict 6 steps ahead. All errors were calculated on the test set of the network that was trained to predict 6 steps ahead.

Data	Network	Iterated	Non-Iterated
Daily SSN	2-2-1	39.5	32.5
Smo. Day. SSN	2-2-1	54.3	22.1
Monthly SSN	18-6-1	26.0	24.0
Smo. Mon. SSN	18-18-1	10.5	7.6
Yearly SSN	12-2-1	33.8	39.0
Yearly SSN	12-18-1	-	28.3
Daily SF	18-2-1	24.3	16.3
Smo. Day. SF	18-2-1	14.9	11.1
Monthly SF	18-6-1	25.2	25.0
Smo. Mon. SF	18-18-1	11.7	9.0
Yearly SF	6-2-1	28.1	33.2
Daily K_p	2-6-1	95.8	96.8
Smo. Day. K_p	18-2-1	61.0	52.8
Monthly K_p	12-2-1	37.9	37.4
Smo. Mon. K_p	18-18-1	17.4	13.5
Yearly K_p	12-6-1	26.7	27.5

and starts to decrease as the prediction is iterated further and further ahead with K_p index. This might be because of the 6 month periodicity that is present in the K_p index.

5.8 Uncertainty and Error

In what has gone before, the RMS error and the time shift plot have been used to judge the accuracy of prediction. In this section I consider how the RMS error may be be used to construct the uncertainty of a prediction. In what follows I make a distinction between "uncertainty" and "error". An uncertainty is a statement about the accuracy of a prediction without reference to the actual value - an uncertainty is of course useful to know in practice when the actual value is not yet known. An error, on the other hand, is the measured difference between the prediction and the actual value.

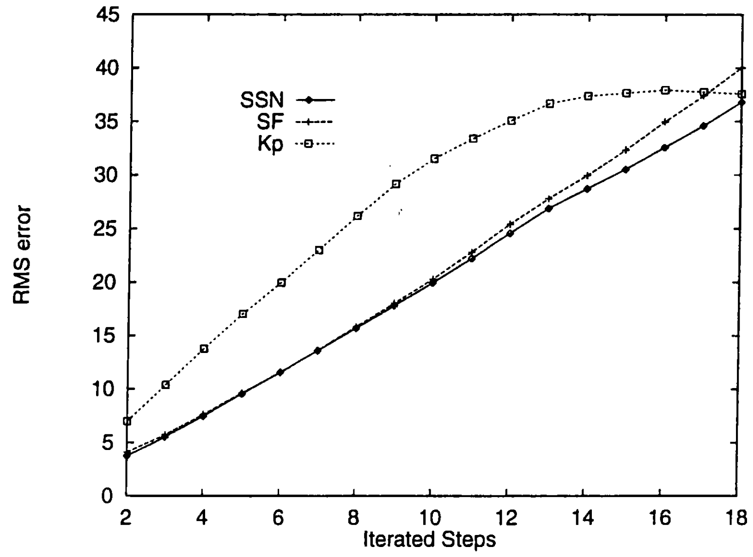


Figure 5.8: The RMS errors for n step ahead iterated prediction of the smoothed monthly series.

5.8.1 The Residuals

I have the defined the residual time series, r_i, in the conventional manner with

$$r_i = \hat{x}_t - x_t$$

where $\hat{x}_t$ is the prediction of x_t. It is obvious from a Figure 5.9 that the time series formed by the residuals possesses neither constant mean nor constant variance. The variance of the residuals being greatest at solar maximum and least at solar minimum. Note that the residuals are usually positive near minima, and also that, around 1957, the mean level of the residuals appears to be slightly negative, almost certainly due to the huge size of the 1957 solar maximum.

Figure 5.10 shows histograms: for all the residuals; for residuals near maxima; and residuals near minima. It is apparent that the minimum predictions do suffer from a positive bias of between 5 and 10 units of solar flux. It is also clear that the residuals have a much larger variance at maxima.

For one month ahead predictions of monthly K_p index, the residuals, shown in Figure 5.9, do not display any obvious structured change in variance or in mean level. A prominent feature of K_p residuals are the large negative spikes, which suggest that on several occasions the network has failed to predict a sudden increase in the level of monthly geomagnetic activity. The histogram in Figure 5.10, indicates a bias in the predictions of about 10 units.

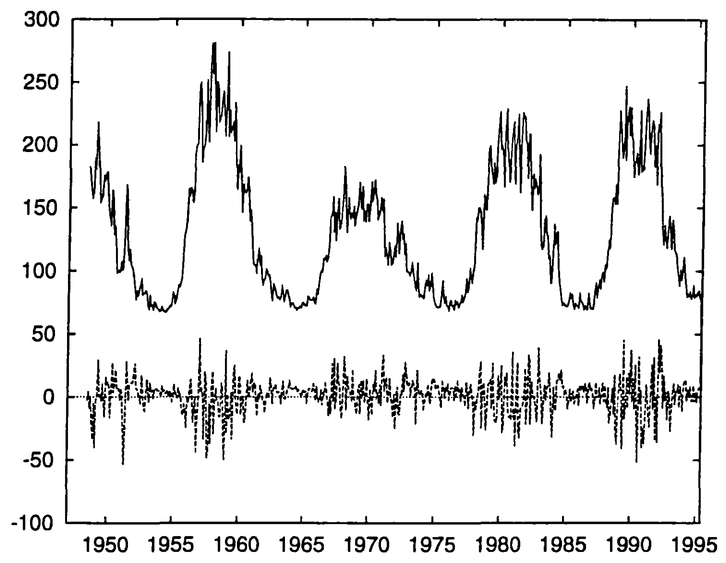


Figure 5.9: The residuals in predicting monthly solar flux one month ahead.

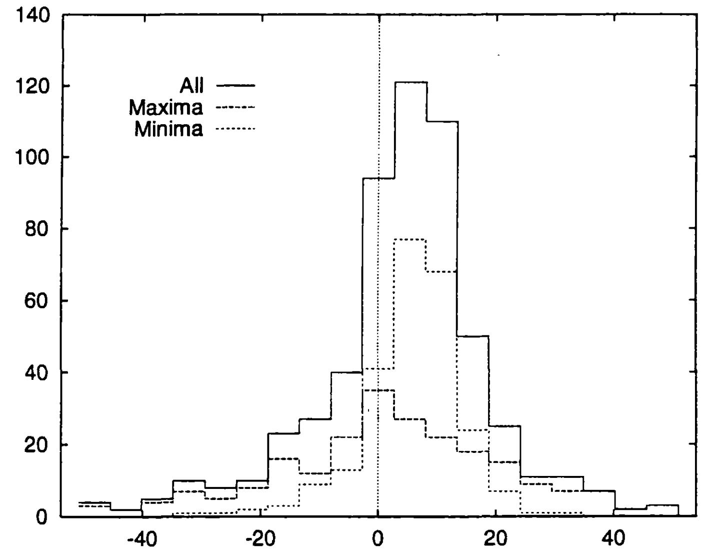


Figure 5.10: Histograms of the residuals in predicting monthly solar flux one month ahead for: all residuals (563 points); residuals near maxima (224 points); and residuals near minima (248 points).

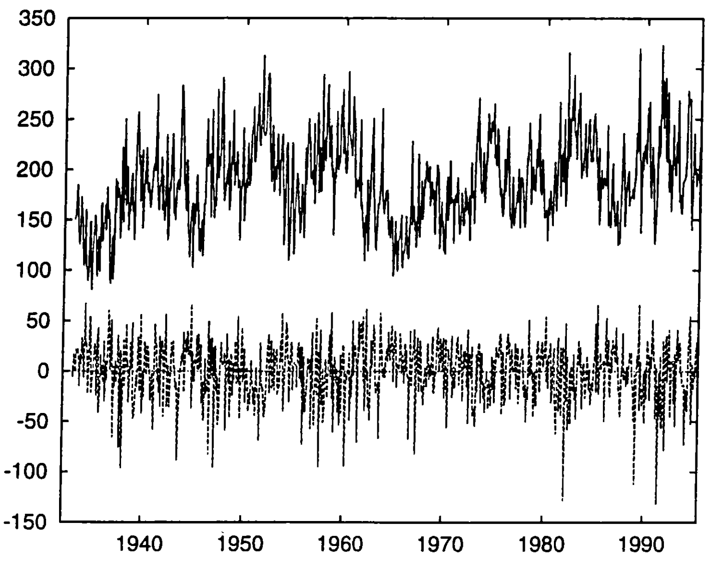


Figure 5.11: The residuals in predicting monthly K_p index one month ahead.

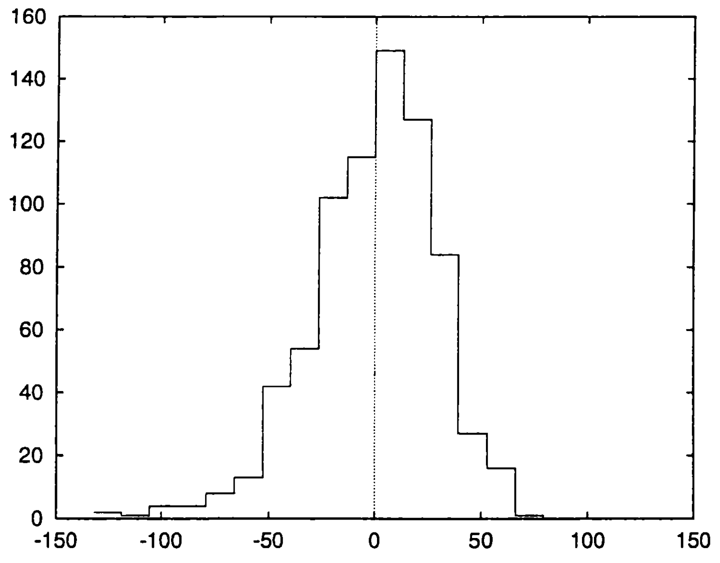


Figure 5.12: Histograms of the residuals in predicting monthly K_p index one month ahead for all residuals (749 points).

5.8.2 Uncertainty and the Solar Cycle

The apparently variable statistical properties of the residuals have several implications when estimating the uncertainty of a prediction. For example, it is not meaningful to make a statement like "the sunspot number for next month will be $X \pm E_{\text{RMS}}$", because for example, the error of a prediction at solar maximum is more likely to be larger (in magnitude) than the error of a prediction made at solar minimum. Also, such use of an RMS error as an uncertainty takes no account of any bias in the prediction.

An improved estimate of prediction uncertainty can be obtained from the histograms of the residuals at maxima and minima. For example, for one month ahead solar flux predictions the following statistics can be calculated from the histograms:

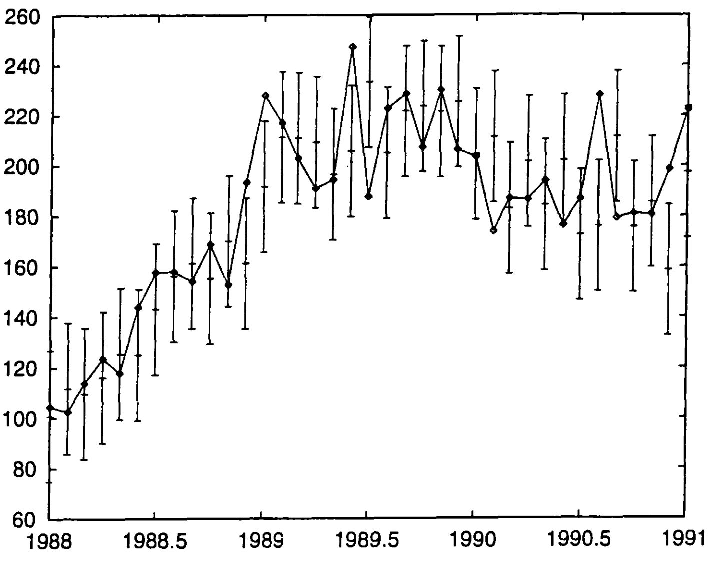
- Maxima - 80% of the residuals lie within ± 26 solar flux units of the true value, and there is no obvious bias in the predictions.
- Minima - 80% of predictions lie within ± 9 solar flux units of the true value, but a bias of 5 to 10 solar flux units is evidenced in the histogram plot.
- Whole data set - 80% of predictions are within ± 17.1 of the true value and again a positive bias is seen.

These numbers can now be used as uncertainties for any future predictions that are made. Figure 5.13 shows the predictions, with their 80% confidence bands, for the minimum and maximum of cycle 22. The size of each error bar is just the appropriate 80% uncertainty value and the error bars for the minimum predictions have been adjusted by 5 units to reflect the bias of the predictions.

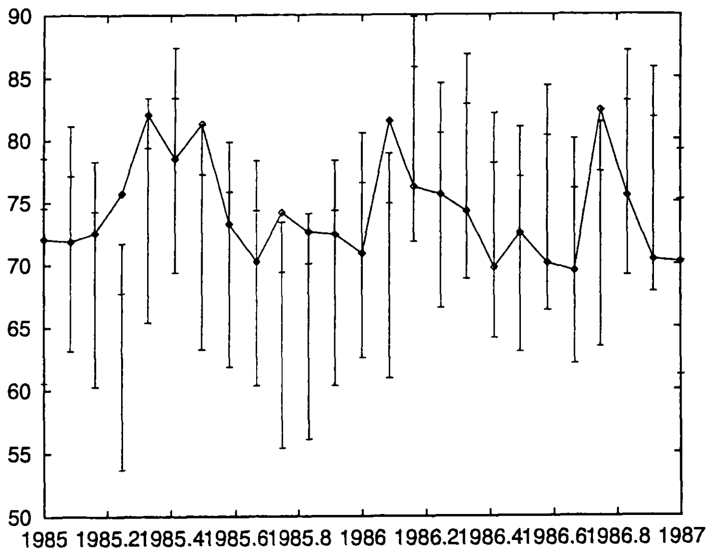
5.8.3 Uncertainties for other series

Tables 5.20, 5.21 and 5.22 show the estimated uncertainties for all predictions obtained in this chapter. The 80% limit is chosen because, for all time series, there are enough predictions in the outer 20% of the residual histogram to make the uncertainty estimate statistically meaningful.

The above method of estimating uncertainty is only necessary (or possible) for the time series that exhibit the solar cycle i.e. the monthly, smoothed monthly and yearly time series of sunspot and solar flux. (In practice however, there was simply not enough data to split the yearly solar flux predictions into minima and maxima in this way.) For the other time series I have constructed the prediction uncertainties using the whole data range.



(a)



(b)

Figure 5.13: Predictions of monthly solar flux at a) solar maximum and b) solar minimum, with errors bars of uncertainty as described in the text.

Table 5.20: This table gives the 80% prediction uncertainties for all the best 1 step head predicting networks

Data	Network	Uncertainty		Bias	
		Min	Max	Min	Max
Daily SSN	2-2-1	14.7		3	
Smo. Day. SSN	6-2-1	3.5		3	
Monthly SSN	18-6-1	15.4	29.5	0	0
Smo. Mon. SSN	18-18-1	1.8	3.2	0.5	0
Yearly SSN	12-2-1	17.9	32.3	0	0
Daily SF	18-2-1	7.4		5	
Smo. Day. SF	18-2-1	2.1		-0.5	
Monthly SF	18-6-1	9.6	26.2	5	0
Smo. Mon. SF	18-18-1	1.5	2.9	0.5	0
Yearly SF	6-2-1	28.0		0	
Daily K_p	2-6-1	89.7		50	
Smo. Day. K_p	18-2-1	16.1		3	
Monthly K_p	12-2-1	35.2		10	
Smo. Mon. K_p	18-18-1	4.4		2.5	
Yearly K_p	12-6-1	30.2		0	

Table 5.21: This table gives the 80% prediction uncertainties for all the best 1 step head predicting networks iterating 6 steps ahead

Data	Network	Uncertainty		Bias	
		Min	Max	Min	Max
Daily SSN	2-2-1	41.0		10	
Smo. Day. SSN	6-2-1	29.2		10	
Monthly SSN	18-6-1	27.0	37.1	0	0
Smo. Mon. SSN	18-18-1	12.2	17.2	2	-5
Yearly SSN	12-2-1	23.3	52.9	0	0
Daily SF	18-2-1	29.4		10	
Smo. Day. SF	18-2-1	24.2		0	
Monthly SF	18-6-1	20.0	33.7	0	0
Smo. Mon. SF	18-18-1	10.2	15.2	6	0
Yearly SF	6-2-1	28.0		0	
Daily K_p	2-6-1	115.7		80	
Smo. Day. K_p	18-2-1	83.1		20	
Monthly K_p	12-2-1	46.4		15	
Smo. Mon. K_p	18-18-1	25.8		10	
Yearly K_p	12-6-1	37.8		0	

Table 5.22: This table gives the 80% prediction uncertainties for the direct 6 step ahead predicting networks. (*This figure is an 85% confidence value)

Data	Network	Uncertainty		Bias	
		Min	Max	Min	Max
Daily SSN	2-2-1	41.5		10	
Smo. Day. SSN	6-2-1	27.8		10	
Monthly SSN	18-6-1	24.4	35.7	0	0
Smo. Mon. SSN	18-18-1	8.9	11.8	0	0
Yearly SSN	12-2-1	19.6	54.9	0	0
Yearly SSN	12-18-1	25.4	50.1*	0	0
Daily SF	18-2-1	23.4		15	
Smo. Day. SF	18-2-1	17.2		15	
Monthly SF	18-6-1	17.6	34.0	-10	-10
Smo. Mon. SF	18-18-1	9.2	12.9	-8	-8
Yearly SF	6-2-1	40.4		-50	
Daily K_p	2-6-1	117.2		50	
Smo. Day. K_p	18-2-1	63.1		10	
Monthly K_p	12-2-1	47.3		-50	
Smo. Mon. K_p	18-18-1	16.9		-15	
Yearly K_p	12-6-1	37.0		75	

5.9 Choice of Training Set

Until now, little has been said about how the training sets were chosen. As a rule of thumb I have generally used about the first 70% of the available data as a training set and last 30% as a test set. Although the 70:30 ratio is used in all cases, it was not necessary (or practical) to use all the available data for every time series. For the daily time series, only the last ten years of data was used, because the inclusion of all data not only slowed down training but seemed to hamper the network's ability to learn to predict. For similar reasons, only monthly sunspot from after 1900 was used to train and test networks.

The choice of training set only presented a serious problem for the daily series of sunspot number and solar flux, where the training set consisted mostly of examples from solar maximum, whilst the test set consisted mainly of examples form solar minimum. To improve upon this situation a network could be trained specifically to predict either maximum or minimum data. Such an approach was taken by Macpherson and Kirkton (1995), but was found to be equally problematic because of the scaling problems introduced by the different sizes of solar maxima. Although not investigated within this thesis, a possible way out of these problems is to use the wavelet transform to separate the long timescale (4000 day) solar cycle from the daily structure.

From the point of view of prediction, the largest maximum of all time, that of cycle 19 in 1957, is a problem. In the monthly and yearly sunspot number predictions of this chapter, this maximum resided in the training set and not the test set. This is perhaps unfair because the network's output has little chance of reaching values that were not represented in the training set. The same architecture (12-2-1) network as used above was trained on a training set of yearly sunspot number from 1850 to 1976, with a test of 1977 to 1994 (87:13 ratio). The test set RMS error in this case was 16.0, a definite improvement on the previous error of 20.2. The question is whether this improvement is entirely due to the fact that the 1957 maximum is not in the new test set, or it is because this new network is simply better at predicting? The obvious way to find out is to re-evaluate the RMS error of the previous network on the new test set - doing so reveals the error to be 20.14. This means that the inclusion of the 1957 maximum is really a necessity, presumably because it prepares the network for the large maxima of cycles 19 and 20. This result also serves as a warning that the comparison of two test set RMS errors from different methods can be meaningless, unless the same test set *and* training set are used in both methods.

5.10 Networks with 2 Hidden Neurons

For the daily and smoothed daily time series, networks with fewer hidden neurons (i.e. 2 or 6) were found to be the most successful. The same is true for predicting the yearly time series. For the monthly series, networks with many hidden neurons (12 or 18) seemed to fare best. What conclusions can be drawn from this? At first sight, it is tempting to conclude that the predictions of the monthly series involve a network doing something more complicated than for a network predicting the daily or yearly time series. After all, a network with only 2 hidden neurons cannot represent some functions that are within reach of an 18 hidden neuron network. On the other hand, the latter network can represent any function that a 2 hidden neuron network can. So, unless it is known that back propagation is hampered by the inclusion of too many hidden neurons, there seems to be little hope of drawing any definite conclusions from the number of hidden neurons in the best networks.

In actual fact, some results in Chapter 3 suggest that back propagation might indeed be hampered by the use of extra hidden neurons. Assuming this to be true then what functions can a network with only 2 hidden neurons actually perform? As shown in Section 4.4, an I-2-1 network can fit any linear function of I variables, or in other words it can represent any I^{th} order linear auto-regressive time series. Looking at the freely iterated predictions of daily, or smoothed daily sunspot number and solar flux (Figures 5.1 and 5.2), it is apparent that the network will iterate almost linearly to some large (or small) value at which the predictions reach a plateau. (In fact the shape of the iterated predictions is reminiscent of the sigmoid activation function.) The interpretation for the daily predictions is therefore that the network has learned to produce some linear function of its inputs so as to predict an increase if its inputs are increasing or a decrease if its inputs are decreasing. When left to iterate freely, it will just carry on until it exceeds the limits in which it learnt to perform the linear function, at which point, the iterated predictions level out because of the shape of the sigmoid activation function. This explanation also accounts for the delay evident in the daily predictions, as networks operating in this way cannot really predict peaks until after they have occurred. The situation for yearly prediction is not so easy to interpret because the network is capable of producing the cyclic rather explosive behaviour.

5.11 Concluding Remarks

The results of this chapter have covered much numerical ground, and have required several months of computer time. But it has to be said that a year, or even ten years, of computer time (at current CPU speeds) would still not be able to answer every question of interest about which neural network, with which training parameters, on what training data provides the best prediction for a particular time series. What the results of this chapter do provide is guidance for any future work with neural networks in how the above issues should be approached. Also provided are the best predictions that I could make (within the constraints of time and equipment) of the different timescales of sunspot number, solar flux and K_p index. To make these predictions more meaningful, I have constructed 80% uncertainty bands for them, and where appropriate I have given separate values for solar maximum and solar minimum predictions.

Since the vast quantity of numerical results in this chapter might make it difficult to see the forest for the trees, a concise but complete summary of the results can be found in Section 7.6. Also in this section is a prediction of the maximum of solar cycle 23.

Chapter 6

New Methods in Neural Network Prediction

"A good Idea or Theory need not be practical or correct." *Anon.*

The emphasis of the last two chapters was to hunt the parameter spaces of FFNN architecture and back-propagation in order to find the network that provided the best RMS prediction accuracy. In this chapter the emphasis is on varying the methods rather than their parameters. Also, the approach of this chapter is more open-minded, in that it seeks to improve the methods of prediction in more than just the "RMS error" sense. For example, RMS prediction accuracy might even be sacrificed for the attainment of some other goal, such as the elimination of delay.

6.1 Wavelet Filtered Time Series Prediction

The wavelet transform was introduced in Section 2.4.3 where it was used to filter daily time series in order to bring out features around the time scale of the 27 day solar rotation. Here the discrete Daubechies wavelet transform is used to split a stretch of a time series into several time-scales, so that each time scale can be predicted by an output from a FFNN. The idea behind, what I shall call "Wavelet Filtered Prediction" or WFP, is illustrated in Figure 6.1. Basically, the network takes its inputs from the original time series and outputs predictions of the wavelet filtered time series. Although it is possible, in principle, to give the network inputs from the wavelet filtered time series, to do so would be inadvisable for the following reasons. Given a data vector comprised of elements

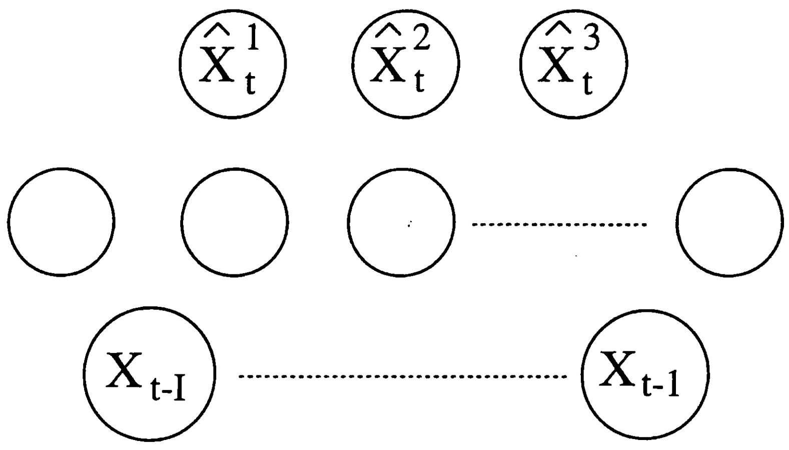


Figure 6.1: A time series X_t is split up into three time-scales using the Daubechies wavelet transform. These three new time series are labelled 1,2 and 3, 1 being the time series of the largest time scale. Taking I inputs from the original time series, the network is required to predict the next values on each of these time scales.

of a time series, i.e.

$$\mathbf{X} = (X_1, \ldots, X_N)$$

the wavelet transform effectively involves performing a convolution with each basis wavelet. Remember that each basis wavelet is formed by translating, and scaling the shape of the mother wavelet. However, if a particular basis wavelet is near (in terms of the scale of the wavelet) one end of the data vector, then it wraps around to the other end in the convolution. The practical implication of this is that each end of the wavelet filtered time series is "contaminated" by the other end, the contamination being most severe for the largest time scale. These end effects make it impossible to use the wavelet filtered time series as inputs, because the inputs required to make an actual prediction of the future, i.e. $X^i_{N-I}, \ldots, X^i_{N-1}$ are contaminated by the behaviour of the time series at the beginning of the data vector. For similar reasons zero-padding the ends of the data vector is equally problematic.

The WFP method promises at least one attractive possibility. If the network is presented with a predictable signal plus constant-mean noise, and the noise is varying on a time-scale that is different from the signal, then the output predicting at the timescale of the noise should learn to always output values close to noise's mean. This provides an interesting test for a suspected white noise component of a signal. On the other hand, if there is reason to believe, a priori, that the behaviour on a particular time scale is white noise-like, and therefore unpredictable, then the corresponding output can be safely removed with no loss of predictive power. The above raises a more general question: from the network's outputs, how can a prediction of the original time series be obtained? The answer is simple, because the wavelet transform is linear,

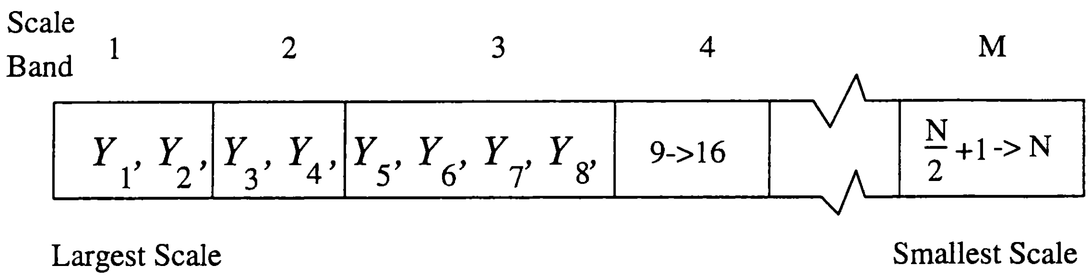
$$\hat{X}_t = \sum_{i=1}^{N_s} \hat{X}_t^i$$

that is the prediction is formed by simply summing the networks outputs at time t[1].

Before carrying out WFP a further question needs to be answered: how many time scales should be used? The discrete Daubechies wavelet transform of a data vector of length $N = 2^M$ will have M distinct time scales, so there can be no more than this number. The following describes, in detail, how the wavelet filtering of a time series into several scale bands is performed. Let $\mathbf{X}$ be the data vector composed of the original time series, and let $\mathbf{Y}$ be the wavelet transformed vector i.e.

$$\mathbf{Y} = DWT[\mathbf{X}]$$

Now, in using the discrete Daubechies transform, as prescribed in Press et al. (1994), the transformed vector is organised as follows:



That is there are N wavelet vector elements with 2 elements in scale band 1, and 2^{n-1} elements in scale band n, for $n = 2, 3, \ldots, M$. If the original time series was monthly then the characteristic time scale of scale band n is 2^{M-n} months. To view the time series on one of these scales all one has to do is set all the other wavelet coefficients to zero and perform the inverse transform, e.g. to obtain the time series in scale band 3,

$$\mathbf{X}^3 = IDWT\left[\mathbf{Y}^3\right]$$

[1]The results here are based on numerical experience and common sense, Section 7.3.2 discusses how these results might be formed on firmer ground.

where

$$\mathbf{Y}^3 = (0, 0, 0, 0, Y_5, Y_6, Y_7, Y_8, 0, 0, 0, \ldots, 0)$$

In practice, it is often more convenient to use several scale bands in the construction of a wavelet filtered time series. In the work that follows I shall use a two scale wavelet filtered time series system, with a large time scale series X_t^1 and a small time scale series X_t^2:

$$\mathbf{X}^1 = IDWT\left[(Y_1, Y_2, Y_3, \ldots, Y_{2^{M/2}-1}, 0, 0, \ldots, 0)\right]$$

and

$$\mathbf{X}^2 = IDWT\left[(0, 0, 0, \ldots, 0, Y_{2^{M/2}}, Y_{2^{M/2}+1}, \ldots, Y_{2^M})\right]$$

The reason why it is preferable to construct fewer wider wavelet filtered time series, each made up from several scale bands, rather than having M wavelet filtered time series each constructed from one scale band, is explained in Figure 6.2, which also shows how the type of wavelet used will affect filtered time series.

To illustrate the above points, I shall demonstrate the WFP method on the time series

$$X_t = 100 \sin \frac{2\pi t}{100} + 100 + a_t$$

the variance of the noise being $16\pi^2$ (this time series was labelled EX2c in the Chapter 4). Figure 6.3 shows the extract of the time series used, and its decomposition into two scales. Since there are 256 data points there were 8 scales bands in the wavelet transform. The first 4 are used to construct the "smooth" time series, whilst the other 4 are used to construct the "detail" time series.

Data in the t range 1811-2028 were used to train the 12-8-2 network, with the rest used as a test set, all errors being calculated on the test set. Figure 6.4 shows the network's predictions of the smooth and detailed time series. As expected the detailed time series predictions are close to zero, possessing a much smaller variance than the detailed time series itself. If the smooth time series predictions are used as predictions of the original time series, then the prediction error is 15.46, which compares very favourably with the networks of Table 4.5 in the last chapter. The 12-6-1 network achieved the best prediction accuracy of 15.25. The error from using the best possible prediction scheme, $\hat{X}_t = 100 \sin \frac{2\pi t}{100} + 100$ is 13.99. Summing the smooth and detail time series for use as predictions yields an error of 15.78.

This last example was clear cut, there was noise on a short timescale and predictable structure on a longer timescale. Likewise, as seen in the last chapter, monthly sunspot number is difficult to predict in the short term, but when smoothed, can be predicted in the long term because of

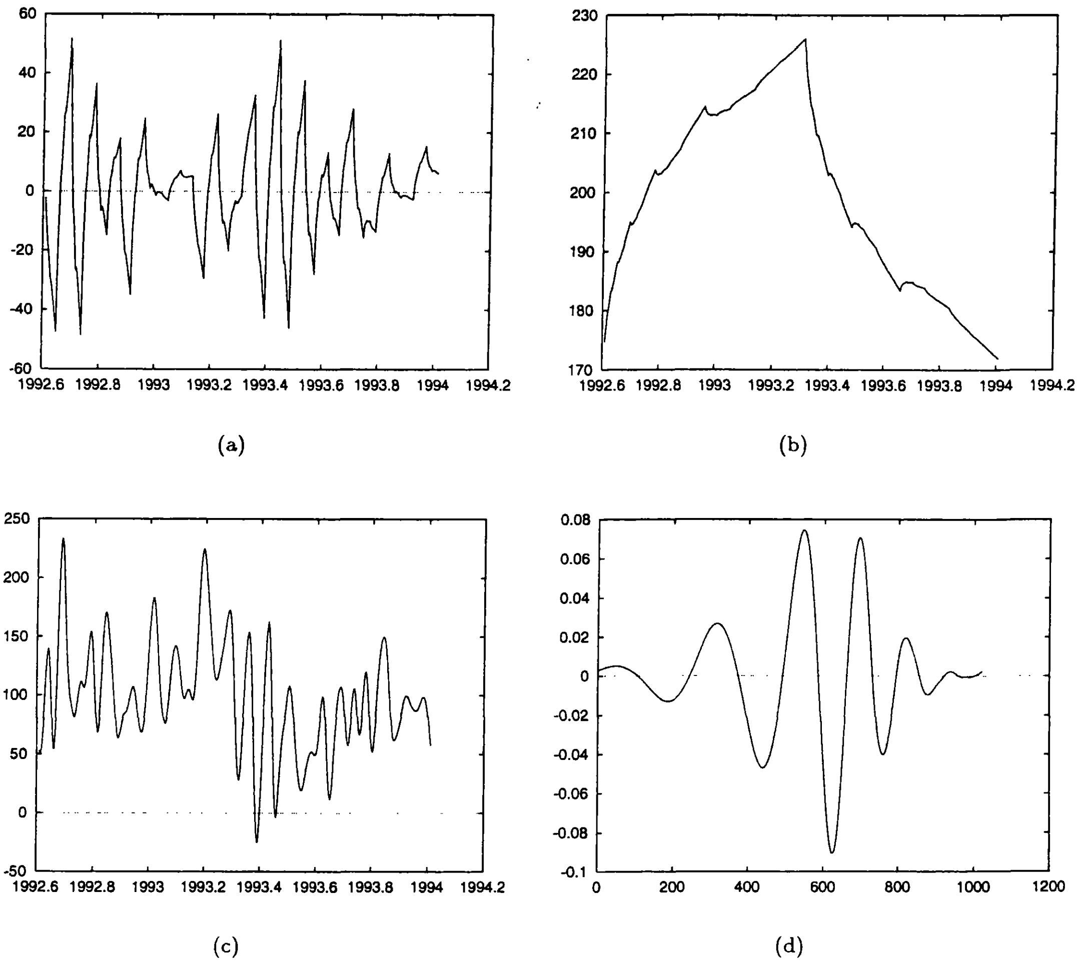


Figure 6.2: Figure a) shows a sequence of the wavelet filtered daily sunspot number, using the Daub4 mother wavelet shape shown in b). The jagged appearance of the Daub4 wavelet is quite apparent in the filtered time series. The fact that the shape of the mother wavelet imprints its form on the filtered time series is a price that has to be paid in using the wavelet transform in this way. However, by using wider scale bands the effect of the mother wavelet's shape is less pronounced (after all, using all time scales yields the original time series). Also, by using a smoother wavelet, such as the Daub20 wavelet d), the appearance of the filtered time series can be made considerably smoother, as shown in c).

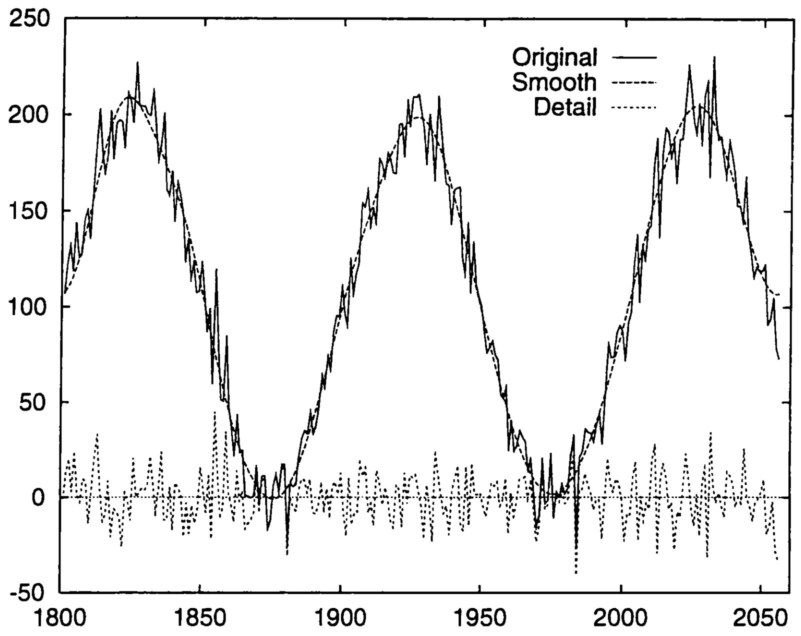


Figure 6.3: This plot shows the wavelet decomposition of a noisy sinusoidal time series into a "smooth" time series and a "detail" time series.

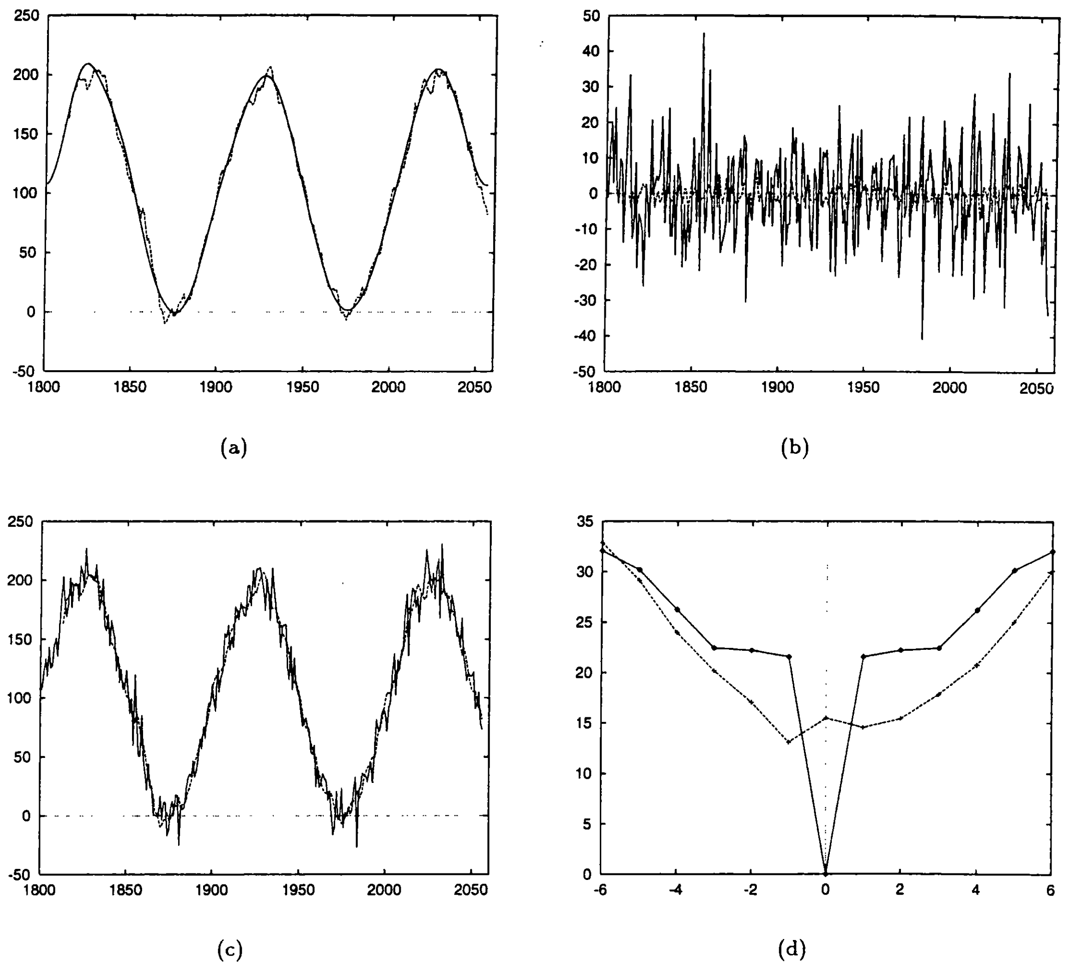


Figure 6.4: a) shows the network's predictions of the smooth time series, b) shows the network's predictions of the detail time series. By using the smooth time series as a prediction of the original time series, as shown in c), an RMS error of 15.46 is achieved. Note that d), the time shift plot of these predictions, indicates that the predictions suffer from delay.

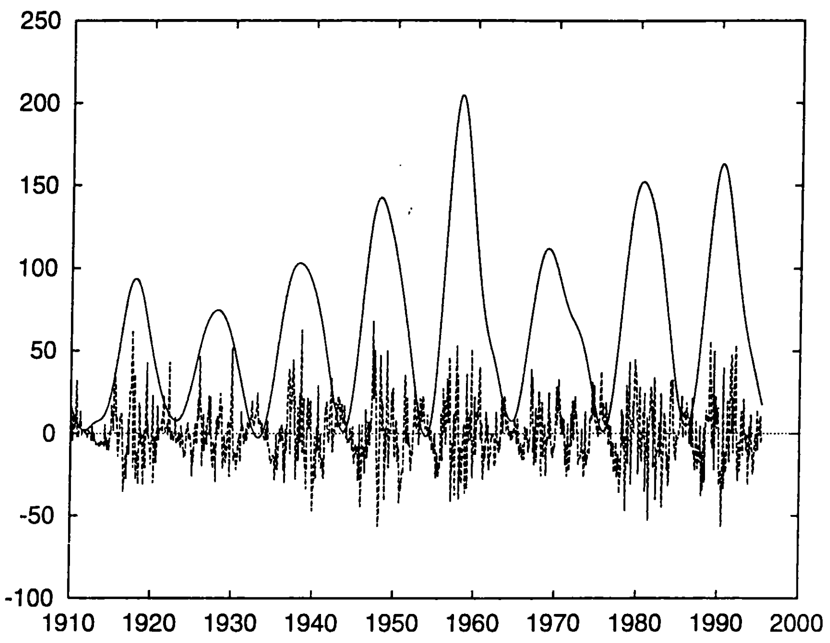


Figure 6.5: This plot shows the wavelet decomposition of monthly sunspot number into a "smooth" time series and a "detail" time series.

the solar cycle. The problem in using the running mean type smoothing is that it is hard to interpret the meaning of the predictions. With the WFP technique the meaning of the smooth time series predictions is clearer: if the smootn time series predictions are added to the detail time series predictions, predictions of the original time series are constructed. As was the case with the previous example, if the short term variation is just noise with a constant mean, then the smooth time series predictions, corrected for the mean noise level, are predictions of the original time series in their own right. However, it is clear from Figure 6.5 that the short term variations in monthly sunspot are not just constant mean white noise because the magnitude of the short term component is maximum when the sunspot cycle is at a maximum.

Since the smooth component of sunspot number is the target of the networks prediction, it is not very meaningful to predict on timescales of less than about six months ahead. The results in Figure 6.6 show that the WFP scheme offer a significant improvement over an echo in predicting 6 months ahead. Notice also that the small variance of the detail time series predictions indicate that the detail time series is effectively unpredictable by these methods. Although the detail time series

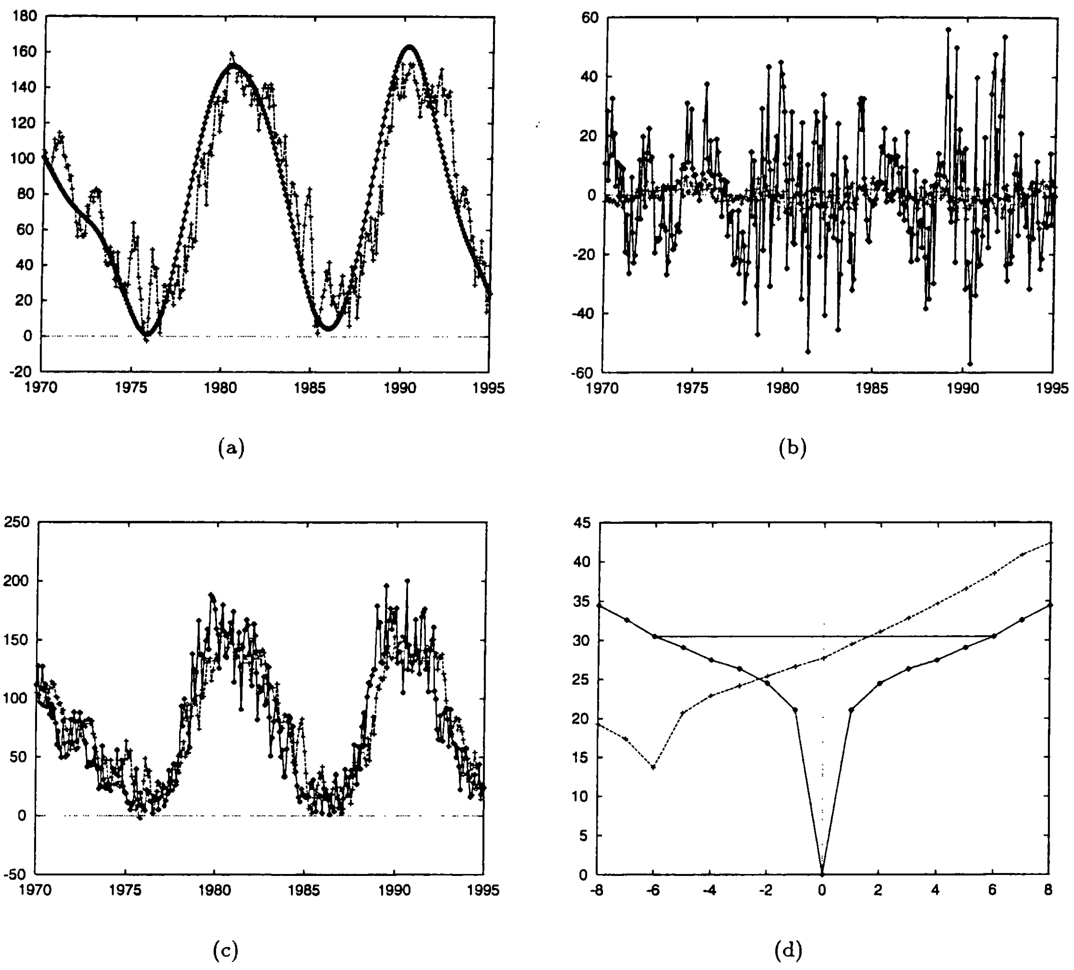


Figure 6.6: These plots show the results of the WFP method prediction 6 months in advance. a) shows the network's predictions of the smooth time series and b) shows the network's predictions of the detail time series. By using the smooth time series as a prediction of the original time series, as shown in c), an RMS error of 27.7 is achieved. Note that d), the time shift plot of these predictions, indicates that the predictions suffer from delay and that the echo error of 30.5 is significantly greater than the WFP prediction error. The 8-8-2 network used was trained using a learning rate of 0.05 until training became unstable, after that the learning rate was reduced to 0.005 and the training continued until again it became unstable. After this no significant improvement could be made by using smaller learning rates.

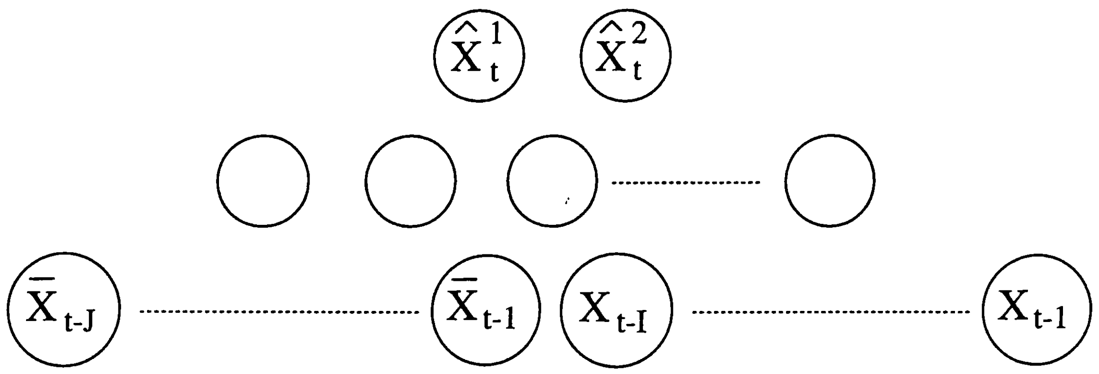


Figure 6.7: The improved WFP method neural network. Running mean inputs have been introduced to allow the network to see recent smooth behaviour of the time series.

is certainly not white noise, its unpredictable nature means that, as before, the smooth time series component alone can be used to provide meaningful predictions.

The reason that the WFP predictions suffer so severely from delay might be because the data at the inputs are so variable whilst the time series to be predicted is so smooth. As explained previously, it is not possible to use the smooth time series itself as an input to the network because of end effects. However, it is possible to give the network some smoother information about the time series by providing it with some running mean inputs. The re-designed network is shown graphically in Figure 6.7, where the running mean series $\bar{X}_t$ is defined as

$$\bar{X}_t = \frac{1}{12}\sum_{n=0}^{11} X_{t-n} \tag{6.1}$$

The results of using this modified version of the WFP method results in much the same RMS error (27.5), but now the delay is less severe. So, the inclusion of the running mean smoothed inputs has made it possible for the network to make a smoother prediction of the future, presumably because it is not too perturbed by spikes and sharp features at its inputs.

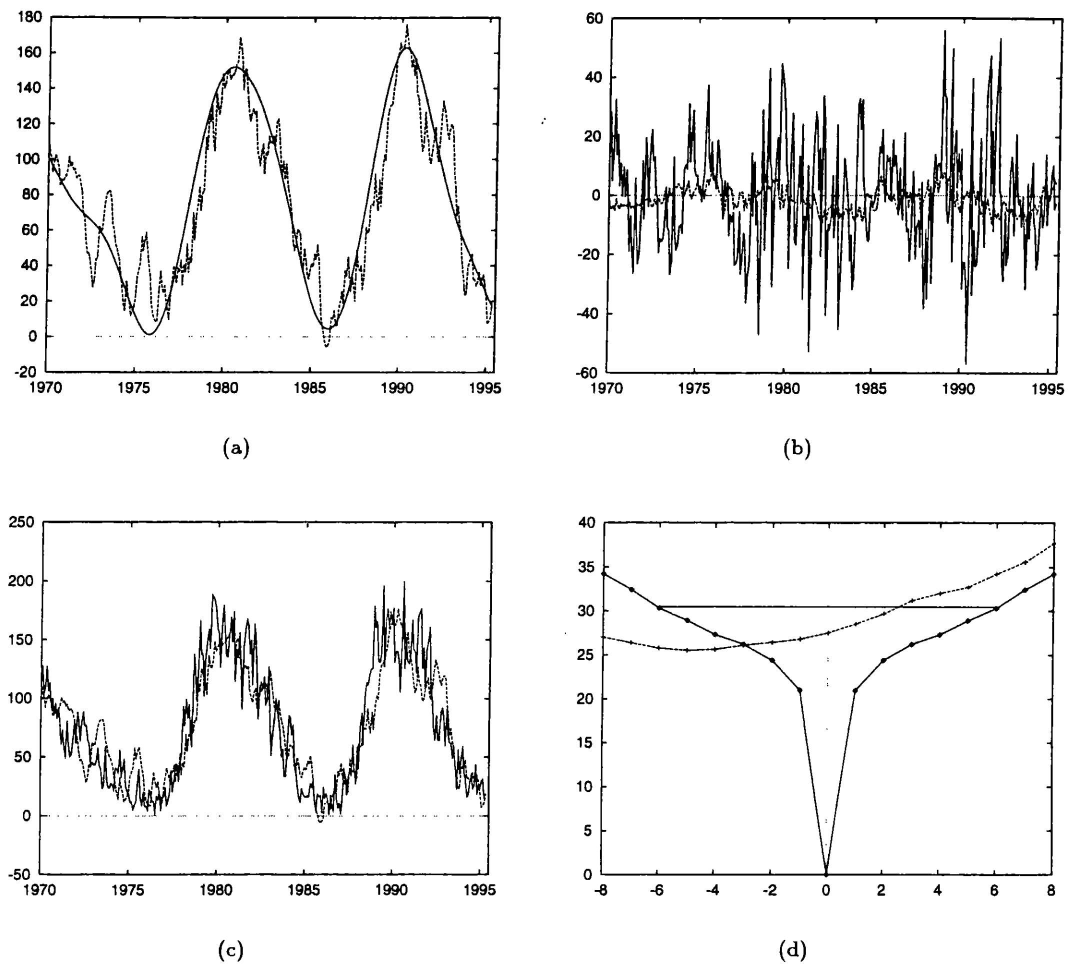


Figure 6.8: These plots show the results of the WFP method with smoothed inputs predicting 6 months in advance. a) shows the network's predictions of the smooth time series and b) shows the network's predictions of the detail time series. By using the smooth time series as a prediction of the original time series, as shown in c), an RMS error of 27.5 is achieved. Note that d), the time shift plot of these predictions, indicates that the predictions suffer from delay and that the echo error of 30.5 is significantly greater than the WFP prediction error. The 12-12-2 network used was trained in a similar fashion to the 8-8-2 network which was used to obtain the results in Figure 6.6. This time, however, four of the inputs presented to the network were from a running mean series described as defined in (6.1).

6.2 The Genetic Algorithm Training Scheme

To understand how a GA can be used to train a NN, consider a population of NNs, all generated randomly (that is the values of their weight connections are set randomly). The following process is performed:

1. Choose a given number of "best" NNs, they are the *parents*, the rest are discarded.
2. Re-stock the discarded population members by randomly selecting two parents and combining them in some manner to produce a *child* - this is called *cross-over*.
3. *Mutate* some or all of the children.
4. Return to step 1.

In principle, this process should be terminated when some generalisation stopping criterion is met. This will be discussed later. The most interesting choices to be made are in laying down the criteria of success. The RMS error of the network's predictions is the obvious candidate, but this neglects the potential power of using a GA. A GA can use any measure that rates the success of a NN in such a way that "good" predictions may be rated over "bad" predictions. The freedom in choosing the measure of success is the GA's strength over Back Propagation. Remember that BP is restricted to the use of an error measure which must be differentiable with respect to the network's weights. The aim is to train or "breed" networks that do not predict with delay, so the measure of a network's success could simply be the RMS error of the network's predictions multiplied by a penalty factor related to the delay in predictions, i.e.

$$E_{\mathrm{GA}} = E_{\mathrm{rms}} P_{\mathrm{delay}}$$

where E_{rms} is just the rms error on the training set and P_{delay} is a penalty term for delay. This error will be hereafter referred to as the *penalised error*. I have found that the following simple penalty scheme worked well in practice

$$P_{\mathrm{delay}} = \begin{cases} 1 & \text{if the time-shift plot is minimum at } \tau = 0 \\ 500 & \text{otherwise.} \end{cases}$$

Although networks exhibiting delay will be at a severe disadvantage in the ranking of the genetic population, the genetic algorithm training method does not guarantee that the trained network will predict without delay. For example, in the case of the AR1 (random walk) time series, the best prediction, in any sense, is just an echo. This means that all the networks will suffer the penalty term, though some will be more adept than others at performing the echo.

Returning to the mechanisms of the GA, there are two practical questions that must be addressed:

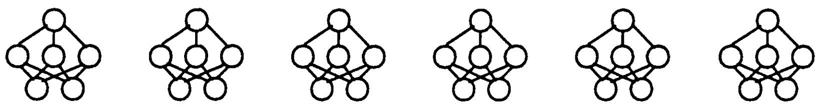
- How should the cross-over be performed?
- How should the mutation be performed?

A typical scheme that I have constructed for performing these operations on FFNNs is described pictorially in Figure 6.9. The *cross-over* operation involves choosing two *parent* networks and breeding a *child* network by taking the inputs weights from one parent and the output weights from the other parent. The design of the mutation step caused more of a problem. At first I bore with tradition and randomly mutated some of the child networks' weights, picking the weights to be mutated at random. This method was unable to train the networks, at least on the timescale of my Ph.D. In an attempt to improve the situation I replaced the random mutation with a few iterations of back-propagation, performed on all of the newly born children. With this scheme neural networks were successfully trained, but the resultant best network was no improvement on a back-propagation trained network, as delay was still evident in the predictions. This was almost certainly because the evolution of the network population had become too directed[2]. The networks that received some back propagation training in one generation would almost certainly become the parents of the next generation. After several generations the population would then consist of very many similar networks that had all achieved success through training by back-propagation. Effectively, the GA scheme was just performing back-propagation in a very inefficient manner. To remedy the situation I then tried to increase the diversity of the network population by adding some random mutation. This, however, did not improve matters because the randomly altered networks died out almost immediately because they were always inferior to the BP altered networks. To find a better compromise between directed evolution and random mutation I tried using BP with a very high learning rate. This time BP was able to provide a direction, to some extent, but at the same time networks could suffer quite random changes due to instability in the BP algorithm. This scheme seemed to work best, its main disadvantage being that it was very slow, though some speed-up is possible if certain programming techniques involving "pointers" are used (see Section B.3).

[2]This problem provides an interesting illustration of why it is undesirable to breed with too much zeal towards a pre-defined master-race.

Genetic Algorithm Training

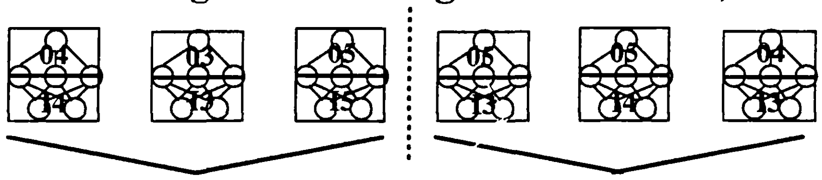
1) Generate many random Neural Networks



2) Test each one and assign it a success rating

E1 E2 E3 E4 E5 E6

3) According to this rating select the best, discard the rest



Parents **Children**

E4 E3 E5 ? ? ?

4) Pick some NNs randomly and mutate them using Back Propagation with a high ε

5) Return to 1) and repeat...

Figure 6.9: A genetic algorithm training scheme for neural networks.

6.2.1 Testing the Method

Referring back to the results of Chapter 4, it is apparent that many of the network predictions showed signs of delay. Further, as shown in Section 4.10, the presence of delay is not necessarily a failure of the networks, because even the best predictions schemes (in terms of RMS error) can produce delayed predictions. The problem lies in the definition of "best" and, in particular, with back-propagation's reliance on a mean-square type error. Re-designing BP to use an error measure that penalises for delay is difficult, because BP deals only with the gradient of the error (w.r.t. the weights) and not with the value of the error itself. It is for this reason that I have appealed to the genetic algorithm scheme, despite its disadvantage in speed.

The genetic algorithm scheme that I have constructed requires several choices to be made:

1. *Population Size* - This is set to 50 for cases studied here.
2. *Number of Parents* - To allow for greater diversity I have chosen a minority of 6 parents in the following tests.
3. *Number of Generations* - This could be determined by some generalisation stopping criteria, e.g. the test set error is increasing - this will be discussed later. Note that the time taken to process one generation depends on the mutation method used, random mutation being by far the fastest option.
4. *Random Mutation* - In the random mutation I have used the following two parameters

   Probability of mutating a given weight connection, p

   Maximum size of the random mutation, A
5. *Back-Propagation Mutation* - In the back-propagation mutation only the learning rate and number of training iterations need to be chosen.

There are, of course, endless variations that could be envisaged, for example, by making the process of choosing a parent a random one, with the most successful networks having the highest probability of selection. However, I shall confine myself to investigating the effect of altering the parameters in points 4) and 5).

The use of **random mutation** is the traditional option in using a genetic algorithm (Goldberg (1989)) because it helps to maintain diversity and because, if a better method of mutating population members is available, then it might well be used in place of the whole genetic algorithm. Now, when a child is constructed from the weights of its parents, there is a fixed probability p that a weight will be mutated by adding some random number in the range $[-A, A]$. The issue here is to balance the need for population diversity with the preservation of established talent. For the random mutation tests performed here, I have used a 2-8-1 FFNN on the MA2 time series, which, when trained with back propagation, produced delayed predictions (see Chapter 4). Each test was run for a maximum of 5,000 generations[3]. In fact, for all the parameters tried, which were all the possible combinations of the following two sets

$$p = (0.001, 0.01, 0.1, 0.7) \times A = (0.001, 0.01, 0.1, 0.5)$$

no progress in training was made at all. In each case the best four parents remained in position, unchallenged, from the first generation, though in some instances the fifth and sixth parents were ousted from parenthood after several thousand generations. It seems, therefore, that the random mutation, as I have implemented it, does not show any promise in training FFNNs to predict time series. One way to improve the prospects of using random mutation comes from the fact that genetic algorithms tend to be employed in searching the parameter space of discrete quantities. This suggests that the weights should be discretised onto an initially coarse grid in weight-space and the genetic algorithm allowed to proceed until little improvement is made from one generation to the next. At this point the discretisation grid is made finer and the process is repeated, making the grid finer and finer as needed. This grid search method is not explored further here.

The **back propagation mutation** must be used with some discretion, otherwise the whole genetic algorithm can degenerate into a very inefficient form of back propagation. In the tests performed here, BP mutation is applied to every newly born child, for a fixed number of training iterations N on the same training set as used before in Chapter 4. The other parameter to be chosen is the learning rate ϵ, for which I have tested the values 0.01, 0.05, 0.1 and 0.5. The main problem with this BP mutation is that it is much slower, because one genetic cycle involves $44N$ back propagation iterations. For this reason, the tests summarised below in Table 6.1 were run for only 100 generations[4]. The results of these tests are very interesting, in that only the use of a larger learning rate succeeds in training networks to predict without delay. This is almost certainly because the use of a small learning rate results in an early dominance of back-propagation trained networks. If this early dominance leads to back-propagation trained networks filling the entire population, then all the networks will compete with the same penalty, and thus the whole process will degenerate

[3] 5,000 generations takes about an hour of run time on a dedicated SPARC 10/40.

[4] For $N = 10$ this took several hours of CPU time on a SPARC 10/40.

Table 6.1: Final RMS test set errors (*not penalised errors*) for the tests on the genetic algorithm training scheme using back propagation mutation. A 2-8-1 network is trained to predict the MA2 time series, without delay. An asterisk indicates the test set predictions showed delay.

Training	*Learning Rate*			
Iterations	0.01	0.05	0.1	0.5
1	0.064*	0.070	0.068*	0.088
5	0.070*	0.069*	0.067	0.078
10	0.069*	0.068*	0.071	0.071

into inefficient back-propagation. Another way of looking at this is to imagine the large learning rates as causing instability in the training, this instability serving as a kind of noise, diverting back-propagation from its directed "down the error slope" path. Figure 6.10 shows the time shift plots for these results. Notice that comparison with the results in Table 6.1 shows that the parameters giving the smallest error, namely (1,0.01), display the most severe delay. The conclusion is that the price to paid for delay-free predictions is indeed a higher RMS error, as was reasoned to be the case in earlier discussions.

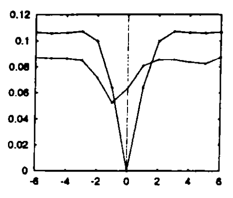
As a final test I tried taking a genetically trained network, that produced non-delayed predictions, and training it further for several thousand BP iterations. This resulted in a network that produced delayed predictions, which, for all intensive purposes, might as well have been trained using back propagation from the beginning.

It is important to note that the above RMS errors and time shift plots were calculated on the test set. This means that the networks have not just learned to avoid delay by over-fitting the training set data. As shown below, in applying the GA training method to networks predicting the natural time series, it seems that such a pit-fall is unavoidable.

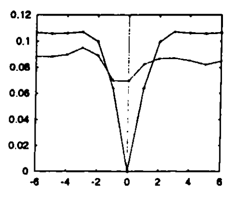
6.2.2 Predicting Natural Time Series Without Delay

First of all I shall use the GA method to train networks to predict the geomagnetic K_p time series without delay.

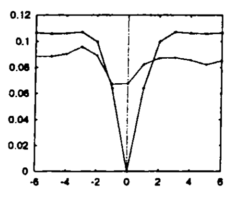
The main problem in making yearly predictions of this series is the lack of data (only 63 years were available at the time of writing). As before I shall use 1933-1976 (44 years) as the training set and 1977-1994 (18 years) as the test set. Despite the shortage of data, the prediction of the yearly K_p index demonstrates that the GA method can be subject to overfitting. The results below will



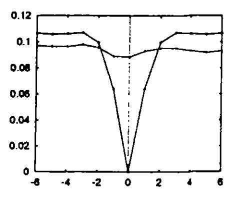
(a) 1,0.01



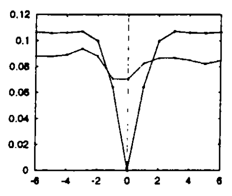
(b) 1,0.05



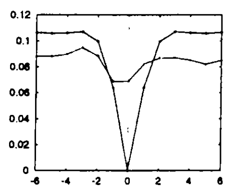
(c) 1,0.1



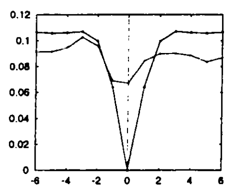
(d) 1,0.5



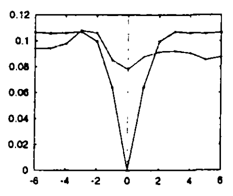
(e) 5,0.01



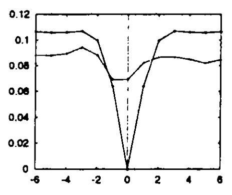
(f) 5,0.05



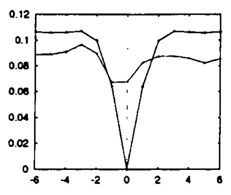
(g) 5,0.1



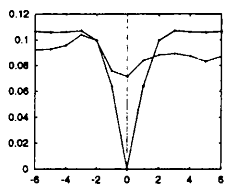
(h) 5,0.5



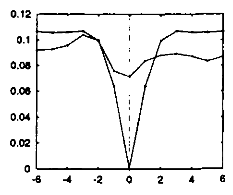
(i) 10,0.01



(j) 10,0.05



(k) 10,0.1



(l) 10,0.5

Figure 6.10: Time shift plots for the tests on the genetic algorithm training scheme using back propagation mutation. The layout of the plots on this page is the same as in Table 6.1.

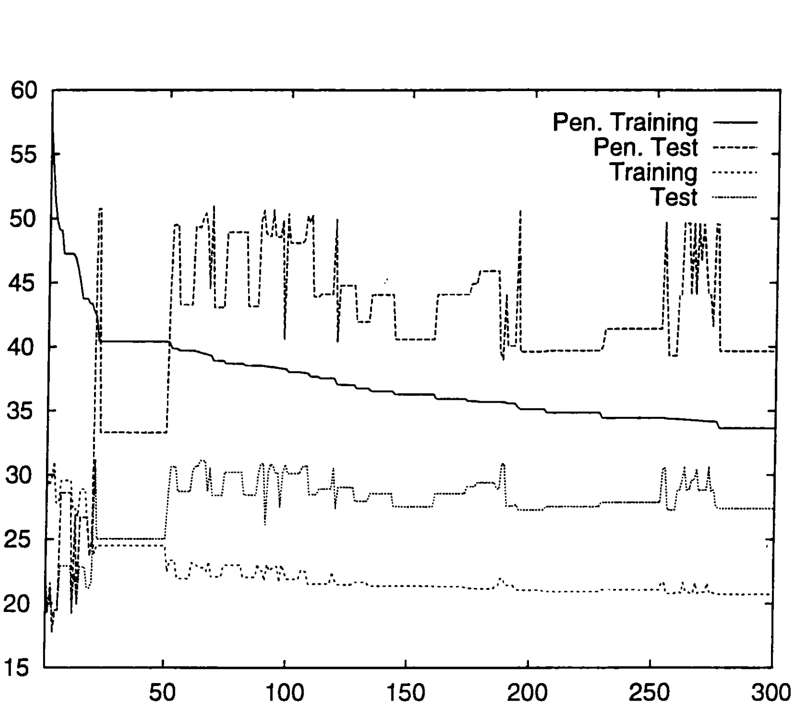


Figure 6.11: The progress of the genetic algorithm training scheme in terms of four error measures of the best network of each generation. The errors are plotted every 10 generations, the numbering on the x-axis of the plot being "No. of generations/10". Note that the penalised errors have been re-scaled for the purposes of plotting.

be used to construct an appropriate generalisation stopping criterion for the GA method.

All the genetic algorithm training runs described below were run for a maximum of 3,000 generations, using a BP mutation step of 5 iterations with a learning rate of 0.1. The criterion for early stopping is defined later, in the light of the yearly K_p index results.

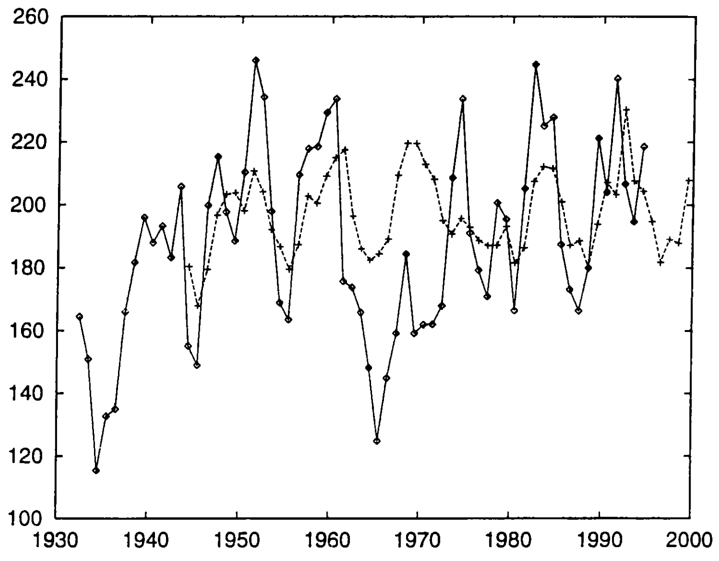
Figure 6.11 plots the four error measures for the best network of each generation, the errors being: the penalised training set error; the penalised test set error; the RMS training set error; and the RMS test set error. The step-like nature of the plots is typical of the genetic algorithm procedure: a network will often remain as the best member of the population for many generations until one of its children, or great^n grandchildren ($n \geq 0$), suddenly improves upon it. Notice that the penalised training set error is the only one that is always decreasing and unlike BP, where the algorithm can become unstable and worsen the network, the GA always improves on, or at least equals, its best error of the previous generation. The penalised test set error shows the most dramatic variations, leaping by more than a factor of two at about generation 200. At the same time notice that the penalised training error shows a marked decrease. This is a classic example of over-fitting, where the training set data is fitted parrot fashion, to the detriment of the network's ability to perform well on the "unseen" test set data. The minimum in error on the test set occurs relatively early on, after only 40 generations - at this point the RMS error of the test set is also at a minimum. Figure 6.12 shows the prediction and time shift plots for the best network of the 40^{th} generation. The predictions appear quite poor, and it seems that the small size of the test set and an element of chance, and not superior predictive ability, have brought about the strange situation where the test set error is so much smaller than the training set error. At generation 400, the genetic algorithm is in the middle of a plateau, where the best network remains in place for several hundred generations. By this time it is clear from Figure 6.13 that some over-fitting has occurred, because the training set predictions show little delay, with peaks and troughs predicted "on-time" (though poorly in size) whilst the test set shows a significant delay. By the last generation (3,000) the training set predictions, shown in Figure 6.14, appear to be quite accurate, with most peaks and troughs predicted on time, despite the slight delay evident in the time shift plot. The test set, however, still shows signs of significant delay and has a much larger RMS error.

To summarise, the 12-6-1 network could not be trained to predict yearly K_p index without delay. Although the network learned to predict the training set with little delay, the predictions of the test set were still heavily delayed, and by the $3,000^{\text{th}}$ generation, the training set error was substantially less than the test set error.

The need for monitoring the generalisation ability of a network trained by the GA method is now clear. There are now two symptoms of over-fitting to be wary of: increase in the test set RMS error, whilst the training set RMS error is decreasing; and delay in the test set predictions whilst the training set predictions show no delay. A stopping criterion, analogous to the one used with BP, based on an increase in the *penalised test set error*, would seem to be an appropriate choice.

The only other format of the K_p index to show significant delay in the tests of Chapter 5 was the monthly series. The genetic algorithm again failed to find a network that predicted this series without delay.

Turning to six month ahead predictions, it was seen in Section 5.7 that even the smoothed monthly predictions suffered from delay, with the minimum in the time shift plot occurring at 4 months. Can the genetic algorithm offer any improvement in this case? As can be seen in Figure 6.15, the predictions obtained from the genetically trained network show little delay on either the training set or the test set, though the price paid in terms of RMS error is extremely high. The error of



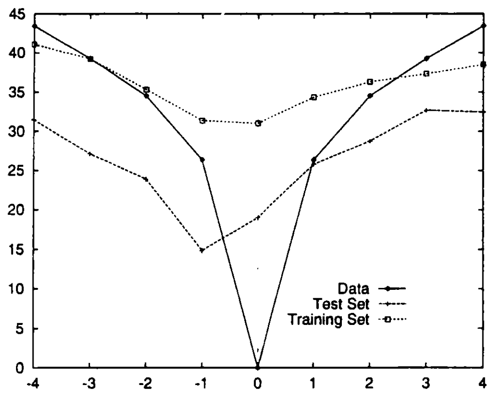
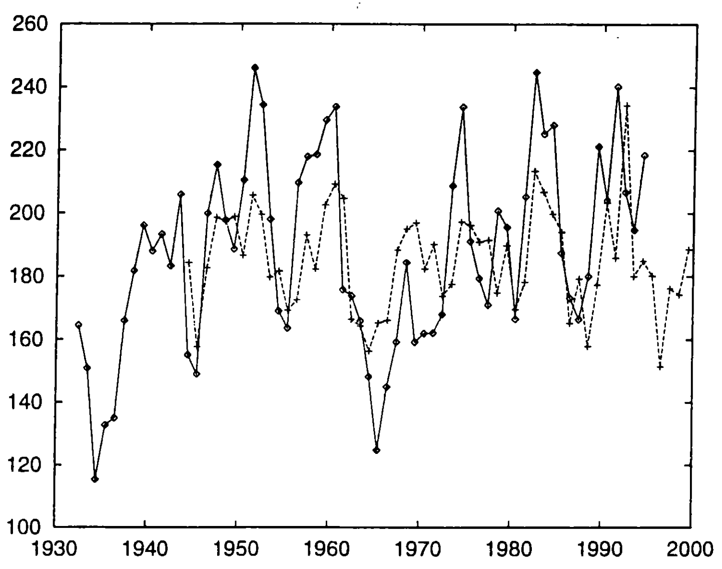


Figure 6.12: The prediction plot and the time shift plot for the best network of generation 40.



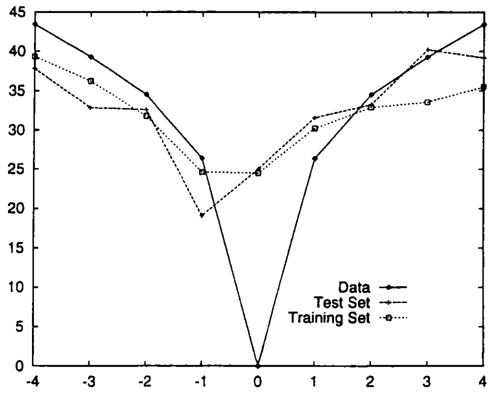
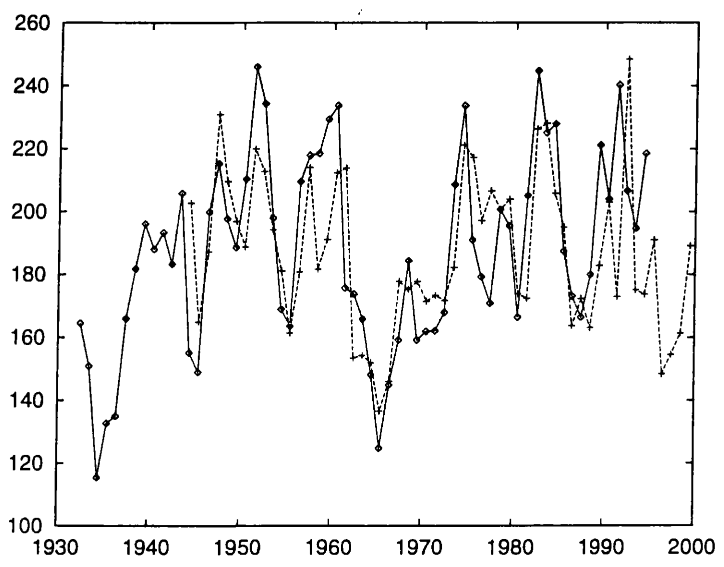


Figure 6.13: The prediction plot and the time shift plot for the best network of generation 400.



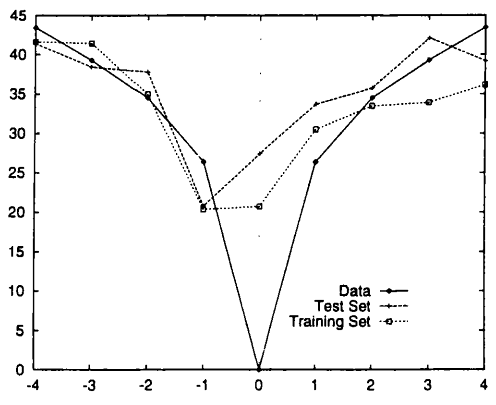


Figure 6.14: The prediction plot and the time shift plot for the best network of generation 3,000.

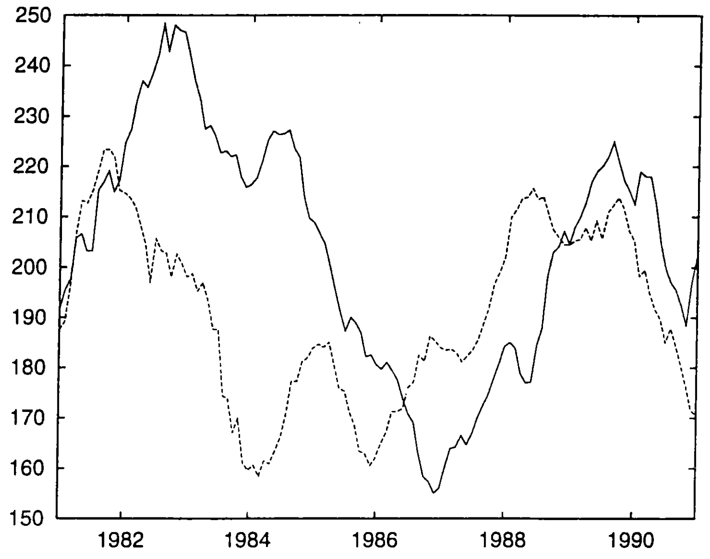
these predictions is 31.7, which is more than twice the error of 14.2, obtained with the BP trained network in Section 5.7. As before, further training resulted in the decrease of the penalised training set error whilst the penalised test set error increased.

The final set of results of this section shows the GA training scheme's success in training a 6-6-1 network to predict the smoothed monthly solar flux six months ahead, without delay. The plot in Figure 6.16 shows its predictions of the 1989 solar maximum, taken from the network's test set. The timing of the prediction is good with the maximum being predicted on time, even though its height is underestimated. The time shift plot shows no delay, and although the prediction error (20.0) is substantially higher than that of the BP trained network (12.8), the error is significantly better than an echo (22.2).

6.2.3 Comments on the Genetic Algorithm Training

The GA training scheme has been demonstrated as successful in obtaining non-delayed six month ahead predictions of smoothed monthly solar flux. The RMS errors of these predictions, however, were higher than the errors for the BP trained network. To see if any improvement was possible in terms of RMS error, the training runs were repeated with different values of learning rate: 0.01, 0.05, 0.5, 1.0. For learning rates of 0.01 and 0.05, as before, the best networks of the last generation produced delayed predictions. With learning rates of 0.5 and 1.0 little progress was made at all beyond the first few generations. It seems, therefore, that a larger RMS error is inevitable in asking for predictions free from delay.

The results concerning K_p index cannot be regarded as a success. In only one case was a network found that produced non-delayed predictions, and the price paid, in terms of RMS error, was unacceptably high. There are two explanations why the networks could not learn to predict without delay. The first is that there do exist networks capable of making non-delayed predictions, but the problem of over-fitting in training prevents them from being found. The second is that there is no way to avoid delay in the predictions, because most peaks and troughs of the time series are simply unpredictable. The first problem might be overcome by brute force, e.g. by performing many GA training runs, until a successful network is found. This is not practical at present given the slow speed of the GA procedure. The second possibility is more serious, and if non-delayed "on-time" predictions are still sought after, then it is necessary to approach the problem for a different direction. The WFP method at the beginning of this chapter is one possibility, because the network's efforts are focussed on predicting the smooth component of the time series. The next section deals with



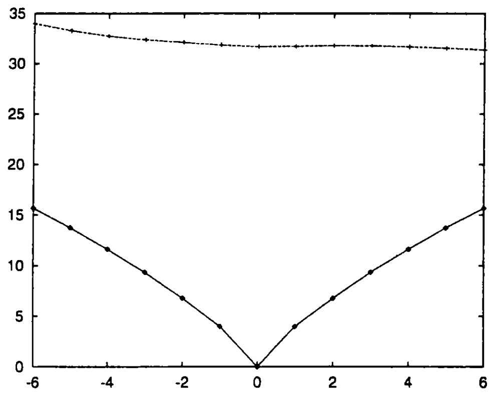
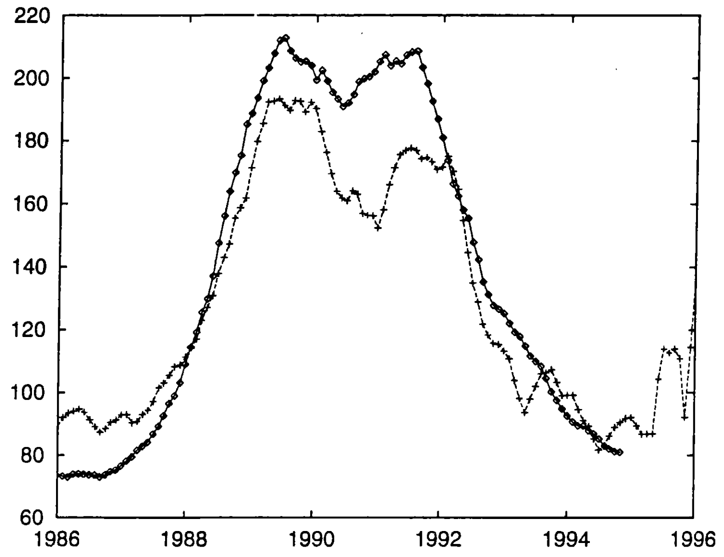


Figure 6.15: The prediction plot and the time shift plot for a GA trained 12-8-1 network predicting smoothed monthly K_p index six months ahead.



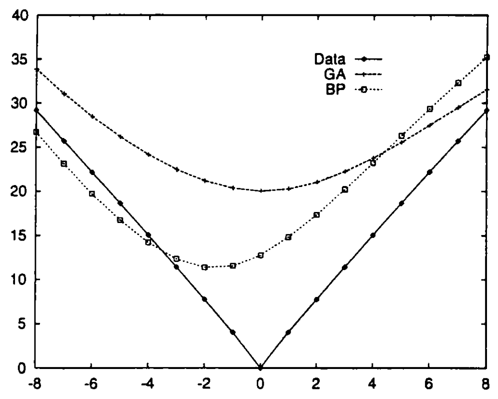


Figure 6.16: The prediction of the 1989 maximum and the accompanying time shift plot for a GA trained network predicting smoothed monthly Solar Flux six months ahead. The time-shift plot for the BP trained network's predictions, from Section 5.7, are shown for comparison.

Table 6.2: RMS errors for networks predicting the daily K_p index. An asterisk indicates that delay was present in the predictions.

Network	SSN	K_p	Error
4-4-1	2	2	112.258*
8-6-1	6	2	114.169
12-12-1	6	6	120.7
2-2-1	0	2	86.4*
Echo	0	0	92.9

another possibility, appealing to the physical and causal connection between the Sun's behaviour and the geomagnetic K_p index.

6.3 Dual Input Data Networks

In this method a network is given inputs from two time series and is required to predict only one of them. Reasons why this might be useful are apparent from the cross-correlation plots of Section 2.4.2, where it was seen that the daily K_p index (during solar maxima) seemed to be most correlated with the sunspot numbers from a 5 or 6 days beforehand. On the timescale of years there is also scope for dual data input networks because, remember that Schatten and Sofia (1987), amongst others, exploited the level of geomagnetic activity at solar minimum to predict the height of the next sunspot maximum. In fact, the dual input method has already been used in conjunction with the WFP method, where the monthly and smoothed monthly data were given as network inputs.

6.3.1 Daily Data

The networks tested here took inputs from the daily sunspot number and the daily K_p and were required to predict the latter 1 day in advance. The networks were trained on the daily data of the years 1989 and 1990, with 1991 data being used as a test set. Very small learning rates, as low as 0.0005, had to be used to train these networks, as higher values caused immediate instability in training. Table 6.2 shows the prediction errors achieved for several different networks.

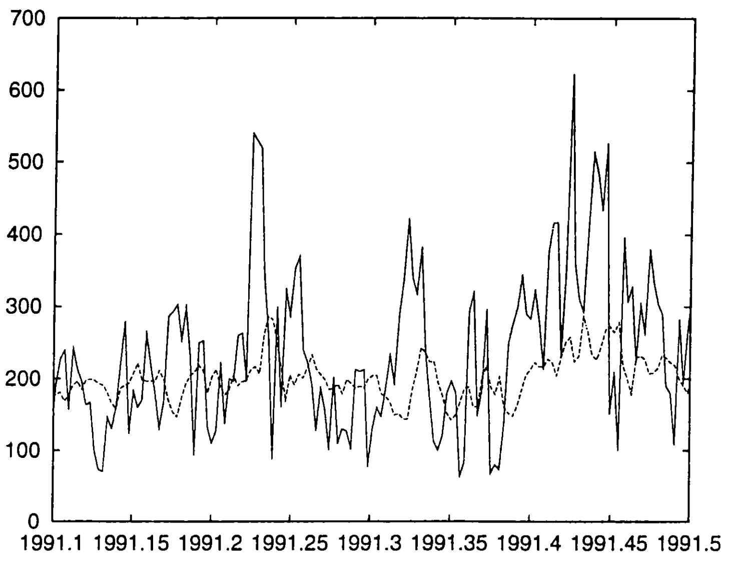
The predictions of the best of the dual input networks, the 4-4-1 network, which takes only 2 inputs from each series, are shown in Figure 6.17. Similar tests were also performed using data from the 1946 maximum, but again the dual input data networks were difficult to train and ended up with much higher prediction errors than either echoing or normal networks.

These results show that the networks cannot exploit any correlation that may exist between the daily sunspot number and the daily geomagnetic activity. In fact, the inclusion of the sunspot number inputs only seems to hinder the training of the networks, preventing them from even reaching the accuracy of an echo.

6.3.2 Yearly Data

12-12-1 networks were used for the yearly dual input predictions, with 6 inputs from one yearly time series and 6 inputs from the other. Figure 6.18 shows the results of using sunspot number and solar flux in predicting one other. As only the data after 1947 can be used to train the sunspot number/solar flux dual input network, it is not fair to compare its results in predicting sunspot number with the best network of Section 5.6, which was trained on the sunspot number as far back as 1850. For this reason a 6-12-1 single data input network was trained on the same training set as the dual input network. The RMS errors of predicting sunspot number using dual data inputs is 27.8 whereas the single data input network achieved a considerably lower error of 19.3. For solar flux the dual data input network's error was 25.3, whilst the previous best network (see Section 5.9) achieved an error of 16.0. As with the daily time series, the training of these networks was quite unstable, requiring learning rates as low as 0.0001 to achieve the end results. With this value of learning rate, over 50,000 iterations were performed to bring the networks to the point where over-fitting forced training to be halted. The time-shift plots for these predictions are not shown here because there was no delay present in either the training set or the test set.

In Section 1.8 a method was reviewed, Schatten and Sofia (1987), that used the level of geomagnetic activity at solar minimum to predict the subsequent solar maximum. The physical reasoning here is that the amount of geomagnetic disturbance at solar minimum would reflect the strength of the Sun's newly formed poloidal field, which in turn would indicate the level of activity at the next maximum. For these reasons I have investigated the prospect of using the dual inputs of yearly K_p index and yearly sunspot number, in predicting the latter a year in advance. The same 12-12-1 as used above will give the network access to the K_p index at the last solar minimum, when it has to predict the maximum of a cycle. The results are plotted in Figure 6.19, and again the time shift plot was omitted as their is no delay in the predictions. The error of these predictions is 22.0. which although better than using the sunspot number/solar flux dual inputs, is still nowhere near as good as the single data input network.



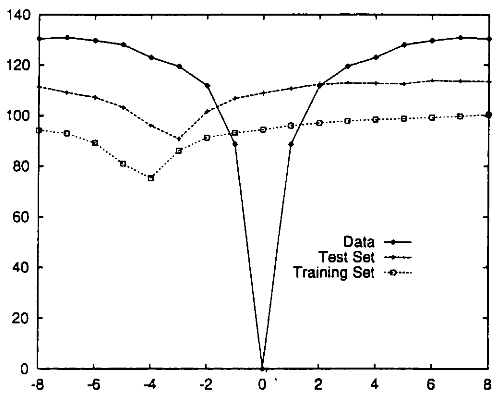
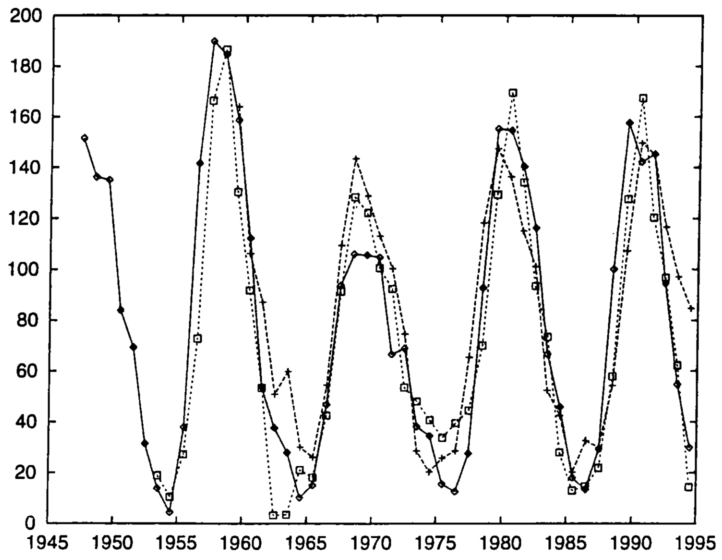
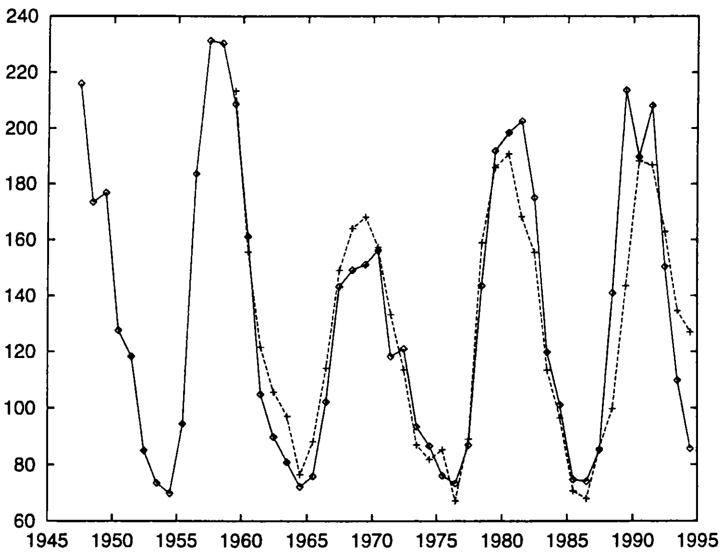


Figure 6.17: The prediction plot and time shift plot for a 4-4-1 network predicting the daily K_p index, by taking the last two days of both sunspot number and K_p index as inputs.



(a)



(b)

Figure 6.18: The prediction plots for a) a 12-12-1 network predicting the yearly sunspot number (crosses) with the predictions of the single data input 6-12-1 network (boxes) and b) a 12-12-1 network predicting the yearly solar flux. In both cases the network takes the last six years of both sunspot number and solar flux as inputs. The training set for these networks was 1947 to 1980, with a test set from 1981 to 1994.

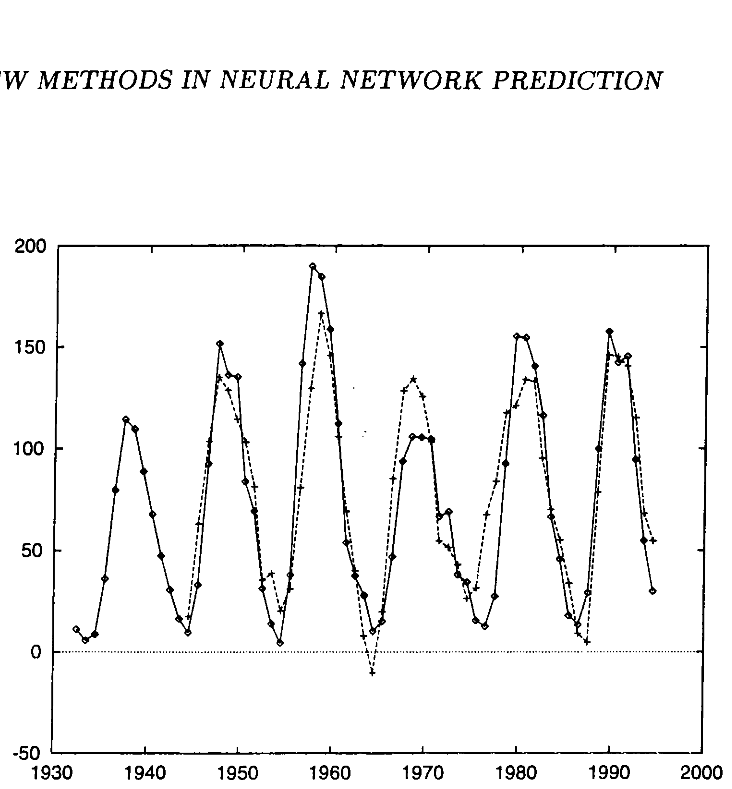


Figure 6.19: The prediction plot for a 12-12-1 network predicting the yearly sunspot number a year in advance. In both cases the network takes the last six years of both sunspot number and K_p index as inputs. The training set for this network was 1932 to 1980, with a test set from 1981 to 1994.

6.4 Concluding Remarks

Of the three new methods that were introduced and tested in this chapter, wavelet filtered prediction and the genetic algorithm training scheme have shown the most promise. The third, dual data input networks, failed to reach even the same level of success as the networks studied in chapter 5.

The test set RMS error for 6 month ahead predictions of monthly sunspot number using the wavelet filtered prediction method, was 27.5, as opposed to 24.5, achieved by the 18-6-1 network in Chapter 5 (on the same test set). Although the WFP error is higher, like is not being compared with like, because the 18-6-1 network was the product of an involved search of network architectures. The WFP method has one very attractive prospect: it discourages the network from trying to fit any short time scale noise. This is an advantage because the network is less prone to over-fitting the training set data. Another advantage of WFP, that has already been remarked upon, is the fact that the predictions have a clearer meaning than the predictions of a running mean smoothed time series. This point is discussed in more detail in Section 7.3.2.

The GA procedure, although successful in the case of smoothed sunspot number, could not produce any networks that predicted the K_p index time series without delay. One apparent problem lay in the over-fitting of the training set data - overfitting being perhaps the easiest way for the GA to find a network that produced non-delayed predictions of the training set. Although it might seem tempting to help the GA along by penalising networks for bad-generalisation to the test set, this would compromise the requirement that the test set must be "unseen" during training.

The reason for the failure of dual input networks is not clear, though the fact that training was unstable might provide a clue. If, for example, it is difficult for a network to ignore one of its inputs, that is, if it is difficult for BP to train a input's weight connections towards zero, then irrelevant inputs might actually hinder the process of training. The further implications of this are discussed in Section 7.5.

The search for improved methods of prediction, whether it is in using neural networks or not, can be very unrewarding, as seemingly good ideas can turn out to be useless. Also, a prediction method judged to be successful on one subset of a given time series might turn out to be a failure on some other subset. The first problem is really a matter of being prepared to invest hard work in an idea which has no guarantee of success. The second demands some caution in announcing that a particular method is successful. For the results in this thesis, I have only inserted the word "significant" where I have had some reason to do so. For example, in saying that a network's test set RMS error is significantly lower than the echoing error, I have validated the use of the words "significantly lower" by calculating the two errors on more than one subset of the test set and checking that the statement is still true.

Chapter 7

Future Work and Conclusions

"Isn't (the) blurring frequently just what one needs?" *L. Wittgenstein*

In this chapter I shall draw together the results of the preceeding chapters to provide a more general, and digestible, summary of what has been achieved by the work of this thesis. In particular, the sheer volume of numerical results in Chapter 5, and the lack of any "theory" with which to compare them, makes their interpretation quite difficult. In this chapter I hope to focus some light on their meaning for the benefit of any future work. Also, throughout the thesis, but especially in the earlier chapters, I charted some theoretical ground by proposing new techniques, such as analytic training (Chapter 3), and reviewing some topics of time series and neural network theory (Chapter 1). This theory met with practice in only the simplest cases, for example in using analytic training to provide the prediction of a linear Auto-Regressive series in Chapter 4. In this chapter I hope to show how the gap between theory and practice can be narrowed by future work. Finally, I will discuss the new numerical techniques proposed and applied in this thesis, for example the employment of the wavelet transform in Chapters 2 and 6 and the use of a genetic algorithm as a training algorithm, also in Chapter 6.

7.1 Analysis of the Time Series

In Chapter 2 the three time series of interest were scrutinised in some detail, in terms of their periodicities and in terms of their auto- and cross-correlations. Sunspot number received special attention because, although over 200 years of data is available, the earlier data is of questionable quality. From about 1850 onwards the sunspot number is deemed to be reliable, e.g. Eddy (1977), before 1850 much of the data was reconstructed by Wolf. It was shown, in terms of properties of the solar cycle (e.g. duration, attack time, decay time etc.), that there was a statistically significant change in its behaviour in the middle of the 19th century. From the data alone there is no way of deciding whether this change is due to the reconstruction by Wolf, or due to some long-term change in the behaviour of the cycle. However, given access to Wolf's methods of reconstruction it may be possible to settle this dilemma. Whichever is the case, since it is clear that the sunspot number time series shows a significant change in behaviour, I decided not to use the the pre-1850 data for any subsequent work in this thesis.

Another issue, studied in Chapter 2, was the stationarity of the sunspot number, with particular attention paid to the possible existence of a "mean cycle". The statistical test constructed and applied found no evidence for a mean cycle in the sunspot number. In fact, the very difficulty in constructing a reliable test (e.g. because of problems in deciding the beginning and end points of a cycle) leads me to feel that no mean cycle exists. In a more practical vein, if the mean cycle leads to a workable prediction scheme, as demonstrated by McNish and Lincoln (1949), Macpherson (1994) and Kerridge et al. (1989), then that is justification in itself for assuming its existence.

7.2 Analytic Training

The method of analytic training, first proposed in Chapter 3, and generalised to deal with *I-H-O* FFNNs in Section A.4, is a technique that yields the network weights needed to fit *O* functions of *I* variables, provided that these functions are analytic in the region of interest. The fit is accomplished by means of a series truncation, and it is the error involved in this truncation that determines the error of fit. It was also shown that when fitting a finite polynomial, the error of fit can be made as small as desired by finding solutions where the input weights are made sufficiently small in absolute size.

7.2.1 Analytic training vs Back propagation

In an attempt to find a connection between theory and practice, I examined the weights of BP trained networks that were trained to fit low order polynomials (Section 3.2). These examinations revealed that back-propagation did not necessarily learn to fit the functions in the way prescribed by analytic training[1]. The exception to this last statement is when fitting a linear function. In

[1] In fact, both back-propagation and analytic training can find many different networks to perform the same task. Despite this however, there is still a clear difference between the results of BP and analytic training because the latter requires the trained network to have small input weights, while the former does not.

particular, it was shown that a 1-1-1 network being trained by BP to echo its input, attempted to proceed through weight space towards the solution prescribed by analytic training. The reason that I say it only "attempted" to do so is because instability in training occurred as the solution was approached. This result is important because, although the network was certainly able to produce an echo in theory, back-propagation was unable to realize this ability in practice. The very fact that networks can perform tasks outwith the reach of back-propagation is a good incentive to search for other training algorithms.

But does such instability completely explain why the end results of back-propagation and analytic training differ? The answer is no, because although the goals of the two methods may appear to be the same, they differ in a fundamental way: the former is discrete whilst the latter is continuous. Back propagation training must always work with a finite number of examples, meaning that the network will learn *any function* that represents the data set, and not necessarily *the function* that may underly the data. With analytic training the network's fit is continuous by design as it is required to fit the analytic function in the region of interest. This result is connected to generalisation ability because it highlights the fact that a BP trained network is under no obligation to learn the function that generated the data. The analytic training method is a step forward because it shows how to force a network to fit a given analytic function continuously over a given region; a task which is beyond the capability of back propagation.

7.2.2 Implications for Time Series Models

In relation to time series, the method of analytic training can be used to show which classes of time series can be well predicted with an *I-H-O* FFNN. For example, consider a non-linear auto-regressive time series of the form

$$X_t = f\left(X_{t-1}, X_{t-2}, \ldots, X_{t-I}\right) + a_t$$

where f is analytic over the range of the time series values. Given enough hidden neurons, f can be approximated to any desired degree of accuracy. Further, if f is a finite polynomial, then only a finite number of hidden neurons is needed to fit it to any desired degree of accuracy. On the other hand, the threshold auto-regressive time series (Section 1.2.5), where f is a discontinuous function would not, in general, be suited to prediction by an *I-H-O* FFNN. Although it has been suggested, with good reason, that recurrent networks are required to predict moving average type time series, e.g. Connor et al. (1994), there is a sub-class of linear moving average time series that are certainly amenable to prediction by FFNNs. These are the "invertible" MA time series. In fact, non-invertible MA series were rarely considered in the "classical" literature, e.g. see Cryer (1986) or Box and Jenkins (1970). The form for an N^{th} order MA time series is

$$X_t = \sum_{n=1}^{N} \theta_n a_{t-n} + a_t$$

and the series is referred to as being invertible if and only if the equation

$$1 - \sum_{n=1}^{N} \theta_n x^n = 0$$

has all its roots outwith the unit circle. If a series is invertible then it can be written in the form of an infinite order AR series:

$$X_t = \sum_{n=1}^{\infty} \phi_n Z_{t-n} + a_t$$

where the ϕ_ns can be expressed in terms of the θ_ns. This means that a FFNN, with a sufficient number of inputs, can predict an invertible MA series. Note that the variance of the noise, and the fall off of the ϕ_ns, determines how many inputs would be required to obtain a reasonable prediction accuracy. This reasoning could be extended to invertible non-linear MA models, though there exists no convenient technique for determining the invertibility of such time series. Though not discussed here, similar arguments might also be made for ARMA type series.

7.3 The Wavelet Transform

7.3.1 Time Series Filtering

In Section 2.4.3 the wavelet transform was employed to filter out unwanted short and long timescale variations in time series. In particular, the daily time series were filtered to highlight variations around the solar rotation period of 27 days. From these filtered versions of the time series "cleaner" correlograms were obtained and on the basis of a few numerical experiments I concluded that the wavelet transform strengthened the correlations that were present without introducing any spurious features into the correlogram. Obviously these numerical experiments can only be taken as being suggestive, not conclusive. To put such use of the wavelet transform on firmer ground a more in depth, theoretical analysis of its statistical properties is required. The Fourier transform is already supported by such results, some of which were put to good use in the analyses of Chapter 2. As an example of what is required, I show below how the expectation and variance of the wavelet transform of a white noise time series can be obtained.

Let the white noise time series f_t is defined such that

$$Cov[f_t, f_s] = E[f_t f_s] = \delta_{t,s}\,\sigma^2 \qquad E[f_t] = 0$$

where $\delta_{t,s}$ is the Kronecker delta function. Using a discrete time wavelet transform, described more fully in Chan (1995), the wavelet transform $\hat{f}(b, a)$ of f_t is

$$\hat{f}(b,a) = \sum_{t=t_1}^{t_2} \psi_{b,a}(t)\, f_t$$

where $\psi_{b,a}(t)$ is the wavelet function of translation b and dilation a. The mean of the wavelet transform is given by

$$\begin{aligned} E[\hat{f}(b,a)] &= E\left[\sum_{t=t_1}^{t_2} \psi_{b,a}(t)\, f_t\right] \\ &= \sum_{t=t_1}^{t_2} \psi_{b,a}(t)\, E[f_t] \\ &= 0 \end{aligned}$$

and the variance is

$$\begin{aligned} Var[\hat{f}(b,a)] &= E[\hat{f}^2(b,a)] \\ &= E\left[\sum_{t=t_1}^{t_2}\sum_{s=t_1}^{t_2} \psi_{b,a}(t)\,\psi_{b,a}(s)\, f_t\, f_s\right] \\ &= \sum_{t=t_1}^{t_2}\sum_{s=t_1}^{t_2} \psi_{b,a}(t)\,\psi_{b,a}(s)\, E[f_t f_s] \\ &= \sum_{t=t_1}^{t_2}\sum_{s=t_1}^{t_2} \psi_{b,a}(t)\,\psi_{b,a}(s)\,\delta_{t,s}\,\sigma^2 \\ &= \sigma^2 \sum_{t=t_1}^{t_2} \psi_{b,a}^2(t) \end{aligned}$$

So, in the region where the wavelet transform is meaningful, i.e. where the wavelet translation parameter b puts the wavelet far (in terms of the scale parameter a) from the data extremes t_1 and t_2, the variance only depends on a. (In other words, there is no dependence on b because the wavelet has the same square-sum, no matter where it is positioned with $[t_1, t_2]$.) The form of this dependence is determined by the choice of the mother wavelet function, and the normalisation of the basis wavelets.

7.3.2 The Wavelet Filtered Prediction Method

In the wavelet filtered prediction (WFP) method, introduced in Section 6.1, a FFNN is expected to predict wavelet filtered time series, given inputs from the original time series. The idea underlying this approach is that, by separating the various time scales of the time series, it is easier (for both human and network) to identify the scales on which its behaviour is predictable. Almost instinctively, a human makes a prediction of a time series by mentally drawing a smoother curve through the points provided (the inputs) and extrapolating it. Exactly how the extrapolation is performed depends on the human's past observation of the time series. With reference to this analogy, the smooth line in WFP is drawn by the wavelet transform, in the hope of encouraging the network to ignore unpredictable variations on smaller (or longer) time scales. The method was applied with some success to the prediction of monthly sunspot number, though it remains for future work to discover how useful the method actually is at predicting other time series. Another problem left open to following work is in making use of the wavelet filtered time series as inputs. Doing so is difficult at present due to the end effects of the wavelet transform. One way of avoiding this problem is to make the network predict further into the future, over the troublesome end effects.

The success of the WFP method, as demonstrated here, is two-fold. First of all, it has been established that the basic method is workable in practice and secondly, the WFP method produces predictions which are more meaningful that the predictions of running mean smoothed time series. Exactly what the phrase "more meaningful" means is rather vague at present, but it must surely be possible to bring rigour to this method by considering the statistical properties of the wavelet transform.

So, to re-state the common bottom-line from both of these wavelet transform applications: *in order for wavelet filtering techniques to become more useful and meaningful in the analysis and prediction of time series, a better understanding must be sought of the statistical properties of the wavelet transform of stochasticly generated data.* By stochasticly generated data, I mean a set of (possibly dependent) random variables, that have many possible realizations.

7.4 Delay

When I first found the problem of delay, it was not clear whether it was a problem at all - after all, did it matter that the prediction curve best matched the time series when slid backwards, as long as the actual prediction error itself was low? This question was clarified by the end of Chapter 4 where it was seen that even the best prediction methods for some time series produced delayed predictions. The reason that these "best" methods suffered from delay was because they were not required to do anything more than minimise the RMS prediction error. So to answer the posed question: yes and no. Yes, it does matter, if it is of importance to predict *when* a particular event will occur, such as a peak in solar or geomagnetic activity. On the other hand, if the goal is to achieve the smallest RMS error, then delay does not matter. I take the former view, that timed advance warning of specific events should be most important.

7.4.1 The Genetic Algorithm Training Scheme

In Chapter 6 a genetic algorithm training scheme was constructed, tested and applied. In this scheme the networks that predicted with delay were penalised so that they would be "bred" out of the population. For the K_p index predictions, no networks were found that predicted without delay. For the smoothed monthly solar flux time series, however, a network was found that could predict six months ahead without delay. The fact that the method failed on the smoothed monthly K_p index whilst it succeeded on the smoothed monthly solar flux could be a statement about the relative predictability of these two series.

The full potential of the genetic algorithm training scheme could not be fully explored here, because the computer time required to train networks in this way (at least on the computers currently available to me) is enormous.

7.4.2 Sliced Prediction

The genetic algorithm scheme tackled the problem of delayed prediction head on, by re-defining what was meant by "good" predictions. Another, more subtle, approach which I refer to as "sliced prediction" is suggested here as a starting point for future work.

Consider an I-H-1 network that is given I inputs from the recent history of a particular time series in the usual manner. Instead of predicting the value of the next element of the time series, the goal is to simply to predict whether the following value is going to be higher or lower than, or about the same as the current one. This can be achieved, for example, by compiling a training set of examples where, the desired output is 1 if the following value is larger, or -1 if the following value is smaller. After training, being careful to avoid overfitting, the hope is that the network will output a value near 1 when the next value is larger and a value near -1 when the next value is smaller. The meaning of values near zero is not clear, and would need be determined by examining test set data.

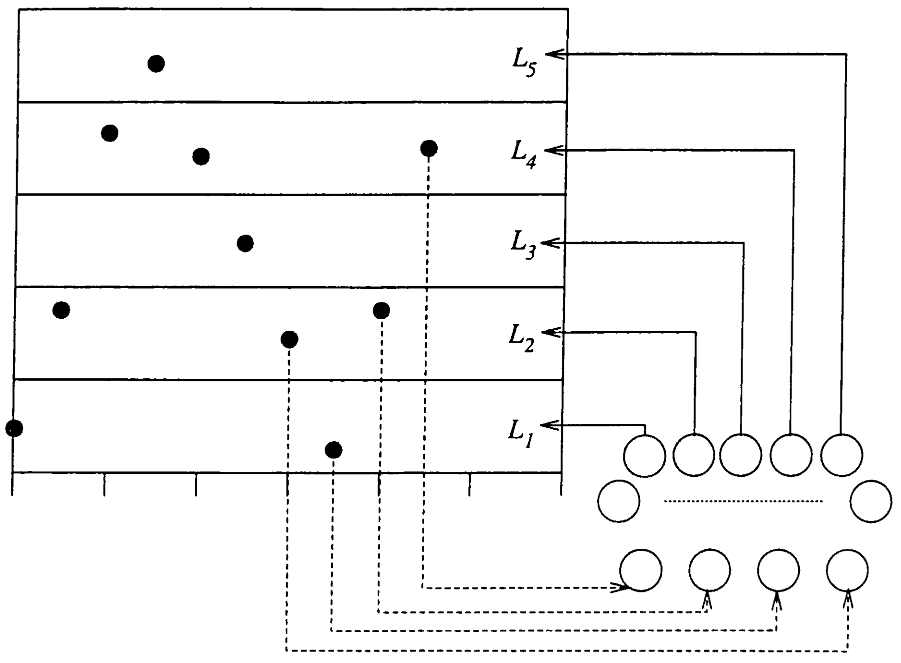


Figure 7.1: An illustration of sliced prediction. A network takes 4 values from a time series and outputs five values, $L_1, \ldots L_5$, which are the "likelihoods" of the next time series value occurring in the indicated bands or slices. The band with the highest likelihood can be taken to be a prediction, with the width of the band indicating the prediction accuracy.

They would presumably mean something like "don't know" or "next value will be similar to current value". If more predictive information is desired, then the scheme can be extended. For example, an *I-H*-2 network could be designed to output two numbers. The two numbers being measures of how likely it will be that the next value will be greater or smaller than the current one. This could be taken further with an *I-H-O* network predicting the likelihoods of the next value falling in *O* bands or slices. This is explained graphically in Figure 7.1.

Further variations of the scheme could supply the smoothed or even the differenced time series as inputs. There might even be some advantage in using the outputs of the *I-H*-2 network as inputs to an *I-H-O* network, or even building a chain of networks to perform successively finer sliced prediction on a time series. Another possibility is to use the outputs of sliced prediction networks as inputs to the conventional networks.

Sliced prediction has a capability possessed by no other prediction method met in this thesis. It is capable of predicting several values that might come next, and by the same token it is also capable of admitting that it cannot predict what comes next. When using a FFNN in the usual manner, even if the future is completely unpredictable, the network must make a prediction. A FFNN employed in the sliced prediction method might well be more honest in the same situation. There is a price to be paid for this, in that the banding of the sliced predictions effectively puts a lower limit on the prediction error, the lower limit corresponding to the width of the bands.

The reason that sliced prediction might avoid delay is that the time series is predicted with the emphasis on direction rather than magnitude. For example, in order to predict when a maximum is reached, it is important to be able to predict that the point following the maximum is going to be less than the maximum. Failure to do so results in delayed predictions. With sliced prediction, the aim is to predict the shape, and not the values, of things to come - *cf.* Wells (1932).

7.5 Dual Input Data Networks

A dual data input network was one that was required to take inputs from two different time series and predict just one of them. The results of Section 6.3 demonstrated that these networks could not even be trained to the same level of success achieved by the conventional networks in Chapter 5. In other words, the inclusion of extra inputs seemed to hinder the training of the networks. This is also confirmed by the the fact that the training of dual data input networks was quite unstable. If it is really the case that BP has difficulty in training networks to ignore inputs, then this a serious flaw in the algorithm. The most obvious way for BP to make a network ignore an input is for it to train that input's weight connections towards zero[2]. As seen in Chapter 4, the training of a (simple) network's input weight toward zero caused problems for BP.

7.6 The Prediction of Solar-Terrestrial Time Series

Lastly, I shall summarize the results of Chapter 5, where networks were used to predict the solar-terrestrial time series. First of all, a search of network architectures was performed for each of the fifteen time series: sunspot number, solar flux and K_p index, each in five formats: daily, smoothed daily, monthly, smoothed monthly and yearly. Then the most successful networks from this search were trained further to see if any improvement was possible. After this the possibilities of probing deeper into the future were investigated by iterating networks on their own predictions and by training networks to predict 6 steps ahead. For all predictions, uncertainties of prediction and levels

[2]The layered structure of a network means that there are also other, less obvious, possibilities

of bias were estimated from residual histograms and tabulated in Tables 5.20, 5.21 and 5.22. Finally some attention was paid to the choice of training set.

The first thing to note is that no network produced predictions that were simply an echo - in all cases the networks equalled, or more usually bettered, the echo error and produced some kind of meaningful predictions. It would also be quite reasonable to say that the sunspot number and solar flux time series were easier to predict than the geomagnetic K_p index time series on all timescales. To see this, simply compare the prediction plots of these series.

The best networks for predicting each time series 1 step ahead were found and trained until no further improvement was possible with back-propagation. It was noted in Section 5.10 that the daily, smoothed daily and yearly series seemed to require networks with only 2 or perhaps 6 hidden neurons, whereas the monthly and smoothed monthly series required typically 12 or 18 hidden neurons. Some reservations in reading too much into these results were expressed in the same Section. Nonetheless, it was still possible to show that the networks predicting the daily and smoothed daily series were performing some simple linear function on their inputs.

Attempts were then made to predict further into the future. The first method tried, that of freely iterated prediction, turned out to be very unreliable. The second method tried used the the best 1 step ahead networks iterating 6 steps ahead - this achieved a reasonable prediction accuracy in most cases. Finally, networks were trained to predict 6 steps ahead directly - this method turned out to be the best for all time series but the yearly ones. The architectures of these 6 step ahead predicting networks were chosen to be the same as the best networks for predicting 1 step ahead. However, it was reasoned that the six year ahead predictions might require a larger network than the 12-2-1 used for 1 step ahead predictions. By training a 12-18-1 network, it was then shown that yearly sunspot number could be directly predicted six years in advance with a better accuracy than iterated prediction.

The effect of using different training sets was discussed in Section 5.9. In particular it was shown that the largest solar maximum in history, that of cycle 19, must really be included in the training set when training a network to predict either sunspot number or solar flux on the timescale of years.

Finally, in order to estimate the accuracy of a prediction before the true value is known, the residuals of prediction were analysed. For each time series and for each of the prediction methods (i.e. 1 step, 6 step iterated and direct 6 step) estimated uncertainties and biases of prediction were obtained and tabulated in Tables 5.20, 5.21 and 5.22.

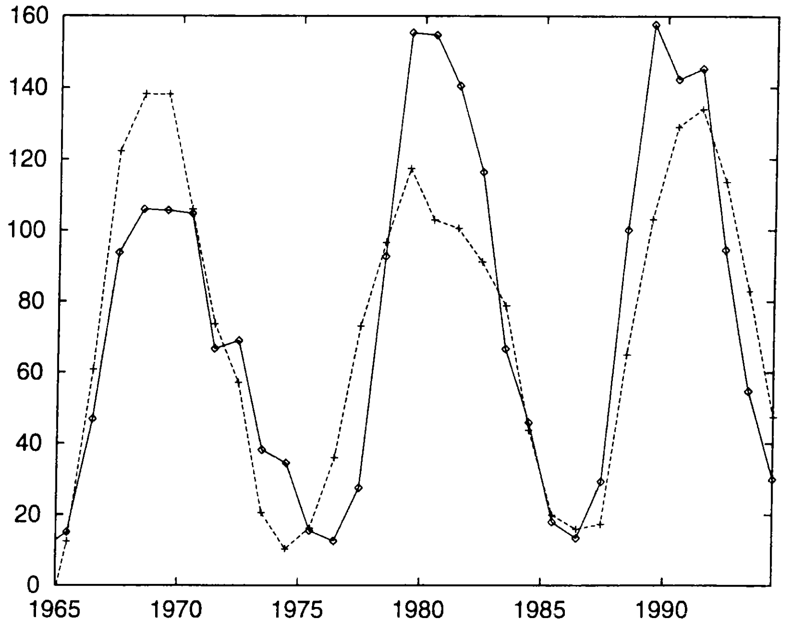


Figure 7.2: The 12-18-1 network's direct 6 year ahead predictions on its test set.

7.7 The Maximum of Cycle 23

At the time of writing (16/10/95), solar cycle 23 has either begun or is just about to begin. Going by recent cycles, this means that the maximum of cycle 23 should occur sometime between 1998 and 2002. Although some of the freely iterated predictions of Chapter 5 did predict a maximum occurring as late as 2001 or as early as 1998, the unreliability of freely iterated prediction really relegates such predictions almost into the realm of amusement. So, since the next maximum may still be six years away, the only network in contention for its prediction is the 12-18-1 network, that directly predicts 6 steps ahead. From Table 5.22 this network predicts values around solar maximum to within ± 50 spots 85% of the time without any perceptible bias. Figure 7.2 shows this network's performance on its test set. Note that the time of maxima for cycles 20 and 21 are predicted correctly, whereas the time of maximum for cycle 22 is two years late (though the prediction does coincide with the secondary maximum of cycle 22). The maximum values of cycles 20, 21 and 22 are predicted with errors 34, -38 and -11 respectively.

Figure 7.3 shows the network's predictions of cycle 23. The maximum is predicted to occur in the year 2000, with a value of 131. As a bonus cycle 24 is predicted (by iteration) to occur in 2011

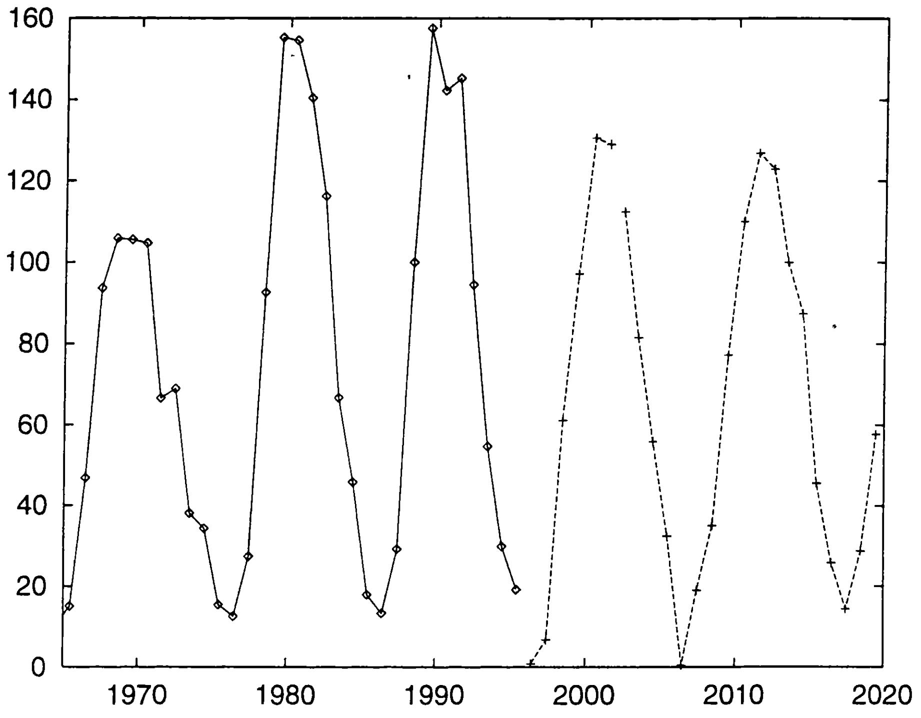


Figure 7.3: The 12-18-1 network's direct 6 year ahead prediction of the maximum of cycle 23. The gap in the predictions (the dotted line) corresponds to the onset of iteration. All predictions up to and including 2001 are *not* iterated. To push the prediction as far into the future as possible I have calculated 1995's yearly sunspot number using all the monthly values available at the time of writing (January to September).

with a maximum value of 127.

Appendix A

Miscellaneous Results

This appendix contains work that went into the methods described in various parts of this thesis. Usually results that required a good deal of effort and time on my part are glibly stated because inclusion of the full detail would obscure the main issue. So for that reason and to provide a complete documentation of the methods that I have used I present the following sections.

A.1 Formatting of data

In Section 2.1 various data were presented in monthly, smoothed monthly and yearly formats, with a few extracts from the daily data. The data ranges used are listed in Table A.1.

This data was obtained by FTP from ftp.ngdc.noaa.gov which is the electronic address of the National Geophysical Data Center, Boulder, Colorado, U.S.A., part of the National Oceanographic and Atmospheric Administration, N.O.A.A. The data file was formatted with each day occupying a separate row, with the various indices occupying different columns. I then split these records into several files, having a separate file for each index. The format of these files was

$$< \text{decimal data} > \quad < \text{time series value} >$$

Data	Start Date	End Data
Sunspot Number	January 1st 1850	May 30th 1995
10.7cm Solar Flux	February 1st 1947	May 30th 1995
K_p Geomagnetic Index	January 1st 1933	May 30th 1995

Table A.1: Ranges for each of the main time series

At this stage I checked the data for missing days or mis-read values. I did this by looking at plots of the data and by counting the number of days since the data record began (there are 53,110 days from January 1st, 1850 to May 30th 1995). The solar flux was the only index for which data were missing for various days. These days were usually weekends and holidays in the 1940s, 1950s and early 1960s. There was also one day, in 1992, for which a line was duplicated in the original data file. Once such irregularities had been removed the monthly versions of the data were constructed, allowing for the missing days in the case of solar flux. Calendar months were used mainly for the sake of convenience in dealing with the time series. From the monthly time series the yearly time series were simply constructed taking the average of the twelve months of the year.

In the earlier stages of work the data was procured from the *World Data Centre A* at the Rutherford-Appleton Lab. in Oxfordshire, England, through a rather more involved electronic retrieval procedure than FTP. This data ran up until April 1994 but in the years between 1992 and 1994 there were marked differences in the K_p index recorded here and the K_p index recorded in the NGDC records. I later read that some of the data released by the World Data Centres in the early 1990s contained errors due to the change-over in processing software at the time. For this reason I chose to accept the NGDC record as being definitive. It is also worth noting that the most recent sunspot numbers in 1995 are still not considered final at the time of writing and may be subject to change. This is because the combination of each observatory's observed sunspot number is not finalised until months after the observations have been made.

A.2 Bin leakage in Power Spectra

In Section 2.2 data bins were summed in an attempt to smooth the spectrum. However, since the time series data were "windowed" before transformation (as all discrete data must be) there was some leakage between neighbouring bins in the spectrum. The question is, after the summing into larger bins how are the effects of this leakage changed. First of all, let all the leakage occur only into the bins on either side of the bin in question, this is a reasonable approximation for the Fourier transform of the Bartlett window, though not for the square window, which has a power spectrum that falls off much more slowly. Let p_i represent the i^{th} frequency bin of the true spectrum and let q_j represent the j^{th} frequency bin of the real-life discrete spectrum so that

$$p_i = \alpha q_i + \frac{1-\alpha}{2}(q_{i-1} + q_{i+1}) \tag{A.1}$$

where α defines the fraction of power retained by the central bin. Now for every even i define the summed re-binned spectra

$$r_{i/2} = p_i + p_{i-1}$$

(note that I only consider the case of 2 bin summation here). Substituting (A.1) into the above expression gives

$$\begin{aligned} r_{i/2} &= \alpha(q_i + q_{i+1}) + \frac{1-\alpha}{2}(q_{i-1} + q_{i+1} + q_i + q_{i+2}) \\ &= \frac{1+\alpha}{2}(q_i + q_{i+1}) + \frac{1-\alpha}{2}(q_{i-1} + q_{i+2}) \end{aligned}$$

The first term in the last line of the above is the contribution from the frequency bins of the true spectrum that are "intended" to be in bin $i/2$ and the other term represents the leaked contribution. So defining the leakage as the ratio of the coefficients of these two terms, the leakage is

$$L_{\mathrm{r}} = \frac{1-\alpha}{1+\alpha}$$

whereas the leakage in q from (A.1) is

$$L_{\mathrm{q}} = \frac{1-\alpha}{2\alpha}$$

Since $0 < \alpha < 1$, $L_{\mathrm{q}} < L_{\mathrm{r}}$, so that after re-binning in this way leakage becomes less important.

A.3 Permuting Groups

In Section 2.3.2 it is necessary to find all the permutations of selecting two subsets of six from a set of 12 without replacement and without caring about the order of the elements in subsets. It can easily be seen that the number of different pairs of subsets is

$$\frac{1}{2}\,{}_{12}C_6 = \frac{1}{2}\frac{12!}{6!\,6!} = 462$$

where the half arises from the fact that choosing the elements of one group decides all the elements of the other. Listing these subsets is a much more complicated matter but can be accomplished by taking the whole set, splitting it into two halves and swapping elements between halves. So using the standard hexadecimal numerals, the 12 elements of the whole set are:

$$\{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ \mathrm{A}\ \mathrm{B}\ \mathrm{C}\}$$

To facilitate the later discussion I use the following notation to denote the swapping procedure:

$$\{(0\ 1\ 0\ 0\ 0\ 0\ ,\ 0\ 0\ 0\ 1\ 0\ 0)\}$$

which means swap the second and tenth elements of the whole set to form the two subsets

$$\{(1\ \mathrm{A}\ 3\ 4\ 5\ 6\ ,\ 7\ 8\ 9\ 2\ \mathrm{B}\ \mathrm{C})\}$$

similarly

$$\{(0\ 1\ 0\ 1\ 0\ 0\ ,\ 0\ 1\ 0\ 1\ 0\ 0)\}$$

is a prescription to swap the second and eighth elements and the fourth and tenth. There is no ambiguity because swapping the second and tenth and the fourth and eighth result in the same subsets since order is unimportant. I shall also use the notation $(n \rightarrow)$ to denote a wrap shifting operation on a set that produces another set composed of the $1, 2, \ldots, n$ right-shifted versions of the original set, e.g.

$$\{0\ 1\ 0\ 0\ 1\ 0\ (3 \rightarrow)\}$$

is a shorthand for the set

$$\{0\ 1\ 0\ 0\ 1\ 0\ ,0\ 0\ 1\ 0\ 0\ 1\ ,1\ 0\ 0\ 1\ 0\ 0\ \}$$

Finally, I use the symbol $\otimes$ to denote the set multiplication operation, so that $A \otimes B$ represents a set containing every pairing of elements in set A with elements in set B.

To generate all 462 different groups there must be 462 distinct swapping operations expressed as the 0,1 strings above. Swapping one element at a time there $\frac{1}{2}({}_6C_1)^2$ distinct subsets possible, swapping two there are $\frac{1}{2}({}_6C_2)^2$ distinct subsets and so on. So it can be seen that

$$\text{Number of subsets } = \frac{1}{2}\sum_{i=0}^{6}({}_6C_i)^2 = \sum_{i=0}^{2}({}_6C_i)^2 + \frac{1}{2}({}_6C_3)^2$$

It is not necessary to swap more than three elements at a time because every subset produced by swapping four or five elements simultaneously can be achieved by swapping one or two elements simultaneously, reflected in the fact that ${}_6C_i ={}_6 C_{6-i}$. So if 36 distinct one element swaps are found and 225 distinct two element swaps are found and 200 distinct three element swaps are found, then 461 distinct subsets have been found, where the remaining is just the original set split down the middle.

The one element swaps are easy to find, they are given by product of the set

$$A_1 = \{1\ 0\ 0\ 0\ 0\ 0\ (5 \rightarrow)\}$$

with itself, i.e. $A_1 \otimes A_1$. The two element swaps will all be distinct from the one element swaps and they are the product of

$$A_2 = \{1\ 1\ 0\ 0\ 0\ 0\ (5 \rightarrow),\ 1\ 0\ 1\ 0\ 0\ 0\ (5 \rightarrow),\ 1\ 0\ 0\ 1\ 0\ 0\ (2 \rightarrow)\}$$

with itself i.e. $A_2 \otimes A_2$, where the last element of the set is only shifted twice because it becomes itself again after 3 shifts. The three element swaps, again automatically distinct from the one and two element swaps, are given by the product of

$$A_3 = \{1\;1\;1\;0\;0\;0\;(2 \rightarrow),\;1\;1\;0\;1\;0\;0\;(5 \rightarrow),\;1\;0\;1\;0\;1\;0\}$$

with $(A_3 + \bar{A}_3)$, where the bar means that each element of A_3 should be inverted, e.g.

$$1\;0\;0\;1\;0\;0\text{ becomes }0\;1\;1\;0\;1\;1$$

That is the set of three element swapping operations is $A_3 \otimes (A_3 + \bar{A}_3)$. And thus all possible subsets can be found.

A.4 Analytic Training - The general case

Here I shall describe more fully the analytic training method that I first proposed in Chapter 3. In that chapter only the case of the 1-H-1 FFNN was addressed, and later in Chapter 4 it was shown how the method could be used to fit an I-H-1 network to a linear function. Now I shall show how the more general case of a I-H-O FFNN can be approached, where each output is required to fit a different multivariate polynomial.

Suppose that output k is required to fit the following function

$$f(x_1, x_2, \ldots, x_I) = \sum_{n=0}^{N} \sum_{i_1,\ldots,i_n} f^n_{i_1,\ldots,i_n} \prod_{j=1}^{n} x_{i_j} \tag{A.2}$$

where the summation symbol $\sum_{i_1,\ldots,i_n}$ is understood to mean a summation in which every $\prod_{j=1}^{n} x_{i_j}$ term appears only once. The k^{th} output of the network is given by

$$C_k(x_1, x_2, \ldots, x_I) = \sum_{j=1}^{H} W_{kj} g\left(\sum_{i=1}^{I} w_{ji} x_i\right) \tag{A.3}$$

and the multi-dimensional Taylor expansion of $g\left(\sum_{i=1}^{I} w_{ji} x_i\right)$ in the I $w_{ji}x_i$ terms about 0 is

$$g\left(\sum_{i=1}^{I} w_{ji} x_i\right) = \sum_{n=0}^{N} \sum_{i_1,\ldots,i_n} g^n_{i_1,\ldots,i_n} \prod_{l=1}^{n} w_{ji_l} x_{i_l} \tag{A.4}$$

where the $g^n_{i_1,\ldots,i_n}$ coefficients can be written as

$$g^n_{i_1,\ldots,i_n} = \frac{1}{n_1! n_2! \ldots n_I!} g^{(n)}(0)$$

where $g^{(n)}(0)$ is the nth order derivative of the input activation function evaluated at 0, and the n_is are just the powers present in the $\prod_{l=1}^{n} x_{i_l}$ term, i.e.

$$\prod_{l=1}^{n} x_{i_l} \equiv \prod_{j=1}^{I} x_j^{n_j}$$

So substituting (A.4) into (A.3) yields the following expression of the kth output of the network

$$\begin{aligned} C_k(x_1, x_2, \dots, x_I) &= \sum_{j=1}^{H} W_{kj} \sum_{n=0}^{N} \sum_{i_1,\dots,i_n} g^n_{i_1,\dots,i_n} \prod_{l=1}^{n} w_{ji_l} x_{i_l} \\ &= \sum_{n=0}^{N} \sum_{i_1,\dots,i_n} g^n_{i_1,\dots,i_n} \left(\sum_{j=1}^{H} W_{kj} \prod_{l=1}^{n} w_{ji_l} \right) \prod_{l=1}^{n} x_{i_l} \end{aligned}$$

Equating the coefficient of each $\prod_{l=1}^{n} x_{i_l}$ term with the corresponding term in the expression for $f(x_1, x_2, \dots, x_I)$ in (A.2) gives rise to the following equations that need to be satisfied

$$g^n_{i_1,\dots,i_n} \left(\sum_{j=1}^{H} W_{kj} \prod_{l=1}^{n} w_{ji_l} \right) = f^n_{i_1,\dots,i_n} \qquad n = 0, 1, \dots, N$$

the remaining terms for $n > N$ are responsible for the error of this fit.

The procedure of analytic training proceeds as follows

1. Choose a set of small input weights, w_{ji_l}
2. Solve the following set of linear equations for the output weights:

$$\left(\sum_{j=1}^{H} W_{kj} \prod_{l=1}^{n} w_{ji_l} \right) = A^n_{i_1,\dots,i_n} \qquad n = 0, 1, \dots, N$$

   where

$$A^n_{i_1,\dots,i_n} = \frac{f^n_{i_1,\dots,i_n}}{g^n_{i_1,\dots,i_n}}$$

3. If the fit is not accurate enough, then return to 1) but choose the input weights so that they are all smaller in absolute value, i.e. $|w_{ji_l}^{\text{new}}| < |w_{ji_l}^{\text{old}}|$

Remarks:

(i) Fitting a multivariate polynomial of order N, in I variables, means that there will be

$$H_0 = \sum_{n=0}^{N} \frac{1}{(I-1)!} \prod_{i=1}^{I-1} (n+i)$$

linear equations to be solved. So to guarantee a solution there must be at least as many hidden neurons as equations to be solved, i.e. $H \geq H_0$. Note though, since many of these equations might not be completely independent of one another there may exist solutions that require less hidden neurons that H_0. A simple linear example of this can be found in Section 4.4.

(ii) When there are many outputs required to fit several polynomials of different orders, the number of neurons should be chosen according to the highest order polynomial to be fitted. For the remaining outputs there will then be more unknowns (i.e. output weights W_{kj}), than equations, so for these outputs the excess weight connections can be set to zero.

(iii) The proof that reducing the magnitudes of the input weights reduces the error of fit is not given here, but the following sketches why the result is plausible. Consider the following set of equations

$$\begin{aligned} W_1 + W_2 + W_3 &= A_0 \\ w_1 W_1 + w_2 W_2 + w_3 W_3 &= A_1 \\ w_1^2 W_1 + w_2^2 W_2 + w_3^2 W_3 &= A_2 \end{aligned}$$

Assuming that a solution can be found for the Ws, each W_i will contain no terms of the order $1/w_i^n$, where $n > 2$. This means that

$$W_i w_i^n \rightarrow 0 \qquad \text{for } n > 2$$

(iv) In performing the solution to the linear equations numerically the small values of input weights give rise to very large values in the output weights. This often causes problems with rounding error and smallest/largest number representations on a computer, even if double precision (64 bit representation) is used.

Appendix B

The Software

This appendix describes one of the 272 different programs that I have written to obtain the results contained within this thesis. The program described here is a typical example of one which implemented the various neural network algorithms. It may not be a shining example of how to program in C (partly because I was learning C when I started writing it and partly because it has evolved as a kind of patch work over three years) but it proved flexible enough to allow easy adaptation to a number of other tasks, such as the wavelet prediction method of Chapter 6. I have not included the input and output routines, as their function is not crucial to understanding the implementation of the neural network algorithms.

The program takes only one argument, the name of what I call the key-file, the file which specifies the network architecture, the training method and parameters, the training data files, output files etc..... This particular incarnation of the program was designed to take inputs from two time series and predict one of them. The number taken from each being determined by NISPLIT. If NISPLIT > NI then inputs are only taken from the time series to be predicted.

In all the neural network programs that I have used, I scale the time series into 0.2 to 0.8 at the outset. This is a necessity if using a *sigmoidal* output activation function because a sigmoid only has a range of $[0, 1]$, where infinite weights are required to output either 0 or 1. However, all the work of this thesis used *linear* output activation functions, so such scaling was performed for the sake of consistency rather than out of necessity.

The training algorithm used is determined by the contents of the key-file, as are the training parameters. If the number of genetic cycles given in the key-file is 0 or 1 then normal back-propagation is performed. If the number of genetic cycles is specified to be greater than 1 then the genetic algorithm training is performed. The method of mutation can be set to random, backpropagation or both, with further parameters specifying the amount of mutation desired. This version of the program was set up to perform genetic algorithm training on a network with only one output predicting at nskip+1 steps ahead. If back-propagation was selected then there can be many outputs, the i^{th} output predicting at $i + 1$ steps ahead.

B.1 The main program

```
#include <stdio.h> /* Various standard C header files ...*/
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "neurine.h" /* ... and my own header file */
```

Almost all the sub-routines used in this programs are declared in the header file "neurine.h", along with all the #defines used in the program. The #define's are:

```
#define NL 3              /* Number of layers */
#define NI nu[0]          /* Number of inputs */
#define NO nu[NL-1]       /* Number of outputs */
#define NISPLIT nisplit /* Where the inputs are split */
#define NUMAX 37          /* Maximum number of neurons per layer */
#define NP np             /* Number of pattern elements */
#define NS ns             /* Number of NNs in genetic population */
#define NSMAX 1           /* Maximum number of NNs in genetic population */
#define NT 20000          /* Max number of data points in time series */
#define BETA 0.5          /* Steepness parameter of the activation function */
#define MAXLINE 1000      /* Max. length of input file text line */
#define CHILD child       /* Number of parents !!!!!!!! */
#define VERSION1 2        /* Version Number */
#define VERSION2 0
#ifndef RAND_MAX          /* Max random number for rand() */
  #define RAND_MAX 2147483647 /* Needed for gcc 2.5.6 for SUNOSIV */
#endif
#define INPUT_FILE infile /* Input file names */
#define INPUT_FILE2 infile2

void zero_odw(); /* Sets last weight changes to zero */

int readkey(char keyfile[],char infile[],char infile2[],char inwfname[],
char outwfname[],double *e1,double *e2,double *de,
int *ndisp,int *ncost, int *imut,
double *mutsize, double *mutprob); /* Reads in the NN specifications */

double prms(FILE *fp); /* Calculates error on the test set */
double trms(FILE *fp); /* Calculates error on the training set */
```

Below are the various global variables used in the program. Unfortunately, there are very many of them and a lot of memory is gobbled up needlessly by the multidimensional weight arrays like w[][][][]. This was because during the start of my Ph.D., when I wrote the basic form of this program, I did not understand the concept of pointers in C. Once I did know how to use pointers, I did not re-write the program because it would have been a mammoth undertaking for which I did not have the time. The most important global variables are:

- train - Number of training iterations
- nu[] - array containing the no. of neurons in each layer
- is - index referring to the current NN in the genetic population
- ncyc - number of cycles to be performed in the Genetic Algorithm
- xi[] - first time series
- xi2[] - second time series
- dxi[] - array containing dates
- v[i][n] - Value of i^{th} neuron in layer n
- w[i][j][n][k] - The value of the weight connection between the j^{th} neuron in layer $n-1$ and the i^{th} neuron in layer n, of the k^{th} NN in the genetic population
- dw[i][j][n][k] - The change to be applied to that weight
- odw[i][j][n][k] - The last change applied to that weight
- iw[i][j][n][k] - The initial weight
- delta[][] - The deltas in the back-prop algorithm
- out[i] - Desired value for output i of current pattern
- ep,ap - The learning rate and momentum
- min,max,min2,max2 - The minimum and maximum values of the two time series

```
time_t T1;
int train,nt,NP,nu[NL],timgra,is,nisplit,start_cyc,ncyc,imut,ns,
    child,smoothing,sampling,linflag,ant[NCHUNKS],finflag,nskip;
double xi[NT],xi2[NT],dxi[NT],v[NUMAX][NL],
      w[NUMAX][NUMAX][NL][NSMAX],dw[NUMAX][NUMAX][NL],out[NUMAX],fltnp;
double ap,be,ep,h[NUMAX][NL],odw[NUMAX][NUMAX][NL],
        iw[NUMAX][NUMAX][NL][NSMAX],delta[NUMAX][NL],min,max,min2,max2;
char infile[300],infile2[300],datatype[100];
```

B.1.1 main

The "main" routine is the first routine called when a C program is executed. The layout of the main program is as follows:

- Initialise the various variables, e.g. random starting points for the network(s) to be trained.
- Start genetic cycle loop

  Mutate the children (BP or random)

  Evaluate each member of the population's success (evaluate)

  Sort according to success (sort)

  Perform cross-over (breed)
- Write the best network's weights to a file

```
main(int argc, char *argv[])
{
double e1,e2,d3,e[NSMAX],mutsize,mutprob,tmp,oldprms;
int i,t=0,k=0,idisp,ndisp,ncost,pe[NSMAX],cycle,pflag=0;
char r[300],s[300],outfile[300],inwfname[300],tmpstring[MAXLINE],
     outwfname[300];
FILE *wfp;

ns=50;
child=6;
start_cyc=0;
if(argc!=2) /* Error in program arguments */
{
  printf("Usage: %s <keyfile>\n",argv[0]);
  exit(1);
}
T1=time(NULL); /* Keep start time */
fprintf(stderr,"train version %d.%d\n",VERSION1,VERSION2);
```

```
if (readkey(argv[1],infile,infile2,inwfname,outwfname,&e1,&e2,&de,
            &ndisp,&ncost,&imut,&mutsize,&mutprob)==1)
{
  printf("Error reading input file\n");
  exit(2);
} /* End If scanf */

if ((nu[0]>NUMAX)||(nu[1]>NUMAX)||(nu[2]>NUMAX))
{
  printf("Too many neurons in a layer\n");
  exit(1);
} /* too many neurons */

if(ns>NSMAX)
{
  printf("Too many NNs  -  Max of %d NNs in a generation\n",NSMAX);
  return 0;
} /* ns */

ep=e1;
be=BETA;
if(ncyc==0 || ncyc==1) /* Set up variables for */
{                      /* Back-propagation */
  ncyc=ns=1;
  child=0;
  nskip=NO-1;
}
else                   /* And genetic cycle */
{
  nskip=nu[NL-1]-1;
  nu[NL-1]=1;
}
```

```
 if (inwfname[0]=='?') /* Random weights */
   wfp = NULL;
 else                   /* or */
 {                      /* Read from a file */
   if ( (wfp = fopen(inwfname,"r"))==NULL )
   {
      printf("Input weight file %s could not be opened\n",inwfname);
      exit(20);
   } /* wfp */
 } /* End if inwfname */
 iweights(wfp);          /* Get initial weights */
 if(wfp!=NULL)
   fclose(wfp);

 if (newset()==1) return 2; /* Read in time series data */

 NP = (int) ( (double) nt * fltnp );

 init();                 /* Set weights to initial values */

 for(i=0;i<NS;i++) pe[i] =i;
 for(cycle=start_cyc;cycle<ncyc;cycle++) /* Genetic cycle loop */
 {
   if((imut&2))               /* Back-Prop mutation if imut =2,3*/
   {
     for(i=child;i<NS;i++) /* Mutate CHILD strings */
     {
       is=pe[i];
       k=0;
       zero_odw(); /* zero old weight corrections */
```

Below are the two main loops for back-propagation - the outer loop is the training loop where k counts the number of times that the entire training set has been presented. The inner loop scans through the training set one step at a time, each time calculating and applying weight corrections from the B.P. algorithm.

```
      do
      {
        t=0;
        t=newpat(0);            /* Rewind through data set to first pattern */
        do
        {
          sweep();     /* Calculate neuron values throughout network */
          backprop(); /* Calculate weight corrections */
          apply();     /* Adjusts weights given the corrections above */
          t++;
          t=newpat(t);  /* Choose next pattern from xi[] */
        } while (t < (NP-NI-nskip) );
        k++;

        tmp=prms(NULL); /* Get test set RMS error */
        if( (k>100) && (ncyc==1) && (oldprms<(tmp)) ) /* Ensure prediction */
          pflag++;              /* error is still being reduced if BP only */

        oldprms = tmp;
        if(pflag>10) pflag=train=k; /* if not then end training */
      } while (k<train);
    } /* i - Child BP mutation loop */
  } /* End if imut */

  if(ncyc!=1) /* If performing the genetic algorithm */
  {
    evaluate(e); /* Evaluate the success of the networks */
    sort(e,pe);
    breed(pe,imut,mutsize,mutprob); /* Perform cross-over */
                            /* If imut is 1,3 random mutation also takes place */
  } /* If ncyc!=1 */
} /* cycle */
```

```
if(ncyc!=1)
{
    evaluate(e);
    sort(e,pe);
}
```

Finally print out best network

```
if ( (wfp = fopen(outwfname,"w"))==NULL )
{
  printf("Output weight file %s could not be opened\n",outwfname);
  exit(30);
} /* wfp */
 mkheader(wfp);
 if(ncyc==1) is=0;
 weights(wfp);
if(pflag==train)
  fprintf(wfp,"#Stopped at %d iterations as predict error was increasing\n",k);
fclose(wfp);

return 0;
} /* End of main */
```

B.1.2 newpat

This routine is used to set network inputs to the following sequence of time series values, i.e. xi[t] to xi[t+NI-1]. If NISPLIT is less than or equal to NI then the inputs will also contain values from the second time series xi2[]. The desired output is placed in out[i].

```
int newpat(int t) /* Starting from element t put NI numbers in the inputs */
{                 /* from xi, and the corresponding outputs in out[] */
int i,imt;
double x;
if(xi[t+NI+nskip] == -999.999) /* If end of chunk before last output */
  t += NI+nskip+1; /* Move to next chunk */
for (i=t+timgra;i<(t+NI);i++) {  /* timgra = 1 for time gradient */
        imt = i - t;
        if (imt>=NISPLIT) /* Use 2nd input data */
           v[(imt)][0] = xi2[i-NISPLIT];
        else
           v[(imt)][0] = xi[i]; /* Use 1rst input data */
        }
if(timgra==1)
 v[0][0] = v[NI-1][0] - v[NI-2][0];  /* Time Gradient as first input */

for (i=0;i<NO;i++)
if(ncyc==1)
  out[i] = xi[t+NI+i];
else
  out[i] = xi[t+NI+nskip];

return t;
} /* END OF newpat */
```

B.1.3 Other routines

The rnd() routine uses the standard C library rand() routine to generate a random number between 0 and 1. In some versions of the program, when "quality" random numbers were desired this routine was replaced by a more reliable generator from the NAG libraries.

```
double rnd()
{
double x;
x = (double) rand() / (1.0+RAND_MAX);
return x;
} /* End of rnd */
```

The "getline" routine is taken from Kernighan and Ritchie (1988), the book from which I learned to program in C. The fgets standard C library routine does the same function, so it's inclusion is really un-necessary.

```
int getline(char s[], int lim, FILE *ifp)
{
        int c,i;

        for(i=0;i<lim-1 && (c=fgetc(ifp))!=EOF && c!='\n';++i)
        s[i] = c;
        if (c=='\n') {
                s[i] = c;
                ++i;
        }
        s[i] = '\0';
        return i;
} /* End of getline */
```

A routione to set weight changes to zero

```
void zero_odw(void )
{
int i,j,m;

for (m=1;m<NL;m++) {
        for (i=0;i<nu[m];i++) {
                for(j=0;j<nu[m-1];j++) {
                        odw[i][j][m] = 0.0;
                        dw[i][j][m] = 0.0;
                }
        }
   }
} /* zero_odw */
```

B.2 The Neural Network Routines

The following routines perform the neural network algorithms and were written to be applicable to any FFNN problem, consequently they contain no reference to the time series arrays - all contact with the rest of the program is via passed indices and the global neuron and weight arrays.

Once the inputs have been set by newpat this routine updates the values of the hidden and output neurons. The variable "linflag" determines whether the output activations are linear (1) or sigmoidal (0).

```
void sweep()
{
int i,j,m;

for (m=1;m<NL;m++) {
        for (i=0;i<nu[m];i++) {
        h[i][m] = hf(i,m);
        if((linflag==0) || (m<(NL-1)))
          v[i][m] = g(h[i][m]);
        else if(linflag==1)
          v[i][m] = h[i][m];
                }
        }
} /* End of sweep */

double hf(int i, int m)
{
        double sum=0;
        int j;
        for(j=0;j<nu[m-1];j++) sum += w[i][j][m][is]*v[j][m-1];
        return sum;
} /* End of hf */

void backprop()
{
int i,j,m;
double sum;
for (i=0;i<NO;i++) {
        if(linflag==0)
          delta[i][NL-1] = gdash(h[i][NL-1]) * (out[i] - v[i][NL-1]);
        else if (linflag==1)
          delta[i][NL-1] =  (out[i] - v[i][NL-1]);
        }

for (m=NL-1;m>1;m--) {
        for(i=0;i<nu[m-1];i++) {
                sum = 0;
                for(j=0;j<nu[m];j++) sum += w[j][i][m][is]*delta[j][m];
                delta[i][m-1] = gdash(h[i][m-1]) * sum;
         }
   }
for (m=1;m<NL;m++) {
        for (i=0;i<nu[m];i++) {
                for(j=0;j<nu[m-1];j++) dw[i][j][m] += ep * delta[i][m] * v[j][m-1];
        }
   }
} /* End of backprop */

void apply()
{
int i,j,m;
for (m=1;m<NL;m++) {
        for (i=0;i<nu[m];i++) {
                for(j=0;j<nu[m-1];j++) {
                        w[i][j][m][is] += dw[i][j][m] + ap * odw[i][j][m];
                       odw[i][j][m] = dw[i][j][m];
                        dw[i][j][m] = 0.0;
                }
        }
   }
} /* End of apply() */
```

```
double g(double x)
{
return 1.0 / ( 1.0 + exp(-2.0 * BETA * x) );
} /* End of g */

double gdash(double x)
{
double gg;
gg = g(x);
gg = gg * gg;
return 2.0 * BETA * gg * exp(-2.0*be*x);
} /* End of gdash */
```

B.3 The Genetic Algorithm Routines

The following routines are used to perform the genetic algorithm training.

```
void evaluate(double *e)
{
for(is=0;is<NS;is++) e[is] =  tcost();

} /* End of evaluate */
```

The "sort" routine is taken from Press et al. (1994). Notice that by using index references to various networks it is not necessary to shunt entire network weight sets around in memory.

```
void sort(double *E,int *PE )
{
        int l,j,ir,i,n=NS,pra;
        double rra;
        for(j=0;j<NS;j++) PE[j] = j;
        l=(n >> 1)+1;
        ir=n;
        for (;;) {
                if (l > 1) {
                        rra=E[(--l) - 1];
                        pra=PE[l-1];
                        }
                else {
                        rra=E[ir-1];
                        pra=PE[ir-1];
                        E[ir-1]=E[0];
                        PE[ir-1]=PE[0];
                        if (--ir == 1) {
                                E[0]=rra;
                                PE[0]=pra;
                                return;
                        }
                }
                i=l;
                j=l << 1;
                while (j <= ir) {
                        if (j < ir && E[j-1] < E[j]) ++j;
                        if (rra < E[j-1]) {
                                E[i-1]=E[j-1];
                                PE[i-1]=PE[j-1];
                                j += (i=j);
                        }
                        else j=ir+1;
                }
                E[i-1]=rra;
                PE[i-1]=pra;
        }
} /* End of sort */
```

"breed" picks 2 parents at random and forms a new child using the "wcopy" routine

```
void breed(int *pe, int imut,double mutsize,double mutprob)
{
int c,p1,p2;

for(c=CHILD;c<NS;c++)
{
  p1 =  ((double)rand()*CHILD) / (1.0+(double)RAND_MAX);
  p2 =  ((double)rand()*CHILD) / (1.0+(double)RAND_MAX);
  wcopy(pe[p1],pe[p2],pe[c],imut,mutsize,mutprob);
} /* c */
} /* End of Breed */
```

The "wcopy" routine takes the input weight connections of parent "p1" and combines them with the output weight connections of parent "p2" to form the new child. This routine can be speeded up by swapping pointers to the input and output weight sets, instead of moving the weights around in memory.

```
void wcopy(int p1, int p2, int c, int imut,double mutsize,double mutprob)
{
int i,j,l,p=p1;

for(l=0;l<2;l++,p=p2)
{
  for(i=0;i<nu[l+1];i++)
  {
```

Network	Predicting	Training Set Size	Training Iterations	Training Time
12-18-1	SSN	790 months	6000	3hr 53min
18-18-1	SSN	790 months	9500	41hr 17min
12-12-1	SF	386 months	2584	2hr 35min
24-24-1	K_p	512 months	17,761	17hr 32min

Table B.1: This table shows typical training times of NNs used in this contract

```
    for(j=0;j<nu[l];j++)
    {
      if((imut&1) && (rnd()<mutprob) ) /* if 1 or 3 and mutprob*/
        w[i][j][l][c] = ( 1.0 + (mutsize*(rnd()-0.5)) ) * w[i][j][l][p];
      else
        w[i][j][l][c] = w[i][j][l][p];
    }
  } /* i */
} /* l */
} /* End of wcopy */
```

B.4 Other Details

The code requires no specialised computer hardware and was developed and run on SUN SPARC-stations. The run time on the fastest workstation available (a SPARC 10/40) varied from seconds to weeks, depending of course on the particular network, time series and training algorithm used. Some typical run-times for back-propagation trained networks using the fastest machine are given in Table 2.1. As can be appreciated from these figures, great patience was required especially if 2 or 3 days of training had resulted in no more than the discovery of a bug.

In addition to the neural network software presented here, many other programs were written to handle the key-files and organise the training of network jobs and process the results. Probably the most essential of these were a collection of UNIX C-Shell scripts that set-up and launched network training runs. This enabled hundreds of jobs to be scheduled whilst I was awake and in the office, and then run whilst I was asleep, hill-walking, doing theory, drinking or engaged in some combination of these activities.

Bibliography

Anderson, J. A. and Rosenfeld, E.: 1988, *Neurocomputing: Foundations of Research*, MIT Press

Aso, T. and Ogawa, T.: 1993, in J. A. Joselyn, H. Lundstedt, and J. Trolinger (eds.), *Proceedings of the International Workshop on Artificial Intelligence Applications in Solar-Terrestrial Physics*, pp 77–82, NOAA Space Environment Laboratory

Box, G. E. and Jenkins, G. M.: 1970, *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco

Bravo, S.: 1994, *A possible scenario for the associated occurence of flares, prominence eruptions, coronal mass ejections, and coronal holes in relation to interplanetary plasma*, preprint

Brigham, E.: 1988, *The fast Fourier transform and its applications*, Englewood Cliffs, New Jersey

Brockett, R. W.: 1976, *Reviews of Modern Physics* **12**, 167

Chan, Y. T.: 1995, *Wavelet Basics*, Kluwer Academic Press, London

Chapman, S. and Bartels, J.: 1940, *Geomagnetism*, Oxford University Press, Oxford

Connor, J. T., Martin, R. D., and Atlas, L. E.: 1994, *IEEE Transactions on Neural Networks* **5**, 240

*Conway, A. J.: 1993, in J. A. Joselyn, H. Lundstedt, and J. Trolinger (eds.), *Proceedings of the International Workshop on Artificial Intelligence Applications in Solar-Terrestrial Physics*, pp 31–36, NOAA Space Environment Laboratory

Cryer, J. D.: 1986, *Tim Series Analysis*, Duxbury Press, Boston

Cybenko, G.: 1989, *Mathematics of Control, Signal and Systems* **2**, 303

Daubechies, I.: 1990, *IEEE Transactions of Information Theory* **36**, 961

Davydov, I. V.: 1986, in *Population Dynamics of aquatic animals in the far eastern seas*, pp 5–17, Vladivostock

DeMeyer, F.: 1981, *Solar Physics* **70**, 259

Eddy, J. A.: 1977, in O. White (ed.), *The Solar Output and its Variations*, pp 51–71, Colorado

Associated University Press, Boulder

Farmer, J. D. and Sidorowich, J. J.: 1987, *Phys. Rev. Lett.* **59**, 845

Fraser-Smith, A. C.: 1972, *Journal of Geophysical Research* **77**, **No. 22**, 4209

Friis-Christensen, E. and Lassen, K.: 1991, *Science-Washington* **254**, 698

Gibson, J.: 1993, *Journal of Geophysical Research* **98**, 18,937

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley

Gonzalez, G. and Schatten, K. H.: 1987, *Solar Physics* **114**, 189

Hale, G. E.: 1912, *Publications of the Astronomical Society of the Pacific* **24**, 223

Hale, G. E., Hellerman, F., Nicholson, S. B., and Joy, A. H.: 1919, *The Astrophysical Journal* **49**, 153

Hargreaves, J. K.: 1992, *The Solar-Terrestrial Environment*, Cambridge University Press, Cambridge

Hertz, J., Krogh, A., and Palmer, R. G.: 1991, *Introduction to the Theory of Neural Computation*, Addison-Wesley

Holland, R. L. and Vaughan, W. W.: 1984, *Journal of Geophysical Research* **89**, 11

Jenkins, G. M. and Watts, D. G.: 1968, *Spectral Analysis and its applications*, Holden-Day, San Francisco

Kapoor, S. G. and Wu, S. M.: 1982, *Journal of Geophysical Research* **87A1**, 9

Kernighan, B. W. and Ritchie, D. M.: 1988, *The (ANSI) C Programming Language - Second Edition*, Prentice-Hall, New Jersey

Kerridge, D., Carlaw, V., and Beamish, D.: 1989, *Development and Testing of Computer Algorithms for Solar and Geomagnetic Activity Forecasting*, European Space Agency Contract Report WM/89/22c, British Geological Society

Kurths, J. and Ruzmaikin, A. A.: 1990, *Solar Physics* **126**, 407

Lapedes, A. and Farber, R.: 1987, *Nonlinear Signal Processing Using Neural Networks: Prediction and System Modelling*, Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM

Liggett, M. A. and Zirin, H.: 1985, *Solar Physics* **97**, 51

Lonnblad, L., Peterson, C., and Rögnvaldsson, T.: 1990, *Physical Review Letters* **65(11)**, 1321

Macpherson, . K. P.: 1994, *Ph.D. thesis*, University of Glasgow, Glasgow

Macpherson, K. and Kirkton, F.: 1995, Private communication

*Macpherson, K. P.: 1993, in J. A. Joselyn, H. Lundstedt, and J. Trolinger (eds.), *Proceedings of the International Workshop on Artificial Intelligence Applications in Solar-Terrestrial Physics*, pp 65–70, NOAA Space Environment Laboratory

McNish, A. G. and Lincoln, J. V.: 1949, *Transactions of the American Geophysical Journal* **30**, 673

Press, Teukolsky, Vetterling, and Flannery: 1994, *Numerical Recipes in C/FORTRAN - second edition*, Cambridge University Press, Cambridge

Priestley, M. B.: 1980, *J. Time Series Anal.* **1**, 47, Reprinted in Anderson and Rosenfeld (1988)

Priestley, M. B.: 1988, *Non-Linear and Non-Stationary Time Series Analysis*, Academic Press Ltd, London

Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: 1986, in D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Chapt. 8, pp 318–362, MIT Press, Reprinted in Anderson and Rosenfeld (1988)

Schatten, K. H. and Sofia, S.: 1987, *Geophysical Research Letters* **14**, 632

Storrie-Lombardi, M. C., Lahav, O., Storrie-Lombardi, L., and Sodr, L.: 1992, *Monthly Notices of the Royal Astronomical Society* **259**, 8

SubbaRao, T. and Gabr, M. M.: 1984, *An Introduction to Bispectral Analysis and Bilinear Time Series Models*, Springer-Verlag, New York

Takens, F.: 1981, in D. Rand and L.-S. Young (eds.), *Detecting Strange Attractors In Turbulance*, Vol. 898 of *Lecture Notes in Mathematics*, pp 366–381, Springer-Verlag

Tong, H.: 1983, *Threshold Models in Non-Linear Time Series Analysis*, Springer-Verlag, New York

Wells, H. G.: 1932, *Things to Come*

Wilson, R. M.: 1990, *Solar Physics* **115**, 125

Yu, X., Chen, G., and Cheng, S.: 1995, *IEEE trans. on Neural Networks* **6**, 669

Yule, G. U.: 1926, *Phil. Trans. R. Soc.* **226**, 267

Zhang, G.: 1988, *Chinese Astronomy and Astrophysics* **12**, 249

Zirin, H.: 1987, *Astrophysics of the Sun*, Cambridge University Press, England