Forecasting Water Resources Variables

Using

Artificial Neural Networks

By

Gavin James Bowden
B.E. Civil/Env (Hons)

Thesis submitted for the degree of
Doctor of Philosophy

The University of Adelaide
School of Civil and Environmental Engineering
Australia

February 2003

To my parents,

Judith and Barry,

a constant source of love

Contents

List of Figures ..... *viii*

List of Tables ..... *xix*

Abstract ..... *xxv*

List of Publications ..... *xxvii*

Statement of Originality ..... *xxix*

Acknowledgments ..... *xxx*

Chapter 1 Introduction *1*

1.1 Reasons for this Research 1

1.2 Objectives and Scope 4

1.3 Layout and Contents of Thesis 5

Chapter 2 Review of Artificial Neural Networks *7*

2.1 Background 7

2.2 Introduction to Artificial Neural Networks 9

2.2.1 Analogy to the Brain 9

2.2.2 History of ANNs 10

2.3 ANNs Compared With Statistical Models 14

2.4 Types of ANN Models 17

2.4.1 Supervised, Feedforward ANNs 20

2.4.1.1 Multilayer Perceptron (MLP) 21

2.4.1.2 Evolutionary ANNs (EANNs) 34

2.4.1.3 General Regression Neural Network (GRNN) 37

2.4.2 Unsupervised ANNs 43

2.4.2.1 Self-Organizing Map (SOM) 43

2.5 Important Steps in the Development of ANN Models for Time Series Forecasting 47

2.5.1 Choice of Performance Criteria 48

2.5.2 Choice of Data Sets, Data Transformation and Preprocessing 50

2.5.3 Determination of Model Inputs 53

2.5.4 Choice of Model Architecture 54

2.5.4.1 Heuristic Approaches 56

2.5.5 Training (Optimisation) 57

2.5.5.1 Choice of Stopping Criteria 58

2.5.5.2 Internal Model Parameters 59

2.5.6 Validation 59

2.5.7 Model Deployment 59

2.6 ANNs in Water Resources Applications 61

Chapter 3 ANN Methodology for Modelling Water Resources Variables 74

3.1 Introduction 74

3.2 Testing For Nonlinearity 75

3.2.1 Introduction 75

3.2.2 Methods 78

3.2.2.1 The BDS test 79

3.2.2.2 Fuzzy Classification System (Kaboudan, 1999) 80

3.2.3 Application of the Nonlinearity Tests to the Synthetic Test Data 84

3.2.3.1 BDS Test Results 85

3.2.3.2 Kaboudan's FCS Results 86

3.2.3.3 Nonlinearity Testing and ANN Modelling 87

3.2.4 Conclusions and Recommendations 90

3.3 Division of Data for ANN Models 92

3.3.1 Introduction 92

3.3.2 Methods 95

3.3.2.1 Data Division Using a Genetic Algorithm 95

3.3.2.2 Data Division Using a SOM 97

3.4 Data Transformation 99

3.4.1 Introduction 99

3.4.2 Methods 103

3.4.2.1 Linear Transformation 103

3.4.2.2 Logarithmic Transformation 103

3.4.2.3 Histogram Equalization (Looney, 1997) 103

3.4.2.4 Kernel Transformation 104
3.4.2.5 Seasonal Standardisation 106
3.4.2.6 Transformation to Normality 107

3.5 Determination of ANN Model Inputs 108
3.5.1 Introduction 108
3.5.1.1 Review of Input Determination in Water Resources ANN Applications.. 110
3.5.2 Methods 115
3.5.2.1 Unsupervised Input Preprocessing 115
3.5.2.2 Supervised Input Determination 117
3.5.3 Application of the Input Determination Methods to Synthetic Data Sets 126

3.6 Choice of Network Type and Architecture 133
3.6.1 Introduction 133
3.6.2 Methods 135
3.6.2.1 Evolutionary Backpropagation Multilayer Perceptron (EBMLP)..... 135
3.6.2.2 General Regression Neural Network (GRNN) 139

3.7 Training (Optimisation) 145
3.7.1 Choice of Performance Measure 145
3.7.1.1 Introduction 145
3.7.1.2 Review of Performance Measures Used in Water Resources Modelling.. 145
3.7.1.3 Method 152
3.7.2 Choice of Optimisation Methods 153
3.7.2.1 Introduction 153
3.7.2.2 Methods 156

3.8 Model Deployment 164
3.8.1 Introduction 164
3.8.2 Methods 166
3.8.2.1 Hybrid SOM-MLP Model 166
3.8.2.2 GRNN Outlier Detector 168

Chapter 4 Case Study I: Salinity in the River Murray 171

4.1 Introduction 171

4.2 Background 174
4.2.1 The Murray-Darling Basin and the River Murray 174
4.2.2 Salinity in the River Murray 178
4.2.3 Forecasting Salinity at Murray Bridge 182

4.3 Available Data 184

4.3.1 Salinity Data 185
4.3.2 Flow Data 196
4.3.3 River Level Data 199

4.4 ANN Model Development 205
4.4.1 Data Transformation 207
4.4.2 Model Inputs 207
4.4.3 Network Architecture 208
4.4.4 Internal Model Parameters 208
4.4.5 Stopping Criterion and Performance Measure 209

4.5 Testing For Nonlinearity 210
4.5.1 Results and Discussion 210
4.5.1.1 BDS Test 210
4.5.1.2 Kaboudan's FCS 210
4.5.2 Summary 211

4.6 Division of Data for ANN Models 212
4.6.1 Model Development 212
4.6.1.1 Data Division 212
4.6.2 Results and Discussion 220

4.7 Data Transformation 226
4.7.1 Model Development 226
4.7.1.1 Data Transformation 227
4.7.2 Results and Discussion 239
4.7.2.1 Real-time forecasting 253

4.8 Determination of Model Inputs 256
4.8.1 Model Development 256
4.8.1.1 Determination of Model Inputs 257
4.8.1.2 Determination of Network Architecture 270
4.8.2 Results and Discussion 271
4.8.2.1 Real-time forecasting 277

4.9 Choice of Network Type and Architecture 283
4.9.1 Model Development 283
4.9.1.1 Determination of Network Architecture 284
4.9.2 Results and Discussion 286
4.9.2.1 EBMLP 286
4.9.2.2 GRNN 289
4.9.2.3 Real-time forecasting 292

4.10 Training (Optimisation) 295

4.10.1 Choice of Performance Measure 295
4.10.2 Choice of Optimisation Method 301
4.10.2.1 Feedforward MLP 301
4.10.2.2 Multiple-Sigma General Regression Neural Network 305

4.11 Model Deployment 311
4.11.1 Backpropagation MLP 311
4.11.2 GRNN 314

4.12 Summary and Conclusions 319
4.12.1 Testing for Nonlinearity 319
4.12.2 Data Division 319
4.12.3 Data Transformation 321
4.12.4 Determination of Model Inputs 321
4.12.5 Choice of Network Type and Architecture 323
4.12.6 Training (Optimisation) 324
4.12.7 Model Deployment 325

Chapter 5 Case Study II: Cyanobacteria in the River Murray 327

5.1 Introduction 327

5.2 Background 329
5.2.1 Factors Affecting the Incidence of Cyanobacterial Blooms 330
5.2.1.1 'Bottom Up' Factors 331
5.2.1.2 'Top Down' Factors 344
5.2.2 Cyanobacterial Toxins, Their Production and Effects 345
5.2.2.1 Effects on Public Health 346
5.2.2.2 Effects on Animal Health 350
5.2.2.3 Ecological Effects 351
5.2.2.4 Effects on Recreation and Tourism 351
5.2.3 Toxic Cyanobacteria in the Murray-Darling Basin 352
5.2.3.1 History of Bloom Formation 353
5.2.3.2 Long Term Methods of Controlling Cyanobacteria 354
5.2.4 Modelling Cyanobacteria 356
5.2.4.1 Process-Based (Deterministic) Models 356
5.2.4.2 Statistically Based Models 357
5.2.4.3 Rule-based (Heuristic) Models 358
5.2.4.4 Artificial Neural Network Models 358
5.2.5 Forecasting *Anabaena* spp. at Morgan 359

5.3 Available Data 360
5.3.1 *Anabaena* spp. 364

5.3.2 Flow 370
5.3.3 River Level 370
5.3.4 Temperature 372
5.3.5 Colour 372
5.3.6 Turbidity 373
5.3.7 pH 374
5.3.8 Silica 375
5.3.9 TKN 376
5.3.10 Total Phosphorus 377
5.3.11 Soluble Phosphorus 379
5.3.12 Wind Run 379

5.4 ANN Model Development 381

5.5 Testing For Nonlinearity 383
5.5.1 Results and Discussion 383
5.5.1.1 BDS Test 383
5.5.1.2 Kaboudan's FCS 384
5.5.2 Summary 384

5.6 Division of Data for ANN Models 385
5.6.1 Results and Discussion 385

5.7 Data Transformation 390
5.7.1 Linear Transformation 395
5.7.2 Histogram Equalization 396

5.8 Determination of Model Inputs 403
5.8.1 Linear Transformation (Raw) Data 404
5.8.1.1 Stepwise PMI Algorithm 404
5.8.1.2 SOM-GAGRNN 406
5.8.2 Histogram Equalization Transformation Data 408
5.8.2.1 Stepwise PMI Algorithm 408
5.8.2.2 SOM-GAGRNN 411

5.9 Choice of Network Type and Architecture 413
5.9.1 Linear Transformation (Raw) Data 413
5.9.1.1 Stepwise PMI Algorithm 413
5.9.1.2 SOM-GAGRNN 417
5.9.2 Histogram Equalization Transformation Data 421
5.9.2.1 Stepwise PMI Algorithm 421
5.9.2.2 SOM-GAGRNN 424
5.9.3 Summary 428

5.10 Training (Optimisation) **430**
5.10.1 Choice of Performance Measure 430
5.10.2 Choice of Optimisation Method 433
5.10.2.1 Number of strata 433
5.10.2.2 The minimum pheromone trail 434
5.10.2.3 Pheromone persistence coefficient 434
5.10.2.4 Number of ants in the population 435
5.10.2.5 Summary 436

5.11 Model Deployment **438**

5.12 Summary and Conclusions **448**

Chapter 6 Conclusions and Recommendations ***451***

6.1 Contributions of the Research **451**

6.2 General Conclusions **456**

6.3 Specific Conclusions **460**
6.3.1 Case Study I: Salinity in the River Murray 460
6.3.2 Case Study II: Cyanobacteria in the River Murray 462

6.4 Recommendations for Future Work **463**

Appendix A Notation ***465***

Appendix B Abbreviations ***471***

Bibliography ***475***

List of Figures

Figure 2.1 (a) A Biological Neuron and (b) An Artificial Neuron Model (PE) (source: Takagi, 1997) 9
Figure 2.2 A Typical ANN Architecture 10
Figure 2.3 A Two-Layer Feedforward ANN Representation of a Multiple Linear Regression Model 15
Figure 2.4 (a) A Feedforward Neural Network and (b) A Feedback Neural Network (source: Takagi, 1997) 17
Figure 2.5 A Simple Feedforward Neural Network Example (source: Takagi, 1997). 21
Figure 2.6 Scenario For Increasing the Learning Rate for a Connection Weight. The Weight Changes Have the Same Sign For Several Time Steps Over a Region of Relatively Low Curvature (source: NeuralWare, 1998b) 29
Figure 2.7 Scenario For Decreasing the Learning Rate for a Connection Weight. The Weight Changes Have Opposite Signs for Several Time Steps Over a Region of Relatively High Curvature (source: NeuralWare, 1998b) 29
Figure 2.8 The Sigmoid Function 32
Figure 2.9 The Hyperbolic Tangent Function 32
Figure 2.10 Coding of the Connectivity Matrix and Decoding to the ANN (source: Miller et al., 1989) 36
Figure 2.11 The GRNN Architecture 38
Figure 2.12 A Typical SOM Consisting of N Inputs in the Input Layer and a 5 by 5 Kohonen Layer 45
Figure 2.13 The Main Steps in the Development of ANN Models (source: adapted from Maier and Dandy, 2000a) 48
Figure 2.14 Distribution of Papers Reviewed By Publication Date 71
Figure 2.15 Distribution of Papers Reviewed By Variable Modelled 71
Figure 3.1 Example of the Sampling Process Used in the SOM Data Division Method 98

Figure 3.2 Input Determination Techniques *114*
Figure 3.3 Actual Bivariate Probability Density Between x and y *123*
Figure 3.4 Estimated Bivariate Probability Density Between x and y - Gaussian Kernel *124*
Figure 3.5 Estimated Bivariate Probability Density Between x and y - City Block Distance Kernel *124*
Figure 3.6 The Divided Difference Method For Parameter Derivatives *141*
Figure 3.7 Schematic Representation of the Discretised Parameter Space *162*
Figure 3.8 Hybrid SOM-BPN model *167*
Figure 3.9 GRNN Outlier Detector *170*
Figure 4.1 The Murray-Darling Basin in South-East Australia (source: Murray-Darling Basin Ministerial Council, 1988) *174*
Figure 4.2 River Murray System – Regulation Structures and Typical Regulated Flows During Peak Irrigation Periods (ML/day) (source: Murray-Darling Basin Commission, 1990) *176*
Figure 4.3 The South Australian Reaches of the River Murray and the River Murray Pipelines (source: Murray-Darling Basin Commission, 1990) *177*
Figure 4.4 Groundwater Salinity in the South Australian Reaches of the River Murray (source: Department for Water Resources, 2001) *179*
Figure 4.5 Contribution of Each Reach of the River Murray to the Average Salinity at Morgan in EC Units and in Percentage (source: Ghassemi et al., 1995) *180*
Figure 4.6 Average Salinity Levels for Current Conditions, 2020, 2050 and 2100 at Various Stations Along the River Murray Assuming That There is No Intervention (source: Department for Water Resources, 2001) *182*
Figure 4.7 Locations of Available Data *186*
Figure 4.8 Salinity at Murray Bridge (MBS) (1987 to 1998) *186*
Figure 4.9 Annual Exceedance Probabilities (AEP) Using Peak Salinities for the Salinity Time Series at Murray Bridge (1987 to 1998) *187*
Figure 4.10 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1987 to 1989) *188*
Figure 4.11 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1987 to 1989) *189*
Figure 4.12 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1987 to 1989) *189*
Figure 4.13 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1987 to 1989) *190*
Figure 4.14 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1990 to 1992) *190*

Figure 4.15 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1990 to 1992) *191*
Figure 4.16 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1990 to 1992) *191*
Figure 4.17 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1990 to 1992) *192*
Figure 4.18 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1993 to 1995) *192*
Figure 4.19 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1993 to 1995) *193*
Figure 4.20 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1993 to 1995) *193*
Figure 4.21 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1993 to 1995) *194*
Figure 4.22 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1996 to 1998) *194*
Figure 4.23 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1996 to 1998) *195*
Figure 4.24 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1996 to 1998) *195*
Figure 4.25 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1996 to 1998) *196*
Figure 4.26 Salinity at Murray Bridge (MBS) and Flow at Lock 1 Lower (L1LF) (1987 to 1998) *197*
Figure 4.27 Salinity at Murray Bridge (MBS) and Flow at Overland Corner (OCF) (1987 to 1998) *198*
Figure 4.28 Salinity at Murray Bridge (MBS) and Flow Downstream of Lock 7 (L7F) (1987 to 1998) *198*
Figure 4.29 Salinity at Murray Bridge (MBS) and River Level at Murray Bridge (MBL) (1987 to 1998) *200*
Figure 4.30 Salinity at Murray Bridge (MBS) and River Level at Mannum (MAL) (1987 to 1998) *201*
Figure 4.31 Salinity at Murray Bridge (MBS) and River Level at Lock 1 Lower (L1LL) (1987 to 1998) *201*
Figure 4.32 Salinity at Murray Bridge (MBS) and River Level at Lock 1 Upper (L1UL) (1987 to 1998) *202*
Figure 4.33 Salinity at Murray Bridge (MBS) and River Level at Morgan (MOL) (1987 to 1998) *202*

Figure 4.34 Salinity at Murray Bridge (MBS) and River Level at Waikerie (WAL) (1987 to 1998) 203
Figure 4.35 Salinity at Murray Bridge (MBS) and River Level at Overland Corner (OCL) (1987 to 1998) 203
Figure 4.36 Salinity at Murray Bridge (MBS) and River Level at Loxton (LOL) (1987 to 1998) 204
Figure 4.37 Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 1 (May 1991 to June 1992) 221
Figure 4.38 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 1 (July 1992 to March 1998) 222
Figure 4.39 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 2 (July 1992 to March 1998) 223
Figure 4.40 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 3 (July 1992 to March 1998) 223
Figure 4.41 Histogram of Salinity at Murray Bridge (Raw Data) 228
Figure 4.42 Histogram of Salinity at Mannum (Raw Data) 228
Figure 4.43 Histogram of Salinity at Morgan (Raw Data) 229
Figure 4.44 Histogram of Salinity at Waikerie (Raw Data) 229
Figure 4.45 Histogram of Salinity at Loxton (Raw Data) 230
Figure 4.46 Histogram of Flow at Overland Corner (Raw Data) 230
Figure 4.47 Histogram of River Level at Lock 1 Lower (Raw Data) 231
Figure 4.48 Histogram of Salinity at Murray Bridge (Log Transformed Data) 232
Figure 4.49 Histogram of Flow at Overland Corner (Log Transformed Data) 232
Figure 4.50 Histogram of River Level at Lock 1 Lower (Log Transformed Data) 233
Figure 4.51 Histogram of Salinity at Murray Bridge (Histogram Equalization Transformed Data) 234
Figure 4.52 Histogram of Flow at Overland Corner (Histogram Equalization Transformed Data) 234
Figure 4.53 Histogram of River Level at Lock 1 Lower (Histogram Equalization Transformed Data) 235
Figure 4.54 Histogram of Salinity at Murray Bridge (Seasonally Standardised Data) 236
Figure 4.55 Histogram of Flow at Overland Corner (Seasonally Standardised Data) 236
Figure 4.56 Histogram of River Level at Lock 1 Lower (Seasonally Standardised Data) 237
Figure 4.57 Histogram of Salinity at Murray Bridge (Transformation to Normality) 238
Figure 4.58 Histogram of Flow at Overland Corner (Transformation to Normality) 238

Figure 4.59 Histogram of River Level at Lock 1 Lower (Transformation to Normality) 239
Figure 4.60 Sensitivity of Salinity Forecasts to the 51 Input Variables for the Model Developed using (a) Logarithmic Transformation; (b) Seasonal Transformation; (c) Transformation to Normality; (d) Linear Transformation 241
Figure 4.61 Flow at Overland Corner and Fourier Series Seasonal Mean (OCF) for (a) Training, Testing and Validation Data (December 1986 to June 1992) and (b) Second Validation Data (July 1992 to March 1998) 242
Figure 4.62 Histogram of ANN Residuals (Linear Transformation) 243
Figure 4.63 Histogram of ANN Residuals (Logarithmic Transformation) 244
Figure 4.64 Histogram of ANN Residuals (Histogram Equalization Transformation) 244
Figure 4.65 Histogram of ANN Residuals (Kernel Transformation) 245
Figure 4.66 Histogram of ANN Residuals (Seasonal Transformation) 245
Figure 4.67 Histogram of ANN Residuals (Transformation to Normality) 246
Figure 4.68 Standardised Residual Versus Predicted Response - 2005 Data Points (Linear Transformation) 246
Figure 4.69 Standardised Residual Versus Predicted Response - 2005 Data Points (Logarithmic Transformation) 247
Figure 4.70 Standardised Residual Versus Predicted Response - 2005 Data Points (Histogram Equalization Transformation) 247
Figure 4.71 Standardised Residual Versus Predicted Response - 2005 Data Points (Kernel Transformation) 248
Figure 4.72 Standardised Residual Versus Predicted Response - 2005 Data Points (Seasonal Transformation) 248
Figure 4.73 Standardised Residual Versus Predicted Response - 2005 Data Points (Transformation to Normality) 249
Figure 4.74 Autocorrelation Plots of ANN Residuals (Linear Transformation) 249
Figure 4.75 Autocorrelation Plots of ANN Residuals (Logarithmic Transformation) 250
Figure 4.76 Autocorrelation Plots of ANN Residuals (Histogram Equalization Transformation) 250
Figure 4.77 Autocorrelation Plots of ANN Residuals (Kernel Transformation) 251
Figure 4.78 Autocorrelation Plots of ANN Residuals (Seasonal Transformation) 251
Figure 4.79 Autocorrelation Plots of ANN Residuals (Transformation to Normality) 252
Figure 4.80 Second Validation Set 14-Day Forecasts for the Model Developed Using Seasonally Transformed Data (July 1992 to March 1998). The Fourier Series Seasonal Mean (MBS) is also Shown For the Salinity at Murray Bridge 254

Figure 4.81 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 1 (Model 1-4) 274
Figure 4.82 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 2 (Model 2-4) 275
Figure 4.83 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 3 (Model 3-2) 275
Figure 4.84 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Empirical Knowledge and Sensitivity Analysis (Model 4) 276
Figure 4.85 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by the Method of Haugh and Box (Model 5) 276
Figure 4.86 Comparison of the Relative Significance of Inputs Obtained Using Sensitivity Analysis and PMI Scores (Model 1-4) 277
Figure 4.87 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 1 (Model 1-4: August 1992 to March 1998) 279
Figure 4.88 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 2 (Model 2-4: August 1992 to March 1998) 279
Figure 4.89 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 3 (Model 3-2: August 1992 to March 1998) 280
Figure 4.90 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by the PCA-GAGRNN Method 2 (Model 6-4: August 1992 to March 1998) 282
Figure 4.91 Single-Sigma GRNN Error Surface in the Vicinity of the Global Minimum 290
Figure 4.92 Test Set RMSEs and Number of Clusters as a Percentage of the Total Number of Training Set Records For Each Single-Sigma GRNN Investigated 291
Figure 4.93 Second Validation Set 14-Day Forecasts for the EBMLP (August 1992 to March 1998) 293
Figure 4.94 Second Validation Set 14-Day Forecasts for the Single-Sigma GRNN (August 1992 to March 1998) 293
Figure 4.95 Test Set Performance of the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 16) 302
Figure 4.96 Test Set Performance of the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 100) 304
Figure 4.97 Comparison of the Number of Iterations Required by MMAS and the GA to Find the Best Set of Sigma Weights 310
Figure 4.98 Results of Not Retraining the MLP Model for the Real-Time Forecasting Test Period (August 1992 to March 1998). The Fourier Series Seasonal Mean is also shown for the Salinity at Murray Bridge. 312

Figure 4.99 Results of Retraining the MLP Model After Each Sample for the Real-Time Forecasting Test Period (August 1992 to March 1998) 313
Figure 4.100 Results of Retraining the MLP Model Using the Hybrid SOM-MLP Model for the Real-Time Forecasting Test Period (August 1992 to March 1998) 314
Figure 4.101 Results of Not Retraining the GRNN Model for the Real-Time Forecasting Test Period (August 1992 to March 1998) 315
Figure 4.102 Results of Retraining the GRNN Model After Each Sample for the Real-Time Forecasting Test Period (August 1992 to March 1998) 316
Figure 4.103 Results of Retraining the GRNN Model After Each Sample Without Waiting 14 Days to Update - Real-Time Forecasting Test Period (August 1992 to March 1998) 316
Figure 4.104 Results of Retraining the GRNN Model Using the GRNN Outlier Detection Method for the Real-Time Forecasting Test Period (August 1992 to March 1998) 317
Figure 5.1 Examples of the Four Main Groups of Algae (source: adapted from Sullivan, 1990) 330
Figure 5.2 The Euphotic Zone (Z_{eu}) Relative to the Epilimnion (Z_m) in Situations With Different Turbidity. A. Euphotic zone is deeper than epilimnion; B. Euphotic zone is not as deep as epilimnion. Secchi depth (Z_s) is included as a rough measure of euphotic depth (Z_{eu}) ($Z_{eu} \cong Z_s \times 2.5$) (source: Chorus and Bartram, 1999) 335
Figure 5.3 Median Silica (1978-1986) and Melosira granulata (1980-1985) Concentrations Along the River Murray (source: Sullivan, 1990) 342
Figure 5.4 Relationship Between Counting Effort and Counting Error (Precision) For Trichomes of Anabaena (source: Laslett et al., 1997) 364
Figure 5.5 Concentrations of Anabaena spp. at Morgan (1980 to 1996) 365
Figure 5.6 Concentrations of Anabaena spp. at Morgan (1980 to 1996) and the Alert Levels 366
Figure 5.7 Number of Records in Each Alert Category For Anabaena spp. at Morgan (1980 to 1996) 367
Figure 5.8 Concentrations of Anabaena spp. at Morgan on a Log Scale (1980 to 1996) 368
Figure 5.9 Histogram of Anabaena spp. at Morgan (Raw Data) 368
Figure 5.10 Histogram of Anabaena spp. at Morgan (Log Transformed) 369
Figure 5.11 Histogram of the Non-Zero Values of Anabaena spp. at Morgan (Log Transformed) 369
Figure 5.12 Concentrations of Anabaena spp. at Morgan and Flow into SA (1980 to 1996) 370
Figure 5.13 Concentrations of Anabaena spp. and River Level at Morgan (1980 to 1996) 371

Figure 5.14 Flow into SA and River Level at Morgan (1980 to 1996) *371*
Figure 5.15 Concentrations of Anabaena spp. and Temperature at Morgan (1980 to 1996) *372*
Figure 5.16 Concentrations of Anabaena spp. and Colour at Morgan (1980 to 1996) *373*
Figure 5.17 Concentrations of Anabaena spp. and Turbidity at Morgan (1980 to 1996) *374*
Figure 5.18 Concentrations of Anabaena spp. and pH at Morgan (1980 to 1996) ... *375*
Figure 5.19 Concentrations of Anabaena spp. and Silica at Morgan (1980 to 1996) *376*
Figure 5.20 Concentrations of Anabaena spp. and TKN at Morgan (1980 to 1996) *377*
Figure 5.21 Concentrations of Anabaena spp. and Total Phosphorus at Morgan (1980 to 1996) *378*
Figure 5.22 Concentrations of Total Phosphorus and Turbidity at Morgan (1980 to 1996) *378*
Figure 5.23 Concentrations of Anabaena spp. and Soluble Phosphorus at Morgan (1980 to 1996) *379*
Figure 5.24 Concentrations of Anabaena spp. at Morgan and Wind Run Below 3m at Blanchetown (1980 to 1996) *380*
Figure 5.25 ANN Model Development Steps for the Anabaena spp. Case Study *382*
Figure 5.26 Histogram of Temperature at Morgan (Raw Data) *391*
Figure 5.27 Histogram of Flow into SA (Raw Data) *391*
Figure 5.28 Histogram of Colour at Morgan (Raw Data) *392*
Figure 5.29 Histogram of Turbidity at Morgan (Raw Data) *392*
Figure 5.30 Histogram of pH at Morgan (Raw Data) *393*
Figure 5.31 Histogram of Silica at Morgan (Raw Data) *393*
Figure 5.32 Histogram of TKN at Morgan (Raw Data) *394*
Figure 5.33 Histogram of Total Phosphorus at Morgan (Raw Data) *394*
Figure 5.34 Histogram of Soluble Phosphorus at Morgan (Raw Data) *395*
Figure 5.35 Histogram of River Level at Morgan (Raw Data) *395*
Figure 5.36 Histogram of Temperature at Morgan (Histogram Equalization Transformed Data) *397*
Figure 5.37 Histogram of Flow into SA (Histogram Equalization Transformed Data) *397*
Figure 5.38 Histogram of Colour at Morgan (Histogram Equalization Transformed Data) *398*
Figure 5.39 Histogram of Turbidity at Morgan (Histogram Equalization Transformed Data) *398*
Figure 5.40 Histogram of pH at Morgan (Histogram Equalization Transformed Data) *399*

Figure 5.41 Histogram of Silica at Morgan (Histogram Equalization Transformed Data) 399
Figure 5.42 Histogram of TKN at Morgan (Histogram Equalization Transformed Data) 400
Figure 5.43 Histogram of Total Phosphorus at Morgan (Histogram Equalization Transformed Data) 400
Figure 5.44 Histogram of Soluble Phosphorus at Morgan (Histogram Equalization Transformed Data) 401
Figure 5.45 Histogram of River Level at Morgan (Histogram Equalization Transformed Data) 401
Figure 5.46 Histogram of log(Anabaena spp.) at Morgan (Histogram Equalization Transformed Data) 402
Figure 5.47 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 1) 414
Figure 5.48 Validation Set GRNN 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 1) 415
Figure 5.49 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 2) 416
Figure 5.50 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 2) 417
Figure 5.51 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 3) 418
Figure 5.52 Validation Set GRNN 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 3) 418
Figure 5.53 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 4) 420
Figure 5.54 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 4) 420
Figure 5.55 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm 421

Figure 5.56 Validation Set GRNN 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm 422
Figure 5.57 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm 423
Figure 5.58 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm 424
Figure 5.59 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN 425
Figure 5.60 Validation Set GRNN 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN 425
Figure 5.61 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN 427
Figure 5.62 Validation Set EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN 427
Figure 5.63 Sigma Weight For Each Input in the MMAS-GRNN 437
Figure 5.64 Results of Not Retraining the GRNN Model for the Real-Time Forecasting Test Period (May 1998 to January 2002) 439
Figure 5.65 Results of Retraining the GRNN Model After Each Sample for the Real-Time Forecasting Test Period (May 1998 to January 2002) 440
Figure 5.66 Results of Retraining the GRNN Model After Each Sample Without Waiting 4 Weeks to update - Real-Time Forecasting Test Period (May 1998 to January 2002) 441
Figure 5.67 Results of Retraining the GRNN Model With 50 Per Cent of the Second Validation Set Samples Randomly Removed and Added to the Calibration Data (Remaining Patterns From May 1998 to January 2002) 441
Figure 5.68 Results of Retraining the GRNN Model Using the GRNN Outlier Detection Method for the Real-Time Forecasting Test Period (May 1998 to January 2002) 442
Figure 5.69 A Comparison of the Autocorrelation Function for the Train/Test/Validation Data and the Second Validation Data 444

Figure 5.70 Comparison of Training/Testing/Validation Data Cross Correlation and Second Validation Data Cross Correlation Between Anabaena spp. and (a) Temperature, (b) Flow, (c) Colour, (d) Turbidity, (e) pH, (f) Silica, (g) TKN, (h) Total Phosphorus, (i) Soluble Phosphorus, and (j) River Level..... 447

List of Tables

Table 2.1 International Journal Papers Reviewed on the Application of ANNs to Water Resources Variables 63
Table 3.1 The Implication Rules (source: Kaboudan, 1999) 82
Table 3.2 The Membership Rules (source: Kaboudan, 1999) 83
Table 3.3 BDS Test Z Statistics, Residuals from ARMA Fit to Synthetic Data Sets 86
Table 3.4 Diagnosis Results of the FCS For the Synthetic Data Sets 87
Table 3.5 Regression to Estimate Omega (source: Kaboudan, 1995) 89
Table 3.6 Dummy Variable "Theta" Activation Range (source: Kaboudan, 1995) 89
Table 3.7 Sample Size Required to Ensure That the Relative Mean Square Error at Zero is Less than 0.1, When Estimating a Standard Multivariate Normal Density Using a Normal Kernel and Bandwidth That Minimises the Mean Square Error at Zero (source: Silverman, 1986) 126
Table 3.8 Modified stepwise PMI input selection algorithm results on the synthetic data test problems 128
Table 3.9 Input Subsets Selected using PCA-GAGRNN and SOM-GAGRNN for the Synthetic Data Test Problems 129
Table 3.10 RMSEs of the GRNN Models Produced from Each Input Determination Method for the Synthetic Data Test Problems 132
Table 3.11 Neural Network Parameters 138
Table 3.12 Genetic Algorithm Parameters 139
Table 3.13 The Ten Performance Measures Investigated in this Study 152
Table 3.14 Default Values for Learning Rate (η) and Momentum (μ) 156
Table 4.1 Entitlement Flows to South Australia 176
Table 4.2 Available Data (Daily) 185
Table 4.3 Statistics of the Salinity Time Series Data (1987 to 1998) 187
Table 4.4 Statistics of the Flow Time Series Data (1987 to 1998) 196
Table 4.5 Regression Statistics for the Flow Time Series Data (1987 to 1998) 197

Table 4.6 Statistics of the River Level Time Series Data (1987 to 1998) *199*
Table 4.7 Regression Statistics for the River Level Time Series Data (1987 to 1998) *200*
Table 4.8 Summary of Default Model Inputs *207*
Table 4.9 Internal Parameters of the EDBD Training Algorithm *208*
Table 4.10 Default Learning Rates and Momentum *209*
Table 4.11 BDS Test Z Statistics, Residuals from ARMA Fit to Salinity Data Set *210*
Table 4.12 Diagnosis Results of the FCS for the Salinity Data Set *211*
Table 4.13 Method 1: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using Conventional Method) *214*
Table 4.14 Method 1 (Data Divided Using Conventional Method): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set *215*
Table 4.15 Method 2: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using a GA) *216*
Table 4.16 Method 2 (Data Divided Using a GA): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set *217*
Table 4.17 Method 3: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using a SOM) *218*
Table 4.18 Method 3 (Data Divided Using a SOM): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set *219*
Table 4.19 RMSEs for the 14-Day Forecasts *220*
Table 4.20 RMSEs for the 14-Day Forecasts *225*
Table 4.21 RMSEs of 6 Different Transformations for the 14-Day Salinity Forecasts at Murray Bridge *240*
Table 4.22 Maximum, Minimum and Ratio of Max./Min. for Each Variable *242*
Table 4.23 Second Validation Set RMSEs of 6 Different Transformations for the 14-Day Salinity Forecasts at Murray Bridge *254*
Table 4.24 Stepwise PMI Input Selection Algorithm Results on Bivariate Salinity Data Sets With a Lagging Window of 30 Days *258*
Table 4.25 Stepwise PMI Input Selection Algorithm Results on Multivariate Salinity Data Set With a Lagging Window of 30 Days *261*
Table 4.26 Stepwise PMI Input Selection Algorithm Results on Bivariate Salinity Data Sets With a Lagging Window of 60 Days *263*
Table 4.27 Stepwise PMI Input Selection Algorithm Results on Multivariate Salinity Data Set With a Lagging Window of 60 Days *265*
Table 4.28 Input Subsets Selected using PCA *267*
Table 4.29 Input Subsets Selected using SOM-GAGRNN *268*

Table 4.30 Input Subset Selected by Maier and Dandy (1997a) Using the Method of Haugh and Box (1977) 270
Table 4.31 Forecasting Errors for ANN Models Using Inputs Obtained by Methods 1, 2 and 3 (the italicised numbers indicate the best model of the four different network architecture investigated) 271
Table 4.32 Comparison of the Best ANN Models Developed Using the Inputs Obtained by Methods 1, 2 and 3 with the Best ANN Models Developed Using the Inputs Obtained in Previous Studies 272
Table 4.33 Second Validation Set RMSEs of the ANN Models Produced from Methods 1, 2 and 3 for the 14-Day Salinity Forecasts at Murray Bridge 280
Table 4.34 Results of the ANN Models Produced Using Inputs Identified by PCA-GAGRNN #2 for the 14-Day Salinity Forecasts at Murray Bridge. The Italicised Numbers Indicate the Best Model 281
Table 4.35 Details of the Learning Rate and Neighbourhood Size Used for Each Size Kohonen Layer 286
Table 4.36 Details of the Ten Best Networks Found During GA Run #1 (Generations=100, Population Size=20, refill strategy: cloning) 287
Table 4.37 Details of the Ten Best Networks Found During GA Run #2 (Generations=100, Population Size=20, refill strategy: random) 288
Table 4.38 Details of the Ten Best Networks Found During GA Run #3 (Generations=50, Population Size=40, refill strategy: cloning) 289
Table 4.39 Training and Testing Set Forecasting Errors for the GRNN With No SOM Clustering and the Four GRNNs that Utilised SOM Clustering 290
Table 4.40 Training and Testing Set Forecasting Errors and Training Times for the Single-Sigma GRNN and the Multiple-Sigma GRNN 291
Table 4.41 Training, Testing and Validation Set Forecasting Errors for the Best GRNN and the Best EBMLP Models 292
Table 4.42 Second Validation Set Forecasting Errors for the Best GRNN and the Best EBMLP Models 294
Table 4.43 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Sets of Weights W_j - MLP 297
Table 4.44 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Weight W_j - GRNN 298
Table 4.45 Ranks From P_{jk} Values Given in Table 4.43 - MLP 299
Table 4.46 Ranks From P_{jk} Values Given in Table 4.44 - GRNN 299
Table 4.47 Merit Numbers Assigned to Various Performance Measures For Both ANN Models 300
Table 4.48 Test Set AAPE Statistics for the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 16) 302

Table 4.49 Test Set AAPE Statistics for the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 100) 304
Table 4.50 Default MMAS Parameters 305
Table 4.51 Run Times and Training and Testing Set Forecasting Errors For Each Number of Strata 306
Table 4.52 Run Times and Training and Testing Set Forecasting Errors For Different Population Sizes - 10000 Strata 307
Table 4.53 Training and Testing Set Forecasting Errors For Different Values of The Minimum Pheromone Trail 307
Table 4.54 Training and Testing Set Forecasting Errors for Different Values of the Pheromone Persistence Coefficient 308
Table 4.55 Run Times and Training and Testing Set Forecasting Errors For Different Population Sizes 309
Table 4.56 MMAS Parameters Determined Using Sensitivity Analyses 309
Table 4.57 Comparison of the Training and Testing Set Forecasting Errors and Training Times for the Multiple-Sigma GRNN Trained Using MMAS and the GA 310
Table 4.58 Real-Time Forecasting Test Period Results for the Three MLP and GRNN Retraining Strategies 312
Table 5.1 Physical-Chemical Factors Affecting Cyanobacterial Blooms (source: Paerl, 1996) 331
Table 5.2 Major Classes of Toxins Produced by Cyanobacteria in Australia and Their Significance to Drinking Water Quality and Public Health (source: adapted from Burch, 1999) 347
Table 5.4 Available Data 360
Table 5.5 Statistics of the Available Time Series 361
Table 5.6 Minimum Counting Errors Based on the Number of Anabaena Trichomes Counted (source: Laslett et al., 1997) 364
Table 5.7 ARMCANZ National Cyanobacterial Managerial Alert Level System For Drinking Water Supplies (source: Jones et al., 2003) 366
Table 5.8 BDS Test Z Statistics, Residuals from ARMA Fit to Log(Anabaena spp.) Data Set 383
Table 5.9 Diagnosis Results of the FCS for the Log(Anabaena spp.) Data Set 384
Table 5.10 Statistics of the Training, Testing and Validation Data Sets (Data Divided Using the Genetic Algorithm) 386
Table 5.12 Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set (Data Divided Using the Genetic Algorithm) 388
Table 5.14 Stepwise PMI Input Selection Algorithm Results on Bivariate log(Anabaena spp.) Data Sets (Lagging Window of 26 Weeks) 404

Table 5.15 Stepwise PMI Input Selection Algorithm Results on Multivariate Log(Anabaena spp.) Data Set (Lagging Window of 26 Weeks) 406
Table 5.16 Input Subsets Selected using SOM-GAGRNN 407
Table 5.17 Stepwise PMI Input Selection Algorithm Results on Bivariate Log(Anabaena spp.) Data Sets Transformed Using Histogram Equalization (Lagging Window of 26 Weeks) 408
Table 5.18 Stepwise PMI Input Selection Algorithm Results on Multivariate Log(Anabaena spp.) Data Set Transformed Using Histogram Equalization (Lagging Window of 26 Weeks) 411
Table 5.19 Input Subsets Selected using SOM-GAGRNN (Data Transformed Using Histogram Equalization) 412
Table 5.20 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning) 416
Table 5.21 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning) 419
Table 5.22 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning) 423
Table 5.23 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning) 426
Table 5.24 RMSEs for Each of the Eight ANN Models Investigated 429
Table 5.25 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Weight W_j - GRNN. Note: Performance Measures 4 and 5 Were Excluded From This Study 431
Table 5.26 Ranks From Pjk Values Given in Table 5.25 - GRNN. Note: Performance Measures 4 and 5 Were Excluded From This Study 432
Table 5.27 Default MMAS Parameters 433
Table 5.28 Run Times and Training and Testing Set Performance For Each Number of Strata 434
Table 5.29 Training and Testing Set Performance For Different Values of The Minimum Pheromone Trail 434
Table 5.30 Training and Testing Set Performance for Different Values of the Pheromone Persistence Coefficient 435
Table 5.31 Run Times and Training and Testing Set Performance For Different Population Sizes 436
Table 5.32 MMAS Parameters Determined Using Sensitivity Analyses 436
Table 5.33 Comparison of the Training and Testing Set Performance for the Single-Sigma and Multiple-Sigma GRNNs 437

Table 5.34 Real-Time Forecasting Test Period Results Obtained Using the Single-Sigma GRNN With Different Retraining Strategies (values in brackets are the corresponding t-values for each r^2 value) 443
Table 5.35 A Comparison of the BDS Test Z Statistics Obtained for the Train/Test/Validation Data and the Second Validation Data 445
Table 5.36 A Comparison of the FCS Diagnosis Results for the Train/Test Validation Data and the Second Validation Data 446

Abstract

Artificial neural networks (ANNs) have shown themselves to be a viable alternative to many traditional water resources models, particularly in the field of forecasting hydrologic variables. However, it has been acknowledged by a number of researchers that there is currently no systematic methodology available for the development of ANN models. In this research, a methodology is formulated for the successful design and implementation of ANN models for water resources applications. Attention is paid to each of the steps that should be followed in order to develop an optimal ANN model. These steps include: determining when ANNs should be used in preference to more conventional statistical models; dividing the available data into subsets for modelling purposes; deciding on a suitable data transformation; determination of significant model inputs; choice of network type and architecture; selection of an appropriate performance measure; training (optimisation) of the network's weights; and deployment of the optimised ANN model in an operational environment.

The developed methodology is successfully applied to two water resources case studies, namely, the forecasting of salinity in the River Murray at Murray Bridge, South Australia and the forecasting of cyanobacteria (*Anabaena* spp.) in the River Murray at Morgan, South Australia. These case studies are used to test and illustrate the methods proposed for each step in the methodology. Guidelines for the use of these methods are presented.

Two state-of-the-art statistical tests for nonlinearity are developed as tools for determining when it is appropriate to proceed with an ANN model, in preference to a conventional linear time series model. Using these tests, the salinity and cyanobacteria time series data are both found to exhibit significant nonlinear serial dependence, thereby justifying the use of ANNs for these case studies.

The way in which the available data are divided into training, testing and validation sets has a significant influence on the performance of an ANN model. Therefore, two methods for dividing data into representative subsets are developed, namely, a genetic algorithm (GA) and a self-organizing map (SOM). Both data division methods are able to lead to the development of ANN models that are capable of producing consistent forecasting results.

A number of data transformations are investigated to determine their effect on ANN performance. Three new transformations are proposed, including, a transformation for removing seasonality from data, a transformation for producing normally distributed

data and a kernel transformation for producing uniformly distributed data. The results from this research reveal that the most suitable transformation is case study dependent.

Three analytical techniques are developed to determine significant model inputs for ANNs. The first technique utilises a partial measure of the mutual information criterion to characterise the dependence between a potential model input and the output variable. The remaining techniques utilise unsupervised approaches (i.e. principal component analysis (PCA) and a SOM) to reduce the input dimensionality, and a hybrid GA and general regression neural network (GRNN) to select inputs that have a significant influence on the model's forecast.

Two types of feedforward networks are considered in this research, namely, the multilayer perceptron (MLP) and the GRNN. An evolutionary method for selecting an appropriate architecture for a MLP is investigated and these models are compared with GRNN models for the salinity and cyanobacteria case studies. The GRNN is found to give superior performance when tested on data within the training domain, whereas the MLPs are more robust when uncharacteristic data are encountered. Two architectural modifications are investigated for the GRNN model and a stochastic optimisation technique inspired by the behaviour of ant colonies is developed as an approach for training GRNNs with multiple sigma weights.

A framework is developed for deploying operational ANN models in a real-world environment. Various retraining strategies are investigated and two new methods for updating ANN models in real-time are developed. These methods are able to provide significant improvement in model performance when compared to the scenario of not retraining the model.

The developed methodology results in ANN models that are successful in providing salinity forecasts, 14 days in advance. ANN models are also successfully developed for predicting the timing and relative magnitude of significant growth events of *Anabaena* spp., 4 weeks in advance, provided that representative data are available in the calibration set. For the cyanobacteria case study, the causal variables are found to interact with concentrations of *Anabaena* in a very complex manner and the most important input variables are temperature, flow, pH and previous values of *Anabaena*.

List of Publications

The following is a list of the publications related to the research presented in this thesis:

Book Chapters:

Bowden, G. J., Dandy, G. C., and Maier, H. R. (2003). "An Evaluation of Methods for the Selection of Inputs for an Artificial Neural Network Based River Model." in *Ecological Informatics: Understanding Ecology by Biologically-Inspired Computation*, F. Recknagel, ed., Springer, Berlin, pp.215-232.

Journal Papers:

Bowden, G. J., Dandy, G. C., and Maier, H. R. (2003). "Data transformation for neural network models in water resources applications." *Journal of Hydroinformatics (in press)*.

Bowden, G. J., Maier, H. R., and Dandy, G. C. (2002). "Optimal division of data for neural network models in water resources applications." *Water Resources Research*, 38(2), pp.2-1 - 2-11.

Conference Papers:

Bowden, G. J., Nixon, J. B., Dandy, G. C., Maier, H. R., and Holmes, M. (2003). "Forecasting chlorine residuals in a water distribution system using a general regression neural network." *International Congress on Modelling and Simulation (Modsim 2003)*, Townsville, Australia, July 14-17.

Bowden, G. J., Dandy, G. C., and Maier, H. R. (2002). "Ant colony optimisation of a general regression neural network for forecasting water quality." *Fifth International Conference on Hydroinformatics*, Cardiff, UK, July, Vol. 1, pp.693-698.

ANNEXG - Artificial Neural Network Experiment Group (Bowden G. J. co-author) (2002). "An international comparative study of artificial neural network techniques for river stage forecasting." *8th National Hydrology Symposium*, British Hydrological Society, Birmingham University, 8-11 September, pp 75-81.

Gee, E. M., Mann, E. H. G., Bowden, G. J., Costelloe, J., Puckridge, J., Reid, J. R. W., and Maier, H. R. (2002). "Modelling fish responses to flow variability in Cooper Creek, South Australia: Preliminary results from studies over two time periods." *Riversymposium 2002*, Brisbane, Australia, September 3-6.

Bowden, G. J., Maier, H. R., and Dandy, G. C. (2001). "Understanding the range of applicability of artificial neural network models." *International Congress on Modelling and Simulation (Modsim 2001)*, Canberra, Australia, December 10-13, Vol. 4, pp.1919-1924.

Bowden, G. J., Dandy, G. C., and Maier, H. R. (2000). "Identification of inputs for artificial neural network (ANN) based ecological modelling." Paper presented at *2nd International Conference on Applications of Machine Learning to Ecological Modelling*, Adelaide, Australia, 27 November - 1 December.

Bowden, G. J., Dandy, G. C., and Maier, H. R. (2000). "Use of evolutionary artificial neural network models to forecast concentrations of cyanobacteria in the River Murray." *Proceedings of Hydro 2000: 3rd International Hydrology and Water Resources Symposium of the Institution of Engineers, Australia*, Perth, Australia, November 20-23, Vol. 1, pp.120-125.

Reports:

Bowden, G. J. (2000). "A Review of Cyanobacteria." *Research Report No. R168*, Department of Civil and Environmental Engineering, Adelaide University, Adelaide, Australia, March.

Statement of Originality

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference is made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

SIGNED:.. .....DATE:....30/04/2003...

Acknowledgments

The author wishes to thank his supervisors, Prof. Graeme Dandy and Dr. Holger Maier from the University of Adelaide for their continual support and guidance in all aspects of the project. Their invaluable help was appreciated and significantly enhanced the work completed in this thesis.

This project was funded by an Australian Postgraduate Award (APA) at the University of Adelaide, with assistance from an industry partner, United Utilities Australia Pty. Ltd. This support is gratefully acknowledged.

Thanks must be given to Neil Palmer from United Utilities Australia for his guidance and for sharing his knowledge and experience in many aspects pertaining to water quality and treatment. Thanks also go to all the staff at United Utilities Australia for providing a welcome and friendly environment during the time spent working at their office.

The author would like to thank Mike Burch and Peter Baker from the Australian Water Quality Centre for their many helpful discussions on cyanobacteria and for providing the data set required for the cyanobacteria case study.

Thanks go to Barry Porter from the Berri office of the South Australian Department for Water Resources for providing the River Murray flow and level data used in this research. Thanks also go to Judy Swann from River Murray Water for her assistance in collating data for the salinity and cyanobacteria case studies.

Special thanks to John Vítkovský from the University of Adelaide for his friendship and for the valuable discussions on various aspects of computer modelling and programming. In addition, heart-felt thanks go to my fellow postgraduate students for their support and friendship and for the many happy times we shared together.

I am enormously grateful to my family who have provided me with encouragement throughout the project, particularly when the task seemed challenging.

Finally, the author would especially like to thank his wife, Alison, for her enduring love and support. This was a source of great stability throughout the entirety of the project. Her kindness and patience was inspiring and will be forever appreciated.

Chapter 1

Introduction

1.1 Reasons for this Research

Artificial neural networks (ANNs) are parallel-distributed information processing systems inspired by the functioning of the human brain and central nervous system. Although ANNs were originally intended to model the cognitive processes at work in the brain, their powerful information processing capabilities were realised with such seminal publications as Hopfield (1982) and Rumelhart et al. (1986). These publications placed ANNs on a much firmer mathematical grounding and paved the way for the wide-scale usage of ANNs as a data analytic tool. With the realisation of the power of this computational tool came unprecedented growth in the utilisation of ANNs for solving industrial problems in areas such as forecasting/prediction, pattern recognition, signal processing, classification and control, amongst others. In fact, ANNs have been used in perhaps almost every branch of engineering and science in a wide array of applications ranging from robotics to financial analysis.

The widespread popularity of ANNs is largely because practitioners have perceived that they are able to overcome many of the difficulties involved in implementing traditional

statistical models. Unlike statistical models, the functional form of an ANN does not need to be specified *a priori* and model complexity can be changed simply by altering the architecture of the network. ANNs provide a unifying framework for implementing a number of statistical models (Cheng and Titterington, 1994; Sarle, 1994) and have been shown mathematically to be universal function approximators (Cybenko, 1989; Hornik, 1991; Leshno et al., 1993; White, 1989). Another advantage of ANNs is that they are able to learn the relationships in a data set and generalise without the need for an explicit understanding of the underlying physics of the system being investigated. Such attributes, along with the ability to process large amounts of data at high speed, have contributed to the rapid growth in ANN-related research.

One area in which ANNs have received increasing attention is the field of water resources modelling. Systems involving natural and physical phenomena are typically influenced by a large number of variables that interact in a complex and often highly nonlinear manner and are commonly characterised by considerable spatial and temporal variability. ANNs are well suited to modelling such systems as they are capable of dealing with nonlinearities, are robust in the presence of noise and perform reasonably well when limited data are available (Burke, 1991). Due to a less than perfect understanding of hydrological systems, data-driven techniques such as ANNs have an important role to play in modelling studies (ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b).

Since the early 1990s, ANNs have been applied to an ever-increasing number of water resources case studies, including: rainfall-runoff modelling; precipitation forecasting; streamflow forecasting; subsurface characterisation; streamflow data infilling; groundwater and water quality modelling (ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b; Dawson and Wilby, 2001; Maier and Dandy, 2000b). These applications have clearly demonstrated the advantages of this technology over conventional methods for difficult or seemingly intractable problems. ANNs have also shown their usefulness for solving many new water resources problems. However, the primary emphasis of this previous research has been directed toward the application rather than the methodology.

Many researches in the field of water resources modelling are not "experts" in ANNs and tend to throw problems blindly at them in a type of "...data in, predictions out" mentality (Sarle, 1994). This has led to many important aspects of the model development process being overlooked. Maier and Dandy (2000b) noted that in most hydrologic applications of ANNs, the model building process has been carried out incorrectly and is often poorly documented. Dawson and Wilby (2001) concur with

this general observation and point out that the field has become characterised by poor modelling practice due to the model development steps being carried out in an unsystematic way. Recently, the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a; 2000b) investigated the role of ANNs in hydrology and undertook a review of a number of hydrologic ANN applications. From this study it was acknowledged that for ANN models, "...there appears to be no established methodology for design and successful implementation". In commenting on the future of ANNs in water resources modelling, Maier and Dandy (2000b) state that, "in order to achieve significant advances in the field, there needs to be a change in the mind-set of users, from the application of basic ANN models to an ever-increasing number of case studies to the development of guidelines for ANN modellers".

In this research, a systematic methodology for the successful design and implementation of ANNs for forecasting water resources variables is formulated. It is the intention that this methodology will assist current and potential users of ANNs in harnessing the full capabilities of this powerful predictive tool and in enabling users to apply the technology in an optimal way to their own case studies. The important issues relating to the model development process are addressed and a number of alternatives for implementing each step in the development process are provided. Since ANNs and statistical models are closely related, there is much to gain from adhering to the general principles that are considered good practice in the development of statistical models. However, it is also important to ensure that the primary focus is on obtaining good forecasts, rather than just statistical optimality, since this was one of the features that has attracted so many hydrologists to ANNs in the first place (Maier and Dandy, 2000a). Therefore, where appropriate, statistical principles have been incorporated into the methodology while keeping the focus on achieving good results. The major areas of the model development process covered in this research include: determining when ANN models should be used in preference to more conventional statistical models; dividing the available data into subsets for modelling; data transformation; determination of significant model inputs; choice of network type and architecture; selection of an appropriate performance measure; training (optimisation) of the network's weights; and deployment of the optimised ANN model in an operational environment.

To illustrate the new ANN methodology presented in this research, two water resources case studies are considered. The first case study involves forecasting salinity levels in the River Murray at Murray Bridge, South Australia and the second case study involves forecasting the incidence of a particular genus of nuisance cyanobacteria (*Anabaena* spp.) in the River Murray at Morgan, South Australia. The River Murray is

the major surface water resource of the state of South Australia and provides the domestic and industrial water supply, which is used by 90% of the state's population. Water from the Murray is pumped several hundreds of kilometres to many major South Australian cities, including Adelaide, Port Augusta, Port Pirie, and Whyalla via a number of pipelines. Two of the major offtakes are located at Murray Bridge and Morgan and consequently, these locations are of particular significance.

In the time since European settlement, there has been a decline in the water quality of the River Murray due to large-scale clearance of land, flow regulation, inefficient agricultural methods and poor catchment management. Alteration of the natural state of the aquatic ecosystem has led to the river experiencing high levels of salinity and an increase in the frequency and duration of blooms of cyanobacteria. Both of these issues pose significant concerns for water management authorities. Given the ability to forecast salinity levels and concentrations of nuisance cyanobacteria, several weeks in advance, it would be possible to put in place strategies to better manage this over-exploited resource. A better understanding of salinity and cyanobacteria in the River Murray and the factors that affect their occurrence can be also be obtained from ANN modelling. Using both case studies, a number of numerical experiments have been conducted to illustrate and explore various alternatives in each phase of the ANN modelling methodology.

1.2 Objectives and Scope

The primary objectives of this research include:

1. To develop a robust methodology for the design and successful implementation of ANN models for the forecasting/prediction of water resources variables.

2. To provide guidelines to assist ANN modellers with the following specific tasks:
   - Identifying when the data under investigation require the use of ANNs as distinct from traditional statistical models.
   - Determining how the available data should be divided into training, testing and validation subsets and diagnosing when model performance is likely to fail due to limited data.
   - Selecting an appropriate data transformation for ANN modelling and understanding how different transformations are likely to affect model performance.
   - Determining an appropriate set of model inputs for the application under investigation.

- Choosing an optimal network architecture.
- Evaluating a large number of potential performance measures and selecting the best one for a given case study.
- Deciding on a suitable training (optimisation) algorithm for determining optimal weights within a network.
- Deploying operational ANN models in a real-world environment and dealing with new, uncharacteristic patterns that may arise (i.e. data unlike any of the data used in developing the model).

The secondary objectives of this research relate to the case studies that are used to illustrate the ANN modelling methodology. These secondary objectives include:

i. The evaluation of the usefulness of ANNs as tools for forecasting salinity levels and growth incidences of *Anabaena* spp. in river systems.

ii. To obtain a better understanding of the timing of high salinity levels and significant growth incidences of *Anabaena* spp. in the River Murray, and the causal factors that affect their occurrence.

iii. To develop optimal ANN models that can be used to forecast salinity levels in the River Murray at Murray Bridge, up to 14 days in advance.

iv. To develop optimal ANN models that can be used to forecast *Anabaena* spp. concentrations in the River Murray at Morgan, up to 4 weeks in advance.

1.3 Layout and Contents of Thesis

In Chapter 2, a review of the current state of the art of ANN modelling is presented with particular emphasis on time series forecasting and the use of ANNs in water resources applications. In this chapter, a background (Section 2.1) and introduction (Section 2.2) to ANNs is provided. In Section 2.3, the similarities and differences between ANNs and statistical models are discussed. Section 2.4 reviews the types of ANN models currently in existence and Section 2.5 addresses in detail the important steps in the development of ANN models for time series forecasting. Finally, in Section 2.6, the usage of ANNs in water resources applications is reviewed and some of the critical issues in this field are addressed.

The ANN methodology for modelling water resources variables is presented in Chapter 3. Section 3.1 provides a background to the chapter, while the individual steps within

the modelling methodology are detailed in Sections 3.2 to 3.8. In Section 3.2, two tests for nonlinearity are described and applied to synthetic data. Their relevance for ANN modelling is also discussed. Two new methods for data division are presented in Section 3.3 and the important issue of data transformation is addressed in Section 3.4. Three methods for ANN input determination are formulated in Section 3.5 and applied to synthetic test problems. The choice of an appropriate network is discussed and methods for determining a suitable architecture are presented in Section 3.6. Training (optimisation) is addressed in Section 3.7, including the selection of appropriate performance measures and the choice of optimisation methods. The approaches that can be employed to deploy an operational ANN model in a real-world environment are presented in Section 3.8.

The ANN methodology is applied to the salinity case study in Chapter 4. An introduction to the chapter is given in Section 4.1 and background literature relevant to the case study is reviewed in Section 4.2. The available data for this study are described in Section 4.3, while ANN model development is discussed in Section 4.4. The steps in the methodology are applied to the salinity data in Sections 4.5 to 4.11 and the results and discussion are given in each of these sections. The chapter closes with a summary and description of the conclusions obtained from the application of the methodology to the salinity case study (Section 4.12).

In Chapter 5, the ANN methodology is applied to the cyanobacteria case study. An introduction to this case study is given in Section 5.1, while a detailed review of the literature relating to cyanobacteria is presented in Section 5.2. In Section 5.3, the available data are discussed and the model development procedure used for this case study is given in Section 5.4. Each step in the modelling methodology is applied to the cyanobacteria case study, and the results are presented and discussed in Sections 5.5 to 5.11. The conclusions obtained from the experiments conducted in this chapter are given in Section 5.12.

The conclusions and recommendations of the research are given in Chapter 6. In Section 6.1, the major contributions that have been made are summarised. Section 6.2 outlines general conclusions relating to the ANN modelling methodology, while Section 6.3 presents the specific conclusions that relate to each of the case studies investigated. Finally, recommendations for future areas of investigation are discussed in Section 6.4.

Chapter 2

Review of Artificial Neural Networks

2.1 Background

The quest to match the functionality of the human brain has given rise to the modern form of artificial neural network (ANN) computing. Although modern computers are extremely fast at performing numerical computation and far exceed the capability of the human brain in this function, there are a number of attributes for which the human brain is superior. These include (Warner and Misra, 1996): the ability to quickly identify features, even in the presence of noise; to understand, interpret and act on probabilistic or fuzzy notions (such as "maybe it will rain tomorrow"); to make inferences and judgements based on past experiences and relate them to situations that have never been encountered before; and to suffer localised damage without losing complete functionality (fault tolerance). While these attributes may be the aims of artificial intelligence *per se*, it is generally accepted that the realisation of each of these in modern computational tools is still remote, and the quest continues.

The driving force behind the development of ANNs has been the attempt to imitate the complex, non-linear and parallel mechanisms involved in the interpretation of

information by biological neural networks (Yang et al., 1998). The collective behaviour of an ANN model, like the human brain, demonstrates the ability to learn, recall and generalise from training sets of data. The following definition of an ANN has been offered by Haykin (1994):

"A neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge."

The ability of ANNs to process large amounts of information makes them very useful for tackling a wide variety of real-world problems. In recent times, ANNs have been realised as a powerful computational tool for such applications as environmental modelling, financial analysis, signal processing, medical diagnostics, classification, pattern recognition, process control and optimisation (NeuralWare, 1998b).

The concept of the artificial neuron was first described by McCulloch and Pitts (1943), however, it was not until the backpropagation training algorithm was published by Rumelhart et al. (1986), that research into applications of ANNs expanded significantly. Since then, thousands of papers have been published on ANNs and the growth in ANN research has accelerated at a rapid rate.

2.2 Introduction to Artificial Neural Networks

2.2.1 Analogy to the Brain

To understand the modern form of ANN computing, it is useful to examine the biological inspiration. In a biological neuron (Figure 2.1a), signals are received from other neurons via input structures called dendrites. If the combined signal is strong enough, the neuron is activated and produces an output. The output leaves the neuron along a component called the axon. The axon splits up and connects to the dendrite along a junction called a synapse. Although the output leaves the neuron as an electrical pulse, it is converted to chemical information at the synapse and as the brain learns, the strength of the synapse is modified (NeuralWare, 1998b).

In ANNs, the unit analogous to the biological neuron is called a processing element (PE) (Figure 2.1b). Input paths (dendrites) connect to the PE via weighted connections and the total input stimulus is calculated by summation. This forms an internal activity level for the PE. This activation level is modified as it passes through a transfer function, thereby producing a single output for the PE. The output is then passed to the weighted input connections of many other PEs in the network. In the artificial neuron, the connection weights are analogous to the synaptic strength of a biological neuron since it is these weights that modify the strength of the input stimuli.

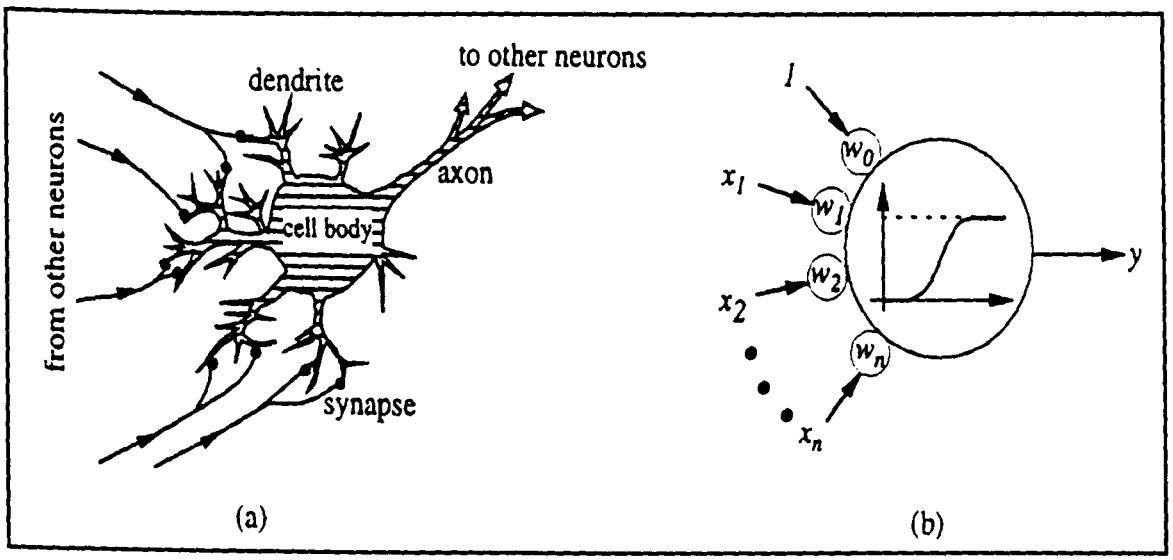

Figure 2.1 (a) A Biological Neuron and (b) An Artificial Neuron Model (PE) (source: Takagi, 1997)

Typical ANNs consists of many processing elements interlinked via weighted connections and arranged in an input layer, one or more hidden layer/s and an output

layer (Figure 2.2). Data propagates through and is operated on by the network, starting with an input stimulus at the input layer and ending with an output stimulus at the output layer (Hecht-Nielsen, 1989).

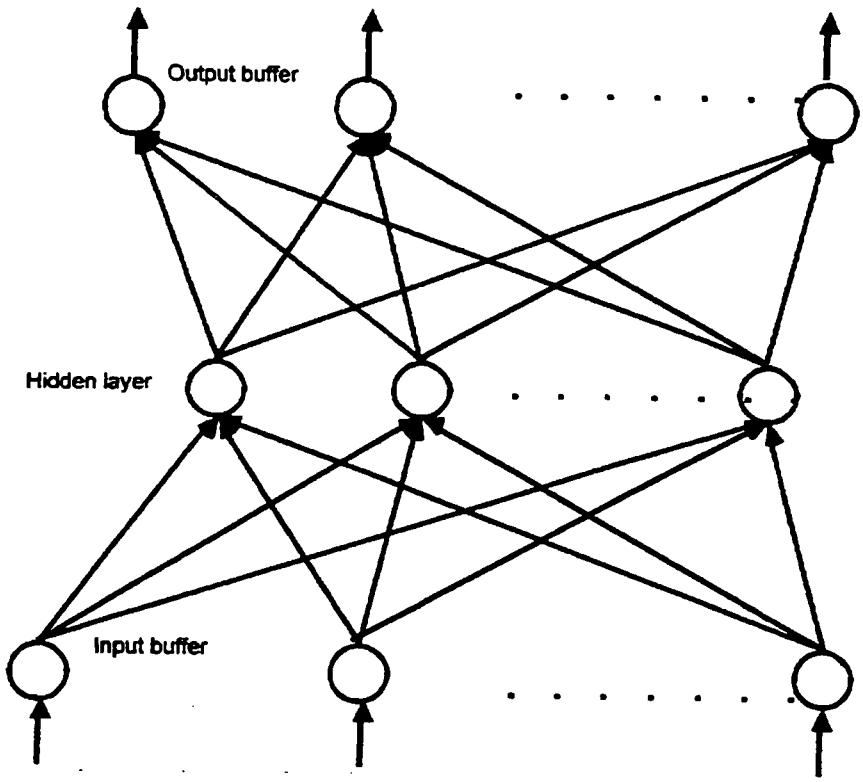

Figure 2.2 A Typical ANN Architecture

2.2.2 History of ANNs

The first research conducted on ANNs commenced in the late nineteenth and early twentieth centuries, when psychologists made attempts to identify the neural basis of intelligence (Mehra and Wah, 1992). The major developments in computation and neuroscience in the early twentieth century came together in the work of McCulloch and Pitts (1943). This seminal paper triggered the first of a pair of revolutions in the history of ANNs. McCulloch and Pitts studied the capabilities of simple ANN models and showed that, in principle, even simple ANNs could compute any arithmetic or logical function.

The next major contribution to the field came from Hebb (1949) who postulated simple rules for associative learning and used these rules to build a qualitative explanation of some experimental results from psychology. This bold step inspired many other researchers to pursue the same theme, which further laid the groundwork for the advent of neural computing (Hecht-Nielsen, 1989). Hebb's proposition stated that the

connection strength between neurons changes as learning occurs and that the connection between two neurons gets stronger if the two neurons are excited simultaneously.

The 1950s heralded a period of strong growth in the field of neural computing and marked the development of several computer models of neural computation. In 1956, the first conference on artificial intelligence was held at Dartmouth. This conference essentially launched the fields of artificial intelligence and neural computing (NeuralWare, 1991). The first successful neurocomputer (the Mark I Perceptron) was developed during 1957 and 1958 by Frank Rosenblatt, Charles Wightman and others. Rosenblatt (1958) showed that if neurons are linked with adjustable strength, they can be trained to perform pattern classifications and can identify both abstract and geometric patterns. The perceptron was a significant departure from the work of McCulloch and Pitts. Perceptrons exhibited distributed representation and predictive power (generalisation), two characteristics that were missing from the McCulloch-Pitts model. Rosenblatt's work on perceptrons sparked a great deal of interest in the rapidly emerging field of neural computing.

Widrow and Hoff (1960) developed the ADALINE (ADAptive LInear NEuron) learning algorithm, which was based on neuron-like elements. The ADALINE and a two layer variant, the MADALINE (Multiple ADALINE) were used for a variety of applications including speech recognition, character recognition, weather prediction and adaptive control (NeuralWare, 1991).

By the mid 1960s neural computing's first revolution marked by its era of initial successes was drawing to a close. The final episode of this era culminated with Minsky and Papert (1969) publishing a book entitled *Perceptrons*, which emphasised the major limitations of the perceptron. It was shown that even basic geometrical tasks could not be performed using a perceptron of reasonable complexity. Due to the linear nature of the perceptron, it was unable to represent the Exclusive OR (XOR) function, which involves separating the binary patterns (0,0), (1,1) from the patterns (1,0), (0,1). To solve this problem required the introduction of a hidden layer of processing elements in between the input and output layers. The outputs of the hidden layer are not available to the outside world. Minsky and Papert (1969) posed the Credit Assignment Problem which dealt with the difficulty of training hidden layer PEs. They summarised this difficulty with the following question: "If the output is in error in a multi-layer network, then how do you determine which interconnection or processing element to adjust?" They also showed that training times increase rapidly for certain problems as the number of inputs increases. These difficulties were unable to be solved adequately at

the time and it was these findings that largely resulted in a drop in the level of funds that were made available for neural computing research.

From 1967 to 1982 very little explicit neural computing research was conducted and the entire field encountered a period of relative quiescence. Despite this, some research into neural networks continued. It was in this "quiet era" that Grossberg (1976) used neurological data to develop neural computing models. One class of networks developed by Grossberg included the Adaptive Resonance Theory (ART) network. Kohonen (1982) was another researcher of this era and was responsible for the introduction of the competitive learning paradigm. In competitive learning, local elements compete to respond to an input stimulus and the winner adapts itself to respond more strongly to that stimulus. The internal organisation of the network is only governed by the inputs, hence, such learning is considered to be unsupervised. The competitive learning paradigm was formulated as a result of a general study on self-organizing maps (NeuralWare, 1991).

Hopfield (1982; 1984) led the renaissance of neural computing and inspired highly qualified scientists, mathematicians and technologists to join the emerging field. The Hopfield Model was a neural computing system consisting of individual processing elements that seek an energy minimum. The operation of the neurons was represented as a thresholding operation and memory was illustrated as information stored in the interconnections between neuron units.

The second revolution in neural computing gained momentum with the increasing power of computers. The speed of modern computers allowed intractable problems to be solved via simulation (Taylor, 1997). But it was the invention of the backpropagation algorithm by Paul Werbos (1974) that contributed to the present day popularity of ANNs. The backpropagation algorithm solves the difficulty of training hidden layer PEs and was independently rediscovered by Parker (1985) and LeCun (1985). The PDP group of Rumelhart et al. (1986) highly publicised the backpropagation algorithm and in so doing, stimulated a wealth of research and resurrected the entire neural network field. The feedforward ANN trained using the backpropagation algorithm is presently the most popular network for current applications of neural computing. The backpropagation algorithm solves the Credit Assignment Problem posed by Minsky and Papert (1969) by assuming that all processing elements and connections are somewhat to blame for an erroneous response. The weights in the hidden layer/s can be adjusted successfully, thereby allowing network to implement the Exclusive OR (XOR) function.

The modern day form of neural computing can be divided into two distinct areas (Taylor, 1997). At one end is the work of those primarily concerned with solving industrial problems by using ANNs as information processing techniques. These people are usually engineers, computer scientists and others in the industrial sector. To these practitioners, ANNs can be seen as one of a number of tools in a toolkit for tackling problems in information processing. At the other end are people interested in understanding living systems, such as biologists, psychologists, philosophers, mathematicians and physicists. These people are generally interested in the whole range of the subject and the interesting new possibilities it presents. In both areas, research growth has been strong over the last decade and because of the variety of avenues currently being explored, it is unlikely to run out of steam as it did in the 1960s and 1970s (Taylor, 1997). In addition to strong research growth, the number of commercial applications of ANNs has also soared and appears to hold much potential for the future. The history of neural computing is discussed in further detail by various authors (see Hecht-Nielsen, 1989; Mehra and Wah, 1992; Taylor, 1997).

2.3 ANNs Compared With Statistical Models

Although research into ANNs had the original goal of creating a plausible model of a biological neural network, researchers soon discovered the power of ANNs as a tool for information processing and data analysis. When ANN models are used as a data analytic tool, they have a similar underlying philosophy to that of many traditional statistical approaches (Maier and Dandy, 2000a). ANN models that learn a desired response via a teacher are referred to as supervised ANNs. In supervised ANN models, the unknown model parameters (connection weights) must be adjusted to achieve the best fit between a set of predicted and observed values. This learning or training phase is no different from the parameter estimation phase in statistical models. In fact, neural networks can be regarded as a type of nonparametric regression model that does not assume any particular functional form but rather, lets the data speak for itself (Warner and Misra, 1996). The branch of statistics that deals with regression analysis has already been recognised to be extended in a positive manner through the use of ANN representations of time series (Breiman, 1996).

In regression, the aim is to model the relationship between the values of the response or dependent variable, denoted as y, and the values of the independent variables, denoted as x_i. McCullagh and Nelder (1989) give the general form of a regression model as:

$$\psi = \sum_{i=0}^{N} \beta_i x_i \tag{2.1}$$

with

$$\psi = h(\mu) \tag{2.2}$$

and

$$E(y) = \mu \tag{2.3}$$

where $h(\cdot)$ is the link function, β_i are the coefficients, N is the number of independent variables, and β_0 is the intercept, hence $x_0 = 1$. The dependent variable y is assumed to have a random component with mean μ and variance σ^2. In addition, the model is assumed to have a systematic component relating the independent variables x_i to a linear predictor and a link function that relates the mean to the linear predictor $\psi = h(\mu)$. If it is assumed that the random component has a normal distribution with a mean of zero and variance σ^2 and that the link function $h(\cdot)$ is the identity function, then the

generalized linear model (GLM) reduces to a multiple linear regression model. The coefficients β_i, are found such that they minimise the sum of squared errors. This involves using a dataset that includes examples of the dependent and independent variables.

The multiple linear regression model is akin to a two-layer feedforward ANN model (Figure 2.3). In this model, the independent variables are the inputs to the network, the coefficients are the connection weights, the intercept corresponds to the bias, the transfer function in the ANN's processing element is the identity function and the dependent variable y corresponds to the network output. An iterative algorithm is used to find the optimal weights in the ANN model such that an objective function (e.g. the sum of squared errors) is minimised. This is in contrast to the multiple linear regression model which has a closed form solution for the coefficients. However, like the multiple linear regression model, a dataset that includes examples of the dependent (input) and independent (output) variables is used to find the appropriate connection weights. In the terminology of the ANN literature, this dataset is called a training set.

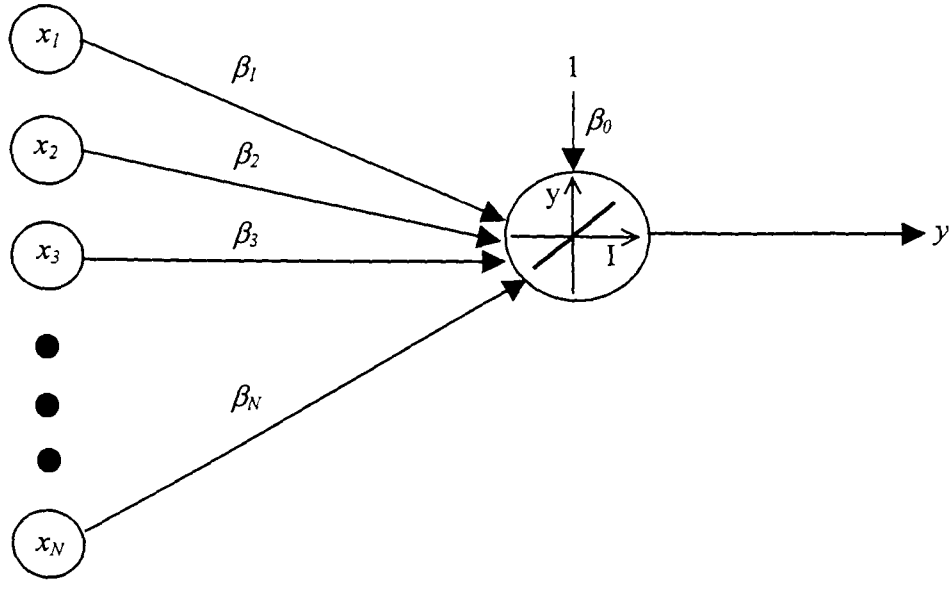

Figure 2.3 A Two-Layer Feedforward ANN Representation of a Multiple Linear Regression Model

It is possible to construct an equivalent ANN model for any GLM by setting the transfer function of the ANN as the inverse of the GLM's link function (Sarle, 1994; Warner and Misra, 1996). In order to model nonlinear processes, nonlinear transfer functions can be used in the ANN model. In addition, to increase the complexity of the ANN model, a third layer between the input and output layer can be added. The third layer is referred to as a hidden layer. Three-layer feedforward ANN models with a suitable

number of hidden units and nonlinear sigmoid transfer functions have been shown to be universal function approximators (Cybenko, 1989; Hornik et al., 1989). In addition to universal function approximation, an added advantage of an ANN model over a GLM is that the functional form for the model does not need to be specified *a priori*. Consequently, an ANN model is a powerful tool when the functional relationship between the independent and dependent variables is unknown *a priori* (Warner and Misra, 1996).

In addition to all forms of regression models, other statistical models that can be represented by ANN models include; discriminant analysis, linear discriminant functions, finite mixture models, classification trees, smoothing splines and time series models such as ARMA and other forecasting models (Cheng and Titterington, 1994; Sarle, 1994; Warner and Misra, 1996). Even though some ANN models have similarities to many standard statistical approaches, they are extremely valuable as they provide a flexible form of implementing these models (Maier and Dandy, 2000a). Model complexity can be changed simply by varying the transfer function or the network architecture. However, the underlying approach adopted in ANN modelling is fundamentally different to that of statistical modelling. This is primarily because ANNs were developed by computer scientists and engineers, rather than statisticians. ANN practitioners place a strong emphasis on prediction accuracy and have the goal of developing methods that work for real-world problems, whereas the main objective of statisticians is to develop universal methodology and to achieve statistical optimality (Maier and Dandy, 2000a). Consequently, the dimensionality of ANN models tends to be higher and they are often used to tackle more complex problems. Despite relying on iterative algorithms and being slower to converge than many of their statistical counterparts, the rapid advances in the speed of modern day computer processors has allowed ANN models to be realised as a practical alternative to the more traditional statistical techniques.

2.4 Types of ANN Models

ANNs can be categorised on the basis of the learning mechanism utilised (i.e. supervised or unsupervised), how the data flows through the network (i.e. feedforward or feedback) and by the type of data they use (i.e. categorical and quantitative) (Lippmann, 1987; Sarle, 1997).

Learning is the process in which the connection weights in the network are adapted in response to the input and/or output stimulus. If an output stimulus is presented, this must correspond to a desired response to a given input and a knowledgeable teacher must provide this desired response (NeuralWare, 1998b). This type of learning is referred to as *supervised learning*. If no desired output is shown, the learning is called *unsupervised learning*. In unsupervised learning, weights are adjusted according to the input values to the network. Most unsupervised ANNs are used for data compression, such as dimensionality reduction or clustering.

Typical network structures for supervised ANNs include the feedforward and feedback network types. In most cases, feedforward networks consist of layered architectures and have feedforward connections which only allow information to flow from the input to the output side (Figure 2.4a) (Maren et al., 1990), whereas feedback networks generally have connections between network outputs and some or all other PEs (Figure 2.4b) (Maren et al., 1990; Takagi, 1997). In feedback networks, information is free to flow through the PEs in both directions. There is no guarantee that feedback networks will reach stability as they are more susceptible to becoming divergent during the training procedure (Takagi, 1997).

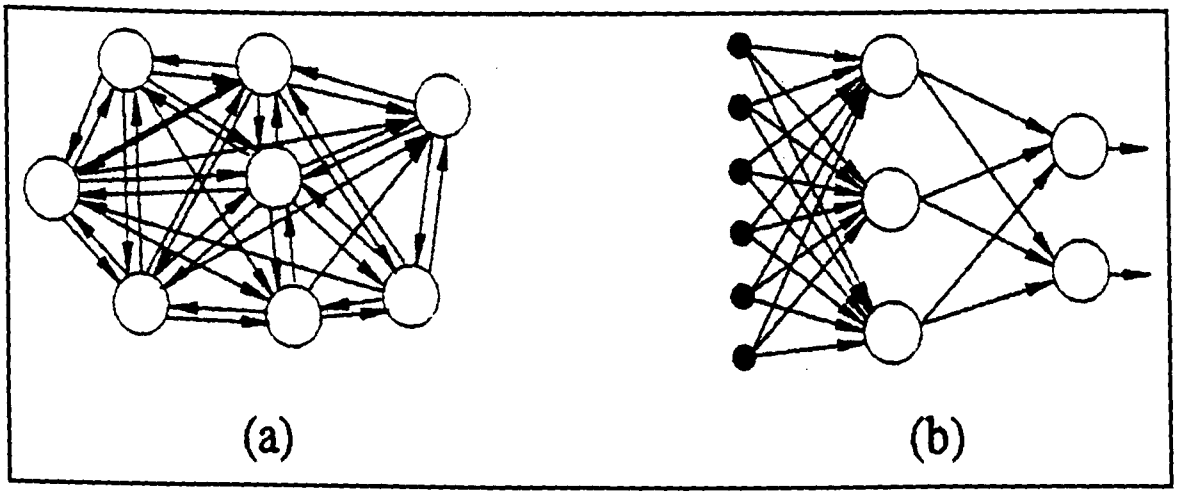

Figure 2.4 (a) A Feedforward Neural Network and (b) A Feedback Neural Network (source: Takagi, 1997)

Another distinction between different types of ANNs can be made based on the type of data they accept. The two major kinds of data are categorical and quantitative. Categorical data only take a finite number of possible values, usually with several examples falling in each category. Unsupervised and supervised ANNs with categorical outputs are primarily used for classification. Quantitative data are continuous-valued and these variables are usually numerical measurements of some attribute. Supervised ANNs with quantitative outputs are generally used for regression.

There are many different types of ANN models, and new models are being reported in the literature on an ongoing basis. The most well known ANN models include (Sarle, 1997):

A. Supervised

1. Feedforward
   - Linear
     - Hebbian - Hebb (1949), Fausett (1994)
     - Perceptron - Rosenblatt (1958), Minsky and Papert (1969), Fausett (1994)
     - ADALINE - Widrow and Hoff (1960), Fausett (1994)
     - Higher Order - Bishop (1995)
     - Functional Link - Pao (1989)
   - Multilayer Perceptron (MLP) - Bishop (1995), Reed and Marks (1999), Fausett (1994)
     - Backpropagation - Rumelhart et al. (1986)
     - Cascade Correlation - Fahlman and Lebiere (1990), Fausett (1994)
     - QuickProp - Fahlman (1989)
     - RProp - Riedmiller and Braun (1993)
   - Radial Basis Function (RBF) networks - Bishop (1995), Moody and Darken (1989)
     - Orthogonal Least Squares (OLS) - Chen et al. (1991)
   - Cerebellar Model Articulation Controller (CMAC) - Albus (1975), Brown and Harris (1994)
   - Evolutionary ANNs (EANNs) - Yao and Liu (1998b), Whitehead and Choate (1996), Angeline et al. (1994)
   - Classification only
     - Learning Vector Quantization (LVQ) - Kohonen (1988), Fausett (1994)

- Probabilistic Neural Network (PNN) - Specht (1990)

- Regression only
  - General Regression Neural Network (GRNN) - Specht (1991)

2. Feedback - Hertz et al. (1991), Medsker and Jain (2000)
   - Bidirectional Associative Memory (BAM) - Kosko (1992), Fausett (1994)
   - Boltzman Machine - Ackley et al. (1985), Fausett (1994)
   - Recurrent time series
     - Backpropagation through time - Werbos (1990)
     - Elman - Elman (1990)
     - Finite Impulse Response (FIR) - Wan (1990)
     - Jordan - Jordan (1986)
     - Real-time recurrent network - Williams and Zipser (1989)
     - Recurrent backpropagation - Pineda (1989), Fausett (1994)
     - Time Delay NN (TDNN) - Lang et al. (1990)

3. Competitive
   - ARTMAP - Carpenter et al. (1991a)
   - Fuzzy ARTMAP - Carpenter et al.(1992), Kasuba (1993)
   - Gaussian ARTMAP - Williamson (1995)
   - Counterpropagation - Hecht-Nielsen (1987; 1988; 1989), Fausett (1994)
   - Neocognitron - Fukushima et al. (1983), Fukushima, (1988), Fausett (1994)

B. Unsupervised - Hertz (1991)

1. Competitive
   - Vector Quantization
     - Grossberg - Grossberg (1976)
     - Kohonen - Kohonen (1988)
     - Conscience - Desieno (1988)
   - Self-Organizing Map
     - Kohonen - Kohonen (1982), Fausett (1994)
     - Generative Topographic Mapping (GTM) - Bishop et al. (1997)
     - Local Linear - Mulier and Cherkassky (1995)
   - Adaptive resonance theory
     - ART 1 - Carpenter and Grossberg (1987b), Fausett (1994)
     - ART 2 - Carpenter and Grossberg (1987a), Fausett (1994)

- ➢ ART 2-A - Carpenter et al. (1991b)
- ➢ ART 3 - Carpenter and Grossberg (1990)
- ➢ Fuzzy ART - Carpenter et al. (1991c)

- Differential Competitive Learning (DCL) - Kosko (1992)

2. Dimension Reduction - Diamantaras and Kung (1996)
   - Hebbian - Hebb (1949), Fausett (1994)
   - Oja - Oja (1989)
   - Sanger - Sanger (1989)
   - Differential Hebbian - Kosko (1992)

3. Autoassociation
   - Linear autoassociator - Anderson et al. (1977), Fausett (1994)
   - Brain State in a Box (BSB)- Anderson et al. (1977), Fausett (1994)
   - Hopfield - Hopfield (1982), Fausett (1994)

C. Nonlearning

1. Hopfield - Hertz et al. (1991)

2. Optimisation Networks - Cichocki and Unbehauen (1993)

The following discussion is restricted to the types of ANN models used in this research. These include: multilayer perceptrons, general regression neural networks and evolutionary ANNs, which can all be classified as supervised feedforward ANNs; and self-organizing maps, which can be classified as an unsupervised ANN that uses competitive learning.

2.4.1 Supervised, Feedforward ANNs

Feedforward ANNs are by far the most common type of ANN employed and are usually trained with a supervised learning algorithm. In feedforward ANNs, the input stimulus can only travel in one direction during operation and all PEs in one layer are only connected to PEs in the next layer. Figure 2.5 shows a simple numerical example of the data flow in a feedforward ANN.

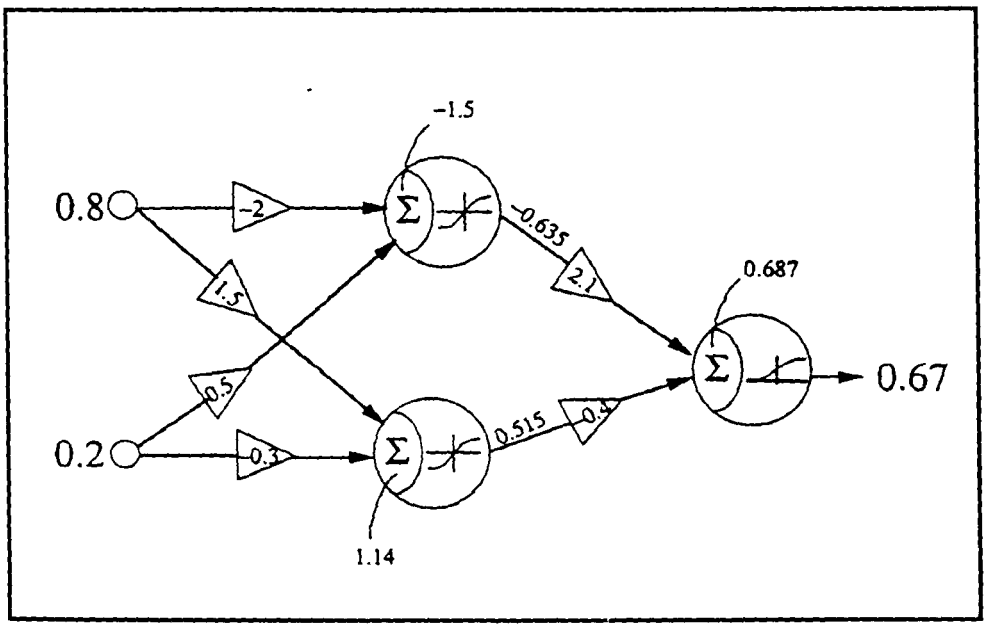

Figure 2.5 A Simple Feedforward Neural Network Example (source: Takagi, 1997)

Feedforward ANNs with multiple layers have been shown to be universal function approximators under specific conditions. Hornik (1991) proved the following general result:

"Whenever the activation function is continuous, bounded and nonconstant, then for an arbitrary compact subset $X \subseteq R^n$, standard multilayer feedforward networks can approximate any continuous function on X arbitrarily well with respect to uniform distance, provided that sufficiently many hidden units are available".

A more general result was proved by Leshno et al. (1993):

"A standard multilayer feedforward network with a locally bounded piecewise continuous activation function can approximate *any* continuous function to *any* degree of accuracy if and only if the network's activation function is not a polynomial".

Similar results have also been shown by Cybenko (1989) and White (1989). The most common form of feedforward ANN is the multilayer perceptron (MLP).

2.4.1.1 Multilayer Perceptron (MLP)

A MLP consists of a network of simple PEs called perceptrons and is the most prevalent ANN in use today (Dawson and Wilby, 2001). Each perceptron is able to compute a single output from multiple real-valued inputs. To do so, the perceptron

computes a linear combination of the inputs according to their input weights and then passes this activation level through a transfer function to produce its output.

A MLP can be trained using a variety of conventional numerical methods (e.g. conjugate gradients, Levenberg-Marquardt algorithm) but the most popular algorithm for training MLPs is the backpropagation algorithm. This algorithm utilises a learning rule known as the generalised delta rule, which is based on the method of steepest gradient descent. Feedforward MLPs trained by this algorithm are commonly referred to as backpropagation ANNs. Although originally proposed by Werbos (1974) in his Ph.D. dissertation at Harvard University, the potential of the backpropagation algorithm for training feedforward MLPs was not realised until Rumelhart et al. (1986) rediscovered the algorithm and showed its ability to solve the Credit Assignment Problem posed by Minsky and Papert (1969). Rumelhart et al. (1986) showed that the Credit Assignment Problem could be solved by assigning part of the blame for erroneous outputs to all PEs (NeuralWare, 1998b).

A typical backpropagation ANN consists of an input layer, one or more hidden layer/s and an output layer. There is no theoretical rule for choosing the number of hidden layers, however, it has been shown that only one hidden layer is required to approximate any continuous function, given that sufficient degrees of freedom (i.e. connection weights) are provided (Cybenko, 1989). In addition, for classification problems, there is evidence that a maximum of three hidden layers are required to solve arbitrarily complex problems (NeuralWare, 1998b). Each layer in the network is fully connected to the succeeding layer and information initially flows forward through the network. At each PE in a layer, the information is received, stored and processed and is then propagated on to PEs in the next layer. To initialise all of the weights within the network, they are set to small random values before training commences. Once the training process begins, the weights are updated iteratively by using the generalised delta rule. The weight update process involves propagating information back through the network. Therefore, the training of a backpropagation ANN involves two distinct phases: a forward pass of the information through the network from the input layer to the output layer; and a backward pass, in which the error from the output layer is backpropagated to the input layer so that the connection weights can be updated. The training process ceases when some termination criterion is satisfied, e.g. a fixed number of iterations have been reached, or there is no further improvement in the error measured on a test set. The standard backpropagation algorithm procedure for an ANN with one hidden layer can be summarised as follows (Fausett, 1994; NeuralWare, 1998b; Smith, 1993):

- Step 0. Initialise weights (set to small random values).

- Step 1. While stopping criterion is false, do Steps 2-9.

- Step 2. For each training sample of the set, do Steps 3-8.

Feedforward:

- Step 3. Each input unit (X_i, i= *1,2, ..., N*) receives input signal x_i, and sends this signal to all units in the next layer (the hidden PEs).

- Step 4. Each hidden PE (Z_j, j= *1,2, ..., p*) sums its weighted input signals:

$$zin_j = v_{oj} + \sum_{i=1}^{N} x_i v_{ij} \tag{2.4}$$

where v_{oj} and v_{ij} are the bias and connection weights from the input layer to the hidden layer, respectively. The transfer function is then applied to compute the output signal:

$$z_j = f(zin_j) \tag{2.5}$$

and the PE then sends this signal to all PEs in the following layer ie. the output layer. Often, the transfer function $f(\cdot)$ is the sigmoid nonlinear function as denoted by:

$$f(x) = \frac{1}{1+\exp(-x)} \tag{2.6}$$

- Step 5. Each output PE (Y_k, k= *1,2, ..., m*) sums its weighted input signals:

$$yin_k = w_{ok} + \sum_{j=1}^{p} z_j w_{jk} \tag{2.7}$$

where w_{ok} and w_{jk} are the bias and connection weights from the hidden layer to the output layer, respectively. The transfer function is then applied to compute the output signal:

$$y_k = f(yin_k) \tag{2.8}$$

Backpropagation of error:

- Step 6. Each output layer PE (Y_k, k= *1,2, ..., m*) has a desired output d_k corresponding to the input training pattern and a measure of the error in achieving this desired output is given by:

$$E = 0.5\sum_{k=1}^{m}\left[(d_k - y_k)^2\right] \tag{2.9}$$

Other error functions whose derivatives can be calculated at the output layer can be substituted for the standard one given in Equation 2.9.

The critical parameter that is passed back to the hidden layer is the error information term e_k, which is given by:

$$\begin{aligned} e_k &= -\frac{\partial E}{\partial yin_k} \\ &= -\frac{\partial E}{\partial y_k}\cdot\frac{\partial y_k}{\partial yin_k} \\ &= (d_k - y_k)f'(yin_k) \end{aligned} \tag{2.10}$$

where $f'(yin_k)$ is the derivative of the sigmoid function and can be expressed as a simple function of itself as follows:

$$f'(yin_k) = f(yin_k)[1 - f(yin_k)] \tag{2.11}$$

The aim of the learning process is to minimise the global error of the system by modifying the weights. To do so, the gradient of the error surface with respect to the weights is used:

$$\delta_{jk} = \frac{\partial E}{\partial w_{jk}} \tag{2.12}$$

The weight correction term for weights from the hidden layer to the output layer can then be calculated as follows:

$$\begin{aligned} \Delta w_{jk} &= -\eta\delta_{jk} \\ &= -\eta\frac{\partial E}{\partial w_{jk}} \end{aligned} \tag{2.13}$$

where η is a learning coefficient and each weight is changed according to the size and direction of the negative gradient of the error surface. The learning rate η determines the absolute size of the weight change during learning and is generally determined using a trial-and-error process. It determines how fast the network learns by adjusting the magnitude of the steps taken down the error surface (Maier, 1995). The partial derivative in Equation 2.13 can be calculated using:

$$\frac{\partial E}{\partial w_{jk}} = \left(\frac{\partial E}{\partial yin_k}\right) \cdot \left(\frac{\partial yin_k}{\partial w_{jk}}\right) = -e_k z_j \tag{2.14}$$

Combining Equations 2.13 and 2.14 gives:

$$\Delta w_{jk} = \eta e_k z_j \tag{2.15}$$

e_k is then passed back to PEs in the previous layer.

- Step 7. Each hidden PE (Z_j, j= *1,2, ..., p*) sums the error information terms from PEs in the output layer:

$$ein_j = \sum_{k=1}^{m} e_k w_{jk} \tag{2.16}$$

then multiplies the above summation by the derivative of its transfer function to calculate its error information term:

$$e_j = ein_j f'(zin_j) \tag{2.17}$$

and then calculates its weight correction term:

$$\Delta v_{ij} = \eta e_j x_i \tag{2.18}$$

Update weights and biases:

- Step 8: Each output PE (Y_k, k= *1,2, ..., m*) updates its bias and weights (j=*0,1, ..., p*):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} = w_{jk}(old) + \eta\, e_k z_j \tag{2.19}$$

Each hidden PE (Z_j, $j = 1,2, \ldots, p$) updates its bias and weights ($i=0,1, \ldots, N$):

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} = v_{ij}(old) + \eta\, e_j x_i \tag{2.20}$$

- Step 9. Test stopping criterion

Variations of the standard algorithm

The standard backpropagation algorithm has a number of deficiencies, which include (Maier, 1995): long training times; susceptibility to overtraining, which leads to poor generalisation; a tendency to get stuck in local minima of the error surface and therefore, a global minimum is not guaranteed; the assumption that all patterns are equally important during training; the need for a large amount of training data, and; sensitivity to the control parameters such as the learning rate. To address these deficiencies, there have been a number of improvements made since backpropagation was reinvented by Rumelhart et al. (1986). Some of the improvements relative to this research are discussed below.

Momentum term (NeuralWare, 1998b; Smith, 1993)

One difficulty with the backpropagation algorithm is the determination of an appropriate value for the learning rate. When the error surface exhibits a high degree of curvature, a small learning rate is required to ensure that divergent behaviour does not occur. Conversely, when the error surface is shallower, larger learning rates are necessary to ensure that learning occurs at an acceptable rate. To resolve this dichotomy the momentum term was introduced. Including the momentum term modifies Equations 2.15 and 2.18 as follows:

$$\Delta w_{jk}(new) = \eta\, e_k z_j + \mu \Delta w_{jk}(old) \tag{2.21}$$

$$\Delta v_{ij}(new) = \eta\, e_j x_i + \mu \Delta v_{ij}(old) \tag{2.22}$$

where μ is the momentum parameter and lies in the range $0 \le \mu < 1$. Momentum is a form of averaging of the weight changes, which tends to reinforce general trends whilst canceling out oscillatory behaviour.

Cumulative update of weights

To improve the convergence speed of the standard backpropagation algorithm, it is possible to only update the weights after a fixed number of training samples have been presented to the network, rather than after every sample. This process is referred to as cumulative backpropagation since the weight changes are accumulated until a fixed number (or batch) of samples have been presented to the network. This gives the opportunity to ensure that the examples in each batch presented to the network are statistically balanced and represent a range of input-output patterns. This increases the speed of training and makes the training process more reliable. The fixed number of training samples presented during the accumulation of weight changes is referred to as the epoch size. The network's weights are updated at the end of each epoch.

Adaptive learning rates and momentum (NeuralWare, 1998b; Smith, 1993)

The method of adaptive learning rates was invented by Jacobs (1988), under the name *delta-bar-delta* (DBD), and was designed to increase the convergence speed of the standard backpropagation algorithm. In this method, the assumption is made that a learning rate that is suitable for one weight dimension may not be appropriate for all weight dimensions. Therefore, the method assigns each weight within the network a different learning rate. The learning rates are adjusted according to the following heuristic: if the direction in which the error decreases at the current weight change is the same as the direction in which it has been decreasing recently, then the learning rate is made larger (Figure 2.6); if the direction in which the error currently decreases is the opposite of the recent direction, then the learning rate is made smaller (Figure 2.7).

Like the standard backpropagation algorithm, the gradient component is used. This is given by:

$$\delta(k) = \frac{\partial E(k)}{\partial w(k)} \tag{2.23}$$

where $\delta(k)$ is the gradient component of the weight change at time k, $E(k)$ is the value of the error at time k and $w(k)$ is the connection weight at time k. Using the generalised delta rule, the weight update can be written as:

$$w(k+1) = w(k) + \eta\delta(k) \tag{2.24}$$

where the learning rate η, is a fixed constant. In the method of adaptive learning rates, η is variable and is denoted as $\eta(k)$. The weight update equation becomes:

$$w(k+1) = w(k) + \eta(k)\delta(k) \tag{2.25}$$

Each learning rate must be changed depending on the direction in which the error has been decreasing "recently". Jacobs (1988) achieved this by using an exponential average of the gradient components $\delta(k)$. This average is defined as:

$$\bar{\delta}(k) = (1-\theta)\delta(k) + \theta\delta(k-1) \tag{2.26}$$

where θ is the parameter that controls how long "recently" means and lies in the range $0 \le \theta < 1$. As θ approaches 0, the average approaches the current value of $\delta(k)$ and as θ approaches 1, the current value of $\delta(k)$ counts less and the past values of the gradient, which have been averaged into $\bar{\delta}(k)$, count more.

If the current gradient component has the same sign as the exponential average of the previous gradient components, the learning rate is incremented by a constant κ. If the current gradient component is of a different sign to the exponential average, the learning rate is reduced by an amount that is proportional to its current value. These heuristic rules are defined mathematically in the equations below:

$$\Delta\eta(k) = \begin{cases} \kappa & \text{if } \bar{\delta}(k-1)\delta(k) > 0 \\ \phi\eta(k) & \text{if } \bar{\delta}(k-1)\delta(k) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

where κ and ϕ are constants and $0 < \phi < 1$.

When using the DBD algorithm, the performance of the network is not very sensitive to the choice of values for θ, κ and ϕ and recommended values that work well across a wide variety of problems have been given by Smith (1993) who proposed using: θ= 0.7, κ= 0.1 and ϕ= 0.5. The DBD algorithm has the advantage of providing substantial increases in the speed of learning when compared to the standard backpropagation algorithm, however, the method does have its shortcomings, which include:

- The standard DBD algorithm does not use the momentum heuristic.
- The small linear increments κ may cause the learning rate to increase to large values such that large jumps are taken in weight space.

- The geometric decrease in the learning parameter is not always fast enough to prevent erratic jumps in weight space.

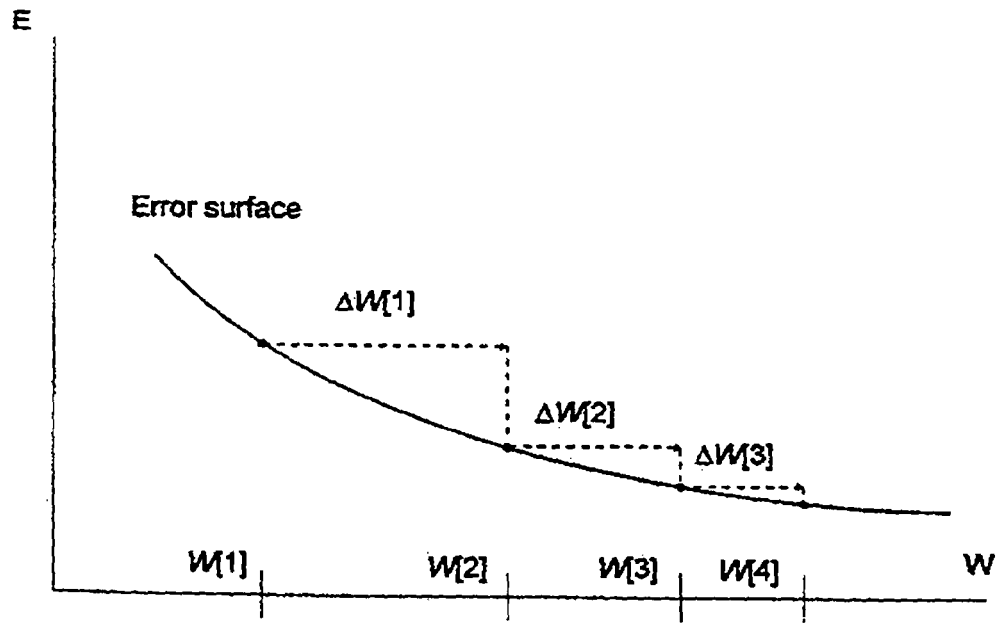

Figure 2.6 Scenario For Increasing the Learning Rate for a Connection Weight. The Weight Changes Have the Same Sign For Several Time Steps Over a Region of Relatively Low Curvature (source: NeuralWare, 1998b)

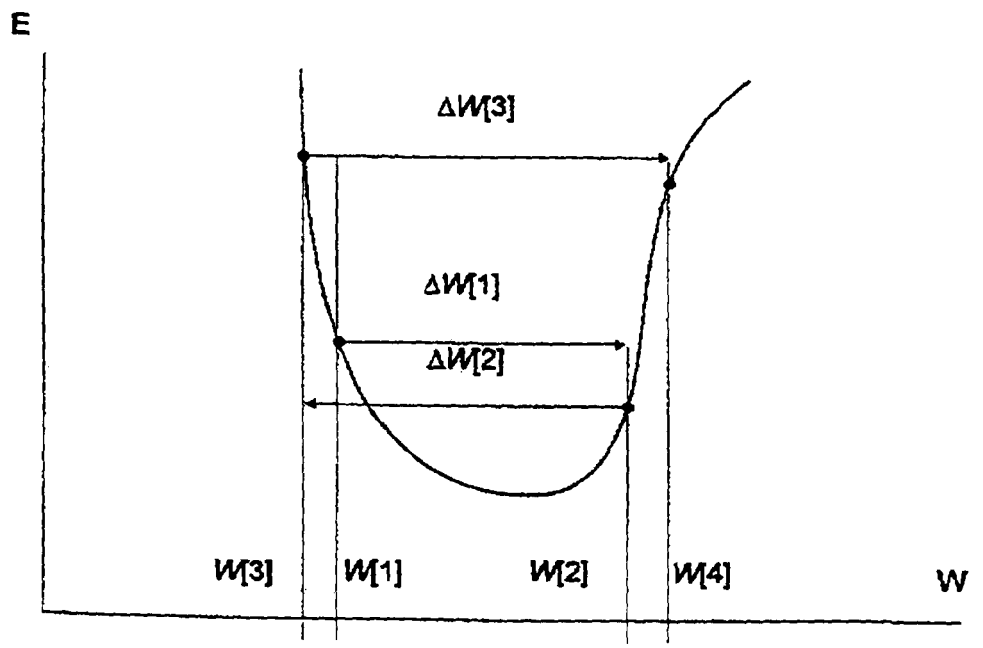

Figure 2.7 Scenario For Decreasing the Learning Rate for a Connection Weight. The Weight Changes Have Opposite Signs for Several Time Steps Over a Region of Relatively High Curvature (source: NeuralWare, 1998b)

The method of an adaptive momentum was invented by Minai and Williams (1990), under the name *extended delta-bar-delta* (EDBD), and was designed to address the shortcomings of the DBD algorithm. In EDBD, the momentum is a function of time with each weight in the network having a different momentum value in much the same way that the DBD algorithm assigns a different learning rate for each weight. The use of a variable learning rate and momentum in the EDBD algorithm gives the following weight update equation:

$$\Delta w(k+1) = \eta(k)\delta(k) + \mu(k)\Delta w(k) \tag{2.28}$$

where $\mu(k)$ is the adaptive momentum at time k. The momentum and learning rate are adjusted according to the following heuristic:

Firstly, the exponential average of the gradient components $\delta(k)$ is calculated in exactly the same way as in the DBD algorithm:

$$\bar{\delta}(k) = (1-\theta)\delta(k) + \theta\delta(k-1) \tag{2.29}$$

Then the learning rate is changed using:

$$\Delta\eta(k) = \begin{cases} \kappa_\eta \exp(-\gamma_\eta \left|\bar{\delta}(k)\right|) & \text{if } \bar{\delta}(k-1)\delta(k) > 0 \\ -\phi_\eta \eta(k) & \text{if } \bar{\delta}(k-1)\delta(k) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

where κ_η is a constant learning rate scale factor, γ_η is a constant learning rate exponential factor and ϕ_η is a constant learning rate decrement factor.

Similarly, the momentum is then changed using:

$$\Delta\mu(k) = \begin{cases} \kappa_\mu \exp(-\gamma_\mu \left|\bar{\delta}(k)\right|) & \text{if } \bar{\delta}(k-1)\delta(k) > 0 \\ -\phi_\mu \eta(k) & \text{if } \bar{\delta}(k-1)\delta(k) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

where κ_μ is a constant momentum rate scale factor, γ_μ is a constant momentum rate exponential factor and ϕ_μ is a constant momentum rate decrement factor.

In the EDBD algorithm, the learning rate and momentum increases were changed from the original DBD algorithm, to be exponentially decreasing functions of the magnitude of the weighted gradient components $|\bar{\delta}(k)|$. This has the effect of giving greater increases in regions where the error surface has smaller curvature than in regions of higher curvature.

To prevent any erratic jumps and oscillations in weight space, an upper limit is placed on all learning rates and momentum values. This is expressed mathematically as:

$$\eta(k) \leq \eta_{\max} \tag{2.32}$$

and

$$\mu(k) \leq \mu_{\max} \tag{2.33}$$

The final step in the EDBD algorithm is to check the accumulated error after each epoch. If the error $E(k)$ is less than the previous minimum error E_{min}, the weights are saved to memory as the current best. A tolerance parameter λ is introduced such that if the current error exceeds the minimum error in accordance with:

$$E(k) > E_{\min}\lambda \tag{2.34}$$

then all weights in the network revert back to the previous best set of weights stored in memory.

Different transfer functions

Alternative transfer functions can be used in place of the sigmoid function, provided that they are differentiable. Bounded functions are advantageous. One transfer function that is widely used is the hyperbolic tangent function, which is a bipolar version of the sigmoid function. The sigmoid function is defined in Equation 2.6 and is a smooth version of a [0,1] step function (Figure 2.8). The hyperbolic tangent function is given by:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.35}$$

and is a smooth version of a [-1,1] step function (Figure 2.9). Like the sigmoid function, the derivative of the hyperbolic tangent function can be expressed in terms of itself:

$$f'(x) = [1 + f(x)] \cdot [1 - f(x)] \tag{2.36}$$

and hence, can readily be incorporated into the backpropagation equations.

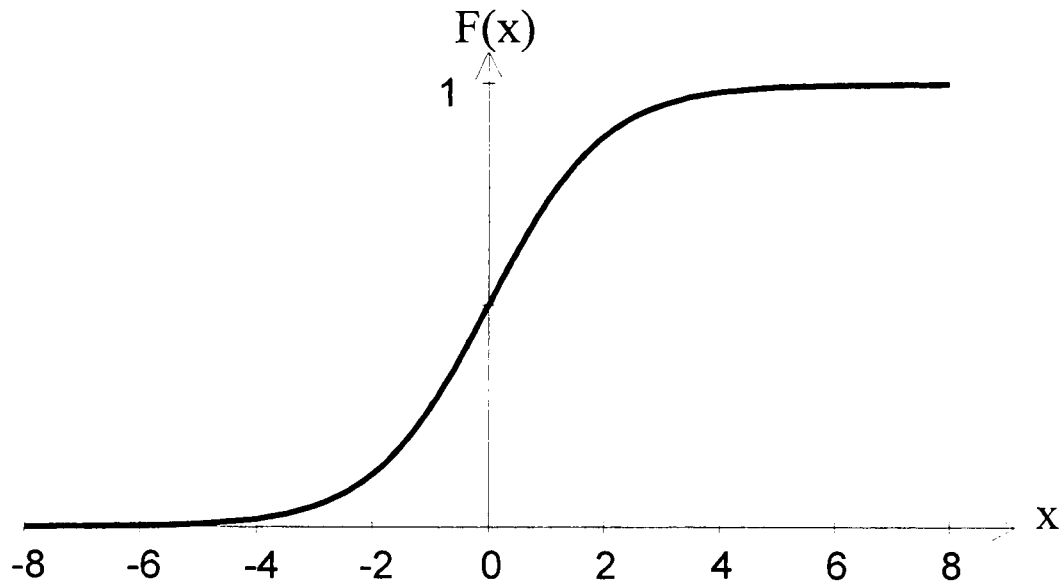

Figure 2.8 The Sigmoid Function

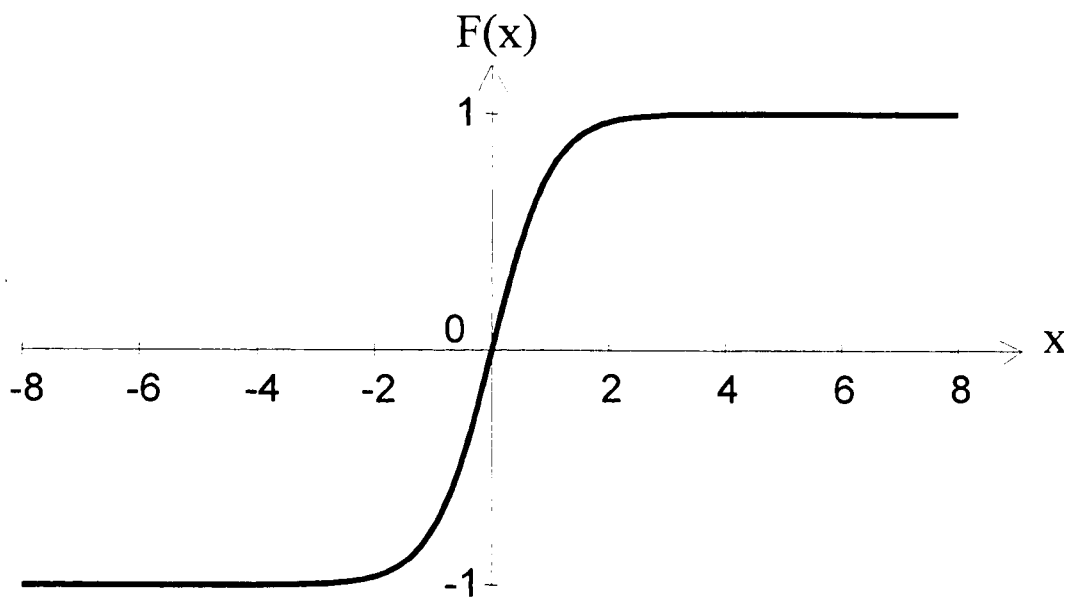

Figure 2.9 The Hyperbolic Tangent Function

There are other functions that may also be used as transfer functions. Some of these include (Imrie et al., 2000; NeuralWare, 1998a):

- Linear

$$f(x) = ax + 0.5 \tag{2.37}$$

The linear function with a gradient specified by the parameter a, simply sums up all inputs to the output layer.

- Sigmoid + linear

$$f(x) = \frac{1}{1+\exp(-\beta x)} + \xi x \tag{2.38}$$

Equation 2.38 combines the advantages of nonlinearity and the ability to produce any output. The parameters β and ξ control the slopes of the sigmoid and linear terms, respectively.

- Cubic polynomial

$$f(x) = \varepsilon x^3 + \varphi x + 0.5 \tag{2.39}$$

The parameters ε and φ control the slopes of the cubic and linear terms.

- Sine function

$$f(x) = \sin(ax) \tag{2.40}$$

This simply takes the trigonometric sine of the input and the parameter a controls the period of the sine function.

- Perceptron

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0 & if\ x \le 0 \end{cases} \tag{2.41}$$

This is a piecewise linear transfer function which is zero or positive.

2.4.1.2 Evolutionary ANNs (EANNs)

A difficulty with MLPs is that there is no complete theory governing the choice of an appropriate neural network architecture. The network architecture consists of the number of hidden layers, the number of PEs in each hidden layer, the transfer functions at each of the PEs and the connectivity of the network. The problem of finding the appropriate inputs is usually isolated as a separate task but can also be considered as part of the overall network architecture. Finding the optimal neural network architecture for a feedforward ANN can be formulated as a search problem, where each point represents a different architecture in architecture space. The aim is to evaluate the performance criteria (e.g. fastest training speed, greatest parsimony, minimum error etc.) of all architectures, so that this performance level forms a surface in this space (Yao and Liu, 1997b). The optimal architecture is then represented in this search space as the highest point on this surface. Manually determining the optimal architecture is a very tedious process and requires that the practitioner has a good knowledge of both ANNs and of the problem to be solved (Yao and Liu, 1997b). In most situations such knowledge is unavailable. Heuristic algorithms that automatically construct or prune nodes in the ANN are a step above manual selection, however, these methods have their shortcomings. These shortcomings primarily result from a susceptibility to becoming trapped in poor local optima due to their hill-climbing nature. Recently, evolutionary artificial neural networks (EANNs) have been proposed (Angeline et al., 1994; Boozarjomehry and Svrcek, 2001; De Falco, 1998; Fogel et al., 1990; Hakkarainen et al., 1996; Kitano, 1994; MacLeod and Maxwell, 2000; Maniezzo, 1994; McDonnell and Waagen, 1994; Miller et al., 1989; Whitehead and Choate, 1994; Whitehead and Choate, 1996; Yao, 1995b; Yao and Liu, 1997a; Yao and Liu, 1998a; Yao and Liu, 1998b). In EANNs, evolutionary algorithms are utilised to search the architecture space for an optimal solution. EANNs avoid the tedious trial-and-error approach by evolving network architecture automatically. They are a special class of networks not only capable of adapting via learning but also by evolution. The EANN approach is a very powerful alternative to the rule-of-thumb or constructive/destructive algorithm approaches.

Miller et al. (1989) has suggested several reasons why evolutionary algorithms are well suited to searching this space. These are:

- The surface is *infinitely large* since the number of possible nodes and connections is unbounded.

- The surface is *non-differentiable* since changes in the number of nodes or connections are discrete and can have a discontinuous effect on an ANN's performance.
- The surface is *complex* and *noisy* since the mapping from an architecture to its performance is indirect and depends entirely on the evaluation criterion used.
- The surface can be *deceptive*, as it is possible for similar architectures to have significantly different performance.
- The surface is *multimodal* as it possible for different architectures to have similar performance.

EANNs result from combinations of artificial neural networks and evolutionary algorithms such as those based on Fogel's evolutionary programming (EP) (Fogel et al., 1966). EP is one of the three broadly related avenues of simulated evolution, the other two being: genetic algorithms and evolution strategies (Fogel and Fogel, 1994). Each of these techniques emphasises a different feature of natural evolution (Fogel, 1994): Genetic algorithms stress chromosomal operators, evolution strategies highlight behavioural changes at the level of the individual and EP stresses behavioural change at the level of the species. These techniques have some similarities. For instance, they all involve population-based search strategies where individuals in the population must compete and exchange information with each other to perform certain tasks. The differences arise from their developmental background; GAs started from the simulation of genetic evolution, whereas EP started from simulating environmental evolution. As a consequence, mutation is the only EP operator. Evolutionary computation is the term that encompasses all of these techniques (Takagi, 1997).

Finding the optimal network architecture (i.e. the number of hidden units and their connection structure) and the training of the network weights are both optimisation problems; discrete for the network architecture and continuous for the network weights. In order to optimise the network architecture, it is essential to also optimise the weights at the same time, since evaluating the network's performance requires that the network has been trained. This can be achieved by using the backpropagation algorithm to optimise the network weights and an evolutionary algorithm to optimise the architecture. This is a good combination, because the evolutionary algorithm conducts a full explorative search which has a high likelihood of finding the global optimum and the backpropagation algorithm exploits additional information (as hill-climbing strategies do). By combining the two techniques, it is possible to tremendously diminish the search space by restricting the search to a set of local optima. In this approach, two time scales are used. A coarse step for the evolutionary algorithm is intertwined with a number of fine steps for the optimisation of the offspring using the

backpropagation algorithm. Before the evaluation of the fitness for crossover (mating) is conducted, the offspring undergo a period of fine-tuning called learning.

When optimising the architecture of an ANN model, the decision variables are discrete and therefore, a binary encoding mechanism is sufficient. Different EANNs can be classified as direct and indirect methods according to their encoding mechanism (Boozarjomehry and Svrcek, 2001). Direct methods represent all information about the structure of the ANN as a binary string and this string is used as the genotype. For example, Miller et al. (1989) constructed the genotype by concatenating the rows (or columns) of the connectivity matrix into a binary string (Figure 2.10). In this scheme, a binary string of length N^2 represents an ANN with N PEs. A disadvantage of direct methods is their poor scalability, since the length of the genotype increases exponentially as the number of PEs in the ANN increases (Kitano, 1994). Indirect methods compress the data corresponding to the structure of the ANN to reduce the length of the binary string. Therefore, conversion of the string to the ANN structure requires at least one level of interpretation. Indirect methods can be further categorised based on the compression and interpretation method they use (Boozarjomehry and Svrcek, 2001).

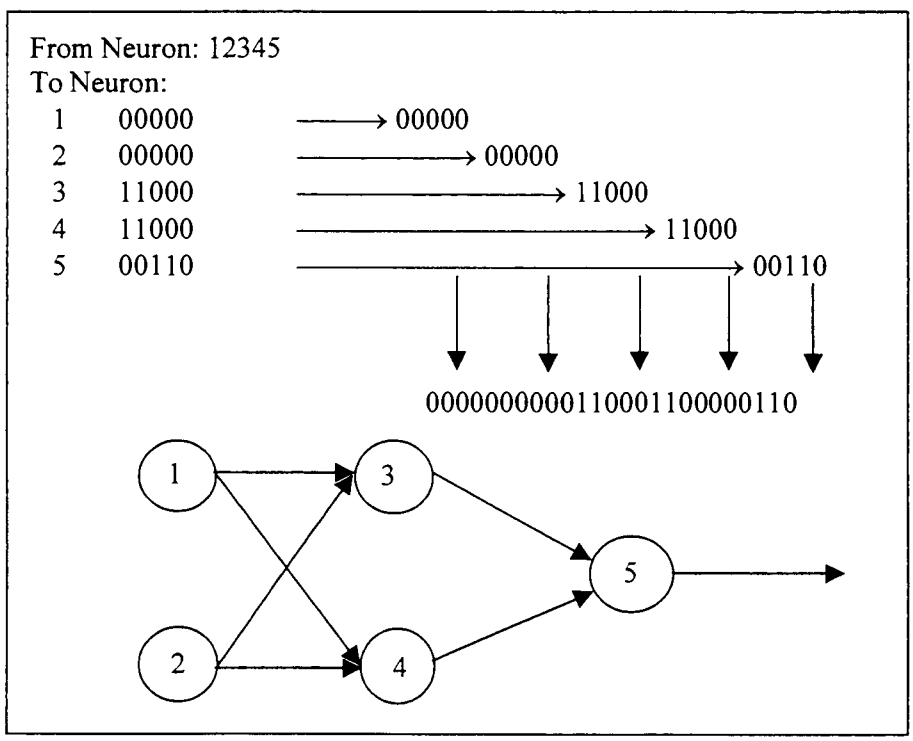

Figure 2.10 Coding of the Connectivity Matrix and Decoding to the ANN (source: Miller et al., 1989)

The evolution of ANN architecture can be described by the following cycle (Yao, 1993):

1. Decode each individual in the current generation into an architecture. In the case of the indirect encoding scheme, decode using the interpretation method.
2. Train each EANN with the decoded architecture by a predefined and fixed learning rule (but some parameters of the learning rule may be adaptive and learned during training), starting from different sets of random initial values of connection weights and, if any, learning rule parameters.
3. Calculate the fitness of each individual (encoded architecture) based on the above training results; e.g., based on the smallest mean square error of training (or testing if more emphasis is put on generalisation ability), the shortest training time, the complexity of the architecture (fewest PEs and connections) etc.
4. Reproduce a number of children for each individual in the current generation with probability of selection calculated according to its fitness or rank.
5. Apply genetic operators, such as crossover, mutation and/or inversion, with a given probability to child individuals produced above, and obtain the new generation

2.4.1.3 General Regression Neural Network (GRNN)

Another type of supervised, feedforward ANN is the GRNN, however, the GRNN is not as widely used as MLPs trained with the backpropagation algorithm. The GRNN is useful for continuous function approximation and is also a universal approximator of smooth functions (Sarle, 1997). Developed by Specht (1991), the GRNN is a type of ANN that does not require any particular form for the regression function to be assumed but rather, utilises a nonparametric estimate of the probability density function of the observed data. In so doing, GRNNs are very similar in their underlying philosophy to statistical models such as Nadaraya-Watson kernel regression (Sarle, 1994). Unlike MLPs trained using the backpropagation algorithm, which take a large number of iterations to converge to the desired solution, the GRNN training patterns are only propagated through the network once. In addition, the network architecture is fixed, however, appropriate model inputs must still be determined.

The GRNN consists of four layers and a diagrammatic representation of this network is shown in Figure 2.11. The input layer is fully connected to the second layer, which is called a pattern layer. The number of PEs in the pattern layer is usually equal to the number of patterns (examples) in the training set because each pattern in the training set is assigned to one pattern layer PE. The incoming weights to the pattern layer correspond to the pattern elements that each PE represents from the training set. For

example, consider a GRNN with three inputs. The first pattern layer PE is assigned to the first training pattern. The weight from the first input to the first pattern layer PE is the value of the first input in this first training pattern. Likewise for the second and third inputs, the weights connecting to this PE will be the values of the second and third inputs, respectively.

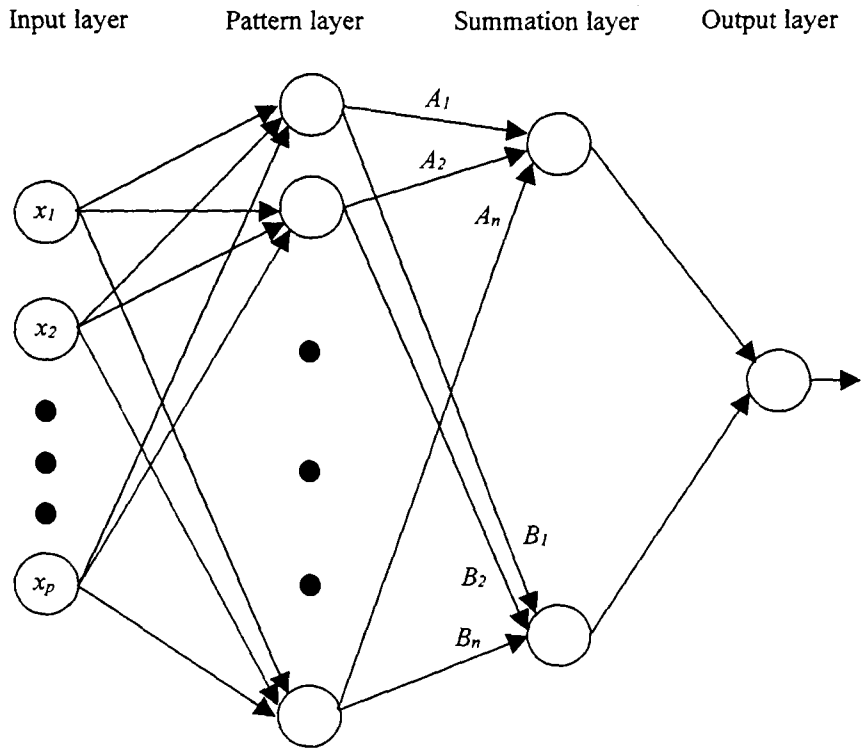

Figure 2.11 The GRNN Architecture

GRNNs work by measuring how far a given pattern is from all other patterns in the training set. Therefore, at each of the pattern layer PEs, the distance between each input pattern and the pattern represented by the PE is computed. For each PE in the pattern layer this is achieved by subtracting each input pattern element from its corresponding weight, and then taking the square or absolute value of these differences depending on whether the Euclidean or city block distance metric is used. The Euclidean distance metric has been recommended by Abu Kiefa (1998) but Specht (1991) found that the city block metric works approximately as well. The advantage of the city block metric is its computational simplicity.

Once the distance has been computed, it is fed into a nonlinear transfer function. Normally the exponential function is used, as given by:

$$f(D_i) = \exp(\frac{-D_j}{2\sigma^2}) \tag{2.42}$$

where D_j is the distance computed for the *jth* pattern layer PE, and σ is a smoothing parameter. Alternative choices for the transfer function can also be used successfully. The smoothing parameter is the only unknown in the GRNN and it determines how tightly the underlying regression surface will be approximated. Increasing the smoothing parameter enables a much more relaxed estimate of the surface fit through the data and increased generalisation ability within the range of the training data. Conversely, as the smoothing parameter decreases, a more accurate fit to the training data is obtained but interpolation ability may decrease, depending on the degree of noise present in the data set.

The output from each of the pattern layer PEs passes to the summation layer, with the pattern and summation layers being fully connected. The summation layer is divided into two types of PEs, namely *A* and *B* units. If one pattern layer PE is assigned to each training pattern, then the *B* unit weights are all set to unity. The *A* unit weights are the actual output values associated with the pattern stored at each pattern layer PE. For example, the weight from the first pattern layer PE to the first *A* summation layer PE is the actual output associated with the first training sample and so on. Both the *A* and *B* summation PEs perform a simple dot product between their weight vectors and the output signals from the pattern layer PEs. The two dot products then pass to the output layer where the output PEs simply divide the *A* summation unit's output by the *B* summation unit's output to produce the output for the network. This output can be interpreted as a type of weighted average of the observed values, where each observed value is weighted exponentially according to its distance from the training observations (Specht, 1991). The main components of the GRNN paradigm are outlined below.

The GRNN paradigm (Specht, 1991)

Assume a vector $\boldsymbol{x}$ of N independent, random variables is used to predict a dependent scalar random variable y. Let $\boldsymbol{X}$ be a particular measured value of the random variable $\boldsymbol{x}$. If the joint density $f(\boldsymbol{X}, y)$ is known, then it is possible to compute the conditional mean of y given $\boldsymbol{X}$ (or regression of y on $\boldsymbol{X}$) by using Equation 2.43.

$$E[y \mid \boldsymbol{X}] = \frac{\int_{\infty}^{\infty} y.f(\boldsymbol{X}, y)\, dy}{\int_{\infty}^{\infty} f(\boldsymbol{X}, y)\, dy} \tag{2.43}$$

If, however, the joint density $f(\boldsymbol{X}, y)$ is not known, then an estimate $\hat{f}(\boldsymbol{X}, y)$ based on a sample of observations of $\boldsymbol{x}$ and y must be used. The GRNN utilises a class of

consistent nonparametric estimators known as Parzen window estimators (Parzen, 1962). Using Parzen window estimation, $\hat{f}(\boldsymbol{X}, y)$ can be written as:

$$\hat{f}(\boldsymbol{X}, Y) = \frac{1}{2\pi^{(p+1)/2}\sigma^{(p+1)}} \cdot \frac{1}{n}\sum_{i=1}^{n} \exp\left[-\frac{(\boldsymbol{X}-\boldsymbol{X}^i)^T(\boldsymbol{X}-\boldsymbol{X}^i)}{2\sigma^2}\right] \cdot \exp\left[-\frac{(Y-Y^i)^2}{2\sigma^2}\right] \quad (2.44)$$

where n is the number of sample observations and p is the dimension of the vector variable $\boldsymbol{x}$ (i.e. the number of inputs). Substituting Equation 2.44 into Equation 2.43 gives the following estimate of the conditional mean:

$$\hat{Y}(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} \exp\left[-\frac{(\boldsymbol{X}-\boldsymbol{X}^i)^T(\boldsymbol{X}-\boldsymbol{X}^i)}{2\sigma^2}\right] \cdot \int_{-\infty}^{\infty} y.\exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right] dy}{\sum_{i=1}^{n} \exp\left[-\frac{(\boldsymbol{X}-\boldsymbol{X}^i)^T(\boldsymbol{X}-\boldsymbol{X}^i)}{2\sigma^2}\right] \cdot \int_{-\infty}^{\infty} \exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right] dy} \quad (2.45)$$

Letting $D_i^2 = (\boldsymbol{X}-\boldsymbol{X}^i)^T(\boldsymbol{X}-\boldsymbol{X}^i)$ and performing the required integrations gives the following estimate of $\hat{Y}(\boldsymbol{X})$:

$$\hat{Y}(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} Y^i \cdot \exp\left(-\frac{D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right)} \quad (2.46)$$

The regression in Equation 2.46 involves summations, is directly applicable to numerical data and can easily be implemented via four layers of parallel neural network architecture as shown in Figure 2.11. In this case, the A summation layer PE has its weights set to the actual output values i.e. A^i=Y^i and the B summation layer PE has its weights set to unity i.e. B^i = 1.

For the estimator to be consistent, the smoothing parameter $\sigma = \sigma(n)$ must be chosen as a decreasing function of n, where:

$$\lim_{n\to\infty} \quad \sigma(n) = 0 \quad (2.47)$$

and

$$\lim_{n\to\infty} \quad n\sigma^p(n) = \infty \quad (2.48)$$

as this will ensure that the estimator asymptotically converges to the underlying joint density function $f(x,y)$ at all points (x,y). If the smoothing parameter σ is made large, then the estimated joint density function is forced to be smooth and in the limit becomes a multivariate Gaussian with covariance $\sigma^2 \boldsymbol{I}$. In this case, the network's output approaches the average value of the training sample outputs. Conversely, a smaller value for the smoothing parameter σ allows the estimated density to assume non-Gaussian shapes with the danger that wild points may have a too great an influence on the estimate. In this case, the output approaches the actual output of the training sample that is closest to the input pattern. Only for intermediate values of the smoothing parameter can the network generalise well (Caudill, 1993).

The value for the smoothing parameter must be determined empirically. A common approach is the holdout method (leave-one-out cross validation) combined with a hill-climbing optimisation. For a given value of σ, the holdout method consists of removing one sample at a time and constructing a network based on all of the other samples. The network is then used to estimate the actual value Y for the removed sample. This process is repeated for each sample and the estimates stored so that the mean squared error (MSE) can be computed between the actual values and the estimates. To implement the hill-climbing optimisation, the value for the smoothing parameter is then changed and the whole holdout process repeated. If the MSE goes up, then the smoothing parameter is changed in the opposite direction. If the MSE goes down, then the smoothing parameter is changed some more in the current direction. The process is terminated when the change in the smoothing parameter only results in a very small change in the MSE. This method is often acceptable because the curve of MSE versus σ usually slopes gently down to the global minimum and exhibits a wide range of values near this minimum (Specht, 1991). This feature would seem to preclude the use of a stochastic optimisation algorithm. Despite this, Hansen and Meservy (1996) used genetic algorithms to determine an optimal value for σ and found that the genetic algorithm approach consistently outperformed the hill-climbing optimisation for their case study.

Improvements to the GRNN

In order to decrease the computational requirements of the GRNN, Specht (1991) suggested that the Gaussian kernel used in Equation 2.44 could be replaced by alternative Parzen windows, one of which is attractive because of its computational simplicity and it results in the following estimator:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y^i \cdot \exp\left(-\frac{C_i}{\sigma}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{C_i}{\sigma}\right)} \tag{2.49}$$

where

$$C_i = \sum_{j=1}^{p} \left| X_j - X_j^i \right| \tag{2.50}$$

This measure is commonly called city block distance.

As the number of training samples increases, the size of the GRNN increases accordingly and training times can become prohibitive for very large networks. To overcome this problem, Specht (1991) proposed the use of clustering techniques (e.g. self-organizing map, k-means clustering) to group training samples. Each cluster or group is represented by only one pattern layer PE, which takes the values of the cluster centre as its incoming weights. In addition, the weights on the summation layer PEs must also be changed. The weights on the B summation layer PE keep count of how many times a pattern has clustered at a particular pattern layer PE. The weights on the A summation layer PE keep count of the sum of the actual output for the training samples. Therefore, if a training observation Y^j is encountered for cluster i, the weights are incremented using:

$$\begin{aligned} A^i(k) &= A^i(k-1) + Y^j \\ B^i(k) &= B^i(k-1) + 1 \end{aligned} \tag{2.51}$$

where $A^i(k)$ and $B^i(k)$ are the values of the weights for cluster i after k observations.

Equation 2.49 is then rewritten as:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A^i \cdot \exp\left(-\frac{C_i}{\sigma}\right)}{\sum_{i=1}^{m} B^i \cdot \exp\left(-\frac{C_i}{\sigma}\right)} \tag{2.52}$$

where $m<n$ is the number of clusters.

The original GRNN configuration can also be improved by introducing separate smoothing parameters for each input variable (Masters and Land, 1997). In this case, Equation 2.52 is adjusted to incorporate separate sigma weights for each input.

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A^i \cdot \exp\left(-C_i\right)}{\sum_{i=1}^{m} B^i \cdot \exp\left(-C_i\right)} \tag{2.53}$$

where

$$C_i = \sum_{j=1}^{p} \frac{\left|X_j - X_j^i\right|}{\sigma_j} \tag{2.54}$$

However, using multiple sigma weights negates the instant learning capacity of the original single sigma version of the GRNN and increases the likelihood of multiple local optima. This makes the model difficult to train in many circumstances, often with significant time requirements. In addition, the error function is often nearly flat, meandering downward slowly, with many twists and turns along the way (Masters and Land, 1997). Various authors have attempted to solve this difficult optimisation problem. Masters (1995a) proposed the use of an enhanced conjugate gradient optimisation algorithm based on cross validation, whilst Chtioui et al. (1999) used a standard conjugate gradient algorithm to optimise the smoothing parameters. However, in Masters and Land (1997) the enhanced conjugate gradient algorithm was stated to have very little global effectiveness and a new training algorithm based on differential evolution (Price and Storn, 1997) was introduced. Masters and Land (1997) found that gradient based optimisation techniques were not suitable when there are more than a few inputs to the network, due to the increased likelihood of multiple local minima. Therefore, if the multiple sigma version of the GRNN has more than a few inputs, stochastic training algorithms are advantageous because of their ability to avoid becoming trapped in these local minima.

2.4.2 Unsupervised ANNs

2.4.2.1 Self-Organizing Map (SOM)

The self-organizing map (SOM) was developed by Kohonen (1982) and arose from attempts to model the topographically organized maps found in the cortices of the more developed animal brains. The underlying basis behind the development of the SOM

was that topolcgically correct maps could be formed in an *N*-dimensional array of processing elements that did not have this initial ordering to begin with. In this way, input stimuli, which may have many dimensions, can come to be represented by a one- or two-dimensional vector which preserves the order of the higher dimensional data (NeuralWare, 1998b).

The SOM employs a type of learning commonly referred to as competitive, unsupervised or self-organizing, in which adjacent cells within the network are able to interact and develop adaptively into detectors of a specific input pattern (Kohonen, 1990). The SOM can be considered to be as "*neural*" because results have indicated that the adaptive processes utilized in the SOM may be similar to the processes at work within the brain (Kohonen, 1990).

The SOM has potential extending beyond its original purpose of modelling biological phenomena. Sorting items into categories of similar objects is a challenging, yet frequent task. The SOM achieves this task by nonlinearly projecting the data onto a lower dimensional display and by clustering these data. This attribute has been used in a wide number of applications ranging from engineering (including image and signal processing and recognition, telecommunications, process monitoring and control, and robotics) to natural sciences, medicine, humanities, economics and mathematics (Kaski et al., 1998).

The self-organizing map algorithm

In competitive learning, neurons in the network adapt gradually to become sensitive to different input categories. The SOM network generally consists of two layers, an input layer and a Kohonen layer (Figure 2.12). The input layer is fully connected to the Kohonen layer, which in most common applications is two-dimensional. None of the PEs in the Kohonen layer are connected to each other. The PEs in the Kohonen layer measure the distance of their weights to the input pattern. During the recall phase, the Kohonen PE with the minimum distance is the winner and has an output of 1.0, whilst the other Kohonen PEs have an output of 0.0.

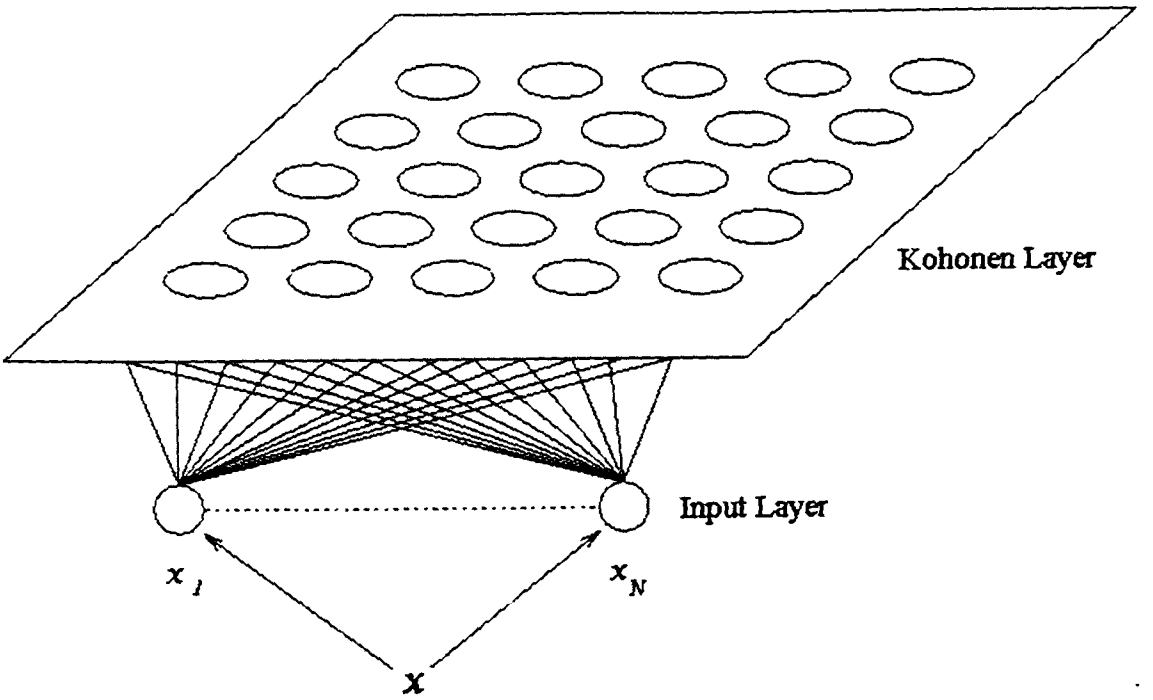

Figure 2.12 A Typical SOM Consisting of N Inputs in the Input Layer and a 5 by 5 Kohonen Layer

The procedure for determining the winning PE is as follows:

The first step is to determine the extent to which the weights of each PE match the corresponding input pattern. If the input data have N inputs and are denoted by:

$$X = (x_i;\ i = 1,...,N) \in \Re^n \tag{2.55}$$

then each of the M PEs in the Kohonen layer will also have N weight values and can be denoted by:

$$W_{ji} = (w_{ji};\ j = 1,...,M;\ i = 1,...,N) \in \Re^n \tag{2.56}$$

For each of the M Kohonen PEs, the distance, such as the Euclidean distance, is calculated using:

$$D_j = \left\| X - W_j \right\| = \left[\sum_{i=1}^{N} \left(x_i - w_{ji} \right)^2 \right]^{\frac{1}{2}}, \quad j = 1,...,M. \tag{2.57}$$

The PE with the lowest value of D_j is the winner during recall. During training, a conscience mechanism adjusts the distances to encourage PEs that are winning with a below average frequency and to negatively adjust PEs that are winning at an above

average frequency. This mechanism ensures that a uniform data distribution develops in the Kohonen layer. In adjusting the distance, a bias, B_j, is added to the distance and forms the new adjusted distance, $D_j^{'}$. The bias is calculated using:

$$B_j = \gamma \left[M \cdot (F_j - 1) \right] \tag{2.58}$$

where γ is a learning coefficient; F_j is the frequency at which PE j has historically won; and M is the number of PEs in the Kohonen layer. Once B_j and D_j are computed, the adjusted distance, $D_j^{'}$ can be calculated using:

$$D'_j = D_j + B_j \tag{2.59}$$

To ensure biological plausibility, lateral interaction with neighbouring PEs is enforced by applying arbitrary network structures called neighbourhood sets, N_c. Throughout the process, all PEs within the winner's neighbourhood set have their weights updated, whilst PEs outside of this set are left intact. The width or radius of N_c can be time variable. The updating process to implement this procedure is given by:

$$W_j(t+1) = \begin{cases} W_j(t) + \alpha(t)(X(t) - W_j(t)) & if\ j \in N_c(t) \\ W_j(t) & if\ j \notin N_c(t) \end{cases} \tag{2.60}$$

where α is a scalar valued adaptation gain $0 < \alpha(t) < 1$ and N_c is the neighbourhood set. After the weights have been updated, the next input is presented to the network and the process continues until convergence has been reached. After successively presenting different inputs to the SOM, the net effect is that the weights reflect the topological relationship that exists within the input data (Islam and Kothari, 2000).

2.5 Important Steps in the Development of ANN Models for Time Series Forecasting

There are a number of key steps involved in developing ANN time series models and these steps have been discussed by various authors (see Dawson and Wilby, 2001; Kaastra and Boyd, 1996; Maier and Dandy, 2000a; Maier and Dandy, 2000b; Masters, 1993). Within each of these steps, there are a number of parameters that must be carefully selected, such as the number of layers, the number of PEs in each layer and various internal parameters associated with the training (optimisation) algorithm used. In fact, ANN models are often criticised because of the large number of parameters that must be selected to generate a good forecast (Kaastra and Boyd, 1996), but it is the large number of ways an ANN model can be organised that accounts for its powerful ability to approximate functions. However, such flexibility in modelling time series data is not without cost. Researchers are faced with the difficult task of selecting the right combination of parameters, often with only broad rules of thumb to guide along the way. Because the application of ANN models to the modelling of time series data is a relatively new field, choosing the appropriate network paradigm often involves trial-and-error processes. Experienced users have their own way of choosing these parameters, but as Breiman (1994) points out, "most of this is folk wisdom, and there is, so far, no handbook on the sacred mysteries of neural network tinkering".

At each stage of the model building process there are various alternatives available. To gain an objective view of the current state of the art, it is important to understand what these alternatives are and what procedures have typically been employed. A typical ANN model development process involves the following steps (Maier and Dandy, 2000b): division of data and data preprocessing, the determination of adequate model inputs and a suitable network architecture, parameter estimation (optimisation) and model validation. The main steps in the model development process and some of the aspects that must be considered at each step are shown in Figure 2.13. The final step in the process is the deployment of the trained ANN model.

Often in many ANN applications, authors fail to describe the model building process adequately or carry out the model building process incorrectly (Maier and Dandy, 2000b). The important steps involved in the development of an ANN model are outlined below.

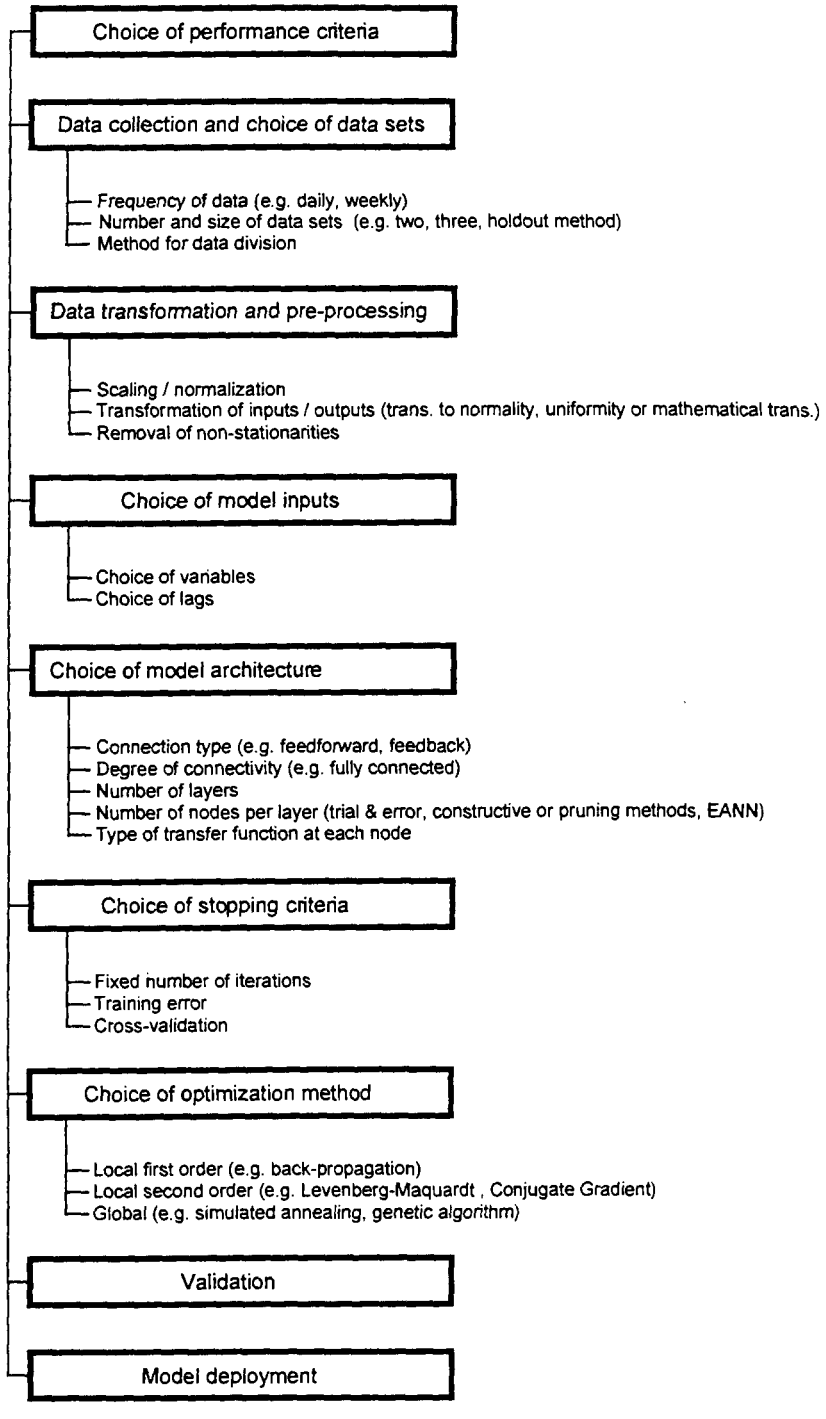

Figure 2.13 The Main Steps in the Development of ANN Models (source: adapted from Maier and Dandy, 2000a)

2.5.1 Choice of Performance Criteria

The performance criteria determine how an ANN model is to be assessed. When ANN models are used for time series forecasting, the most common performance criterion is prediction accuracy. To measure the prediction accuracy, an objective function (error measure) is used, such as the mean square error (MSE). Diskin and Simon (1977) point out that in any model building process "...the selection of an objective function for the

optimization procedure is itself a subjective decision which influences the optimal values of the model parameters. Thus the optimal set of parameters is optimal only in the context of the objective function selected". The same is true in ANN modelling. The choice of an appropriate objective function will have a significant impact on subsequent steps, such as the selection of network architecture and weight optimisation techniques. Various authors have proposed a number of measures of prediction accuracy (see Dawson and Wilby, 2001; Hecht-Nielsen, 1989; Lachtermacher and Fuller, 1994; Maier and Dandy, 1996b; Masters, 1993). The most commonly used measures of prediction accuracy include: the mean squared error (MSE), the root mean squared error (RMSE), the mean squared relative error (MSRE), the average absolute percentage error (AAPE), the coefficient of efficiency (CE), and the coefficient of determination (r^2) (see Equations 2.61 - 2.66).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2.61}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2} \tag{2.62}$$

$$MSRE = \frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - \hat{x}_i)^2}{x_i^2} \tag{2.63}$$

$$AAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|x_i - \hat{x}_i|}{x_i} \tag{2.64}$$

$$CE = 1 - \frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.65}$$

$$r^2 = \left[\frac{\sum_{i=1}^{n}(x_i - \bar{x})(\hat{x}_i - \tilde{x})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(\hat{x}_i - \tilde{x})^2}}\right]^2 \tag{2.66}$$

where $\hat{x}_i$ are the n predicted values, x_i are the actual or desired values, $\bar{x}$ is the mean of the actual values, and $\tilde{x}$ is the mean of the predicted values.

To assess the model's true generalisation ability, the prediction accuracy is usually calculated on data that have not been used in the model building process (Maier and Dandy, 2000b). Cheng and Titterington (1994) define generalisation ability as the model's ability to perform well on data that were not used to calibrate it. Generalisation ability is a function of the ratio of the number of training samples to the number of connection weights. Therefore, if the ratio is too small, overfitting of the training data and poor generalisation ability may result. To help avoid overfitting of the data, various measures of prediction accuracy have been devised that take into account the parsimony of the model. Commonly used measures that penalise oversized networks include the Akaike information criterion (AIC) (Akaike, 1974) (Equation 2.67) and Schwarz's Bayesian information criterion (BIC) (Schwarz, 1978) (Equation 2.68).

$$AIC = n\ln\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2\right] + 2p \tag{2.67}$$

$$BIC = n\ln\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2\right] + p\ln(n) \tag{2.68}$$

where p is the number of parameters fitted in the model. More advanced measures that also take parsimony into account have been proposed in the literature and include the Vapnik-Chervonenkis (VC) dimension (Abu-Mostafa, 1989) and the network information criterion (NIC) (Murata et al., 1994).

It is also important to note that a model's generalisation ability is significantly affected by the degree to which the training and validation sets represent the population to be modelled (Bowden et al., 2002; Maier and Dandy, 2000b). If the training and validation sets are not representative of the same population, then poor model performance may result. Generalisation ability is also affected by the stopping criteria, which is used to determine when to stop training the ANN model (see Section 2.5.5.1).

2.5.2 Choice of Data Sets, Data Transformation and Preprocessing

An important step in any modelling application is to collect the appropriate data. When collecting the data, both the cost and availability of the relevant data are important considerations. In addition, certain precautions should also be taken. For instance, the data should be checked for errors by examining the logical consistency, ranges and missing observations. If the data do contain any missing values, these can be dealt with by utilising data infilling techniques (see Elshorbagy et al., 2000b; Khalil et al., 2001).

Before any preprocessing of the data can be performed, the data set must be split into the appropriate subsets i.e. training, testing and validation. The training set consists of the set of examples used for learning, i.e. the examples used to fit the connection weights within the network. The test set consists of a set of examples to tune the parameters (including the architecture) but not the weights of an ANN. If cross-validation is used, the test set is used to decide when to stop training the ANN. The validation set is an independent set of examples used only to assess the performance (generalisation ability) of a fully-specified ANN. Careful selection of these subsets must be made because ANNs are unable to extrapolate beyond the range of their training data (Flood and Kartam, 1994). Hence, if the data used for validation contain values outside of the range of those used in testing, poor generalisation ability may result. In addition to this, it is important that the training and validation data sets are representative of the same population (Maier and Dandy, 2000b). In cases where data are limited, it becomes increasingly difficult to assemble a representative validation data set.

A number of methods have been proposed to overcome the problem of data limitation. Masters (1993) proposed a holdout method whereby a small data set is withheld for validation and the network is trained on the remaining data. The generalisation ability of the network is obtained with the aid of the validation set and a different subset of the data is then selected and withheld, with the process repeated. This process continues until the generalisation ability for the entire data set has been determined. Another alternative is to use a subset of the data as a test set in an exploratory trial phase to determine how long training should be carried out in order to achieve acceptable generalisation ability (Maier, 1995; Maier et al., 1998). The subset is then added to the remaining training data and the network is then trained on the combined data set for a fixed number of epochs as determined from the trial phase. Another popular method is known as cross-validation and it is a technique that was developed by Stone (1974). This method is fairly data intensive as it involves the use of three data sets; a training set, testing set and a validation set. In this method a test set is used to assess the performance of the model at various stages of learning. Since the data set used to determine the true generalisation ability of the model cannot be used in the training process, cross-validation requires an independent validation data set. The same also applies to cases where the internal parameters or network architecture is optimised by a trial-and-error process. Cross-validation can be used to determine when to stop training the model.

Once the data have been divided into appropriate subsets, it is important to then preprocess and transform the data. Preprocessing and transforming the data can improve the performance of the network considerably (Atiya and Shaheen, 1999) and is used to minimise noise, detect trends, highlight important relationships and flatten the distribution of the variable to enable the ANN to better learn the patterns in the data. ANNs are pattern matchers and by preprocessing the data, it is possible to make it easier for the network to extract useful information from the inputs and associate it with the desired outputs.

Most variables span different ranges, therefore it is usual practice to scale the real world data into ranges that are acceptable to the network (NeuralWare, 1998c). For example, if a network uses a hyperbolic tangent transfer function in the output layer, the actual outputs will lie between -1.0 and +1.0. For this network to learn effectively, it is important that the desired outputs are scaled from their real world values to a range of -0.8 to +0.8. It is usual practice not to scale the values to the extreme limits of the transfer function as the size of the weight updates may become small, thereby causing flat spots to occur whilst training (Maier and Dandy, 2000b). However, if there is a rigid upper or lower bound in the data, then it is acceptable to scale to the extreme limit of the range.

In certain cases, the performance of ANN models can be improved by removing any trends, heteroscedasticity and seasonal or cyclical fluctuations, however, the issue of stationarity is rarely considered (Masters, 1993). Stationarity may be of concern if the trained models are likely to deal with data that fall outside of the range used for training (i.e. if the data contain trends). In comparing the effect of input data with and without seasonal variation, Maier (1996a) found that ANNs had the ability to cater to irregular seasonal variation via the use of their hidden nodes.

Two common data transformations that can be used are first differencing and taking the natural log of a variable. In first differencing, any linear trends are removed from the data by using the changes in a given variable. However, this technique can lead to large variances in the data set. By taking a logarithmic transformation, data that are characterised by both small and large values may be compressed together.

It is also possible to use ratios of input variables as a data transformation. This can highlight important relationships in the data and has the added advantage of creating a more parsimonious model since the number of input neurons can be effectively reduced. If a logarithmic transformation is then used, the ratio relationship will be converted to a subtractive one which can simplify and improve the training of the

network (Masters, 1993). Another potential transformation is a simple or exponential moving average. Moving averages can be used to smooth both input and output data and consequently are particularly useful when variables contain small fluctuations that only serve to add noise to the network.

As another preprocessing step, it is possible to filter data by removing observations from the training and testing sets to create a more uniform distribution (Kaastra and Boyd, 1996). However, the filtering employed must be consistent with the objectives of the researcher.

2.5.3 Determination of Model Inputs

A commonly employed method of input determination is to use *a priori* knowledge of the system under investigation to select a set of candidate model inputs. However, where the potential number of input variables to an ANN is large and no *a priori* knowledge is available to suggest which variables to include, the selection process is inherently difficult. In many ANN applications, practitioners give very little attention to the task of input selection (Maier and Dandy, 2000b).

The input selection problem can be formulated as having a set of input variables to an ANN, and an output value which can be used to evaluate the fitness or merit of the network using those variables. The number of input values can be referred to as the number of dimensions. Hence a 100 dimensional problem has 100 inputs. The aim is to find the subset of inputs yielding the smallest predictive error (i.e. the network of highest fitness). In complex applications, the number of dimensions can be quite large and this problem is further exacerbated in time series studies, where appropriate lags must be chosen for each variable. Therefore, analytical techniques for determining the optimal subset of inputs present the modeller with a distinct advantage.

Maier and Dandy (1997a) investigated the use of two different analytical procedures (a statistical approach and a neural network approach) for determining the inputs to a neural network model of a multivariate time series. These two methods were then compared to an empirical approach in conjunction with sensitivity analysis. Lachtermacher and Fuller (1994) proposed fitting a univariate Box-Jenkins model to a time series, which is used as a guide for determining the inputs for simple univariate ANN models. However, this method had not been extended to multivariate cases. For multivariate ANN models, Maier and Dandy (1997a) proposed using the method of Haugh and Box (1977), which uses cross-correlation analysis to determine the strength of the relationship between the input time series and the output time series at various

lags. The other analytical technique considered by Maier and Dandy (1997a) was the neural network-based approach, which involved developing N-bivariate models, where N is the number of input variables. Each model relates lagged values for a single input to the output variable. Sensitivity analyses are used to determine which lags are significant. In the sensitivity analyses, each of the inputs is increased by 5% in turn, and the sensitivity is calculated using Equation 2.69.

$$\text{Sensitivity} = \frac{\%\text{ change in output}}{\%\text{ change in input}} \times 100 \tag{2.69}$$

Plots of the sensitivities for each of the lags are then inspected and the significant lags are chosen using judgement. Maier and Dandy (1997a) found that for a particular case study, the generalisation ability of the three ANN models developed was comparable, regardless of which technique was used to determine the inputs.

A stepwise procedure can also be used to determine model inputs (see Maier et al., 1998). This method involves developing N-bivariate models, where N is the number of input variables. Each bivariate model relates one input to the output. The input variable from the bivariate model that gives the smallest root mean square error (RMSE) is then selected as a significant input and included in the model. Subsequently, N-1 models are developed by combining this variable with each of the remaining variables. This procedure can then be repeated with three input variables, four input variables etc., until the addition of any extra variables does not improve model performance. The disadvantages of this procedure are that it is computationally intensive and the synergistic effect of certain combinations of variables may be overlooked.

2.5.4 Choice of Model Architecture

There are many different types of neural network structures but the most popular for time series models are multilayer, feedforward neural networks (Kwok and Yeung, 1997a). Methods that employ optimisation algorithms to find a set of weights often assume the network topology to be fixed. This approach is only useful if the network architecture has been chosen correctly. If the network is too small, there may be difficulties in learning the problem well. If the network is too large, overfitting, decreased processing speed and poor generalisation ability may result. Determining the network architecture is a very important step in model development, however, it is also one of the most difficult tasks (Maier and Dandy, 2000b).

The numbers of PEs in the input and output layers are fixed by the number of inputs and outputs respectively. It is common practice to fix the number of hidden layers in the network and then to choose the number of PEs in each of these layers. It has been shown that only one hidden layer is required to approximate any continuous function (Cybenko, 1989). For this reason most studies only utilise one hidden layer. The hidden nodes (non-linear processing elements) encode the higher order constraints of the input data by varying the weights governing the strengths of connections between nodes in order to minimise predictive error (Broomhead and Lowe, 1988). The usual method of selecting the number of hidden PEs is a rule-of-thumb approach. For instance, Hecht-Nielsen (1989) proposed the following upper limit for the number of hidden layer nodes:

$$N^H \leq 2N^I + 1 \tag{2.70}$$

where N^H is the number of hidden layer nodes and N^I is the number of inputs. Fletcher and Goss (1993) proposed a number of hidden layer nodes as follows:

$$\left[\ 2(N^I)^{0.5} + N^O\ \right] \leq N^H \leq (2N^I + 1) \tag{2.71}$$

where N^O is the number of outputs. However, Rogers and Dowla (1994) proposed the following relationship to avoid overfitting the training data:

$$N^H \leq \frac{N^{TR}}{N^I + 1} \tag{2.72}$$

where N^{TR} is the number of training samples.

Often the guidelines are not suitable for particular applications because the choice of network architecture is very problem specific and in addition to this, different guidelines suggest different architectures. In most case studies, the ANN architecture is designed through a very tedious trial-and-error approach and is then fixed during the training process. Relying on a manual search to explore the large number of possible ANN architectures is impractical. Hence, there is a need to investigate automated approaches for determining the optimal network architecture.

Two broad methodologies exist for automating the search for optimal network architecture and these can be divided into heuristic approaches and evolutionary approaches. The evolutionary approaches have been discussed in Section 2.4.1.2. The heuristic approaches are discussed below.

2.5.4.1 Heuristic Approaches

Numerous constructive and pruning algorithms have been described in the literature (see Chen et al., 1997; Doering et al., 1997; Fahlman and Lebiere, 1990; Kwok and Yeung, 1997a; Kwok and Yeung, 1997b; Reed, 1993; Setiono, 1997) with the aim of automatically generating the optimal ANN architecture. While these techniques have certain advantages, "such *structural hill climbing* methods are susceptible to becoming trapped at structural local optima" (Angeline et al., 1994). However, these algorithms are an improvement over a manual search of the solution space.

Pruning algorithms generally start with a network that is larger than needed. This is then trained until a suitable solution is found. After this, the hidden units or weights that are no longer actively used are removed or disabled and training recommences. A review of pruning algorithms is given in Reed (1993).

The main shortcoming with pruning algorithms is that they can have difficulty finding the most compact network because, in many applications, there are many medium sized networks that perform adequately (Bebis and Georgiopoulos, 1994).

Constructive algorithms approach the network architecture optimisation problem from the opposite direction to pruning algorithms. The search commences with a small network and additional hidden units and weights are added until an appropriate solution is determined. Kwok and Yeung (1997a) have identified a number of reasons why this approach has advantages over the pruning method. These are:

- It is straightforward to specify an initial network, whereas for pruning algorithms, one does not know in practice how big the initial network should be.
- Constructive algorithms always search for smaller networks first. Hence, they are more computationally efficient than pruning algorithms, in which the majority of the training time is spent on networks larger than necessary.
- Since many networks with different sizes may be capable of implementing acceptable solutions, constructive algorithms are likely to find smaller network solutions than pruning algorithms. Smaller networks are more efficient in forward computation and can be described with a simpler set of rules. Functions of individual hidden units may also be more easily visualised and the amount of training data required for good generalisation can be reduced.
- Pruning algorithms usually measure the change in error when a hidden unit or weight is removed. However, such changes can only be approximated for

computational efficiency, and hence, may introduce large errors, especially when many units are to be pruned.

The problems that have been identified with constructive algorithms are concerned with connecting new hidden units to the existing network, determining the (new and existing) weights in the network once connections to the new hidden units are made and knowing when to stop the addition of new hidden units (Kwok and Yeung, 1997b). A taxonomy of constructive algorithms is presented in Kwok and Yeung (1997a). Of these, perhaps the most widely used constructive algorithm is cascade correlation (Fahlman and Lebiere, 1990).

Cascade Correlation (CC) algorithms

Cascade correlation algorithms are all variants of the method originally proposed by Fahlman and Lebiere (1990). If a new hidden PE is to be created, a connection is established not only to the network's inputs and outputs but also to every existing hidden PE within the network. In this fashion, each new hidden PE adds a one unit hidden layer to the network, leading to a cascade architecture. First the weights feeding into the new hidden unit are trained, whilst the weights in the rest of the network are kept fixed. In many variations, there is a pool of hidden unit candidates that each use a different random seed. The candidate that optimises an objective function in input training is inserted into the network. The next step is to keep its weights constant and the weights connecting the new hidden unit to the output are then trained.

2.5.5 Training (Optimisation)

Training an ANN model involves optimising the connection weights and is equivalent to the parameter estimation phase in statistical models. The search for an appropriate set of weights can be considered as a complex, combinatorial optimisation problem. The optimisation problem is often formulated as the minimisation of an error function such as the mean squared error between the actual and the desired output (Yao, 1995a). However, there are practical difficulties in optimising the error because the error surface may be prone to many local minima that may be far removed from the global optimum solution (Fogel et al., 1990).

In general, there are local and global optimisation methods. Two types of local optimisation exist: first-order and second-order methods (Maier and Dandy, 2000b). First-order methods are linear and employ a gradient descent technique, whereas second-order methods are based on a quadratic model. Both methods adjust the

connection weights in the ANN iteratively in order to minimise the error. However, local optimisation methods are prone to becoming stuck in local minima. The momentum term and five to ten random starting sets of initial weights can help alleviate this problem (Kaastra and Boyd, 1996). Global optimisation methods, such as genetic algorithms (Goldberg, 1989) and simulated annealing (Laarhoven and Aarts, 1987), have the ability to escape from local optima and can thus find optimal or near-optimal weight configurations.

2.5.5.1 Choice of Stopping Criteria

The training process is usually stopped when certain criteria have been met. If training is stopped too early, the model has not had a chance to learn the relationships in the data and poor generalisation ability will result. Likewise, if training is stopped too late, the model may overfit the data and learn the inherent relationships of the training set itself, hence, poor generalisation ability will also result.

There are two differing schools of thought about when training should be stopped (Kaastra and Boyd, 1996). The first emphasises the difficulty of reaching the global minimum and the danger of becoming stuck in a local minimum. In this school of thought, the modeller should only stop training when there is no further improvement of the error function (Masters, 1993). The network is said to have converged at the point in which there is no further improvement. The second view advocates using a number of train-test interruptions to gauge the model's performance (NeuralWare, 1998b). In this approach, training is stopped at a predetermined number of iterations and the generalisation ability of the model is evaluated on a test set, then training is resumed. The network for which the testing set error is the lowest is chosen, since this model is assumed to generalise best.

Both schools of thought acknowledge that generalisation ability on the validation set is the ultimate goal and both methodologies evaluate a large number of networks using testing sets. However, the two approaches differ on the point of overtraining versus overfitting. In the convergence approach there is no such thing as overtraining, only overfitting (Kaastra and Boyd, 1996). Overfitting is linked to the ratio of the number of training samples to the number of connection weights (Maier and Dandy, 2000b). Hence, the solution to this is to reduce the number of hidden PEs (or hidden layers) and/or increase the number of training samples. Whereas the train-test approach aims to prevent overfitting by stopping training based entirely on the network's ability to generalise.

2.5.5.2 Internal Model Parameters

In most ANN models, there are a number of internal model parameters usually associated with the optimisation algorithm. These must also be selected carefully and are usually found using a trial-and-error process. The internal parameters of a backpropagation ANN include: initial weights, epoch size, learning rate and the momentum. To further complicate the selection, there are different variants of the standard gradient descent backpropagation algorithm which each have their own set of internal parameters.

2.5.6 Validation

After the training (optimisation) phase is complete, the next step is to test the performance of the trained network on an independent data set. As Maier and Dandy (2000b) noted, "it is vital that the validation data should not have been used as part of the training process in *any capacity*". This includes optimising the model inputs, network geometry and/or internal model parameters or to decide when to stop training. Poor validation can result from overfitting, inadequate network architecture and lack of, or inappropriate data preprocessing of training/testing/validation data (Maier and Dandy, 2000b).

2.5.7 Model Deployment

Although this is the last step, model deployment requires careful consideration even before the data are collected. Variables can only be used as inputs if there is an assurance that the data will be available on an ongoing basis. Most ANN software packages provide the capability to deploy a working model, either within the program itself, or as an executable file. In some instances it is preferable to implement a trained network within a spreadsheet environment. By knowing the architecture, transfer function and weights, it is a relatively easy task to reconstruct the network within the spreadsheet, however it is important that all scaling, data transformation and other parameters remain constant from testing to actual use.

Once the network has been deployed, its performance may degrade over time unless it is regularly retrained. Even with periodic retraining there is a small possibility that network performance may decrease if the input variables selected become less important in the future. To address these issues and to maintain the integrity of the model, it is important that components of the modelling process be repeated at periodic intervals. It has been recommended that the frequency of retraining for a deployed

ANN model should be the same as the length of the testing set used by the final model (Kaastra and Boyd, 1996). Periodic recalibration ensures that the network performance is continually maintained and the full benefit of an ANN model (i.e. its ability to adapt to changing conditions) can be fully exploited.

2.6 ANNs in Water Resources Applications

In recent years, ANNs have been used increasingly to forecast the behaviour of complex systems, such as systems involving natural and physical phenomena (Gershenfeld and Weigend, 1994). Part of this popularity has arisen from the perception that ANNs are able to overcome many of the difficulties involved in implementing traditional statistical models. This is largely because ANNs are not subject to the same restrictive rules that govern many statistical models. The emphasis in ANN modelling is on methods that work and prediction accuracy is the primary objective, whereas statisticians aim for universal methods and statistical optimality (Maier and Dandy, 2000a).

The use of ANN models in forecasting applications is a relatively new field but one which is growing at a rapid rate. Hiew and Green (1992) made the optimistic claim that "well-designed forecasting systems that incorporate neural network technology signal the beginning of a new era in the evolution of forecasting and decision support systems". Some authors (e.g Chatfield, 1993) were cautious of such remarks, questioning whether ANNs were in fact just a passing fad. However, the passing of time has not mellowed the remark made by Hiew and Green, and almost a decade later there have been thousands of successful applications of neural network forecasting models. Throughout this time, there has been a tendency to throw problems blindly at ANNs in the hope that they will be able to solve everything (Flood and Kartam, 1994). This fact was noted by Sarle (1994), who believed that ANN users often have a type of "...data in, predictions out" mentality.

One area in which ANN models have shown success is the prediction and forecasting of water resources variables. The success of ANN models in these applications, combined with their relative ease of implementation, has led to ANN models becoming a popular and widely used tool for practitioners in the field of water resources modelling. Forecasting water resources variables is a "noisy" application because the systems modelled are typically influenced by a large number of different inputs, which interact in a complex, often highly nonlinear fashion. In practice, observations of only a small subset of the most significant inputs are usually available for modelling purposes. From this small subset of inputs, the system behaviour needs to be estimated or generalised for future events. Fortunately, many inputs that affect the time series are serially correlated, and that causes a time series pattern that can give some hint of the future (Chakraborty et al., 1992).

In most water resources forecasting applications, a time series $x(1)$, ..., $x(t)$ exists where it is required to forecast the value of $x(t+1)$. The inputs to the network are usually chosen as the previous values $x(t-k+1)$, ..., $x(t)$ and the output is the forecast. The model is calibrated and validated on sufficiently long data sets, which are subsets of a larger historical time series. Values or external variables from other time series that have a correlated or causal relationship to the series to be forecasted are often also used as inputs to the model. For example, in forecasting river flow, such a time series could be the rainfall in the catchment. In the past, practitioners in the field of water resources modelling used simple linear regressions or time series models to obtain approximations of the relationships between variables. Such simple models were primarily used because of the restrictive nature governing the development of more complex statistical models. Complex statistical models are difficult to apply to real-world problems because many oversimplifying assumptions often need to be made. However, as pointed out by Maier and Dandy (2000b), ANNs have placed complicated statistical models well within the grasp of practitioners. Simply changing the network architecture and varying the transfer functions used by the ANN can alter model complexity and nonlinearities can be incorporated easily.

To review the current state of the art, 152 international journal papers on the application of ANNs in water resources modelling have been examined. Due to the large number of such papers, the review in this section is restricted to be general in nature and to papers published until the end of 2001. Because of the rapid increase in papers, and widespread nature of the journals in which they appear, it is unlikely that complete coverage has been achieved. Table 2.1 shows the journal papers reviewed by variable modelled and location of application. This is representative of published research in this area since 1992. The papers are grouped according to the variable modelled and appear in ascending order of publication year within each of these groups. Where multiple entries appear for the same paper, these represent different models that were developed (e.g. different type of ANN or different case studies).

A distribution of papers by year of publication is given in Figure 2.14 and by variable modelled in Figure 2.15. It can be seen that the growth in research in this limited subset of ANN application has been strong with the first papers appearing in 1992 and the number of papers published increasing rapidly in the last few years. Of these papers, it is evident that forecasting of flow (including rainfall-runoff, reservoir inflows and streamflow) was the aim of a significant majority (38%) of the papers. Forecasting water quality (including algal concentrations, cyanobacterial concentrations, *chlorophyll*-a, cryptosporidium, giardia, nitrate, nitrite, ammonia, phosphate, pesticide, total volatile organic compounds, sulphate, salinity, pH, macroinvertebrates, colour,

potassium chloride, residual chlorine and temperature) was the second largest application area (26%), followed by rainfall forecasting (20%). Of the three remaining groups, 6% of the papers dealt with forecasting water levels (including water table levels and river stage), 3% of the papers modelled soil water content and lastly, 7% of the papers dealt with miscellaneous applications (including specific yield, reservoir operation, classifying hydrologically homogenous regions, flow regimes, drought severity, generating pareto fronts, modelling wave equations, pan evaporation, water distribution network states and land use of catchment).

The majority of papers provided a good description of basic ANN theory, the case study under investigation and the results obtained but many provided poor discussion of the modelling process. This meant that the optimality of the results obtained could not be assessed in many cases. In addition, it would be extremely difficult to reproduce the results or make meaningful comparisons between the performance of different models. Very few of the papers assessed the performance of the developed ANNs by comparison with a more conventional modelling approach. While this may be unnecessary for some applications, a comparison would enhance applications in which a statistical or process-based model has traditionally been used and would help identify applications in which ANN models perform better, and as a consequence, help to define their boundaries of applicability.

Table 2.1 International Journal Papers Reviewed on the Application of ANNs to Water Resources Variables

Ref.	Author(s)	Year	Variable	Location(s)
1	Scardi	(1996)	Algal conc.	Chesapeake Bay and Delaware Bay (USA)
2	Recknagel et al.	(1997) [1]	Algal conc.	Lakes in Japan and Finland, River Darling (Australia)
3	Whitehead	(1997) [1]	Algal conc.	River Thames (England)
4	Yabunaka et al.	(1997) [1]	Algal conc.	Lake Kasumigaura (Japan)
5	Recknagel et al.	(1998) [1]	Algal conc., Zooplankton	Lake Kasumigaura (Japan)
6	Maier and Dandy	(1997b) [1]	Cyanobact. Conc.	River Murray (Australia)
7	Recknagel	(1997) [1]	Cyanobact. Conc.	Lake Kasumigaura (Japan)
8	Maier et al.	(1998) [1]	Cyanobact. Conc.	River Murray (Australia)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
9	Maier et al.	(2000) [1]	Cyanobact. Conc.	River Murray (Australia)
10	Karul et al.	(1998)	Chlorophyll-a	Keban Dam Reservoir (Turkey)
11	Crespo and Mora	(1993) [1,2]	Flow	Pisuena River (Spain)
12	Kang et al.	(1993) [2]	Flow	Chang (Korea)
13	Karunanithi et al.	(1994) [1,2]	Flow	Huron River (USA)
14	Hsu et al.	(1995) [1,2]	Flow	Leaf River (USA)
15	Lorrai and Sechi	(1995) [1,2]	Flow	Araxisi River (Italy)
16	Raman and Sunilkumar	(1995) [1,2]	Flow	Reservoirs in India
17a	Smith and Eli	(1995) [1,2]	Flow (single)	N/A
17b	Smith and Eli	(1995) [1,2]	Flow (multiple)	N/A
18	Clair and Ehrman	(1996) [1,2]	Flow, Carb., Nit.	Canadian Rivers
19	Minns and Hall	(1996) [1,2]	Flow	N/A
20	Poff et al.	(1996) [1,2]	Flow	Little Patuxent and Independence Rivers (USA)
21	Muttiah et al.	(1997) [1,2]	Flow	US River Basins
22	Shamseldin	(1997) [1,2]	Flow	Catchments in Nepal, China, Ireland, USA, Australia
23	Sureerattanan and Phien	(1997) [1,2]	Flow	Mae Klong River (Thailand)
24	Tawfik et al.	(1997) [1,2]	Flow	River Nile
25	Braddock et al.	(1998) [2]	Flow	A blind catchment in Australia
26a	Dawson and Wilby	(1998) [1,2]	Flow	River Mole (UK)
26b	Dawson and Wilby	(1998) [1,2]	Flow	River Amber (UK)
27a	Fernando and Jayawardena	(1998) [1,2]	Flow	Kaminonsha (Japan)
27b	Fernando and Jayawardena	(1998) [1,2]	Flow	Kaminonsha (Japan)
28	Furundzic	(1998) [2]	Flow	River Lim Basin (Yogoslavia)
29	Golob et al.	(1998) [1,2]	Flow	Soča River (Slovenia)
30a	Jayawardena and Fernando	(1998) [1,2]	Flow	Kaminonsha (Japan)
30b	Jayawardena and Fernando	(1998) [1,2]	Flow	Kaminonsha (Japan)
31a	Thirumalaiah and Deo	(1998a) [1,2]	Flow	Bhasta River (India)
31b	Thirumalaiah and Deo	(1998a) [1,2]	Flow	Bhasta River (India)
31c	Thirumalaiah and Deo	(1998a) [1,2]	Flow	Bhasta River (India)
32	Abrahart et al.	(1999)	Flow	Upper Wye River (UK)
33	Atiya et al.	(1999) [2]	Flow	River Nile (Egypt)
34	Campolo et al.	(1999) [2]	Flow	River Arno (Italy)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
35	Chang and Hwang	(1999) [2]	Flow	Shen-cei Creek (Taiwain)
36	Danh et al.	(1999) [2]	Flow	Da Nhim, La Nga (Vietnam)
37	Dawson and Wilby	(1999) [2]	Flow	River Mole (UK)
38	Dibike and Abbott	(1999)	Tidal Flow	Donegal Bay (Ireland)
39	Frakes and Yu	(1999)	Flow	Susquehanna River Basin (USA)
40	Jain et al.	(1999)	Flow	Orissa (India)
41	Lange	(1999) [2]	Flow	Zeller Bach, Windbach (Germany)
42a	Previdi et al.	(1999)	Flow	Fort Lauderdale (USA)
42b	Previdi et al.	(1999)	Flow	Munkerisparken (Denmark)
43a	Sajikumar and Thandaveswara	(1999) [2]	Flow	River Lee (UK)
43b	Sajikumar and Thandaveswara	(1999) [2]	Flow	Thuthapuzha River (India)
44	Savic et al.	(1999)	Flow	Kirkton Catchment (UK)
45	Tokar and Johnson	(1999) [2]	Flow	Little Patuxent River (USA)
46	Zealand et al.	(1999) [2]	Flow	Winnipeg River (Canada)
47	Anmala et al.	(2000)	Flow	Council Grove, El Dorado and Marion Watersheds (USA)
48	Coulibaly et al.	(2000a)	Flow	Chute-du-Diable Watershed (Canada)
49a	Dawson et al.	(2000)	Flow	River Mole (UK)
49b	Dawson et al.	(2000)	Flow	River Amber (UK)
50	Elshorbagy et al.	(2000a)	Flow	Red River Valley (Canada)
51	Elshorbagy et al.	(2000b)	Flow	English River (Canada)
52a	Elshorbagy et al.	(2000c)	Flow	English River (Canada)
52b	Elshorbagy et al.	(2000c)	Flow	Little River (USA)
52c	Elshorbagy et al.	(2000c)	Flow	Reed Creek (USA)
53	Gautam et al.	(2000)	Flow	Tono (Japan)
54a	Imrie et al.	(2000)	Flow	River Trent (UK)
54b	Imrie et al.	(2000)	Flow	River Dove (UK)
55a	Jain and Chalisgaonkar	(2000)	Flow	Kolar River (India)
55b	Jain and Chalisgaonkar	(2000)	Flow	Narmada River (India)
55c	Jain and Chalisgaonkar	(2000)	Flow	Experimental
56	Kocjančič and Zupan	(2000)	Flow	Brebovščica River (Slovenia)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
57	Lauzon et al.	(2000)	Flow	15 Watersheds from Saguenay-Lac-Saint-Jean Hydrographic System (Canada)
58	Thirumalaiah and Deo	(2000)	Flow, Water level	Bhatsa Dam (India)
59a	Tokar and Markus	(2000)	Flow	Fraser River (USA)
59b	Tokar and Markus	(2000)	Flow	Raccoon Creek (USA)
59c	Tokar and Markus	(2000)	Flow	Little Patuxent River (USA)
60a	Zhang and Govindaraju	(2000)	Flow	Council Grove Watershed (USA)
60b	Zhang and Govindaraju	(2000)	Flow	El Dorado Watershed (USA)
60c	Zhang and Govindaraju	(2000)	Flow	Marion Watershed (USA)
61	Abrahart and See	(2001)	Flow	Upper River Wye (UK)
62	Chang and Chen	(2001)	Flow	Da-cha River (Taiwan)
63	Coulibaly et al.	(2001b)	Flow	Chute-du-Diable Watershed (Canada)
64	Coulibaly	(2001c)	Flow	Chute-du-Diable Watershed (Canada)
65	Guo	(2001)	Flow	N/A
66a	Khalil et al.	(2001)	Flow	English River (Canada)
66b	Khalil et al.	(2001)	Flow	Osilinka River (Canada)
66c	Khalil et al.	(2001)	Flow	Graham River (Canada)
66d	Khalil et al.	(2001)	Flow	Halfway River (Canada)
66e	Khalil et al.	(2001)	Flow	Nagagami River (Canada)
67a	Kim and Barros	(2001)	Flow	Williamsburg (USA)
67b	Kim and Barros	(2001)	Flow	Raystown (USA)
67c	Kim and Barros	(2001)	Flow	Newport (USA)
67d	Kim and Barros	(2001)	Flow	Loyalsockville (USA)
68	Lekkas et al.	(2001)	Flow	Colwick (UK)
69	Mason et al.	(1996) [2]	Specific yield	Synthetic data
70	Neelakantan and Pundarikanthan	(2000)	Reservoir operation	Chennai (India)
71	Hall and Minns	(1999)	Hydrologic-ally homogenous regions	Catchments in South West England and Wales
72	Cai et al.	(1994)	Flow Regimes	Experimental Rig
73	French et al.	(1992) [1]	Rainfall	N/A
74	Allen and Le Marshall	(1994) [1]	Rainfall	Melbourne (Australia)
75	Goswami and Srividya	(1996) [1]	Rainfall	India

Continued

Ref.	Author(s)	Year	Variable	Location(s)
76	Chow and Cho	(1997) [1]	Rainfall	Hong Kong
77a	Hsu et al.	(1997) [1]	Rainfall	Japan
77b	Hsu et al.	(1997) [1]	Rainfall	Florida (USA)
78a	Loke et al.	(1997) [1,2]	Rainfall	Denmark
78b	Loke et al.	(1997) [1,2]	Runoff coeff.	Catchments in Europe and America
79	Miller	(1997) [1]	Rainfall	Western Pacific
80	Tsintikidis et al.	(1997) [1]	Rainfall	Western Pacific
81	Venkatesan et al.	(1997) [1]	Rainfall	India
82	Xiao and Chandrasekar	(1997) [1]	Rainfall	Central Florida (USA)
83	Zhang et al.	(1997)	Rainfall	Numerous locations in the USA
84	Kuligowski and Barros	(1998a) [1]	Rainfall	Mount Carmel (USA)
85	Kuligowski and Barros	(1998b) [1]	Rainfall	Mid-Atlantic Region (USA)
86	Kuligowski and Barros	(1998c) [1]	Rainfall	Youghiogheny River and Swatare Creek basins (USA)
87	Guhathakurta et al.	(1999)	Rainfall	India
88	Hsu et al.	(1999)	Rainfall	Japan
89	Bodri and Čermák	(2000)	Rainfall	Moravia (Czech Republic)
90	Grecu and Krajewski	(2000)	Rainfall	Tulsa (USA)
91	Luk et al.	(2000)	Rainfall	Upper Parramatta River Catchment (Australia)
92a	Rajeevan et al.	(2000)	Rainfall	North West India
92b	Rajeevan et al.	(2000)	Rainfall	Peninsula India
93	Sahai et al.	(2000)	Rainfall	India
94	Silverman and Dracup	(2000)	Rainfall	California (USA)
95	Toth et al.	(2000)	Rainfall	Apennines Mountains (Italy)
96a	Sorooshian et al.	(2000)	Rainfall	Florida (USA)
96b	Sorooshian et al.	(2000)	Rainfall	Texas (USA)
97a	Burian et al.	(2001)	Rainfall	Warrior Lock and Dam Site (USA)
97b	Burian et al.	(2001)	Rainfall	Tuscaloosa Airport (USA)
97c	Burian et al.	(2001)	Rainfall	Dauphin Island (USA)
98	Luk et al.	(2001)	Rainfall	Upper Paramatta River Catchment (Australia)
99	Michaelides et al.	(2001)	Rainfall	Cyprus
100	Olsson et al.	(2001)	Rainfall	Chikugo River Basin (Japan)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
101	Schoof and Pryor	(2001)	Rainfall, temperature	Indianapolis (USA)
102	Wu et al.	(2001)	Rainfall	Yangtze River (China)
103	Shin and Salas	(2000)	Drought Severity	Southwestern Colorado (USA)
104	Liong et al.	(2001)	Pareto Front	N/A
105	Shukla et al.	(1996) [1]	Water table level	N/A
106	Yang et al.	(1996) [1]	Water table level	N/A
107	Yang et al.	(1998)	Water table level	N/A
108	Yang et al.	(2000)	Water table level	Ontario (Canada)
109	Coulibaly et al.	(2001a)	Water table level	Burkina Faso
110a	Thirumalaiah and Deo	(1998b) [1,2]	Water level	River Godavari (India)
110b	Thirumalaiah and Deo	(1998b) [1,2]	Water level	River Godavari (India)
110c	Thirumalaiah and Deo	(1998b) [1,2]	Water level	River Godavari (India)
111	See and Openshaw	(1999) [2]	Water level	Ouse River (England)
112	Liong et al.	(2000a)	Water level	Dhaka (Bangladesh)
113	Liong et al.	(2000b)	Water level	Dhaka (Bangladesh)
114	See and Abrahart	(2001)	Water level	River Ouse (England)
115	Wen and Lee	(1998)	Water quality	Tou-Chen River Basin (Taiwan)
116	Schleiter et al.	(1999)	Water quality	River Lahn (Germany)
117	Brion et al.	(2001)	Crypto-sporidium	Delaware River (USA)
118	Neelakantan et al.	(2001)	Crypto-sporidium, Giardia	Delaware River (USA)
119	Lischeid et al.	(1998)	Nitrate	Lehstenbach and Steinkreuz Catchments (Germany)
120	Ray and Klindworth	(2000)	Pesticide, Nitrate	Rural Private Wells (USA)
121	Aguilera et al.	(2001)	Nitrate, Nitrite, Ammonia, Phosphate	Almeria Coastline (Spain)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
122	Johnson and Rogers	(2000)	Total Volatile Organic Compounds	Lawrence Livermore National Laboratory and Vicinity (USA)
123	Lischeid	(2001)	SO_4	Lehstenbach Catchment (Germany)
124a	De Silets et al.	(1992) [1]	Salinity (bottom)	Chesapeake Bay (USA)
124b	De Silets et al.	(1992) [1]	Salinity (total)	Chesapeake Bay (USA)
125	Maier and Dandy	(1996a)	Salinity	River Murray (Australia)
126	Maier and Dandy	(1996b) [1]	Salinity	River Murray (Australia)
127a	Bastarache et al.	(1997) [1]	Salinity	Moose Pit Brook (Canada)
127b	Bastarache et al.	(1997) [1]	Salinity	Pine Martin Brook (Canada)
127c	Bastarache et al.	(1997) [1]	pH	Moose Pit Brook (Canada)
127d	Bastarache et al.	(1997) [1]	pH	Pine Martin Brook (Canada)
128	Maier and Dandy	(1997a)	Salinity	River Murray (Australia)
129	Maier and Dandy	(1998a)	Salinity	River Murray (Australia)
130	Maier and Dandy	(1998b)	Salinity	River Murray (Australia)
131	Maier and Dandy	(1999)	Salinity	River Murray (Australia)
132a	Maier and Dandy	(2001)	Salinity	River Murray (Australia)
132b	Maier and Dandy	(2001)	Cyanobact. conc.	River Murray (Australia)
133	Moatar et al.	(1999)	pH	Middle Loire River (France)
134	Cannon and Whitfield	(2001)	pH	Kanaka Creek (Canada)
135	Ruck et al.	(1993)	Macro-invertebrates	Rivers in the UK
136	Zhang and Stanley	(1997)	Raw-water colour	North Saskatchewan River (Canada)
137	Zhang and Stanley	(1999)	Raw-water colour	Rossdale (Canada)
138	Lu et al.	(1998)	Potassium chloride	N/A
139	Lertpalangsunti et al.	(1999)	Water demand	Regina (Canada)
140a	Rodriguez et al.	(1997a)	Residual chlorine	Quebec (Canada)
140b	Rodriguez et al.	(1997a)	Residual chlorine	Sainte Foy (Canada)

Continued

Ref.	Author(s)	Year	Variable	Location(s)
141	Rodriguez et al.	(1997b)	Residual chlorine	England
142	Gumrah et al.	(2000)	Chlorine conc., hydraulic head	N/A
143	Corchado and Fyfe	(1999)	Temperature	N/A
144	Altendorf et al.	(1999)	Soil water content	Oklahoma (USA)
145	Islam and Kothari	(1999)	Soil water content	Southern Great Plains (USA)
146	Koekkoek and Booltink	(1999)	Soil water content	Scotland, Netherlands
147	Vila et al.	(1999)	Soil water content	Languedoc Plain (France)
148	Pachepsky et al.	(1999)	Soil hydraulic conductivity	N/A
149	Dibike et al.	(1999)	Wave equations from hydraulic data	N/A
150	Bruton et al.	(2000)	Pan evaporation	Georgia (USA)
151	Gabrys and Bargiela	(1999)	Water distribution network states	N/A
152	Brion and Lingireddy	(1999)	Land use of catchment	Jacobson Reservoir (USA)

[1] Papers included in the review conducted by Maier and Dandy(2000b)
[2] Papers included in the review conducted by Dawson and Wilby(2001)

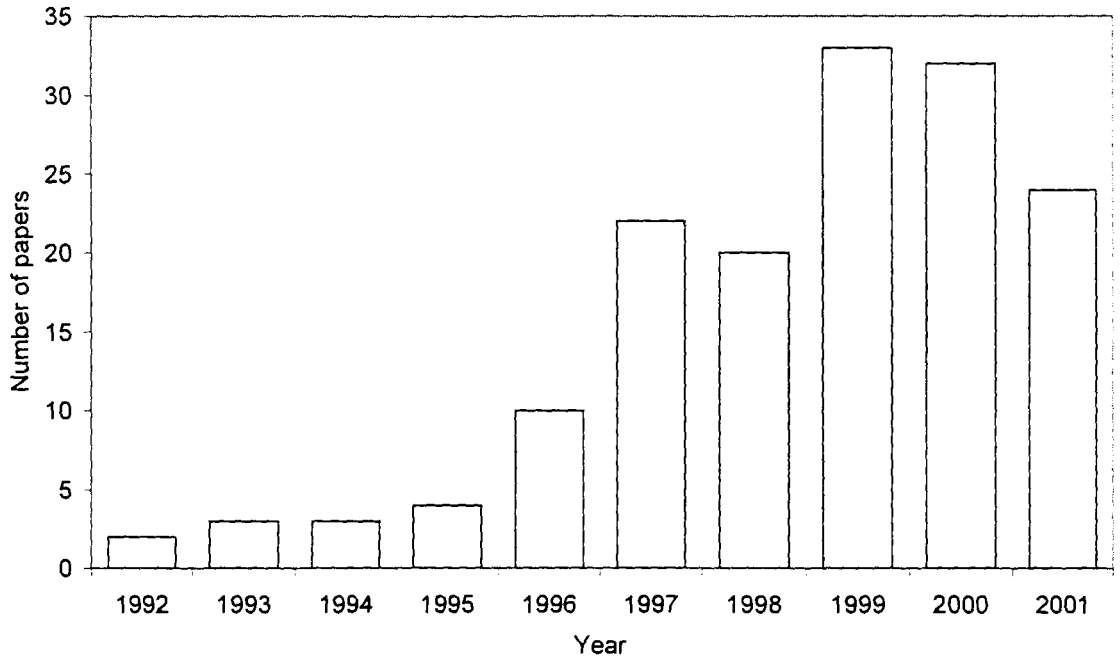

Figure 2.14 Distribution of Papers Reviewed By Publication Date

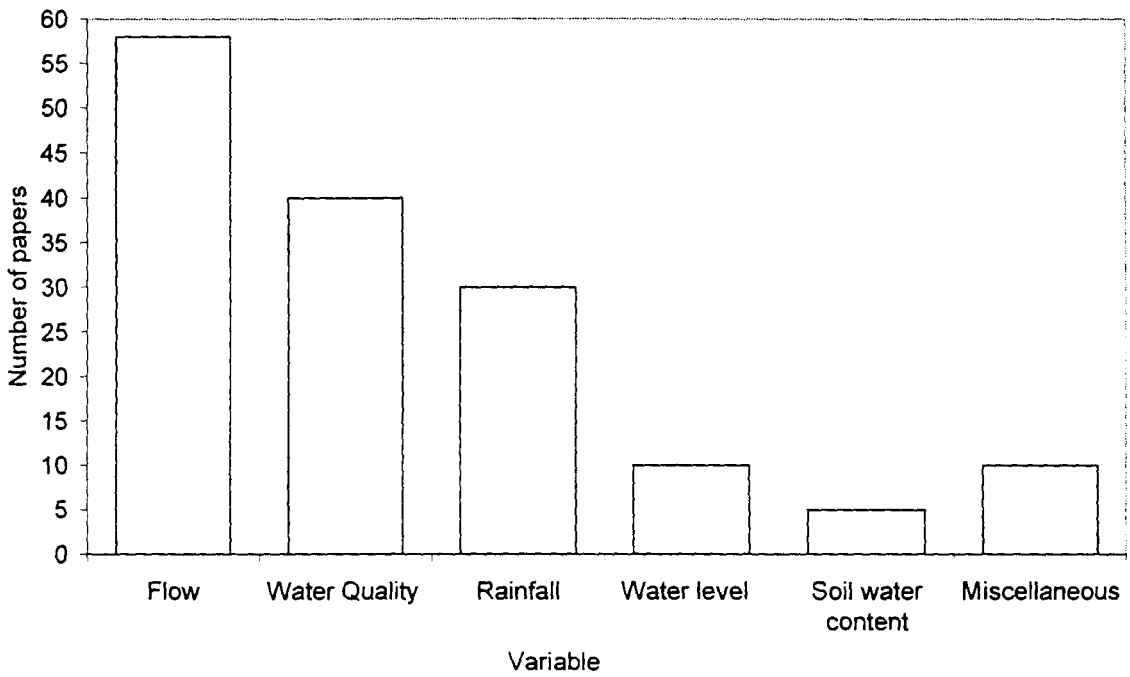

Figure 2.15 Distribution of Papers Reviewed By Variable Modelled

Dowla and Rogers (1995) suggest there are three stages that most new technologies go through. As Govindaraju and Rao (2000) point out, the same is true for ANNs in water resources modelling. In the first few years, the new technology is applied to old problems previously solved by existing methods. These applications are useful for demonstrating that the technique has the ability to outperform existing methods.

Subsequently, the new technology is applied to unsolved problems amenable to solution by the new method, and finally, the technology is applied to new problems. The review undertaken illustrates that ANNs have moved through the first and second stages and are well into the third stage, whereby they are being used to tackle an increasing number of new problems. However, up until now, the emphasis has largely been on the application rather than on the method. In fact, in a number of papers, the methodology was flawed. The main areas of concern include: using validation data in some way during the training process; using a fixed number of iterations as the stopping criterion and then (in some cases) comparing the results of models developed with different inputs, architecture and internal parameters; arbitrarily selecting the model inputs, architecture and internal model parameters; and finally, scaling the input data to the extreme range of the transfer function used in the output layer. In general terms, little attention is paid to the important areas of data transformation and preprocessing, division of the available data into training, testing and validation subsets, selection of the appropriate model inputs and architecture and choice of a suitable optimisation algorithm. In the words of Maier and Dandy (2000b) "... there needs to be a change in the mindset of users from the application of basic ANN models to an ever increasing number of case studies, to the development of guidelines for ANN modellers". Some authors (e.g. Abrahart et al., 1999; Coulibaly et al., 2001c; Dawson and Wilby, 2001; Elshorbagy et al., 2000a; Imrie et al., 2000; Kocjancic and Zupan, 2000; Luk et al., 2000; Maier and Dandy, 1997a; Maier and Dandy, 1998a; Maier and Dandy, 1998b; Maier and Dandy, 1999; Vila et al., 1999) have begun to address the lack of suitable guidelines by examining various aspects of ANN modelling methodology, however, this is only the first step in a continued research effort.

Recently, the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, (2000b) reviewed papers on the hydrologic application of ANN models. The broad areas of application reviewed included rainfall-runoff modelling, streamflow prediction, groundwater modelling, water quality forecasting, precipitation forecasting and other miscellaneous hydrologic applications. It was found that in most cases ANNs performed as well as existing models but because the knowledge encapsulated within the ANN's weights is difficult to access, they should not be considered as a panacea for hydrologic problems. Rather the Task Committee preferred to perceive ANNs as an alternative modelling tool worthy of further investigation. This led the Task Committee to raise a number of future avenues of ANN research/application to further the role of ANNs in hydrology. These areas were identified by proposing the following questions (ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b):

1. Can ANNs be made to reveal any physics?
2. Can an optimal training set be identified?
3. Can ANNs improve on time series analysis?
4. Can training of ANNs be made adaptive?
5. Are ANNs good extrapolators?

As direct attempts are made to develop a robust methodology for the application of ANNs to water resources variables, these questions will begin to be unraveled and new questions may begin to arise. Because of the complex nature of hydrological problems, the emphasis in the future will still need to be on prediction accuracy rather than on statistical optimality, however, statistical principles will need to play an important role in the development of a suitable methodology. The weight of future research must lie in the development of a methodology appropriate for the successful design and implementation of ANNs for a wide variety of water resources problems likely to be encountered in the real world.

Chapter 3

ANN Methodology for Modelling Water Resources Variables

3.1 Introduction

This chapter presents a methodology for the development of ANN models in water resources applications. In the methodology, the major steps required to develop an ANN model are considered. In addition, the current limitations involved with each step are identified and then addressed through the development of new approaches, or by the implementation of existing approaches in a novel way.

A new step has been introduced in this methodology, which involves testing the available time series data for evidence of nonlinearity. This is the first step in the methodology, and can be used to gain valuable insight into the nature of the data under investigation. It has important ramifications regarding the type of model that should be used, and the complexity of the required model.

3.2 Testing For Nonlinearity

3.2.1 Introduction

In the last decade, ANNs have been used extensively to forecast/predict a wide range of variables in the hydrology and water resources fields. However, ANNs should not be perceived as a *panacea*, but rather, as one of a number of similar tools in a toolkit for tackling problems in water resources modelling. As suggested by Maier and Dandy (2000b), there is a need to develop methods for determining when ANN models should and should not be used. Until now, research efforts have largely been focussed on the application of ANNs to an ever increasing number of case studies, without regard to the need to use such models in preference to more traditional methods.

In order to make an informed decision about the appropriate modelling methodology, there is a need for good knowledge regarding the data under investigation. For example, if the data can be generated from a linear model, then it is probably advantageous to use traditional statistical methods that are well equipped for dealing with linear processes. Even though it is possible to use ANNs without consideration of the nonlinearity of the data, results from empirical trials suggest that there is nothing to gain from using ANNs to model a linear process. Weigend et al. (1990) compared ANNs with simple linear regression for nonlinear sunspot data and a chaotic time series arising from a computational ecosystem. To analyse the ability of ANNs to model nonlinear time series, they defined a $\Re$-measure, as given by:

$$\Re = \frac{\text{residual variance (nonlinear model)}}{\text{residual variance (linear model)}} \tag{3.1}$$

where the value of $\Re$ should lie between 0 and 1, since the performance of a model should improve if additional nonlinearities are used. Weigend et al. (1990) compared both a linear model and an ANN to model a linear autoregressive AR(2) process. In this case, they obtained an $\Re \approx 1.0$ as expected, indicating that no further improvement resulted from the use of a nonlinear ANN model. In addition, if the data are found or are known to have linear dependence, a linear model should be applied as such models are usually simpler to employ, involve fewer parameters and are preferred according to Occam's razor (that is, assuming both the linear and nonlinear models are of equal performance). Furthermore, the methods for the development of linear time series models are well established.

If, on the other hand, the data have nonlinear serial dependence, then the use of ANNs should be considered as an alternative to traditional statistical methods. This is because ANNs are flexible nonlinear models that are capable of universal function approximation (Cybenko, 1989; Hornik, 1991; Hornik et al., 1989; Leshno et al., 1993; White, 1989). The methods for the development of traditional nonlinear models are not well established and practitioners are often faced with the difficult task of choosing from a large number of alternatives. It has been shown that ANNs provide a unifying framework for the implementation of a wide range of traditional statistical models but unlike traditional statistical methods, ANNs have the added advantage that there is no need to specify the particular form of the model *a priori* (Cheng and Titterington, 1994; Sarle, 1994). Instead, the optimal model type is chosen based on the data itself. Results from empirical trials suggest that for data with nonlinear serial dependence, ANNs are capable of outperforming traditional methods. For example, Weigend et al. (1990) found that for most of the sunspot data and all of the chaotic data the value of $\Re$ was significantly less than 1, indicating improved performance of the ANN models over linear regression for these data sets with nonlinear serial dependence. This finding was in agreement with Hill et al. (1996), who stated that ANNs should make better forecasts if the underlying model is nonlinear and that this improvement will become increasingly more apparent as the forecast horizon is lengthened. This is because the predictions of nonlinear models will increasingly depart from those of linear models over time. Alon et al. (2001) also investigated the use of an ANN model and compared it with three traditional linear methods (Winter's exponential smoothing, Box-Jenkins ARIMA model, and multivariate regression) to forecast aggregate retail sales. They found that the ANN model outperformed the traditional linear methods and was able to capture dynamic nonlinear trends and seasonal patterns, as well as the interactions between them.

Recent advances in statistics have given rise to a number of algorithms suitable for addressing whether nonlinear modelling techniques are justified for univariate time series. Here the question only concerns whether a particular time series contains evidence of nonlinear serial dependence and not what nonlinear model will provide a representation of the actual generating mechanism responsible for the data. It is important to note that in the analysis of nonlinear time series, there is no comprehensive model identification algorithm analogous to the method proposed by Box and Jenkins (1970) for linear processes.

Recently, Ashley and Patterson (2001) compared the power of several nonlinearity tests against a number of alternative data generating processes commonly employed in the time series literature. The nonlinearity tests investigated included the test for

autoregressive conditional heteroskedasticity (ARCH) effects proposed by McLeod and Li (1983), the LM test for ARCH effects (Engle, 1982), the BDS test proposed by Brock et al. (1996), the Tsay (1986) test for quadratic and serial dependence, the bicovariance test due to Hinich (1996), and the bispectral test proposed in Hinich (1982). A review of each of these tests can be found in Ashley and Patterson (2001). They found that the BDS test had the greatest power in diagnosing nonlinearity when tested on a variety of data generating processes, and Ashley and Patterson (2001) concluded that "the BDS test is the best test of this group for use as a nonlinearity screening test". While the BDS test is useful for detecting nonlinearity, it is unable to discriminate between different types of nonlinearity i.e. whether the process is nonlinear deterministic, chaotic or nonlinear stochastic with some degree of noise. In addition, the BDS test is unable to provide information on the degree of noise present in the data.

Recently, Kaboudan (1999) developed a fuzzy classifier that is capable of classifying data into broad classes of time series structures. In addition to determining whether data are linear or not, the classifier is able to diagnose the degree and type of nonlinearity. When tested on synthetic data generated by linear stochastic, chaotic, nonlinear stochastic and independent and identically distributed (iid) models, the fuzzy classifier correctly diagnosed all structures. Kaboudan's fuzzy classification system (FCS) is also able to determine the amount of noise present in the data and can be extended to explicitly determine the signal-to-noise ratio of a given data set.

Testing for nonlinearity in time series data has been performed rarely in the water resources modelling literature. Sivakumar et al. (1999) used the method of surrogate data in an attempt to detect nonlinearity in daily rainfall data from six stations in Singapore. In the method of surrogate data (Theiler et al., 1992), firstly a null hypothesis is specified, such as a linear process. Then surrogate data sets are generated which are consistent with this null hypothesis, and finally, a discriminating statistic is evaluated for the original and each of the surrogate data series. If the discriminating statistic is significantly different for the original time series when compared to the surrogates, then the null hypothesis is rejected and nonlinearity is detected. However, in the study conducted by Sivakumar et al. (1999), the generated surrogate data only preserved the stochastic properties of the original rainfall data and not the linear autocorrelation. Hence, the null hypothesis in this implementation is one of temporally independent data. Their test is not a test for nonlinearity as reported, but rather, a test for any dynamics at all (linear and nonlinear). Kugiumtzis (1999) has shown that the method of surrogate data lacks robustness as a method for detecting nonlinearity, especially when a single discriminating statistic is used. Results on synthetic and real-

world data were found to vary depending on the method used and its parameters, the algorithm used to generate the surrogate data and the data of the process under investigation. In addition to seldomly being used in the water resources literature, nonlinearity tests have not been used to make informed decisions regarding the ANN model building process.

The objectives of the research presented in Section 3.2 are:

- To use the BDS test and Kaboudan's fuzzy classification system as a means for determining whether ANNs should be considered as viable alternatives to conventional methods when modelling water resources time series.
- To illustrate how the tests work and how they can be applied to real-world data sets.
- To provide a comparison of the tests on both generated synthetic data and real-world data; and to suggest how the additional information gained from Kaboudan's fuzzy classification system can be used in ANN model development.

3.2.2 Methods

In general, time series data generating processes may be linear or nonlinear, and deterministic or stochastic (Kaboudan, 1999). Traditional time series methods are well developed for detecting linear structure and both the BDS test and Kaboudan's Fuzzy classifier require that the time series data are linearly filtered by using an autoregressive - moving average (ARMA) model of order (p, q), given by:

$$X_i = \sum_{k=1}^{p} a_k X_{i-k} + \sum_{m=1}^{q} b_m e_{i-m} + e_i \tag{3.2}$$

where X is the time series data, a_k and b_m are the autoregressive and moving average coefficients, respectively, and e is a random noise process with a mean of zero and variance σ^2. The ARMA model order can be selected by minimising an appropriate statistic. In this study, Schwarz's approximation of the Bayesian Information Criterion Statistic (BIC) (Schwarz, 1978) was used. This is given by:

$$BIC = -2\ln[\text{maximum likelihood}] + 2n_p \ln(n) \tag{3.3}$$

where n_p is the total number of AR and MA parameters ($p + q$), and n is the number of observations. It is possible to compute the BIC for a large number of lags p and q. If a linear structure is detected, the BIC will reach a minimum and then start to increase. The minimum BIC point identifies the the order (p, q) to use in the linear filtering.

Using the estimated equation it is possible to calculate the residuals after linear filtering. Both the BDS test and Kaboudan's fuzzy classifier are then applied to these residuals known to be "linear-free" data.

Both of the complexity tests employed herein are based on the correlation integral (Grassberger and Procaccia, 1983). The correlation integral was introduced to attempt to measure the fractal dimension of deterministic data by measuring the frequency with which temporal patterns are repeated. The integral uses phase-space reconstructions of the time series. For a time series X_i, where $i = 1, 2, \ldots, n$, the phase space can be constructed by utilising the method of delays, which is given by:

$$\boldsymbol{X}_j = (X_j, X_{j+\tau}, X_{j+2\tau}, \ldots, X_{j+(m-1)\tau}) \tag{3.4}$$

where $j=1,2, \ldots, n-(m-1)\tau$, m is the dimension of $\boldsymbol{X}_j$ and is called the embedding dimension and τ is the delay (lag) time taken to be some suitable multiple of the sampling time. To simplify the calculation it is common to set $\tau = 1$. The correlation integral is then estimated using:

$$C(m,\varepsilon,n) = \lim_{n\to\infty} \frac{1}{n^2} \sum_{k,j} H(\varepsilon - \|\boldsymbol{X}_k - \boldsymbol{X}_j\|) \tag{3.5}$$

where $H(\cdot)$ is the Heaviside unit step function with $H(y) = 1$ for $y>0$ and $H(y) = 0$ for $y\leq 0$, where $y = \varepsilon - \|\boldsymbol{X}_k - \boldsymbol{X}_j\|$, $\|\boldsymbol{X}_k - \boldsymbol{X}_j\|$ denotes the suprenum norm, $1 \leq j, k \leq n$, n is the number of data points, ε is a separation scale, and m is the embedding dimension used to create the phase-space reconstruction of the time series.

3.2.2.1 The BDS test

The BDS test (Brock et al., 1996) is well known in the time series literature. The test uses the correlation integral to estimate a coefficient W. W is defined as follows:

$$W(m,\varepsilon,n) = \frac{B(m,\varepsilon,n)}{S_m} \tag{3.6}$$

where

$$B(m,\varepsilon,n) = \sqrt{n}\left[C(m,\varepsilon,n) - C(1,\varepsilon,n)^m\right] \tag{3.7}$$

and S_m is the standard deviation of $B(m,\varepsilon,n)$. Under the null hypothesis of pure whiteness (iid random), the coefficient $W(m,\varepsilon,n)$ converges asymptotically to the standard normal distribution $N(0,1)$ with a mean of zero and unit variance. If the process is non iid, then at a given level of significance, say 5%, the absolute value of the estimated statistic will be larger than the critical Z, and the null hypothesis is rejected. The sampling distribution of the BDS test statistic is not known for the nulls of nonlinearity, chaos or linearity. However, if the original series is prefiltered using an ARMA model such that all linear structure is removed, the BDS test can then be used to determine if evidence exists of any remaining dependence in the data. If all linear serial dependence has been removed, any remaining dependence must be nonlinear, this includes, nonlinear deterministic, nonlinear stochastic and chaotic time series structures. If there is no linear structure to filter out and the null hypothesis not rejected, then the process is assumed to be iid random.

The BDS test has two variables that must be selected; the embedding dimension m and the separation scale ε. Selecting appropriate values of these parameters can be a complication in using the BDS test. In this application, the approach used by Barnett et al. (1997) has been adopted and ε has been set equal to the standard deviation of the data. In accordance with Barnett et al. (1997), at this chosen setting of ε, the BDS test statistic has been calculated for all values of the embedding dimension from 2 to 8.

3.2.2.2 Fuzzy Classification System (Kaboudan, 1999)

The FCS proposed by Kaboudan (1999) utilises two subclassifiers to detect the degree of linearity and the degree of nonlinearity, respectively. To detect the degree of linearity, the r^2 statistic is used, as given by:

$$r^2 = \left(\frac{\left[n(\sum XY) - (\sum X)(\sum Y)\right]^2}{\left[n\sum X^2 - (\sum X)^2\right]\left[n\sum Y^2 - (\sum Y)^2\right]} \right)^2 \tag{3.8}$$

where X is the actual time series data set, Y is the estimated set and n is the number of observations. The r^2 statistic depicts the degree of linearity in the series as it measures the percentage of variation captured by the ARMA model. To classify the r^2 values, they are heuristically divided into four classes: high; average; low; and, very low. If the r^2 value is high, then a strongly linear (SL) component is present in the time series. Other classes include fairly linear (FL), weakly linear (WL), and not linear (NotL). Each class is assigned members with membership functions that are mutually exclusive and collectively exhaustive, however, the classes' boundaries are made to overlap. For

example, if a r^2 value of 0.75 is obtained from linear filtering, then intuitively this data set is a member of the SL and FL classes but not a member of WL or NotL.

To detect the degree of nonlinearity, the recently introduced θ statistic is used (Kaboudan, 1995). The θ statistic is based on the correlation exponent equation developed in chaos theory, as given by:

$$\nu = \frac{\ln C(m,\varepsilon_1,n) - \ln C(m,\varepsilon_2,n)}{\ln(\varepsilon_1) - \ln(\varepsilon_2)} \tag{3.9}$$

where $C(m,\varepsilon,n)$ is a correlation integral defined in Equation 3.5 and the constants ε_1 and ε_2 are selected such that $C(\cdot)>0$. In accordance with Brock et al. (1991), ε_1 is selected as the standard deviation of the data (σ) and ε_2 selected equal to 0.75σ. To compute the θ statistic, the correlation exponent is measured twice, once for the residuals after linear filtering and again for a random shuffle of the residuals. For the random shuffle, the correlation exponent is measured using the same values for ε_1 and ε_2. The shuffling is achieved using the bootstrap method outlined in Efron and Tibshirani (1993). If determinism is present in the residuals, the shuffling process destroys it. In this case, the ratio of the two exponents will be <1 and will approach zero if the data are noise free. If there is no determinism remaining in the residuals (iid random), the shuffling does not destroy any deterministic pattern and the ratio of the two exponents will approximately equal 1. Because the shuffling process is achieved using a random number generator, a distribution of similar θ values will result and it is necessary to complete a large number of shuffles per series to obtain a reasonable estimate of θ. Each value of θ is itself an average of the ratios obtained at embedding dimensions m=2, ..., 10 because the embedding dimension at which the integral estimate starts to change as a result of shuffling is not known (Kaboudan, 1999). A single θ estimate is given by:

$$\theta_k = (M-1)^{-1} \sum_{m=1}^{M} \frac{\ln C_Y(m,\varepsilon_1,n) - \ln C_Y(m,\varepsilon_2,n)}{\ln C_S(m,\varepsilon_1,n) - \ln C_S(m,\varepsilon_2,n)} \tag{3.10}$$

where S is the randomly shuffled set of Y and M is the maximum embedding dimension used. The average θ is calculated by completing $k=1, \ldots, K$ shuffles. Kaboudan (1999) used K=1000 and the average $\theta = \sum\theta_k/K$ to classify nonlinearity. In a separate study, Kaboudan (1998) found that the θ statistic was asymptotically normally distributed for all data structures and at a series' sample size of $n \geq 1000$.

Like the r^2 statistic, θ is also in the [0,1] interval and is divided into six classes with fuzzy boundaries including very low, low, average, above average, high, and very high. A very low value of θ indicates that the data contain no noise and are nonlinear (NL). The other classes used include chaotic (CHT), nonlinear with low noise (NL-LN), medium noise (NL-MN), high noise (NL-HN), and white noise (WN). These classes were identified by empirical evidence deduced from previous studies (see Kaboudan, 1995).

Table 3.1 contains the implication rules for defining the levels of linearity and nonlinearity. These have been translated into the membership functions shown in Table 3.2. In total, Kaboudan (1999) has identified 42 membership rules. To classify a series, the r^2 obtained triggers off one or more of rules 1-16 and the θ value estimated from the residuals will fire one or more of rules 17-42. The linear and nonlinear classifications' output is then combined using the standard max-min methodology to produce a symbolic representation of the data series' dynamic structure.

Table 3.1 The Implication Rules (source: Kaboudan, 1999)

Rule no.	IF	THEN
1	r^2 is high	The process is SL
2	r^2 is average	The process is FL
3	r^2 is low	The process is WL
4	r^2 is very low	The process is NotL
5	θ is very low	The process is NL
6	θ is low	The process is CHT
7	θ is average	The process is NL-LN
8	θ is above average	The process is NL-MN
9	θ is high	The process is NL-HN
10	θ is very high	The process is WN

Table 3.2 The Membership Rules (source: Kaboudan, 1999)

Class	{Members of Class	→ Degree of Membership	Rule no.
SL	$r^2 \le 0.70$	→ 0	1
	$0.70 < r^2 \le 0.90$	→ (1-(0.90-r^2)/0.2)	2
	$r^2 > 0.9$	→ 1	3
FL	$r^2 \le 0.40$	→ 0	4
	$0.40 < r^2 \le 0.60$	→ (1-(0.60-r^2)/0.2)	5
	$0.60 < r^2 \le 0.70$	→ 1	6
	$0.70 < r^2 \le 0.80$	→ (1-(r^2-0.70)/0.1)	7
	$r^2 > 0.80$	→ 0	8
WL	$r^2 \le 0.10$	→ 0	9
	$0.10 < r^2 \le 0.30$	→ (1-(0.30-r^2)/0.2)	10
	$0.30 < r^2 \le 0.40$	→ 1	11
	$0.40 < r^2 \le 0.50$	→ (1-(0.40-r^2)/0.1)	12
	$r^2 > 0.9$	→ 0	13
NotL	$r^2 > 0.15$	→ 0	14
	$0.08 < r^2 \le 0.15$	→ (1-(r^2-0.08)/0.07)	15
	$r^2 \le 0.08$	→ 1	16
NL	$\theta > 0.40$	→ 0	17
	$0.20 < \theta \le 0.40$	→ (1-(θ-0.20)/0.2)	18
	$\theta \le 0.20$	→ 1	19
CHT	$\theta > 0.65$	→ 0	20
	$0.50 < \theta \le 0.65$	→ (1-(θ-0.50)/0.15)	21
	$0.40 < \theta \le 0.50$	→ 1	22
	$0.30 < \theta \le 0.40$	→ (1-(0.4-θ)/0.1)	23
	$\theta \le 0.30$	→ 0	24
NL-LN	$\theta > 0.75$	→ 0	25
	$0.65 < \theta \le 0.75$	→ (1-(θ-0.65)/0.1)	26
	$0.55 < \theta \le 0.65$	→ 1	27
	$0.45 < \theta \le 0.55$	→ (1-(0.55-θ)/0.1)	28
	$\theta \le 0.45$	→ 0	29
NL-MN	$\theta > 0.85$	→ 0	30
	$0.75 < \theta \le 0.85$	→ (1-(θ-0.75)/0.1)	31
	$0.65 < \theta \le 0.75$	→ 1	32
	$0.55 < \theta \le 0.65$	→ (1-(0.65-θ)/0.1)	33
	$\theta \le 0.55$	→ 0	34
NL-HN	$\theta > 1.0$	→ 0	35
	$0.95 < \theta \le 1.0$	→ (1-(θ-0.95)/0.05)	36
	$0.85 < \theta \le 0.95$	→ 1	37
	$0.75 < \theta \le 0.85$	→ (1-(0.85-θ)/0.1)	38
	$\theta \le 0.75$	→ 0	39
WN	$\theta \le 0.95$	→ 0	40
	$0.95 < \theta \le 1.0$	→ (1-(1.0-θ)/0.05)	41
	$\theta > 1.0$	→ 1	42

3.2.3 Application of the Nonlinearity Tests to the Synthetic Test Data

In order to demonstrate how the BDS test and Kaboudan's FCS work, they were applied to four synthetic data sets, where the dynamics were known *a priori*. Data sets I, II and III each consist of a sample size of 2000 points and were generated using the models specified below. Data set IV consists of 4000 points, and was generated using a surrogate data algorithm. It is important to note that the surrogate data were not used to detect nonlinearity, but rather to compare the results of the BDS test and Kaboudan's FCS when tested on a data set that was known to be linear by construction.

Data Set I:

The model used to generate these data was the fully deterministic, chaotic Feigenbaum recursion of the form:

$$y_t = 4y_{t-1}(1-y_{t-1}) \tag{3.11}$$

with $y_0 = 0.1$.

Data Set II:

The model used to generate these data was an ARMA(2,1) model of the form:

$$y_t = 0.8y_{t-1} + 0.15y_{t-2} + e_t + 0.3e_{t-1} \tag{3.12}$$

with $y_0 = 1$, $y_1 = 0.7$ and where e_t is a random noise process with a mean of zero and unit variance.

Data Set III:

The model used to generate these data was the chaotic Mackey-Glass differential delay equation of the form:

$$\frac{dy}{dt} = \frac{ay(t-\tau)}{1+[y(t-\tau)]^{10}} - by(t) \tag{3.13}$$

with $a = 0.2$, $b = 0.1$, $\tau = 30$.

To solve the Mackey-Glass equation, a fourth-order Runge-Kutta procedure was employed with an integration step $\Delta t = 0.05$. The initial value was set to $y_0 = 1.2$, and the integration process sampled the data every $N = \tau / \Delta t$, or every 600 points.

Data Set IV:

These data are a surrogate of the univariate salinity time series at Murray Bridge for the period 01-12-1986 to 31-12-1997. The surrogate data were generated under the null hypothesis of a linear Gaussian stochastic process i.e. all the structure in the time series is given by the autocorrelation function, or equivalently by the Fourier power spectrum and any nonlinear phase relations that may exist are destroyed. The algorithm used to create the surrogate is the unwindowed Fourier transform (FT) described in Theiler et al. (1992). Firstly, the Fourier transform of the data is calculated for positive and negative frequencies $f = 0, 1/n, 2/n, ..., 1/2$, and without the benefit of windowing. Then to randomise the phases, each complex amplitude is multiplied by $e^{i\phi}$, where ϕ is randomly and independently chosen for each frequency from the interval $[0,2\pi]$. To ensure the inverse Fourier transform is real (with no imaginary components), the phases are made symmetrical, so that $\phi(f) = -\phi(-f)$. Finally, to produce the surrogate data, the inverse Fourier transform is calculated.

3.2.3.1 BDS Test Results

The results obtained using the BDS test on the synthetic data series are given in Table 3.3. The BDS test identified nonlinearity in data sets I and III by strongly rejecting the null hypothesis of iid random. The BDS test did not reject the null hypothesis of iid random for data sets II and IV. This is a good result for this test as data sets I and III were the nonlinear chaotic data and data sets II and IV were the synthetic linear data. These results are consistent with the findings in the literature, confirming the high power of the BDS to discriminate between chaotic and iid alternatives.

It is important to note that the precise order of the linear model was known for data set II, however, the test's prior linear filter was not estimated to be the correct ARMA(2,1) process. There may be a danger of incorrectly rejecting the null hypothesis if the correct linear filter is not used as some of the test's sensitivity to nonlinearity may result from any linear dependence still remaining in the data (Barnett et al., 1997). The results obtained for data set II were not affected by this and the BDS test was still able to successfully accept linearity for this data set.

Table 3.3 BDS Test Z Statistics, Residuals from ARMA Fit to Synthetic Data Sets

Data Set	Fitted ARMA Order (p, q)	Embedding Dimension	BDS Z Statistic	Decision
I (Feig)	(0, 0)	2	290.51	Reject linearity (very strongly)
		3	280.64	
		4	272.07	
		5	265.23	
		6	268.92	
		7	279.17	
		8	291.10	
II (ARMA)	(1, 1)	2	-0.13	Accept linearity
		3	-0.61	
		4	-0.85	
		5	-0.60	
		6	-0.69	
		7	-0.76	
		8	-0.65	
III (M-G)	(0, 0)	2	31.31	Reject linearity (very strongly)
		3	31.69	
		4	38.36	
		5	45.49	
		6	52.72	
		7	65.34	
		8	83.88	
IV (Salin. Surrogate)	(2, 0)	2	-0.89	Accept linearity
		3	-0.74	
		4	-1.23	
		5	-1.22	
		6	-0.95	
		7	-0.85	
		8	-0.84	

3.2.3.2 Kaboudan's FCS Results

The results with the FCS applied to the synthetic data sets are presented in Table 3.4. The FCS correctly diagnosed data sets I and III as chaotic and the two linear data sets (II and IV) were correctly identified as strongly linear with a white noise component. This is a good result for the FCS and is indicative of the power of this system in discriminating between different time series structures. The FCS had the added advantage over the BDS test because it was able to provide additional information regarding the degrees of linearity and nonlinearity. For example, for data set II, the

BDS test did not reject the null hypothesis of linearity while the FCS was able to provide additional information, indicating that the data set was strongly linear with a white noise component. For data sets I and III, the BDS simply rejected the null hypothesis of linearity while the FCS indicated that both data sets were chaotic. The additional information that the FCS is able to provide is an important advantage over the BDS test.

Table 3.4 Diagnosis Results of the FCS For the Synthetic Data Sets

Data Set	Fitted ARMA Order (p, q)	r^2	θ	Final Diagnosis
I (Feig)	(0, 0)	0.00	0.36	CHT
II (ARMA)	(1, 1)	0.88	1.00	SL-WN
III (M-G)	(0, 0)	0.00	0.49	CHT
IV (Salin. Surrogate)	(2, 0)	0.99	1.01	SL-WN

3.2.3.3 Nonlinearity Testing and ANN Modelling

Given the information derived from the nonlinearity tests, it is useful to address how this information can be employed in developing ANN forecasting models. Such additional information can be used in a number of ways, including: to gauge the likely success of the ability of the model to provide accurate forecasts; to determine appropriate forecasting periods; and finally, to determine the appropriate ANN architecture to use. For example, if the data are found to be independent and identically distributed (iid) random, then good model performance cannot be expected. If the data are found to have a chaotic structure, then there are two related consequences to consider (Weigend et al., 1990). Firstly, since the uncertainty of the prediction increases exponentially as the forecasting horizon is increased, chaos will preclude any long-term predictability. Secondly, short-term predictability is possible if the model is able to capture the short-term structure of the time series. Therefore, the forecasting horizon needs to be selected accordingly. If the data are found to be noisy, then the ratio of the number of weights to the number of training samples should be reduced in order to avoid overfitting.

The FCS provides a complexity measure θ of the time series being modelled and as such, can be used to determine the complexity of the required ANN. This provides an intuitive means of selecting the optimal number of nodes needed in the hidden layer of an ANN. Recently, incremental neural networks have been proposed in the literature (see Chentouf et al., 1997; Jutten and Chentouf, 1995). A constructive approach is used to select the optimal number of hidden nodes by adding in hidden nodes, one at a

time. In their single-hidden-layer approach, Jutten and Chentouf (1995) create a new hidden node at each step, connect it to the inputs and then train it to learn the error of the network. After convergence, the node is then connected to the existing network to reduce the error on the training set and the combined network is retrained. This step is repeated until the error of the entire network can be considered as noise and hence, no more hidden nodes are required. However, deciding upon an appropriate stopping criterion is difficult since in many real-world applications, the amount of noise in the data being modelled is not known. Jutten and Chentouf (1995) developed a stopping criterion based on a statistical nullity test of the estimated parameters of the new hidden node. However, a stopping criterion based directly on the noise in the data does not exist at present.

It is proposed in this thesis that an appropriate stopping criterion for the method of incremental neural networks can be deduced from the θ statistic. Kaboudan (1995) developed a relationship between θ and the signal-to-noise ratio λ of a time series. The signal-to-noise ratio is defined as:

$$\lambda = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{3.14}$$

Conventionally, this measure is converted to decibels (dB) and defined as ω, where

$$\omega = 10\mathrm{Log}_{10}\lambda \tag{3.15}$$

By generating noise-corrupted signals with a desired and deliberately controlled λ value and then computing the corresponding value of θ, Kaboudan (1995) was able to determine that a nonlinear relationship exists between θ and λ. After testing several model specifications, a reasonable nonlinear regression model to reproduce ω was found and included dummy variables to accommodate intercept shifting. The final equation is presented in Table 3.5.

The θ ranges that activate the dummy variables are shown in Table 3.6. Therefore, to estimate an output variables' unknown λ, the average value of θ is substituted into the equation shown in Table 3.5. The corresponding value of ω is determined and then converted into a value for λ using:

$$\lambda = 10^{\frac{\omega}{10}} \tag{3.16}$$

Table 3.5 Regression to Estimate Omega (source: Kaboudan, 1995)

Dependent Variable:	Omega		
Estimation Method:	Least Squares		
r Bar**2	0.99		
Regression F Statistic:	16528		
Durbin-Watson Statistic:	1.76		
Q Statistic:	249.52		
Variable	**Coeffic.**	**Std. Error**	**t-Stat.**
Constant	17.746	0.089	199.86
Theta	-13.546	0.132	-102.45
DV0	-6.200	0.089	-69.97
DV1	-4.524	0.070	-64.33
DV2	-2.120	0.067	-31.68
DV3	-1.077	0.063	-16.99
DV30	1.608	0.070	22.90
DV40	2.705	0.071	38.04
DV50	3.613	0.071	50.54

Table 3.6 Dummy Variable "Theta" Activation Range (source: Kaboudan, 1995)

IF	THEN Value of each dummy variable is:						
Theta Range	DV0	DV1	DV2	DV3	DV30	DV40	DV50
0.99 = or < Theta	1	0	0	0	0	0	0
0.95 = or < Theta < 0.99	0	1	0	0	0	0	0
0.90 = or < Theta < 0.95	0	0	1	0	0	0	0
0.85 = or < Theta < 0.90	0	0	0	1	0	0	0
0.33 = or < Theta < 0.34	0	0	0	0	1	0	0
0.32 = or < Theta < 0.33	0	0	0	0	0	1	0
Theta < 0.32	0	0	0	0	0	0	1
All other values	0	0	0	0	0	0	0

Given a value for λ and the fact that the variance of a data set σ^2_{data} is comprised of the variance of the signal σ^2_{signal} and the variance of the noise σ^2_{noise}, it is then possible to use Equation 3.14 to determine the variance of the noise in the output variable to be modelled. Hence, an appropriate stopping criterion for an incremental neural network is to terminate the algorithm when the variance of the ANN's error is approximately equal to the variance of the noise calculated in the actual output data (dependent variable):

$$\sigma^2_{error} \approx \sigma^2_{noise} \tag{3.17}$$

where σ^2_{error} is the variance of the ANN's residual error and σ^2_{noise} is the variance of the noise in the output data and is determined using θ. In this way, an optimal number of hidden nodes can be selected using only training data, yet still avoiding overfitting. The potential of this approach for use as a stopping criterion in an incremental neural network is only noted here. The actual application and testing of this criterion was considered to be beyond the scope of this thesis.

3.2.4 Conclusions and Recommendations

Two nonlinearity tests based on the correlation integral were used to detect the presence or absence of nonlinear dynamics in a number of synthetic univariate data sets. It was apparent that both methods exhibited consistency in their inference for all data sets tested. In addition, both tests correctly detected the presence or absence of nonlinearity in the synthetically generated data sets where the dynamics were known *a priori*. However, the advantage of Kaboudan's FCS over the BDS test is the ability to distinguish between different types of nonlinearity. For the synthetic data, Kaboudan's FCS correctly diagnosed the Feigenbaum recursion data and the Mackey-Glass equation data as chaotic, whereas the results from the BDS test simply tell us that these data are nonlinear.

The information provided by the results of these two nonlinearity tests can be used to aid the ANN model development process in a number of ways. For the data under consideration, the information gleaned from the tests can be used to decide whether it is suitable to use ANNs as a modelling tool. The information may also be used to determine the number of hidden layer nodes to select for the ANN. This second point only applies to the FCS, which has the added advantage that it can be extended to explicitly calculate the variance of the noise present in a data set. Given this capability, a new stopping criterion for incremental neural networks was introduced based on the complexity measure θ of the time series. θ is first used to calculate the signal-to-noise ratio and the estimated variance of the noise in the output variable can then be used to determine the complexity of the required ANN.

The work presented in this section has identified a number of avenues of future work. There are three important areas that require further investigation:

1. The performances of ANNs and linear models need to be compared using time series data that belong to different classifications, as identified by the nonlinearity tests.

2. The possibility of extending these tests to multivariate time series data needs further investigation. The two nonlinearity tests considered in the present study are both based on univariate time series. When a univariate model is to be developed, the decision from the nonlinearity tests is fairly straightforward. An ANN model should be used if the time series is diagnosed as nonlinear and a linear model (e.g. ARMA) should be used if the time series is diagnosed as linear. If, however, a multivariate model is being developed, as is often the case, then the results from the nonlinearity tests must be interpreted carefully. If the output time series is found to be nonlinear, and autoregressive (lagged) inputs of the time series tested are being used in the model, then an ANN model should be used. This is because the underlying model will still need to be nonlinear even with the addition of the other input variables. If the time series is found to be linear, and additional input variables are to be used, then there is no guarantee that a multivariate linear model will be suitable. This is because the additional input variables may introduce nonlinear cross-dependence that cannot be accounted for by a linear model. One possible way of diagnosing this is to prefilter the output time series using a multivariate (rather than a univariate) ARMA model and perform the nonlinearity tests on the residuals. If the residuals are still found to be iid random, then it may be appropriate to proceed with a multivariate linear model.

3. Finally, the utility of using the signal-to-noise ratio to estimate the complexity of the required ANN needs to be tested using a number of synthetic and real-world data sets.

3.3 Division of Data for ANN Models

3.3.1 Introduction

An important step that is often overlooked in ANN modelling is the division of the available data into subsets. When cross-validation is used as a stopping criterion, three data sets are needed, namely training, testing and validation sets. Three data sets are also required when optimising network architecture or internal model parameters such as the learning rate or momentum. As pointed out by Maier and Dandy (2000b), the validation data must not be used in any capacity in the model development process. The training data are used to find an optimal set of connection weights, the test data are used to choose the best network configuration, and once an optimal network has been found, a validation set is required in order to test the true generalisation ability of the model.

Recent studies have found that the way the available data are divided into subsets can have a significant influence on an ANN's performance (Maier and Dandy, 1996b; Tokar and Johnson, 1999). This is because ANNs are typically unable to extrapolate beyond the range of the data used for training (Flood and Kartam, 1994; Minns and Hall, 1996; Tokar and Johnson, 1999). For adequate generalisation ability, the training and validation sets must therefore be representative of the same population (Masters, 1993). However, Maier and Dandy (2000b) found that in most of the papers they reviewed, the data were divided on an arbitrary basis, without any consideration being given to their statistical properties. As a consequence, in many cases, the optimality of the results presented was difficult to assess.

Flood and Kartam (1994) point out that the number of training samples can significantly influence a network's performance. Increasing the number of training samples provides more information about the shape of the solution surface(s), and thus increases the potential level of accuracy that can be achieved by the network. However, in most practical circumstances, data availability and cost impose obvious limitations on the amount of data available and hence, the size of the training set. Thus, the *proportion* of samples to include in each of the subsets is an important consideration.

It has been acknowledged in the past that an ANN is susceptible to becoming "...a prisoner of its training data" (Minns and Hall, 1996). During prediction, the model is likely to perform poorly if faced with inputs that are far removed from the examples that it saw during training. By using the widest limits of examples during training, it is possible to prevent an ANN from the need to extrapolate (ASCE Task Committee on

Application of Artificial Neural Networks in Hydrology, 2000a). Consequently, *which* samples to include in the training set is also very important.

Recently, attempts have been made to ensure that the statistical properties of each subset of data are similar. Braddock et al. (1998) selected three years of data for the acceptance (validation) set so that the mean, standard deviation and range of these years contained the top, bottom and middle values of the available data. The training and testing sets were selected so that they were statistically representative of the entire data set. Campolo et al. (1999) divided daily data from January 1992 to August 1993 into training, testing and validation sets, taking care that the mean, variance, maximum and minimum value of each subset were most similar. The proportion of data in each subset was assigned arbitrarily, with the training, testing and validation subsets consisting of 400, 120 and 81 samples, respectively. Whilst this methodology determines which samples should be used in each of the subsets, the authors provide no information on how to ensure the similarity of the statistics. In addition, it does not solve the problem of choosing what proportion of data to use in each subset.

Tokar and Johnson (1999) created rainfall-runoff ANN models using wet-, dry-, and average-year data to illustrate the impact of the content of the training data on network prediction accuracy. It was observed that the wet-year models outperformed the dry- and average-year models based on goodness-of-fit statistics. The reason given was that the wet-year models included information on both high- and low-flow conditions. Consequently, it was more likely that in order to predict the patterns in the test set, an interpolation rather than an extrapolation was required.

Ray and Klindworth (2000) recognised that the training and testing sets must be representative of the same population. To achieve this, they made their training sets large enough to represent the full population and data for the testing set were randomly selected and manually checked to ensure that they had similar characteristics to the training data. No information was provided detailing how this was achieved.

Despite these studies, no systematic approach has been developed for the optimal division of data for ANN models. Recently, the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a) discussed future avenues of ANN research/application and noted the important issue of data division. The lack of an established procedure led the Task Committee to pose the following question, "Can an optimal training set be identified?" The ASCE Task Committee observed that an optimal training set would adequately represent the modelling domain but at the same time, would employ the minimum number of samples for achieving this objective.

Repetitive data only serve to slow down ANN training and by keeping the training set concise, software and hardware constraints can be satisfied more easily (Dawson and Wilby, 1998). The ASCE Task Committee elaborated further, stating, "Very often we may have no alternative but to proceed with limited data. Under these circumstances can we say when generalisation will fail so that we understand the range of applicability of the ANN?" (ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000a). Conversely, it may be equally important to ask, when a model does perform poorly, are tools available to critically dissect the model and determine if the data sets were in fact representative of the same population?

One of the objectives of this research is to present a methodology that can be used to determine the optimal division of data for ANN models. Two important considerations shall be addressed, including:

1. *What proportion* of the data should be used for each of the training, testing and validation sets?
2. *Which* samples should be used in each of the sets?

Another important objective is the development of a technique that can be used to diagnose why an ANN model has performed poorly.

Two new methods for the optimal division of available data will be explored in this research and these methods will be compared to a more conventional approach. The first method (Section 3.3.2.1) employs a genetic algorithm (GA) to divide the data so as to minimise the statistical difference (as measured by the mean and standard deviation) between training, testing and validation data sets. The second data division method (Section 3.3.2.2) employs a self-organizing map (SOM) (Kohonen, 1982), to cluster similar data records together. An equal number of data records can then be sampled from each cluster to produce training, testing and validation sets with similar statistical properties, whilst using a minimum number of data records. The conventional approach, for comparative purposes, simply involves arbitrarily dividing the data into subsets without consideration of the statistical parameters. This is the approach most often employed in the literature.

3.3.2 Methods

3.3.2.1 Data Division Using a Genetic Algorithm

A genetic algorithm is a powerful optimisation technique inspired by the principles of natural evolution and selection (Goldberg, 1989). GAs have been widely used as an optimisation tool in a number of water resources case studies (e.g. Dandy et al., 1996; Liong et al., 2001; Simpson et al., 1994). In this study, a GA has been applied to the problem of dividing the data for an ANN model into three statistically similar subsets. For example, if there are 60 data samples that must be divided into training, testing and validation sets consisting of 40, 10 and 10 data samples respectively, then there are $\frac{60!}{40! \times 10! \times 10!} = 7.7 \times 10^{20}$ ways of arranging the data samples. A GA can be used to search through this large space and determine the optimal arrangement based on an objective function. In this study, the aim is to arrange the available data into three statistically similar subsets of fixed size.

Outline of the Genetic Algorithm

The GA used for data division sorts the samples into training, testing and validation sets by using a set of pseudo random numbers. The decision variable governing the arrangement of the data samples is a pseudo random number seed, chosen to be in the range [1, 100,000]. This range was selected to provide a reasonable size search space. The GA string therefore consists of a single integer between 1 and 100,000. The pseudo random number seed controls the generation of a pseudo random sequence of numbers. To generate this sequence, pseudo random number generating software is used. The sequence of numbers generated depends on the initial seed and the particular software. The pseudo random number sequence is placed alongside the data samples and the contiguous block of data is sorted using these pseudo random numbers. In so doing, the data samples are arranged into subsets and the objective function is evaluated. Penalty constraints are added to ensure that the maximum and minimum values of each input and output variable are included in the training set, rather than in the testing or validation sets. As discussed in Section 3.3.1, training the ANN model on the extreme range of values available removes the need for the network to extrapolate. It is important to point out that there may be a trade-off between keeping the statistics of the training, testing and validation sets the same and ensuring that the extreme values are in the training set. Consequently, using penalty constraints was deemed to be a superior option for dealing with this trade-off as opposed to simply removing the extreme values from the data and placing them in the training set.

To determine the "fitness" of each solution an objective function is required. In this application, a suitable objective function to minimise is the sum of the absolute difference in mean and standard deviation values between each pair of the three subsets.

A floating point GA was used as experiments conducted by Michalewicz (1994) and Wright (1991), have shown that floating point representation is faster, more consistent run to run, provides higher precision than binary coding and allows greater freedom to use different mutation and crossover techniques based on the real representation.

The tournament selection scheme was used, where strings are paired randomly and the string with the higher fitness in the pair progresses to the next generation (Goldberg and Deb, 1991). This scheme is referred to as binary or 2-member tournament. Since only half of the strings progress, another tournament is held with another set of random pairs and the winners make up the other half of the crossover pool for the next generation. Two copies of the best string progress and no copies of the worst string are replicated. The tournament selection scheme was chosen as it has comparable growth ratios with other schemes but has a better time complexity than many other selection algorithms (Goldberg and Deb, 1991).

In this application, the linear crossover operator (Wright, 1991) has been used as it avoids problems associated with real crossover. From the two parent random number seeds, p_1 and p_2, three new random seeds are generated, namely $\frac{1}{2}p_1 + \frac{1}{2}p_2$, $\frac{3}{2}p_1 - \frac{1}{2}p_2$ and $-\frac{1}{2}p_1 + \frac{3}{2}p_2$. Since the seed needs to be an integer, the resultant random seeds are rounded to integers. The best two of the three random seeds are then chosen.

Non uniform mutation (Michalewicz, 1994), designed for use with floating point GAs, was used in this procedure. If a mutation is to occur at generation t, then a random number seed v_k is mutated to produce v'_k where:

$$v'_k = \begin{cases} v_k + \Delta(t, UB - v_k) \\ v_k - \Delta(t, v_k - LB) \end{cases} \begin{array}{l} \text{if a random digit is 0} \\ \text{if a random digit is 1} \end{array} \qquad (3.18)$$

and LB and UB are lower and upper domain bounds of the variable v_k. As described by Michalewicz (1994), the function $\Delta(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\Delta(t, y)$ being close to 0 increases as t increases. This causes the operator

to uniformly search the space initially (i.e. when t is small), and very locally at later stages, thereby increasing the probability of generating the new number closer to its predecessor than a random choice. The following function is used:

$$\Delta(t, y) = y.(1 - r^{(1-\frac{t}{T})^b}) \tag{3.19}$$

where r is a random number in the range [0,1], T is the maximum number of generations, and b is a system parameter determining the degree of dependency on the iteration number (b = 5 was used in this study).

The ANN input and output data are used in this procedure and are scaled to the range [0,1]. The reason for this is twofold. Firstly, the scaling prevents inputs/outputs with much larger values from dominating the evolutionary process. Secondly, scaling to the interval [0,1], enables penalty constraints to be included more easily (i.e. the maximum and minimum values can be identified by the GA as the zeros and ones).

3.3.2.2 Data Division Using a SOM

The use of the SOM in water resources applications has been fairly limited. Applications include the estimation of rainfall rates from infrared satellite and ground surface data (Hsu et al., 1997), rainfall-runoff modelling (Hsu et al., 2002a; Hsu et al., 2002b), the identification of flow regimes in horizontal air-water flow in an experimental pipeline (Cai et al., 1994) and the classification of flood data into classes defined by Representative Regional Catchments (RRCs) (Hall and Minns, 1999).

In this research, the SOM is used to cluster the input and output data into training, testing and validation sets that have similar statistical properties. The SOM is implemented using the *NCS NeuFrame* software. To cluster the data, the inputs and their corresponding output are presented to the network as the SOM's inputs. The software default parameters are used for the learning rate, neighbourhood size and number of epochs. The output of the SOM is obtained using a Dynamic Patterns grid, which shows a dynamic representation of the nodes that are winning each pattern (Figure 3.1). Each individual cell in the grid represents a node in the Kohonen layer. There is no theoretical principle for determining the optimum size of the Kohonen layer (Cai et al., 1994), hence, the Kohonen layer was kept large enough to ensure that the maximum number of clusters were formed from the training data. Once the clusters are formed, three data records from each cluster are sampled (i.e. one for each of the training, testing and validation sets). In the instance that a cluster only contains one

record, then this record is placed in the training set. If a cluster contains two records, then one record is placed in the training set and the other is placed in the testing set.

The sampling process used in the SOM data division method is illustrated in Figure 3.1. Each square grid represents a node in the Kohonen layer and the circles represent the data samples in each cluster. In this example, three records are sampled from cluster #1, including one record for the training set, one for the testing set and one for the validation set. This process is repeated for each cluster in the Kohonen layer.

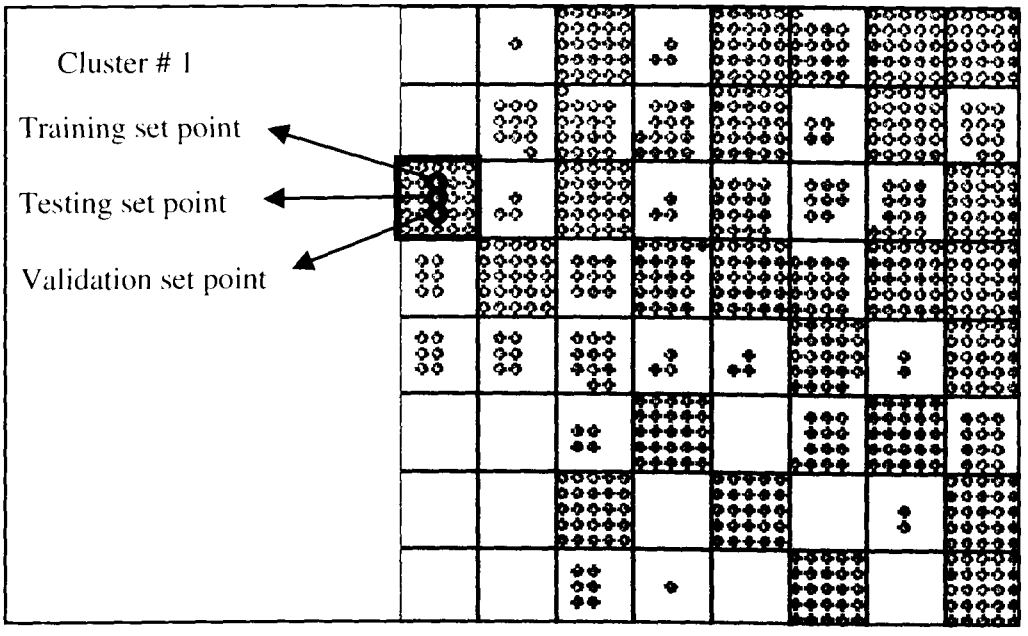

Figure 3.1 Example of the Sampling Process Used in the SOM Data Division Method

3.4 Data Transformation

3.4.1 Introduction

In the review conducted by Maier and Dandy (2000b) and the subsequent papers reviewed in this thesis, it was found that data transformations were performed rarely in the application of ANNs to the modelling of water resources variables. In some of the papers reviewed, the data were not scaled to a range commensurate with the transfer function in the output layer by using a linear transformation. In addition to this, the probability distribution of the data was not considered in any of the papers.

In the past, it has commonly been perceived in the literature that data used by ANN models do not need to be transformed. However, more recently it has been suggested that certain transformations may actually improve the performance of ANN models. Shi (2000) described three broad classes of data transformation. These include: (1) linear transformation; (2) statistical standardisation; and (3) mathematical functions.

Linear transformation is by far the most widely employed data transformation technique in ANN modelling applications. The dataset is usually scaled to the range [0,1] or [-1,1] by using the original data range as a scalar. The objective of linear transformation is to ensure that all variables receive equal attention during the training process and that the variables are scaled in a way so as to be commensurate with the limits of the transfer functions used in the output layer. For a multi layer perceptron (MLP), it is more useful to scale the data to the range [-1,1] rather than [0,1]. This is because the hidden nodes in a MLP each define a hyperplane and the connection weights from the input layer to the hidden layer determine the orientation of the hyperplane and the bias determines the distance of the hyperplane from the origin. When the network is initialised, it is usual to set the bias terms as small random numbers and hence, the hyperplanes pass close to the origin. Therefore, if the data are not centred around the origin, the hyperplanes may fail to pass through the data cloud and with such a poor initialisation, local minima are likely to occur (Sarle, 1997).

In statistical standardisation, the computation involves subtracting a measure of location, such as the mean and then dividing by a measure of scale, such as the standard deviation. As mentioned previously, any scaling that sets the measure of central tendency to zero will be beneficial during the initialisation of a MLP.

Mathematical function transformation applies a mathematical function to the data, for example, taking the logarithm of the data to stabilise the seasonality and variance

(Faraway and Chatfield, 1998). In statistical models, non-linear mathematical transforms, such as taking the square root or logarithm of the data, are widely used to transform the data to approximate a Gaussian distribution to minimise the effect of extreme values.

Recently, Shi (2000) proposed a new type of transformation, called distribution transformation, for transforming the inputs to an ANN model. This method transforms a stream of random data distributed on any range to uniformly distributed data points on [0,1]. Since ANNs are only useful for interpolation purposes, by transforming the input data to uniformity, a continuous and smooth mapping of the input variables to the output can be achieved. Distribution transformation requires that a distribution be fitted to each of the input variables. By using the relationship between the probability distribution function (PDF) and the cumulative distribution function (CDF), any distribution in the range can be transformed to a uniform distribution on [0,1] (Shi, 2000).

In general, the primary motivations for data transformation in ANN models are to scale the data in order to be commensurate with the transfer function in the output layer, to standardise each of the variables, to provide a suitable initialisation of the ANN and to modify the distribution of the input variables to provide a better mapping to the outputs. Most traditional statistical models also require that the data are normally distributed before the model coefficients can be estimated efficiently and, if this is not the case, suitable transformations to normality need to be found (Maier and Dandy, 2000b). Burke (1991) suggested that ANNs overcome this problem, as the probability distribution of the input data does not need to be known. However, there has been some confusion in the literature over this issue. For example, as pointed out by Fortin et al. (1997), if the mean squared error (MSE) is used as the objective function in training the ANN, this corresponds to a maximum likelihood estimation only under the hypothesis of normal (or at least symmetrical) random shocks. In linear time series models, such as the mixed autoregressive - moving average (ARMA) model of order (p, q) (Equation 3.2), it is apparent that the data must be transformed to normality if the random shock component, e, is to be normally distributed. However, in nonlinear models such as ANNs, it is not apparent that the data need to be transformed to normality if the random shock component, e, is to be normally distributed. For example, consider an ANN model with input variables x_{t-1}, x_{t-2}, ..., x_{t-p}, one hidden layer consisting of H hidden layer nodes and one output node, as given by:

$$\hat{x}_t = \phi_o\left\{\varpi_0 + \sum_{h=1}^{H} \varpi_h \phi_h\left(\beta_0 + \sum_{j=1}^{p} \beta_{jh} x_{t-j}\right)\right\} + e_t \tag{3.20}$$

where ω_0 and β_0 denote the bias weights from the constant input to the output and hidden layers respectively, ω_h denotes the weights from the hidden layer to the output layer, β_{jh} denotes the weights from the input layer to the hidden layer, and the errors e_t have a mean zero, variance σ^2 and are independent across training cases. The two functions ϕ_h and ϕ_o denote the transfer functions used at the hidden and output layers, respectively. It is apparent from Equation 3.20 that transforming the input and output variables to normality will not necessarily guarantee a random shock component that is normally distributed. To the author's knowledge, the effect of transforming the ANN's inputs and outputs to normality has not been investigated in the literature.

In the statistical literature, transformations or differencing is often used to transform a non-stationary process into a stationary process. It is not clear whether such a transformation would improve the results of an ANN model when the data are non-stationary. In their review paper, Maier and Dandy (2000b) found that the issue of stationarity is largely ignored in papers on the application of ANNs to water resources variables.

Faraway and Chatfield (1998) developed ANN models for the well-known set of airline data and considered the effect of removing the seasonal component on the ANN's forecasting ability. Two alternative approaches were considered. In the first approach, the linear trend was removed from the data and the seasonal trend was removed by subtracting the monthly averages (model 2). In the second approach, first-order and seasonal differencing were applied to the logarithms of the data (model 3). Neither model 2 nor 3 were able to improve upon the forecasting ability of the ANN model developed using the raw data (model 1). Furthermore, the use of differencing to remove the seasonality may not be desirable as it can produce a forecast variance that increases without bound as the forecasting period increases (Stedinger, 1996 *pers. comm.*).

The aim of this research is to investigate the effect of different transformations on the performance of ANN models for forecasting water resources variables. The transformations that will be investigated include:

1. Linear Transformation

This is by far the most commonly employed data transformation. The distribution of the raw data is not altered but rather, the data are rescaled to a range that is commensurate with the output layer transfer function.

2. Logarithmic Transformation

This transformation is commonly used for hydrological data that are truncated at zero and have positive skewness. Taking the logarithm of the data also converts a multiplicative seasonal relationship to additive.

3. Histogram Equalization Transformation

The method of distribution transformation (Shi, 2000) is dependent on fitting a PDF to each of the input variables at an acceptable level of significance. If the fit is poor, the transformed series will not be uniformly distributed when the CDF is used to transform the data. In many cases it is not possible to fit a distribution to the random input data at an acceptable level of significance, therefore, in this research it is proposed to use a discrete version of distribution transformation that utilises the histogram. This transformation is known as histogram equalization (Looney, 1997) and is outlined in Section 3.4.2.3.

4. Kernel Transformation

A new transformation has been developed in this research, which is similar to the histogram equalization transformation but uses univariate kernels to approximate the distribution of each input series. By numerically integrating the density estimates, it is possible to approximate the CDF, and in so doing, transform each input series to uniformity. The kernel transformation is outlined in Section 3.4.2.4.

5. Seasonal Standardisation

In this transformation, the deterministic seasonality is removed by subtracting a seasonally varying mean and dividing by a seasonally varying standard deviation. The ANN model is then developed on the seasonally standardised data.

6. Transformation to Normality

It is unclear whether transforming the input and output data to normality will improve the forecasting ability of an ANN model. Therefore, to determine the effect on the model's generalisation ability, a normalising transformation of the inputs and outputs is also investigated.

3.4.2 Methods

3.4.2.1 Linear Transformation

A linear transformation is simple and widely used, and is typically performed by using the original data range to rescale the series to a range that is commensurate with the output transfer function (Equation 3.21). It should be noted that the inputs and outputs are scaled individually.

$$X_{Ni}^{T} = \left[x_{Ni} \times \left(\frac{x_{high}^{T} - x_{low}^{T}}{x_{\max} - x_{\min}} \right) \right] + \left(\frac{x_{\max} \times x_{low}^{T} - x_{\min} \times x_{high}^{T}}{x_{\max} - x_{\min}} \right) \tag{3.21}$$

where X_{Ni}^{T} is the i-th data point of the Nth transformed data series, x_{Ni} is the i-th data point of the Nth original data series, $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of the original data range and x_{high}^{T} and x_{low}^{T} are the new maximum and minimum values for the transformed data series and are dependent on the transfer function in the output layer. For example, as the outputs for the hyperbolic tangent transfer function are between -1 and 1, the data are generally scaled in the range -0.9 to 0.9 or -0.8 to 0.8. For MLPs, the values are not usually scaled to the extreme limits of the transfer function, as the size of the weight updates may become extremely small with flatspots occurring during training (Maier and Dandy, 2000b).

3.4.2.2 Logarithmic Transformation

Logarithmic transformations are commonly used in applications of hydrological modelling and are useful for time series data that are characterised by a distribution with positive skewness. In such instances, a logarithmic transformation is commonly used to compress the distribution of the variable (Masters, 1993). A logarithmic transformation converts multiplicative relationships to additive, which is believed to simplify and improve network training (Masters, 1995b).

3.4.2.3 Histogram Equalization (Looney, 1997)

This transformation is applicable when the original data series contain spacings that are disproportionate and do not increase or decrease monotonically across the interval. Histogram equalization transformation is nonmonotonically nonlinear and approximately equalises the number of data points in each subinterval of equal length.

The first step in this procedure is to perform a linear transformation on each of the N inputs, such that they are all scaled to fall into [0,1]. For each input, the range [0,1] is then partitioned into P subintervals $\{I_1,...,I_P\}$ of equal length and the function $h(p)$ is assigned to the proportion of values in the pth interval. This process yields a histogram, defined by:

$$h(p) = \frac{M_P}{n} \tag{3.22}$$

where M_P is the number of x_{Ni} values that belong to the pth interval I_P and n is the total number of data points in the series. To then approximately equalise the number of values in each subinterval, a histogram equalization transformation H_N is performed on each of the N component values. The histogram equalization transformation is given by:

$$H_N(x) = (\frac{1}{n})\sum\{M_k : k \le p \text{ and } x \in I_P\} = \sum_{(r=1,p;x\in I_P)} h(r) \tag{3.23}$$

Each value of x between 0 and 1 is remapped into the sum of histogram values for all subintervals up to and including the one in which x lies. The resulting transformed series is uniformly distributed and will also be between 0 and 1 since:

$$\frac{M_1}{n} + ... + \frac{M_P}{n} = 1 \tag{3.24}$$

Linear transformations are then applied to the inputs and outputs, in order to scale each series to a range that is commensurate with the output layer transfer function. It is important to point out that the aim of this transformation is to allow the ANN to perform a better mapping of the inputs to the output. Consequently, only the inputs are transformed using histogram equalization. In addition, the output cannot be transformed using this method, as it is not possible to back-transform the data into their original values since the transformation discretises the data.

3.4.2.4 Kernel Transformation

In kernel transformation, rather than approximate the underlying PDF using a histogram or attempting to fit an appropriate PDF, the PDF is estimated using a kernel estimator. The main drawbacks associated with using a histogram as a nonparameteric density estimator is that the estimated probabilities are discrete in nature and the estimates are very sensitive to the location and the width of the individual bins (Sharma et al., 1998).

Estimating the densities using a kernel estimator is an extension of the histogram but has the advantage that it provides a smooth, continuous probability estimate. A univariate kernel estimator with kernel K is defined by:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{3.25}$$

where $K(\cdot)$ is a kernel function centred at each data point x_i, and h is the bandwidth (also called the window width or smoothing parameter by some authors), which specifies the spread of individual kernel functions. In order for the probability density estimate to integrate to one, the kernel function must be a legitimate PDF. The most widely used is the Gaussian kernel function, and since this is a practical choice, it has also been used in this study:

$$K(x) = \frac{1}{(2\pi)^{1/2}} \exp(-x^2/2) \tag{3.26}$$

Numerous methods are in existence for selecting an appropriate bandwidth h (see Scott, 1992; Sharma et al., 1998; Silverman, 1986). In this research, a widely used guideline known as Silverman's rule-of-thumb (Silverman, 1986) was used to select an appropriate bandwidth for each input series. The bandwidth was computed as follows:

$$h = 0.9 \cdot \min\left[\sigma, \left(\frac{Q_3 - Q_1}{1.34}\right)\right] \cdot n^{-1/5} \tag{3.27}$$

where σ is the standard deviation of the input series, Q_3 and Q_1 are the third and first quartiles, respectively, and, n is the number of data points.

In kernel transformation, each input series is taken in turn, and its probability density estimates are calculated. Once the probability density estimates have been computed, the resulting series is then sorted such that the corresponding values in the original time series are in ascending order. The probability density estimates are then integrated numerically to approximate the CDF $\hat{F}(x_n)$ for the n^{th} data point in the series. In this research, the trapezoidal rule was used to perform the required integration:

$$\hat{F}(x_n) = \sum_{i=1}^{n} \left(g(x_{i+1}) - g(x_i)\right) \cdot \left(\frac{\hat{f}(x_{i+1}) + \hat{f}(x_i)}{2}\right) \tag{3.28}$$

where $g(x)$ is the original (raw) series and $\hat{f}(x)$ are the corresponding probability density estimates. Once the integration has occurred, the series is then sorted back to its original ordering. The resulting transformed series is an approximation of the CDF and is uniformly distributed on the interval [0,1]. The data are then linearly transformed to a range that is commensurate with the output transfer function. Like histogram equalization, the aim of this transformation is to allow the ANN to perform a better mapping of the inputs to the output. Consequently, only the inputs are transformed.

3.4.2.5 Seasonal Standardisation

When modelling real-world hydrologic processes, it is common to treat the non-stationarity resulting from the seasonality in the data as a deterministic rather than a statistical phenomenon. Therefore, transformation rather than differencing is to be used to produce a stationary time series in this research.

The first step in removing the seasonal component of the data is to fit a separate Fourier series to the mean and standard deviation of each time series. Using the seasonally varying mean and standard deviation, each time series can then be standardised by using the following equation:

$$X^T(t) = \frac{x(t) - \mu_s(t)}{\sigma_s(t)} \tag{3.29}$$

where $X^T(t)$ is the deseasonalised transformed variable; $x(t)$ is the raw data at time t; $\mu_s(t)$ is the function for the seasonally varying mean and $\sigma_s(t)$ is the function for the seasonally varying standard deviation. Both $\mu_s(t)$ and $\sigma_s(t)$ are fitted by functions of the form:

$$f(t) = a_0 + \sum_{j=1}^{M} a_j \sin(jw_0 t + \phi_j) \tag{3.30}$$

where the coefficients a_0, a_j, w_0 and ϕ_j are found using the MS Excel Solver add-in and the number of sine curves M needed to accurately approximate $f(t)$ is found by adding sine curves until a significant percentage (>99 percent) of the variance is captured by the fitted function. The resulting deseasonalised data set can then be used to develop ANN models. To convert the transformed output from the ANN model back into real-world values, the inverse of Equation 3.30 is used.

3.4.2.6 Transformation to Normality

Inverse transformation is commonly used in Monte Carlo Simulation to generate random distributions from the uniform distribution on [0,1]. In this research, a new two-step transformation to normality is proposed which combines the histogram equalization transformation with the inverse transformation. The first step is to use histogram equalization to transform each of the inputs and outputs to uniformity on [0,1]. The second step is to then use a functional approximation of the inverse transformation to transform each input and output series to normality. In this study, the functional approximation of the inverse transformation proposed by Beasley and Springer (1977) was used. Finally, a linear transformation is applied to ensure that the data are scaled to a range suitable for the transfer function in the output layer.

This two-step numerical procedure has the disadvantage that the data are discretised in the histogram equalization step and therefore, the ANN model's output cannot be transformed back to the exact real-world values. However, by using the raw and transformed values in the training set, it is possible to back-transform the ANN model's output via linear interpolation. An analytical transformation to normality would have been better if a successful one could have been found, but they were all found to be unsuccessful for the data sets under consideration in this research.

3.5 Determination of ANN Model Inputs

3.5.1 Introduction

One of the most important steps in the ANN development process is the determination of significant input variables. Usually, not all of the input variables collected will be equally informative since some may be correlated, noisy or have no significant relationship with the output variable being modelled. Despite this, in most ANN applications, very little attention is given to the important task of selecting appropriate model inputs (Maier and Dandy, 2000b). This is primarily because ANNs belong to the class of data driven approaches, whereas conventional statistical methods are model driven (Chakraborty et al., 1992). In the latter, the model's structure is determined *a priori* by using empirical or analytical approaches, before estimating the unknown model parameters, whereas data driven approaches are usually assumed to be able to determine which model inputs are critical. This has meant that ANN practitioners often present a large number of inputs to the ANN and rely on the network to determine the critical model inputs. There are a number of shortcomings associated with this approach, and these include the following (Back and Trappenberg, 1999; Maier and Dandy, 1997a; Zheng and Billings, 1996):

- As input dimensionality increases, the computational complexity and memory requirements of the model increase.
- Learning becomes more difficult with irrelevant inputs.
- Misconvergence and poor model accuracy may result from additional unrequired inputs due to an increase in the number of local minima present in the error surface.
- Understanding complex models is more difficult than simple models, which give comparable results.
- Due to the curse of dimensionality, some types of ANN models with many irrelevant inputs behave poorly since the network uses almost all its resources to represent irrelevant portions of the input-output mapping.
- Other types of networks that can efficiently concentrate on important regions of the input space require more data to efficiently estimate the connection weights when irrelevant inputs are included.

Consequently, there are obvious advantages in using analytical procedures to select an appropriate set of inputs for ANN models. The challenge of input determination is to select a subset of inputs from all potential inputs that will lead to a superior model as measured by some optimality criterion. For d potential inputs, there are 2^d-1 input subsets, hence, it is possible to test all subset combinations for small values of d, but

for large values of d, as is often the case for complex problems, efficient algorithms are required. The problem is further exacerbated in time series studies, where appropriate lags must also be chosen. The difficulty lies in determining how many lagged values to include from each input time series. In general, if there are N input time series ($x_{j,t-1}$, $x_{j,t-2}$, ..., $x_{j,t-n}$, j=1, 2, ..., N), then the problem is finding the maximum lag for each input time series (k_j: $k_{j,max} < n$, j=1,2, ..., N) beyond which, values of the input time series have no significant effect on the output time series. The maximum lag is also called the memory length. As the length of memory used increases, so too the number of inputs and the complexity of the ANN model increases.

Input determination can be divided into two broad stages. Firstly, unsupervised input preprocessing (i.e. discarding redundant inputs) and secondly, supervised input selection (i.e. using the output variable in an analytical procedure to determine the significant inputs). In unsupervised input preprocessing, the original set of inputs is processed to produce a subset of inputs containing as much information as possible from the original set. This subset of inputs can then be used in a supervised input selection process to determine which of these inputs are related in some way to the output. Unsupervised input preprocessing methods are all *model-free*, meaning that they do not depend in any way on a pre-existing model, whereas supervised input selection methods can be either *model-free* or *model-based*. In *model-based* methods, the input selection problem can be formulated as having a set of input variables to a model, and an output value which can be used to evaluate the fitness or merit of the model using those variables. For example, for the problem of forecasting cyanobacteria, the inputs to a model may be causal variables such as turbulence (flow), water temperature, turbidity, colour and nitrogen and phosphorus concentrations. From these variables, a subset of inputs must be selected that yield the model of highest fitness. This subset of inputs forms a N-dimensional input vector $\boldsymbol{X}^N$, which can be used to forecast the concentration of cyanobacteria Y. The aim of the model is to produce a relationship of the form:

$$Y = f(\boldsymbol{X}^N) + e \tag{3.31}$$

where e is the model error term to be minimised. The model's fitness can be determined using an appropriate measure of the model error e.g. smallest root mean square prediction error (RMSE). Therefore, by using an efficient search procedure, the most suitable input subset $\boldsymbol{X}^n$ can be identified as the model with the smallest RMSE.

In *model-free* approaches, instead of using a model to characterize the relationship between input and output, some statistical measure of dependence is used (e.g.

correlation). If the dependence between input and output is found to be significant, then the input is retained in the final subset, otherwise it is discarded as it is unlikely to improve predictive ability when used in a model.

3.5.1.1 Review of Input Determination in Water Resources ANN Applications

Maier and Dandy (2000b) reviewed 43 papers (published up until the end of 1998) on the application of ANNs to modelling water resources variables. They found that in many cases, the lack of a methodology for determining the input variables raised doubt about the optimality of the inputs obtained. In some instances, inputs were chosen arbitrarily. In other cases, *a priori* knowledge was used and when different methods were employed, such as trial-and-error, often the validation data were used as part of the training process. More recently, the importance of input determination has begun to be recognised in the water resources literature, however, there are still only a small number of papers that treat it as an important step in the modelling process. The main approaches that have been employed in the water resources ANN modelling literature can be broadly classified into five methods. These include:

1. Methods that rely upon the use of *a priori* knowledge of the system being modelled: Usually some degree of *a priori* knowledge is used to specify the initial set of candidate inputs. This has been acknowledged by the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a), who pointed out that in hydrological applications of ANNs, a good level of understanding of the hydrologic system being modelled is very important in order to select appropriate model inputs. If important candidate inputs are not included, then some information about the system may be lost and, on the other hand, if spurious inputs are included, it may tend to confuse the training process. If the relationship to be modelled is not well understood, then an analytical technique, such as cross-correlation, is often employed (e.g. Golob et al., 1998; Huang and Foo, 2002; Luk et al., 2000; Sajikumar and Thandaveswara, 1999; Silverman and Dracup, 2000). Many of the papers reviewed relied solely on the use of *a priori* knowledge to select the appropriate model inputs (e.g. Campolo et al., 1999; Jayawardena et al., 1997; Thirumalaiah and Deo, 2000). Although *a priori* identification is widely used in many ANN applications and is necessary to define a candidate set of inputs, it is dependent on an expert's knowledge, and hence, is very subjective and case dependent. Intuitively, the preferred approach for determining appropriate inputs and lags of inputs, involves a combination of *a priori* knowledge and analytical approaches (Fernando and Jayawardena, 1998; Maier and Dandy, 1997a; Maier et al., 1998).

2. Methods based on linear cross-correlation:

Cross-correlation methods represent the most popular analytical techniques employed to select appropriate inputs. For example, Imrie et al. (2000) developed flow forecasting ANNs for two catchments in the UK and used a cross-correlation analysis to identify suitable lags of time series from upstream gauging stations as inputs to the ANN model. Sajikumar and Thandaveswara (1999) used the cross-correlogram of rainfall and runoff to determine the linear component of the memory length of rainfall required by their flow forecasting model. Maier and Dandy (1997a) used the method of Haugh and Box (1977), which uses cross-correlation analysis to determine the strength of the relationship between the input time series and the output time series at various lags. Values of the cross-correlation function were deemed to be significant if they satisfied the following equation

$$\left| r_{z_{i,t} z_{j,t}} \right| \geq 2 \frac{1}{\sqrt{n}} \tag{3.32}$$

where $r_{z_{i,t} z_{j,t}}$ is the sample cross-correlation between time series $z_{i,t}$ and $z_{j,t}$ and n is the number of data points. Other papers reporting the use of cross-correlation to select ANN inputs included: Huang and Foo (2002); Golob et al. (1998); Brion et al. (2001); Lekkas et al. (2001); and, Coulibaly et al. (2000a). The major disadvantage associated with using cross-correlation is that it is only able to detect linear dependence between two variables. Therefore, cross-correlation is unable to capture any nonlinear dependence that may exist between the inputs and the output, and may possibly result in the omission of important inputs that are related to the output in a nonlinear fashion.

3. Methods that utilise a heuristic approach:

Heuristic methods are also commonly used in the literature. In this approach, various ANN models are trained using different subsets of inputs. For example, Jain et al. (1999) developed a univariate ANN for reservoir inflow prediction. To select a suitable number of lags, ANNs consisting of different numbers of lags were trialled until the best model was identified. In a similar approach, Jain and Chalisgaonkar (2000) tried different combinations of stage and discharge inputs in order to model river-rating curves.

Another commonly used heuristic approach is stepwise selection of inputs. This method is often employed to avoid total enumeration i.e. the consideration of all subsets. The two standard stepwise approaches are forward selection and backward selection. Forward selection is the more commonly used approach and begins by finding the best single input and selecting it for the final model. In general, given a set

of selected inputs, the input that improves the model's performance most is added to the final model. This process is then repeated. Backward elimination starts with a set of all inputs, and sequentially deletes the input that reduces performance the least. Maier and Dandy (1998) used a forward stepwise selection method to determine the important input variables for forecasting the concentration of the cyanobacteria *Anabaena* spp. in the River Murray, Australia. In their approach, N-bivariate models were developed, where N is the number of possible input variables and each model relates one input to the output. The input variable from the bivariate model that gave the smallest modelling error was then selected as a significant input and included in the model. N-1 models were then developed by combining this variable with each of the remaining variables. This procedure was then repeated for three, four, five, etc. input variables until the addition of any extra variable did not significantly improve model performance. Tokar and Johnson (1999) developed ANN models to forecast daily runoff as a function of daily precipitation, temperature, and snowmelt for the Little Patuxent River watershed in Maryland, USA. Initially a simple model was selected by representing streamflow at the present time t as a function of precipitation at time t. Precipitation at time t-1 was then added to the model. If the goodness-of-fit statistics were significantly better, the precipitation at time t-2 was also added to the model. The procedure was repeated by adding precipitation at previous time periods as inputs until there was no change in the goodness-of-fit statistics. Next, another input variable such as temperature, snowmelt equivalent, evaporation or streamflow at previous time periods was added to the best-fit model obtained from the previous step. The procedure was repeated for each variable and halted when all available input variables were exhausted.

The main disadvantage of these approaches is that they are based on trial-and-error, and as such, there is no guarantee that they will find the globally best subset of inputs. Another disadvantage of the stepwise approaches is that they are computationally intensive.

4. Methods that extract knowledge contained within the trained ANN:
Sensitivity analyses (Section 2.5.3) are the most commonly used method of extracting information from a trained ANN (e.g. Liong et al., 2000b; Maier and Dandy, 1996b; Maier and Dandy, 1997a; Maier et al., 1998; Schleiter et al., 1999). Plots of the sensitivities for each of the inputs are usually inspected and the significant inputs are chosen using judgement. However, the difficulty with this approach is choosing a reasonable value to perturb the input by and selecting the appropriate cut-off point for input significance.

Abrahart et al. (2001) used a different approach known as saliency analysis to disaggregate a neural network solution in terms of its forecasting inputs. In this approach, the effect that each input has on the error function is examined by removing one input at a time from the trained ANN model. The ANN model investigated was a one-step-ahead flow forecasting model for the Upper River Wye in Central Wales. The saliency analysis was achieved by zeroing one input data stream at a time, performing the forecasting using the trained ANN, replacing the input data stream after the computation and repeating the process on the next input and so on. In this way, the relative importance of each input was determined by looking at the change in forecasting error and by examining plots of the flood hydrograph. Abrahart et al. (2001) stated that this approach differs to sensitivity analysis in that it does not investigate the rate of change of one factor with respect to the change in another, but rather, disaggreagates the solution in a piecemeal fashion to determine the most influential input variable and the relevant importance of each input. However, the main disadvantage of their approach is that they did not retrain the ANN after removing each input. It is possible that clamping the input to zero may produce nonsensical outputs if zero is not a reasonable value for that input in the input-output pattern under investigation. As pointed out by Sarle (1997), this is particularly the case if the inputs are statistically dependent, because the effects of different inputs cannot generally be separated.

5. Methods that use various combinations of the above four approaches:
Many papers report the combined use of some of the above approaches. For example, Schleiter et al. (1999) used a number of input determination methods including Pearson correlation, stepwise forward regression analysis and sensitivity analysis to select appropriate inputs for the modelling of water quality, bioindication and population dynamics in lotic streams. Silverman and Dracup (2000) used *a priori* knowledge of the system under investigation and combined this with a trial-and-error approach to determine a suitable set of inputs for long-range forecasts of precipitation in California using ANNs.

Some alternative methods for ANN input determination that could not be classified into the above five categories were also used in the water resources literature. For example, Braddock et al. (1998) used the input identification methodology introduced by Lachtermacher and Fuller (1994) and extended this methodology to the multivariate case. In this method, a transfer function, as described by Box and Jenkins (1970), is fitted to the data to determine if a Box-Cox transformation of the data is required and to determine the number of lagged dependent and independent variables which should be used. However, this has the major limitation of only capturing the inputs that are

linearly correlated with the output. In another approach, Abrahart et al. (1999) used a genetic algorithm to optimise the inputs to an ANN model used to forecast runoff from a small catchment.

In the present study, as a first step in preprocessing the input variables, unsupervised techniques (Section 3.5.2.1) have been used to reduce the dimensionality of the input space. These techniques are principal component analysis (PCA) and the self-organizing map (SOM). PCA was selected because it is a commonly used procedure in the statistics literature for reducing the dimensionality of input subsets, however, it does have the limitation of only considering linear dependence between the variables. Therefore, the SOM input reduction technique was devised in this research, since it is capable of dealing with both linear and nonlinear dependence between variables. Once redundant inputs have been removed, the input subsets obtained from each unsupervised method are further refined using a supervised input selection method (Section 3.5.2.2). Both a *model-based* and a *model-free* supervised input determination technique were considered. The *model-based* method used in this research is a hybrid genetic algorithm and general regression neural network (GAGRNN). The *model-free* method that was investigated is a stepwise procedure based on the concept of partial mutual information (PMI). This method does not require the variables to be preprocessed using an unsupervised technique because it is capable of dealing with redundant input variables and can account for both linear and nonlinear dependence between variables. The input determination techniques used in this research are summarised in Figure 3.2.

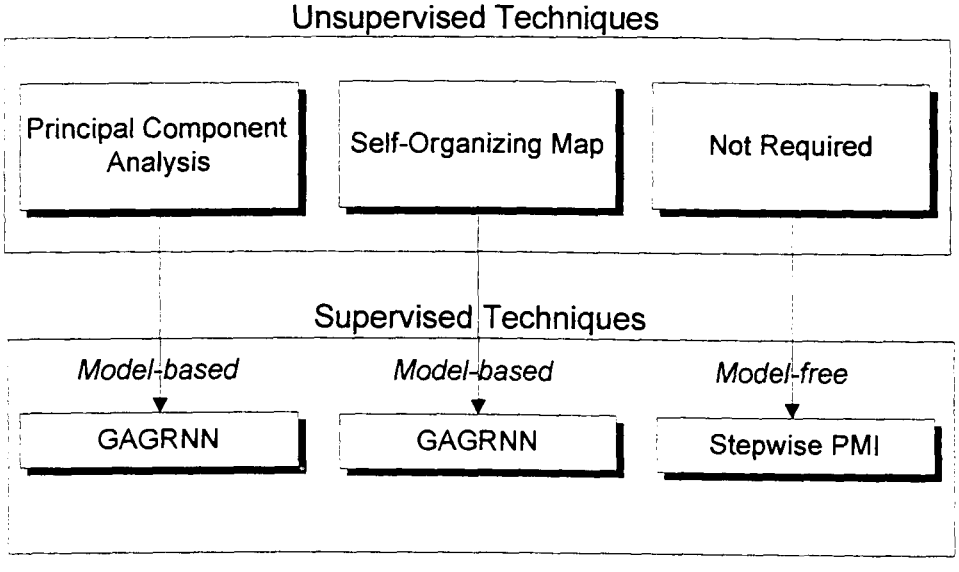

Figure 3.2 Input Determination Techniques

3.5.2 Methods

3.5.2.1 Unsupervised Input Preprocessing

Principal Component Analysis

When many potential variables are available, PCA can be used to reduce the dimensionality of the input data set. By using principal components (PCs), the variables can be transformed to a new, smaller set of variables, which capture most of the information in the original data set. This is achieved by computing factors (new variables) as linear combinations of the old variables. The weights are selected in a way that ensures that some optimality criterion is maximised (Masters, 1995b).

To commence the procedure, a single linear combination of all variables is sought such that the majority of the variation in the training set is captured. After a single dominant factor is found, it is then necessary to find a second factor that captures the remaining information not explained by the first factor. The second PC is chosen such that it is orthogonal to the first. Next, a third factor is sought that best captures the remaining information and is orthogonal to the first and second. The process continues until all the variance in the data set is accounted for. If there are p input variables of interest, it is hoped that m, where $m<<p$, different PCs will account for most of the variation. As the interrelations among the variables increases, the proportion of variance explained by the first few components also increases. Hence, it is common for most of the important information to be concentrated in the first few principal components, with the system noise falling mostly in later components that can be discarded (Masters, 1995b). However, it is possible, but usually unlikely, that by discarding the later components, some important information may be lost (Jolliffe, 1986; Masters, 1995b). An added advantage of PCA is that each of the computed factors are independent of each other and there is no redundancy in the information that they contain (Masters, 1995b).

If the input variables have different units, it is necessary to standardise the data. PCA attempts to capture variation. Numerically, a variation in flow between 10,000 and 20,000 ML/day is much greater than a variation in river level between 1.0 and 5.0 m. However, the effect of each of these variables on the system being investigated may be rather similar and the information content of the flow data is not inherently greater. Hence, by standardising the data, all variables are put on an equal basis in the analysis.

An important consideration in PCA is the determination of m, i.e. the number of PCs to retain. Various rules have been devised for determining a suitable value of m and many

of these are discussed in Jolliffe (1986). The rule selected for this research is one that has been used successfully in the meteorological literature by Jones et al. (1983), but was originally devised by Guiot (1981). In this method, PCs are retained if their cumulative eigenvalue product exceeds one. This criterion tends to retain more PCs than many of the other methods commonly used, however, Jones et al. (1983) found the results it produced to be satisfactory.

In this research, PCA is conducted on the lags of each variable, in order to find a subset of PCs. Next, the PCs that have been identified for each variable are combined into one data set. PCA is then conducted on this data set to reduce the total number of PCs.

Self-Organizing Map

The SOM (Section 2.4.2.1) is a powerful technique capable of ordering multivariate data by similarity, while preserving the topological structure of the data. In this research, the SOM is used to cluster the input variables into groups of similar inputs. By then sampling one input from each cluster, it is possible to remove highly correlated, redundant variables from the original data set. The SOM is implemented using the *NCS NeuFrame* software. To cluster the data, the input variables are presented to the network as the SOM's inputs. As is the case with PCA, the data are standardised to put each variable on equal basis. The output of the SOM is obtained using a Dynamic Patterns grid, which shows a dynamic representation of the nodes that are winning each pattern. Each individual cell in the grid represents a node in the Kohonen layer. There is no theoretical principle for determining the optimum size of the Kohonen layer (Cai et al., 1994), hence, the Kohonen layer should be kept large enough to ensure that the maximum number of clusters are formed from the training data. Once the clusters are formed, one input from each cluster is sampled and used in the final subset of input variables.

The SOM input procedure is conducted for each variable, in order to find a subset of lags with limited cross-dependence. Next, the lags that have been identified for each variable are combined into one data set. The SOM input determination procedure is then conducted again, this time to ensure that there is limited cross-dependence between the variables selected.

3.5.2.2 Supervised Input Determination

Substantial variation amongst the input variables does not necessarily imply any relationship with the output variable. Therefore, after the input dimensionality has been reduced using the unsupervised procedures (i.e. PCA and SOM), supervised procedures must ultimately decide upon which variables have the most significant relationship with the output.

Hybrid Genetic Algorithm and General Regression Neural Network

The *model-based* supervised input determination technique considered in this research is a hybrid GA and GRNN (GAGRNN). The advantages of using a GRNN include:

- Nonlinear relationships between the inputs and output can be modelled.
- The network architecture is fixed and does not need to be determined.
- Training times are much quicker than other ANNs, such as MLPs trained using backpropagation.

The speed at which a GRNN can be trained is a very important consideration because typically, many thousands of networks, each consisting of different subsets of inputs, need to be trained in the GA optimisation process. A detailed overview of GAs is provided in Goldberg (1989).

GAs are well suited to the task of selecting an appropriate combination of inputs to a model as they have the ability to search through large numbers of combinations, where interdependencies between variables may exist. For the problem of determining inputs to a GRNN, a population of randomly selected GRNNs is generated, each with a different subset of input variables as depicted by a binary string. Hence, for N variables the string will have length N. If the jth value of the string is equal to one, then the jth input is included, otherwise, if it equals zero, it is not included. The models are trained and the output of each GRNN is used to determine the predictive error, or fitness of the solution. In this research, the RMSE between the actual and predicted values is used to determine the fitness of the model. Based on the fitness of each member in the population, a selection scheme can then be employed to create a new population for the next generation. In this way, fit solutions are allowed to live and subsequently breed in the process of crossover. In the crossover process, two parent strings are cut and part of the strings exchanged to produce two new individuals. To maintain genetic diversity and ensure that no important genetic

material is overlooked, a mutation operation introduces small random changes. The process is continued in an iterative manner until the error from the GRNN model converges to an acceptable value or a prespecified maximum number of generations is completed. Due to the selective pressure applied over the generations, the overall trend is the evolution of higher fitness chromosomes representing optimal input subsets. The commercially available software package, *NeuroGenetic Optimizer* (*NGO*) (BioComp Systems Inc., 1998) was used in this research to implement the GAGRNN.

Stepwise Partial Mutual Information Input Determination Algorithm

Nonlinear relationships are commonly encountered in water resources modelling and cannot be suitably detected or quantified by using methods based on the linear correlation between two variables (Harrold et al., 2001). The use of such methods in hydrological modelling can lead to fitted models that do not adequately represent the system they are attempting to model. Recently, Sharma (2000) proposed an input determination method based on the partial mutual information (PMI) criterion to overcome the limitation of the correlation coefficient in selecting appropriate model inputs. The PMI criterion is able to adequately capture all dependence (linear and nonlinear) between two variables and avoids the need for making any major assumptions about the underlying model structure and as such, can be considered a *model-free* approach. The PMI criterion is an extension of the concept of mutual information.

Mutual information measures the dependence between two random variables (Yang et al., 2000). The mutual information function between variables X and Y is given by:

$$MI = \int\int f_{X,Y}(x,y)\log_e\left[\frac{f_{X,Y}(x,y)}{f_X(x)f_Y(y)}\right]\mathrm{d}x\mathrm{d}y \tag{3.33}$$

where $f_X(x)$ and $f_Y(y)$ are the marginal PDFs of X and Y, respectively, and $f_{X,Y}(x,y)$ is the joint (bivariate) PDF of X and Y. The mutual information measures the reduction in uncertainty of Y due to knowledge of the variable X (Cover and Thomas, 1991). If there is no dependence between X and Y, then the two random variables are statistically independent and by definition the joint probability density $f_{X,Y}(x,y)$ would equal the product of the marginal densities ($f_X(x)f_Y(y)$). In this case, the MI score in Equation 3.33 would equal a value of 0, since the ratio of the joint and marginal densities would equal unity and hence, the logarithm would equal 0. Conversely, if the random variables were strongly related, the MI score would take on a high value.

Given any bivariate sample, the discrete version of Equation 3.33 can be used to estimate the MI score, and is given by:

$$MI = \frac{1}{n}\sum_{i=1}^{n} \log_e \left[\frac{P_{X,Y}(x_i, y_i)}{P_X(x_i) P_Y(y_i)} \right] \tag{3.34}$$

where x_i and y_i are the i-th bivariate sample pair in a sample of size n, and $P_X(x_i)$, $P_Y(y_i)$ and $P_{X,Y}(x_i,y_i)$ are the respective univariate and joint probability densities estimated at the sample data points. Using Equation 3.34, the three probability distributions $P_X(x)$, $P_Y(y)$ and $P_{X,Y}(x,y)$ are used to describe, in probabilistic terms, the relationship between an input variable X and an output Y. However, the key to successful computation of the MI score in Equation 3.34 is the accurate estimation of these probability distributions from a finite set of examples. The original MI algorithms utilised crude measures to approximate the probability densities, such as a histogram with fixed bin width (e.g. Fraser and Swinney, 1986; Zheng and Billings, 1996). More recently, kernel density estimation techniques have been used because of their stability, efficiency and robustness (Sharma, 2000). In the study conducted by Sharma (2000), the Gaussian kernel function (Scott, 1992) was used and is given by:

$$\hat{f}_X(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi)^{d/2} \lambda^d \det(\boldsymbol{S})^{1/2}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T \boldsymbol{S}^{-1} (\boldsymbol{x}-\boldsymbol{x}_i)}{2\lambda^2} \right) \tag{3.35}$$

where $\hat{f}_X(\boldsymbol{x})$ is the multivariate kernel density estimate of the d-dimensional variable set $\boldsymbol{X}$ at coordinate location $\boldsymbol{x}$; $\boldsymbol{x}_i$ the i-th multivariate data point, for a sample of size n; $\boldsymbol{S}$ is the sample covariance matrix of the variable set $\boldsymbol{X}$; det(·) represents the determinant operation; and λ is the smoothing parameter, known as the "bandwidth" of the kernel density estimate. In order to produce an accurate density estimate, λ must be chosen very carefully. Small values of λ tend to lead to density estimates that give too much emphasis to individual points. In this case, spurious fine structure may become present with bumps in the tails of the probability density. Larger values of λ tend to over-smooth the probability density with all detail, spurious or otherwise, becoming obscured. Only for intermediate values of λ will a good density estimate be obtained. Various rules-of-thumb are available in the literature to help choose an optimal value of the bandwidth λ. Sharma (2000) used the Gaussian reference bandwidth (Scott, 1992) because it is relatively simple and computationally efficient:

$$\lambda_{ref} = \left(\frac{4}{d+2} \right)^{1/(d+4)} n^{(-1/(d+4))} \tag{3.36}$$

where n and d are the sample size and dimension of the multivariate variable set, respectively.

The MI criterion can be used to identify appropriate model inputs but is not directly able to deal with the issue of redundant inputs. For example, if an input x has a strong dependence relationship with the output variable y as indicated by a high MI score, then a second input $z = 2x$ will also have a high MI score. Simply using the MI score, the new variable z would be considered as a significant input, however, z is a redundant input because it is already described by the pre-existing input x. To avoid this problem, Sharma (2000) introduced the concept of partial mutual information (PMI), which provides a measure of the partial or additional dependence the new input can add to the existing prediction model. The PMI between the output (dependent variable) y and the input (independent variable) x, for a set of pre-existing inputs z, can be given by:

$$PMI = \int\int f_{X',Y'}(x',y') \log_e \left[\frac{f_{X',Y'}(x',y')}{f_{X'}(x') f_{Y'}(y')} \right] dx'dy' \tag{3.37}$$

where

$$x' = x - E[x|z] \qquad y' = y - E[y|z] \tag{3.38}$$

where $E[\cdot]$ denotes the expectation operation. By using the conditional expectations in Equation 3.37, the variables x' and y' only contain the residual information after the effect of the pre-existing set of inputs z has been taken into consideration.

The discrete version of Equation 3.37 can be used to provide a sample estimate of the PMI score, and is given by:

$$PMI = \frac{1}{n} \sum_{i=1}^{n} \log_e \left[\frac{P_{X',Y'}(x'_i, y'_i)}{P_{X'}(x'_i) P_{Y'}(y'_i)} \right] \tag{3.39}$$

where x'_i and y'_i are the i-th residuals in a sample of size n, and $P_{X'}(x'_i), P_{Y'}(y'_i)$ and $P_{X',Y'}(x'_i, y'_i)$ are the respective marginal and joint probability densities.

To estimate the conditional expectation $E[x|z]$, Sharma (2000) uses the kernel density estimator given in Equation 3.35 and the fact that the conditional probability density $f_{x|z}(x|z)$ can be estimated as the ratio $f_{x,z}(x,z)/f_z(z)$, where the denominator refers to the marginal probability density of the pre-existing variable set z. By then estimating these

probability densities as the summation of n kernel functions, each conditional to z, the expectation of this conditional PDF becomes the summation of the expectation of each of these conditional kernels, as given by:

$$E[x|z] = \frac{1}{N}\sum_{i=1}^{N} w_i (x_i + (z - z_i)^T \boldsymbol{S}_{zz}^{-1} \boldsymbol{S}_{xz} \tag{3.40}$$

where $\boldsymbol{S}_{xz}$ is the sample cross-covariance between x and z, and $\boldsymbol{S}_{zz}$ is the sample covariance of z.

$$w_i = \frac{\exp\left(-\frac{(z - z_i)^T \boldsymbol{S}_{zz}^{-1} (z - z_i)}{2\lambda^2}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{(z - z_j)^T \boldsymbol{S}_{zz}^{-1} (z - z_j)}{2\lambda^2}\right)} \tag{3.41}$$

The final component of the stepwise PMI predictor selection approach proposed by Sharma (2000) is the calculation of a suitable confidence measure to indicate whether the selected input is a significant predictor of the system being modelled. The significance measure used is the upper 95th percentile PMI score for the case where independence is forced between the input under consideration and the output. To force independence, 100 randomly shuffled sequences (surrogates) of the input are created by rearranging or bootstrapping the input variable. By construction, these surrogates satisfy the null hypothesis of independence between the input and output and can be used to calculate the randomised sample PMI scores. The decision rule for selecting an input is that the sample PMI score for that input be greater than the estimated 95th percentile randomised sample PMI score. In other words, the null hypothesis of independence must be rejected for the input to be considered significant, and if this is the case, it is reasonable to expect that there is less than 5% chance that the two variables are truly independent.

The complete stepwise predictor selection algorithm outlined in Sharma (2000) proceeds as follows:

1. Identify the set of variables that could be useful predictors of the system being modelled. Denote this variable set as z_{in}. Denote the vector that will store the final predictors of the system as z. This is a null vector at the start of the algorithm.
2. Estimate the PMI between the dependent variable y and each of the plausible new predictors in z_{in}, conditional to the pre-existing predictor set z.
3. Identify the variable in z_{in} having the highest PMI score in step 2.

4. Estimate the 95th percentile randomised sample PMI score for the variable identified in step 3.
5. If the PMI score for the identified variable is higher than the 95th percentile randomised sample PMI score of step 4, include the variable in the predictor set z, and remove it from z_{in}. If the dependence is not significant, go to step 7.
6. Repeat steps 2-5 as many times as are needed.
7. This step will be reached only when all the predictors have been identified.

Improvements to the original algorithm:

In this research, the stepwise predictor selection algorithm proposed by Sharma (2000) has been modified to incorporate the following two improvements:

1.) The Gaussian kernel function (Equation 3.35) used in the kernel density estimator has been replaced by the city block distance kernel as given by:

$$\begin{aligned}\hat{f}_X(\boldsymbol{x}) &= \frac{1}{n(2\lambda)^d}\sum_{i=1}^{n}\prod_{j=1}^{d} e^{-|x_j - x_{ij}|/\lambda} \\ &= \frac{1}{n(2\lambda)^d}\sum_{i=1}^{n}\exp\left[-\frac{1}{\lambda}\sum_{j=1}^{d}|\boldsymbol{x}_j - \boldsymbol{x}_{ij}|\right]\end{aligned} \tag{3.42}$$

where $\hat{f}_X(\boldsymbol{x})$ is the multivariate kernel density estimate of the d-dimensional variable set $\boldsymbol{X}$ at coordinate location $\boldsymbol{x}$, $\boldsymbol{x}_i$ the i-th multivariate data point, for a sample of size n, and λ is the smoothing parameter, known as the "bandwidth" of the kernel density estimate. The univariate version of this kernel was first proposed by Parzen (1962). However, theorem 4.1 in Cacoullous (1966) shows that Parzen's results can be extended to the multivariate case when the multivariate kernel is a product of a number of univariate kernels as is the case in Equation 3.42. The main advantage of Equation 3.42 is its computational simplicity and consequent reduction in the amount of processing time required. To verify that the city block distance kernel was able to approximate a joint probability density to a suitable level of accuracy, it was tested on a data set consisting of a pair of 500 random deviates from a standard bivariate normal distribution. A bivariate probability density function $f(x,y)$, written for two normally distributed parameters x and y, is given by:

$$f(x,y) = \frac{e^{-Q(1-\rho^2)/2}}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \tag{3.43}$$

where the variable Q is equal to:

$$Q = \frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} \tag{3.44}$$

and σ_x is the standard deviation in parameter x, σ_y is the standard deviation in parameter y, and ρ is the correlation coefficient defined as:

$$\rho = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \tag{3.45}$$

where σ_{xy} is the covariance between parameters x and y. Using the 500 values of x and y, the actual bivariate probability density $f(x,y)$ was calculated using Equation 3.43. Two types of plots of the probability density function are useful, a 3-dimensional plot and a contour plot (Figure 3.3).

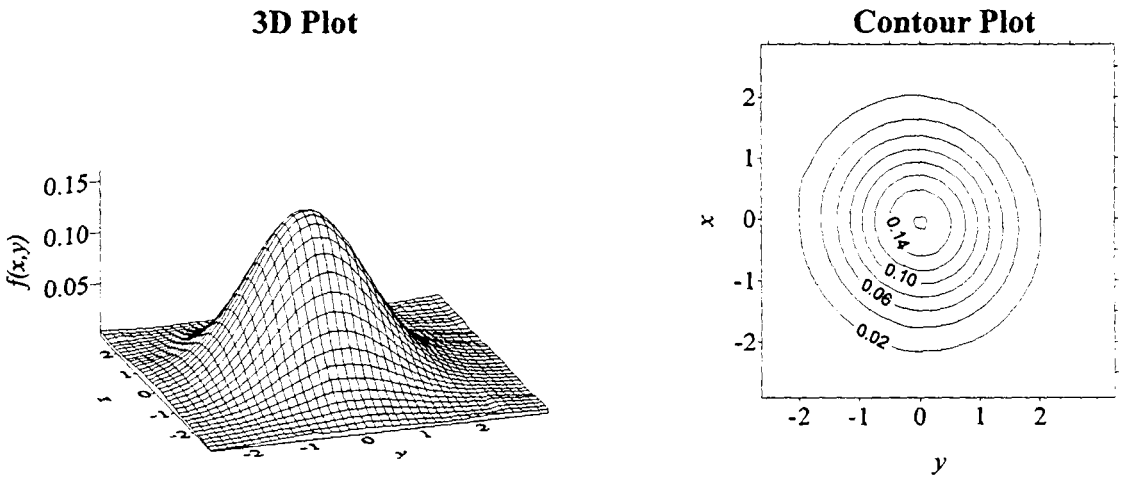

Figure 3.3 Actual Bivariate Probability Density Between x and y

As is expected, Figure 3.3 shows there is no correlation between the variables x and y as indicated by the perfectly concentric rings on the bivariate probability density contour plot. This data set formed the reference set with which to compare the estimated bivariate probability density $\hat{f}(x,y)$ from the Gaussian and city block distance kernels. For both kernels, the bandwidth was estimated using the Gaussian reference bandwidth (Equation 3.36). Figure 3.4 shows the estimated bivariate probability density produced by the Gaussian kernel. From the 3-dimensional plot it can be seen that the overall density is reasonably well approximated. The contour plot shows that there is very little correlation between x and y, however the rings are not perfectly concentric because it is only an estimate of the true probability density based on a sample of 500 values. The correlation coefficient between the actual bivariate

normal probability density $f(x,y)$ and the estimated bivariate normal probability density $\hat{f}(x,y)$ is equal to 0.98, indicating a good approximation of the actual density.

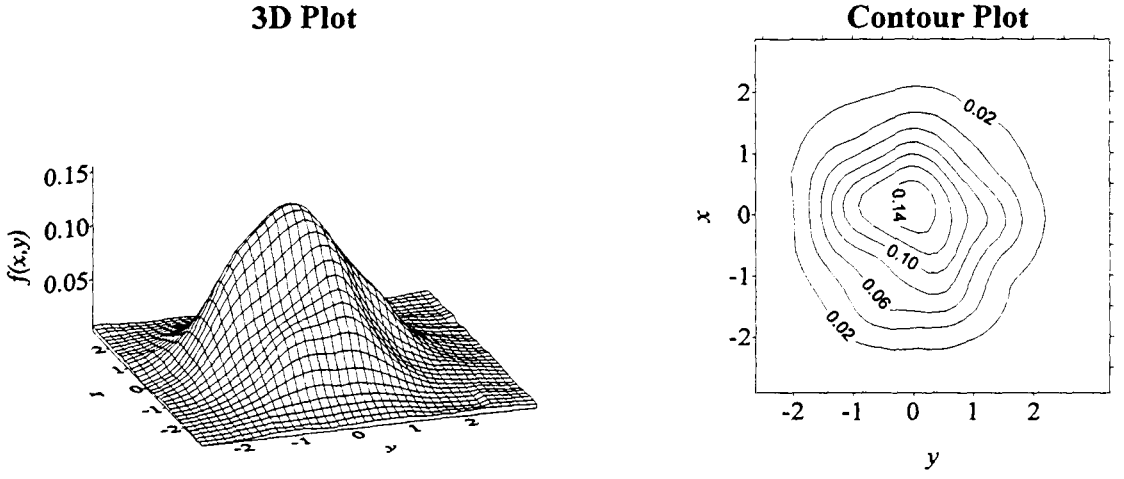

Figure 3.4 Estimated Bivariate Probability Density Between x and y - Gaussian Kernel

Figure 3.5 shows the estimated bivariate probability density produced by the city block distance kernel. From the 3-dimensional plot it can be seen that the overall density is also reasonably well approximated but is not as smooth as the estimate produced by the Gaussian kernel. The contour plot shows that there is very little correlation between x and y, but once again because it is only an estimate, the rings are not perfectly concentric. The estimate produced by the city block distance kernel is very similar to the estimate produced by the Gaussian kernel and the correlation coefficient between the actual bivariate normal probability density $f(x,y)$ and the estimated bivariate normal probability density $\hat{f}(x,y)$ is also equal to 0.98, indicating a good approximation of the actual density.

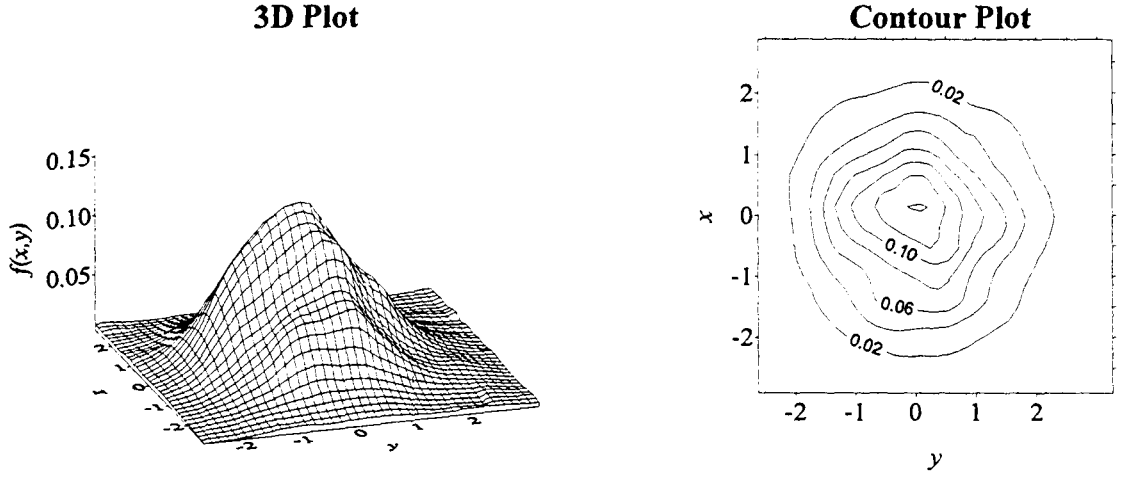

Figure 3.5 Estimated Bivariate Probability Density Between x and y - City Block Distance Kernel

Based on the similar approximation abilities of the Gaussian and city block distance kernels, it was decided to proceed with the city block distance kernel because of its computational simplicity and the consequent reduction in processing time required.

2.) When computing the conditional expectations in Equation 3.39, a GRNN has been used instead of the regression proposed by Sharma (2000) in Equations 3.40 and 3.41. The GRNN also uses the city block distance kernel function and therefore, also has the advantage of computational simplicity.

A trial was conducted whereby the original algorithm was further adapted to include a joint estimate of the mutual information, measured between a set of inputs and the output, as given by:

$$MI = \frac{1}{n}\sum_{i=1}^{n} \log_e \left[\frac{P_{X,Y}(x_{1_i}, x_{2_i}, ..., x_{d_i}, y_i)}{P_X(x_{1_i}, x_{2_i}, ..., x_{d_i}) P_Y(y_i)} \right] \tag{3.46}$$

where there are d inputs in the subset and $x_{1_i}, x_{2_i}, ..., x_{d_i}$, y_i are the i-th multivariate sample in a sample of size n, $P_Y(y_i)$ is the univariate density of the output and, $P_X(x_{1_i}, x_{2_i}, ..., x_{d_i})$, and $P_{X,Y}(x_{1_i}, x_{2_i}, ..., x_{d_i}, y_i)$ are the joint probability densities of the inputs and the inputs and output, respectively. The stepwise procedure proposed by Sharma (2000) only considers the PMI between a single input and the output (bivariate case). Instead of the stepwise procedure, it was initially envisaged that a genetic algorithm could be used to maximise the joint PMI between a subset of inputs and the output. However, experimentation with this approach proved unsuccessful. The reason for this arises from the fact that joint measures of the mutual information criterion become increasingly inaccurate as the dimensionality increases if the sample size of the data is not large enough. This is because the kernel density estimates require increasingly large sample sizes as the number of inputs (dimensionality) increases. This characteristic is shown in Table 3.7, which gives, as a function of dimension, the sample size required to achieve a mean square error less than 0.1 when the true density is unit multivariate normal and the density is estimated at the point **0**, with a bandwidth chosen to minimise the mean square error at this point.

It can be seen from Table 3.7 that the limited availability of data for most water resources case studies would preclude the use of this approach if there were more than 4 or 5 inputs being considered. For example, if there were 5 inputs, the dimensionality of the kernel density estimate $P_{X,Y}(x_{1_i}, x_{2_i}, ..., x_{d_i}, y_i)$ would be 6, requiring a sample size of 2790. With 5 inputs there are only 31 possible input subsets and complete

enumeration should be used instead of an input determination procedure. In addition, for non-Gaussian distributions and/or very sparse data sets, even larger sample sizes may be required. Therefore, the genetic algorithm and joint PMI approach to input determination was not investigated further in this study. But rather, the original stepwise PMI algorithm proposed by Sharma (2000) was used with the two modifications to the original algorithm as described above.

Table 3.7 Sample Size Required to Ensure That the Relative Mean Square Error at Zero is Less than 0.1, When Estimating a Standard Multivariate Normal Density Using a Normal Kernel and Bandwidth That Minimises the Mean Square Error at Zero (source: Silverman, 1986)

Dimensionality	Required Sample Size
1	4
2	19
3	67
4	223
5	768
6	2790
7	10700
8	43700
9	18700
10	842000

3.5.3 Application of the Input Determination Methods to Synthetic Data Sets

Since changes were made to the original stepwise PMI algorithm, the modified algorithm's ability was tested on four data sets with known dependence attributes. In addition, it was considered necessary to compare all of the input determination methods on synthetic data, before applying these methods to the real-world case studies. The first three models used to generate the synthetic data are time series models used by Sharma (2000) to test the original stepwise PMI algorithm. The fourth model is a nonlinear system. The models are:

AR1:

$$x_t = 0.9x_{t-1} + 0.866e_t \tag{3.47}$$

where e_t is a Gaussian random deviate with a zero mean and unit standard deviation.

AR9:

$$x_t = 0.3x_{t-1} - 0.6x_{t-4} - 0.5x_{t-9} + e_t \tag{3.48}$$

where e_t is a Gaussian random deviate with a zero mean and unit standard deviation.

TAR2 – Threshold Autoregressive order 2:

$$x_t = \begin{cases} -0.5x_{t-6} + 0.5x_{t-10} + 0.1e_t, & \text{if } x_{t-6} \le 0 \\ 0.8x_{t-10} + 0.1e_t, & \text{if } x_{t-6} > 0 \end{cases} \tag{3.49}$$

Nonlinear System:

$$y = (x_2)^3 + \cos(x_6) + 0.3\sin(x_9) \tag{3.50}$$

For the first three time series models, five hundred and twenty data points from each of the above synthetic models were generated with the first twenty points being discarded to reduce the effect of an arbitrary initialisation. The first fifteen lags were chosen as potential model inputs. From these potential inputs, the final input subset was selected using each input determination technique. For the nonlinear system, fifteen standard Gaussian random variables were generated, each consisting of five hundred data points. These fifteen variables were chosen as the potential model inputs and the final input subset was selected using each input determination technique.

The results of the modified stepwise PMI algorithm applied to the four test problems are shown in Table 3.8. The bold numbers in the table represent the higher of the PMI and the 95th percentile randomised sample. The last row in each model's results represents the cutoff point i.e. the input that was not included because the 95th percentile randomised sample was greater than the corresponding PMI score. From Table 3.8 it can be seen that the modified stepwise PMI input selection algorithm was able to correctly select the right inputs for each of the four models. This is an improvement over the original algorithm used by Sharma (2000), which selected one input in addition to what is necessary in the case of the two variable threshold autoregressive model (TAR2). Therefore, the modified stepwise PMI input selection algorithm can be considered a suitable means of correctly identifying the inputs to both linear as well as more complex nonlinear models.

An advantage of the stepwise PMI algorithm is that additional information about the system is also obtained, including the order of significance of the inputs and the relative

significance. For example, for the AR9 model in Equation 3.48, the autoregressive coefficients for lags 1, 4 and 9, are 0.3, -0.6 and -0.5, respectively. The stepwise PMI algorithm correctly selected the inputs in order of greatest significance i.e. lag 4 first, lag 9 second and then lag 1 with PMI scores of 0.354, 0.204 and 0.104, respectively. As a result, important information about the system under investigation can be obtained when using the stepwise PMI algorithm.

Table 3.8 Modified stepwise PMI input selection algorithm results on the synthetic data test problems

Model	Predictors	PMI	95th Percentile Randomised Sample PMI
AR1	x_{t-1}	**0.604**	0.080
	x_{t-9}	0.029	**0.033**
AR9	x_{t-4}	**0.354**	0.080
	x_{t-9}	**0.204**	0.059
	x_{t-1}	**0.104**	0.044
	x_{t-3}	0.026	**0.034**
TAR2	x_{t-10}	**0.363**	0.098
	x_{t-6}	**0.077**	0.064
	x_{t-9}	0.053	**0.060**
Nonlinear	x_2	**0.558**	0.097
	x_9	**0.082**	0.031
	x_6	**0.072**	0.026
	x_{10}	0.009	**0.013**

Table 3.9 shows the subsets of inputs selected by the PCA-GAGRNN and SOM-GAGRNN input determination procedures for each test problem, after i.) the unsupervised techniques were applied, and ii.) the supervised technique was applied. In this experiment, the GA had a population size of 50 and was run for 100 generations, thereby resulting in the development of 5000 GRNN models. Out of the three time series problems (AR1, AR9 and TAR2), the maximum number of inputs remaining after the application of the unsupervised technique was 12. In this case, total enumeration of the search space would result in the development of 2^{12}-1 = 4095 GRNN models. Hence, the GAGRNN conducted more evaluations than total enumeration and would not be used in reality, but since the purpose of this experiment was to compare the different input determination procedures, the GAGRNN was used instead of total enumeration.

It is also important to note that PCA reduces the original input set to a number of PCs, each of which is a linear combination of all of the original inputs. As a result, it is not

possible to determine the exact inputs to the system for this method, but rather, the original variability in the first fifteen lags is captured by a number of PCs and the GAGRNN selects the PCs required to minimise the test set RMSE. Neither PCA nor the SOM were used for the nonlinear system test problem since all fifteen candidate inputs were already independent.

Table 3.9 Input Subsets Selected using PCA-GAGRNN and SOM-GAGRNN for the Synthetic Data Test Problems

Test Problem	Input Determination Method	After Unsupervised Technique	After Supervised Technique
AR1	PCA-GAGRNN	PCs 1, 2, 3, 4, 5	PCs 1, 2, 3, 4, 5
	SOM-GAGRNN	Lags 1, 4, 6, 9, 12, 14	Lag 1
AR9	PCA-GAGRNN	PCs 1, 2, 3, 4, 5, 6, 7, 8	PCs 1, 2, 3, 4, 5, 6, 7, 8
	SOM-GAGRNN	Lags 1, 3, 5, 6, 7, 8, 9, 11	Lags 1, 3, 5, 7, 8, 9
TAR2	PCA-GAGRNN	PCs 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	PCs 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12
	SOM-GAGRNN	Lags 1, 3, 4, 5, 6, 9	Lags 3, 4, 5, 6
Nonlinear	GAGRNN	Not Used	Inputs 2, 14

Note: No unsupervised technique (PCA or SOM) was used for the nonlinear system test problem since all inputs are independent.

The results in Table 3.9 indicate that the number of PCs required increased as the complexity of the problem increased. In the first two time series test problems (AR1 and AR9), the GAGRNN retained all of the PCs. For the TAR2 test problem, one PC was omitted by the GAGRNN. The PC omitted for this test problem was PC 11, and examination of the eigenvectors obtained for PC 11 indicated the reason for its omission. The actual input variables for the TAR2 test problem are lags 6 and 10 and the coefficients for lags 6 and 10 were 0.004 and –0.006, respectively. Such small coefficients for the two important input variables would have meant that PC 11 offered very little information about the system and hence, this PC was not selected by the GAGRNN.

Table 3.9 also shows that the SOM was able to reduce the original dimensionality of the time series problems by clustering similar inputs together. However, since it is an unsupervised technique, there is no guarantee that the actual inputs to the test problem

are included in the resulting subset after this technique has been applied. Only the input closest to the centre of the cluster is selected. Therefore, if the actual test problem input clusters with other inputs, it is possible that one of the other inputs may be selected instead of the actual test problem input. If these variables are closely related, then the effect of selecting either one should make little difference to the final model from a predictive sense. For the AR1 test problem, the actual input (i.e. lag 1) was included in the subset obtained after applying the SOM technique. The GAGRNN was able to successfully determine the actual input for this problem. For the AR9 test problem, only two of the three actual test problem inputs (i.e. lags 1 and 9) were included in the subset obtained after applying the SOM technique. The remaining actual input (i.e. lag 4) clustered with lag 3, and since lag 3 was closest to the centre of the cluster, this lag was selected in preference to lag 4. After applying the GAGRNN to the subset of inputs obtained after the SOM technique, lags 1, 3, and 9 were included in the final subset of inputs, however, lags 5, 7, and 8 were also included. For the TAR2 test problem, only one of the two actual test problem inputs (i.e. lag 6) was included in the subset obtained after applying the SOM technique. The other actual input (i.e. lag 10) clustered with lag 9, and since lag 9 was closest to the centre of the cluster, this lag was selected in preference to lag 10. The GAGRNN was then applied to the subset of inputs obtained after the SOM technique and lags 3, 4, 5 and 6 were included in the final subset of inputs, however, lag 9 was not included. Finally, for the nonlinear system test problem, the GAGRNN only found one of the actual inputs (i.e. input 2) and also included input 14 in the final subset. This is most likely due to the fact that the nonlinear system in Equation 3.50 is comprised of three nonlinear terms added together. Input 2 appears in the first term, and this term has a much larger magnitude than both the second and third terms. Since the first term dominates the nonlinear system equation, the input from this term was selected by the GAGRNN.

The stepwise PMI algorithm was successful in identifying all of the actual inputs for each test problem, whereas the SOM-GAGRNN technique failed to identify all of the actual inputs for the AR9, TAR2 and nonlinear system test problems, and often included additional inputs. The inability of the GAGRNN to select the actual model inputs is most likely due to the fact that the GRNN is only able to provide an approximation of the true underlying function. This approximation may be adequate from a predictive sense, but is less useful when knowledge of the system is required since the actual model inputs could not be identified. An additional reason that may have contributed to the inability of the GAGRNN in selecting the actual model inputs may have resulted from the objective function used during the GA optimisation. The test set RMSE was used and it is possible that this is not the most suitable means for evaluating the fitness of different input subsets. The use of an alternative objective

function may give different results. This is symptomatic of all *model-based* approaches, which are prone to the difficulties involved in selecting an appropriate objective function. *Model-free* approaches, such as the stepwise PMI algorithm, have the advantage that they avoid the need to select an objective function.

From the point of view of gaining insight into the underlying processes, it is important that the input determination technique is able to identify the actual inputs. However, from the point of view of predictive performance, identifying the actual inputs may be less important as a closely related input may suffice. Consequently, to compare the input determination procedures using predictive performance, GRNN models were developed for each input subset. Since GRNN models had already been developed as prt of the PCA-GAGRNN and SOM-GAGRNN input determination techniques, the only additional models that needed to be developed were those using the inputs identified by the modified stepwise PMI algorithm. For each synthetic model, the 500 data points were divided into statistically similar training and testing sets. 80% of the data (400 data points) were used for the training set and 20% (100 data points) were kept for testing purposes. For a strict comparison of each model's true generalisation ability, an independent validation set should be used, however, since the aim of this experiment was to study how model performance varied with different input subsets, it was not deemed necessary to use a validation set.

The results of the comparison are shown in Table 3.10. All methods provided suitable means of determining the appropriate inputs from a predictive sense, as indicated by the similar RMSEs obtained from each input determination procedure for each test problem. For the AR1 test problem the results from the stepwise PMI algorithm and the SOM-GAGRNN methods were identical, since both methods identified the actual model input (i.e. lag 1). For the AR9 test problem, the stepwise PMI algorithm performed marginally better than the PCA-GAGRNN and SOM-GAGRNN techniques when measured using the testing set RMSE. For the TAR2 test problem, PCA-GAGRNN provided the best test set performance. For the nonlinear system test problem, the GAGRNN performed slightly better on the test set than the model developed using the inputs identified by the stepwise PMI algorithm. However, the difference is unlikely to be significant. Based on the results obtained from this study, if predictive importance is the primary objective, then the PCA-GAGRNN, SOM-GAGRNN or the stepwise PMI algorithm could be used. However, if an understanding of the system under investigation is also an important objective, then it is recommended that the stepwise PMI algorithm is used as it was able to successfully identify the true model inputs to each of the test problems considered.

Table 3.10 RMSEs of the GRNN Models Produced from Each Input Determination Method for the Synthetic Data Test Problems

Test Problem	Input Determination Method	Training Set RMSE	Testing Set RMSE
AR1	PCA-GAGRNN	0.229	0.552
	SOM-GAGRNN	0.443	0.485
	Stepwise PMI	0.443	0.485
AR9	PCA-GAGRNN	0.173	0.606
	SOM-GAGRNN	0.333	0.673
	Stepwise PMI	0.369	0.586
TAR2	PCA-GAGRNN	0.205	0.668
	SOM-GAGRNN	0.548	0.754
	Stepwise PMI	0.626	0.731
Nonlinear	GAGRNN	0.164	0.219
	Stepwise PMI	0.064	0.223

3.6 Choice of Network Type and Architecture

3.6.1 Introduction

The choice of network type involves selecting an appropriate ANN model for the application under investigation. Once the appropriate network type has been chosen, the network architecture must be selected, which includes determining the number of connection weights (free parameters) and the way that information flows through the network. An appropriate architecture can be thought of as one that yields the best performance in terms of error minimisation, but also retains a compact and simple structure. This task is one of the most important, but also one of the most difficult in the model building process (Maier and Dandy, 2000b). As the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000a) have identified, "no unified theory exists for determination of such an optimal ANN architecture".

Feedforward ANNs, where nodes in one layer are only connected to nodes in subsequent layers, are the most commonly used type of network for prediction and forecasting applications (Kwok and Yeung, 1997a). However, feedforward ANNs are only special cases of recurrent networks. In recurrent networks, nodes in one layer can be connected to nodes in the next layer, the previous layer and even to themselves. The advantage of feedback connections is that the network is able to model dynamic relationships implicitly since the output of a neuron is essentially an implied function of previous and present inputs (Sajikumar and Thandaveswara, 1999). However, this type of dynamic information can easily be incorporated into a feedforward ANN explicitly by including lagged inputs.

Feedforward ANNs are also the most common network type for the prediction and forecasting of water resources variables. Coulibaly et al. (1999) reviewed a number of papers on ANN modelling in hydrology and found that about 90% of the experiments reviewed made use of feedforward MLPs trained by the standard backpropagation algorithm. A similar finding was apparent in the review conducted by Maier and Dandy (2000b), who found that over 95% of the papers reviewed made use of feedforward ANNs. Dawson and Wilby (2001) found that all of the papers they reviewed on ANN hydrological modelling made use of feedforward ANNs.

In this research, feedforward ANNs have been investigated because (i) they have been found to perform well in comparison with recurrent ANNs in many applications (e.g. Khotanzad et al., 1997), (ii) of all the models currently in use, their processing speed is among the fastest (Masters, 1993), (iii) as previously mentioned, a feedforward MLP

with a sufficient number of hidden nodes is able to approximate any continuous function to any degree of accuracy, providing efficient training is performed (Cybenko, 1989; Hornik et al., 1989) and, (iv) there are no clear practical advantages in using recurrent networks over feedforward networks with limited time windows (Hochreiter and Schmidhuber, 1997).

As a starting point, Dawson and Wilby (2001) recommend that a backpropagation MLP should be used, since this provides a suitable benchmark against which to evaluate and compare other models. In a MLP, the number of hidden layers are usually fixed and the number of connection weights are varied by choosing the number of nodes in each of the hidden layers. Hornik et al. (1989) and Cybenko (1989) have shown that only one hidden layer is required to approximate any continuous function, provided there are a sufficient number of connection weights. However, this proof was based on the premise that an optimal set of connection weights has been identified. In many practical situations this is not the case and some functions may be better approximated using two hidden layers (Cheng and Titterington, 1994; Flood and Kartam, 1994). Flood and Kartam (1994) have pointed out that the use of more than one hidden layer provides greater flexibility and can enable approximation of complex functions using fewer connection points. They recommended that two hidden layers should be used as a starting point.

Selecting the number of nodes to use in the hidden layers is a critical aspect since this controls the number of connection weights used in the model. There is a need to strike a balance between having a sufficient number of free parameters (connection weights) to adequately learn the problem and having too many free parameters such that the network is susceptible to overtraining. The aim of training is to minimise the network's predictive error, however, in the presence of noise, reducing the error beyond a certain point can result in overtraining (Maier and Dandy, 2000b). An overtrained network overfits the training data since the ANN has learnt the idiosyncrasies in the training set and therefore, has poor generalisation ability.

A wide variety of methods have been invented to determine the number of nodes to use in the hidden layer. The most common of these methods are rule-of-thumb guidelines (e.g. Equations 2.70 – 2.72, Section 2.5.4) and heuristic methods, which can be further divided into constructive and pruning approaches. These are also discussed in Section 2.5.4. The disadvantage of rule-of-thumb approaches is that determining a suitable network architecture is highly problem dependent and hence, the guidelines may not suggest an optimal number of hidden layer nodes for the application being investigated. In addition, different guidelines suggest different architectures and it is not always clear

which guideline should be used. The heuristic approaches are not without their shortcomings either since they are susceptible to becoming trapped in structural local optima and only investigate a restricted subset of all possible network architectures (Angeline et al., 1994). To overcome many of these problems, evolutionary ANNs (EANNs) have been proposed by numerous authors (see Section 2.4.1.2).

In the present study, a feedforward MLP trained using the backpropagation algorithm has been investigated. This model represents the most commonly used ANN in modelling water resources variables and provides the benchmark against which other types of ANN models can be compared. To determine the suitable architecture for the MLP, an evolutionary backpropagation MLP (EBMLP) (Section 3.6.2.1) was used. The EBMLP makes use of a GA to attempt to optimise the number of hidden layers, the number of nodes in each of the hidden layers, and the transfer function at each node. For comparison with the EBMLP model, an additional feedforward ANN known as the general regression neural network (GRNN) (Section 3.6.2.2) was also investigated. The GRNN architecture is fixed and does not require optimisation.

3.6.2 Methods

3.6.2.1 Evolutionary Backpropagation Multilayer Perceptron (EBMLP)

The EBMLP is an evolutionary artificial neural network (EANN) with a feedforward MLP architecture, trained using the backpropagation algorithm. A review of EANNs has been presented in Chapter 2. After reviewing the international journal papers published before the end of 2001 on the application of ANNs to water resources prediction/forecasting, only one paper found was found that used genetic algorithms to optimise ANN architecture. Abrahart et al. (1999) used a genetic algorithm to optimise the architecture of a feedforward MLP rainfall-runoff ANN model trained using the RPROP algorithm (Riedmiller and Braun, 1993). The genetic algorithm methodology was compared with two pruning methodologies for architecture determination. The first was magnitude based pruning, which eliminates unwanted links, and the second was skeletonization, which eliminates unwanted nodes. A statistical assessment of the results was conducted and it was discovered that there was no clear "winner", with the networks developed using the pruning and genetic algorithm methodologies producing similar results.

In this study, the EBMLP was used to optimise the architecture because the very large number of potential networks precludes a total enumeration of the search space. This is especially true of networks that make use of a second hidden layer. In addition,

structural hill-climbing techniques, such as the heuristic pruning or constructive algorithms are susceptible to becoming trapped in local minima and only investigate a restricted subset of all possible network architectures.

To implement the EBMLP in this research, the commercially available software package, *NeuroGenetic Optimizer* (*NGO*) (BioComp Systems Inc., 1998) was used. Some of the features of the software package are described below:

1. The training and testing sets can be automatically scaled to a user defined range, producing linearly transformed training and testing sets. The ranges used in conjunction with the positive output transfer functions (i.e. linear and sigmoid) were -1.0 to +1.0 for the network inputs and +0.2 to +0.8 for the network outputs. The ranges used in conjunction with the positive/negative output transfer function (i.e. hyperbolic tangent) were -1.0 to +1.0 for the network inputs and -0.8 to +0.8 for the network outputs.
2. Cross-validation training is supported, including a user defined minimum number of training set passes. This parameter sets the first training pass for which NGO will consider the termination criteria. This allows completion of the initial phase of learning before considering stopping training. In addition, a maximum number of passes must also be specified to conserve time in the optimisation process. Training is stopped when a user specified number passes of the training set occur without finding a new minimum test set error or when the maximum number of passes through the training set has been reached.
3. Multiple hidden layers can be enabled. Selecting this feature directs NGO to include a second hidden layer in the genetic algorithm search. This feature was enabled in this research to determine if a second hidden layer was required.
4. The number of hidden layer nodes can be limited. This feature limits the search space of the genetic optimisation to 8, 16, 32, 64, 128, or 256 hidden nodes. The setting applies to all hidden layers within the network.
5. An option is available to build smaller networks. Selecting this option encourages NGO to build networks that are more parsimonious, since networks with a larger number of hidden layer nodes are penalised. This is achieved in NGO by using a user defined hidden node influence parameter which influences the degree the system optimises for accuracy versus network compactness. The hidden node influence parameter rewards networks with fewer hidden nodes.
6. The ANN's inputs can be optimised using the genetic algorithm search, however, since the inputs have already been optimised during the input determination phase of the methodology (Section 3.5), the inputs were fixed and were not optimised during the determination of the network architecture.

7. Three different transfer functions can be included in the genetic algorithm search. These include linear, sigmoidal and hyperbolic tangent transfer functions. The transfer functions in NGO are an attribute of the nodes rather than the layers, meaning that networks can use multiple transfer functions per layer. For example, a 20 node hidden layer may comprise of 10 nodes using the hyperbolic tangent transfer function, 5 nodes using the sigmoid transfer function and 5 nodes using the linear transfer function. This increases the model's complexity and provides greater flexibility.
8. The NGO uses fast backpropagation and linearly decreases the learning rate and momentum as training progresses. Different starting and ending learning rates can be set for the different layers. A starting and ending value for the momentum can also be set.
9. The initial weight distribution can be randomly generated within a range set by the user. The default range was chosen which is -0.3 to +0.3 for all networks.
10. The population size and maximum number of generations can be set by the user. The maximum population size is 300 and there is no limit on the maximum number of generations.
11. The genetic algorithm includes percentage selection, in which the chromosomes (networks) are ranked according to their fitness and the best x% in the population are retained.
12. Since the selection technique discards poor chromosomes, the population needs to be refilled at each generation. NGO provides two methods, including random refilling, which randomly creates new population members, and cloning, which clones the survivors of the selection process to refill the population.
13. Two crossover methods are available. The first is one-point crossover, whereby a cut point is selected and the bits (genetic material) are exchanged between the cut point and the end of the string of the parents, essentially swapping tails. The second method is a two-point crossover whereby there are two cut points and the bits between these two positions in the parent chromosome strings are exchanged.
14. Two methods of mutation are available. The first method is random exchange, whereby two points in a given chromosome string are selected and the values at those points are exchanged. The second method is section reversal, where a section of the chromosome is reversed. The mutation rate can be set by the user.
15. The performance measure used to evaluate the fitness of each network can be selected by the user. The five provided in NGO include; root mean squared error, mean squared error, R-squared, average absolute error and relative accuracy.
16. The fitness of each network can be calculated using proportions of the performance measures on the training and testing sets. In this research, the fitness of each network was selected as 100% of the test set RMSE.

17. The stopping criteria supported by NGO include; the number of generations run, elapsed runtime, best network accuracy found and population convergence. Population convergence refers to stopping the genetic algorithm search based on the similarity of members of the population. To measure the population convergence, a convergence factor is used which measures the average standard deviation of genes in the population. Once this convergence factor has been reached, the system is halted.

Lists of the neural and genetic parameters used in the implementation of the EBMLP are given in Table 3.11 and Table 3.12, respectively. The software defaults and recommendations were used wherever possible.

Table 3.11 Neural Network Parameters

Parameter	Value/Selection
Minimum passes through the training set	200
Maximum passes through the training set	300
Number of retries without finding a new minimum error before training is stopped	70
Multiple hidden layers enabled	Yes
Transfer functions used	Hyperbolic tangent, sigmoidal and linear
Starting/ending learning rate for hidden layer 1	0.4 to 0.1
Starting/ending learning rate for hidden layer 2	0.4 to 0.1
Starting/ending learning rate for output layer	0.2 to 0.1
Starting/ending momentum	0.2 to 0.1

Table 3.12 Genetic Algorithm Parameters

Parameter	Value/Selection
Population size	20
Max. number of generations	100
GA stopping criteria	Max. number of generations reached
Neural node influence	0.1
Selection percentage	50%
Refill strategy	Cloning
Crossover	Single-point
Mutation	Random exchange
Mutation rate	0.25

3.6.2.2 General Regression Neural Network (GRNN)

A review of the GRNN has been presented in Chapter 2. The review of ANN applications in water resources modelling (Section 2.6) revealed that there were no journal papers reporting the use of the GRNN in this field. Therefore, to the best knowledge of the author, the current research represents the first use of this type of network in the field of water resources prediction/forecasting.

Training a GRNN involves determining a suitable value for the sigma weight (smoothing constant) σ. The most common methods for determining a suitable value for the sigma weight are based on trial-and-error (e.g. Caudill, 1993). However, as Specht (1991) states, the curve of mean squared error versus σ typically exhibits a wide range of values near the minimum, and hence, it is not difficult to pick a good value for σ. In addition, the curve is usually parabolic in shape, and because of this, the inverse Hessian method is used in this research to determine a near-optimal value of σ, since it exhibits quadratic convergence near the minimum.

The inverse Hessian method (also known as Newton's method) is a second-order local optimisation method. The relationships for this method can be derived from the objective function following Press et al. (1992). Given a suitable objective function E

(e.g. MSE), a truncated Taylor series expansion of the objective function is performed about a point σ_0, which is sufficiently close to some minimum point σ:

$$E(\sigma) \approx E(\sigma_0) + \frac{\partial E}{\partial \sigma}\delta\sigma + \frac{1}{2}\frac{\partial^2 E}{\partial \sigma}\delta\sigma^2 + O(\|\delta\sigma\|^3) \tag{3.51}$$

Substituting (σ-σ_0) for $\delta\sigma$ simplifies Equation 3.51. The derivatives in the second and third terms of Equation 3.51 are called the gradient and Hessian of the objective function, respectively. Since the gradient and the Hessian are evaluated at the point σ_0, they are therefore constants giving:

$$E(\sigma) = E(\sigma_0) + \nabla E|_{\sigma_0}\ (\sigma - \sigma_0) + \frac{1}{2}(\sigma - \sigma_0)H|_{\sigma_0}\ (\sigma - \sigma_0) + O(\|\sigma - \sigma_0\|^3) \tag{3.52}$$

where $\nabla E|_{\sigma_0}$ is the gradient and $H|_{\sigma_0}$ is the Hessian (both evaluated at σ_0). The differentiation of Equation 3.52 with respect to the parameter (σ) produces:

$$\nabla E = \nabla E|_{\sigma_0} + (\sigma - \sigma_0)H|_{\sigma_0} \tag{3.53}$$

At the minimum of the objective function, the gradient equals zero, therefore, the gradient (∇E on the left hand side of Equation 3.53) is set to zero. By rearranging Equation 3.53, it is then possible to obtain an expression relating a point near the minimum to the minimum point. The inverse Hessian method for function minimisation is given by:

$$\sigma = \sigma_0 - \nabla E|_{\sigma_0} \cdot H^{-1}|_{\sigma_0} \tag{3.54}$$

From Equation 3.54, it can be seen that the inverse Hessian method requires calculation of the gradient and Hessian. The simplest method to generate parameter derivatives is the divided difference method. Using the divided difference method the gradient can be approximated using:

$$\frac{\partial E}{\partial \sigma} \approx \frac{E(\sigma + \delta\sigma) - E(\sigma)}{\delta\sigma} \tag{3.55}$$

where the partial derivative of the objective function E is found with respect to the sigma weight σ and $\delta\sigma$ is the perturbation in the sigma weight. Decreasing the size of the perturbation $\delta\sigma$ increases the accuracy until roundoff errors become a problem, due to approximations when storing a number in computer memory.

When the divided difference method is used to calculate the Hessian, three points are required (Figure 3.6). Therefore, a positive perturbation $\delta\sigma$ and a negative perturbation $-\delta\sigma$ must be used. To approximate the gradient at a point σ, the divided differences between points 2 and 1 and points 3 and 1 (Figure 3.6) are calculated and then averaged:

$$\begin{aligned}\frac{\partial E}{\partial \sigma} &\approx \frac{\left[\left(\frac{E(\sigma+\delta\sigma)-E(\sigma)}{\delta\sigma}\right)+\left(\frac{E(\sigma)-E(\sigma-\delta\sigma)}{\delta\sigma}\right)\right]}{2} \\ &\approx \frac{E(\sigma+\delta\sigma)-E(\sigma-\delta\sigma)}{2\delta\sigma}\end{aligned} \tag{3.56}$$

To approximate the Hessian at a point σ, the divided differences between points 2 and 1 and points 3 and 1 (Figure 3.6) are calculated and are assumed to apply at points 1a and 1b (Figure 3.6), respectively. A divided difference is then calculated between points 1a and 1b (Figure 3.6) as shown in Equation 3.57.

$$\begin{aligned}\frac{\partial^2 E}{\partial \sigma} &\approx \frac{\left[\left(\frac{E(\sigma+\delta\sigma)-E(\sigma)}{\delta\sigma}\right)-\left(\frac{E(\sigma)-E(\sigma-\delta\sigma)}{\delta\sigma}\right)\right]}{\delta\sigma} \\ &\approx \frac{E(\sigma+\delta\sigma)-2E(\sigma)+E(\sigma-\delta\sigma)}{\delta\sigma^2}\end{aligned} \tag{3.57}$$

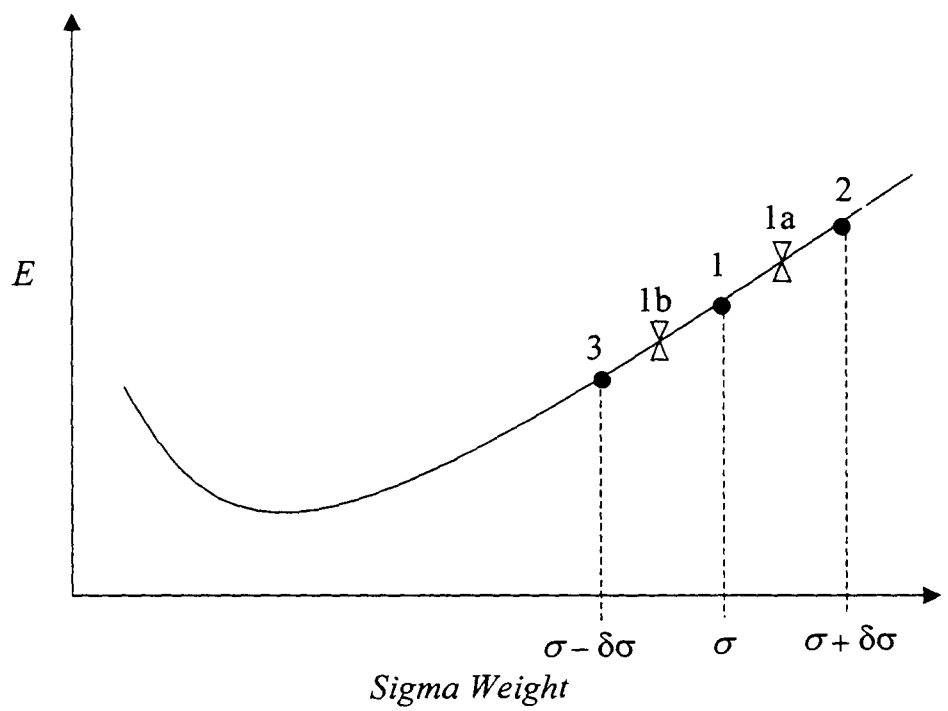

Figure 3.6 The Divided Difference Method For Parameter Derivatives

The architecture is fixed in a GRNN and consists of an input layer, a pattern layer, a summation layer and an output layer (see Chapter 2, Figure 2.11). The number of nodes in the input layer is fixed by the number of model inputs, the number of nodes in the pattern layer is determined by the number of training samples and the summation layer comprises of two nodes for each output in the network. The first node in the summation layer is an *A* summation unit and the second is a *B* summation unit. If one pattern layer node is assigned to each training pattern, then the *B* unit weights are all set to unity. In this case, the *A* unit weights are the actual output values associated with the pattern stored at each pattern layer node.

The GRNN is a memory-based network and as the number of training samples increases, the size of the GRNN increases accordingly. To overcome the problem of very large training sets, Specht (1991) proposed an adaptation to the GRNN which involves the use of clustering techniques (e.g. self-organizing map, k-means clustering) to group training samples. Each cluster or group is represented by only one pattern layer node, which takes the values of the cluster's centroid as its incoming weights. In addition, the weights on the summation layer node must also be changed. The weights on the *B* summation layer node keep count of how many times a pattern has clustered at a particular pattern layer node. The weights on the *A* summation layer node keep count of the sum of the actual output for the training samples. Therefore, if a training observation Y^j is encountered for cluster *i*, the weights are incremented using:

$$\begin{aligned} A^i(k) &= A^i(k-1) + Y^j \\ B^i(k) &= B^i(k-1) + 1 \end{aligned} \tag{3.58}$$

where $A^i(k)$ and $B^i(k)$ are the values of the weights for cluster *i* after *k* observations.

Restating Equation 2.52, the GRNN equation now becomes:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A^i \cdot \exp\left(-\frac{C_i}{\sigma}\right)}{\sum_{i=1}^{m} B^i \cdot \exp\left(-\frac{C_i}{\sigma}\right)} \tag{3.59}$$

where $m<n$ is the number of clusters and $C_i = \sum_{j=1}^{p} \left| X_j - X_j^i \right|$. This adaptation is considered in this section since it involves a change to the original architecture. To implement the clustering of the training data, a self-organizing map was used.

Another adaptation to the original GRNN is the inclusion of separate sigma weights for each of the model inputs. Breiman et al. (1977) suggested that when using Parzen window estimation, a better estimation of the densities can be obtained by using a different sigma weight for each input. Masters and Land (1997) and Chtioui (1999) used separate sigma weights for each input as an implicit way of scaling the inputs according to their relative predictive importance. Separate sigma weights can easily be incorporated into the GRNN equations. Restating Equation 2.53, for a *d*-dimensional input set the GRNN equation becomes:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A^i \cdot \exp\left(-C_i\right)}{\sum_{i=1}^{m} B^i \cdot \exp\left(-C_i\right)} \tag{3.60}$$

where

$$C_i = \sum_{j=1}^{d} \frac{\left|X_j - X_j^i\right|}{\sigma_j} \tag{3.61}$$

When multiple sigma weights are used, the fast training capability of the original single-sigma version of the GRNN is negated and there is also an increased likelihood of multiple local optima. This makes the model difficult to train in many circumstances. In addition, the error function is often nearly flat, meandering downward slowly, with many twists and turns along the way (Masters and Land, 1997). Masters and Land (1997) attempted to solve this difficult optimisation problem and found that gradient based optimisation techniques were not suitable when there are more than a few inputs to the network, due to the increased likelihood of multiple local minima. Therefore, if the multiple sigma version of the GRNN has more than a few inputs, stochastic training algorithms are advantageous because of their ability to avoid becoming trapped in these local minima. In this research, a GA was used to determine a near-optimal set of sigma weights for the multiple-sigma GRNN. The floating point GA outlined in Section 3.3.2.1 was used, which utilises 2-member tournament selection, linear crossover and non-uniform mutation.

The formulation used for applying the GA to the optimisation of a GRNN is as follows: The objective is to find a set of sigma weights that minimise the objective function, as given by:

$$E(\bar{\sigma}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{3.62}$$

where E is the objective function (i.e. MSE), $\bar{\sigma}$ is the vector of sigma weights, $\hat{x}_i$ are the n predicted values in the test set and x_i are the actual or desired values in the test set. If there are d inputs to the network, then the vector of sigma weights will also be d-dimensional and the total search space S of $\bar{\sigma}$ is given by:

$$S = \left\{\sigma_j \mid \sigma_j^- < \sigma_j \leq \sigma_j^+; j = 1, 2, ..., d\right\} \tag{3.63}$$

where σ_j^- and σ_j^+ denote the lower and upper bounds of the parameter σ_j, respectively and are determined using *a priori* knowledge. The lower bound was set to 0 and the upper bound was set to 10 to allow a sufficient range for the sigma weights. The floating point GA described in Section 3.3.2.1 was then used to determine a near-optimal set of sigma weights for the GRNN. The objective (fitness) function in this case was the test set RMSE.

In this study, the software required to implement the various GRNN models was developed in Fortran. A single-sigma GRNN using the original architecture implementation (i.e. one pattern layer node for each training sample) was trained using the inverse Hessian method. Next, a hybrid SOM-single-sigma GRNN was developed. This model utilised the SOM to cluster the training set and effectively reduce the size of the GRNN model since one pattern layer node is assigned to each cluster centroid. The effect of reducing the GRNN's size was evaluated by using different size Kohonen layers in the SOM. As the size of the Kohonen layer is increased, so too the number of clusters increases and hence, the size and training times required by the GRNN also increase. Finally, a multiple-sigma GRNN trained using a floating point GA was developed and then compared to the best single-sigma GRNN.

3.7 Training (Optimisation)

3.7.1 Choice of Performance Measure

3.7.1.1 Introduction

In general, there are three performance criteria upon which an ANN model may be assessed, and these are (Maier and Dandy, 2000a):

- Training speed.
- Processing speed during recall.
- Prediction accuracy.

Unless training and processing speed is important (e.g. Shukla et al., 1996; Yang et al., 1996), the most commonly used performance criteria in water resources modelling is prediction accuracy.

In supervised ANN learning, the observed data are compared with the predictions generated by the ANN model. During this learning process, an optimisation algorithm is employed to determine a set of network weights that minimise some performance measure (objective function). However, the type of performance measure used in the optimisation process can have a significant influence on the final values of the model weights and hence, the model's generalisation ability (Diskin and Simon, 1977). There are numerous measures of prediction accuracy in existence and the unifying theme is that the measure must provide an accurate quantitative description of the model's predictive ability. Since the measures are commonly evaluated on an independent test set, the model's ability to generalise is assessed. A review of the performance measures commonly used in the field of water resources modelling is presented below. The ten performance measures investigated in this research are identified by the equations for E_k, $k = 1, \ldots, 10$.

3.7.1.2 Review of Performance Measures Used in Water Resources Modelling

Finding a near-optimal set of weights for an ANN model can be expressed in mathematical terms by the following general problem statement:

$$\text{minimise } E(w); \qquad E(w) = E\{O_i, P_i(w)\}_{i=1,\ldots,n}, \quad w \in W \tag{3.64}$$

where E is the performance measure of the model (objective function), O_i is the observed value (an element of a set of measured data), P_i is the predicted model output, n is the sample size and $\boldsymbol{W}$ is the set of feasible model weights with $\boldsymbol{w} \in \boldsymbol{W}$ a vector of chosen weights.

With a single modelled variable (output), the performance measure is often derived in the following form (Van der Molen and Pintér, 1993):

$$E(w) = \left[\frac{1}{n} \cdot \sum_{i=1}^{n} |O_i - P_i(w)|^{\tau} \right]^{1/b}, \quad 1 \le \tau \le \infty, \quad 1 \le b \le \infty \tag{3.65}$$

The factor $1/n$ allows for the comparison of the performance measure with different sample sizes. In the water resources literature, the most common cases of Equation 3.65 include: minimisation of the averaged absolute deviations (average absolute error, AAE) (Equation 3.66), the averaged least squared estimation problem (root mean squared error, RMSE) (Equation 3.67), and the minimisation of the maximum discrepency between the set of model outputs and the observations (minimax objective function) (Equation 3.68).

$$E_1 = \frac{1}{n} \cdot \sum_{i=1}^{n} |O_i - P_i(w)|, \qquad \tau = 1, \; b = 1 \tag{3.66}$$

$$E_2 = \left[\frac{1}{n} \cdot \sum_{i=1}^{n} |O_i - P_i(w)|^2 \right]^{1/2}, \quad \tau = 2, \; b = 2 \tag{3.67}$$

$$E_3 = \frac{1}{n} \cdot \max_i |O_i - P_i(w)|, \qquad \tau = \infty, \; b = 1 \tag{3.68}$$

As discussed by Van der Molen and Pintér (1993), the mathematical properties of Equation 3.65 are such that increasing τ ensures that the more significant discrepancies between model output and observed values (i.e. outliers) are penalised more heavily. The two possible extremes are represented by $\tau = 1$ and $\tau = \infty$, i.e. Equations 3.66 and 3.68, respectively. Equation 3.66 includes all pairs of predicted and observed values with equal weight, while Equation 3.68 considers only the maximum deviation. The most commonly used value for τ is two. Setting $b = 1$ gives rise to the mean squared error (MSE) (Equation 3.69), which is well-known in the literature.

$$\text{MSE} = \frac{1}{n} \cdot \sum_{i=1}^{n} |O_i - P_i(w)|^2, \qquad \tau = 2, \; b = 1 \tag{3.69}$$

Squaring is commonly used in statistics because squares can be more easily manipulated mathematically than absolute values. However, Legates and McCabe (1999) have noted that the use of squares tends to create an arbitrarily greater influence on the statistic by way of the larger values. Dawson and Wilby (2001) recognise that the MSE is suitable as a general measure of model performance, but is inadequate at identifying specific regions of model deficiency.

Other commonly used performance measures are derived by replacing the absolute difference between P_i and O_i in Equation 3.65, by the absolute relative difference (Equation 3.70). The underlying idea behind this performance measure is that larger absolute valued model predictions and measurements are permitted to have larger inherent errors (Van der Molen and Pintér, 1993). Karunanithi et al. (1994) suggest that squared errors (e.g. MSE) are more suitable at measuring the goodness of fit for peak values, while relative errors provide a more balanced representation of the goodness of fit for moderate values. Using an absolute relative difference gives rise to the following general performance measure:

$$E(w) = \left[\frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{(O_i - P_i(w))}{O_i} \right|^{\tau} \right]^{1/b}, \quad O_i \neq 0,\ 1 \leq \tau, b \leq \infty \tag{3.70}$$

In the water resources literature, the most common cases of Equation 3.70 include minimisation of the average absolute relative error, which is commonly expressed as a percentage error (AAPE) (Equation 3.71), and the mean squared relative error (MSRE) (Equation 3.72).

$$E_4 = 100 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{(O_i - P_i(w))}{O_i} \right|, \qquad O_i \neq 0,\ \tau = 1,\ b = 1 \tag{3.71}$$

$$E_5 = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{(O_i - P_i(w))}{O_i} \right|^2, \qquad O_i \neq 0,\ \tau = 2,\ b = 1 \tag{3.72}$$

Other performance measures that are commonly used in hydrological modelling are correlation-based measures, and include the coefficient of efficiency (CE) (Equation 3.73) and the coefficient of determination (r^2) (Equation 3.74) (Dawson and Wilby, 2001). It should be noted that unlike the general formulation given in Equation 3.64, both of these measures are to be maximised.

$$E_6 = 1 - \frac{\sum_{i=1}^{n}(O_i - P_i(w))^2}{\sum_{i=1}^{n}(O_i - \overline{O})^2} \tag{3.73}$$

$$E_7 = \left[\frac{\sum_{i=1}^{n}(O_i - \overline{O})(P_i(w) - \overline{P}(w))}{\sqrt{\sum_{i=1}^{n}(O_i - \overline{O})^2 \sum_{i=1}^{n}(P_i(w) - \overline{P}(w))^2}} \right]^2 \tag{3.74}$$

where $\overline{O}$ is the mean of the observed values and $\overline{P}$ is the mean of the predicted values.

The CE (also referred to as the determination coefficient, the efficiency index, F index and R^2) is the ratio of the MSE to the variance of the observed data, subtracted from unity. It measures the ability of a model to predict values, which are different from the mean. Legates and McCabe (1999) provided the following example to illustrate the functioning of the CE as a performance measure: If the square of the differences between the model predictions and the observations is as large as the variability in the observed data, then CE = 0.0. If it exceeds it, then CE < 0.0, since in this case, the mean is a better predictor than the model. If it is less than it, then CE > 0.0 up to a maximum of 1.0 for perfect correlation. A value of zero for the CE indicates that the model provides predictions that are the same as simply using the mean of the observed data, while negative values indicate that the model provides worse predictions than using the mean of the observed data. The CE ranges from -∞ for the worst case, to +1 for perfect correlation.

The coefficient of determination (r^2) is the square of the Pearson product-moment correlation coefficient and measures the proportion of variability in the observed data that is explained by the model (Legates and McCabe, 1999). Accordingly, it ranges from 0.0 to 1.0 with higher values indicating a better agreement between the modelled and observed values. Legates and McCabe (1999) point out that the main disadvantage of r^2 is that it is unable to account for additive and proportional differences between the modelled and observed data. In addition, they also illustrate that r^2, like other squared measures, is over-sensitive to outliers, which leads to a bias towards the extreme events. The CE performance measure also suffers from a sensitivity to outliers but it is an improvement over r^2 since it is able to account for additive and proportional differences between the modelled and observed data (i.e. it is sensitive to differences in the means and variances of the modelled and observed data). Both the CE and r^2 are

useful because they are measures that are independent of scale and hence, can be used for comparisons between different studies (Dawson and Wilby, 2001).

Two other performance measures used in this research are taken from the study conducted by Diskin and Simon (1977), who analysed twelve objective functions commonly used in hydrologic models. Diskin and Simon (1977) found that across a wide variety of watersheds, models and applications, the best objective function was a variant of the AAE (denoted as AAE2, Equation 3.75), which measures the sum of the absolute deviations divided by the sum of the observed values. Since the sum of observed values is a constant, this measure provides results equivalent to the AAE (Equation 3.66). Since the AAE is already being investigated in this research, there was no need to consider AAE2.

$$\text{AAE2} = \frac{\sum_{i=1}^{n} |O_i - P_i(w)|}{\sum_{i=1}^{n} O_i} \tag{3.75}$$

The second best function in their study was the sum of squared errors (SSE) (Equation 3.76), however, this provides equivalent results as the RMSE (Equation 3.67) or the MSE (Equation 3.69). Hence, in this research, the MSE and the SSE were not considered as potential performance measures, but rather, only the RMSE was used.

$$\text{SSE} = \sum_{i=1}^{n} (O_i - P_i(w))^2 \tag{3.76}$$

The third and fourth best objective functions in the study conducted by Diskin and Simon (1977) are shown in Equations 3.77 and 3.78, respectively. A general discussion outlining the properties of each of these functions is provided in Simon (1975).

$$E_8 = 2\left|\frac{(\overline{O} - \overline{P}(w))}{(s_O + s_P)}\right| + 2\left|\frac{(V_O - V_P)}{(V_O + V_P)}\right| + 2\left|\frac{(\overline{O} - \overline{P}(w))}{(\overline{O} + \overline{P}(w))}\right| \tag{3.77}$$

where s_O and s_P is the standard deviation of the observed values and the predicted values, respectively and V_O and V_P is the variance of the observed values and the predicted values, respectively.

$$E_9 = \frac{\sum_{i=1}^{n}\left[(O_i)^{1/2} - (P_i(w))^{1/2}\right]^2}{\sum_{i=1}^{n} O_i} \tag{3.78}$$

The final performance measure investigated in this research is the peak and low flow criterion (PLC) proposed by Coulibaly et al. (2001c). This criterion was formulated to help identify a more appropriate ANN model for forecasting peak and low flows. Such a model represents a trade-off between the different objectives of accurately forecasting the peak flow events and accurately forecasting the low flow events. Using a number of different input sets, Coulibaly et al. (2001c) found that using the PLC to develop an ANN produced a model that was better able to forecast the extreme hydrological events when compared to models that were developed using either RMSE or the CE as the performance measure. The PLC objective function is given by:

$$E_{10} = P_k \times L_k \tag{3.79}$$

where P_k is the peak flow criterion given by:

$$P_k = \frac{\left[\sum_{i=1}^{n_p}(O_{pi} - P_{pi}(w))^2 \cdot (O_{pi})^2\right]^{1/4}}{\left[\sum_{i=1}^{n_p}(O_{pi})^2\right]^{1/2}} \tag{3.80}$$

where n_p is the number of peak flows greater than one-third of the mean peak flow observed, and L_k is the low flow criterion, which is given by:

$$L_k = \frac{\left[\sum_{j=1}^{n_l}(O_{lj} - P_{lj}(w))^2 \cdot (O_{lj})^2\right]^{1/4}}{\left[\sum_{j=1}^{n_l}(O_{lj})^2\right]^{1/2}} \tag{3.81}$$

where n_l is the number of flows lower than one-third of the mean flow observed.

Despite Coulibaly et al. (2001c) developing this performance measure for the application of flow forecasting, it is apparent that this criterion can be generalised to other types of hydrological data, where it is also important to produce good forecasts for both the peak and low values of the time series. In this research, a modification was

made to Equation 3.79 so that the performance measure could be applied to the modelling of other hydrological variables, besides flows. In this modification, the peak criterion is restricted to values of the observed time series that are greater than the third quartile (75th percentile) and the low criterion is restricted to values of the observed time series that are less than the first quartile (25th percentile). In this case, n_p is the number of observed values greater than the third quartile of the observed data, and n_l is the number of observed values less than the first quartile of the observed data. This criterion was also found to be easier to implement.

The ten performance measures used in this research are summarised in Table 3.13. There are other measures that have been employed in a limited number of applications, however, many of these are variations of the measures reviewed above. For example, Zhang and Govindaraju (2000) used a measure they defined as a normalised mean square error (NMSE), which was the ratio of the MSE to the variance of the observed data (Equation 3.82).

$$\text{NMSE} = \frac{\frac{1}{n}\sum_{i=1}^{n}(O_i - P_i(w))^2}{\sigma^2} \tag{3.82}$$

where σ^2 is the variance of the observed data. Even a cursory examination of Equation 3.82 reveals similarities with the CE performance measure given in Equation 3.73. The NMSE is identical to the second term of Equation 3.73 (i.e. the component that is subtracted from unity). Tokar and Markus (2000) use a variation of the NMSE, which is the ratio of the RMSE to the standard deviation of the observed values. Another measure used by Atiya and Shaheen (1999) is the normalised RMSE (NRMSE), which is defined as ratio of the square root of the sum of squared errors to the square root of the sum of squared observed values (Equation 3.83).

$$\text{NRMSE} = \frac{\sqrt{\sum_{i=1}^{n}(O_i - P_i(w))^2}}{\sqrt{\sum_{i=1}^{n}(O_i)^2}} \tag{3.83}$$

Campolo et al. (1999) used the percentage of values exceeding the RMSE ($\%N_{RMSE}$) as a performance measure for modelling flow data. Other measures that have also been used in the literature include: the pooled mean squared error (PMSE) (Elshorbagy et al., 2000a); %MF, the percent error in modelled maximum flow relative to the observed data (Furundzic, 1998; Hsu et al., 1995); and, %VE, the percent error in modelled

runoff volume (Hsu et al., 1995). The $\%N_{RMSE}$ and PMSE were not considered to be sufficiently different from the other measures to justify their usage in this study. %MF and %VE are applicable to the modelling of flows, and as such, were not used for the case studies considered in this research.

Table 3.13 The Ten Performance Measures Investigated in this Study

Equation	Performance Measure	Abbreviation
E_1	Average absolute error	AAE
E_2	Root mean squared error	RMSE
E_3	Minimisation of the maximum discrepancy	Minimax
E_4	Average absolute percentage error	AAPE
E_5	Mean squared relative error	MSRE
E_6	Coefficient of efficiency	CE
E_7	Coefficient of determination	r^2
E_8	Diskin and Simon #1	DS1
E_9	Diskin and Simon #2	DS2
E_{10}	Peak and low flow criterion	PLC

3.7.1.3 Method

Dawson and Wilby (2001) recognise that once armed with a wide array of performance measures "... the problem then becomes deciding which (if any) are most appropriate to a particular application". Diskin and Simon (1977) have proposed a procedure in which a number of possible functions can be considered and compared and a suitable function selected for the application under investigation. Following Diskin and Simon (1977), the following procedure may be formulated in the context of selecting a suitable performance measure for an ANN model:

1. Define a number of performance measures and use each one in turn with the cross-validation technique to determine when to stop training an ANN model. In the cross-validation technique, an independent test set is used to assess the performance of the model at various stages of learning. Using different performance measures to assess the model's performance will result in training being stopped at different stages and hence, a different set of weights will be found using each performance measure.
2. The set of weights W_j obtained for a given ANN model are assumed to be near-optimal for the objective function E_j considered. Using these weights W_j, the values of other performance measures are computed for the test set and used as measures of model performance. These values are denoted by P_{jk} and the closeness of these values to the minimum value of another performance measure E_k indicates how

well one performance measure (E_k) can be represented by another performance measure (E_j).

3. Arranging the results obtained by the above steps in a tabular form produces a matrix with elements P_{jk}. The elements give the value of each function denoted by the index k, and are used as a measure of model effectiveness for all sets of model weights derived by use of other performance measures identified by the index j.
4. The matrix containing the actual values of the performance measures P_{jk} is transformed to another matrix in which ranks are assigned to the various performance measures instead of the actual values. The ranks R_{jk} are assigned according to the relative magnitude of the computed values of P_{jk} in each group of values identified by index k, starting with a value of 1 for the lowest value or in the case of correlation measures, a value of 1 for the highest value. The ranks assigned to the various performance measures obtained by the use of a set of weights $\overline{W}_j$ are thus integers indicating how well this set of weights could give rise to low values of other performance measures.
5. A set of optimal weights $\overline{W}_j$, which could produce low values of the ranks R_{jk} for a large number of other performance measures, may be considered to be superior to other sets of near-optimal weights. In this case, the performance measure E_j used in the derivation of this set of weights may, therefore, be adopted as the most suitable performance measure. The "best" performance measure of those considered can be defined and identified by adding up the ranks assigned to all functions according to each value of j (i.e. $\sum_j R_{jk}$), and choosing the function for which the sum has the lowest value.

In this research, the above procedure has been employed to determine a suitable performance measure for ANN modelling. It is important to note that the procedure must be repeated for each application and for each type of ANN model since the selection of an appropriate performance measure is both application- and model-dependent.

3.7.2 Choice of Optimisation Methods

3.7.2.1 Introduction

In ANN modelling, the optimisation method used for determining the weights within an ANN model is commonly referred to as the "training algorithm" or the "learning rule" (Cheng and Titterington, 1994). The selection of a suitable optimisation method is another important step in the model development process. For feedforward MLPs, the most commonly used optimisation method is the backpropagation algorithm. However,

there are a number of limitations associated with the backpropagation algorithm, which are primarily associated with its long training times, susceptibility to becoming stuck in local minima and the large number of internal model parameters used by the algorithm (Parisi et al., 1996). A number of alternative algorithms have been proposed in the literature, which attempt to overcome many of the difficulties associated with the standard backpropagation algorithm. Some of these algorithms make use of heuristics in an attempt to automatically adjust the learning rate and/or the momentum parameters (e.g. Jacobs, 1988; Minai and Williams, 1990), while others are based on second-order optimisation algorithms (e.g. Battiti, 1992; Fahlman, 1989; Le Cun, 1991). The use of second-order methods can significantly speed up the training process because they are based on a quadratic model, and as such, can achieve superlinear asymptotic convergence (Parisi et al., 1996).

In the application of ANN models to water resources modelling, the backpropagation algorithm is the most commonly used method of training (Maier and Dandy, 2000b). More recently, there have been a number of applications utilising alternative training algorithms. Lischeid (1998) investigated the use of the QuickProp, RProp and Backpercolation algorithms for training feedforward MLPs for water quality modelling. The RProp algorithm was found to yield superior performance, however, no comparative results were presented or discussed. The QuickProp algorithm has also been used in a number of other studies including, Karunanithi et al. (1994); Muttiah et al (1997); Maier and Dandy (1999); and Lekkas et al. (2001). Huang and Foo (2002) used the conjugate gradient algorithm to train a feedforward MLP to model salinity variations in the Apalachicola River, Florida. The conjugate gradient algorithm differs from normal gradient descent in the way the gradient is calculated and how the weights are subsequently updated. It is generally considered to be faster than standard backpropagation (Adeli and Hung, 1995). The conjugate gradient algorithm has also been used by Shamseldin (1997) who used the algorithm to train a feedforward MLP for rainfall-runoff modelling. Another algorithm that has been used to train feedforward MLPs is the Levenberg-Marquardt (LM) algorithm (Coulibaly et al., 2001b; Moatar et al., 1999). The LM algorithm is a hybrid optimisation method that combines the inverse Hessian method with gradient descent (Press et al., 1992).

Amongst the papers using alternative training algorithms, only a few of these also provided a comparative study. Huang and Foo (2002) compared the conjugate gradient algorithm with the standard backpropagation algorithm and found that the use of the conjugate gradient algorithm did not provide a significant improvement in the forecasting results but was several times faster to train. Thirumalaiah and Deo (1998a; 1998b; 2000) compared the conjugate gradient algorithm, the QuickProp algorithm and

the backpropagation algorithm for a number of hydrological forecasting applications. In general, each of the training algorithms was found to produce acceptable results, however, the results obtained were found to be very case study dependent, with different training algorithms performing better for different case studies. Kocjančič and Zupan (2000) compared backpropagation with the LM algorithm for modelling river flows and found that the best model was obtained when using the LM algorithm. Maier and Dandy (1999) investigated the relative performance of various first- and second-order algorithms for training feedforward MLPs. Three performance criteria were used in assessing model performance, including: generalisation ability, as measured by RMSE and AAPE; parsimony, measured by the number of hidden layer nodes; and, training speed, measured by the time taken to meet the stopping criteria. Maier and Dandy (1999) found that the generalisation ability of the first-order methods was better than that of the second-order methods investigated. In addition, the methods that automatically adjusted the size of the steps to be taken in weight space exhibited little variation in their results, regardless of the starting position in weight space. In their study, Maier and Dandy (1999) also noted the dependency of their results on the case study considered and the internal model parameters used. Therefore, the comparison of a number of alternative optimisation methods is an important step in the overall model development process and needs to be conducted for each study investigated.

In this research, the methodology outlined by Maier and Dandy (1999) has been used to compare a number of optimisation algorithms for training feedforward MLPs. It was not considered necessary to investigate alternative training algorithms for the single-sigma GRNN because of the simple nature of the optimisation problem. In Section 3.6.2.2 it was noted that if the multiple-sigma version of the GRNN has more than a few inputs, stochastic training algorithms are advantageous because of their ability to avoid becoming trapped in these local minima. To overcome this difficult optimisation problem, the use of a GA was proposed. An alternative and new stochastic optimisation algorithm within the family of evolutionary techniques, known as ant colony optimisation (ACO) (Dorigo and Di Caro, 1999) can also be used to train the multiple-sigma GRNN. In this research, the use of ACO to train a multiple-sigma GRNN is investigated as an alternative to the GA.

3.7.2.2 Methods

Feedforward MLPs

The NeuralWorks Professional II/Plus software supports the following methods for training feedforward MLPs: generalized delta (GD) rule; the normalized cumulative delta (NCD) rule; the delta-bar-delta (DBD) algorithm; the extended delta-bar-delta (EDBD) algorithm; the QuickProp (QProp) algorithm; and, the MaxProp (MProp) algorithm. A detailed description of the delta rule and the DBD and EDBD algorithms is provided in Section 2.4.1.1 and a brief description is provided below. A more detailed description of the QuickProp and MaxProp algorithms is also provided below.

Generalized Delta Rule:

The standard backpropagation uses the GD learning rule, which is a first-order method based on the method of steepest (or gradient) descent. Using larger step sizes in weight space during the initial stages of learning helps to speed up the training process and provides a mechanism for the network to escape local minima in the error surface (Maier and Dandy, 1999). As training progresses and the network converges to the local minimum in the error surface, the steps taken in weight space can be reduced to increase the likelihood of finding the local minimum. Therefore, in the NeuralWorks Professional II/Plus software, the default learning rates and momentum values were decreased as training progressed in accordance with the schedule shown in Table 3.14. The learn count in Table 3.14 refers to the number of training samples presented to the network. It is also important to note that when the GD rule is used, the weights are updated after the presentation of each training sample to the network.

Table 3.14 Default Values for Learning Rate (η) and Momentum (μ)

Layer	Parameter	Learn Count				
		0 - 10,000	10,001 - 30,000	30,001 - 70,000	70,000 - 150,000	150,001 +
Output	η	0.15	0.075	0.01875	0.00117	0
	μ	0.40	0.200	0.05000	0.00312	0.0001
Hidden 1	η	0.30	0.150	0.03750	0.00234	0.0001
	μ	0.40	0.200	0.05000	0.00312	0.0001
Hidden 2	η	0.20	0.100	0.02500	0.00156	0.0001
	μ	0.40	0.200	0.05000	0.00312	0.0001

Normalized Cumulative Delta Rule:

The NCD is a simple variant of the GD rule. The only difference is that the delta weight (or weight change) is accumulated over a number of training samples presented to the network. The number of training samples for which the accumulation occurs is defined by the epoch size. As the epoch size increases, the gradient tends to be "smoothed-out" and the search tends to focus in the direction of the true gradient (Maier and Dandy, 1999).

Delta-Bar-Delta Algorithm:

Jacobs (1988) developed the DBD algorithm to address the limitations of the NCD learning rule. In the DBD algorithm, each weight within the network is assigned a separate learning rate and these are adjusted in response to changes in the error surface. If the sign of the error surface gradient remains unchanged for several iterations, the learning rates are increased, since this is an indication that the minimum lies ahead. Conversely, if the sign of the error surface gradient changes for several consecutive iterations, then the learning rate is decreased as this indicates that the minimum is being jumped over. Despite the advantage of automatically tuning the learning rate parameter, the DBD algorithm does have the disadvantage of introducing a number of empirical parameters, which control the rate at which the learning rate adjustment occurs. However, in practice, the algorithm is not overly sensitive to the choice of these empirical parameters (Smith, 1993). The three empirical parameters used by the DBD algorithm θ, κ and ϕ are discussed in Section 2.4.1.1. In this research, the default parameters recommended by the makers of the NeuralWorks Professional II/Plus software were used: θ=0.7, κ=0.01 and ϕ=0.5.

Extended Delta-Bar-Delta:

The EDBD algorithm (Minai and Williams, 1990) extends the DBD algorithm by including a time varying momentum term for each weight in the network. In addition, the adjustment of both the learning rate and the momentum follows an exponentially decreasing function and an upper limit is placed on all learning rates and momentum values to prevent any erratic jumps and oscillations in weight space.

QuickProp Algorithm:

The first-order methods described above use the gradient (partial first derivative of the error with respect to the weights) to descend down the error surface until a minimum is found. Since the gradient is only known at one point at any given time, then theoretically, an infinite number of small steps are required to reach the minimum error. This results in the slow-convergence exhibited by first-order methods. To overcome this, Fahlman (1989) developed the QuickProp algorithm, which uses information about the curvature of the error surface in an attempt to take a large step towards the minimum error. The QuickProp algorithm achieves this by combining two traditional approaches (NeuralWare, 1997):

- The learning rate is adjusted dynamically (either globally or separately for each weight) by using heuristic information.
- An approximation of the second derivative of the error with respect to each weight (i.e. Hessian) is used. This avoids the computational expense of calculating the second derivative.

The QuickProp algorithm is loosely based on the inverse Hessian method but can be considered more of a heuristic method than a formal second-order method. In the implementation of the algorithm, two important assumptions have been made by Fahlman (1989). These include:

- The error versus weight curve can be approximated by a convex parabola (i.e. a parabola with a positive definite Hessian matrix).
- The change in the slope of the error curve is only affected by the weight whose next step is being calculated rather than all other weights that are changing at the same time.

To approximate the Hessian, the QuickProp algorithm makes use of the previous and current error gradients and the difference between the previous weight and the current weight (i.e. weight change). Using this information it is possible to make a large step toward the minimum of the parabola without the costly calculation of the second derivative. If however, the current error gradient is in the same direction and is the same magnitude or even greater than the previous error gradient, then the step that is computed may be infinite or such that the step is away from the minimum. To overcome this, Fahlman (1989) introduced a maximum growth factor μ, and if this is

the case, then the step size is computed as the product of the maximum growth factor and the previous weight change (i.e. $\mu\Delta w^{(t-1)}$).

Following Fahlman (1989), an outline of the QuickProp equations is presented below. In the QuickProp algorithm, each of the weights are updated independently, therefore, in the equations below, the weight indices have been omitted.

If the gradient is given by:

$$h^{(t)} = \frac{\partial E}{\partial w^{(t)}} \tag{3.84}$$

and the inverse Hessian step is given by:

$$\Delta q^{(t)} = \frac{\Delta w^{(t)} \cdot h^{(t)}}{h^{(t-1)} - h^{(t)}} \tag{3.85}$$

Then the QuickProp algorithm calculates the change in each weight using the following equation:

$$\Delta w^{(t)} = \varepsilon L^{(t)} + \alpha Q^{(t)} \tag{3.86}$$

where ε and α are the learning coefficients for the linear and quadratic terms, respectively,

$$L^{(t)} = \begin{cases} h^{(t)} & \text{if} \quad h^{(t)} \cdot h^{(t-1)} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.87}$$

and

$$Q^{(t)} = \begin{cases} \mu\Delta w^{(t-1)} & \text{if} \quad h^{(t)}\left[h^{(t)} - \left(\frac{\mu}{\mu+1}\right)h^{(t-1)}\right] \geq 0 \\ \Delta q^{(t)} & \text{otherwise} \end{cases} \tag{3.88}$$

In the weight update calculation, a weight decay coefficient, the previous value of the weight and the computed weight change are used as follows:

$$w^{(t)} = (1-\delta)w^{(t-1)} + \Delta w^{(t)} \tag{3.89}$$

where δ is the decay rate. If the weight is too small, it is clipped by setting it to 0. This is expressed mathematically by the following:

$$\text{if} \quad \left|w^{(t)}\right| < \kappa, \qquad w^{(t)} = 0 \tag{3.90}$$

where κ is the weight clipping threshold.

MaxProp Algorithm:

The MaxProp algorithm is a simple variant of the QuickProp algorithm. To compute the step taken in weight space it uses the maximum of the steps obtained from the QuickProp algorithm and the delta rule. The main advantage of the MaxProp algorithm is that it is able to improve convergence when the training data possess a large degree of noise (NeuralWare, 1997). Using the MaxProp algorithm, the weight change is given by:

$$\Delta w^{(t)} = \begin{cases} \alpha \cdot \mu \Delta w^{(t-1)} & \text{if} \quad h^{(t)}\left[h^{(t)} - \left(\dfrac{\mu}{\mu+1}\right)h^{(t-1)}\right] \geq 0 \\ sign\left(h^{(t)}\right) \cdot \max\left(\left|\alpha \cdot \Delta q^{(t)}\right|, \left|\varepsilon \cdot h^{(t)}\right|\right) & \text{if} \quad h^{(t)} \cdot h^{(t-1)} \geq 0 \\ \alpha \cdot \Delta q^{(t)} & \text{otherwise} \end{cases} \tag{3.91}$$

The weight update is then the same as the QuickProp algorithm (Equation 3.89) and also incorporates weight decay and weight clipping.

Following the study conducted by Maier and Dandy (1999), 30 different ANN models (each using a different set of initial weights) were trained for each of the optimisation methods investigated. This was considered necessary so that a fair comparison between the different optimisation methods could be obtained. The default epoch size of 16 was used for each of these trials. It has been suggested that the QuickProp and MaxProp algorithms work best if the epoch size is set equal to the training size, or in the case of large training sets, if an epoch size of 100 is used (NeuralWare, 1997). Therefore, an epoch size of 100 was also investigated in this study.

Multiple-Sigma General Regression Neural Network

Ant Colony Optimisation (ACO) was considered as an alternative stochastic training algorithm for multiple-sigma GRNNs. ACO algorithms are based on the concept of

artificial ants finding the shortest route from a nest to a food source by laying down pheromone trails. The formulation used in this research for applying ACO to the optimisation of a multiple-sigma GRNN is as follows:

The objective is to find a set of sigma weights that minimises a performance measure, such as the MSE, as given by:

$$g(\bar{\sigma}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{3.92}$$

where g is performance measure, $\bar{\sigma}$ is the vector of sigma weights, $\hat{x}_i$ are the n predicted values in the test set and x_i are the actual or desired values. If there are d inputs to the network, then the vector of sigma weights will also be d-dimensional and the total search space of $\bar{\sigma}$ is given by:

$$S = \left\{\sigma_j \mid \sigma_j^- \le \sigma_j \le \sigma_j^+; j = 1, 2, ..., d\right\} \tag{3.93}$$

where σ_j^- and σ_j^+ denote the lower and upper bounds of the parameter σ_j, respectively and are determined using *a priori* knowledge. The optimisation of sigma weights is a continuous problem and since ACO was originally developed for ordered problems, the search space S must be discretised. The interval $\left[\sigma_j^-, \sigma_j^+\right]$ of each smoothing parameter is divided into a number of b strata. Each stratum is represented by the middle of the stratum, which gives a total of $B = b^d$ permutations or possible pathways through the total space of sigma weights. A graphical representation of the sigma weight discretisation is shown in Figure 3.7.

By letting $\{\sigma_{ij};\ i = 1, \ldots, b_j$ and $j = 1, \ldots, d\}$ be the stratum i of parameter σ_j in the search space of S, it is now possible to apply an ACO algorithm. There are many variants of the original ACO algorithm, and in this study the *Max-Min* Ant System (MMAS) has been used because of its superior performance on benchmark problems (Stützle and Hoos, 2000).

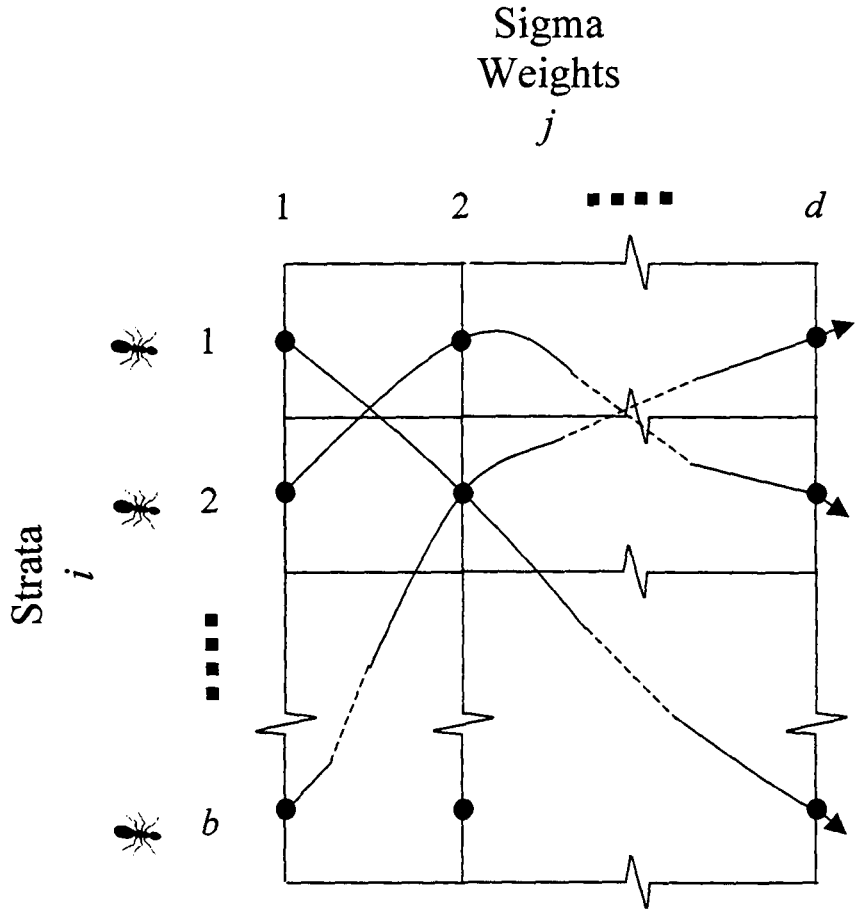

Figure 3.7 Schematic Representation of the Discretised Parameter Space

In MMAS, a population or colony of m artificial ants is used to traverse the discretised search space, making decisions at each decision point. A cycle or iteration of MMAS consists of three main steps: generation of m trial solutions by the traversal of the entire set of parameters by a colony of m ants, calculation of the objective function $g(\bar{\sigma})$ for each of the m trial solutions, and updating of the concentration of the pheromone trails. Each arc (i, j) of the search space is assigned a variable τ_{ij} for the artificial pheromone trail. The pheromone trail deposited on a particular arc is proportional to the fitness, as estimated by the ants, of using that arc to build a good set of sigma weights. The decision rule of ant k located in node i makes use of the pheromone trails τ_{ij} to compute the probability with which it should choose node $j \in N_i$ as the next node to move to, where N_i is the set of one-step neighbours of node i:

$$p_{ij}^{k}(t) = \tau_{ij}(t) / \sum_{j \in N_i} \tau_{ij}(t) \tag{3.94}$$

MMAS differs from many other ACO algorithms as it uses only a single ant to update the pheromone trails after each iteration. The pheromone trail update rule is given by:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{3.95}$$

where ρ (with $0 \leq \rho < 1$) is the trail persistence (thus, 1-ρ models the evaporation of pheromone), $\Delta\tau_{ij}^{best} = 1/g(\bar{\sigma}^{best})$ is the amount of pheromone laid down and $g(\bar{\sigma}^{best})$ denotes the solution cost of the iteration-best $(\bar{\sigma}^{ib})$ solution. Evaporating the pheromone helps avoid unlimited accumulation of pheromone trails. When an arc is not chosen by the ants, its pheromone trail decreases exponentially, thereby allowing the algorithm to "forget" bad choices over time. Using only the iteration-best ant to update the pheromone is a way of exploiting the best solutions during the search, however, to avoid search stagnation, pheromone trails on each arc need to be limited to an interval $[\tau_{min}, \tau_{max}]$. At the start of the algorithm, the pheromone trails are initialised to τ_{max} to promote higher exploitation of all solutions. In MMAS, the maximum pheromone trail τ_{max} is set to an estimate of the asymptotically maximum value:

$$\tau_{ij}^{\max}(t) = \left(\frac{1}{1-\rho}\right)\left(\frac{1}{g(\bar{\sigma}^{gb})}\right) \tag{3.96}$$

where $g(\bar{\sigma}^{gb})$ denotes the solution cost of the global-best $(\bar{\sigma}^{gb})$ solution. Stützle and Hoos (2000) used an empirically derived formula to determine the minimum pheromone trail τ_{min}, however, in this study τ_{min} was determined using trial-and error and this approach was found to give improved performance.

3.8 Model Deployment

3.8.1 Introduction

The final stage in the model development process is the deployment of the optimised model in an operational environment. This stage defines how the model is to be used in a real-world forecasting capacity and how periodic recalibration (if any) is to be performed so that the model is able to adapt in real-time to incorporate new information. Most hydrologic processes are dynamic and evolve with time and limited training data are generally available, so that not all possible patterns are contained in the training set. Therefore, recalibration of the ANN model is an important consideration. Despite the need for periodic updating, researchers seldom discuss the many practical issues associated with deployment of the trained model.

As discussed previously, a limitation of ANNs is that (like other empirical methods) they are unable to reliably extrapolate outside the range of the data used for training (Flood and Kartam, 1994). Accordingly, if the data used to train the ANN model are limited, it is very difficult to determine when the model will fail to generalise and to understand the range of applicability of the ANN model. It has been acknowledged in the past that an ANN is only as good as the data upon which it has been trained (Minns and Hall, 1996). Once the ANN model has been deployed in an operational sense, it is likely to perform poorly if faced with inputs that are far removed from the examples that it was presented with during training. This led the ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000b) to pose the following question, "... can we say when generalization will fail so that we understand the range of applicability of the ANN?" Once the model has been calibrated and deployed, this equates to knowing when the model is likely to fail and when the model needs to be recalibrated to incorporate new, uncharacteristic patterns that have not been used in training.

To improve generalisation ability beyond the calibration range, Imrie et al. (2000) added a guidance system to the original cascade correlation learning architecture used in their study and analysed the effect of using different output activation functions. The guidance system involved adjusting the cascade correlation algorithm to include cross-validation learning. It was found that the guidance system improved the results on a validation set and increased the maximum flow prediction. The use of a cubic polynomial as the output activation function was found to further enhance the capability of the ANN models to extrapolate beyond the calibration range. However,

no system was developed to detect uncharacteristic data, apart from comparing the maximum and minimum values in the training and validation sets.

Other real-time hydrological forecasting experiments have been investigated in the literature (see Coulibaly et al., 2000a; Coulibaly et al., 2000b; Thirumalaiah and Deo, 2000), however, to date no system has been developed to determine when the deployed model needs to be recalibrated. In this research, these three options were considered:

1. No recalibration;
2. Recalibration at some arbitrary time interval (e.g. at every time step new data become available); and,
3. Recalibration given some knowledge of when a pattern that is outside of the training domain is encountered.

Recently, Bowden et al. (2002) proposed a method for diagnosing uncharacteristic data patterns using a SOM. It was found that by combining the new data with the training data and clustering these data using the SOM, regions of poor performance could be identified by examining the resulting clusters. If the new data formed a cluster that did not contain training data, then these data were diagnosed as uncharacteristic. It was found that the ANN model performed poorly on these uncharacteristic data since it had not considered these events during training. This is because ANNs are exceptionally good at interpolation, but since the transfer function (usually sigmoid or hyperbolic tangent) saturates, extrapolation is unreliable. To determine when the ANN is extrapolating rather than interpolating, it is necessary to know what the distribution of the training data is, however, this can be rather difficult to determine with a large number of inputs. One way to address this problem is by plotting a histogram of each input in the training set, to see which values are most common and which values are rare or absent from the training set. But this is somewhat subjective and becomes increasingly difficult as the number of inputs increases. In the present study, two methods are used to diagnose when a new d-dimensional data pattern differs from all d-dimensional patterns in the training set, where d is the number of inputs. These are a SOM and a GRNN-based statistical test.

In the first approach, a hybrid model was developed consisting of two components. The first component is a SOM that combines each new input pattern with the data used for training and determines if the new pattern clusters within the training domain. The second component is a MLP trained using the backpropagation algorithm, which is used to perform the forecast. When a new input pattern is found to be uncharacteristic, there is a large degree of uncertainty associated with the corresponding forecast and

consequently, a warning is issued. This pattern is then placed in the training set. Once the corresponding output has been collected, the MLP is retrained with this pattern included. In this way, the ANN model is able to adapt to new information as it becomes available.

A second method has also been developed in this research to determine when retraining is required. This method involves using a GRNN model to detect outlying or uncharacteristic data patterns by performing a statistical test on the output of the GRNN's *B* summation unit. For uncharacteristic data, the output from the *B* summation unit will be small and a warning is issued if it is significantly smaller than the training set *B* summation unit outputs. If an uncharacteristic data pattern is detected, it is then added to the training set. Once the corresponding output value has been collected, the GRNN model is retrained.

In this research, three recalibration scenarios were investigated for both the SOM-MLP and GRNN models, including: (1) no recalibration; (2) recalibration after every new data sample is collected; and, (3) recalibration when an uncharacteristic pattern is detected by the SOM or GRNN procedure.

3.8.2 Methods

3.8.2.1 Hybrid SOM-MLP Model

The SOM implemented in this research consisted of a 20 by 20 Kohonen layer grid. There is no theoretical principle for determining the optimum size of the Kohonen layer grid (Cai et al., 1994), hence, the grid size was optimised by trial-and-error. During training, the learning rate used in the SOM decreased linearly from an initial value of 0.7 down to 0.01 using:

$$\alpha(i) = \max\left[\left(\alpha(0)(1-\frac{i}{rlen})\right), 0.01\right] \tag{3.97}$$

where $\alpha(i)$ is the learning rate at iteration i, $\alpha(0)$ is the initial learning rate and *rlen* is the running length of the training i.e. number of samples fed to the network. The neighbourhood size (N_c) was also a function of the training time and its size decayed linearly as training progressed, in accordance with:

$$N_c = \max\left[\text{int}\left(D(1-\frac{i}{rlen})\right), 1\right] \tag{3.98}$$

where D is the maximum dimension of the Kohonen layer's columns and rows i.e. D=max(*column dimension, row dimension*). The SOM was trained for a total of 300 epochs. The MLP used in the hybrid model was the optimised forecasting model developed using the preceding model development steps.

The hybrid model was developed by combining the SOM, which determines when an uncharacteristic input pattern is encountered, with the MLP, which performs the forecasting (Figure 3.8). After the SOM clusters the data, the proposed hybrid model diagnoses each new pattern as either characteristic or uncharacteristic depending on the presence or absence of training data in the new sample's cluster.

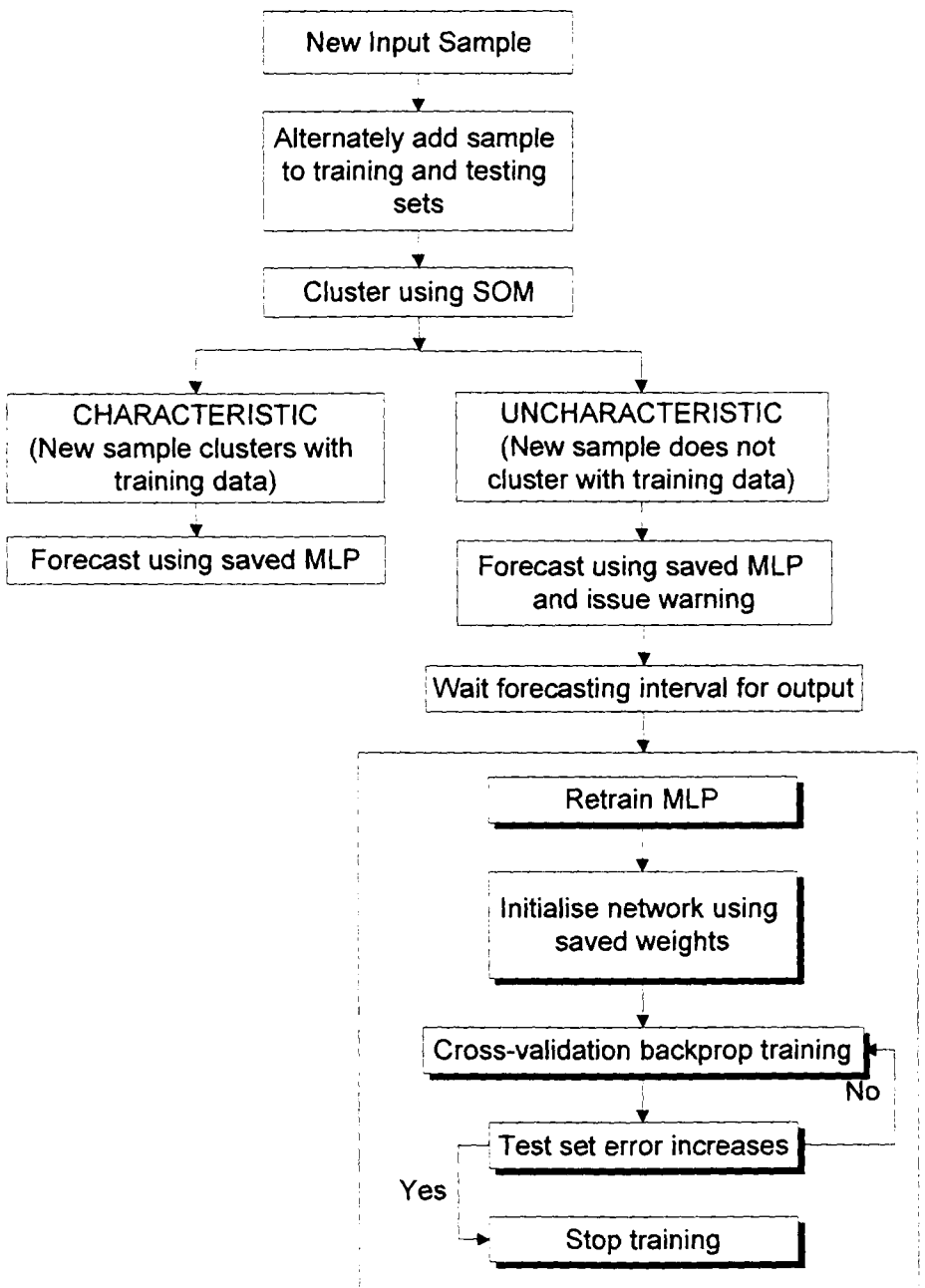

Figure 3.8 Hybrid SOM-BPN model

To help ensure that the training and test sets remain statistically representative of the same population, new samples are alternately placed in each set. When a new sample is added to the training set, it is then clustered using the SOM component of the hybrid model. If the new sample is found to be uncharacteristic, it is necessary to wait for a period of time equal to the forecasting interval for the corresponding output to become available, before the MLP can be retrained. However, if the new sample is found to be characteristic data, the currently saved MLP is used to obtain the forecast. When the MLP is retrained, the weights are initialised by using the saved weights of the previous model. This was found to significantly decrease the time needed to retrain the MLP. The software required to implement the hybrid SOM-MLP real-time forecasting model was developed in Fortran.

3.8.2.2 GRNN Outlier Detector

This method is similar in spirit to the hybrid SOM-MLP outlined above, however, rather than use a SOM to diagnose outlying data, a statistical test is performed on the GRNN's B summation unit output. In addition, a GRNN rather than a MLP is used to perform the forecast. The single-sigma GRNN was used in this research because of the speed at which it can be trained. The detection and retraining procedure adopted is shown in Figure 3.9.

The GRNN method of detecting outlying patterns is similar to the method used by Tax and Duin (1998), who estimated the input data density and rejected the objects in low probability areas. To perform the density estimation, one approach they used was Parzen kernel estimation. In this approach there are as many kernels as there are training patterns and the density of each training sample is estimated using:

$$\hat{f}(X) = \frac{1}{2\pi^{d/2}\sigma^{d}} \cdot \frac{1}{n}\sum_{i=1}^{n} \exp\left[-\frac{(X - X^{i})^{T}(X - X^{i})}{2\sigma^{2}}\right] \tag{3.99}$$

where n is the number of patterns in the training set, d is the number of inputs and σ is the smoothing parameter or width of the kernels. Once the densities had been estimated, Tax and Duin (1998) rejected uncertain patterns by using a rejection threshold of 1% (i.e. patterns with a probability of 1% or less were rejected). This was performed by assuming the measured quantity on the patterns is Gaussian distributed. The densities for the new patterns were then calculated and compared to those of the training patterns. If the difference between the new pattern and the mean of the training patterns was larger than three times the standard deviation of the distribution of the

training data, the new pattern was rejected. This approach was also adopted for the GRNN outlier detector. Since the GRNN equations are based on Parzen density estimation (see Section 2.4.1.3), the output from the B summation unit is given by:

$$\hat{f}(\boldsymbol{X}) = \sum_{i=1}^{N} \exp\left[-\frac{(\boldsymbol{X} - \boldsymbol{X}^i)^T (\boldsymbol{X} - \boldsymbol{X}^i)}{2\sigma^2} \right] \tag{3.100}$$

By simply rescaling this quantity by $\frac{1}{2\pi^{d/2}\sigma^d} \cdot \frac{1}{N}$, the Parzen densities in Equation 3.99 can be obtained. Therefore, the output from the B summation units can be tested for the occurrence of an outlier using the same statistical test described by Tax and Duin (1998), which results in the following:

Reject $\boldsymbol{X}$ if:

$$\log(\hat{f}(\boldsymbol{X})) < \varepsilon\left[\log(\hat{f}(\boldsymbol{X}^{tr}))\right] - 3.0 \cdot \text{stdev}\left[\log(\hat{f}(\boldsymbol{X}^{tr}))\right] \tag{3.101}$$

where $\boldsymbol{X}^{tr}$ are the training set samples. The log of the densities is taken since the probabilities can differ by orders of magnitude and by taking the log, the values come more or less in the same scale. When a new sample is encountered, the density is calculated using the output from the B summation unit and this is compared to the densities obtained for the training data using the above statistical test. If the new sample is found to be uncharacteristic, a forecast is still performed, however, a warning is issued to indicate that there is a large degree of uncertainty associated with the forecast. Before retraining the GRNN model to incorporate this new sample, it is necessary to wait for a period of time equal to the forecasting interval so that the corresponding output can be obtained. For example, if the forecasting interval is 14 days, then the corresponding output will not be sampled until 14 days have passed. Hence, the model cannot be retrained until this output has been sampled. This was enforced in the simulation to reflect the practical realities of the model deployment process. The software required for the GRNN outlier detector was developed in Fortran.

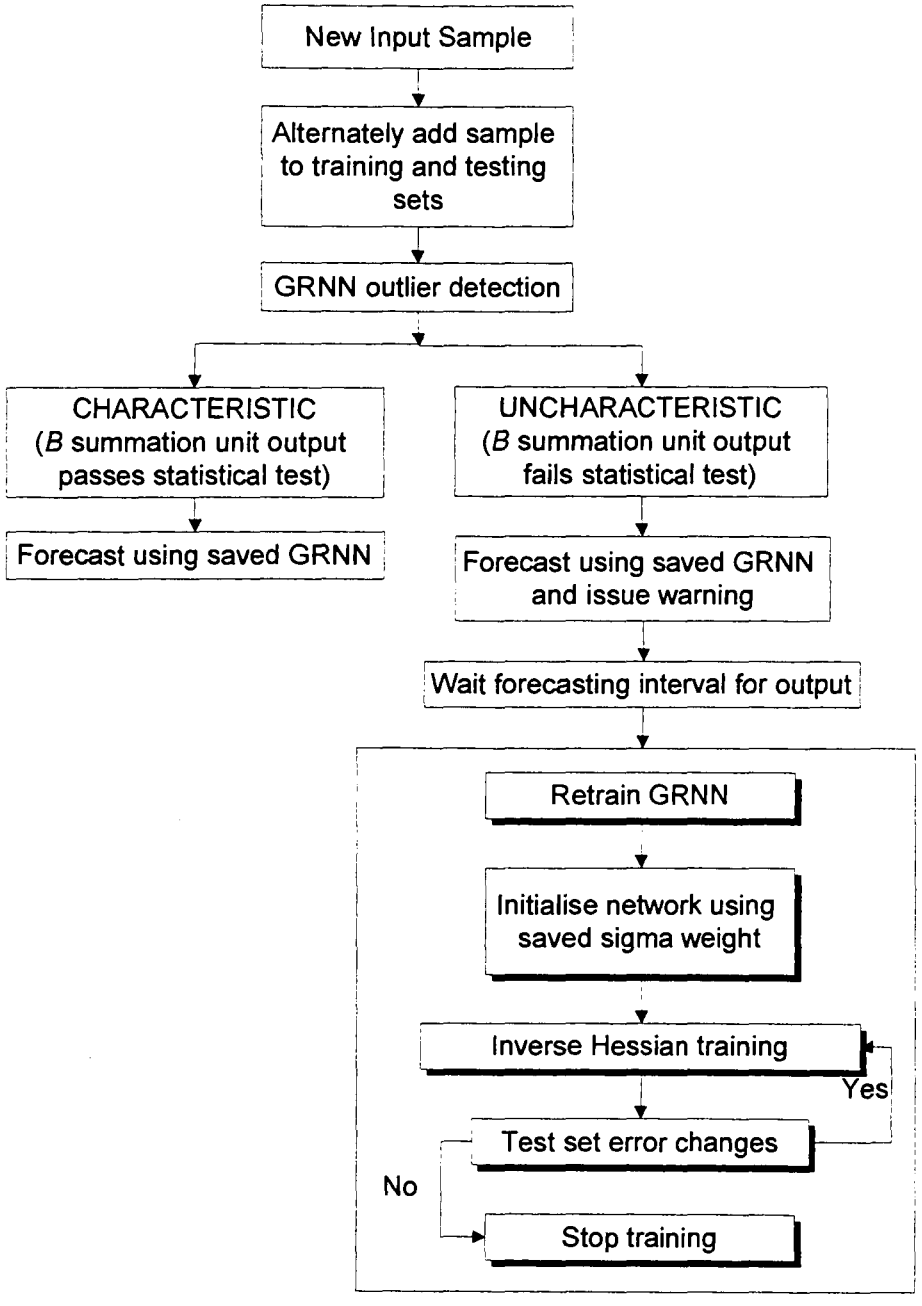

Figure 3.9 GRNN Outlier Detector

Chapter 4

Case Study I: Salinity in the River Murray

4.1 Introduction

Human induced salinisation is an age-old problem affecting many irrigated areas in the world's arid and semi-arid countries (Ghassemi et al., 1995). One such region is the South Australian portion of the Murray-Darling Basin, which has seen a dramatic increase in salinity since the land use changes resulting from European settlement. The widespread clearance of native vegetation combined with inefficient irrigation practices and flow regulation has had a negative impact on the water quality of the River Murray. This has had a deleterious effect on those who rely on this resource for a variety of domestic, industrial and agricultural uses. A recent audit conducted by the Murray-Darling Basin Commission (1999), determined that the average salinity of the lower River Murray is likely to exceed the 800 electrical conductivity (EC) units threshold for desirable drinking water in the next 50-100 years. In addition, by the year 2020 the probability of exceeding this threshold will be approximately 50%.

Adelaide is the capital city of South Australia and has a population just over 1 million. In terms of the ratio of rainfall to runoff, South Australia is often referred to as the driest state in the driest inhabited continent on earth. Since there are only a few alternative sources of water supply, 80% of South Australia's urban population are dependent upon the River Murray as a source of water. In Adelaide, users are reliant on the River Murray for about 50% of the total water supply on average with this figure rising to 90% during times of drought (Department for Water Resources, 2001). The River Murray lies approximately 80 km east of Adelaide and water is pumped to the city via two pipelines. The high salinity of this water has increased the corrosion of pipes and fittings, increased the usage of soaps and detergents and resulted in soil salinisation and a consequent reduction in crop yield (Maier, 1995). High salinity levels have been estimated to cost urban and industrial users in Adelaide AUD$30 million per year (Dwyer Leslie Pty Ltd., 1984).

In this chapter, the ANN methodology outlined in Chapter 3 was applied to the development of a salinity forecasting model at Murray Bridge, South Australia. Water is abstracted at this location and pumped to Adelaide via the Murray Bridge-Onkaparinga pipeline. Given the ability to forecast salinity, it is possible that more water could be pumped at times of low salinity and less water pumped during times of higher salinity. Dandy and Crawley (1992) developed a model for optimising pumping policies taking salinity into account. In their study, it was shown that the average salinity of the water supplied to Adelaide could be reduced by 10% if salinity forecasts several weeks in advance were available and pumping policies were modified accordingly.

Maier and Dandy (1996b) have previously developed ANN models for the same case study. In their study, ANNs trained using the backpropagation algorithm were found to be a useful tool for forecasting salinity in the River Murray at Murray Bridge. The forecasting period selected in the study was 14 days, as this is the minimum forecasting length required to enable changes to be made to the pumping schedule. Various input subsets were tested and the best set of inputs was identified using sensitivity analyses in conjunction with *a priori* knowledge to reduce the total number of model inputs. The ANN models were not found to be sensitive to different learning rates and different network architecture. The AAPE and the RMSE of the 14-day salinity forecasts averaged over 4 years (1988-1991) were 6.4% and 46.1 EC units, respectively.

The primary objective of this chapter is to use the salinity case study to illustrate and explore the following questions relating to the ANN methodology discussed in Chapter 3:

- Does the time series being modelled (i.e. salinity at Murray Bridge) possess nonlinear serial dependence and therefore, warrant the use of ANNs?
- How should the available data be divided into subsets and how does this affect the ANN's performance?
- Is it possible to diagnose when an ANN model will perform poorly based on the training data?
- Is it better to use a linear transformation of the data or should the data be transformed to normality before training a network?
- Does a logarithmic transformation of the data improve model performance?
- If the ANN model inputs are transformed to uniformity, does this allow for a better mapping to the output?
- Should the deterministic seasonality be removed from the data or is it better to use the raw data in training?
- What is the most suitable method for determining the inputs to ANN models?
- Is a feedforward MLP's performance improved by using two hidden layers?
- Is a GA a suitable method for optimising the architecture of a MLP?
- How do the forecasts obtained by a GRNN model compare to those obtained by a feedforward MLP trained using backpropagation?
- When a GRNN is used, should the training data be clustered to reduce the number of pattern layer nodes?
- Is it better to use a single- or a multiple-sigma GRNN?
- What performance measure should be used to assess the ANN models developed?
- How do different optimisation algorithms affect the performance of a MLP?
- What is the effect of using ant colony optimisation to train a multiple-sigma GRNN?
- Once the optimised ANN model is deployed in an operational environment, how should the model be retrained?

A secondary objective of this research is to develop a model aimed at producing the best possible forecasts of salinity in the River Murray at Murray Bridge. This model can be used to optimise pumping schedules and also to provide accurate advice to irrigators.

4.2 Background

4.2.1 The Murray-Darling Basin and the River Murray

The Murray-Darling river system has a catchment area of 1.073 million km^2, or nearly 14% of Australia's land area (Figure 4.1). It is the fourth longest river system in the world, but has an annual flow that is low by world standards. Approximately 57% of the Murray-Darling Basin is in New South Wales, 25% is in Queensland, 12% is in Victoria and 6% is in South Australia. About half of Australia's gross primary production comes from the land and water resources of the Basin and this production is valued at some AUD$10 billion annually (Murray-Darling Basin Ministerial Council, 1989). The Basin includes 20 major rivers, hundreds of smaller tributaries and a number of large water supply reservoirs and irrigation dams in south-eastern Australia (Walker, 1986).

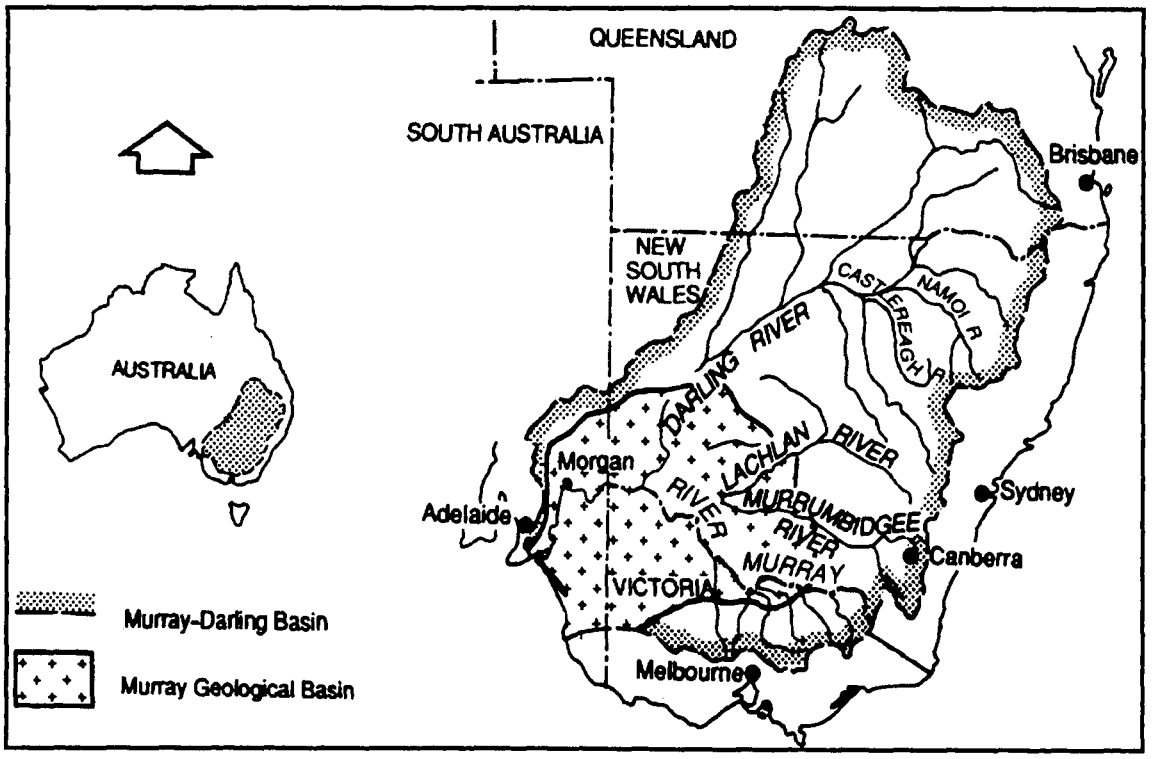

Figure 4.1 The Murray-Darling Basin in South-East Australia (source: Murray-Darling Basin Ministerial Council, 1988)

The headwaters of the River Murray are located in the Australian Alps near Mt. Kosciusko, New South Wales. The mean annual rainfall in this region is about 1500 mm (Ghassemi et al., 1995). The high rainfall combined with snowmelts ensures that flow is relatively rapid in the upper reaches of the Murray. Further downstream, the flow slows quite considerably as the river broadens and begins to meander. The

decrease in flow is also exacerbated by a significant decline in rainfall in the lower reaches. In the South Australian portion of the Basin, the mean annual rainfall is 250 mm (Ghassemi et al., 1995). By the time the Murray meets the sea near Goolwa, South Australia, the water has traveled about 2500 km.

Near Wentworth in New South Wales, the other major river in the basin, the Darling River, joins with the Murray. The waters of the Darling River are considerably more turbid than those originating in the upper reaches of the Murray. Consequently, at the Murray-Darling confluence there is a visible distinction between the two waters. The Darling water carries large quantities of montmorillonite clay, which is very fine and approaches a colloidal suspension (Shafron et al., 1990). Under "normal" flow conditions, these particles do not settle out and are carried along the length of the Murray in South Australia.

Prior to river regulation, the capricious nature of the River Murray meant that it was a highly variable resource for those that relied upon it for water supply. Commencing in 1919, a number of river regulation structures were built to overcome the effects of water shortages during droughts (Figure 4.2). Currently, the River Murray is still intensely regulated by a system of 14 locks and weirs, 5 barrages and 4 major storages. The locks and weirs were constructed to improve the use of the river for both navigation and water supply purposes. The barrages were constructed to prevent seawater from travelling upstream during periods of low flow. The storages were built as drought protection initiatives, with the aim of securing water during periods of high flow for release at times of low flow. There are three major storages on the River Murray, and these include: Lake Victoria (680 GL), Hume Reservoir (3040 GL) and Dartmouth Reservoir (4000 GL). The fourth major storage is Menindee Lakes (1680 GL), which is located on the Darling River (Maier, 1995).

The overall aim of regulating the Murray has been to meet the water requirements of the stakeholding states (i.e. New South Wales, Victoria and South Australia), while ensuring that there is minimum wastage. A water sharing agreement between the three states determines how the River Murray's water is divided between them. However, if there is a declared drought, the three states share the water equally. The main features of this agreement include (Murray-Darling Basin Commission, 1990):

- New South Wales and Victoria have equal share of the flow at Albury (which is just downstream of Hume reservoir, Figure 4.2).
- Victoria and New South Wales have control of their tributaries below Albury.

- Victoria and New South Wales provide South Australia with a guaranteed entitlement flow. This minimum entitlement flow ensures that Adelaide has a guaranteed water supply, even during dry periods. The entitlement flows to South Australia vary on a monthly schedule and are shown in Table 4.1.

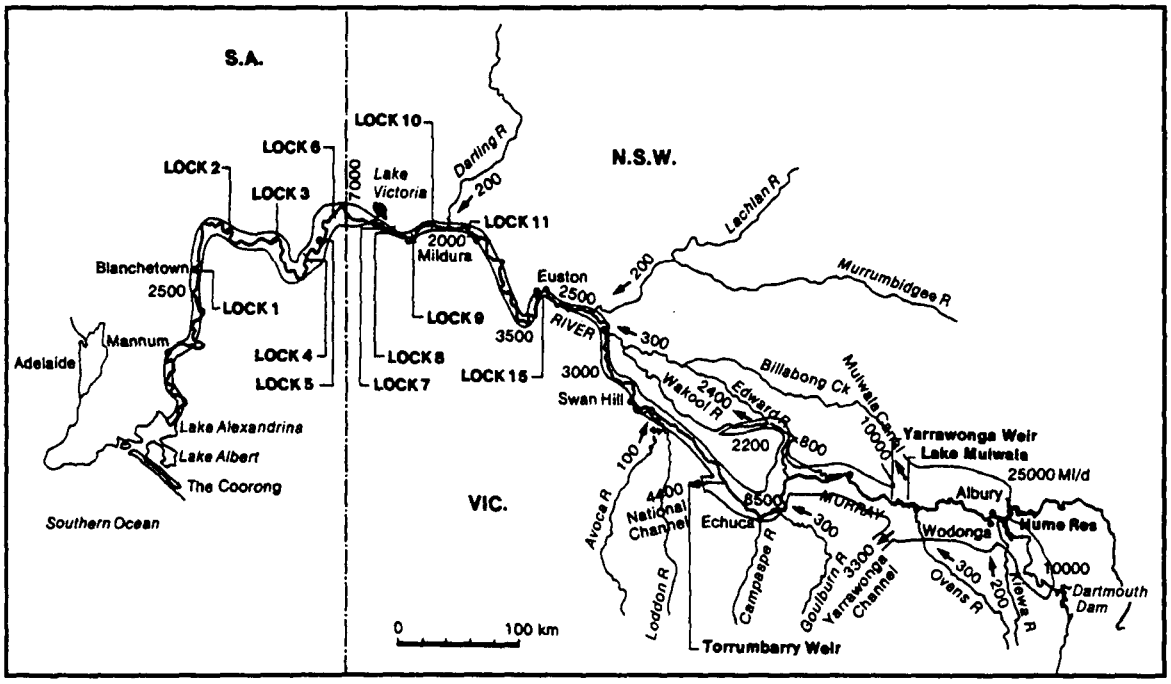

Figure 4.2 River Murray System – Regulation Structures and Typical Regulated Flows During Peak Irrigation Periods (ML/day) (source: Murray-Darling Basin Commission, 1990)

Table 4.1 Entitlement Flows to South Australia

Month	Entitlement Flow (ML/day)
January	7,000
February	6,900
March	6,000
April	4,500
May	3,000
June	3,000
July	3,500
August	4,000
September	4,500
October	5,500
November	6,000
December	7,000

A number of pipelines have been constructed in South Australia, which abstract water at various locations along the River Murray (Figure 4.3). These pipelines provide water for Adelaide, Port Pirie, Whyalla, Port Augusta and smaller country towns on the Yorke Peninsula, and in the central and south-east regions of South Australia. The majority of Adelaide's water supply is derived from catchments in the Mt. Lofty Ranges, however, a significant amount of its total water supply comes via the Murray Bridge-Onkaparinga and the Mannum-Adelaide pipelines. The water abstracted from the Murray is prone to high levels of salinity. The World Health Organisation's (1984) maximum desirable salinity level for human consumption is 500 mg/L (800 EC units). In comparison, average salinity levels in the River Murray at Morgan are currently 570 EC units and are set to increase to 790 EC units in 50 years and 900 EC units in 100 years (Murray-Darling Basin Commission, 1999).

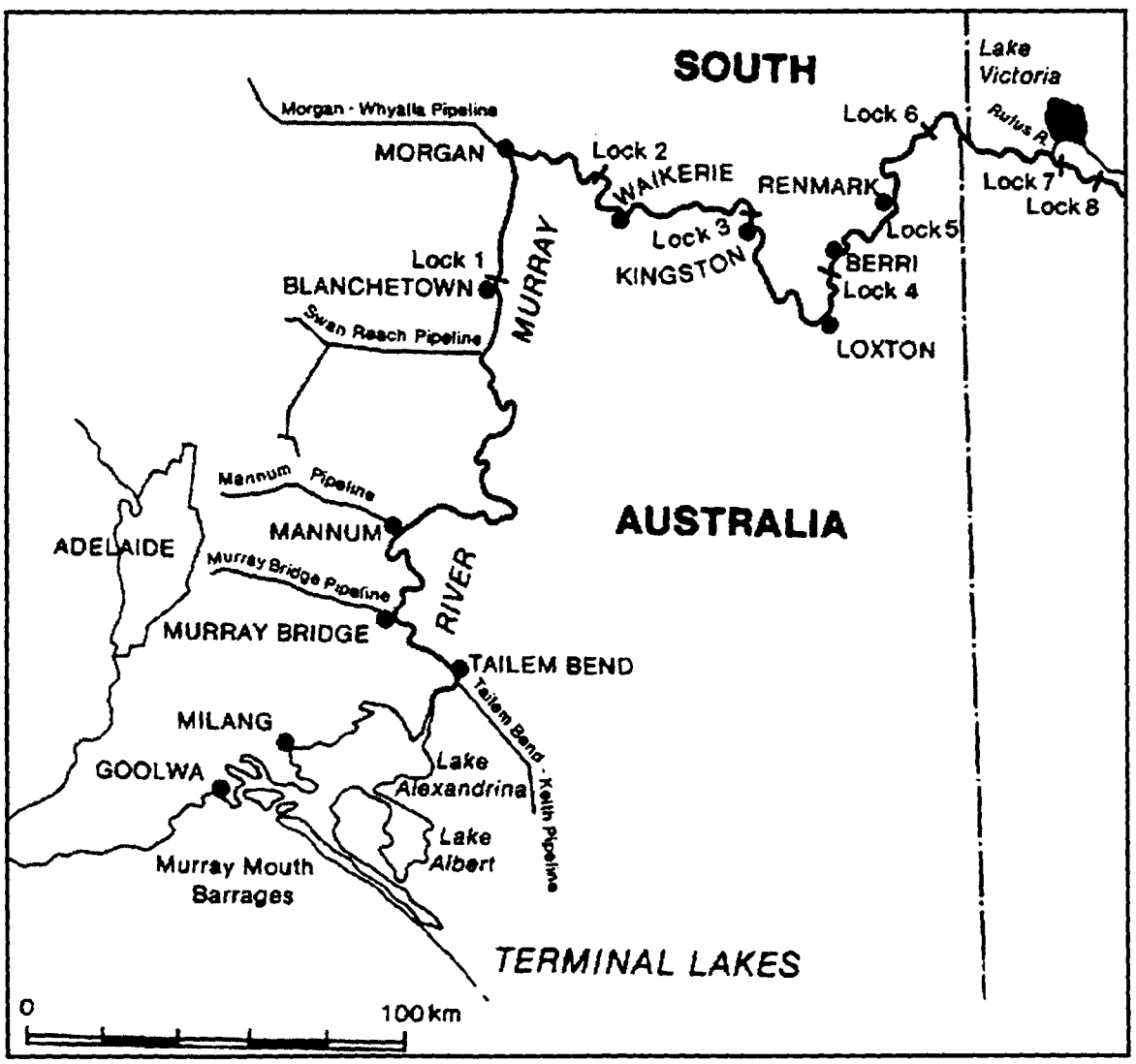

Figure 4.3 The South Australian Reaches of the River Murray and the River Murray Pipelines (source: Murray-Darling Basin Commission, 1990)

4.2.2 Salinity in the River Murray

The Murray-Darling Basin is currently facing a major salinisation problem. Since European settlement, large areas of the Basin have been cleared of native vegetation to make way for agriculture. In addition, large-scale irrigation has been established and the river system has been controlled and modified. All of these factors have contributed to a rise in the groundwater table, thereby bringing natural salt to the land surface and to the rivers. In 1987, it was estimated that 96,000 hectares of the Basin's irrigated land were salt affected and 560,000 hectares had water tables within 2 metres of the land surface (Murray-Darling Basin Commission, 1999).

Evans (1989) estimated that approximately 30% of the current salt load in the River Murray is human induced. This level is currently increasing due to a number of factors, including (Ghassemi et al., 1995):

- Large increases in the amount of water that is diverted from the rivers and streams in the Murray-Darling Basin. The diversion is currently averaging 9500 GL per annum.
- Increased drainage flows from irrigated areas, which are becoming increasingly saline due to rising water tables.
- Increased groundwater flow directly into the River Murray and its tributaries.

Regional groundwater systems within the Murray-Darling Basin drain naturally into the River Murray. The salinities of the groundwater in the Basin are very high, and in some cases exceed that of seawater (Figure 4.4). Prior to European settlement, this was the main source of salt entering the lower reaches of the River Murray. Since European settlement, the natural process of salt entering the river has been increased. Deep-rooted native vegetation has been replaced by shallow rooted crops and pastures, which have less capacity to make use of the rainfall that infiltrates the soil (Department for Water Resources, 2001). In addition, excess irrigation water is often applied to flush salts beyond the root zone. This has resulted in a gradual filling of shallow aquifers and the development of groundwater mounds, which eventually increases the discharge into the river via groundwater accessions. It is anticipated that impacts associated with the increased recharge of highly saline aquifers may take centuries to reach equilibrium, and the initial impacts are still yet to be felt (Department for Water Resources, 2001). The salinity audit conducted by the Murray-Darling Basin Commission (1999) has determined that the annual movement of salt within the Basin will double in the next 100 years with average river salinities set to rise significantly.

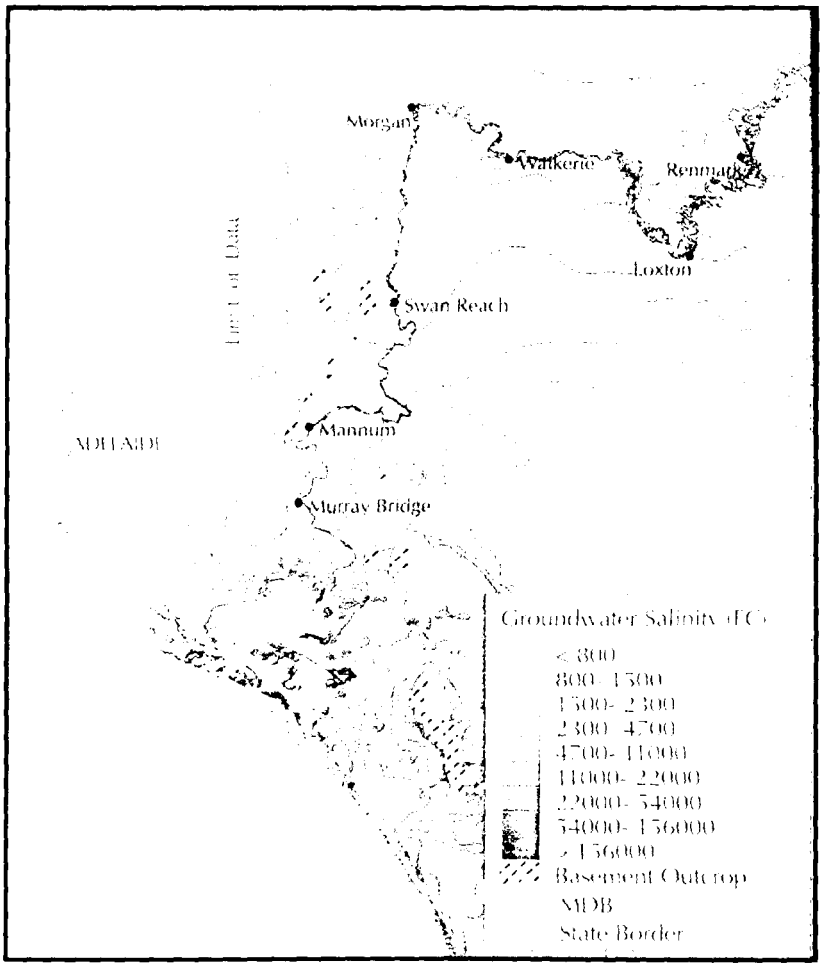

Figure 4.4 Groundwater Salinity in the South Australian Reaches of the River Murray (source: Department for Water Resources, 2001)

Saline groundwater accessions have also been exacerbated by river structures such as locks and weirs (Ghassemi et al., 1995). Such structures increase the water level by several metres and the resulting hydrostatic pressure is enough to force saline groundwater into the river on the downstream side of the river structure.

The impact of saline accessions on the river is dependent on the salt load entering the river and the capacity of the river to dilute that load. Obviously, the capacity to dilute the salt load is directly dependent upon the river's flow. Consequently, there is an inverse relationship between river flow and the level of salinity present in the river: the lower the flow the higher the salinity and *vice versa*. Salinity concentration (mg/L) is equal to the salt load (tonnes/day) divided by the flow (GL/day). In the River Murray, salinity is usually measured and reported in electrical conductivity (EC) units. Salts dissociate into charged ions in water and the electrical conductivity of the solution increases in proportion to the concentration of charged ions present in solution. Therefore, measuring the electrical conductivity is a convenient way of estimating the

salinity level (Murray-Darling Basin Commission, 1999). EC units are defined as micro Siemens / cm at 25° Celsius. The conversion factor between EC units and salinity or total dissolved solids (mg/L) depends on the exact ionic composition of the water. A typical conversion factor for the River Murray is 0.6, i.e. TDS (mg/L) = (0.6) EC units (Murray-Darling Basin Commission, 1990).

Due to the increases in human-induced salinisation, the lower reaches of the Murray have experienced rising salinity levels for many decades. In the headwaters, the annual average salinity is less than 40 EC units, but by the time this water reaches Morgan in South Australia, the average has risen to 618 EC units due to saline accessions (Figure 4.5).

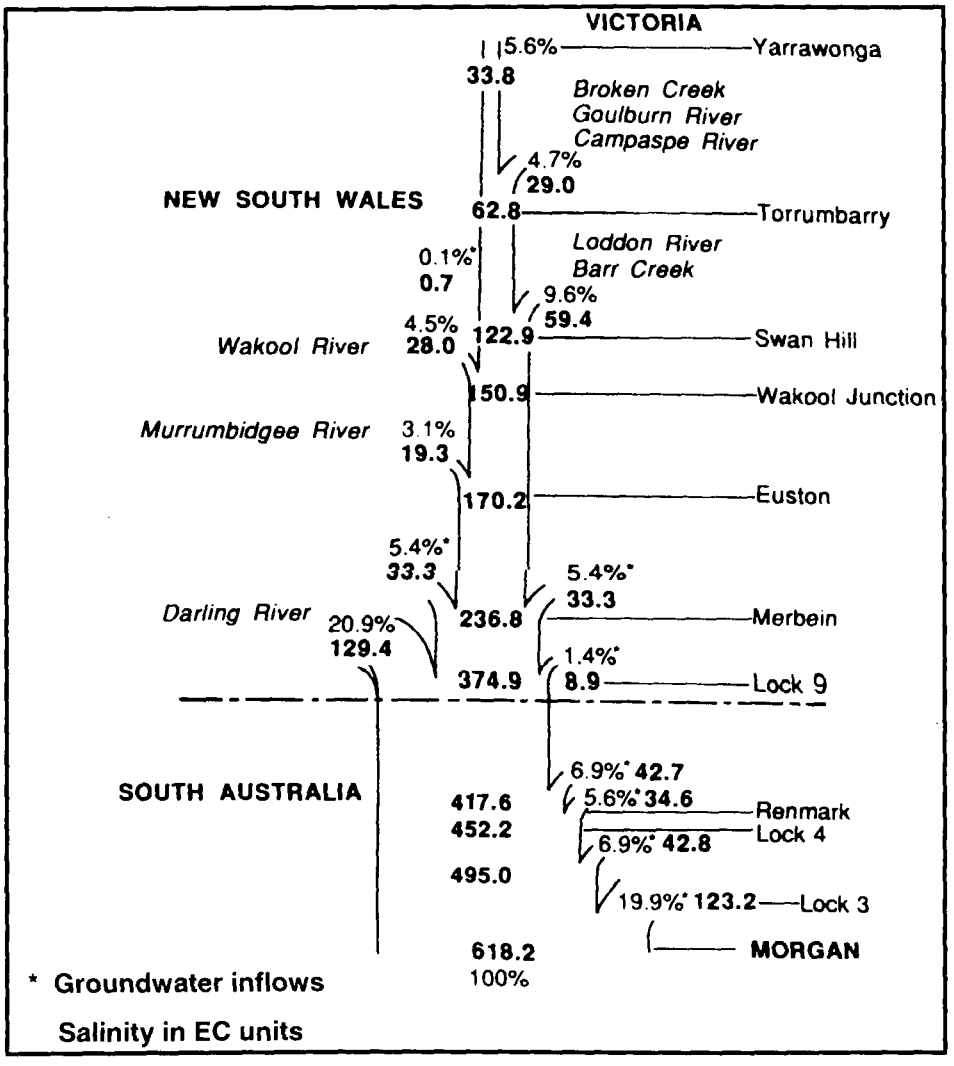

Figure 4.5 Contribution of Each Reach of the River Murray to the Average Salinity at Morgan in EC Units and in Percentage (source: Ghassemi et al., 1995)

The two most common impacts associated with salinity are those that result from saline water being used for agricultural, urban and industrial uses and those that result from

saline groundwater tables rising under land both in urban and rural areas. The following impacts can be expected as salinity levels rise (Department for Water Resources, 2001):

- If the salinity level reaches 650 EC units, damage starts to occur to irrigated crops.
- Once levels reach 800 EC units, water used for drinking purposes becomes increasingly unpalatable and scale problems start occurring in water supply infrastructure.
- At levels greater than 1500 EC units, consumptive uses of the water are severely restricted. Irrigation of a large majority of horticultural crops, leguminous pastures and forage crops is not possible. Ecosystems are adversely affected with lethal impacts on many aquatic macrophytes and some riparian vegetation. In addition, adverse impacts occur on many invertebrate species.

In 1989, the Murray-Darling Basin Ministerial Council (1989) put in place a Salinity and Drainage Strategy with the aim of providing a framework for managing salinity in the River Murray and land salinisation. Under the Strategy, the amount of salt entering the river was reduced through the construction of salt-interception schemes and through the implementation of salinity and land and water management plans by the various state governments (Murray-Darling Basin Commission, 1999). However, the positive effects achieved by the Strategy are likely to be overtaken unless new remedial actions are taken. Figure 4.6 shows the predicted increases in River Murray salinity if no further intervention is taken. It can be seen that salinities in the upper reaches are relatively low but increase rapidly in the lower reaches, clearly indicating the impacts that have resulted from irrigation, river regulation and natural saline groundwater inflows along the Murray in South Australia. At present, the average salinity is about 600 EC units by the time the water has reached Murray Bridge. With no new remedial action taking place, this is predicted to increase to about 960 EC units by the year 2100.

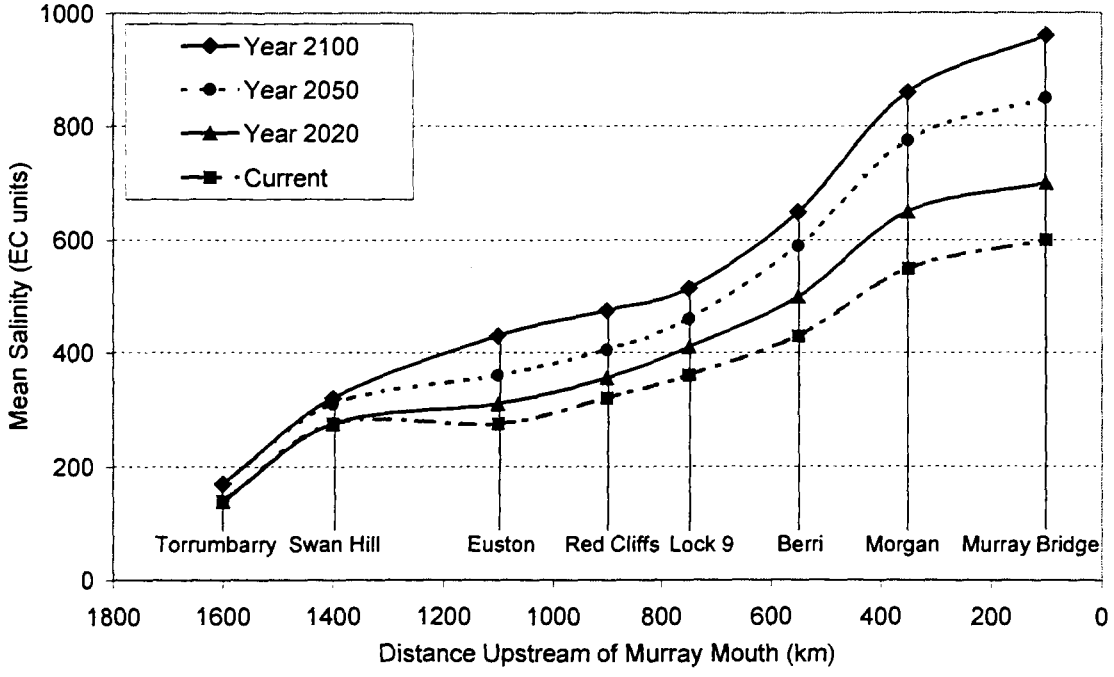

Figure 4.6 Average Salinity Levels for Current Conditions, 2020, 2050 and 2100 at Various Stations Along the River Murray Assuming That There is No Intervention (source: Department for Water Resources, 2001)

4.2.3 Forecasting Salinity at Murray Bridge

The case study used in this chapter involves the development of a salinity forecasting model for the River Murray. Murray Bridge (Figure 4.3) was selected as the modelling location since a significant proportion of Adelaide's water supply is abstracted from the River Murray at this point. If salinity can be forecast accurately several weeks in advance, it is then possible to make appropriate adjustments to pumping regimes in order to take into account the salinity of the water. Dandy and Crawley (1992) have developed an optimisation model which identifies pumping policies that minimise the total system costs, including the economic costs of salinity. By knowing the exact salinity several weeks into the future, it was found that the average salinity of the water supplied to Adelaide could be reduced by 75 EC units. However, it was found that altering the pumping policies to take salinity into account increased the average pumping cost by around AUD$2.0 million per year. This resulted in an estimated reduction in damage costs to domestic and industrial consumers of AUD$6.0 million. Therefore, the benefit-cost ratio of altering the pumping operating policies was estimated to be 3.0.

Although the development of a salinity model for Murray Bridge is important to develop optimal pumping policies, in the current research, the salinity case study was

primarily chosen because it provides a suitable study for illustrating the proposed ANN modelling methodology. A considerable amount of research has already been conducted using this case study (see Maier, 1995; Maier and Dandy, 1996a; Maier and Dandy, 1996b; Maier and Dandy, 1997a; Maier and Dandy, 1998a; Maier and Dandy, 1998b; Maier and Dandy, 1999; Maier and Dandy, 2001). It is envisaged that the proposed experimentation outlined in Chapter 3 will build upon this previous research and will address a number of the limitations and alternatives associated with each of the steps involved in the current ANN modelling methodology. The salinity case study has the following features, which exploit the strength of ANNs and make it a suitable case study for such experimentation:

- A large set of data consisting of many potential inputs and thousands of daily observations is available.
- Nonlinear relationships are suspected but have not been confirmed.
- There is considerable noise (measurement and sampling) present in the collected data.
- The exact mathematical relationship between the variables is difficult to prescribe.

Maier (1995) has also considered models of the ARIMA type for modelling salinity at Murray Bridge. A univariate time series (ARIMA) model and a multivariate time series (VARIMA) model were developed and compared to a univariate ANN (UANN) and a multivariate ANN (MANN). The ARIMA type time series models outperformed the ANN models when a short forecasting horizon was used (i.e. 1 day), while the ANN models were better suited to a longer forecasting horizon (i.e. 14 days). A possible cause for this may be the fact that ANN models base their forecasts on the underlying relationships that generated the time series. An additional reason for this may arise because the ANN models can be applied directly to longer forecasting horizons, without the need to apply the forecast recursively. The multivariate ANN outperformed the univariate ANN for all forecasting periods. The VARIMA model performed only marginally better than the ARIMA model, which suggested that the benefit of using causal variables in the time series model was only minimal. The development of the ANN models was found to be far simpler than the time series models considered.

In order to facilitate comparisons with the studies undertaken by Maier and Dandy, a forecasting period of 14 days has been selected for the present research. This forecasting period is significant because it is the minimum time required to make short-term adjustment to the pumping schedule. Short-term forecasts are also useful in providing accurate advice to irrigators.

4.3 Available Data

The salinity data set was compiled by Maier (1995) and consisted of data supplied by the Engineering and Water Supply Department (EWS) of South Australia for the period 01-12-1986 to 30-06-1992. Salinity transport in a river is affected by upstream salinities and flow, while saline groundwater accessions are affected by flow, river levels and groundwater levels. Consequently, upstream salinities, flow, river levels and groundwater levels were selected as the dominant variables affecting the salinity in the River Murray at Murray Bridge. Each of the variables, with the exception of groundwater levels, were measured on a daily basis. However, Maier (1995) noted that the groundwater levels are likely to change very slowly with time, and as such, are unlikely to exert a significant influence on the temporal variation of groundwater accessions. Consequently, the available data (salinities, flows and river levels) were deemed suitable in accounting for salinity transport and saline groundwater accessions. No information was available on other forms of saline accessions, which include: saline inflows from drains, natural surface runoff and the flushing of anabranches and backwaters.

In the present study, the database was extended to include data on the same variables for the period 01-07-1992 to 01-04-1998. An additional variable was also included in the database, which was the flow time series recorded just downstream of Lock 7. The Murray-Darling Basin Commission supplied the additional salinity data and the South Australian Department for Water Resources supplied the additional flow and river level data. The salinity, flow and river level data used in this research are summarised in Table 4.2 along with the abbreviations used for each variable in the remainder of this chapter. It should be noted that the river level data used in this research are given above the Australian Height Datum (AHD). The locations of each variable are presented in Figure 4.7.

In any modelling study it is important to gain familiarity with the data under investigation by inspecting plots of the available time series data. In order to perform a visual inspection of the data, each of the variables listed in Table 4.2 have been plotted below. This helps to highlight any seasonal effects, trends, heteroskedasticity, outliers, discontinuities and spatial relationships that may exist in the data.

Table 4.2 Available Data (Daily)

Location	Data Type	Abbreviation
Murray Bridge	Salinity	MBS
Mannum	Salinity	MAS
Morgan	Salinity	MOS
Waikerie	Salinity	WAS
Loxton	Salinity	LOS
Lock 1 Lower	Flow	L1LF
Overland Corner	Flow	OCF
Downstream of Lock 7	Flow	L7F
Murray Bridge	River Level	MBL
Mannum	River Level	MAL
Lock 1 Lower	River Level	L1LL
Lock1 Upper	River Level	L1UL
Morgan	River Level	MOL
Waikerie	River Level	WAL
Overland Corner	River Level	OCL
Loxton	River Level	LOL

4.3.1 Salinity Data

The output (dependent) variable that is being modelled in this research is the salinity at Murray Bridge. A plot of the time series of salinity at Murray Bridge (MBS) from 01-01-1987 to 01-04-1998 is shown in Figure 4.8. With the exception of 1994 and 1997, a definite seasonal variation is apparent. The seasonality manifests itself as high salinities in the first half of the year and low salinities in the second half of the year. It is also apparent that the seasonal pattern is irregular, with the start, end and duration of the high and low salinity periods varying from year to year. To determine if a trend was present in the data, a linear regression with time was performed using the salinity data (Figure 4.8) and the slope of the regression equation was tested for statistical significance. The slope coefficient was divided by the estimated standard error of the slope coefficient to give a t-observed = 3.49. For a one-sided test, with 4107 degrees of freedom, the critical t-value at the $\alpha = 0.05$ significance level is 1.65. Since the absolute value of t-observed is greater than t-critical, the slope can be considered statistically significant and hence, a trend is present in the data. Over the 11 years and 3 months considered, this equated to an increase in the average salinity of approximately 3.6 EC units per year. Over this period, the salinity ranged from 212 to 1247 EC units, with a mean of 607 EC units and a standard deviation of 214 EC units.

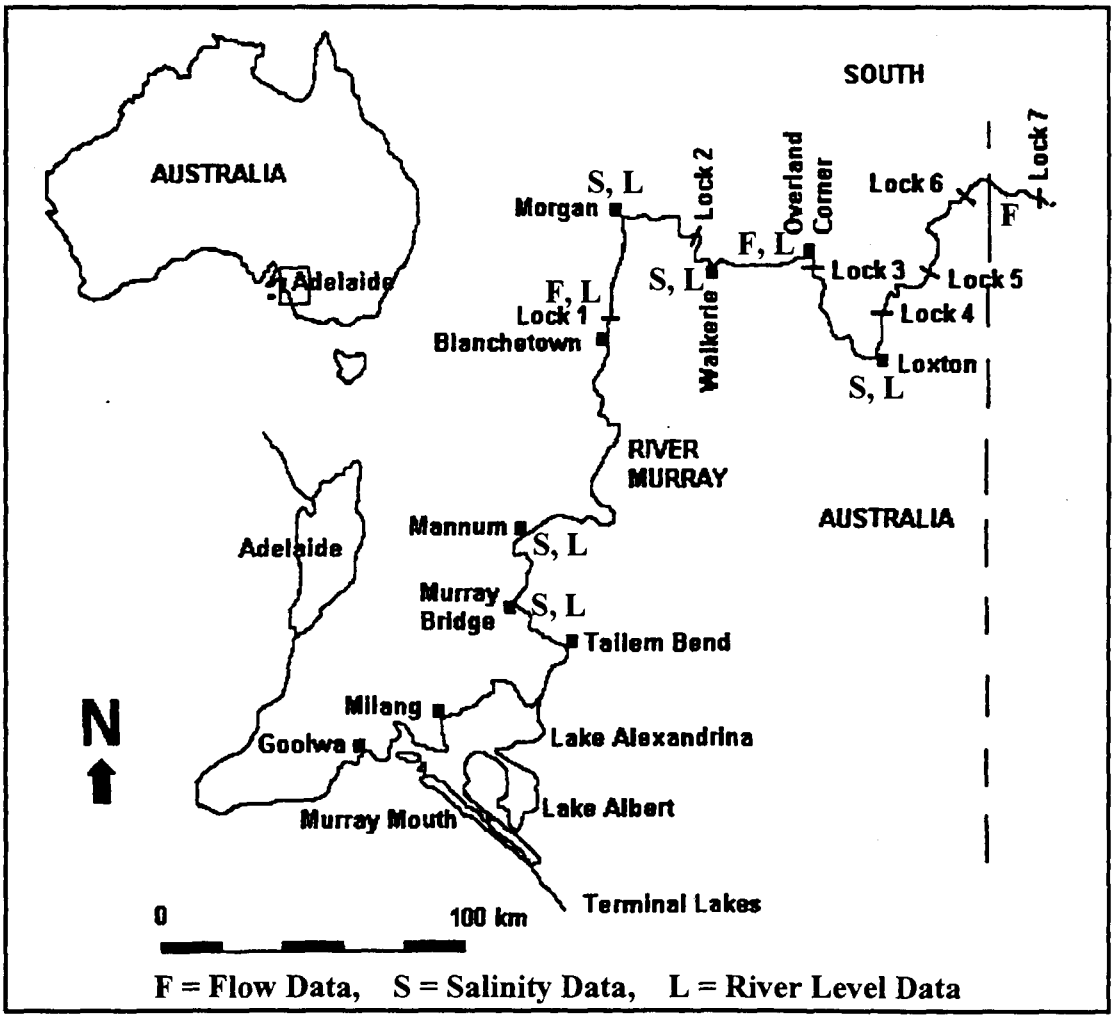

Figure 4.7 Locations of Available Data

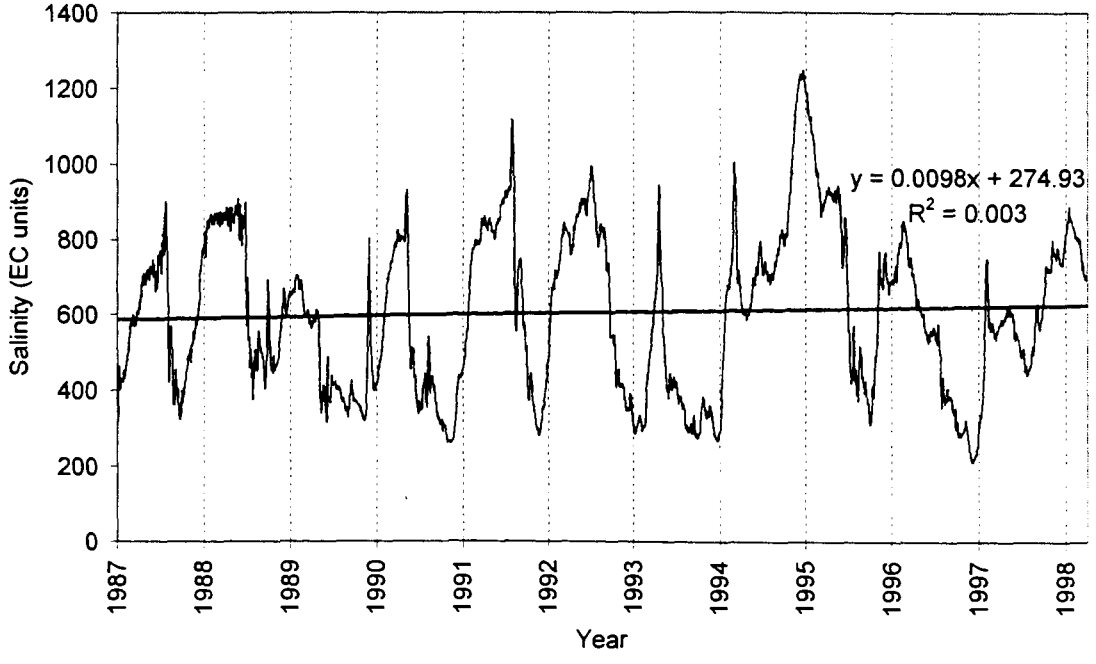

Figure 4.8 Salinity at Murray Bridge (MBS) (1987 to 1998)

Figure 4.9 shows the annual exceedance probabilities (AEP) obtained using the peak annual salinity at Murray Bridge over the period 01-01-1987 to 31-12-1997. Based on the available data, it is apparent that a peak annual salinity at Murray Bridge of 1200 EC units has an AEP of approximately 1 in 8 years. During the same period, the peak annual salinity at Murray Bridge exceeded 800 EC units 100% of the time.

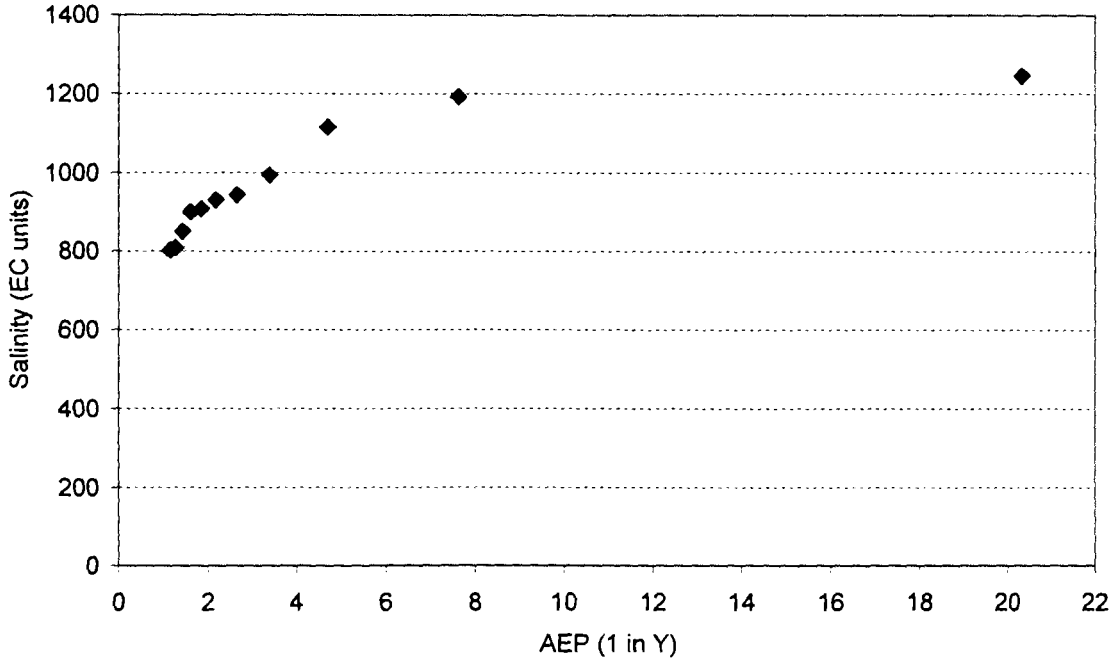

Figure 4.9 Annual Exceedance Probabilities (AEP) Using Peak Salinities for the Salinity Time Series at Murray Bridge (1987 to 1998)

A statistical summary (mean, standard deviation, maximum and minimum) of the salinity time series at Murray Bridge and upstream locations is presented in Table 4.3. It is apparent that the mean, standard deviation and maximum values increase with distance downstream, confirming the effect of saline accessions discussed in Section 4.2.2.

Table 4.3 Statistics of the Salinity Time Series Data (1987 to 1998)

Variable	Mean (EC units)	Standard Deviation (EC units)	Maximum (EC units)	Minimum (EC units)
MBS	607	214	1247	212
MAS	574	196	1164	209
MOS	574	189	1091	177
WAS	551	179	1056	201
LOS	488	139	907	193

Three yearly plots of the salinity at Murray Bridge and the various upstream locations are presented in Figure 4.10 to Figure 4.25. These plots reveal useful information about the relationship between salinity at upstream locations and salinity at Murray Bridge. In a small number of instances, the salinity at an upstream location is greater than that measured at Murray Bridge, even after taking the travel times into account. These phenomena are anomalous, because salinity in the river can be considered conservative and usually increases with distance downstream, due to saline accessions. Examples of this occur in Figure 4.10 near day 546, Figure 4.11 near day 1055, Figure 4.12 near day 1052, Figure 4.18 near days 69, 846 and 874 and Figure 4.23 near day 370. These isolated events may have been caused by localised saline slugs emanating from accessions upstream from the sampling site. It is conceivable that these slugs were then diluted once mixing occurred with distance traveled downstream, giving rise to lower salinity levels as measured at the Murray Bridge sampling site.

It is also evident from the plots that large saline accessions occur between Loxton and Waikerie. These appear as sharp peaks, or significantly magnified peaks, on the plots of salinity at Waikerie that are not present, or are significantly smaller, on the plots of salinity at Loxton. Examples of this occur in Figure 4.12 near days 675 and 1052 and in Figure 4.23 near day 370.

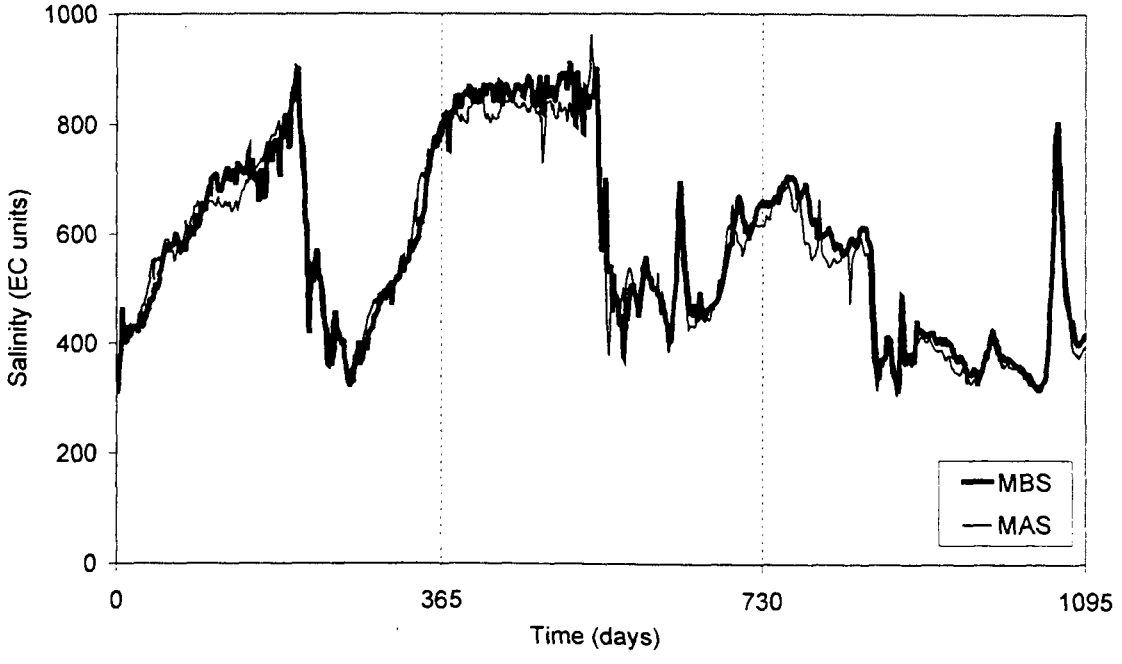

Figure 4.10 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1987 to 1989)

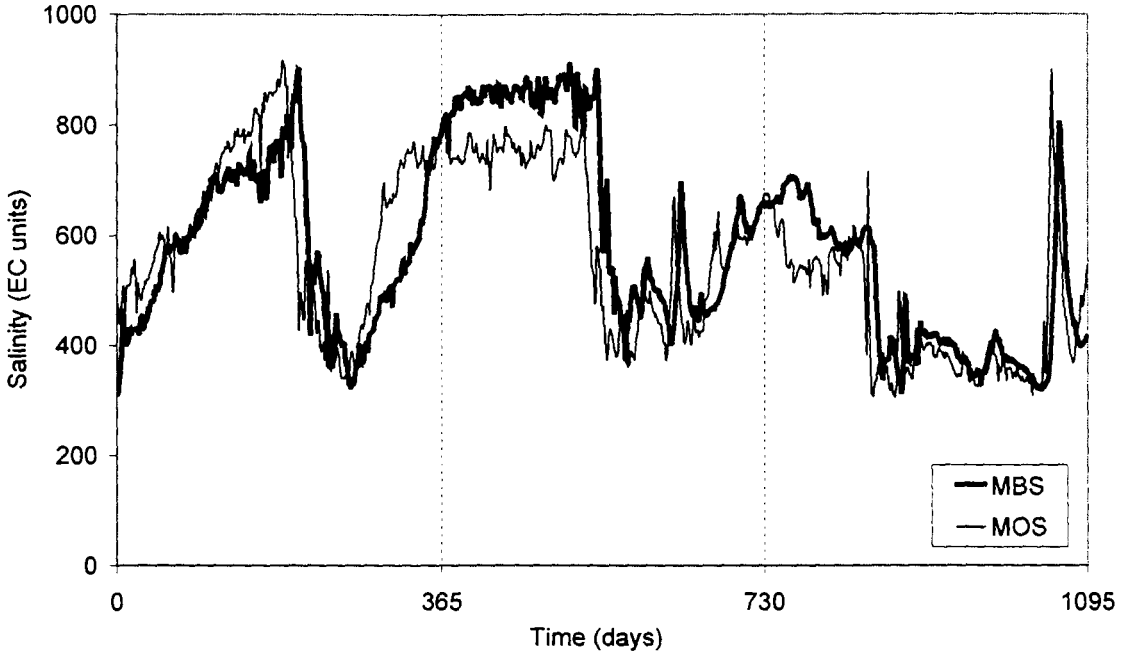

Figure 4.11 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1987 to 1989)

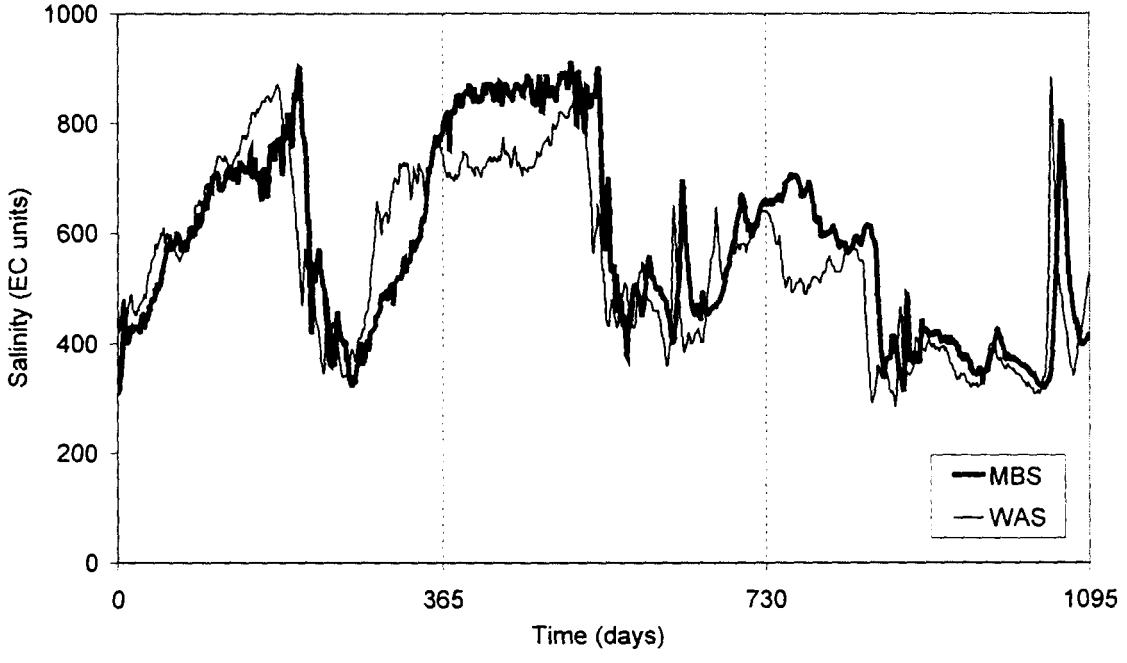

Figure 4.12 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1987 to 1989)

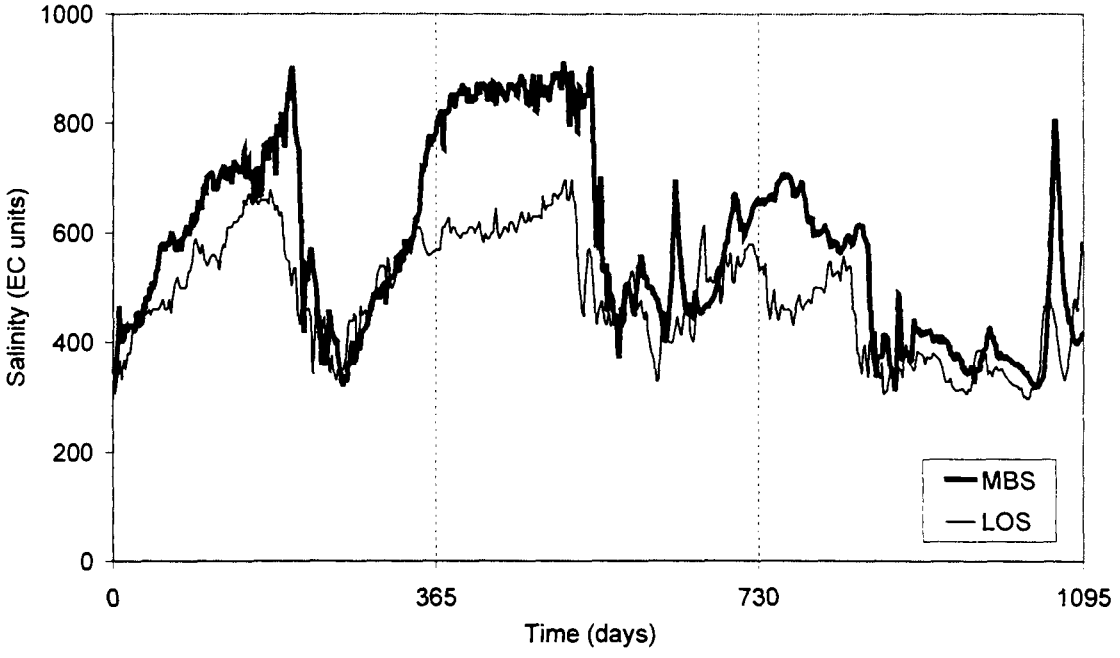

Figure 4.13 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1987 to 1989)

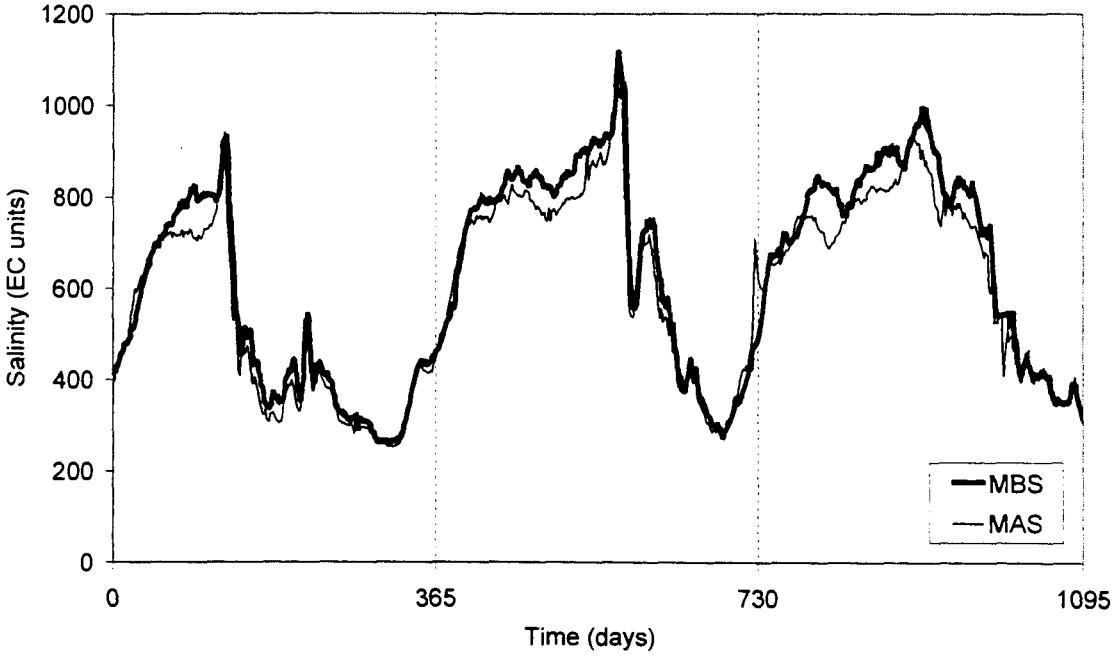

Figure 4.14 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1990 to 1992)

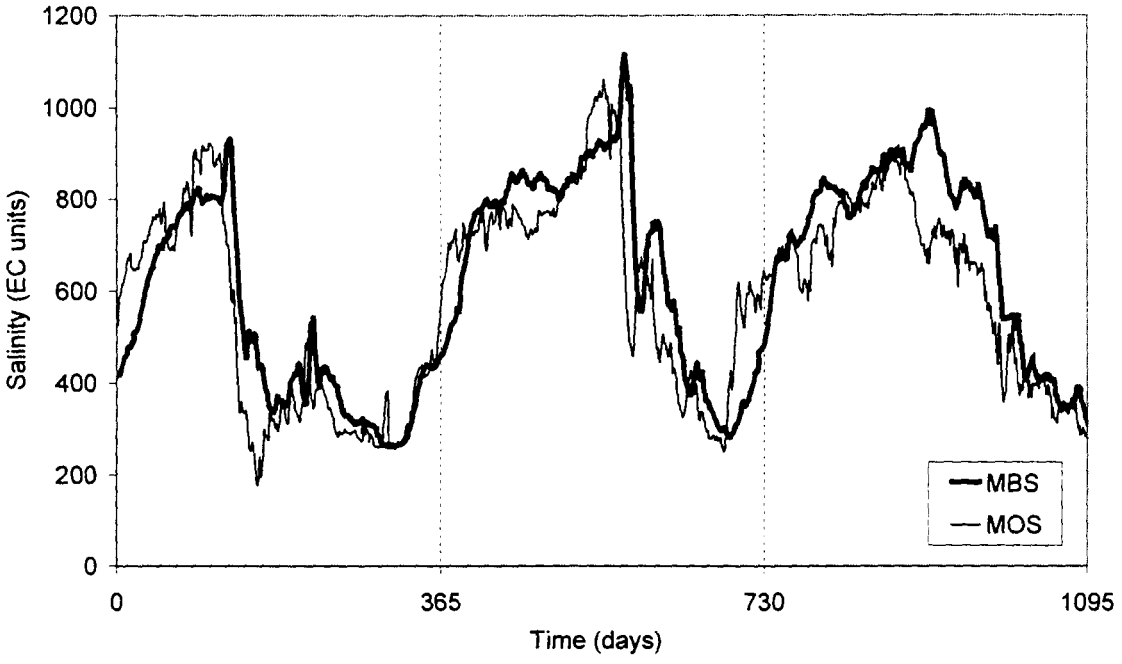

Figure 4.15 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1990 to 1992)

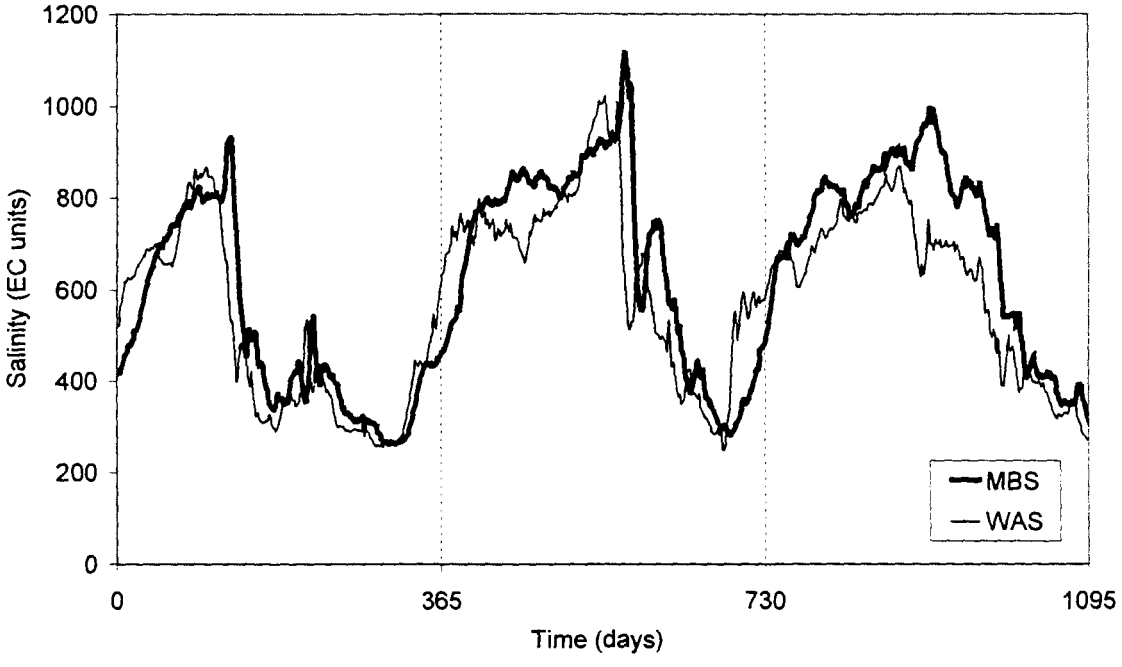

Figure 4.16 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1990 to 1992)

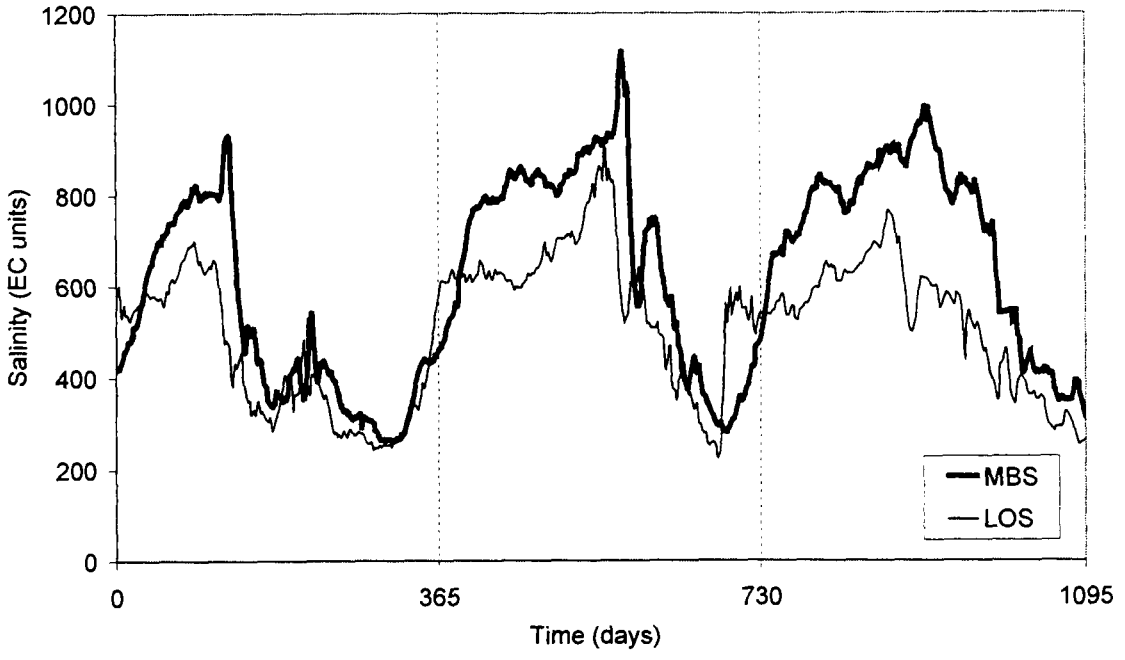

Figure 4.17 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1990 to 1992)

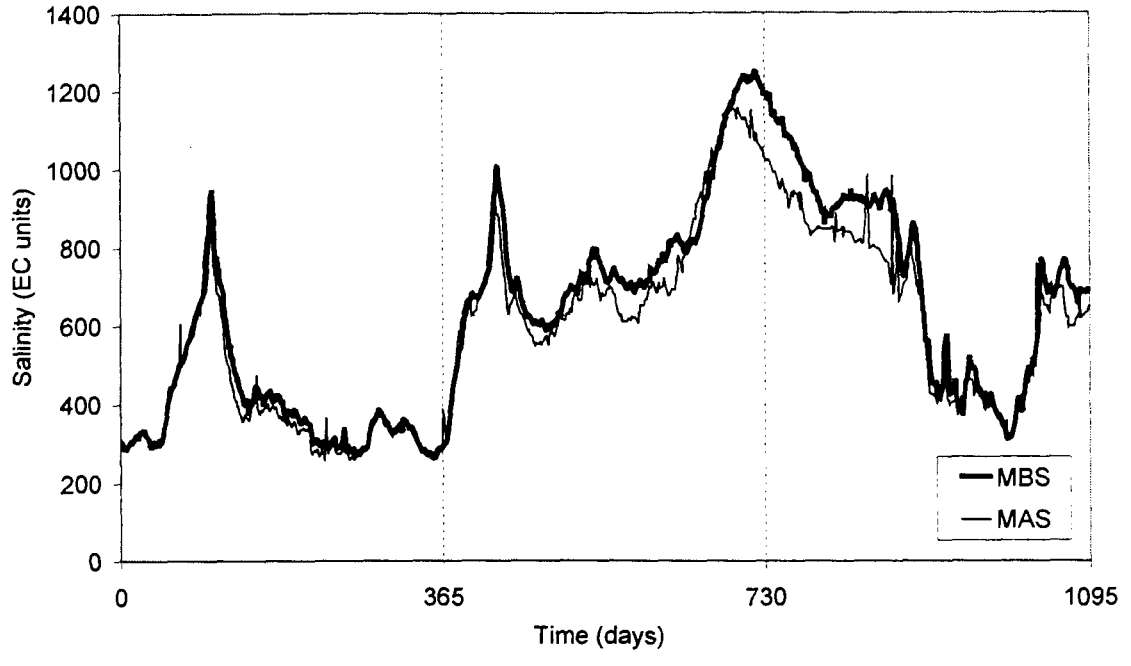

Figure 4.18 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1993 to 1995)

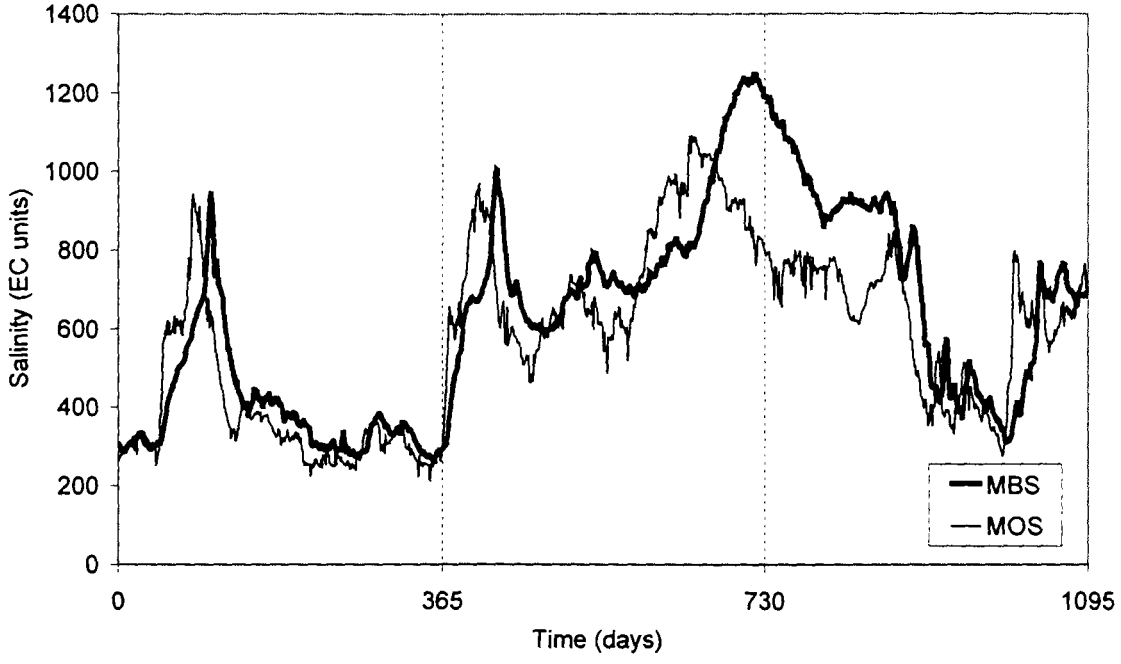

Figure 4.19 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1993 to 1995)

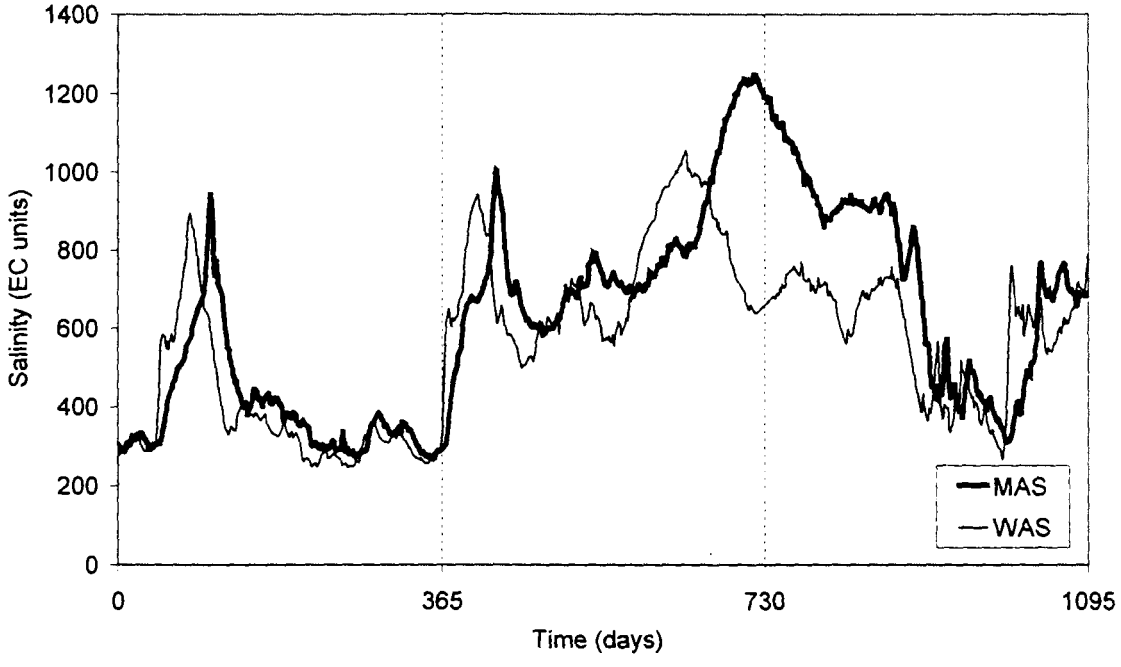

Figure 4.20 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1993 to 1995)

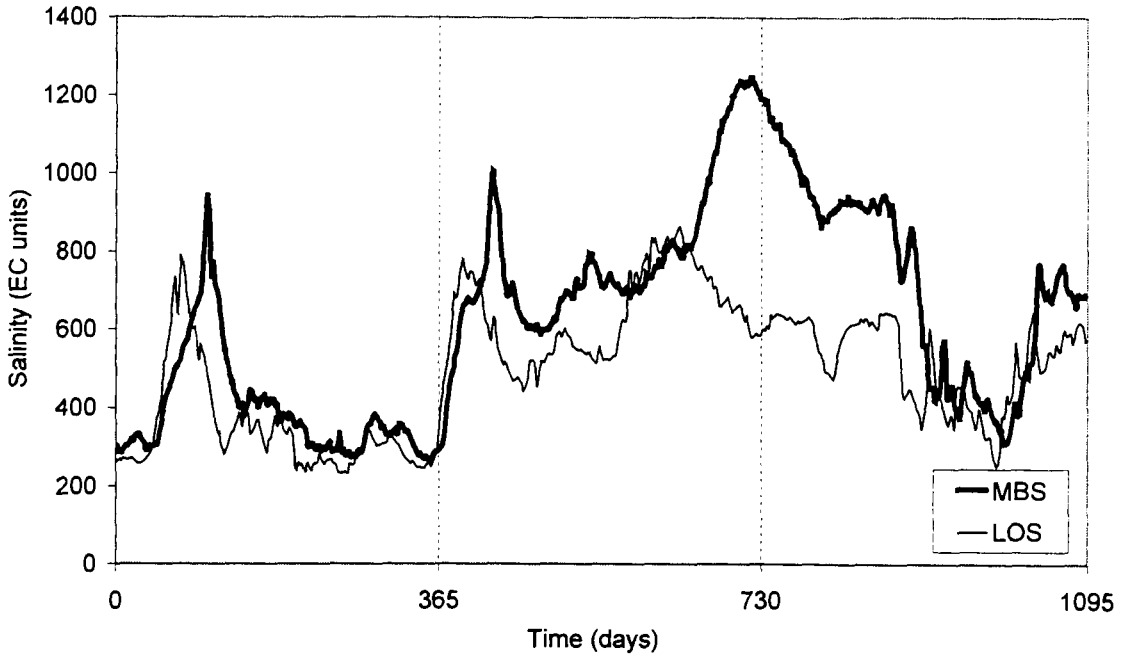

Figure 4.21 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1993 to 1995)

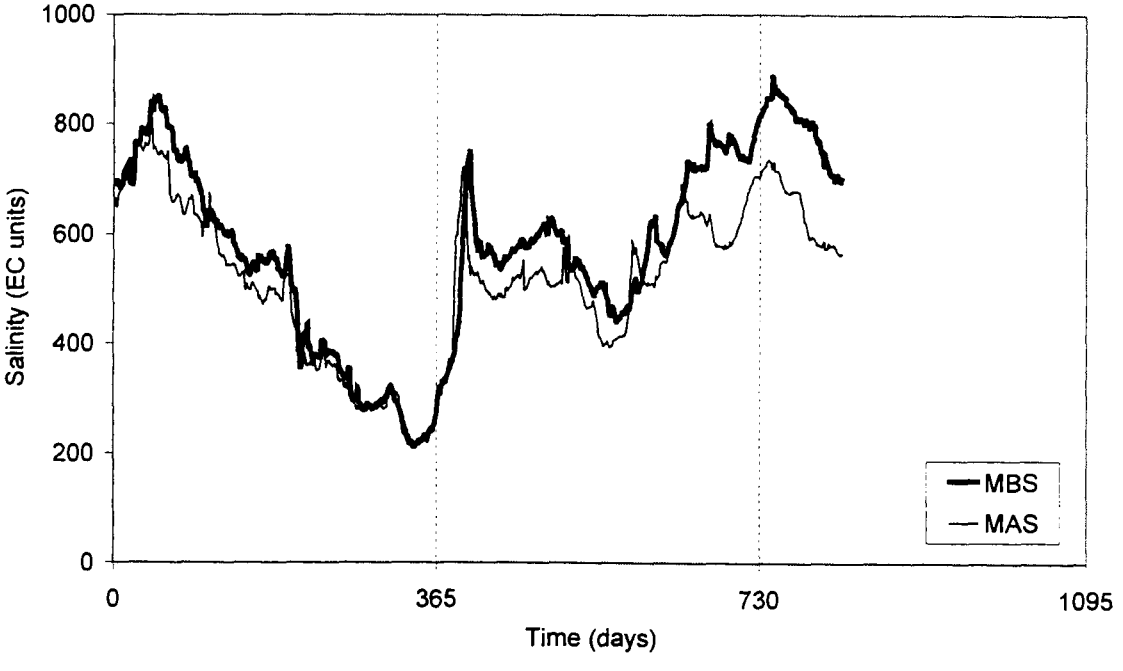

Figure 4.22 Salinity at Murray Bridge (MBS) and Salinity at Mannum (MAS) (1996 to 1998)

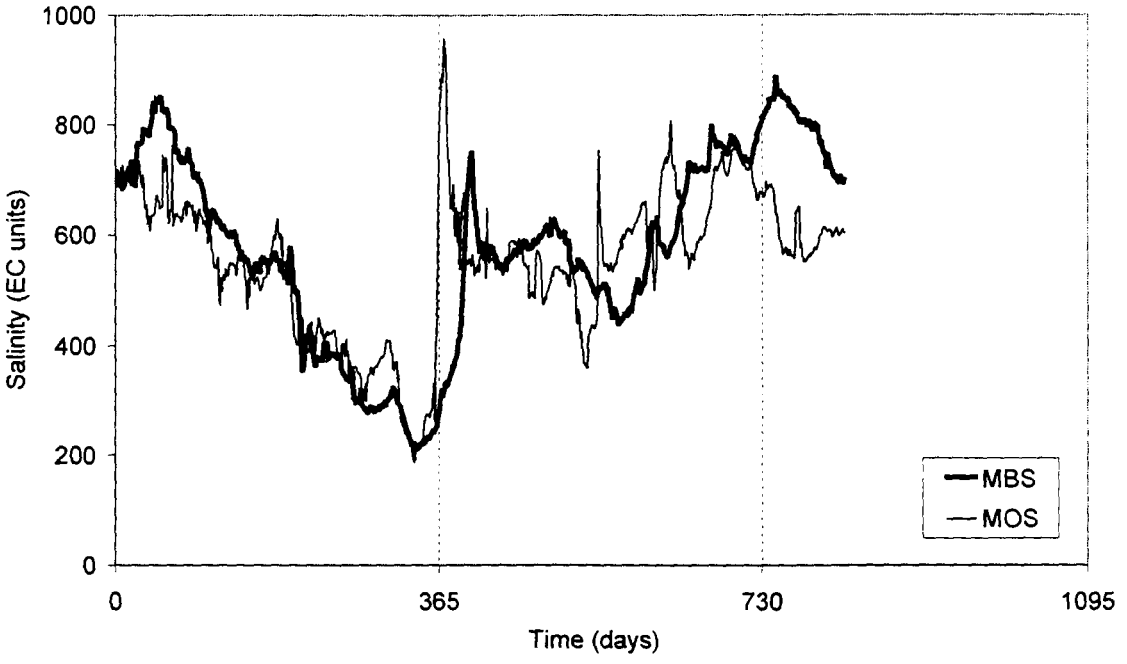

Figure 4.23 Salinity at Murray Bridge (MBS) and Salinity at Morgan (MOS) (1996 to 1998)

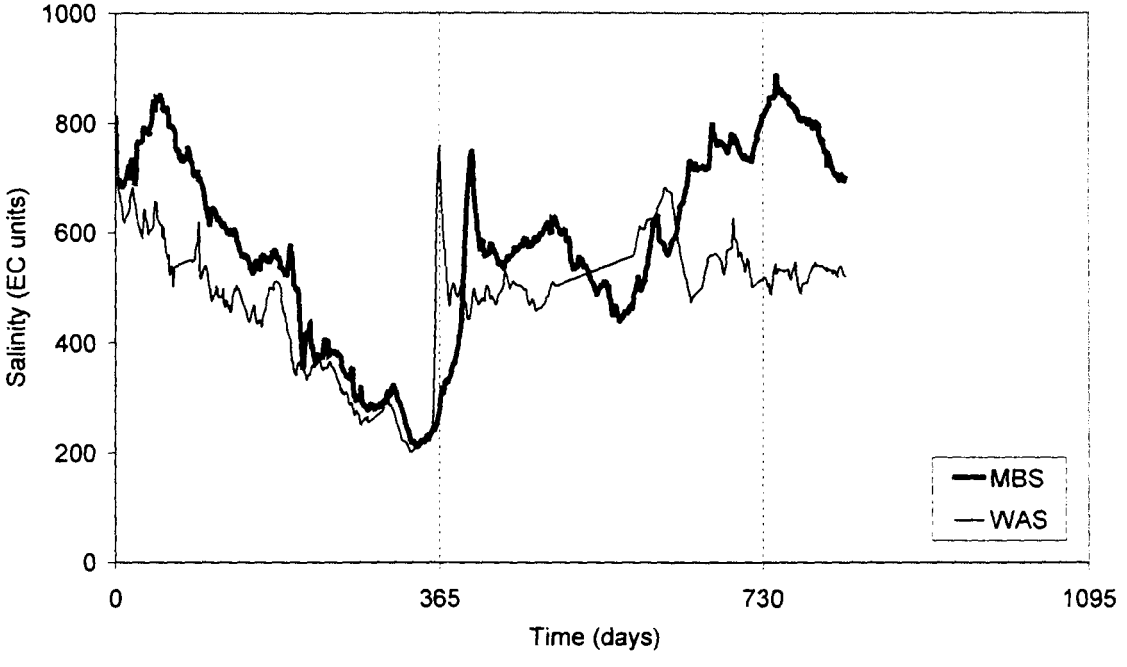

Figure 4.24 Salinity at Murray Bridge (MBS) and Salinity at Waikerie (WAS) (1996 to 1998)

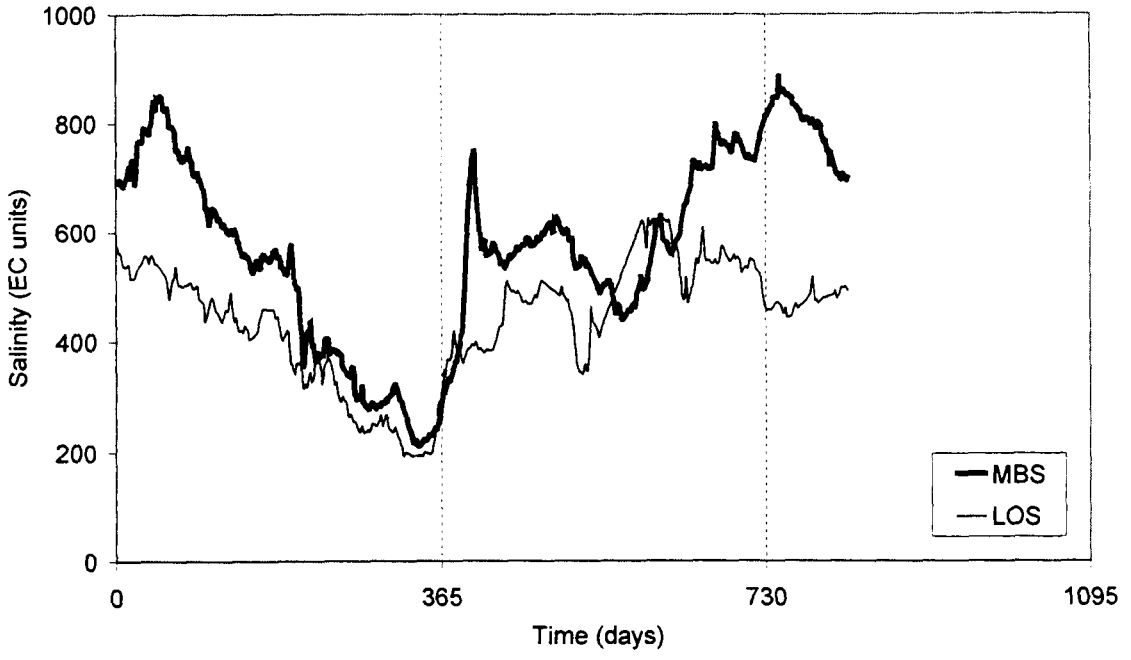

Figure 4.25 Salinity at Murray Bridge (MBS) and Salinity at Loxton (LOS) (1996 to 1998)

4.3.2 Flow Data

A statistical summary (mean, standard deviation, maximum and minimum) of the flow time series used for this case study is presented in Table 4.4. It is apparent that the mean and minimum values decrease with increasing distance downstream. This is due to abstractions from the river. The standard deviation remains relatively constant throughout the lower reaches of the River Murray and the maximum value increases with increasing distance downstream.

Table 4.4 Statistics of the Flow Time Series Data (1987 to 1998)

Variable	Mean (ML/day)	Standard Deviation (ML/day)	Maximum (ML/day)	Minimum (ML/day)
L1LF	20237	24628	115000	1026
OCF	21014	23993	114315	1769
L7F	21946	24608	109831	2034

Plots of the flow time series and the salinity time series at Murray Bridge are shown in Figure 4.26 to Figure 4.28. The most notable feature of these plots is the strong inverse relationship exhibited between flow and salinity. To determine if the flow data possessed a trend, a linear regression with time was performed for each time series and

the slope of the regression equation was tested for statistical significance (Table 4.5). The slope coefficient was divided by the estimated standard error of the slope coefficient to give the values of *t*-observed. For a one-sided test, with 4107 degrees of freedom, the critical *t*-value at the $\alpha = 0.05$ significance level is 1.65. It is evident that for each flow variable, the absolute value of *t*-observed is greater than *t*-critical. Therefore, a trend is present in each of the flow time series. A strong seasonal component is also apparent in the flow data, with periods of low flow occurring during the first half of the year and periods of high flow during the second half. Like the salinity data, this seasonal variation is irregular with the onset, magnitude and duration of the high and low flow events differing from year to year. No high flow events were recorded during 1994 and 1997 at any of the three locations considered.

Table 4.5 Regression Statistics for the Flow Time Series Data (1987 to 1998)

Variable	Slope Coefficient	Intercept	Standard Error of Slope	Slope Coefficient Significance Test	
				t-critical	*t*-observed
L1LF	-1.05	22402	0.32	1.65	-3.28
OCF	-1.58	24270	0.32	1.65	-4.94
L7F	-1.40	24814	0.32	1.65	-4.38

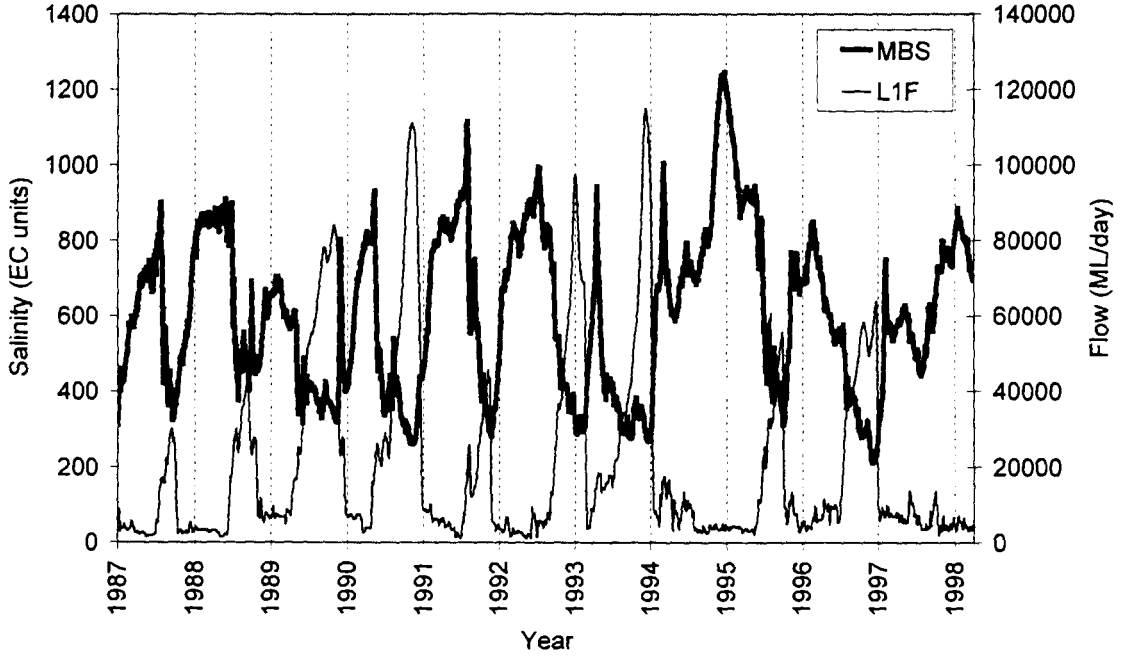

Figure 4.26 Salinity at Murray Bridge (MBS) and Flow at Lock 1 Lower (L1LF) (1987 to 1998)

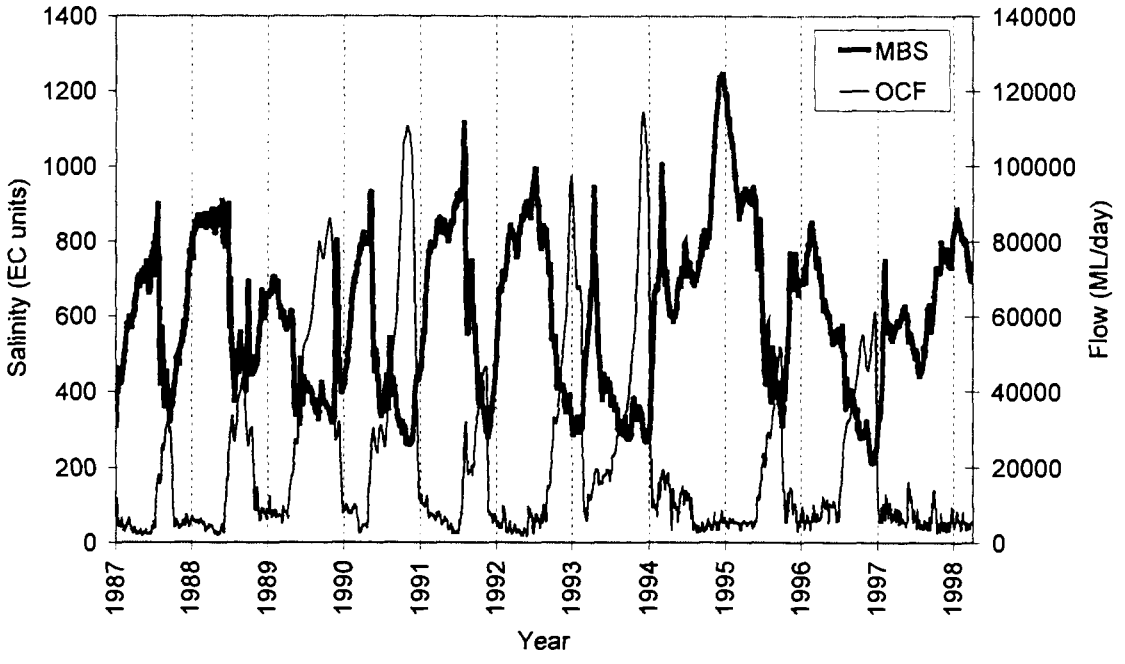

Figure 4.27 Salinity at Murray Bridge (MBS) and Flow at Overland Corner (OCF) (1987 to 1998)

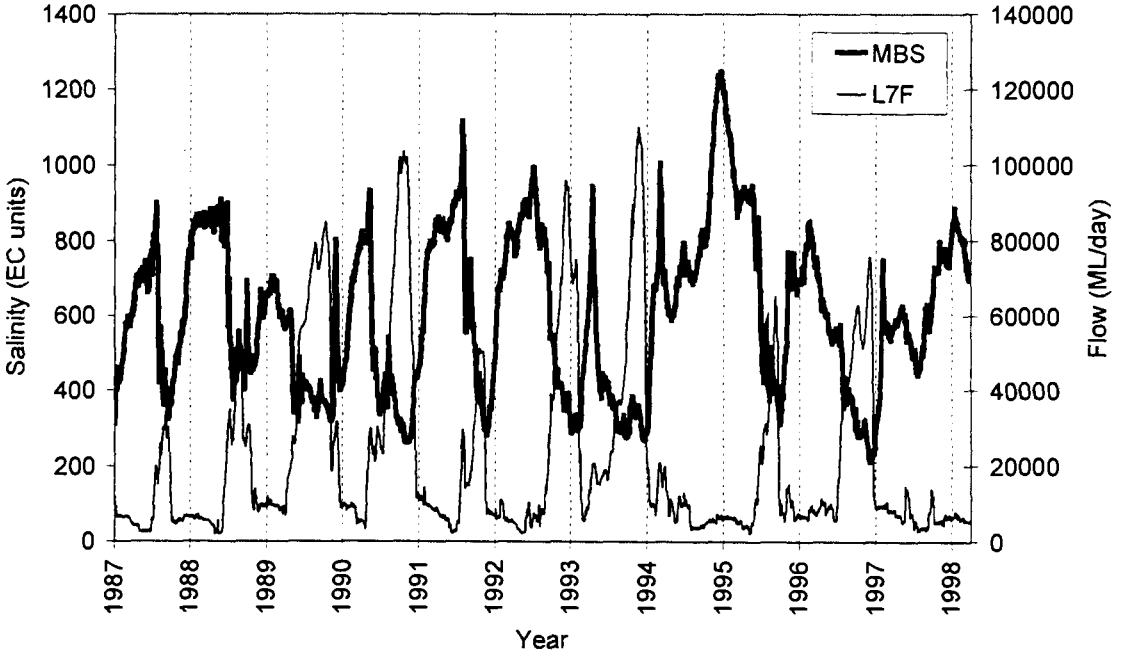

Figure 4.28 Salinity at Murray Bridge (MBS) and Flow Downstream of Lock 7 (L7F) (1987 to 1998)

4.3.3 River Level Data

The summary statistics (mean, standard deviation, maximum and minimum) of the river level time series data are presented in Table 4.6. As expected, the mean, maximum and minimum river levels decrease with increasing distance downstream.

Table 4.6 Statistics of the River Level Time Series Data (1987 to 1998)

Variable	Mean (m)	Standard Deviation (m)	Maximum (m)	Minimum (m)
MBL	0.76	0.14	1.57	0.41
MAL	0.83	0.21	1.89	0.49
L1LL	1.58	1.16	5.38	0.46
L1UL	3.36	0.36	5.38	2.88
MOL	3.87	1.02	7.65	3.10
WAL	6.68	0.88	10.24	6.13
OCL	7.14	1.29	11.58	6.15
LOL	10.77	1.10	14.12	9.85

Plots of the river level time series and the salinity at Murray Bridge are shown in Figure 4.29 to Figure 4.36. The river levels are positively correlated with flow, and therefore, exhibit an inverse relationship with the salinity time series. The river level data follow the same irregular seasonal fluctuation as the flow data. To determine if the river level data possessed a trend, a linear regression with time was performed for each time series and the slope of the regression equation was tested for statistical significance (Table 4.7). The slope coefficient was divided by the estimated standard error of the slope coefficient to give the values of t-observed. For a one-sided test, with 4107 degrees of freedom, the critical t-value at the $\alpha = 0.05$ significance level is 1.65. It is evident that for each river level variable, the absolute value of t-observed is greater than t-critical. Therefore, a trend is present in each of the river level time series.

For each of the locations investigated, there was very little fluctuation in the river level during 1994 and 1997, which corresponds to the low flow events during these years. It is evident from the plots that the level data recorded at Murray Bridge and Mannum exhibit a higher degree of noise than the river levels recorded at the upstream locations (i.e. Lock 1, Morgan, Waikerie and Loxton). This is because the river levels at the upstream locations are highly controlled by the system of locks in place in that region. Mannum and Murray Bridge are the two locations in this study that are furthest from this system of locks. As such, there is a higher degree of fluctuation in the river level at these downstream locations, since the river is not as highly regulated in this section of

the river. The plots also reveal that the level at Lock 1 Upper remains fairly constant, except for extreme flood events. This is to be expected, since the locks are operated to maintain a constant weir pool level at all times in order to facilitate navigation and water supply offtakes.

Table 4.7 Regression Statistics for the River Level Time Series Data (1987 to 1998)

Variable	Slope Coefficient	Intercept	Standard Error of Slope	Slope Coefficient Significance Test	
				t-critical	*t*-observed
MBL	-0.000015	0.79	0.000002	1.65	-8.21
MAL	-0.000012	0.86	0.000003	1.65	-4.39
L1LL	-0.000076	1.73	0.000015	1.65	-4.98
L1UL	-0.000016	3.39	0.000005	1.65	-3.46
MOL	-0.000052	3.98	0.000013	1.65	-3.93
WAL	-0.000039	6.76	0.000012	1.65	-3.40
OCL	-0.000073	7.29	0.000017	1.65	-4.29
LOL	-0.000056	10.89	0.000015	1.65	-3.84

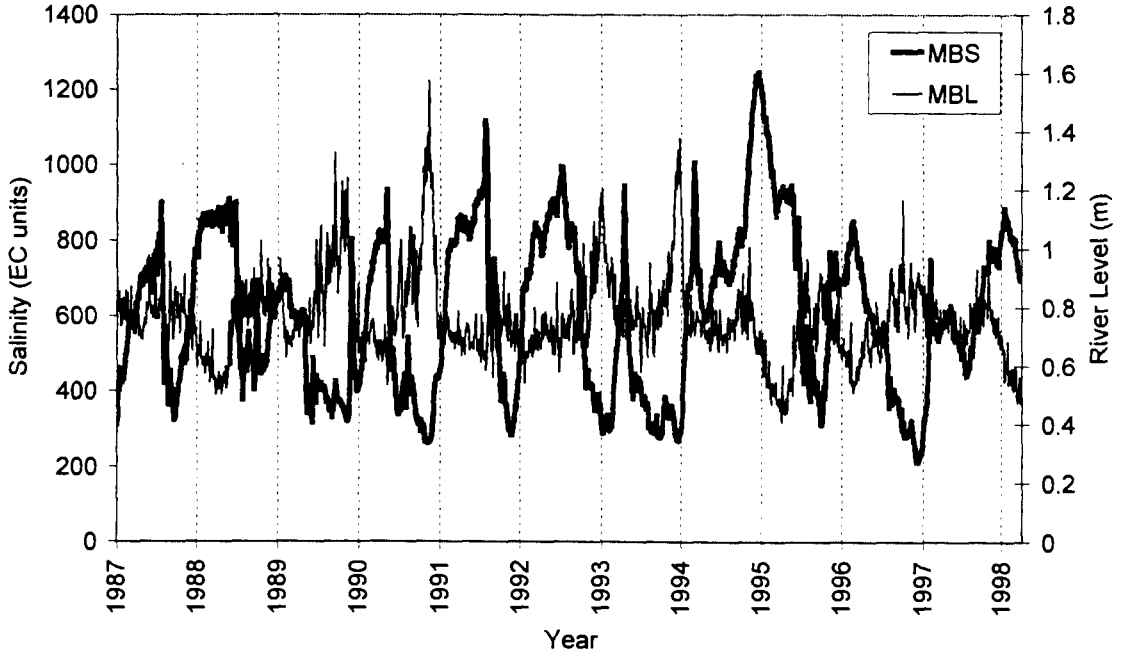

Figure 4.29 Salinity at Murray Bridge (MBS) and River Level at Murray Bridge (MBL) (1987 to 1998)

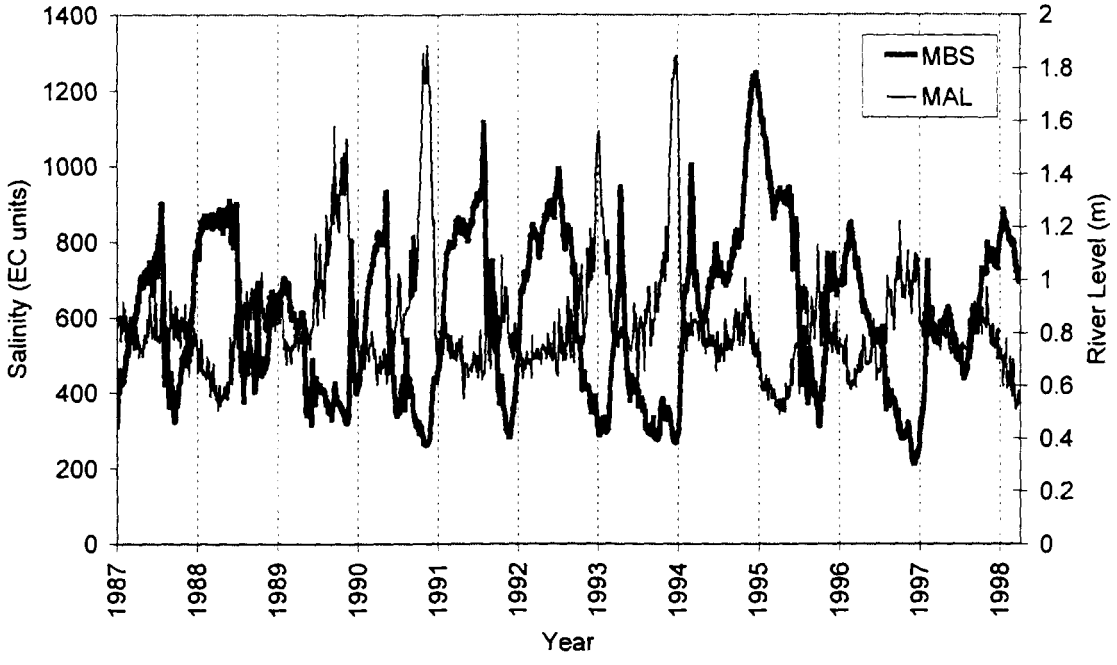

Figure 4.30 Salinity at Murray Bridge (MBS) and River Level at Mannum (MAL) (1987 to 1998)

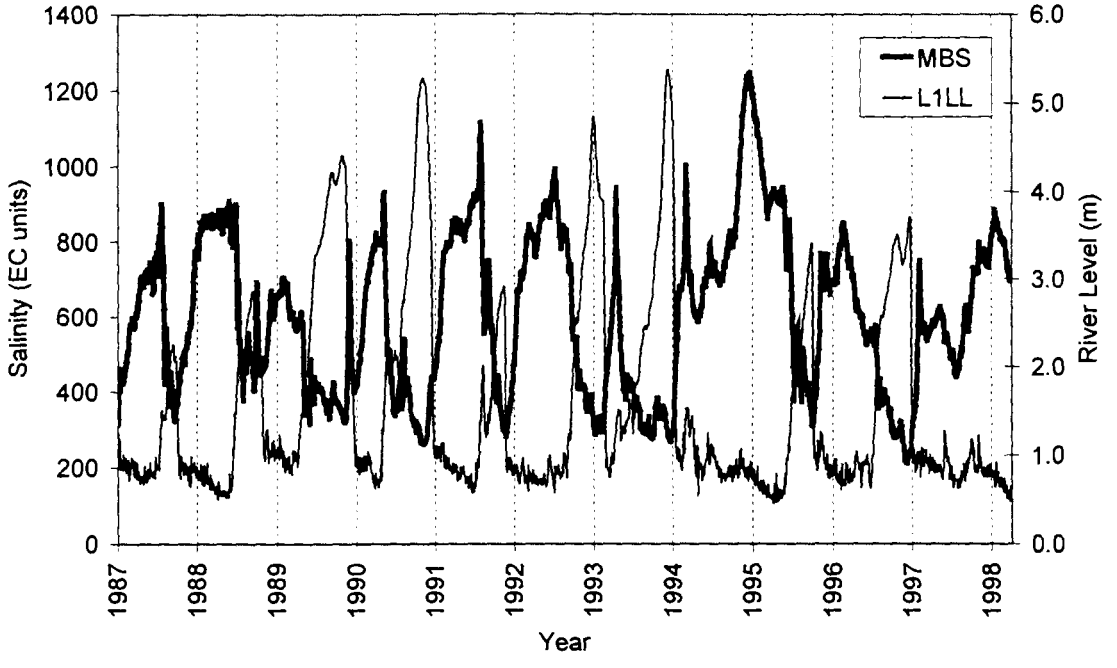

Figure 4.31 Salinity at Murray Bridge (MBS) and River Level at Lock 1 Lower (L1LL) (1987 to 1998)

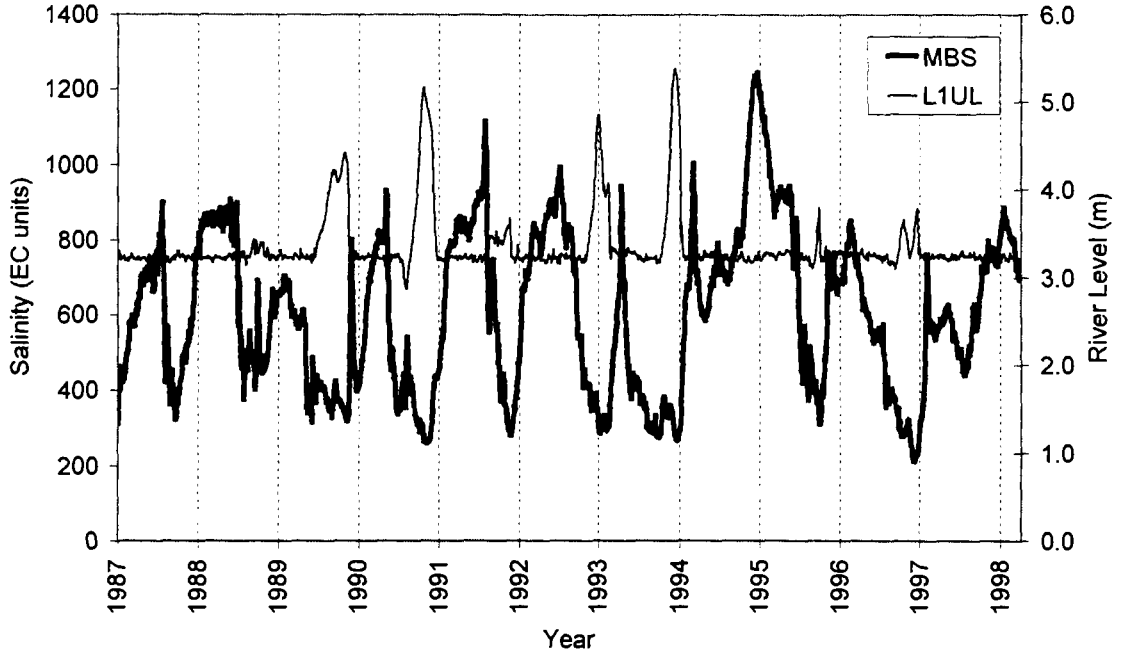

Figure 4.32 Salinity at Murray Bridge (MBS) and River Level at Lock 1 Upper (L1UL) (1987 to 1998)

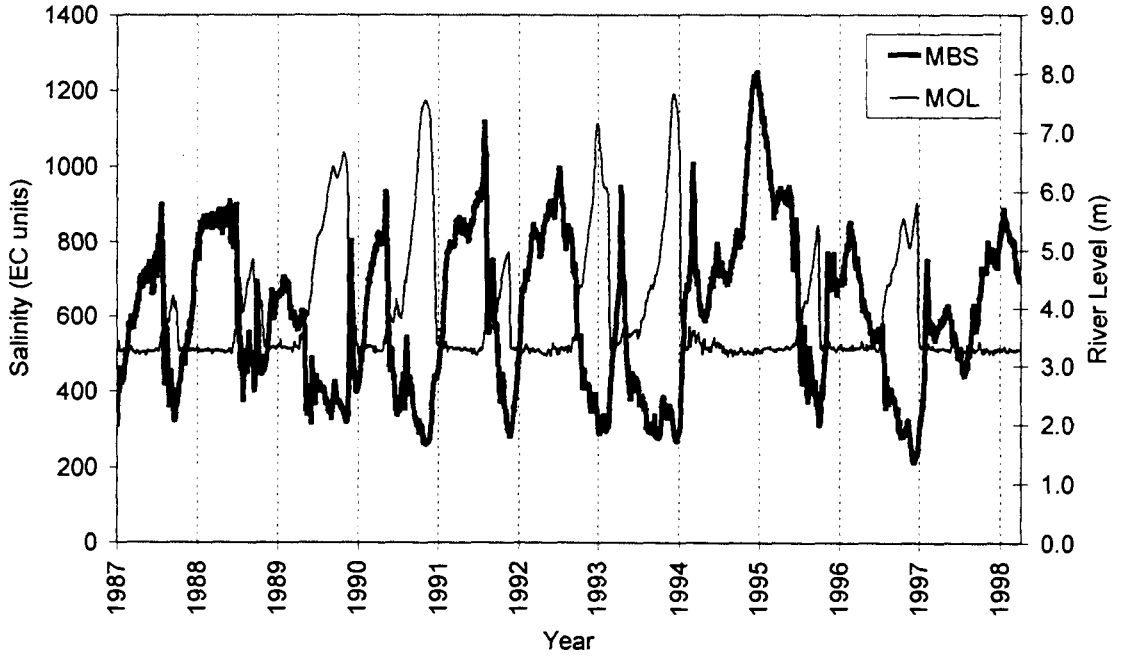

Figure 4.33 Salinity at Murray Bridge (MBS) and River Level at Morgan (MOL) (1987 to 1998)

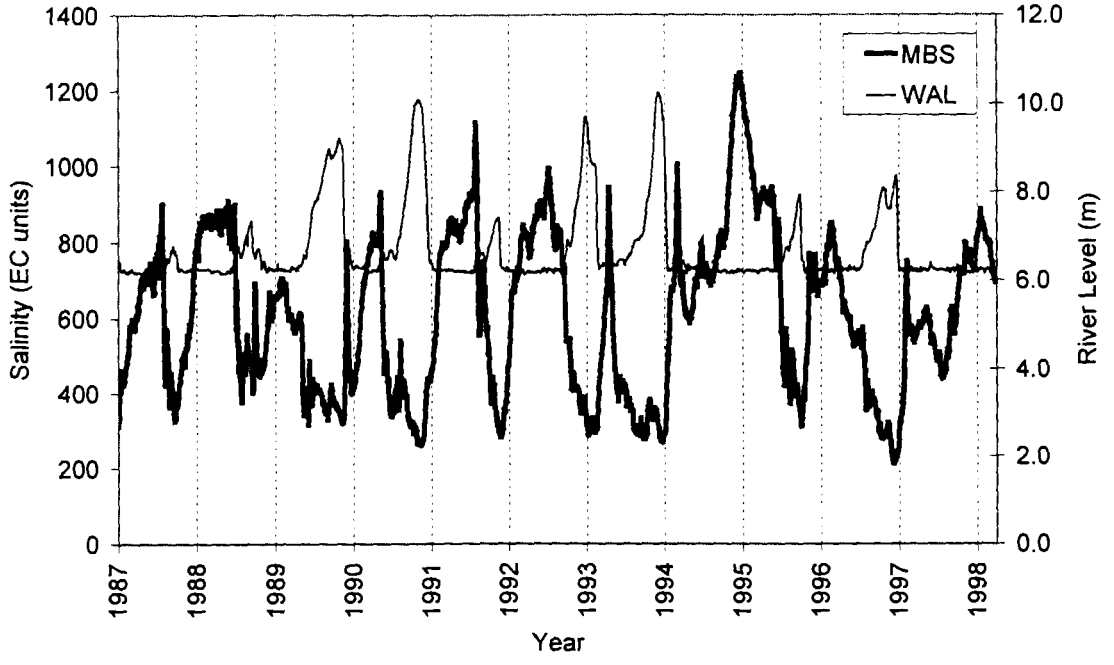

Figure 4.34 Salinity at Murray Bridge (MBS) and River Level at Waikerie (WAL) (1987 to 1998)

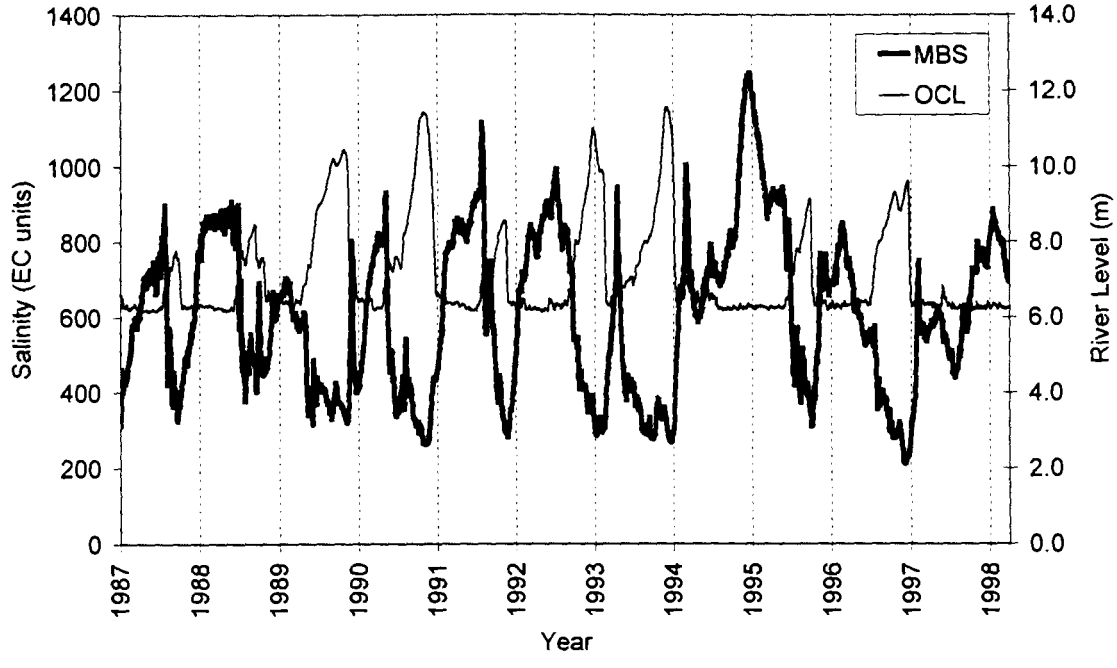

Figure 4.35 Salinity at Murray Bridge (MBS) and River Level at Overland Corner (OCL) (1987 to 1998)

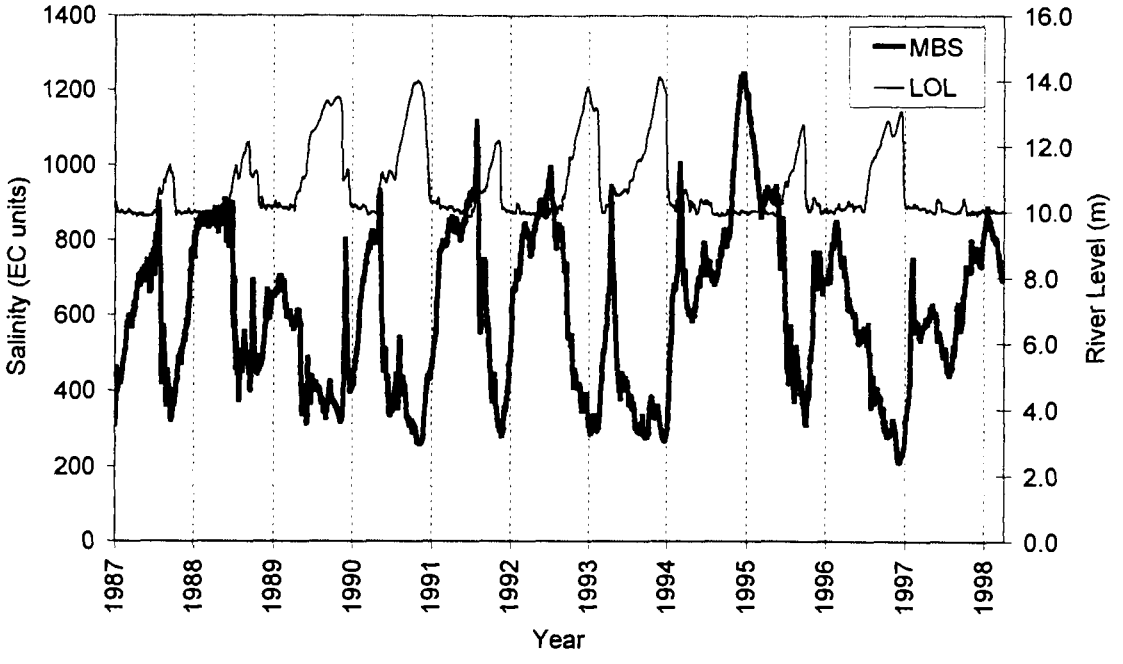

Figure 4.36 Salinity at Murray Bridge (MBS) and River Level at Loxton (LOL) (1987 to 1998)

4.4 ANN Model Development

The development of an ANN model is usually conducted in a sequential manner. Typically, the data are divided into subsets, preprocessed and transformed, the appropriate model inputs are determined, a suitable network architecture and training (optimisation) algorithm is chosen and finally, the model is then validated using some performance criteria. However, each of the modelling steps are interconnected and dependent in some way on the preceding and subsequent steps in the development process. For example, when determining which transformation of the data performs best for the case study under investigation, a set of model inputs must be assumed. At the same time, when the appropriate inputs are determined, it is necessary to assume a certain data transformation. So in effect the modeller faces the age-old question – "what came first, the chicken or the egg?"

To circumvent this problem, the approach adopted in this research was to use a series of defaults for each of the subsequent steps in the modelling methodology. These default selections were based on previous research conducted by Maier and Dandy, using the salinity case study. Each step was considered in isolation, while holding constant the remaining steps using the defaults. Once the empirical trials had been completed for a particular step, the option found to give the best performance was then used during the experimentation performed for each of the remaining steps. For example, if the GA data division method was found to provide superior performance, then this method would be used for all of the models developed during the subsequent steps.

Maier and Dandy (1996b) used feedforward MLPs trained using the backpropagation algorithm to forecast salinity in the River Murray at Murray Bridge. This type of ANN model was found to perform well for this case study and therefore, was adopted as the default network type for the present research. Feedforward MLPs are the most widely used type of ANN for the forecasting/prediction of water resources variables (as discussed in Section 3.6) and therefore, provide a good benchmark against which other models may be compared.

The feedforward MLP models were implemented using the commercially available software package NeuralWorks Professional II/PLUS (NeuralWare, 1998b). This software package has the following features, which are relevant to this research:

1. The training and testing sets can be automatically scaled to a user defined range, producing linearly transformed training, testing and validation sets.

2. The number of nodes in the input layer, the first hidden layer, the second hidden layer and the output layer can be selected.
3. Different learning rates and momentum values can be specified for each layer, and these can be made to vary as training progresses.
4. Six training algorithms are supported, including the following first-order methods: the generalized delta (GD) rule; the normalized cumulative delta (NCD) rule; the delta-bar-delta (DBD) algorithm; and, the extended delta-bar-delta (EDBD) algorithm, and the following second-order methods: the QuickProp (QProp) algorithm; and, the MaxProp (MProp) algorithm. The internal parameters for each training algorithm can be selected by the user.
5. Three different transfer functions can be chosen, including: linear; sigmoid; and, hyperbolic tangent transfer functions. These are implemented on a layer by layer basis i.e. each PE in a given layer must have the same transfer function but different layers may have different transfer functions.
6. During training, the network connection weights can be saved at pre-determined intervals.
7. The initial network weights are randomly generated from a uniform distribution within a range set by the user. In this research, the range was chosen to be -0.1 to +0.1 for all ANNs.
8. Cross validation training of networks is supported via the "SaveBest" function.
9. A capability known as UserIO programming is available for customising the performance measure used during cross-validation training.
10. The only facilities that exist for determining the optimal network architecture are the cascade correlation algorithm and a crude weight pruning technique. No evolutionary methods (as discussed in Section 2.4.1.2) are available.

Maier and Dandy (1999) performed an empirical comparison of various methods used to train feedforward MLPs for salinity forecasting. In this study, it was found that first-order methods provided better generalisation ability when compared to second-order methods. Of the first-order methods considered, the extended delta-bar-delta (EDBD) algorithm produced the best results on average. The EDBD algorithm is also recommended by the makers of NeuralWorks Professional II/PLUS. Therefore, this variant of the standard backpropagation algorithm was used as the default training algorithm.

The first step in the proposed modelling methodology involves testing the data for nonlinearity. Since this step does not require the development of an ANN model, no defaults were necessary. Default selections were required for the data transformation, the model inputs, the network architecture, the internal model parameters and the

stopping criteria and performance measure. The default selections used for each of these are outlined below.

4.4.1 Data Transformation

The default transformation used in this research was to take a linear transformation of the data. This is the most commonly employed transformation in the ANN literature and is used to ensure that the data are scaled to a range that is commensurate with the transfer function used in the output layer. The ranges used in conjunction with the hyperbolic tangent transfer function are -1.0 to +1.0 for the network inputs and -0.8 to +0.8 for the network outputs.

4.4.2 Model Inputs

Maier and Dandy (1996b) found that the ANN models trained on the input set shown in Table 4.8 performed best for the salinity case study. Consequently, these 51 inputs were used as the default set of inputs for the ANN models developed in this research.

In the procedure adopted by Maier and Dandy (1996b), *a priori* knowledge was used to define a set of candidate model inputs. Information about salinity travel times and groundwater accessions was used to define this initial set. Future values of flow (i.e. negative lags) were included as they could be obtained using an existing flow forecasting model. Future values of river level at Lock 1 and Overland Corner were also included, as rating curves exist at these locations relating flow to river level, so that the flow forecasts could be used to obtain level forecasts. Inspection of the plots of the input time series combined with sensitivity analyses were used to subsequently reduce the initial set of candidate model inputs. This procedure resulted in the 51-input set shown in Table 4.8.

Table 4.8 Summary of Default Model Inputs

Variable	Location	Abbreviation	Lags (days)	Total No.
Salinity	Murray Bridge	MBS	1, 3, ..., 11	6
Salinity	Mannum	MAS	1, 3, ..., 15	8
Salinity	Morgan	MOS	1, 3, ..., 15	8
Salinity	Waikerie	WAS	1, 2, ..., 5	5
Salinity	Loxton	LOS	1, 2, ..., 5	5
Flow	Overland Corner	OCF	-19, -17, ..., 7	14
Level	Lock 1 Lower	L1LL	-3, -1, ..., 5	5
Total Number of Inputs				51

4.4.3 Network Architecture

The number of nodes in the input and output layers are fixed by the number of inputs and outputs respectively. It is common practice to fix the number of hidden layers in the network and then to choose the number of nodes in each of these layers. It has been shown that only one hidden layer is required to approximate any continuous function, given that sufficient degrees of freedom (i.e. connection weights) are provided (Cybenko, 1989). Hence, one hidden layer was utilised as the default in this study. Maier and Dandy (1998a) conducted empirical trials on the 51-input salinity data set and determined that 30 hidden layer nodes provided optimal performance. Consequently, a network with 51 nodes in the input layer, 30 hidden layer nodes and 1 node in the output layer was used as the default network architecture. Maier and Dandy (1998a) also investigated the effect of using linear, sigmoid and hyperbolic tangent transfer functions. The best generalisation ability was obtained when the hyperbolic tangent transfer function was used, however, the performance of the network trained using the sigmoid transfer function was comparable. The best result obtained when the linear transfer function was used was considerably worse than the other two. Therefore, in the present study, the default transfer function was the hyperbolic tangent transfer function.

4.4.4 Internal Model Parameters

The EDBD training algorithm involves the selection of a number of internal parameters (as discussed in Section 2.4.1.1). However, in practice, the performance of the network is not very sensitive to the choice of these parameters (Smith, 1993). The default values recommended by NeuralWare (1998b) were used as these have been determined using the experience gained from developing networks for a variety of applications. The values for each parameter are shown in Table 4.9.

Table 4.9 Internal Parameters of the EDBD Training Algorithm

EDBD Parameter	Value
Learning rate scale factor (κ_η)	0.095
Learning rate exponential factor (γ_η)	0
Learning rate decrement factor (ϕ_η)	0.1
Momentum scale factor (κ_μ)	0.01
Momentum exponential factor (γ_μ)	0
Momentum decrement factor (ϕ_μ)	0.01
Convex factor (θ)	0.7

Using the 51-input salinity data set, Maier and Dandy (1998a) found that the ANN models were not sensitive to the learning rate, momentum and epoch size. Therefore, the software defaults were also used for these parameters. The default learning rates and momentum are given in Table 4.10, and the default epoch size was 16.

Table 4.10 Default Learning Rates and Momentum

Layer	Learning Rate	Momentum
Input	0.30	0.4
Hidden1	0.25	0.4
Output	0.15	0.4

4.4.5 Stopping Criterion and Performance Measure

To ensure that overtraining did not occur (i.e. when the network performs well on the training data, but poorly on independent test data), cross-validation was used as the stopping criterion. The benefits of using cross-validation as the stopping criterion during ANN training have been discussed by Amari (1997). In this approach, a test set is used to determine the ANN's generalisation ability. The test set root mean squared error (RMSE) is calculated every 1000 iterations and the network with the best test results is saved during the run. After 200,000 iterations with no further improvement in the test set results, training is stopped and the network that performed best on the test set is used as the final model. Since this procedure uses the test data in the calibration phase, an independent validation set was used for all of the methods investigated in order to assess the true generalisation ability of the models developed. The RMSE was used as the performance measure as it places greater emphasis on larger forecasting errors and allowed for comparison with the results obtained in the studies conducted by Maier and Dandy.

4.5 Testing For Nonlinearity

The nonlinearity tests described in Section 3.2 were applied to the salinity time series at Murray Bridge to gain insight into the nature of the data and, consequently, determine whether ANNs provide a suitable modelling methodology for these data. This data set consists of daily salinity levels recorded in the River Murray at Murray Bridge, South Australia during the period 01-12-1986 to 01-04-1998.

4.5.1 Results and Discussion

4.5.1.1 BDS Test

Table 4.11 shows the results obtained when the BDS test was applied to the salinity data set. For this data set linearity was strongly rejected, suggesting that the salinity time series has nonlinear serial dependence. It can also be seen that the inference was robust to the setting of embedding dimension within the 2 to 8 range.

Table 4.11 BDS Test Z Statistics, Residuals from ARMA Fit to Salinity Data Set

Data Set	Fitted ARMA Order (p, q)	Embedding Dimension	BDS Z Statistic	Decision
Salinity at Murray Bridge	(5, 0)	2	29.92	Reject linearity (strongly)
		3	32.29	
		4	32.99	
		5	33.85	
		6	35.35	
		7	37.88	
		8	41.42	

4.5.1.2 Kaboudan's FCS

The results obtained when the FCS was applied to the salinity data set are reported in Table 4.12. It can be seen that these results are in agreement with those obtained using the BDS test, however, some additional information has also been obtained. The FCS correctly diagnosed the nonlinear component of the salinity data set but also identified a strongly linear component. In addition, the FCS was able to identify a medium level of noise associated with the nonlinear component of this data set.

Table 4.12 Diagnosis Results of the FCS for the Salinity Data Set

Data Set	Fitted ARMA Order (p, q)	r^2	θ	Final Diagnosis
Salinity at Murray Bridge	(5, 0)	0.99	0.77	SL-NL-MN[a]

[a]Strongly linear - nonlinear - medium noise

4.5.2 Summary

When the BDS test was applied to the salinity data, these data were diagnosed as significantly nonlinear. When Kaboudan's FCS was applied to the salinity data, they were found to possess a strongly linear component and a nonlinear component with a medium level of noise. Since a multivariate model is being developed in this research, the results from the nonlinearity tests must be interpreted carefully. The output time series (i.e. salinity at Murray Bridge) was found to be nonlinear, and autoregressive (lagged) inputs of this time series are being used in the model. Therefore, an ANN model should be used because the underlying model will still need to be nonlinear even with the addition of the other input variables.

4.6 Division of Data for ANN Models

Two new methods for data division were proposed in Chapter 3, namely the GA data division method and the SOM data division method. In this section, these methods were used to divide the salinity data set into training, testing and validation sets. The new data division methods were also compared to the conventional approach commonly employed in the literature, which involves an arbitrary division of the data. ANN models were developed using the data sets produced by each data division method so that model performance could be compared. A new method was also developed for diagnosing when an ANN is likely to perform poorly. This method is based on the SOM and was applied to the salinity data.

4.6.1 Model Development

The modelling step under investigation in this section is the division of the available data into subsets. Therefore, all other aspects of the modelling process were held constant in accordance with the default selections outlined in Section 4.4.

4.6.1.1 Data Division

The three different data division methods were performed to examine their effect on the ANN's performance. In Method 1, a conventional data division technique was employed, whereby the sets were divided on an arbitrary basis and the statistical properties of the respective data sets were not considered. This approach is consistent with the approach that was employed in many papers on the application of ANNs to water resources variables (Maier and Dandy, 2000b). After accounting for the appropriate lags of each variable, a total of 2005 data records were available for training, testing and validation. From these data records, 1604 records (80%) were used for calibration and 401 records (20%) were used for validation. The time order of the data was not changed (i.e. the first 1604 records were used for calibration and the next 401 records for validation). The 1604 records in the calibration set were further divided into 1283 training records (80%) and 321 testing records (20%). The statistical parameters for the training, testing and validation sets obtained using Method 1 are shown in Table 4.13. From Table 4.13 it can be seen that the statistical parameters vary widely between training, testing and validation sets and, in general, the statistics are not in good agreement. Hypothesis tests about the difference between the means of two samples (t-test) and about the difference in the variance of two samples (F-test) were performed and are shown in Table 4.14. For each input and output variable, the testing and validation data sets were compared with the training sets and a significance level of

$\alpha = 0.05$ was chosen. In the *t*-test it was hypothesized that there was no difference between the means of the two data sets. Likewise, in the *F*-test it was hypothesized that there was no difference in the variances of the two data sets. For the data division performed using Method 1, both of these hypotheses were rejected for each variable in the testing and validation sets.

In Method 2, the GA data division technique was used. When using this technique, it is still necessary to determine the proportion of data to use for each of the subsets. For consistency, the same proportion used using Method 1 was employed (1283 training records, 321 testing records and 401 validation records). The statistical parameters for the training, testing and validation sets obtained in Method 2 are shown in Table 4.15. It can be seen that the statistics are in good agreement. Table 4.15 also shows that for each variable, the training set contains the maximum and minimum values. The only exception is the Morgan Salinity variable, in which the training set contains the maximum value but not the minimum value. Hypothesis tests were also performed for the data sets obtained using Method 2 (Table 4.16). The null hypotheses were not rejected for all variables except that the *F*-test null hypothesis was rejected at the 0.05 level for the Loxton salinity test set. This suggests that the Loxton salinity variability in this test set was different to the training set. With the exception of this one variable in the test set, the GA data division technique used in Method 2 was able to produce three data sets that were representative of the same population.

Method 3 employed the use of the SOM data division technique. This technique avoids the need to arbitrarily select which proportion of data to include in each subset since only a minimum number of records are required (i.e. one record from each cluster is used for each subset) and superfluous data records are not used in the ANN. Using a SOM with a 10 x 10 Kohonen layer, the 2005 data samples were clustered into 49 clusters, with each cluster consisting of more than 3 records. Hence, the training, testing and validation sets each comprised of 49 records. The statistical parameters for the training, testing and validation sets obtained using Method 3 are shown in Table 4.17. Once again, it can be seen that the statistics are in good agreement. The *t*-test and *F*-test null hypotheses were not rejected for all input and output variables (Table 4.18). Hence, the statistics of all three data sets can be assumed to be the same.

Table 4.13 Method 1: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using Conventional Method)

Variable and Data Set	Mean	Standard Dev.	Max.	Min.	Inter-quartile Range (IQR)
Input Variable 1: Murray Bridge Salinity (EC units)					
Training	588.0	172.9	931.0	261.3	291.6
Testing	515.7	212.2	862.9	262.3	444.7
Validation	694.7	213.6	1115.7	282.2	342.1
Input Variable 2: Mannum Salinity (EC units)					
Training	574.5	165.7	960.4	281.1	281.7
Testing	495.2	205.0	826.6	253.2	438.4
Validation	661.4	192.4	1075.4	269.7	265.7
Input Variable 3: Morgan Salinity (EC units)					
Training	572.0	170.0	921.3	176.9	320.2
Testing	498.7	201.9	798.9	258.6	420.4
Validation	670.7	205.6	1061.3	249.4	307.4
Input Variable 4: Waikerie Salinity (EC units)					
Training	562.5	160.7	880.3	278.9	289.9
Testing	503.5	196.3	796.6	254.8	421.1
Validation	660.3	190.8	1021.0	247.4	265.2
Input Variable 5: Loxton Salinity (EC units)					
Training	491.1	107.3	698.3	263.3	181.5
Testing	449.6	155.0	697.4	244.1	328.0
Validation	583.4	144.5	906.7	224.7	162.1
Input Variable 6: Overland Corner Flow (ML/day)					
Training	19956	21620	85963	1796	23995
Testing	41318	36690	110618	4764	59893
Validation	12678	13477	46664	1769	15828
Input Variable 7: Lock 1 Lower River Level (m)					
Training	4.0	1.0	4.4	0.5	1.2
Testing	2.5	1.6	5.3	0.7	2.9
Validation	1.1	0.7	2.9	0.6	0.7
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)					
Training	588.8	171.6	931.0	310.6	291.6
Testing	535.9	217.6	862.9	262.3	452.6
Validation	698.3	216.5	1115.7	282.2	353.6

Table 4.14 Method 1 (Data Divided Using Conventional Method): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set

Variable and Data Set	t-value	t-crit. α=0.05	t-test H_o: $\mu_1 = \mu_2$	F-value	F-crit. α=0.05	F-test H_o: $\sigma_1^2 = \sigma_2^2$
Input Variable 1: Murray Bridge Salinity (EC units)						
Testing	5.65	1.96	reject	1.51	1.18	reject
Validation	9.11	1.96	reject	1.53	1.17	reject
Input Variable 2: Mannum Salinity (EC units)						
Testing	6.43	1.96	reject	1.53	1.18	reject
Validation	8.15	1.96	reject	1.35	1.17	reject
Input Variable 3: Morgan Salinity (EC units)						
Testing	5.99	1.96	reject	1.41	1.18	reject
Validation	8.73	1.96	reject	1.46	1.17	reject
Input Variable 4: Waikerie Salinity (EC units)						
Testing	4.98	1.96	reject	1.49	1.18	reject
Validation	9.29	1.96	reject	1.41	1.17	reject
Input Variable 5: Loxton Salinity (EC units)						
Testing	4.53	1.96	reject	2.09	1.18	reject
Validation	11.81	1.96	reject	1.81	1.17	reject
Input Variable 6: Overland Corner Flow (ML/day)						
Testing	10.01	1.96	reject	2.88	1.18	reject
Validation	8.05	1.96	reject	2.57	1.18	reject
Input Variable 7: Lock 1 Lower River Level (m)						
Testing	16.03	1.96	reject	2.56	1.18	reject
Validation	64.82	1.96	reject	2.04	1.18	reject
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)						
Testing	4.05	1.96	reject	1.61	1.18	reject
Validation	9.26	1.96	reject	1.59	1.17	reject

Table 4.15 Method 2: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using a GA)

Variable and Data Set	Mean	Standard Dev.	Max.	Min.	Inter-quartile Range (IQR)
Input Variable 1: Murray Bridge Salinity (EC units)					
Training	596.6	197.0	1115.7	261.3	369.1
Testing	604.3	201.8	1077.8	265.8	390.9
Validation	596.4	188.4	1045.4	265.2	347.8
Input Variable 2: Mannum Salinity (EC units)					
Training	579.5	186.6	1075.4	253.2	344.2
Testing	578.9	188.2	1072.8	255.8	360.4
Validation	578.5	176.5	982.5	259.4	306.7
Input Variable 3: Morgan Salinity (EC units)					
Training	581.0	190.1	1061.3	182.8	341.0
Testing	575.8	199.2	1043.9	206.1	362.6
Validation	580.4	183.5	1022.7	176.9	333.6
Input Variable 4: Waikerie Salinity (EC units)					
Training	573.6	178.5	1021.0	247.4	314.4
Testing	569.7	191.7	1008.3	251.8	344.1
Validation	571.7	174.2	1005.2	257.6	295.5
Input Variable 5: Loxton Salinity (EC units)					
Training	503.3	130.0	906.7	224.7	218.4
Testing	502.0	142.4	900.4	228.8	247.5
Validation	502.4	126.1	857.8	244.1	200.3
Input Variable 6: Overland Corner Flow (ML/day)					
Training	21770	24932	110618	1769	25349
Testing	22816	25785	110056	1820	26417
Validation	21685	25134	110496	1943	24196
Input Variable 7: Lock 1 Lower River Level (m)					
Training	1.6	1.2	5.3	0.5	1.3
Testing	1.6	1.2	5.3	0.5	1.3
Validation	1.6	1.2	5.3	0.5	1.3
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)					
Training	603.0	195.4	1115.7	262.3	366.7
Testing	598.1	205.7	965.7	264.4	400.3
Validation	603.0	190.3	1102.5	263.4	352.8

Table 4.16 Method 2 (Data Divided Using a GA): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set

Variable and Data Set	*t*-value	*t*-crit. α=0.05	*t*-test H_o: $\mu_1 = \mu_2$	*F*-value	*F*-crit. α=0.05	*F*-test H_o: $\sigma_1^2 = \sigma_2^2$
Input Variable 1: Murray Bridge Salinity (EC units)						
Testing	0.61	1.96	don't reject	1.05	1.18	don't reject
Validation	0.02	1.96	don't reject	1.09	1.18	don't reject
Input Variable 2: Mannum Salinity (EC units)						
Testing	0.05	1.96	don't reject	1.02	1.18	don't reject
Validation	0.10	1.96	don't reject	1.12	1.18	don't reject
Input Variable 3: Morgan Salinity (EC units)						
Testing	0.42	1.96	don't reject	1.10	1.18	don't reject
Validation	0.06	1.96	don't reject	1.07	1.18	don't reject
Input Variable 4: Waikerie Salinity (EC units)						
Testing	0.33	1.96	don't reject	1.15	1.18	don't reject
Validation	0.19	1.96	don't reject	1.05	1.18	don't reject
Input Variable 5: Loxton Salinity (EC units)						
Testing	0.15	1.96	don't reject	1.20	1.18	reject
Validation	0.12	1.96	don't reject	1.06	1.18	don't reject
Input Variable 6: Overland Corner Flow (ML/day)						
Testing	0.65	1.96	don't reject	1.07	1.18	don't reject
Validation	0.06	1.96	don't reject	1.02	1.18	don't reject
Input Variable 7: Lock 1 Lower River Level (m)						
Testing	0	1.96	don't reject	1.00	1.18	don't reject
Validation	0	1.96	don't reject	1.00	1.18	don't reject
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)						
Testing	0.39	1.96	don't reject	1.11	1.18	don't reject
Validation	0	1.96	don't reject	1.05	1.18	don't reject

Table 4.17 Method 3: Statistics of the Salinity Training, Testing and Validation Data Sets (Data Divided Using a SOM)

Variable and Data Set	Mean	Standard Dev.	Max.	Min.	Inter-quartile Range (IQR)
Input Variable 1: Murray Bridge Salinity (EC units)					
Training	587.7	202.4	981.7	267.2	346.8
Testing	586.5	202.5	1004.8	261.3	357.0
Validation	587.7	201.7	1047.0	263.9	333.9
Input Variable 2: Mannum Salinity (EC units)					
Training	575.8	191.5	966.2	281.1	345.0
Testing	577.6	191.3	992.5	282.2	327.5
Validation	588.3	192.8	1075.4	286.1	320.6
Input Variable 3: Morgan Salinity (EC units)					
Training	578.4	194.6	1031.8	282.8	345.0
Testing	579.8	192.0	1011.4	290.3	317.2
Validation	574.6	185.4	987.3	287.3	320.3
Input Variable 4: Waikerie Salinity (EC units)					
Training	570.2	184.2	937.2	278.9	327.0
Testing	569.0	181.4	943.1	284.4	314.7
Validation	567.1	176.6	946.4	293.5	301.0
Input Variable 5: Loxton Salinity (EC units)					
Training	494.2	132.1	777.1	263.3	236.3
Testing	496.5	129.8	783.5	279.8	214.3
Validation	494.5	129.2	784.0	274.9	218.6
Input Variable 6: Overland Corner Flow (ML/day)					
Training	21294	22483	97229	2240	22595
Testing	21518	22886	99065	2232	22097
Validation	21872	23374	100676	2069	23606
Input Variable 7: Lock 1 Lower River Level (m)					
Training	1.6	1.0	4.5	0.6	1.2
Testing	1.6	1.0	4.6	0.6	1.1
Validation	1.6	1.0	4.7	0.6	1.1
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)					
Training	573.0	191.7	922.9	300.6	338.9
Testing	576.1	187.9	921.1	286.8	343.8
Validation	578.1	185.4	919.2	284.4	317.6

Table 4.18 Method 3 (Data Divided Using a SOM): Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set

Variable and Data Set	*t*-value	*t*-crit. α=0.05	*t*-test H_o: $\mu_1 = \mu_2$	*F*-value	*F*-crit. α=0.05	*F*-test H_o: $\sigma_1^2 = \sigma_2^2$
Input Variable 1: Murray Bridge Salinity (EC units)						
Testing	0.09	1.96	don't reject	1.00	1.18	don't reject
Validation	0	1.96	don't reject	1.01	1.18	don't reject
Input Variable 2: Mannum Salinity (EC units)						
Testing	0.15	1.96	don't reject	1.00	1.19	don't reject
Validation	1.14	1.96	don't reject	1.01	1.17	don't reject
Input Variable 3: Morgan Salinity (EC units)						
Testing	0.12	1.96	don't reject	1.03	1.19	don't reject
Validation	0.35	1.96	don't reject	1.10	1.18	don't reject
Input Variable 4: Waikerie Salinity (EC units)						
Testing	0.11	1.96	don't reject	1.03	1.19	don't reject
Validation	0.30	1.96	don't reject	1.09	1.18	don't reject
Input Variable 5: Loxton Salinity (EC units)						
Testing	0.28	1.96	don't reject	1.04	1.19	reject
Validation	0.04	1.96	don't reject	1.05	1.18	don't reject
Input Variable 6: Overland Corner Flow (ML/day)						
Testing	0.16	1.96	don't reject	1.04	1.18	don't reject
Validation	0.44	1.96	don't reject	1.08	1.17	don't reject
Input Variable 7: Lock 1 Lower River Level (m)						
Testing	0	1.96	don't reject	1.00	1.18	don't reject
Validation	0	1.96	don't reject	1.00	1.17	don't reject
Output 1: Murray Bridge Salinity at (t + 14) days (EC units)						
Testing	0.26	1.96	don't reject	1.04	1.19	don't reject
Validation	0.48	1.96	don't reject	1.07	1.18	don't reject

4.6.2 Results and Discussion

The RMSEs of the 14-day forecasts for the data sets obtained using each data division method are summarised in Table 4.19. It can be seen that the model developed using Method 1 produced a much larger error on the testing and validation sets than the models developed using Methods 2 and 3. However, in Method 1, the model performed well on the training set. This suggests that the arbitrary data division used in Method 1 produced a training set that was not representative of the entire population, which is in agreement with the statistics shown in Table 4.13. In Table 4.19 it is also important to observe that unlike the model developed using Method 1, the models developed using Methods 2 and 3 produced results that were consistent for all three data sets. Again, this is in agreement with the statistics obtained in Table 4.16 and Table 4.18. The GA data set division produced results with RMSEs that ranged from 33.9 to 39.1 EC units. The SOM data set division produced results with RMSEs ranging from 35.5 to 39.1 EC units. This is in direct contrast to the model developed in Method 1, which produced results over a much wider range, including RMSEs ranging from 38.2 to 58.8 EC units. The results obtained using an arbitrary division of the data are likely to be significantly different each time it is performed, unlike the results obtained using data division Methods 2 and 3, which tend to produce very consistent results.

Table 4.19 RMSEs for the 14-Day Forecasts

Data Set	Training Set RMSE (EC units)	Testing Set RMSE (EC units)	Validation Set RMSE (EC units)
Method 1: Conventional Division	38.2	51.6	58.8
Method 2: GA Division	39.1	33.9	38.8
Method 3: SOM Division	39.1	37.1	35.5

The above results show that the GA-based approach and the SOM approach are suitable methods for ensuring that the training, testing and validation sets are representative of the same population and hence, provide an appropriate means for data division. The SOM approach has the additional advantage that a training set can be constructed using the minimum number of samples. The results also show that for good performance, the data sets need to have similar statistical properties.

To understand why the ANN developed using Method 1 performed poorly, it is necessary to critically dissect the model and examine the regions of poor performance. The SOM provides a useful technique for achieving this purpose. Figure 4.37 shows the validation results of the ANN developed using Method 1, with 4 regions of poor performance identified. An analysis was conducted by clustering all of the input and output data from the training, testing and validation sets using a SOM. Once the clusters had been formed, it was possible to examine each region of poor performance and determine if any data representative of that region had been included in the training set. This was done by inspecting the clusters. If the training set did not contain data representative of that region, then it is expected that the model may perform poorly, since the model had not been trained for this event.

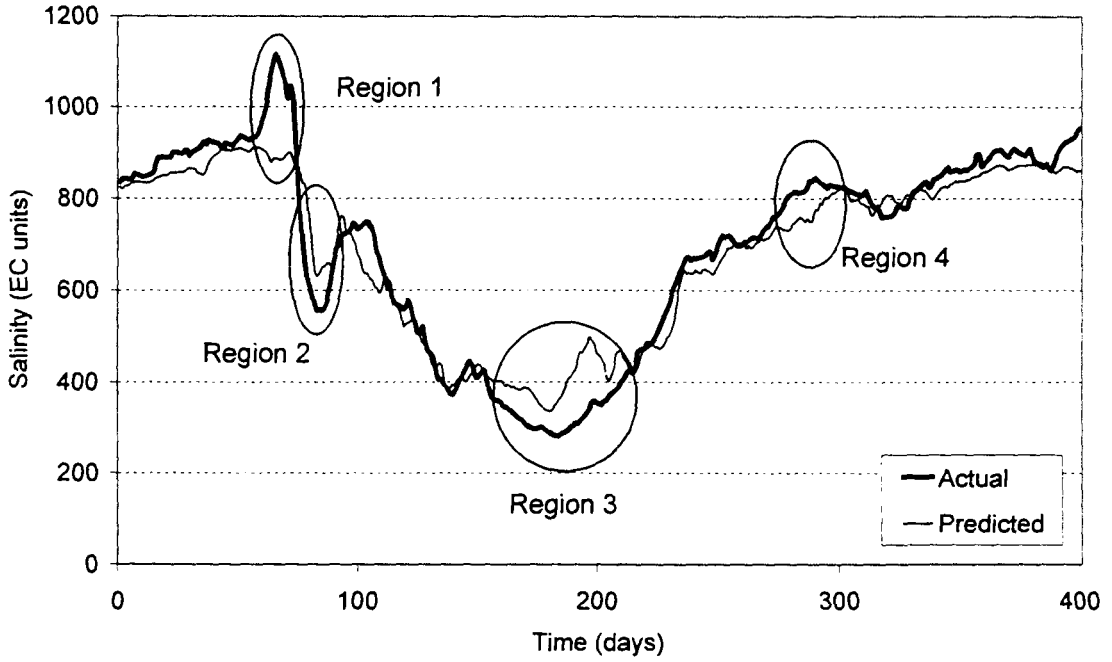

Figure 4.37 Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 1 (May 1991 to June 1992)

After performing the analysis for Region 1 (Figure 4.37), it was found that all of the data in this region were located in a single cluster. This single cluster only contained validation data and hence, the observed pattern was not represented in the training set. This explains why the model developed using Method 1 was unable to match the peak observed in Region 1. The data contained in Region 2 (Figure 4.37) were split into 2 clusters in the SOM. However, both of these clusters only contained validation data. Region 3 data (Figure 4.37) were divided into 6 clusters. Four of these clusters only contained validation data and the remaining 2 clusters contained a small amount of training data. Region 4 data (Figure 4.37) were split into 2 clusters, both of which only

contained validation data. By using a SOM to analyse the regions of poor performance, it was possible to verify that the major cause of the poor performance was due to a lack of representative data in the training set. However, it should be noted that there are other possible factors that may have contributed to the poor model performance in these regions. For example, the model may simply be unable to predict the deterministic component of the data in these regions due to the long forecast lead-time of 14 days.

In a real-world scenario, the developed ANN model would need to produce forecasts based on new data, the statistics of which are unknown. In addition, each of the data division methods investigated produced different training, testing and validation sets. To fairly evaluate and compare the performance of each technique, it was considered necessary to test each model on the same data set. Hence, a second, independent validation data set was used, consisting of daily data from the period 15-07-1992 to 13-03-1998. The models developed using Methods 1, 2 and 3 were used to obtain 14-day forecasts for this second validation set. Plots of the 14-day forecasts obtained when Methods 1, 2 and 3 were used are shown in Figure 4.38, Figure 4.39 and Figure 4.40, respectively.

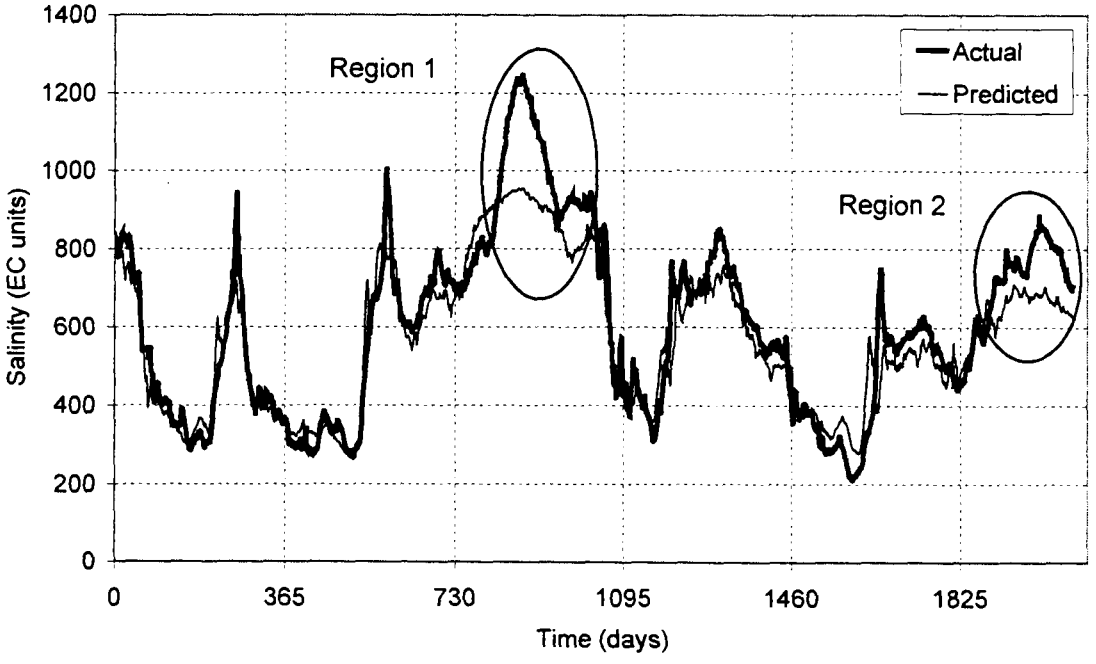

Figure 4.38 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 1 (July 1992 to March 1998)

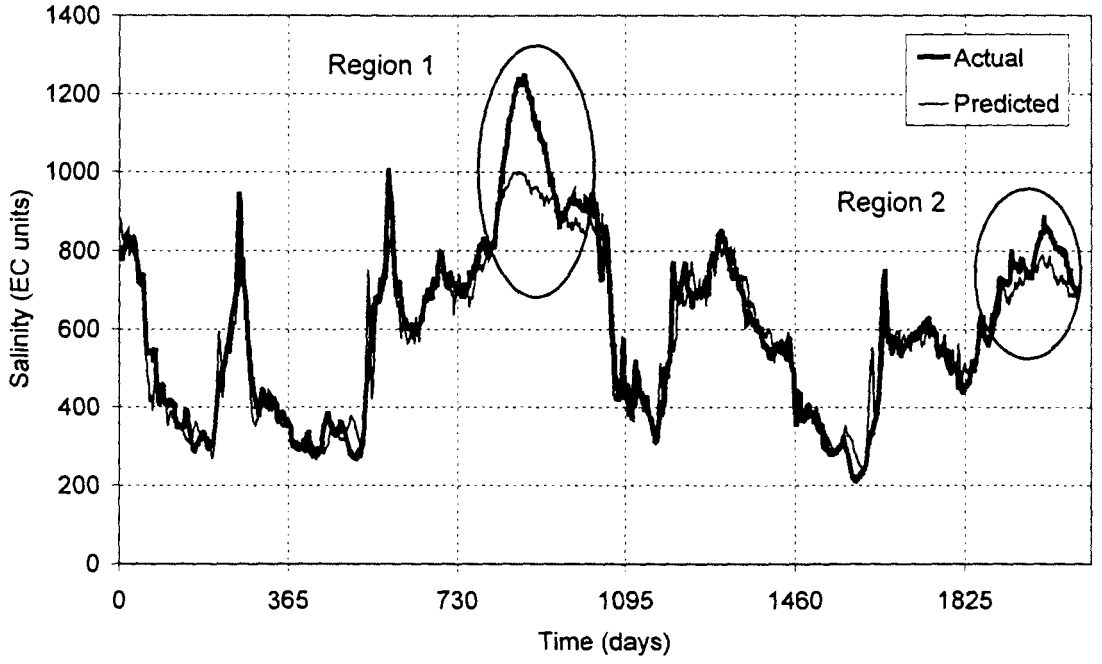

Figure 4.39 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 2 (July 1992 to March 1998)

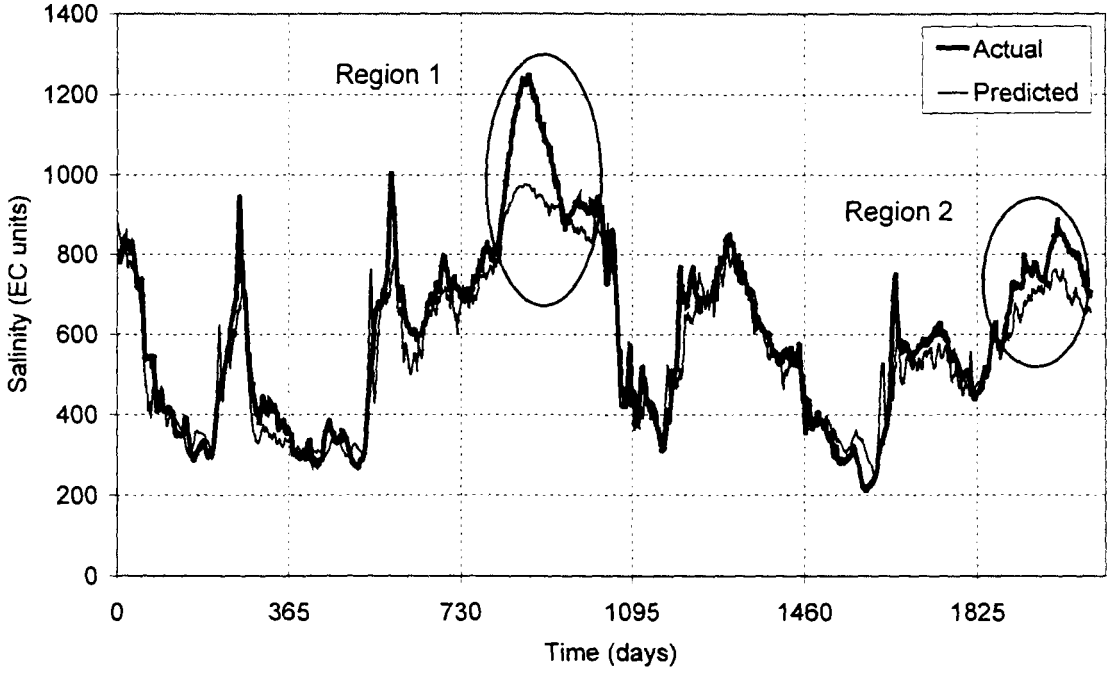

Figure 4.40 Second Validation Set 14-Day Forecast of Salinity at Murray Bridge for the Model Developed in Method 3 (July 1992 to March 1998)

It can be seen that the most notable regions of poor performance have been identified as Regions 1 and 2. The models developed using all three methods performed poorly in

both Regions 1 and 2, and to investigate the cause of this poor performance, the SOM technique was used to analyse the data. To perform the analysis, the data from the second validation set were combined with all of the data used in the development of the models (i.e. the training, testing and validation data). The SOM was used to cluster the data and each of the clusters was inspected. It was discovered that all of the data from Region 1 were contained in 6 clusters, however, all of these clusters only contained data from the second validation set. Hence, no data representative of Region 1 had been used to develop (i.e. train, test or validate) any of the models. This explains why the models developed using all three methods performed poorly on this region. Likewise, the data from Region 2 were contained in 6 clusters, 5 of which only contained data from the second validation set. One cluster contained 7 samples from the second validation set as well as training data. The models developed using Methods 2 and 3 were able to perform well on these 7 points.

The performances of the models developed using the 3 methods for the second validation set are summarised in Table 4.20. To obtain a fair representation of each model's performance, another calculation of the RMSEs was performed with the data from Regions 1 and 2 removed, since these were unique data points and no data representative of these 2 regions had been used to train the models. Even after removing Regions 1 and 2, the results shown in Table 4.20 are not as good as those obtained in Table 4.19. The larger RMSE values obtained on the second validation set may be attributed to the very large size of the data set, which was comprised of 2068 records. This highlights the importance of periodically retraining the ANN model at regular intervals when it is to be used in a real-world situation. It should be noted that when retraining the model there is a delay time of 14 days, which must elapse before retraining can take place for a given sample. It is expected that if retraining is undertaken, the RMSE would be better than that obtained when there is no retraining (i.e. the first column results in Table 4.20). However, the RMSE obtained after retraining is still likely to be worse than the RMSE obtained when Region 1 and 2 are simply removed (i.e. the second column results in Table 4.20). This is because the retraining will not enable perfect prediction of the deterministic component of Regions 1 and 2, because of the delay time (14 days), which must elapse before retraining can commence.

Table 4.20 RMSEs for the 14-Day Forecasts

Data Set	Second Validation Set	Second Validation Set (Regions 1 and 2 Removed)
	RMSE (EC units)	RMSE (EC units)
Method 1: Conventional Division	86.1	67.7
Method 2: GA Division	65.3	51.8
Method 3: SOM Division	77.6	62.6

From Table 4.20, it can be seen that the GA data division (Method 2) performed best out of the 3 techniques. The SOM data division (Method 3) slightly outperformed the conventional approach (Method 1). However, it is worth pointing out that the model developed using the SOM data division technique was only trained on 49 data records, whereas the models developed using Methods 1 and 2 were trained using 1283 data records. The ability of the SOM technique to outperform the conventional method, whilst only using 49 training samples, provides further evidence of its superiority at developing a representative training set with a minimum number of samples.

To investigate whether 49 training samples were able to capture enough of the variance in the available data set, an ANN model was developed using 1 data sample from each cluster for the testing and validation sets, while using the remaining data samples in the training set. However, this only increased the noise in the forecasts and when tested on the second validation set, no improvement was made over the model developed using 49 training samples. This suggests that the 49 samples were able to provide an adequate approximation of all samples contained in the available data set.

4.7 Data Transformation

A step that should be considered when developing ANN models for water resources applications is the selection of an appropriate transformation of the data. In general, the primary motivations for data transformation are:

- To scale the data so as to be commensurate with the transfer function in the output layer.
- To standardise each of the variables.
- To provide a suitable initialisation of the ANN.
- To modify the distribution of the input variables to provide a better mapping to the outputs.

Six different data transformations were presented and discussed in Chapter 3. These were: linear transformation, logarithmic transformation, histogram equalization, kernel transformation, seasonal standardisation and transformation to normality. To investigate the effect that each transformation has on ANN performance, the salinity case study was used to develop ANN models. The performance of each model resulting from each transformation was then compared and discussed.

4.7.1 Model Development

In Section 4.6, the GA and SOM data division methods were found to give comparable performance. A disadvantage of the GA data division method is that the proportion of data assigned to the training, testing and validation sets must be chosen, whereas the SOM data division avoids this. However, the SOM data division method is slightly more difficult to implement as it requires the selection of the Kohonen layer (grid size) and the choice of which samples to discard from each cluster. Therefore, the GA data division method was adopted for the remaining modelling steps as it is slightly easier to implement. All other aspects of the modelling process were held constant in accordance with the default selections outlined in Section 4.4.

The main objective of this component of the research was to compare different data transformation techniques. Therefore, by using the GA to divide the data, a fair comparison could be made between the different models. This is because the models are tested and validated on data that are statistically representative of the data used in the training process. This provides the most rigorous test of a model's performance based on the data transformation method, since other sources of poor performance such as attempting to validate the model on data outside the range used in training are

effectively eliminated. In this section, the proportion of data assigned to the training, testing and validation sets was the same as that described in Section 4.6.1.1 for the GA data division.

4.7.1.1 Data Transformation

The six transformations outlined in Section 3.4 were investigated in this research. These include: linear transformation, logarithmic transformation, histogram equalization, kernel transformation, seasonal standardisation and transformation to normality. It is important to note that only the calibration data (i.e. training and testing data) were used to determine the parameters required by each transformation, thereby leaving the validation data as an independent set to compare each transformation technique.

Before transforming each variable, it is first necessary to consider the distribution of the raw data. The BestFit statistical analysis software package (Palisade Corp., 1997) was used in an attempt to fit a distribution to each input variable. BestFit includes 28 distribution types, however, it was not possible to fit any of these distributions at an acceptable level of significance to any of the variables used in this research.

Histogram plots for each variable used in this investigation are shown in Figure 4.41 to Figure 4.47. Also shown on each of the plots is the Gaussian expectation, which allows comparison of each histogram with the normal distribution. As expected, the distribution of each salinity variable (Figure 4.41 to Figure 4.45) is similar. The distribution appears to be bimodal, with the possible exception of salinity at Loxton (Figure 4.45). The distribution of flow at Overland Corner (Figure 4.46) and river level at Lock 1 Lower (Figure 4.47) are both heavily skewed with an extended right hand tail. This type of distribution is common for these types of hydrological data. A logarithmic transformation may prove useful in compressing the distribution of these two variables.

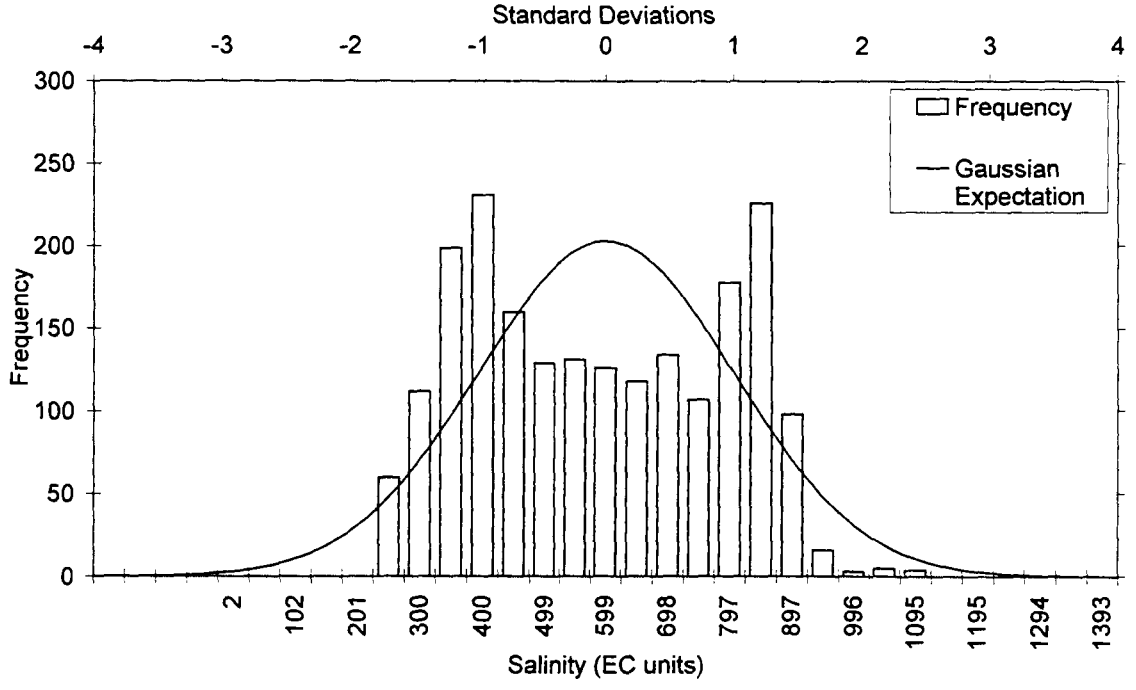

Figure 4.41 Histogram of Salinity at Murray Bridge (Raw Data)

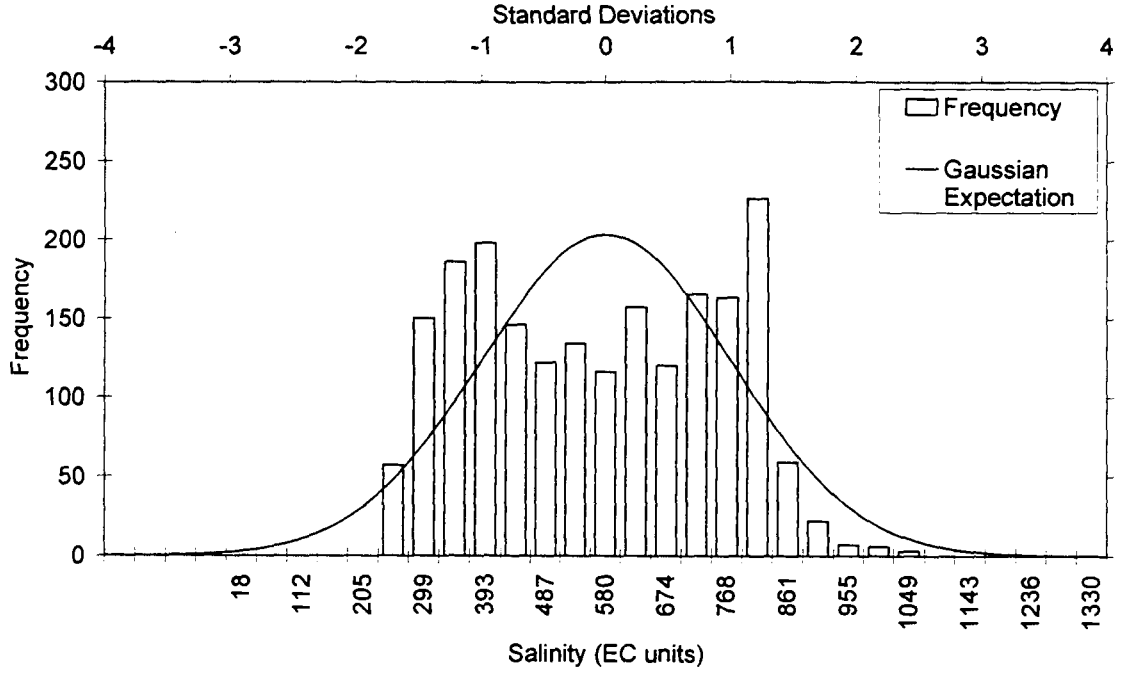

Figure 4.42 Histogram of Salinity at Mannum (Raw Data)

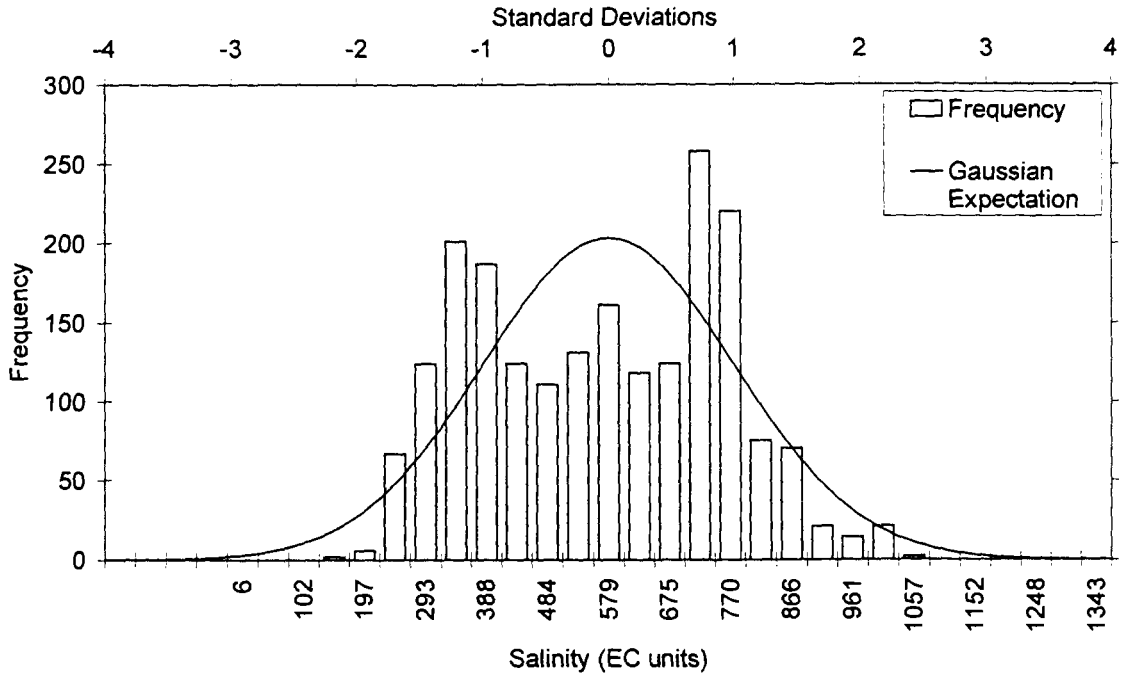

Figure 4.43 Histogram of Salinity at Morgan (Raw Data)

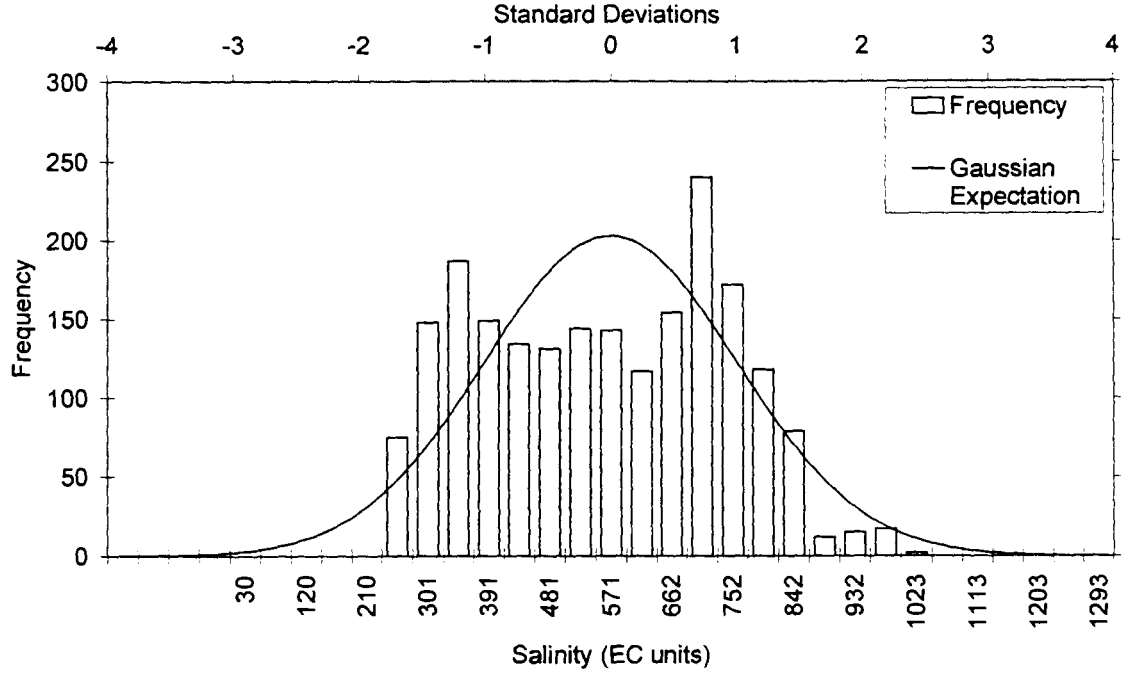

Figure 4.44 Histogram of Salinity at Waikerie (Raw Data)

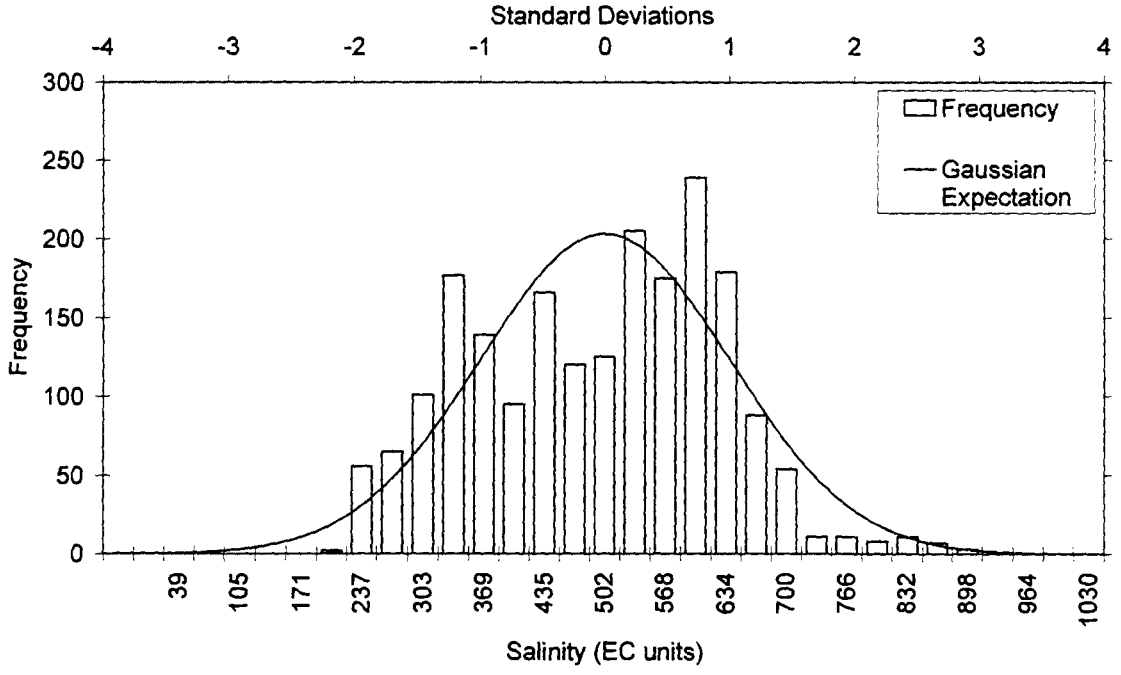

Figure 4.45 Histogram of Salinity at Loxton (Raw Data)

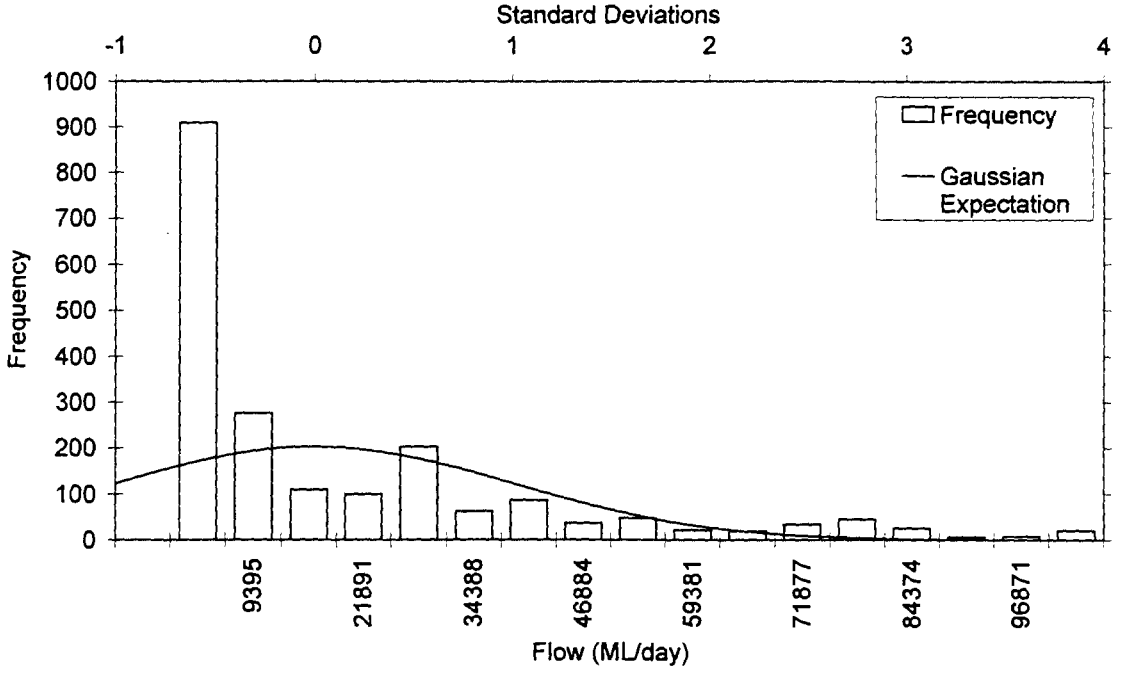

Figure 4.46 Histogram of Flow at Overland Corner (Raw Data)

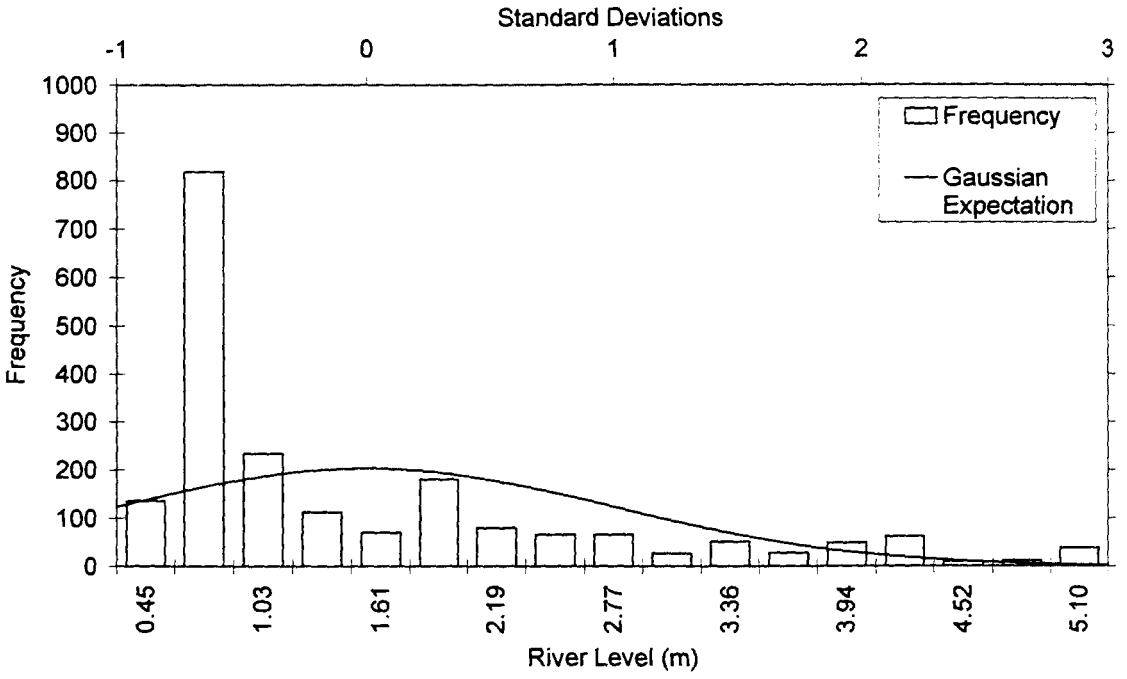

Figure 4.47 Histogram of River Level at Lock 1 Lower (Raw Data)

Linear transformation

The salinity data were linearly transformed by using the original data range to rescale the series to a range that was commensurate with the output transfer function. Since the hyperbolic tangent transfer function was used in the output layer, the network inputs were scaled between -1.0 and +1.0, and network outputs were scaled between -0.8 and +0.8. Linear transformation does not alter the distribution of the variable and therefore, histogram plots of the linearly transformed data are not presented here.

Logarithmic transformation

A logarithmic transformation of the salinity data was performed to determine the effect of a compressing transform on network training. Histograms of salinity at Murray Bridge, flow at Overland Corner and river level at Lock 1 Lower, after the logarithmic transformation has been applied, are shown in Figure 4.48, Figure 4.49 and Figure 4.50, respectively. For brevity, histograms of the other upstream salinity time series data were not shown here as they were similar to the histogram obtained for salinity at Murray Bridge (Figure 4.48). Most notably, the logarithmic transformation was able to compress the distribution of the flow (Figure 4.49) and river level (Figure 4.50) variables, however, after transformation, both distributions appear to be multimodal.

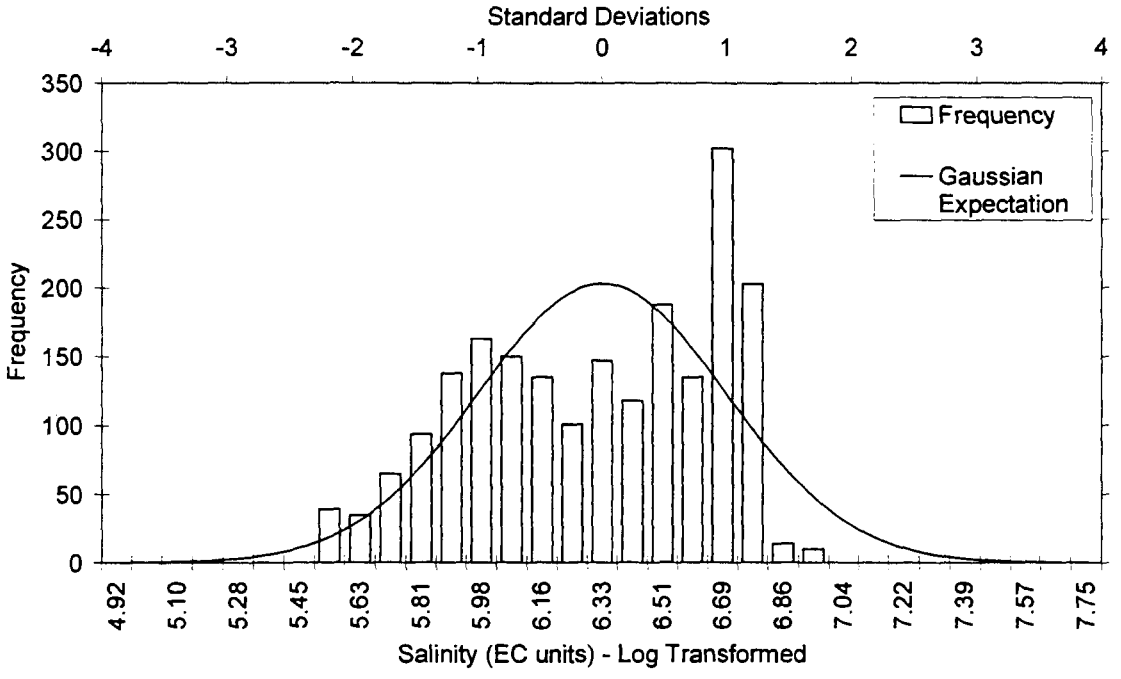

Figure 4.48 Histogram of Salinity at Murray Bridge (Log Transformed Data)

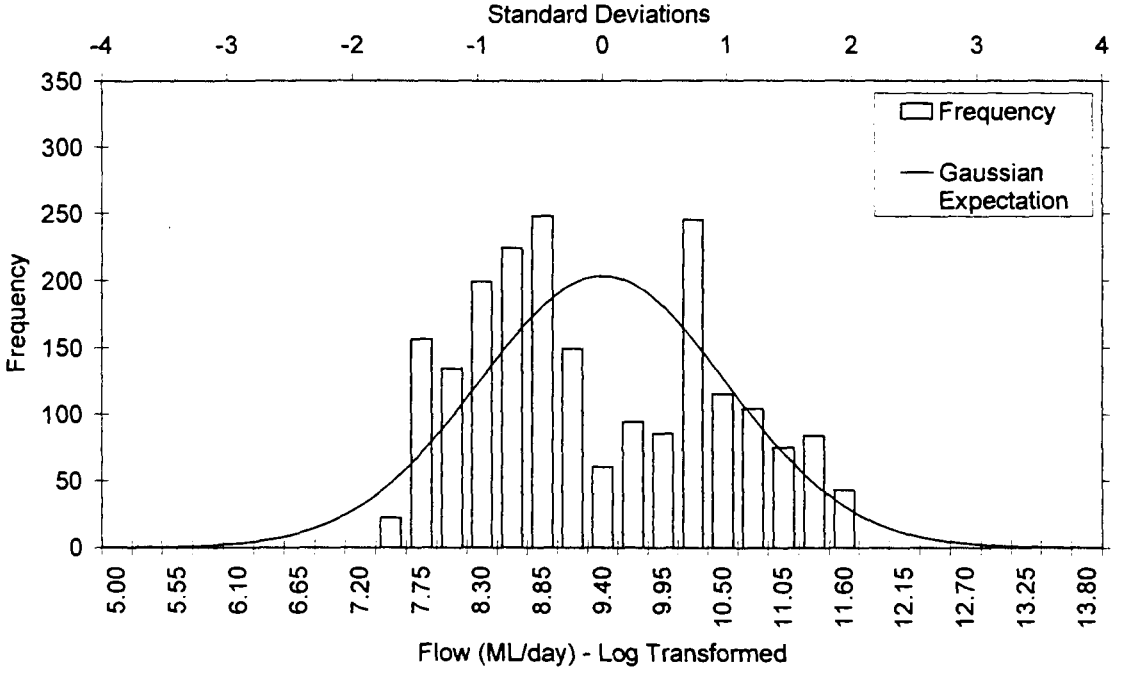

Figure 4.49 Histogram of Flow at Overland Corner (Log Transformed Data)

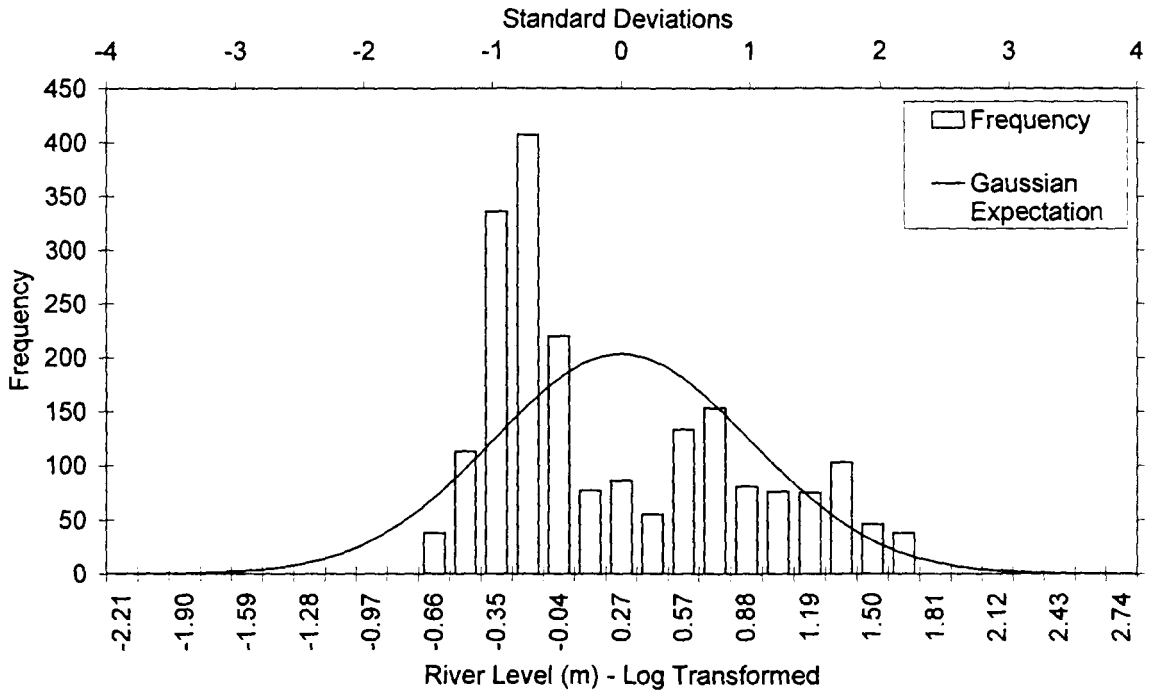

Figure 4.50 Histogram of River Level at Lock 1 Lower (Log Transformed Data)

Histogram equalization

The third transformation investigated was histogram equalization, which produces data that are approximately uniformly distributed. It has been suggested in the literature (e.g. Shi, 2000) that the use of uniformly distributed inputs in an ANN model allows for a better mapping to the output variable. Histograms of salinity at Murray Bridge, flow at Overland Corner and river level at Lock 1 Lower, after the histogram equalization, are shown in Figure 4.51, Figure 4.52, and Figure 4.53, respectively. Histograms of the other upstream salinity time series data were not shown here as they were similar to the histogram obtained for salinity at Murray Bridge. It is apparent from the plots that histogram equalization was successful in transforming the distribution of each input variable to approximate uniformity.

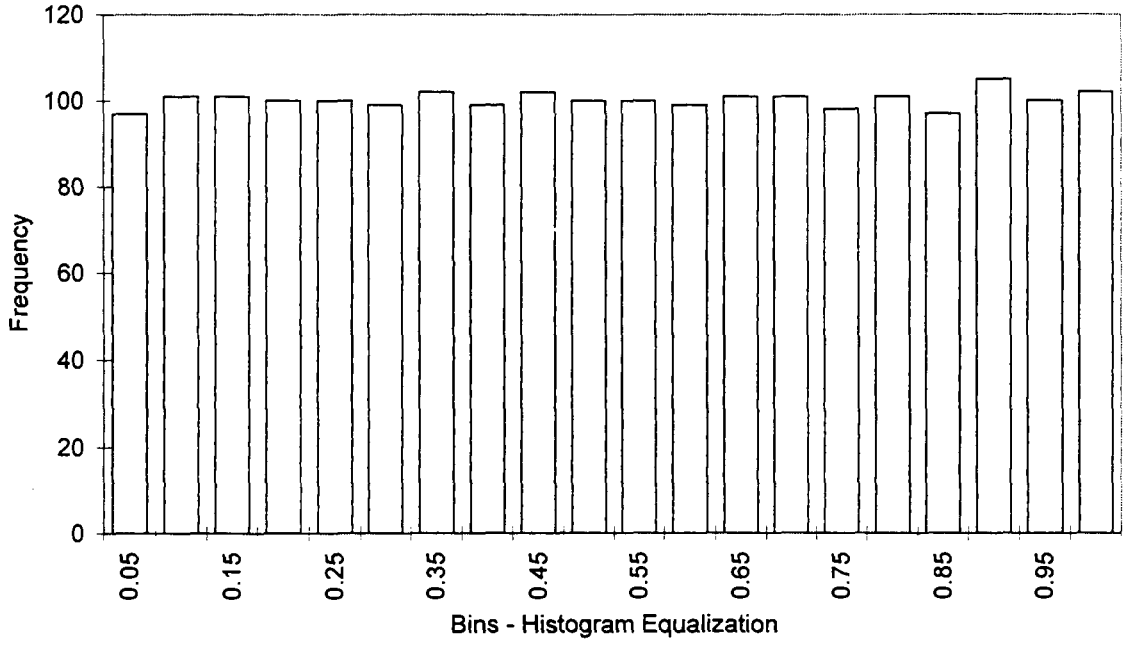

Figure 4.51 Histogram of Salinity at Murray Bridge (Histogram Equalization Transformed Data)

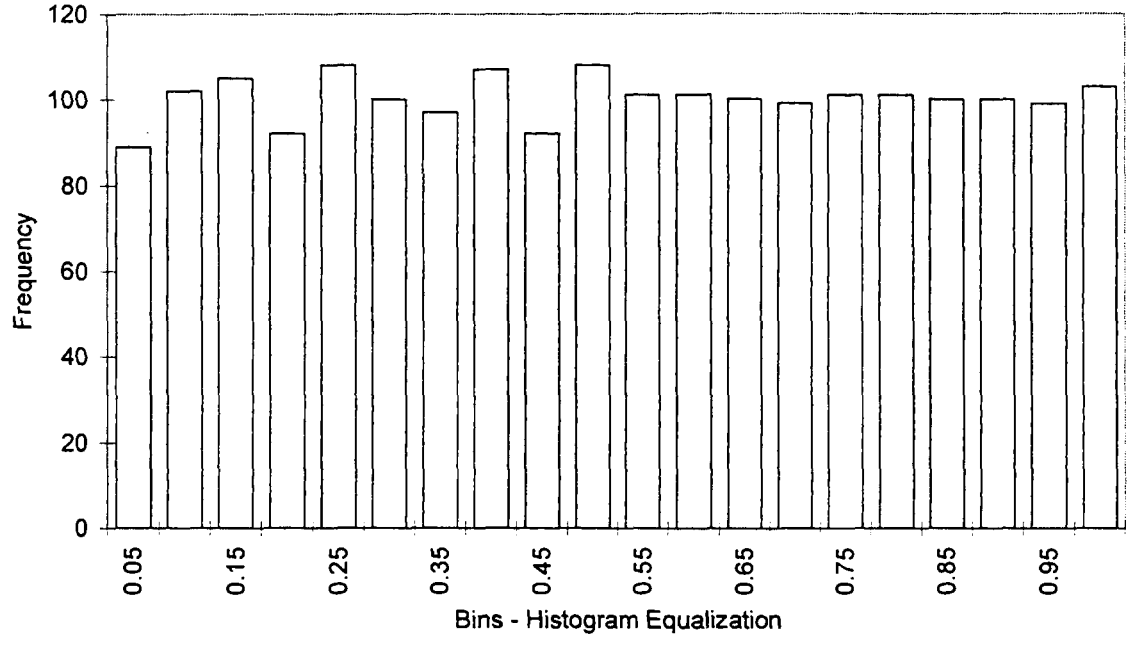

Figure 4.52 Histogram of Flow at Overland Corner (Histogram Equalization Transformed Data)

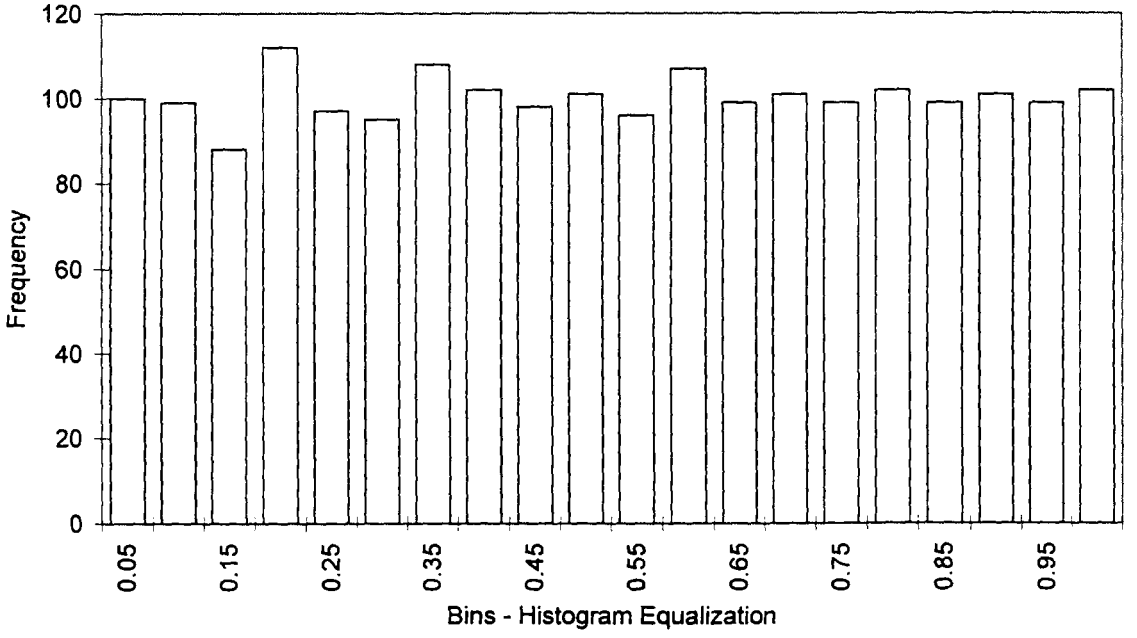

Figure 4.53 Histogram of River Level at Lock 1 Lower (Histogram Equalization Transformed Data)

Kernel transformation

Kernel transformation is similar to histogram equalization in that the goal is to produce data that are approximately uniformly distributed. However, kernels are used rather than a histogram to approximate the distribution of the raw data. Histograms of the time series data transformed using kernel transformation are not shown here as they were similar to the histograms obtained using histogram equalization. Like histogram equalization, kernel transformation was also successful in producing data that were approximately uniformly distributed.

Seasonal standardisation

The aim of seasonal standardisation was to investigate the effect of removing the deterministic seasonal component present in the salinity, flow and river level time series data. In so doing, the resulting time series can be considered stationary. Histograms of salinity at Murray Bridge, flow at Overland Corner and river level at Lock 1 Lower, after seasonal standardisation, are shown in Figure 4.54, Figure 4.55 and Figure 4.56, respectively.

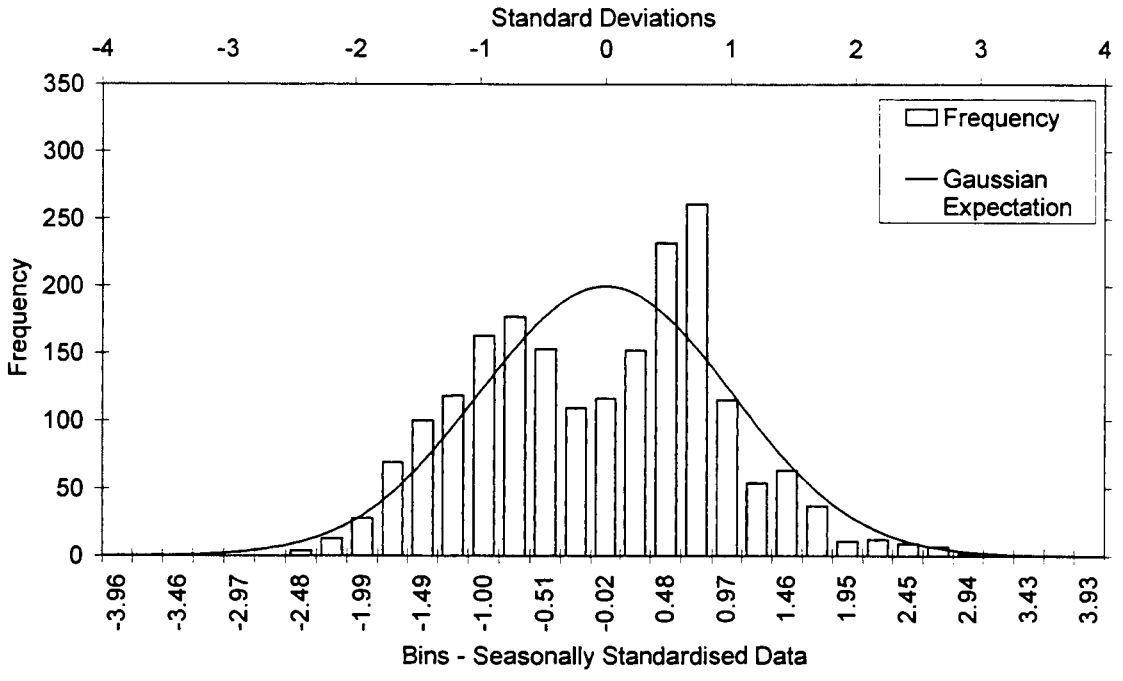

Figure 4.54 Histogram of Salinity at Murray Bridge (Seasonally Standardised Data)

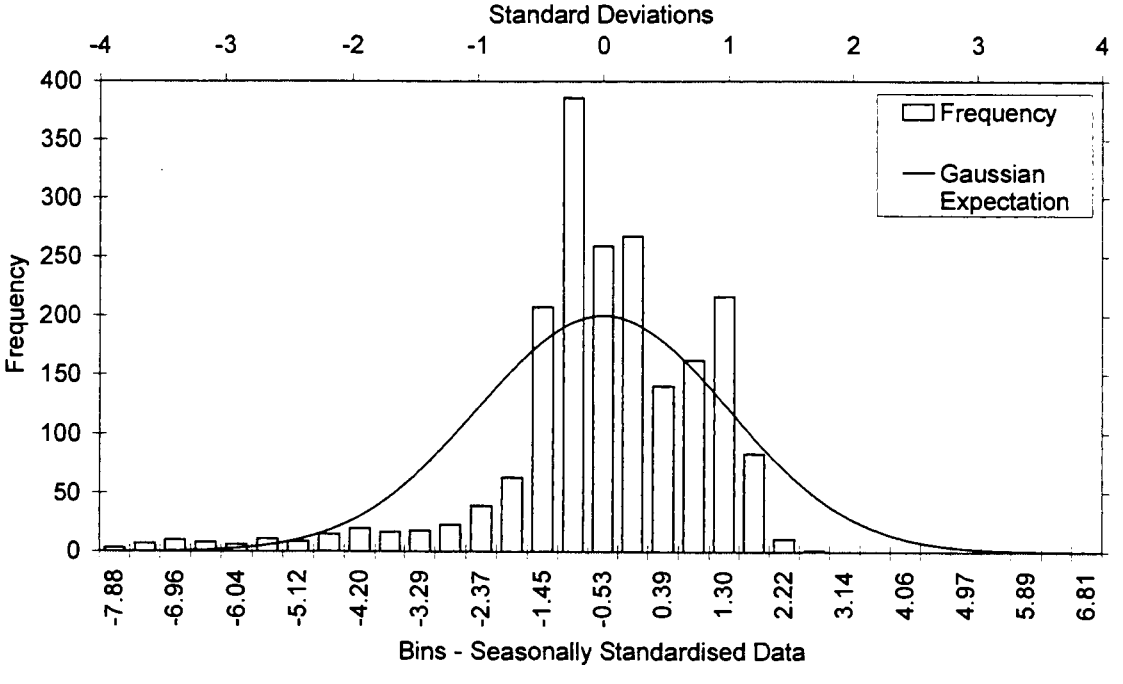

Figure 4.55 Histogram of Flow at Overland Corner (Seasonally Standardised Data)

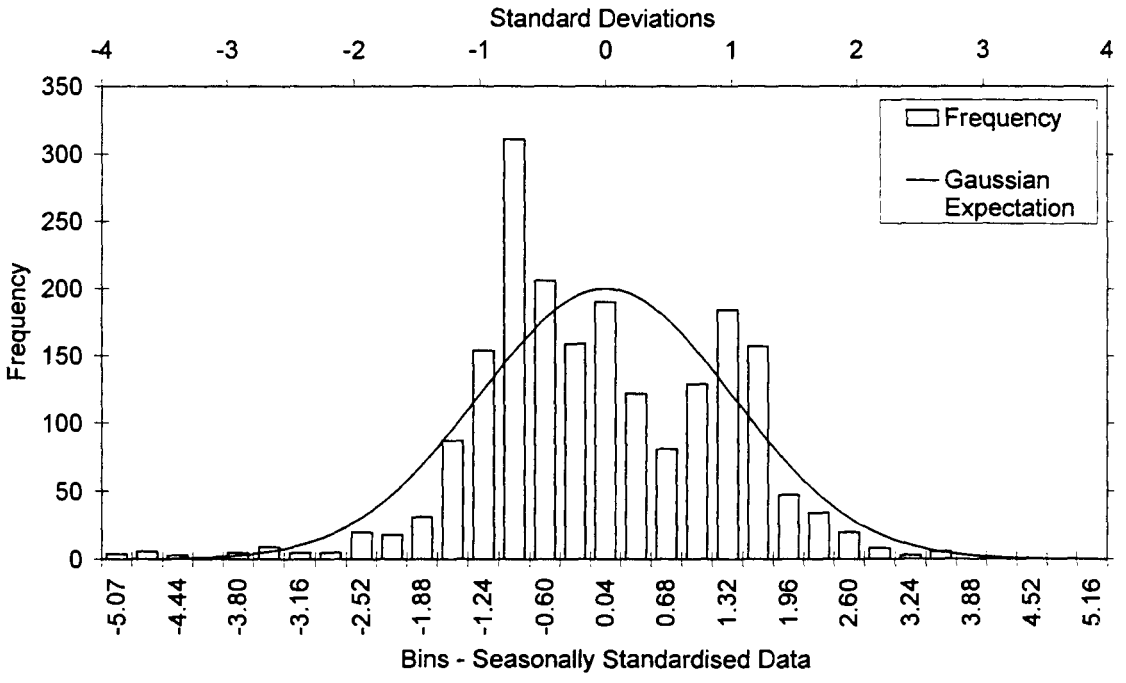

Figure 4.56 Histogram of River Level at Lock 1 Lower (Seasonally Standardised Data)

Transformation to normality

The final transformation investigated was the effect of transforming the data to normality. It has been suggested in the literature that an ANN's performance may be improved if the data are normally distributed (e.g. Fortin et al., 1997; Masters, 1993). Histograms of salinity at Murray Bridge, flow at Overland Corner and river level at Lock 1 Lower, after the transformation to normality, are shown in Figure 4.57, Figure 4.58 and Figure 4.59, respectively. From the plots, it is apparent that the transformation developed in this research was successful in producing transformed variables with a Gaussian distribution.

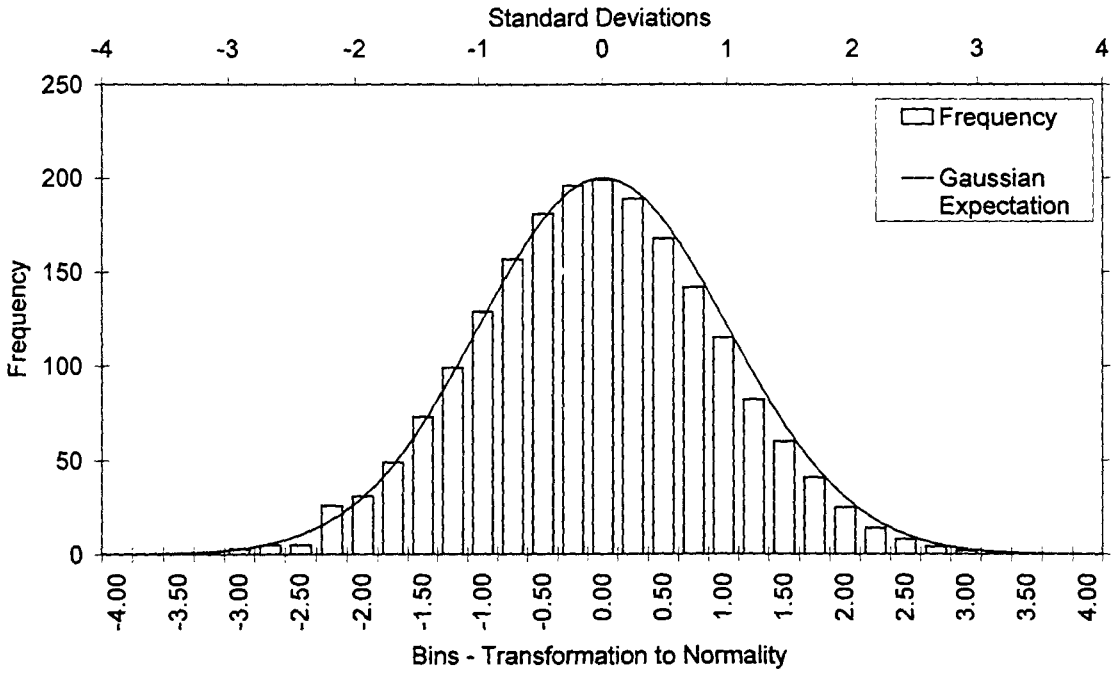

Figure 4.57 Histogram of Salinity at Murray Bridge (Transformation to Normality)

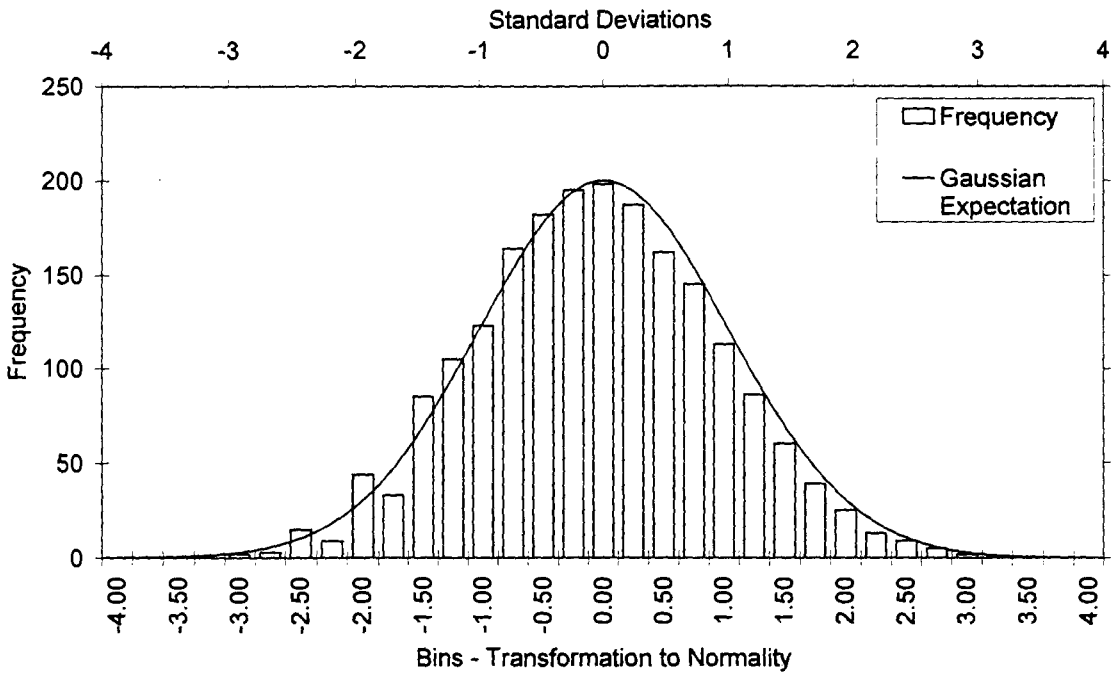

Figure 4.58 Histogram of Flow at Overland Corner (Transformation to Normality)

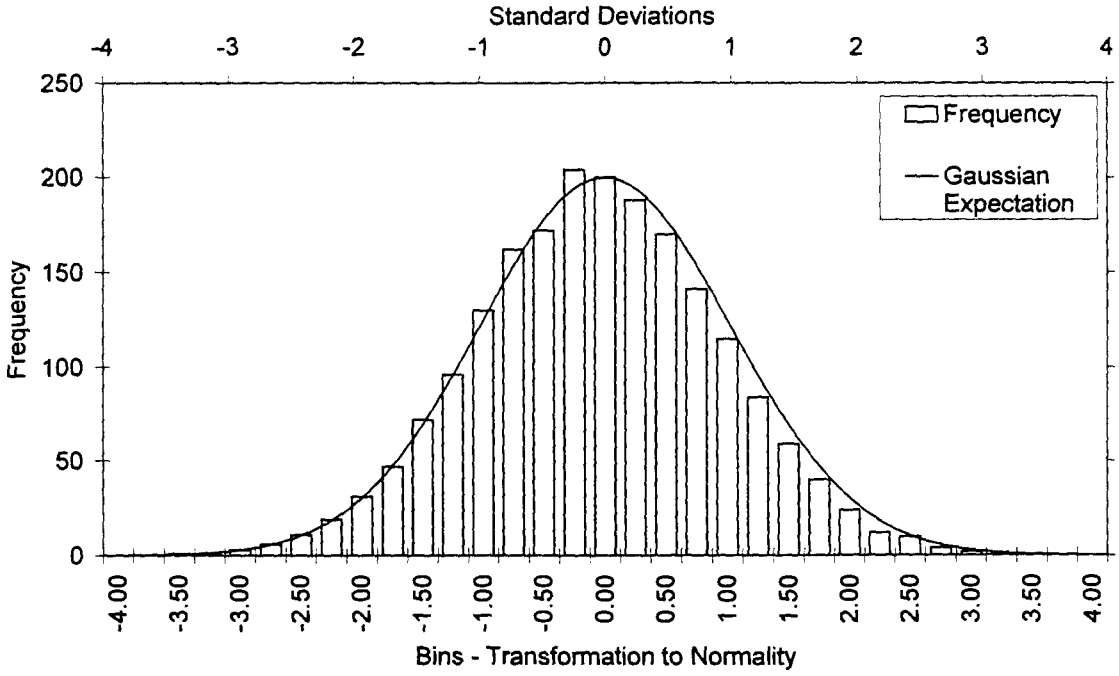

Figure 4.59 Histogram of River Level at Lock 1 Lower (Transformation to Normality)

4.7.2 Results and Discussion

The RMSEs for the MLP models using the linear, logarithmic, histogram equalization, kernel, seasonal and normality transformations are summarised in Table 4.21. It can be seen that the models developed using the linear, histogram equalization and kernel transformations performed significantly better than those developed using the logarithmic, seasonal and normality transformations for the training, testing and validation sets. These results are consistent with Faraway and Chatfield (1998), whose results indicated no improvement when a logarithmic transformation was used and that predictive ability deteriorated when the seasonality was removed from the data.

To investigate why the logarithmic, seasonal and normality transformations produced larger forecasting errors, sensitivity analyses were conducted. As part of the sensitivity analyses, each of the inputs were increased by 5% in turn, and the change in the output caused by the change in the input was calculated. The sensitivity of each input is given by:

$$\text{Sensitivity} = \frac{\%\text{ change in output}}{\%\text{ change in input}} \times 100 \qquad (4.1)$$

The results of the sensitivity analyses that were performed for the models developed using the logarithmic, seasonal and normality transformations are shown in Figure 4.60a, Figure 4.60b and Figure 4.60c, respectively. For comparison, a sensitivity analysis was performed for the model developed using linear transformation (Figure 4.60d).

Table 4.21 RMSEs of 6 Different Transformations for the 14-Day Salinity Forecasts at Murray Bridge

Data Set	Training Set	Testing Set	Validation Set
	RMSE (EC units)	RMSE (EC units)	RMSE (EC units)
Linear Transformation	32.5	31.2	36.5
Logarithmic Transformation	54.0	55.8	54.2
Histogram Equalization	24.8	32.6	28.3
Kernel Transformation	25.3	28.2	30.0
Seasonal Transformation	54.1	54.3	53.4
Transformation to Normality	54.6	56.4	56.6

From the sensitivity plots it is apparent that the most significant input for all 4 models is Waikerie salinity at lag 1. The second most significant variable for the logarithmic, seasonal and normality transformations is salinity at Mannum at lag 1. For the linear transformation, the second most significant variable is river level at Lock 1 Lower forecast 3 days ahead. The most notable difference in the sensitivity plots is that the model developed using linear transformation assigned a much greater significance to the flow and river level input variables. The inability of the models developed using the logarithmic, seasonal and normality transformations to make use of the information contained in the flow and river level data may have resulted in the higher forecasting errors. One reason this may have occurred for the model developed using a logarithmic transformation arises from the compressing nature of this transform.

Table 4.22 shows the training set maximum and minimum values and the ratio of max./min. for each of the variables used in this case study. It is evident that the ratio of max./min. is relatively low for the salinity variables, ranging from 4.0 - 5.8. However, the ratio of max./min. for the flow and river level data is 62.5 and 10.6, respectively. Consequently, when the logarithmic transformation is applied to these latter variables, they undergo a much greater compression and this could have distorted the information originally contained in the flow and level data.

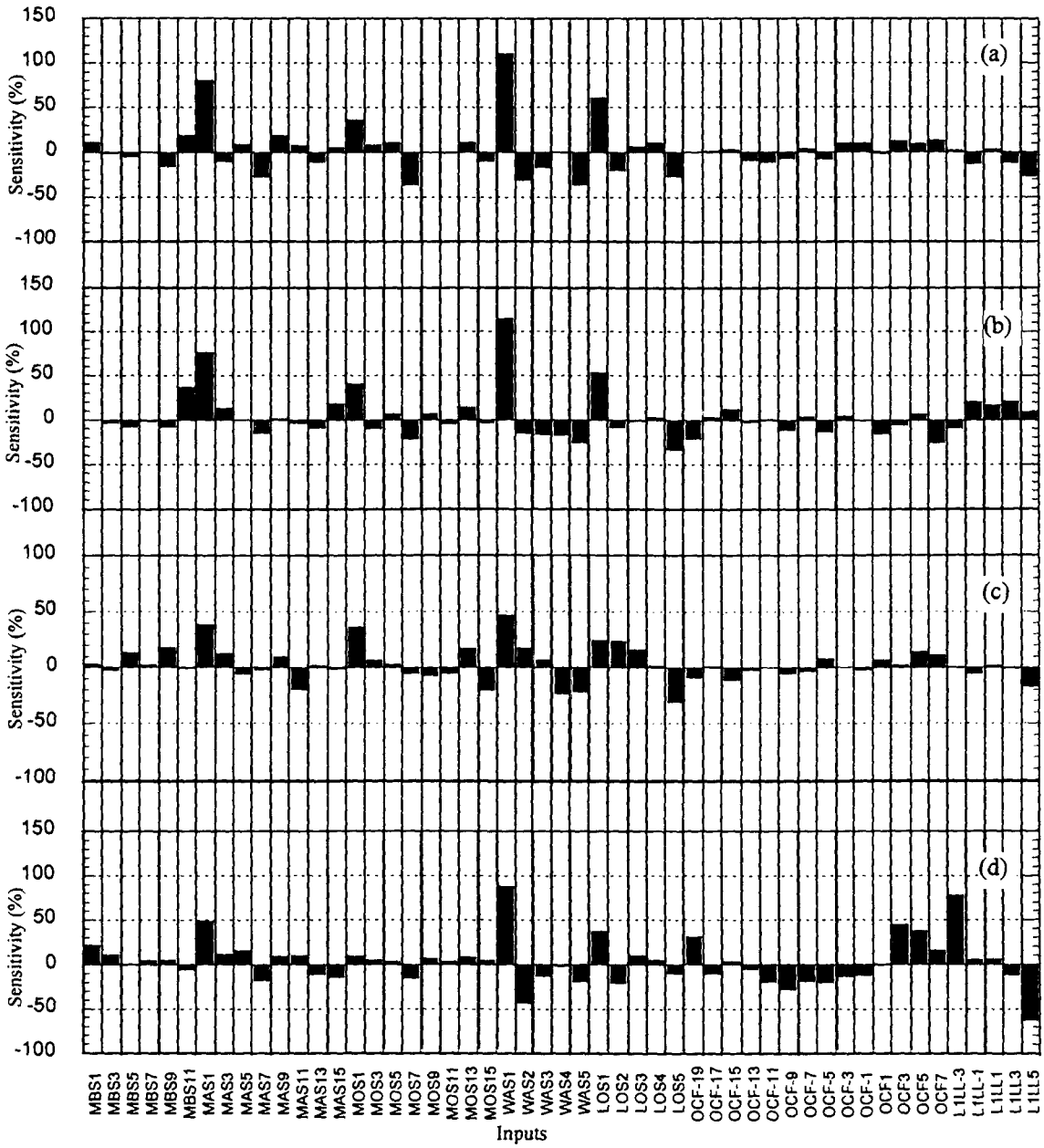

Figure 4.60 Sensitivity of Salinity Forecasts to the 51 Input Variables for the Model Developed using (a) Logarithmic Transformation; (b) Seasonal Transformation; (c) Transformation to Normality; (d) Linear Transformation

Table 4.22 Maximum, Minimum and Ratio of Max./Min. for Each Variable

Variable	Maximum	Minimum	Ratio (max./min.)
MBS	1116	261	4.3
MAS	1075	253	4.2
MOS	1061	183	5.8
WAS	1021	247	4.1
LOS	907	225	4.0
OCF	110618	1769	62.5
L1LL	5.3	0.5	10.6

Figure 4.61a shows the actual values of the flow at Overland Corner used in the training, testing and validation sets and the seasonal cycle fitted to the mean of these data. It can be seen that the effect of flow is largely accounted for by the seasonal cycle in its mean. So the flow itself may not be significant for a seasonal model.

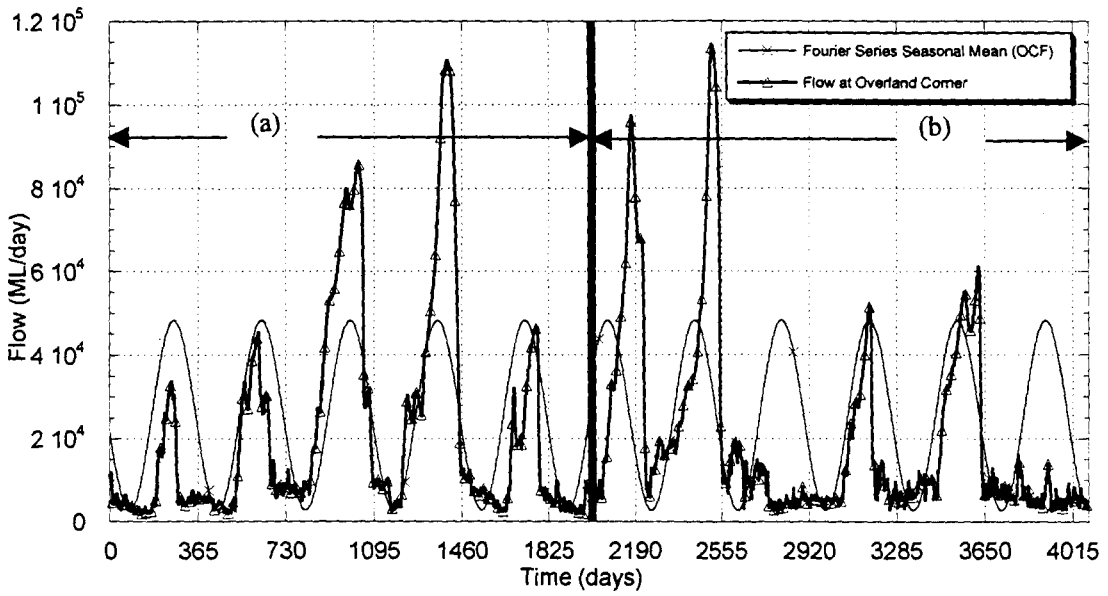

Figure 4.61 Flow at Overland Corner and Fourier Series Seasonal Mean (OCF) for (a) Training, Testing and Validation Data (December 1986 to June 1992) and (b) Second Validation Data (July 1992 to March 1998)

The best result on the independent validation set came from the model developed using the histogram equalization transformation. This result is in accordance with Shi (2000) who found that transforming the input data to uniformity provided a smooth and continuous mapping of the input variables to the output variable and performed better than the model developed using linear transformation. Kernel transformation, which

also transforms the input data to uniformity, produced the second best validation set result.

Diagnostic checks were performed on each of the 6 ANN models by examining the error residuals ($\hat{e}_t$). In general, these errors are caused by the random shock (noise) component in the data (e_t), and the inability of the model to perfectly predict the deterministic components of the data. Consequently, $\hat{e}_t$ is only an estimate of the true random error e_t. If the model fits the data well, then these residuals should satisfy the following assumptions:

- The expected value of $\hat{e}_t$ is zero.
- The variance of $\hat{e}_t$ is a constant σ^2.
- The errors are statistically independent of each other.
- The errors are normally distributed.

For each of the 6 transformations, the assumptions were tested by plotting histograms of the residuals (Figure 4.62 to Figure 4.67), by plotting the standardised residual versus the predicted response (Figure 4.68 to Figure 4.73) and by plotting the autocorrelation function of the residuals (Figure 4.74 to Figure 4.79). The model residuals were obtained by concatenating the training, testing and validation data sets and then chronologically ordering these data to obtain the original multivariate time series data. Each of the 6 models were then used to obtain forecasts for these data and the residuals were calculated.

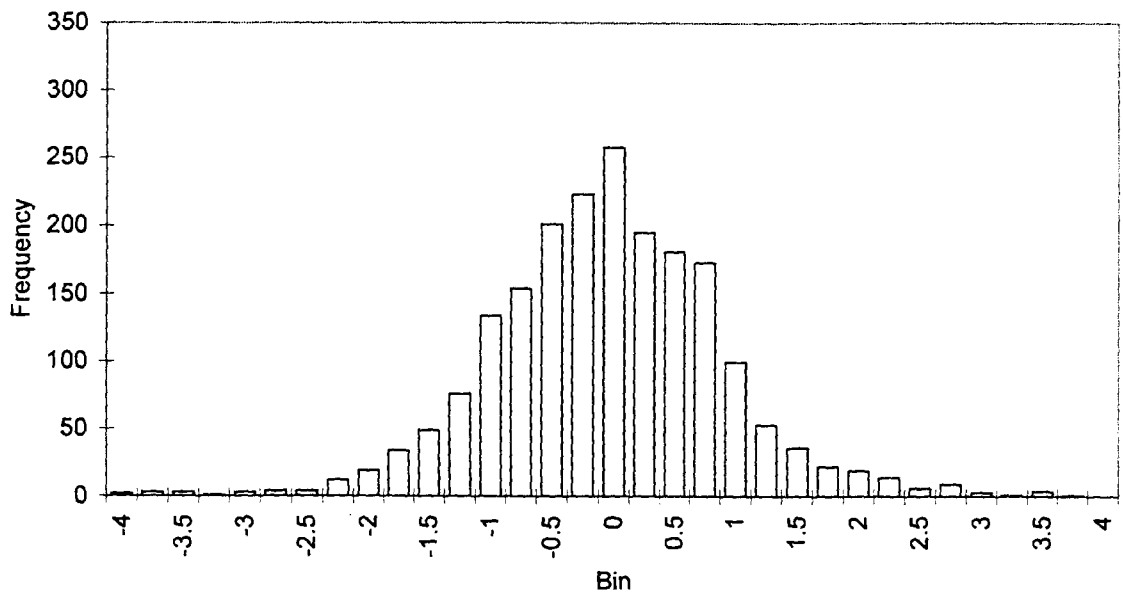

Figure 4.62 Histogram of ANN Residuals (Linear Transformation)

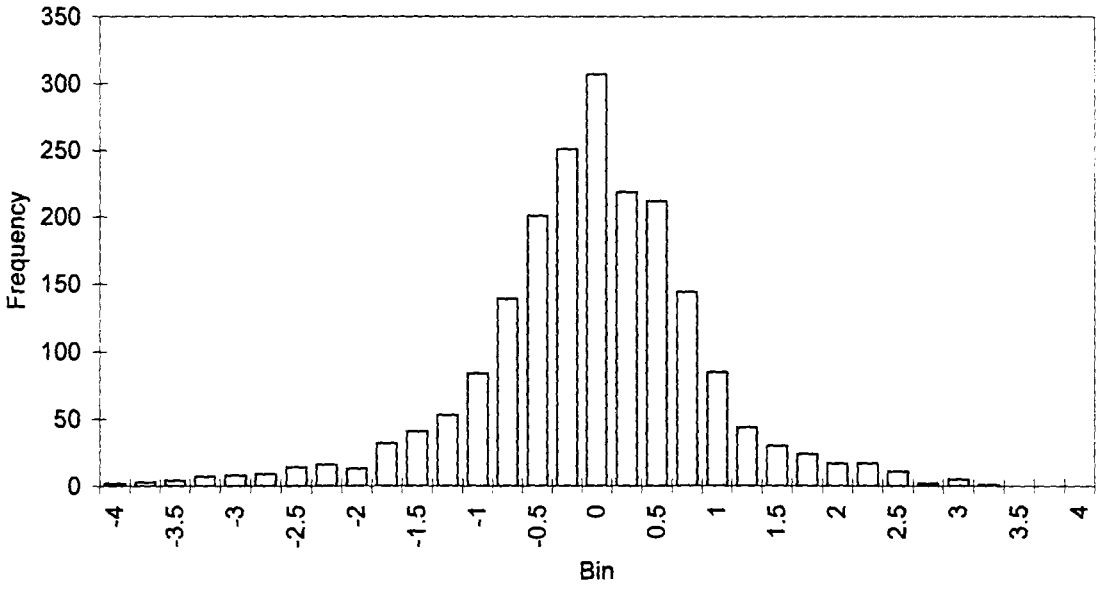

Figure 4.63 Histogram of ANN Residuals (Logarithmic Transformation)

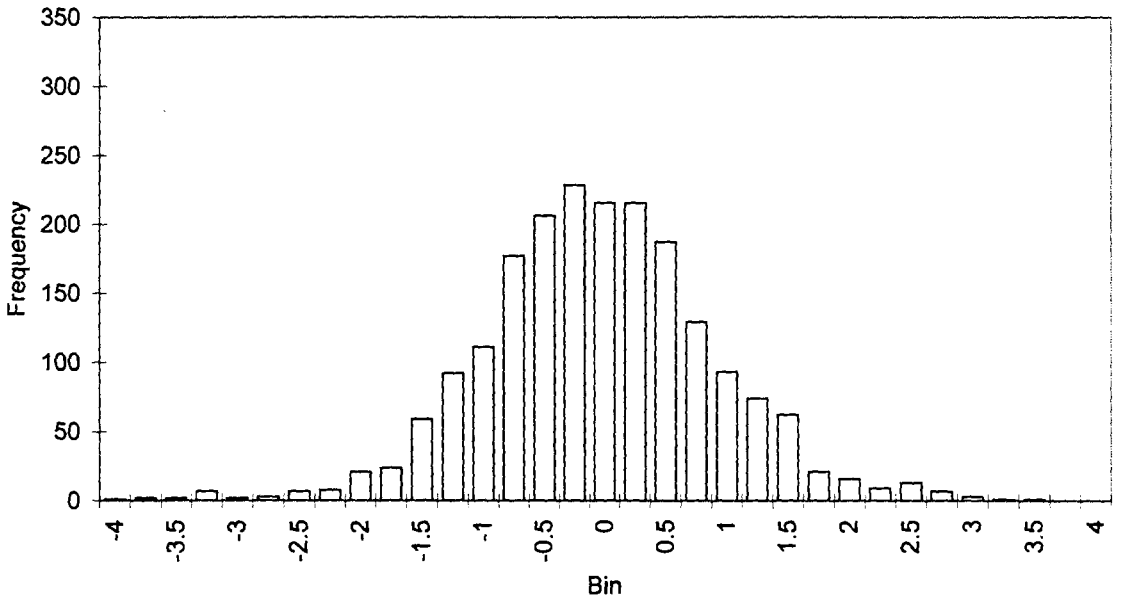

Figure 4.64 Histogram of ANN Residuals (Histogram Equalization Transformation)

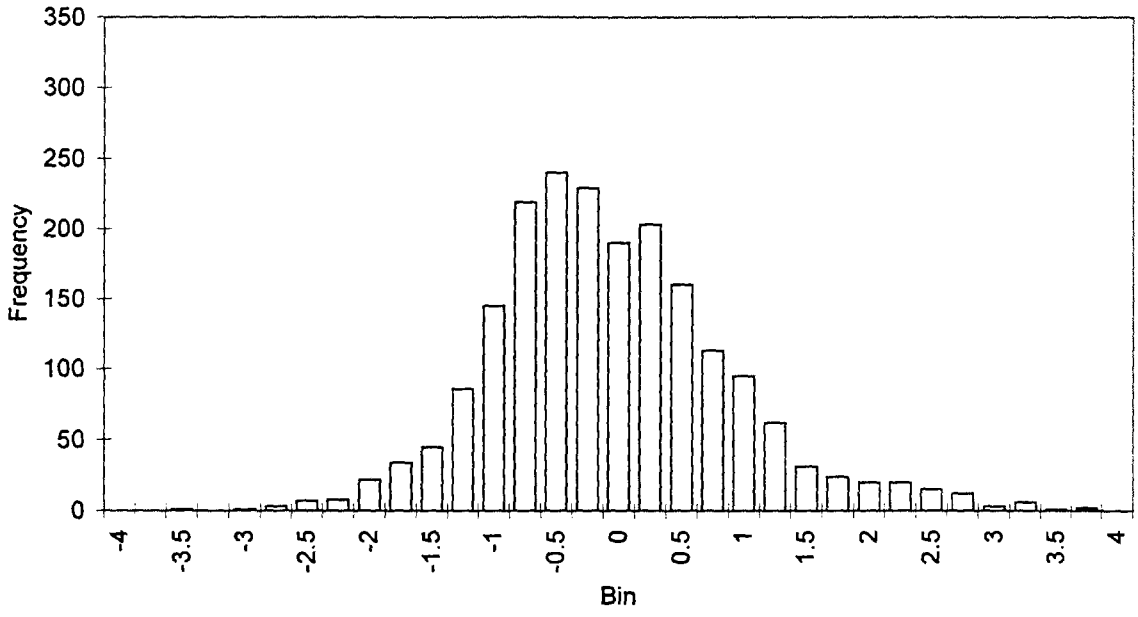

Figure 4.65 Histogram of ANN Residuals (Kernel Transformation)

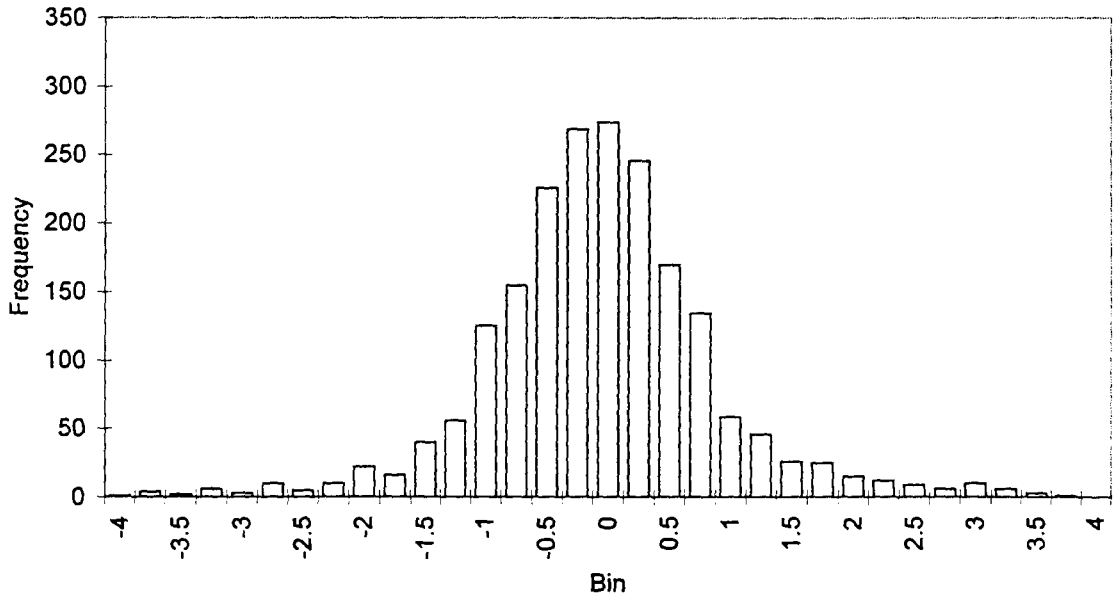

Figure 4.66 Histogram of ANN Residuals (Seasonal Transformation)

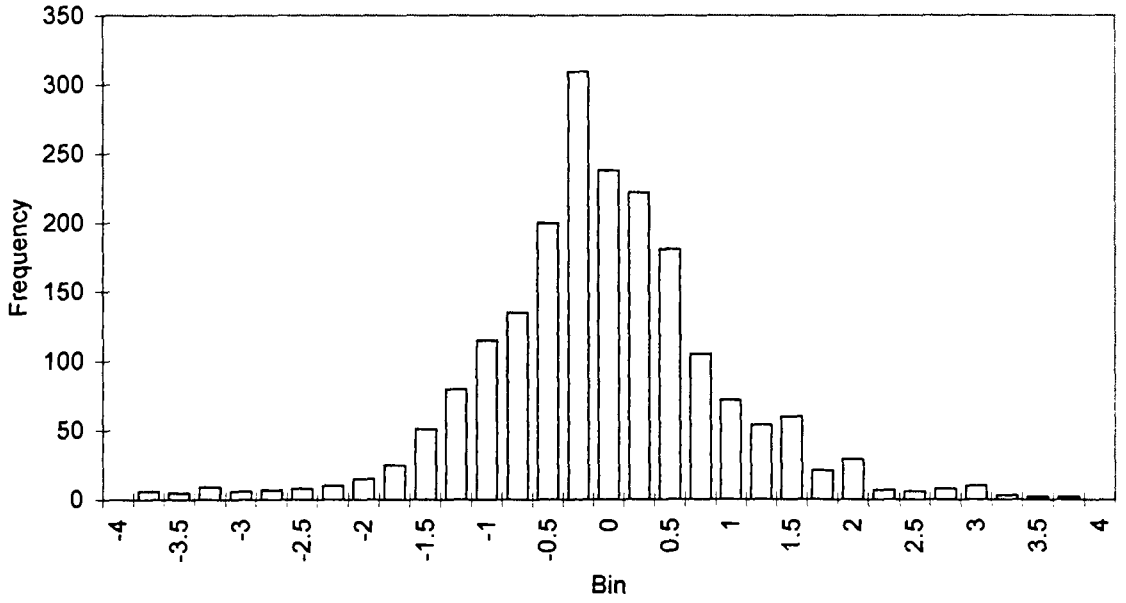

Figure 4.67 Histogram of ANN Residuals (Transformation to Normality)

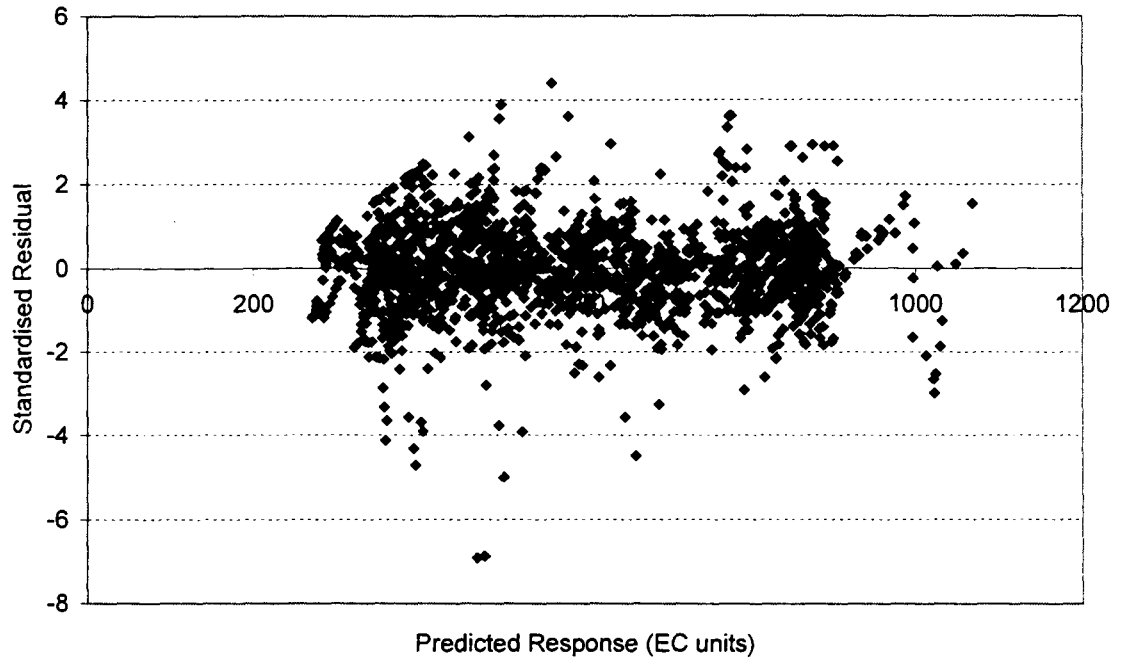

Figure 4.68 Standardised Residual Versus Predicted Response - 2005 Data Points (Linear Transformation)

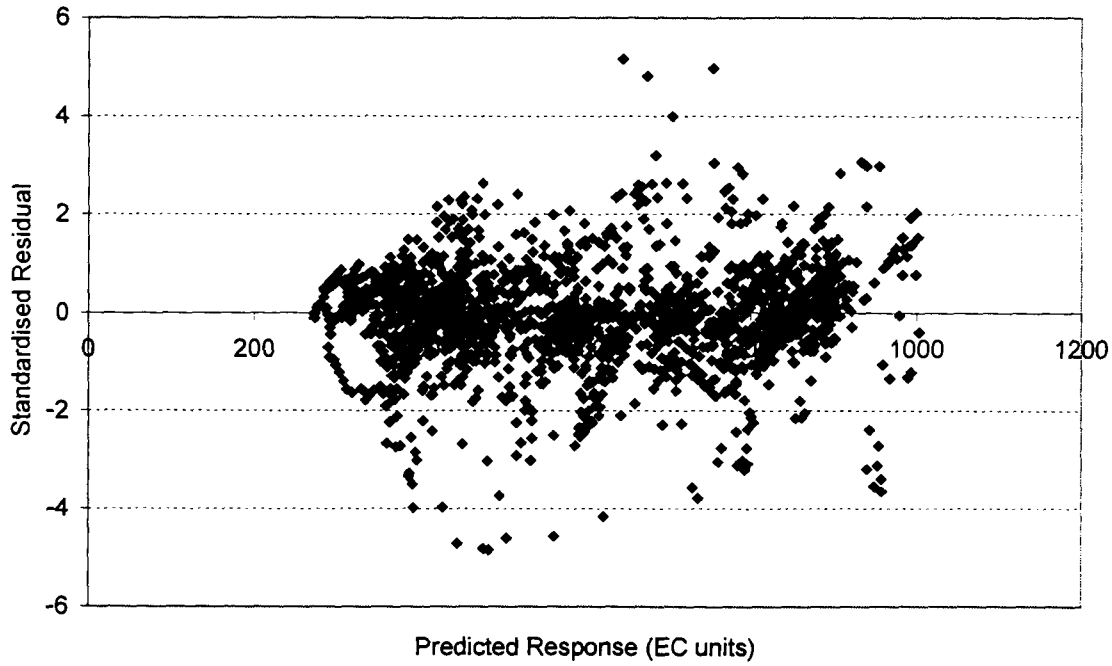

Figure 4.69 Standardised Residual Versus Predicted Response - 2005 Data Points (Logarithmic Transformation)

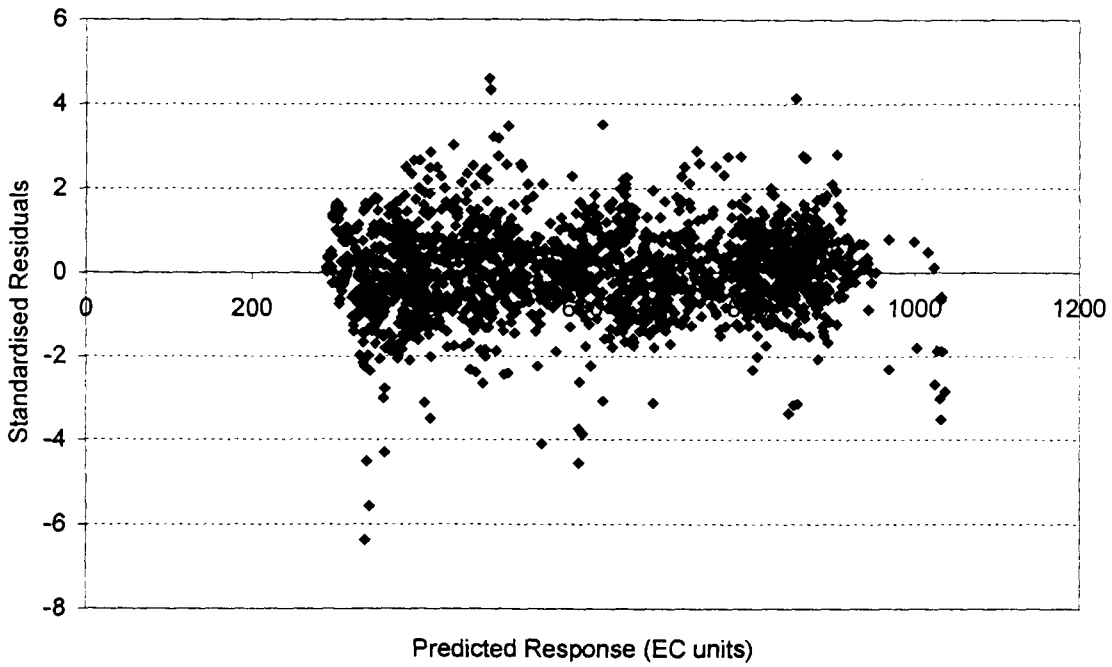

Figure 4.70 Standardised Residual Versus Predicted Response - 2005 Data Points (Histogram Equalization Transformation)

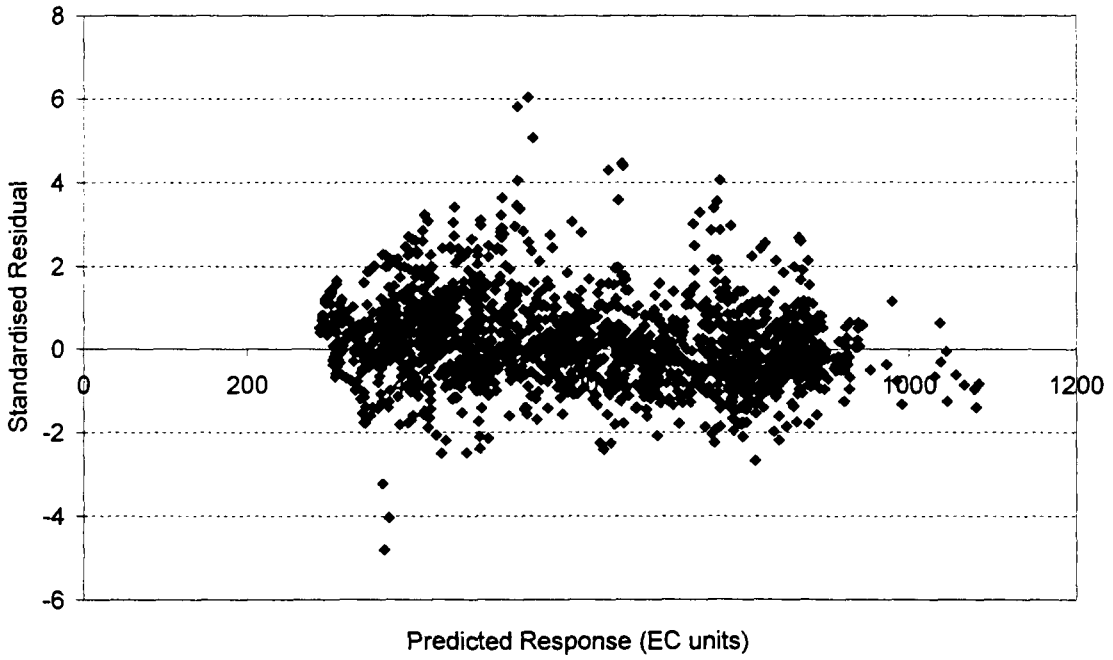

Figure 4.71 Standardised Residual Versus Predicted Response - 2005 Data Points (Kernel Transformation)

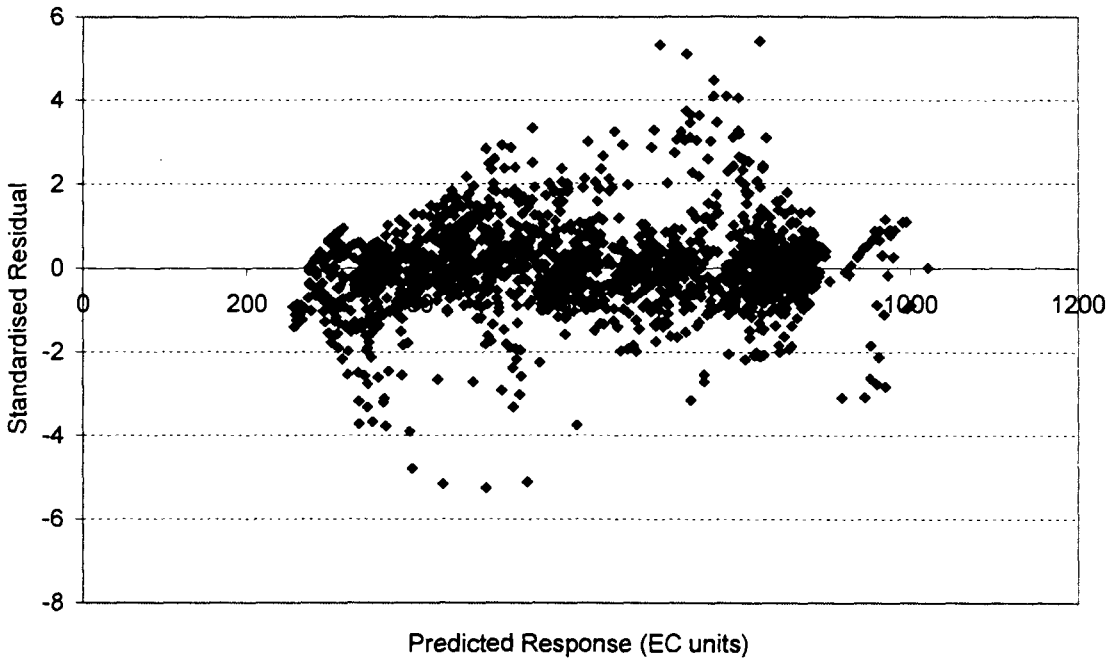

Figure 4.72 Standardised Residual Versus Predicted Response - 2005 Data Points (Seasonal Transformation)

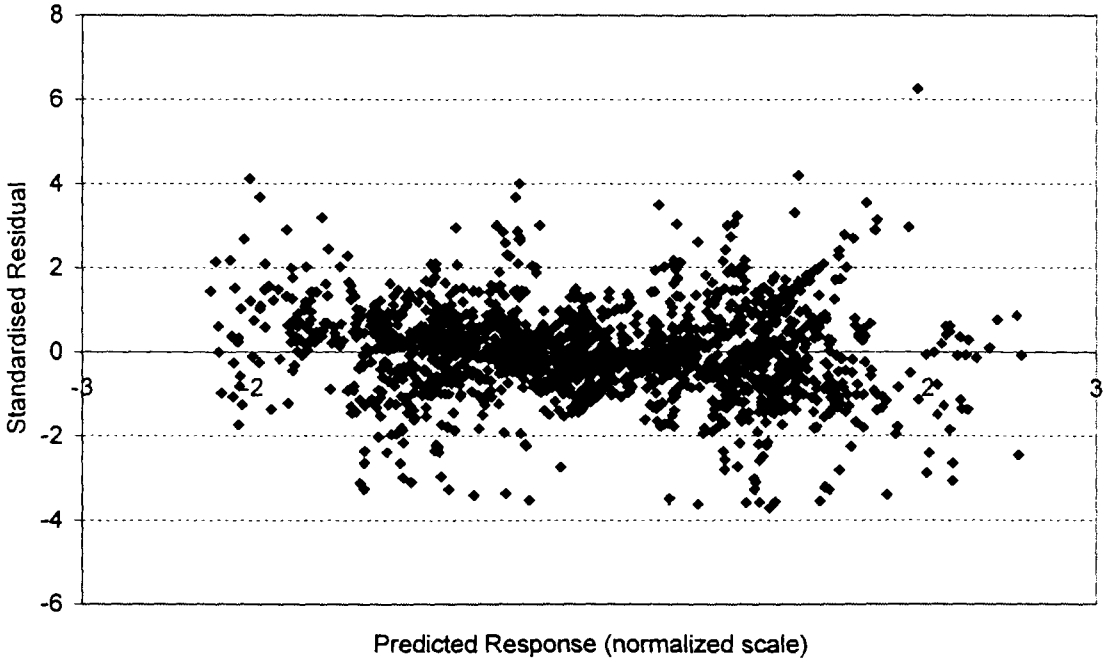

Figure 4.73 Standardised Residual Versus Predicted Response - 2005 Data Points (Transformation to Normality)

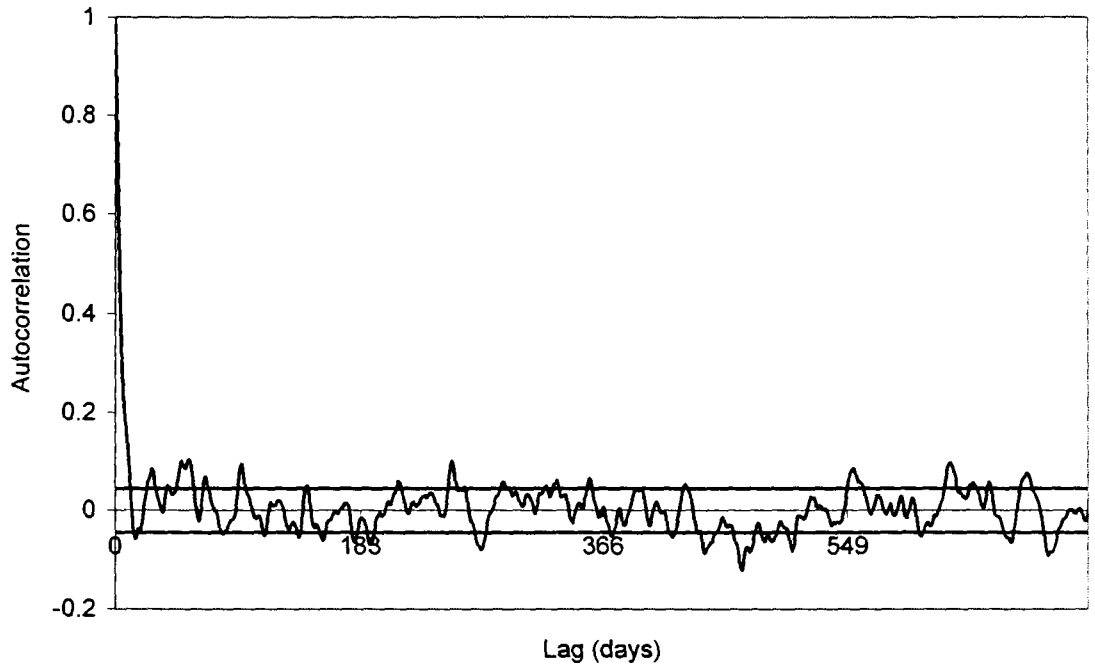

Figure 4.74 Autocorrelation Plots of ANN Residuals (Linear Transformation)

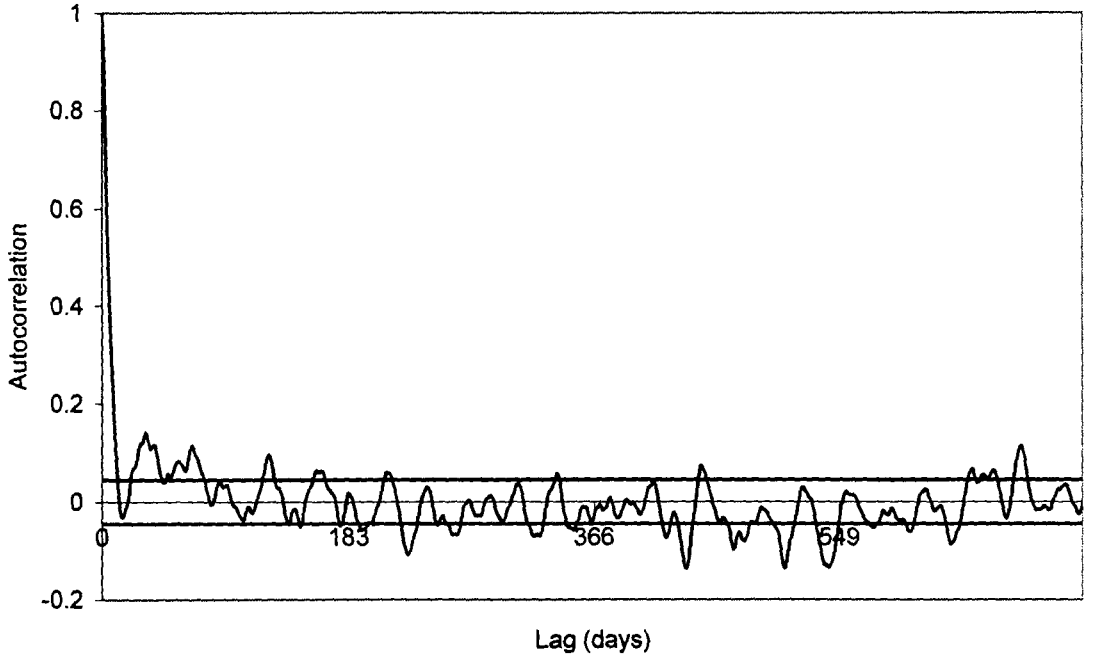

Figure 4.75 Autocorrelation Plots of ANN Residuals (Logarithmic Transformation)

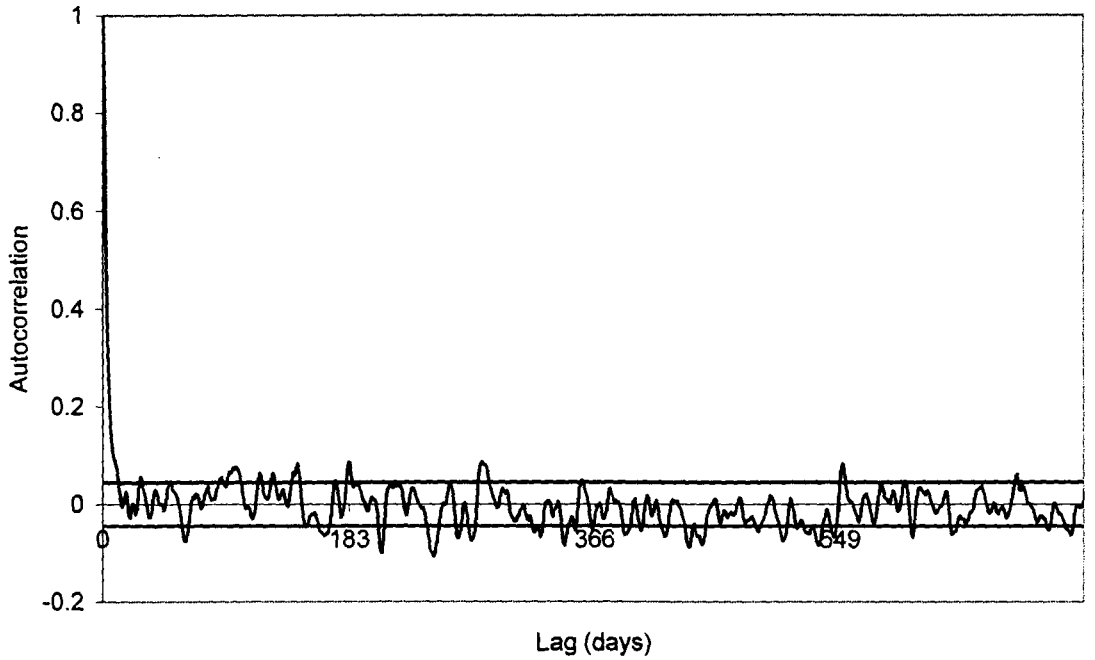

Figure 4.76 Autocorrelation Plots of ANN Residuals (Histogram Equalization Transformation)

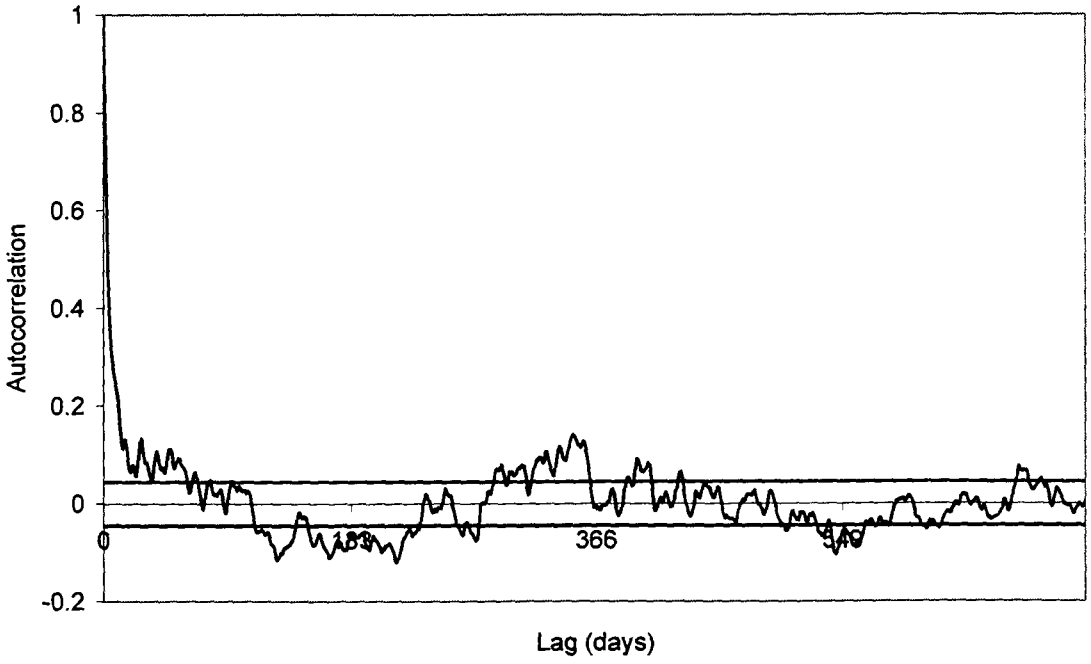

Figure 4.77 Autocorrelation Plots of ANN Residuals (Kernel Transformation)

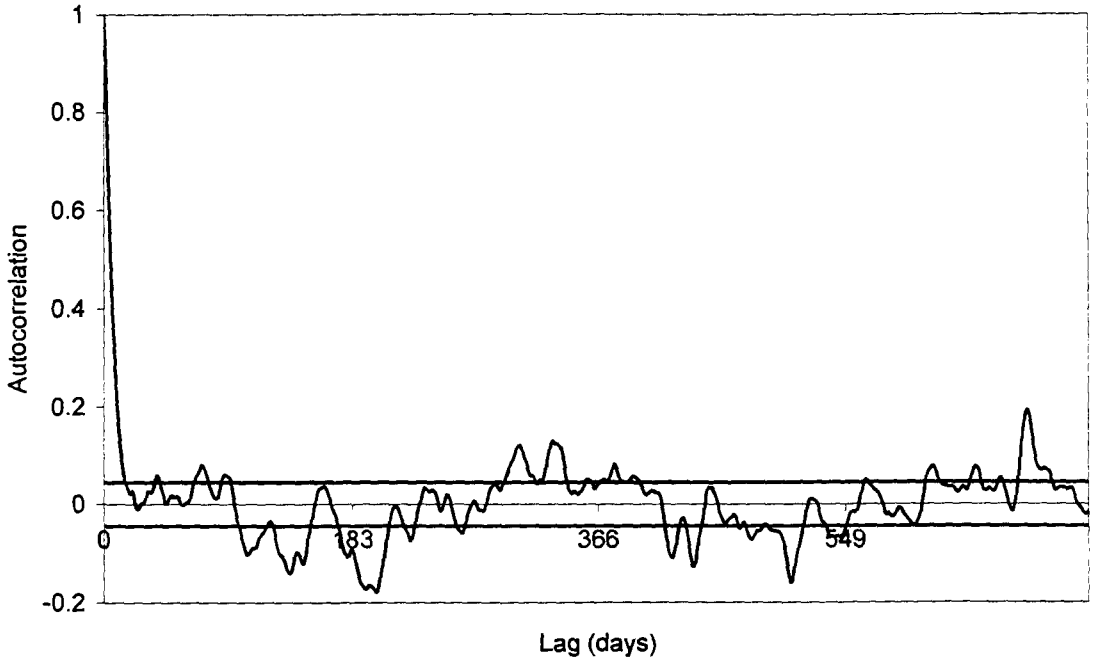

Figure 4.78 Autocorrelation Plots of ANN Residuals (Seasonal Transformation)

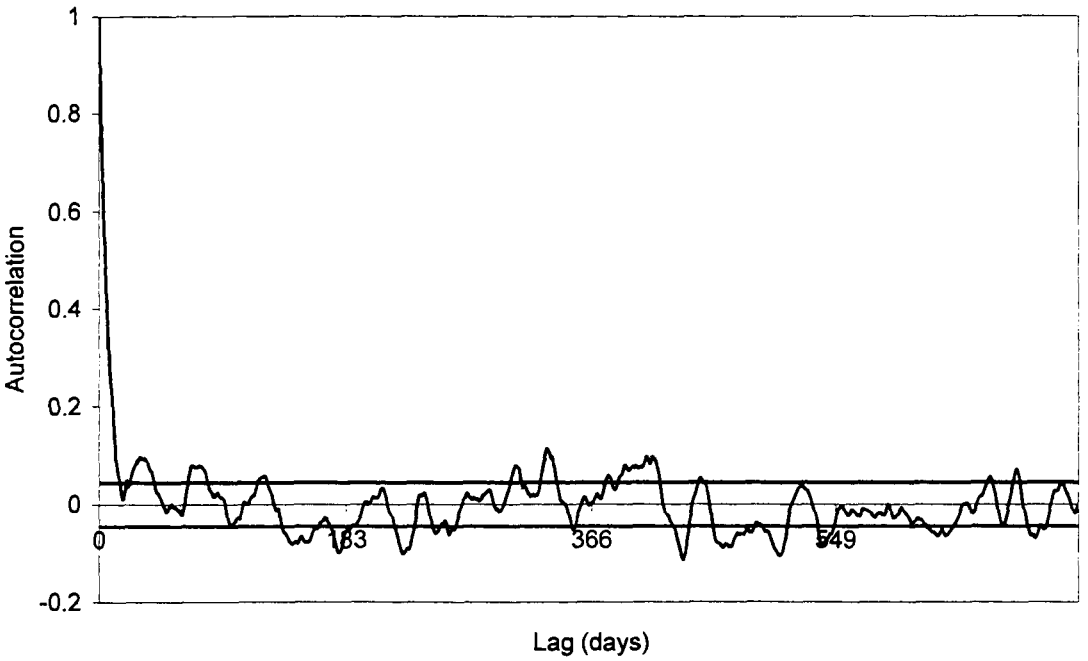

Figure 4.79 Autocorrelation Plots of ANN Residuals (Transformation to Normality)

From the histogram plots (Figure 4.62 to Figure 4.67) it can be seen that the residuals for all models were approximately normally distributed, satisfying assumption 4, with a mean of approximately zero, satisfying assumption 1. Figure 4.68 to Figure 4.73 show that for each of the 5 models, the 2005 data points appear to be scattered randomly and largely contained within a parallel band with approximately 95% falling between -2 and 2, thereby satisfying assumption 2. In Figure 4.74 to Figure 4.79, the 95% confidence limits for the autocorrelations are shown as estimated from statistically independent residuals. If most of the autocorrelation plot falls inside of the 95% limits, then the assumption that the residuals are statistically independent is not inconsistent with the data (assumption 2). This appears to approximately hold true for the linear and histogram equalization transformations (Figure 4.74, Figure 4.76). However, the residuals from the models developed using the log, kernel, seasonal and normal transformations appear to violate the statistical independence assumption.

It is interesting to note that the residuals from the model developed using the linear transformation satisfy assumptions 1-4 above. Therefore, the objective function used to calibrate the ANN model (mean square error) is justified as it maximises the likelihood of the model under the hypothesis of normal random shocks. If the random shock component were known to be skewed, then the objective function would need to be adjusted accordingly in order to obtain maximum likelihood estimates of the model parameters (Fortin et al., 1997). From these results, it is apparent that it is not

necessary to transform the data to normality for the ANN's error residuals to be approximately normally distributed.

4.7.2.1 Real-time forecasting

The six ANN models developed using each transformation technique could be deployed in a real-world forecasting scenario. In such a case, it is likely that in time, the models would encounter data outside of the calibration range. The model's robustness would directly depend on how accurately it could produce forecasts for such uncharacteristic data. To investigate the robustness of the ANN models developed using each transformation, a second, independent validation set was used, consisting of daily data from the period 15-07-1992 to 13-03-1998. This second validation set is the same set used in Section 4.6 and has been shown to contain regions of data outside of the calibration range. The models developed using the six transformation techniques were used to obtain 14-day forecasts for this second validation set.

In Table 4.23, the RMSEs for the second validation set are shown, and as expected, these are significantly higher for all models than the training set RMSEs obtained in Table 4.21. This is due to the presence of uncharacteristic data i.e. new patterns that the models have not been trained on. The model developed using linear transformation appears to be the most robust, as indicated by the lowest forecasting error on the second validation set. It is surprising that the model developed using histogram equalization performed significantly worse on this set. This indicates that although this model was able to learn the relationships in the training, testing and validation data, it was not robust when dealing with uncharacteristic data in the second validation set.

The model that gave the highest RMSE on the second validation set was the ANN developed using the seasonally transformed data. The second validation set forecasts for this model and the Fourier series for the seasonally varying mean (MBS) are shown in Figure 4.80. It is important to note that the Fourier function for the seasonally varying mean was developed using the training data. However, it is apparent that the seasonality in the second validation set does not follow the same fluctuations and this is highlighted by the seasonal mean moving out of phase with the actual salinity time series. This is especially noticeable in Regions 1 and 2 indicated in Figure 4.80. In both of these regions, the actual time series did not follow the typical seasonal cycle and consequently, the resulting forecasts underestimated the true values. Figure 4.61b shows a plot of flow at Overland Corner and the Fourier series for the seasonally varying mean (OCF) for the second validation set. It is evident that the high salinity events in Regions 1 and 2 of Figure 4.80 correspond to uncharacteristic low flow events

in Figure 4.61b. This highlights the need for periodic refitting of the Fourier series and retraining of the ANN model.

Table 4.23 Second Validation Set RMSEs of 6 Different Transformations for the 14-Day Salinity Forecasts at Murray Bridge

Data Set	Second Validation Set	Second Validation Set with Uncharacteristic Data Removed
	RMSE (EC units)	RMSE (EC units)
Linear Transformation	54.7	32.7
Logarithmic Transformation	74.6	52.3
Histogram Equalization	61.1	45.9
Kernel Transformation	85.5	44.7
Seasonal Transformation	115.7	59.2
Transformation to Normality	89.5	60.4

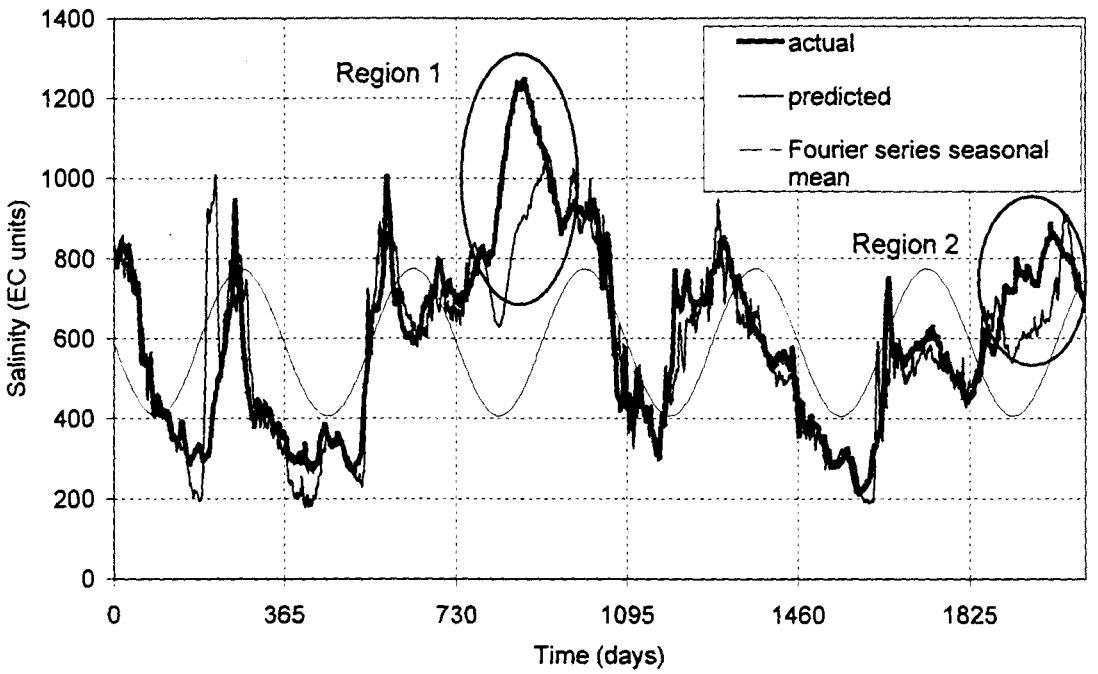

Figure 4.80 Second Validation Set 14-Day Forecasts for the Model Developed Using Seasonally Transformed Data (July 1992 to March 1998). The Fourier Series Seasonal Mean (MBS) is also Shown For the Salinity at Murray Bridge

To diagnose regions of poor performance resulting from uncharacteristic data, a SOM diagnosis procedure was proposed in Section 4.6.2. In this procedure, the calibration data are clustered with the validation set using a SOM. By inspecting the resulting

clusters of data, it is possible to discern uncharacteristic data i.e. the validation data that form clusters on the Kohonen grid that contain no patterns from the training set. The data in such clusters can be considered to lie outside of the training domain and since ANNs are unable to extrapolate beyond the training range (Flood and Kartam, 1994), poor generalisation ability can be expected on these data. In the present study, the SOM diagnosis procedure was conducted using the second validation set and the uncharacteristic data points were removed from the set. The models developed using the 6 transformations were then used to obtain forecasts for this truncated data set. The results of this experiment are shown in Table 4.23 and as expected, the RMSEs were reduced quite considerably for all models. The models developed using the linear, logarithmic, seasonal and normality transformations all produced forecast errors that were very similar to their errors on the training set. The models developed using the histogram equalization and kernel transformation data produced a forecast error that was still higher than their training error. However, in general, these results verify the effectiveness of the SOM diagnosis procedure in determining the range of applicability of the ANN models.

4.8 Determination of Model Inputs

The three input determination procedures proposed in Chapter 3 (i.e. Stepwise PMI algorithm, PCA-GAGRNN and SOM-GAGRNN) were used to determine the optimal set of inputs for forecasting salinity at Murray Bridge, 14 days in advance. In order to assess the suitability of these three methods for determining inputs to multivariate ANN models, all three sets of inputs were used to develop ANN models for forecasting salinity in the River Murray, at Murray Bridge. The ANN models developed in this study were also compared with ANN models developed using the inputs obtained in two previous studies conducted for the same case study by Maier and Dandy (1996b) and Maier and Dandy (1997a).

Sixteen input variables were considered as candidate model inputs for this case study. These include: salinities at Murray Bridge (MBS), Mannum, (MAS), Morgan (MOS), Waikerie (WAS) and Loxton (LOS), flows at Lock 1 Lower (L1LF), Overland Corner (OCF) and downstream of Lock 7 (L7F), as well as river levels (which are given above the Australian Height Datum (AHD)) at Murray Bridge (MBL), Mannum, (MAL), Lock 1 Upper (L1UL), Lock 1 Lower (L1LL), Morgan (MOL), Waikerie (WAL), Overland Corner (OCL) and Loxton (LOL).

4.8.1 Model Development

The GA data division method (Section 3.3.2.1) was used to divide the data into statistically representative subsets. The data from 01-12-1986 to 30-06-1992 were used to create the training, testing and validation data sets. The first 30 lags of each variable were considered as potential inputs and this resulted in a total of 1983 data records being available, from which 1586 records (80%) were used for calibration and 397 records (20%) were used for validation. The 1586 records in the calibration set were further divided into 1269 training records (80%) and 317 testing records (20%).

In Section 4.7, a linear transformation of the data was found to be the most robust transformation for the salinity case study. Consequently, the data were linearly transformed by using the original data range to rescale the series to a range that was commensurate with the output transfer function. The ranges used in conjunction with the hyperbolic tangent output transfer function were -1.0 to +1.0 for the network inputs and -0.8 to +0.8 for the network outputs.

The three input determination procedures outlined in Section 3.5 were used to find an appropriate set of model inputs. Since the number of inputs obtained using each

method differed from the default number of 51 (Section 4.4.2), it was no longer appropriate to use the default network architecture. Therefore, a trial-and-error method was used to determine a suitable architecture for the models developed with each set of inputs. All other aspects of the modelling process were held constant in accordance with the default selections outlined in Section 4.4.

4.8.1.1 Determination of Model Inputs

Only the calibration data from the training and testing sets were used in the input determination phase. The three new methods were then used to select suitable model inputs. These were the stepwise PMI algorithm (Method 1), the PCA-GAGRNN (Method 2) and the SOM-GAGRNN (Method 3). The inputs obtained from each of these methods are summarised below.

Determination of model inputs using the stepwise PMI algorithm (Method 1)

The stepwise PMI algorithm for selecting appropriate inputs is described in detail in Section 3.5.2. The algorithm uses the partial mutual information criterion to determine the strength of the relationship between the output time series and the input time series at various lags. The inputs are selected in a stepwise procedure, and an input is selected if the partial (or additional) dependence it can add to the existing prediction model is significant. The stepwise PMI algorithm is a *model-free* approach to input determination and requires very little *a priori* knowledge of the system under investigation.

The stepwise PMI algorithm was implemented using a two step procedure. In the first step, if the number of candidate variables is d (i.e. x_1, x_2, x_3, ..., x_d) and the output variable is y, then the stepwise PMI algorithm is run d times. For the first variable x_1, its own past values ($x_{1,t-1}$, $x_{1,t-2}$, ..., $x_{1,t-kmax}$) are the potential inputs, where k_{max} refers to the maximum lag that has been included as a potential input. If *a priori* knowledge about the relationship between the input and output time series is available, then k_{max} can be chosen such that the lags of the input variable that exceed k_{max} are not likely to have any significant effect on the output time series. If no *a priori* knowledge is available, k_{max} has to be chosen using a trial-and-error approach to ensure that the lagging window is sufficiently large to capture all significant lags. For example, if k_{max} is chosen by the stepwise PMI algorithm as a significant lag, then this may indicate that other lags greater than k_{max} are also significant and the lagging window may need to be extended. For the first variable x_1, the PMI is calculated between the inputs ($x_{1,t-1}$, $x_{1,t-2}$, ..., $x_{1,t-kmax}$) and y. Likewise, for the second variable x_2, its own past values are the

potential inputs and the PMI is calculated between these inputs and y and so on for each of the d variables. In this way, the first step determines the significant lags for each variable. Because there are only two variables used each time the algorithm is run (e.g. x_1 and y, x_2 and y, and so on), this step is hereby referred to as the bivariate stage.

In the second step, the significant lags selected in the previous step from each of the d variables are combined to form a subset of N candidate inputs, where $N<<d\times k_{max}$. The stepwise PMI algorithm is then run using this subset in order to determine the final set of significant model predictors. Since many of the d variables may be included in the subset of N candidate inputs, this step is referred to as the multivariate stage.

As a starting point for the salinity data, the first 30 lags of each of the sixteen candidate variables were considered as potential model inputs (i.e. k_{max} = 30), giving a total of 480 possible inputs. Therefore, in the bivariate stage, the stepwise PMI algorithm was applied to the first 30 lags of each of the sixteen variables in turn, and these results are summarised in Table 4.24. The bold numbers in the table represent the higher of the PMI and the 95th percentile randomised sample. The last row in each model's results represents the cutoff point i.e. the input that was not included because the 95th percentile randomised sample was greater than the corresponding PMI score.

Table 4.24 Stepwise PMI Input Selection Algorithm Results on Bivariate Salinity Data Sets With a Lagging Window of 30 Days

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
MBS	1	**0.768**	0.027
	23	**0.064**	0.027
	30	**0.032**	0.025
	8	**0.025**	0.024
	14	0.015	**0.024**
MAS	1	**0.843**	0.029
	22	**0.056**	0.026
	30	**0.034**	0.025
	7	**0.031**	0.025
	11	0.016	**0.024**
MOS	1	**0.769**	0.03
	30	**0.131**	0.021
	16	**0.031**	0.017
	8	0.014	**0.016**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
WAS	1	**0.811**	0.03
	30	**0.132**	0.018
	16	**0.02**	0.013
	7	**0.013**	0.012
	5	0.009	**0.013**
LOS	1	**0.787**	0.034
	30	**0.144**	0.027
	11	**0.035**	0.019
	19	0.016	**0.017**
L1LF	6	**0.452**	0.03
	30	**0.081**	0.025
	1	0.018	**0.021**
OCF	7	**0.465**	0.029
	30	**0.081**	0.025
	1	**0.025**	0.022
	20	0.013	**0.02**
L7F	8	**0.488**	0.027
	30	**0.093**	0.023
	1	**0.023**	0.02
	21	0.013	**0.019**
MBL	1	**0.309**	0.041
	22	**0.101**	0.042
	15	**0.061**	0.039
	30	**0.052**	0.039
	10	0.034	**0.038**
MAL	1	**0.342**	0.038
	30	**0.088**	0.035
	15	**0.041**	0.035
	5	0.023	**0.035**
L1UL	1	**0.198**	0.037
	30	**0.083**	0.035
	16	0.024	**0.036**
L1LL	6	**0.490**	0.028
	30	**0.084**	0.024
	1	**0.027**	0.021
	20	0.016	**0.020**
MOL	1	**0.392**	0.030
	30	**0.099**	0.026
	15	0.019	**0.023**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
WAL	1	**0.344**	0.030
	30	**0.085**	0.026
	13	0.02	**0.025**
OCL	6	**0.452**	0.028
	30	**0.086**	0.024
	1	**0.024**	0.022
	19	0.014	**0.020**
LOL	7	**0.473**	0.027
	30	**0.082**	0.023
	1	**0.029**	0.021
	19	0.014	**0.019**
Total	48		

From Table 4.24 it can be seen that for each of the salinity variables, the most significant lag was lag 1, with a PMI ranging from 0.768 for MBS up to 0.843 for MAS. For the flow variables, the most significant lag ranged from lag 6 for L1LF, with a PMI value of 0.452, up to lag 8 for L7F, with a PMI value of 0.488. For the river level variables, lag 1 was predominantly the most significant lag selected, however for L1LL, OCL and LOL, the most significant lags were 6, 6 and 7, respectively. As expected, the PMI values for the majority of river level variables were similar to those of the flow variables, with PMI values ranging from 0.198 for L1UL to 0.490 for L1LL. The bivariate stage of the stepwise PMI algorithm reduced the original 480 inputs to 48 inputs. Of particular concern was the inclusion of lag 30 as a significant input for each of the sixteen variables. This indicated that a lagging window of 30 days was unlikely to be sufficient to capture all of the significant lags. However, before increasing the lagging window, the multivariate stage of the stepwise PMI algorithm was conducted to determine if any of the lag 30 inputs would be selected in the final set of significant model inputs.

In the multivariate stage, the lags selected from each of the sixteen variables were combined into one data set consisting of 48 inputs and the stepwise PMI algorithm was used to determine the final set of model inputs. The results from the multivariate stage are given in Table 4.25. Since lag 30 inputs were included in the final set of inputs, it was decided to increase the lagging window to 60 days and repeat the two stage stepwise PMI input selection procedure.

Table 4.25 Stepwise PMI Input Selection Algorithm Results on Multivariate Salinity Data Set With a Lagging Window of 30 Days

Variable	Lag	PMI	95th Percentile Randomised Sample PMI
MAS	1	**0.843**	0.029
WAS	1	**0.193**	0.026
L7F	1	**0.036**	0.013
LOS	30	**0.025**	0.012
L1UL	1	**0.019**	0.011
MBS	30	**0.012**	0.007
L1UL	30	0.007	**0.008**
Total	6		

Setting k_{max} = 60 for each of the sixteen variables gave a total of 960 potential model inputs. Considering the first 60 lags of each variable as potential inputs reduced the total number of data points that were available and necessitated the remaining data to be re-split to ensure statistically similar training, testing and validation sets. In total, 1953 data records were available, from which 1562 records (80%) were used for calibration and 391 records (20%) were used for validation. The 1562 records in the calibration set were further divided into 1250 training records (80%) and 312 testing records (20%). The data were divided using the GA data division technique. The results obtained from the bivariate stage of the stepwise PMI algorithm are shown in Table 4.26. During this stage, the original 960 inputs were reduced to 66 inputs. Some physical interpretation of these results is possible, however, if the results could be completely understood in a physical sense then there would be no need for a sophisticated input determination procedure, since *a priori* knowledge would suffice.

Since the forecasting period is 14 days, it is expected that the ideal inputs would have salinity travel times between their particular upstream location and Murray Bridge that would be greater than 14 days. From Table 4.26, the salinity variable with the highest PMI value is MAS lag 1, with a score of 0.838. During times of low flow (6500 ML/day), the travel time between Mannum and Murray Bridge is approximately 14 days (Maier and Dandy, 1996b). Hence, for a 14-day forecast at Murray Bridge, MAS lag 1 is an ideal input during times of low flow. The second largest PMI value for the salinity variables was obtained by WAS lag 1, with a score of 0.805. During times of intermediate flows (16000 – 17000 ML/day), the travel time between Waikerie and Murray Bridge is approximately 14-16 days (Maier and Dandy, 1996b). Hence, for a 14-day forecast at Murray Bridge, WAS lag 1 is an ideal input during times of intermediate flow. The third largest PMI value for the salinity variables was obtained by LOS lag 1, with a score of 0.784. During times of high flows (30000+ ML/day), the

travel time between Loxton and Murray Bridge is approximately 14.5 days (Maier and Dandy, 1996b). Consequently, for a 14-day forecast at Murray Bridge, it follows that LOS lag 1 would be the most suitable input during times of high flow.

For the flow variables, the most significant lags were lag 6 for L1LF, with a PMI value of 0.465, lag 6 for OCF, with a PMI value of 0.478 and lag 8 for L7F, with a PMI value of 0.501. Inspection of the plots of flow versus salinity indicates that the major variations in flow tend to lead the major variations in salinity by approximately 20-22 days. Consequently, lags 6-8 would be the most important for a 14-day salinity forecast. This is in good agreement with the lags selected by the stepwise PMI algorithm.

The river level variables were included to account for groundwater accessions, which occur along the river and are a function of river level, groundwater level and flow. The groundwater levels change very little with time and may be assumed to remain constant. There is currently no information available to suggest which lags of flows and river level are critical in accounting for this type of phenomenon (Maier and Dandy, 1996b). The stepwise PMI algorithm found that lag 1 was the most significant lag selected for all river level variables except for L1LL, OCL and LOL, in which case, the most significant lags were 6, 5 and 7, respectively.

For the majority of the salinity, flow and river level variables, once the information contained in the lower lags had been accounted for by the first significant input, the second most significant input was a higher lag, ranging from lag 43 to lag 60. This is because the information contained in the first selected input is removed from each of the remaining inputs and the output by using a nonlinear regression. Since the residuals are used to select the second most significant input, the inputs still containing additional information are more likely to be the higher lag inputs. Similarly, for the third significant input, the general trend for the majority of variables was the selection of an intermediate lag, since after removing the information of both the low and higher lag inputs these are the inputs still containing some useful, yet independent information. The fourth significant input was usually a lower lag input, indicating that the first selected lower lag input had not exhausted all useful information contained in the lower lag inputs. For all variables, the PMI score rapidly diminished after the selection of the first or second significant input. The results from the multivariate stage of the stepwise PMI algorithm are given in Table 4.27. During this stage, the 66 inputs were reduced to a final subset of 13 significant model inputs. Only one input at lag 60 (i.e. MOS lag 60) was selected in the final set of inputs. This input had a relatively low PMI score of

0.0054. Therefore, it was concluded that a lagging window of k_{max}= 60 was sufficiently large enough for the salinity case study under investigation.

Table 4.26 Stepwise PMI Input Selection Algorithm Results on Bivariate Salinity Data Sets With a Lagging Window of 60 Days

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
MBS	1	**0.765**	0.027
	56	**0.074**	0.027
	19	**0.036**	0.026
	7	0.023	**0.024**
MAS	1	**0.838**	0.029
	57	**0.069**	0.026
	12	**0.041**	0.025
	34	**0.028**	0.023
	5	0.019	**0.022**
MOS	1	**0.767**	0.030
	47	**0.143**	0.021
	20	**0.050**	0.019
	60	**0.023**	0.015
	33	0.013	**0.014**
WAS	1	**0.805**	0.029
	43	**0.140**	0.019
	20	**0.036**	0.014
	58	**0.020**	0.011
	9	**0.011**	0.010
	32	0.009	**0.010**
LOS	1	**0.784**	0.032
	50	**0.153**	0.027
	25	**0.071**	0.021
	60	**0.022**	0.014
	13	**0.015**	0.012
	34	0.090	**0.010**
L1LF	6	**0.465**	0.028
	60	**0.106**	0.024
	27	**0.037**	0.021
	1	**0.022**	0.020
	46	**0.019**	0.018
	59	0.008	**0.019**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
OCF	6	**0.478**	0.027
	48	**0.116**	0.024
	60	**0.027**	0.021
	1	**0.028**	0.022
	28	0.017	**0.019**
L7F	8	**0.501**	0.027
	55	**0.128**	0.023
	32	**0.039**	0.020
	1	**0.029**	0.019
	60	0.014	**0.017**
MBL	1	**0.306**	0.041
	57	**0.109**	0.042
	21	**0.078**	0.040
	11	**0.050**	0.039
	34	**0.041**	0.037
	45	0.028	**0.037**
MAL	1	**0.340**	0.037
	57	**0.099**	0.036
	24	**0.067**	0.035
	10	**0.036**	0.035
	40	0.029	**0.033**
L1UL	1	**0.201**	0.036
	60	**0.115**	0.035
	39	**0.049**	0.034
	20	0.030	**0.035**
L1LL	6	**0.503**	0.027
	46	**0.102**	0.023
	60	**0.030**	0.021
	1	**0.034**	0.020
	25	**0.020**	0.017
	36	0.008	**0.018**
MOL	1	**0.403**	0.028
	60	**0.121**	0.025
	25	**0.044**	0.024
	45	**0.025**	0.022
	2	0.023	**0.023**
WAL	1	**0.354**	0.029
	60	**0.103**	0.026
	40	**0.040**	0.026
	14	0.024	**0.025**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
OCL	5	**0.465**	0.027
	60	**0.119**	0.024
	31	**0.038**	0.022
	1	**0.026**	0.021
	48	0.019	**0.019**
LOL	7	**0.488**	0.025
	48	**0.121**	0.024
	60	**0.029**	0.020
	1	**0.035**	0.021
	28	0.016	**0.017**
Total	66		

Table 4.27 Stepwise PMI Input Selection Algorithm Results on Multivariate Salinity Data Set With a Lagging Window of 60 Days

Variable	Lag	PMI	95th Percentile Randomised Sample PMI
MAS	1	**0.8383**	0.0292
WAS	1	**0.1923**	0.0264
L7F	1	**0.0360**	0.0125
WAS	43	**0.0252**	0.0106
L1UL	1	**0.0189**	0.0118
LOS	25	**0.0189**	0.0077
MAL	57	**0.0126**	0.0054
MBL	1	**0.0075**	0.0044
MOS	60	**0.0054**	0.0019
MBL	11	**0.0024**	0.0012
MBL	21	**0.0013**	0.0006
MBL	34	**0.0008**	0.0003
MBL	57	**0.0003**	0.0002
MOS	20	0.0002	**0.0003**
Total	13		

Determination of model inputs using PCA-GAGRNN (Method 2)

The PCA-GAGRNN input determination method is described in Section 3.5.2. This method utilises an unsupervised method (PCA), to reduce the input dimensionality, and a supervised method (GAGRNN), to decide which PCs are important for forecasting salinity at Murray Bridge, 14 days in advance. The GAGRNN is a *model-based*

approach to input determination and does not directly rely on *a priori* knowledge of the system under investigation. The GA optimises the final input subset based on forecasting error and the tendency is to only include in the model those inputs that contribute to reducing the forecasting error. However, some *a priori* knowledge is assumed in determining the initial set of candidate variables and to select the maximum number of lags k_{max} to be considered as potential inputs. In the absence of *a priori* knowledge for selecting k_{max}, the PCA-GAGRNN has the disadvantage that it is unable to determine if the lagging window k_{max} has been made large enough to capture all significant lags. In accordance with the findings obtained using Method 1, $k_{max} = 60$ was used for each of the sixteen variables.

Due to the large number of potential inputs (960), PCA was performed using a two step procedure in the same way the stepwise PMI algorithm was implemented. In the first step, PCA was conducted on the lags of each variable in turn, in order to find a subset of PCs. Since the PCA is conducted on one variable at a time, this stage is referred to as univariate PCA. In the second step, the PCs that have been identified for each variable are combined into one data set. PCA was then conducted on this data set to capture interrelations among variables and to reduce the total number of PCs. This second step is referred to as multivariate PCA. The PCA results are given in Table 4.28. Combining the PCs selected from each of the sixteen variables gave a total of 86 inputs. These 86 inputs were then reduced to 36 PCs during the multivariate PCA. The supervised component of the technique (i.e. GAGRNN) used these 36 PCs to select the important PCs for forecasting salinity at Murray Bridge, 14 days in advance. In this research, the GA had a population size of 50 and was run for 100 generations. The objective function was the test set RMSE. Using this technique, the following 16 PCs were selected as the final set of model inputs: PCs 1, 4, 7, 8, 11, 12, 14, 15, 21, 22, 23, 24, 26, 27, 31 and 33. Although only 16 PCs were chosen, each PC is a linear combination of the primary inputs and it is difficult to extract any physical meaning from these results.

Table 4.28 Input Subsets Selected using PCA

Input	Unsupervised Technique	
	Univariate PCA	Multivariate PCA
MBS	PCs 1, 2, ..., 6	PCs 1, 2, ..., 36
MAS	PCs 1, 2, ..., 6	
MOS	PCs 1, 2, ..., 7	
WAS	PCs 1, 2, ..., 7	
LOS	PCs 1, 2, ..., 6	
L1LF	PCs 1, 2, ..., 5	
OCF	PCs 1, 2, ..., 4	
L7F	PCs 1, 2, ..., 4	
MBL	PCs 1, 2, ..., 9	
MAL	PCs 1, 2, ..., 7	
L1UL	PCs 1, 2, ..., 4	
L1LL	PCs 1, 2, ..., 4	
MOL	PCs 1, 2, ..., 5	
WAL	PCs 1, 2, ..., 4	
OCL	PCs 1, 2, ..., 4	
LOL	PCs 1, 2, ..., 4	
Total	86	36

Determination of model inputs using SOM-GAGRNN (Method 3)

The final method investigated in this research is the SOM-GAGRNN (Section 3.5.2). This method is similar to Method 2, but uses a SOM for the unsupervised input preprocessing. The advantage of the SOM is that it is able to take into account nonlinear cross-dependence between inputs. For consistency, k_{max} = 60 was used for each of the sixteen variables. Due to the large number of variables, the SOM was also implemented in a univariate step to determine the significant lags for each variable, and then in a multivariate step to ensure limited cross-dependence between the lags selected from each variable. The results of the SOM-GAGRNN input determination procedure are given in Table 4.29. It is difficult to extract physical meaning from the final set of inputs selected by this method. The input variable with the highest PMI score in Method 1 (i..e MAS at a lag of 1 day) was also selected by the SOM-GAGRNN technique. As discussed previously, the travel time between Mannum and Murray Bridge is approximately 14 days during times of low flow. Therefore, for a 14-day forecast at Murray Bridge, MAS lag 1 is an ideal input during times of low flow.

Table 4.29 Input Subsets Selected using SOM-GAGRNN

Input	Unsupervised Technique		Supervised Technique
	Univariate SOM	Multivariate SOM	GAGRNN
MBS	Lags 1, 5, 6, 7, 10, 11, 12, 15, 16, 17, 21, 22, 23, 26, 28, 30, 34, 36, 38, 41, 42, 45, 49, 51, 54	Lags 1, 11, 21, 34, 45	Lags 21, 45
MAS	Lags 1, 3, 5, 7, 8, 10, 12, 13, 15, 17, 20, 22, 25, 30, 34, 37, 49, 51, 53, 55, 56	Lags 1, 7, 17, 34, 49	Lags 1, 7, 34, 49
MOS	Lags 1 23, 26, 27, 29, 31, 32, 38, 41, 42, 44, 46, 48, 49, 55, 57, 58	Lags 1, 23, 46, 55	Lag 23
WAS	Lags 1, 6, 8, 11, 16, 20, 23, 37, 39, 41, 43, 44, 50, 52, 53, 54, 56, 57, 58	Lags 16, 37	Lags 16, 37
LOS	Lags 1, 6, 8, 10, 18, 21, 22, 23, 25, 28, 29, 31, 35, 38, 40, 49, 51, 53, 55, 56	Lags 1, 28, 49	Lags 1, 49
L1LF	Lags 1, 5, 6, 7, 9, 11, 12, 14, 15, 17, 19, 20, 22, 24, 26, 30, 32, 34, 37, 39, 42, 46, 49, 52	Lags 1, 5, 9, 14, 22, 30, 37, 42, 52	Lags 5, 37, 52
OCF	Lags 1, 4, 5, 6, 8, 9, 11, 12, 14, 16, 18, 22, 23, 26, 31, 35, 38, 50, 51, 53, 55, 56	Lags 1, 8, 11, 16, 22, 31, 50	Lags 8, 11, 31
L7F	Lags 1, 4, 6, 9, 12, 17, 18, 20, 22, 25, 26, 28, 31, 34, 36, 40, 43, 46	Lags 1, 9, 17, 22, 28, 34, 43	Lags 22, 28
MBL	Lags 1, 19, 23, 25, 30, 32, 33, 37, 39, 41, 48, 51, 52, 53, 55, 56, 57	Lags 1	Lag 1
MAL	Lags 1, 30, 33, 34, 36, 38, 39, 41, 42, 43, 45, 46, 47, 49, 50, 51, 53, 54, 55, 57, 58, 59	-	-
L1UL	Lags 1, 22, 25, 26, 28, 29, 30, 34, 37, 38, 39, 41, 42, 43, 45, 46, 48, 51, 53, 55	Lag 1	-
L1LL	Lags 1, 4, 5, 8, 10, 11, 14, 15, 17, 20, 21, 24, 29, 33, 36, 48, 52, 54, 55	Lags 1, 29	Lag 29

Continued.

Input	Unsupervised Technique		Supervised Technique
	Univariate SOM	Multivariate SOM	GAGRNN
MOL	Lags 1, 24, 27, 28, 30, 32, 33, 35, 36, 41, 44, 45, 47, 49, 51, 52, 54, 55, 56	-	-
WAL	Lags 1, 6, 9, 11, 20, 23, 24, 27, 29, 31, 32, 35, 36, 38, 41, 42, 45, 48, 50, 56	Lag 1	-
OCL	Lags 1, 5, 7, 10, 14, 17, 20, 32, 35, 36, 38, 39, 41, 42, 43, 45, 46, 48, 51, 53, 55	Lag 1	-
LOL	Lags 1, 5, 7, 9, 16, 19, 20, 21, 23, 25, 27, 29, 33, 36, 38, 48, 50, 52, 54, 55	Lags 1, 16, 25, 33, 48	-
Total	324	53	21

Comparison with previous studies

Previous studies conducted by Maier and Dandy (1996b) and Maier and Dandy (1997a) investigated methods for determining an appropriate set of model inputs for forecasting salinity at Murray Bridge, 14 days in advance. The best set of inputs obtained in each of the aforementioned studies was used to develop ANN models in this research. This provided a basis for comparison between the new input determination methods developed in this research, and the methods used in previous studies.

In the study conducted by Maier and Dandy (1996b), *a priori* information about salinity travel times between upstream sites and Murray Bridge was used in order to choose the initial model inputs. To subsequently reduce the total number of inputs, a sensitivity analysis was conducted on the entire model and a final set of inputs selected. In total, 51 inputs were chosen using this method (Table 4.8) and these included future values of flow and river level data as inputs (i.e. OCF lags -19, -17,..., -1 and L1LL lags -3, -1). To aid the comparison with the input determination methods investigated in the current research, future values of flow and river level data were not included as inputs, thereby giving a reduced set of 39 inputs.

In the study conducted by Maier and Dandy (1997a), the best method for input determination was found to be the method of Haugh and Box (1977). This method utilises cross-correlation analysis to determine the strength of the relationship between the input time series and the output time series at various lags. However, Maier and Dandy (1997a) did not consider river level time series as potential inputs. In addition,

the only flow time series considered was the flow at Lock 1 Lower. A summary of the best set of inputs obtained from their study is given in Table 4.30.

Table 4.30 Input Subset Selected by Maier and Dandy (1997a) Using the Method of Haugh and Box (1977)

Input	Lags
MBS	1, 2
MAS	1, 2, ..., 4
MOS	1, 2, ..., 7
WAS	1, 2, ..., 11
LOS	1, 2, ..., 13
L1LF	1, 2, ..., 10
Total	47

4.8.1.2 Determination of Network Architecture

Since the number of inputs varies depending on the input determination method used, it was no longer considered acceptable to use the default architecture optimised by Maier and Dandy (1998a) for the 51 input salinity data set. Instead, to select an appropriate architecture, the procedure used by Maier and Dandy (2001) was adopted. This involves using the guidelines advocated by Hecht-Nielson (1987) and Rogers and Dowla (1994) in Equations 2.70 and 2.72, respectively. The lower value from the two guidelines was selected as the starting point for a trial-and-error method for determining a suitable number of hidden layer nodes. The effect of decreasing the number of hidden nodes by ten and twenty nodes was investigated. A fourth model architecture was also investigated by using the results of the first three models in an attempt to "zero in" on the best architecture. The network architecture investigated for each of the new input determination methods is given in Table 4.31 (Models 1-1, 1-2, 1-3, 1-4; 2-1, 2-2, 2-3, 2-4; and 3-1, 3-2, 3-3, 3-4).

The effect of network architecture for the ANN models developed using the input subsets identified by the previous studies (i.e. Maier and Dandy (1996b) and Maier and Dandy (1997a)), had already been investigated in those studies. The best architecture found for each of these models is given in Table 4.32 (Models 4 and 5).

4.8.2 Results and Discussion

ANN models were developed using the inputs obtained from the two previous input determination studies and the three new input determination methods (Methods 1, 2 and 3) for forecasting salinity at Murray Bridge, 14 days in advance. The RMSEs for the models developed using the inputs obtained by Methods 1, 2 and 3 are given in Table 4.31.

Table 4.31 Forecasting Errors for ANN Models Using Inputs Obtained by Methods 1, 2 and 3 (the italicised numbers indicate the best model of the four different network architecture investigated)

Input Data Set	Model	Architecture	Training Set	Testing Set	Validation Set
			RMSE (EC units)	RMSE (EC units)	RMSE (EC units)
Method 1	1-1	13-27-1	35.4	37.9	39.1
	1-2	13-17-1	38.0	37.4	40.9
	1-3	13-7-1	35.7	33.7	39.5
	1-4	*13-32-1*	*29.3*	*30.8*	*34.0*
Method 2	2-1	16-33-1	51.1	53.5	56.3
	2-2	16-23-1	47.7	52.5	52.0
	2-3	16-13-1	93.7	97.1	99.7
	2-4	*16-30-1*	*39.6*	*45.3*	*45.3*
Method 3	3-1	21-43-1	36.7	39.1	40.0
	3-2	*21-33-1*	*30.5*	*38.0*	*36.2*
	3-3	21-23-1	38.7	45.8	44.6
	3-4	21-21-1	32.1	38.1	38.5

From Table 4.31 it can be seen that for each method, the network architecture had a significant impact on the generalisation ability of the networks. The sensitivity to network architecture was similar for the models developed using Methods 1 and 3 with both models exhibiting variations of approximately 7-8 EC units in the RMSE when averaged across the training, testing and validation sets. The models developed using the inputs identified by Method 2 were extremely sensitive to the architecture. For Method 2, using 13 hidden nodes (Model 2-3) resulted in a large increase in the forecasting error, indicating there were insufficient hidden nodes to learn the underlying relationship between the inputs and the output. It is important to note that Method 2 utilised PCA, which takes a weighted average of all 960 inputs to produce a set of PCs. Therefore, each of these PCs may have contained some irrelevant information, which could not be discarded by the GAGRNN. This irrelevant information tended to

contaminate the final set of inputs and caused the models developed using this method to be very sensitive to the number of hidden nodes used.

Table 4.32 summarises the results obtained from the best models developed using the inputs from each new method and provides a comparison of these results with the results obtained from the best models using the inputs from the two previous studies. Using the independent validation set to compare the inputs obtained using each method, it is apparent that the model developed using the inputs from Method 1 (Model 1-4) had the best generalisation ability. However, the generalisation ability of the models developed using Methods 1 and 3 were very similar. The validation RMSE of the model with inputs obtained using Method 1 was 34.0 EC units, compared with a validation RMSE of 36.2 EC units when Method 3 was used. Based on generalisation ability, both Methods 1 and 3 provided the most suitable means for selecting inputs, however, Method 1 had the advantage that it produced a more parsimonious model, since it only required 13 inputs compared to the 22 inputs identified by Method 3.

Table 4.32 Comparison of the Best ANN Models Developed Using the Inputs Obtained by Methods 1, 2 and 3 with the Best ANN Models Developed Using the Inputs Obtained in Previous Studies

Input Data Set	Model	Architecture	Training Set RMSE (EC units)	Testing Set RMSE (EC units)	Validation Set RMSE (EC units)
Method 1	1-4	13-32-1	29.3	30.8	34.0
Method 2	2-4	16-30-1	39.6	45.3	45.3
Method 3	3-2	21-33-1	30.5	38.0	36.2
Maier and Dandy (1996b) (*a priori* + sensitivity)	4	39-30-1	43.0	40.3	43.0
Maier and Dandy (1997a) (Method of Haugh and Box)	5	47-35-1	43.9	38.2	46.2

Of the three new techniques, Method 2 had the poorest generalisation ability, as indicated by a validation RMSE of 45.3 EC units. As mentioned previously, PCA retains a large amount of the original information contained in the 960 inputs. The supervised input determination (GAGRNN) only selects the PCs that are important for

forecasting salinity at Murray Bridge. However, even if a PC is selected as an important input, it may still contain a significant amount of ineffective information. This is because each PC is a linear weighted average of all 960 inputs and is therefore, likely to contain many ineffective inputs. Even though the PC itself is a useful input, it is contaminated with a large amount of ineffective information, which adversely affected the generalisation ability of this model.

Comparing the two previous studies, the results indicate that the inputs obtained using *a priori* knowledge combined with a sensitivity analysis produced the model with the best generalisation ability (Model 4). The validation RMSE obtained by this model was 43.0 EC units, which is still significantly larger than the validation RMSEs obtained by Models 1-4 and 3-2. In addition, Models 1-4 and 3-2 had the added advantage of using considerably fewer inputs, and the methods used to develop these models (Methods 1 and 3) were not reliant on empirical knowledge of the system.

The other model developed using the inputs from a previous study was Model 5, which had a validation RMSE of 46.2 EC units. These inputs were found using the method of Haugh and Box, which relies on cross-correlation analyses and as such, has the limitation that it is only able to account for linear dependence between the inputs and output. This method also has the limitation that it does not determine the significant lags for a specific forecasting period (e.g 14 days in the case study considered), which would have increased the number of inputs and therefore, the network size and training time. The addition of unnecessary inputs and the possible omission of any important lags that are nonlinearly related to the output are the most probable cause of the reduced generalisation ability of Model 5. However, it must be pointed out that it is unlikely that the difference in validation RMSE between Models 4 and 5 is significant. This is because both models had very similar training and testing RMSEs. This finding is in agreement with the results obtained by Maier and Dandy (1997a), who also found very little difference between these two input determination procedures.

It is interesting to note that Methods 1 and 3 both identified very different sets of inputs. The inputs found using Method 1 were characterised by a prevalence of inputs at a lag of 1 day, whereas the inputs determined by Method 3 were generally at longer lags. Only 2 river level inputs were found to be important by Method 3, whereas Method 1 identified 7 significant river level inputs. Method 3 also identified a much larger number of flow inputs.

In Figure 4.81 to Figure 4.85 plots are shown of the validation set 14-day forecasts for the best models developed using each input determination method. These plots indicate

that the three new methods each produced models that performed consistently over the range of salinity values encountered in the validation set. Model 2-4 produced fewer extreme under- or over-predictions, however it is apparent from Figure 4.81 to Figure 4.83 that the average magnitude of the under- or over-predictions was much smaller for both Models 1-4 and 3-2. For the intermediate to high salinity values (600- 1200 EC units), it is apparent that Model 4 tended to under-predict the actual salinity (Figure 4.84), while Model 5 tended to over-predict the actual salinity for salinity values greater than 800 EC units (Figure 4.85).

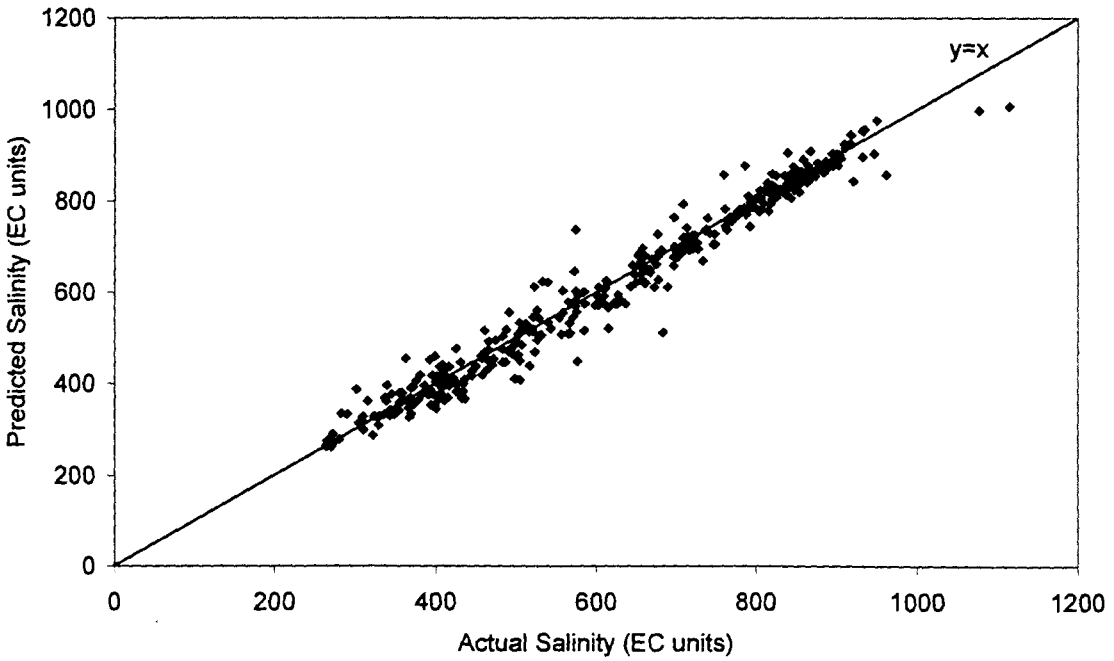

Figure 4.81 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 1 (Model 1-4)

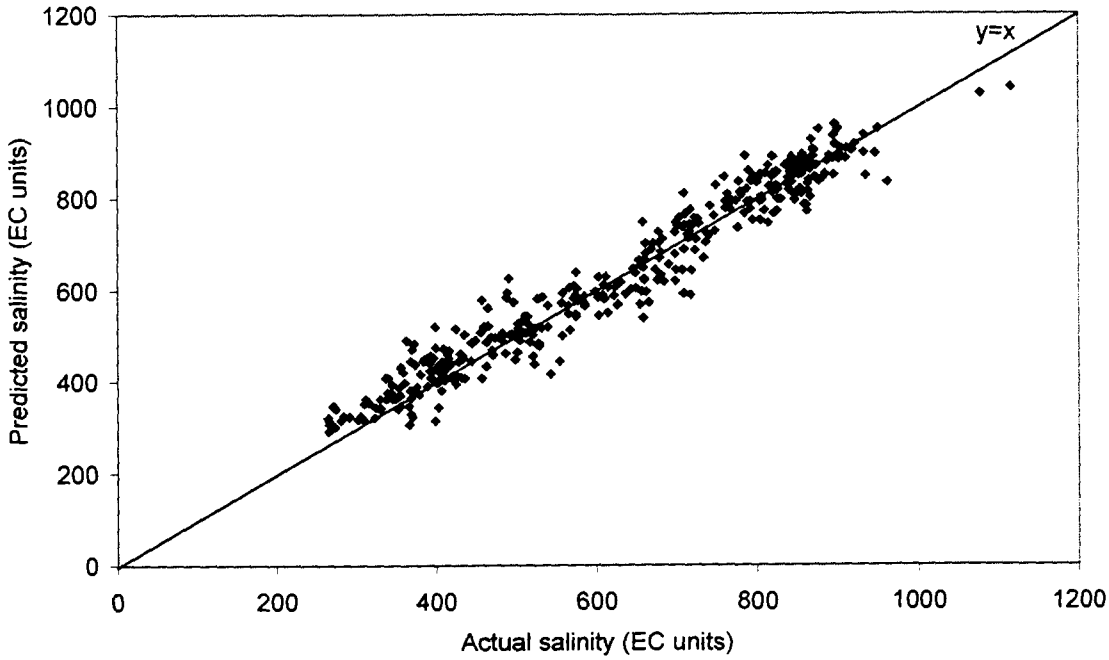

Figure 4.82 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 2 (Model 2-4)

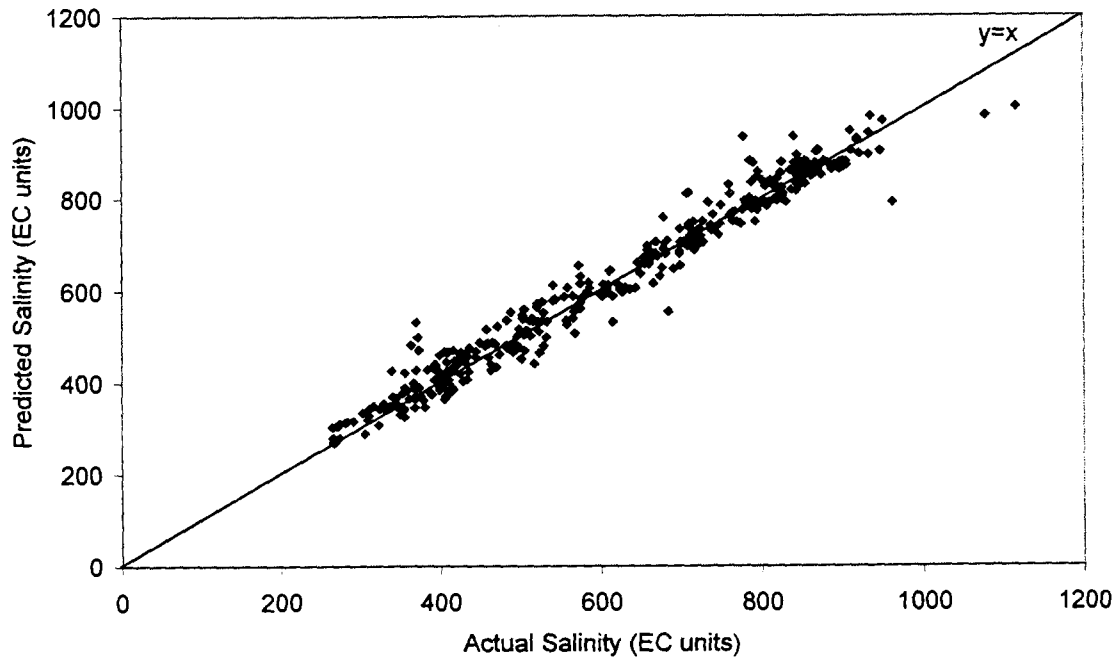

Figure 4.83 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 3 (Model 3-2)

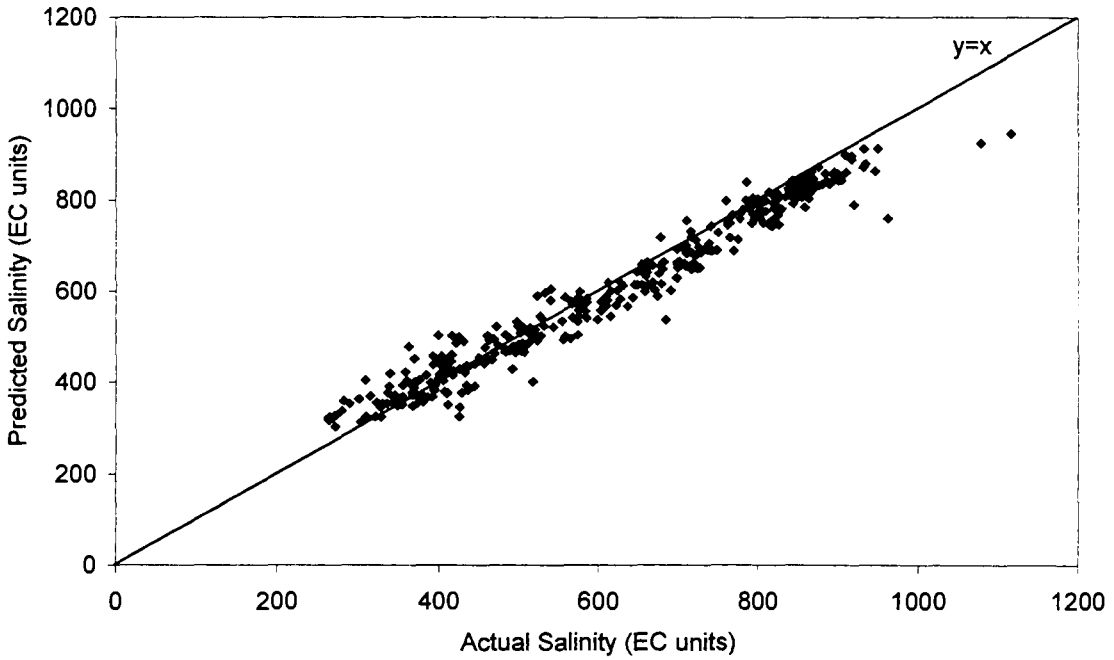

Figure 4.84 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Empirical Knowledge and Sensitivity Analysis (Model 4)

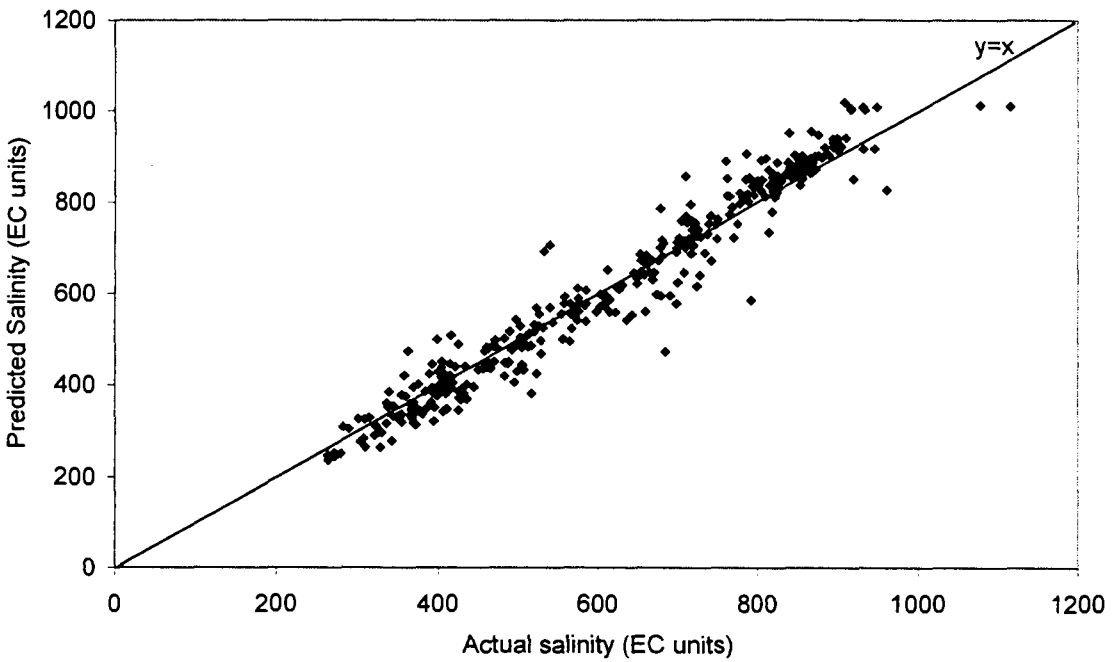

Figure 4.85 Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by the Method of Haugh and Box (Model 5)

An additional advantage of Method 1 is that the PMI scores obtained for each input yield valuable information about the system under investigation, without the need to

develop a model of the system. To investigate this attribute, the PMI scores of each of the inputs identified by Method 1 were compared to the results of a sensitivity analysis conducted using Model 1-4 (Figure 4.86). In performing the sensitivity analysis, each of the network's inputs were increased by 5% in turn. The percentage change in the output as a result of the increase in each of the inputs was then calculated. The equation for calculating the sensitivity of the inputs is given as Equation 4.1 (see Section 4.7.2). The testing data were used to calculate the sensitivity of each input.

From Figure 4.86 it can be seen that there is close agreement between the importance of each of the 13 inputs as measured by the sensitivity and the PMI scores. The five most important inputs used by the ANN model were the five most significant inputs identified by the stepwise PMI algorithm. The remaining inputs with very low sensitivities also had low PMI scores. Therefore, Method 1 provides a means for gaining insight into the order of importance among the significant inputs, and as such, provides valuable information about the system under investigation.

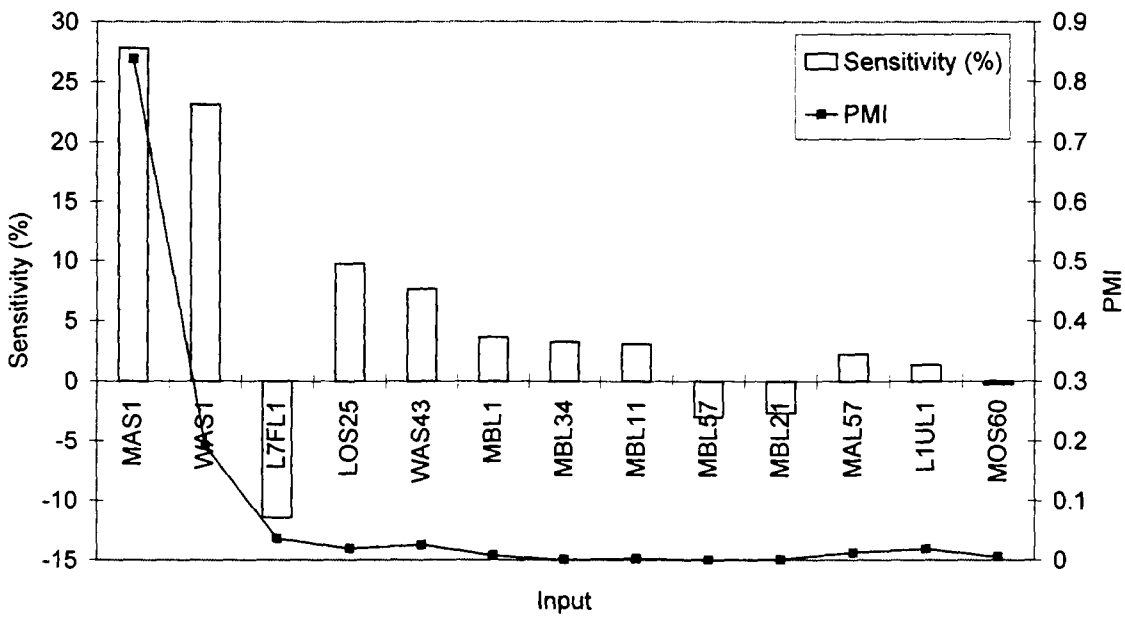

Figure 4.86 Comparison of the Relative Significance of Inputs Obtained Using Sensitivity Analysis and PMI Scores (Model 1-4)

4.8.2.1 Real-time forecasting

To evaluate and compare how each of the models developed using the input subsets identified by Methods 1, 2 and 3 would perform in a real-time forecasting simulation, the second validation set data from 1992 to 1998 were used. As discussed in Sections

4.6.2 and 4.7.2.1, these data are known to contain uncharacteristic data that are outside of the domain of the training, testing and validation data. As a consequence, validating each model on these data provides a realistic indication of how the model would perform when deployed in a real-world application with no model updating (i.e. no retraining). After taking the appropriate lags for each input variable, this data set consisted of 2028 samples from 29-08-1992 to 18-03-1998. Plots of the 14-day forecasts obtained from Methods 1, 2 and 3 are shown in Figure 4.87, Figure 4.88 and Figure 4.89, respectively. For the second validation set data, the most notable regions of poor performance identified in Section 4.6.2 were also found to be regions of poor performance for each of the three models investigated here. These regions of poor performance have been identified in Figure 4.87, Figure 4.88 and Figure 4.89 as Regions 1 and 2. In Section 4.6.2, the SOM analysis technique was used to diagnose that the majority of the data in Regions 1 and 2 were uncharacteristic and outside of the domain of the calibration data. Therefore, it is expected that model performance would be adversely affected in these regions since the models were not calibrated for these events.

In Figure 4.88, the 14-day forecasts were able to account for the major variations in salinity, however, there was a large degree of noise present in the forecasts. This model was developed using the inputs identified by Method 2. As discussed previously, extraneous inputs are included in the PCs used as inputs for this method. It is believed that these extraneous inputs resulted in the large degree of noise present in the forecast produced by this model.

The performances of the three methods for the second validation set are summarised in Table 4.33. To obtain a fair representation of each model's performance, another calculation of the RMSEs was performed with the data from Regions 1 and 2 removed, since these were uncharacteristic data points and no data representative of these two regions had been used to train the models. Despite removing the uncharacteristic regions, the RMSEs were still larger than those obtained in Table 4.32. This can be attributed to the very large size of the second validation data set, which comprised 2028 data records and provides further evidence of the need to periodically retrain the model.

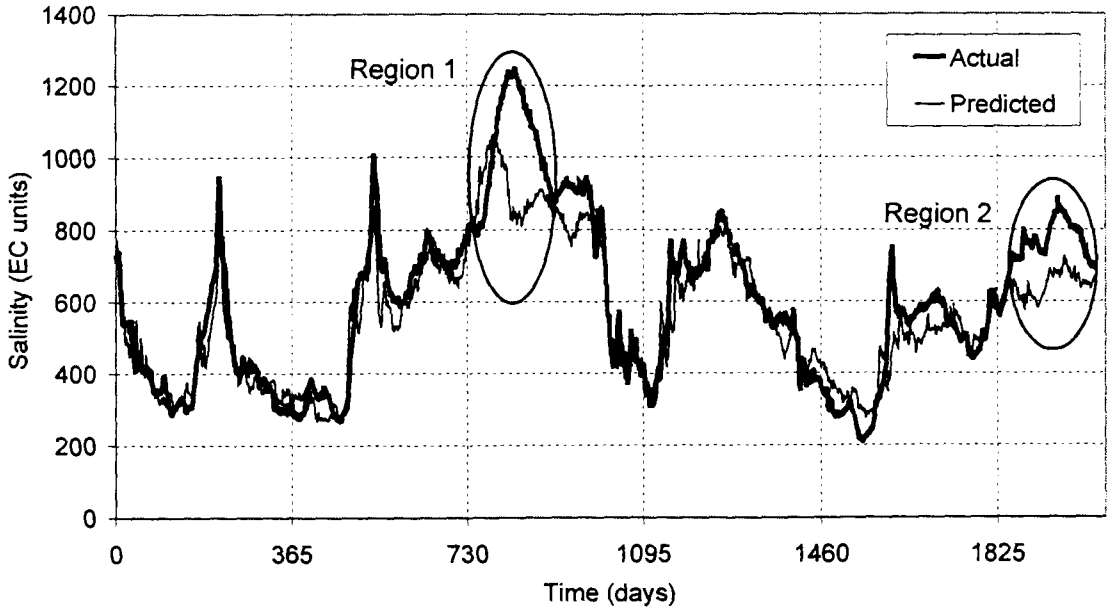

Figure 4.87 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 1 (Model 1-4: August 1992 to March 1998)

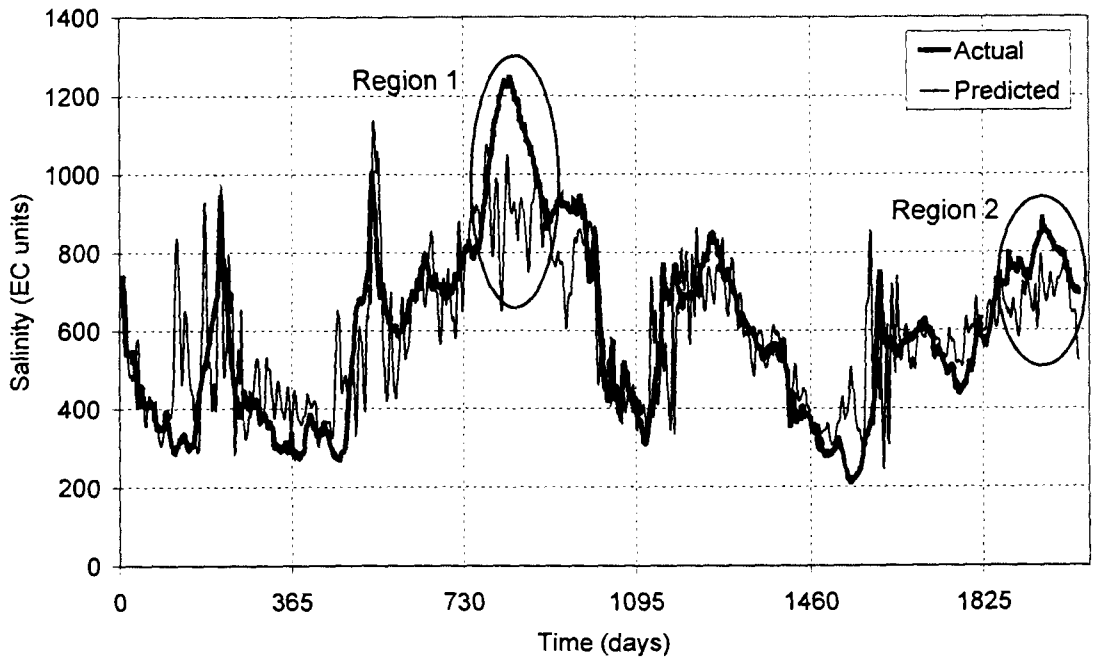

Figure 4.88 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 2 (Model 2-4: August 1992 to March 1998)

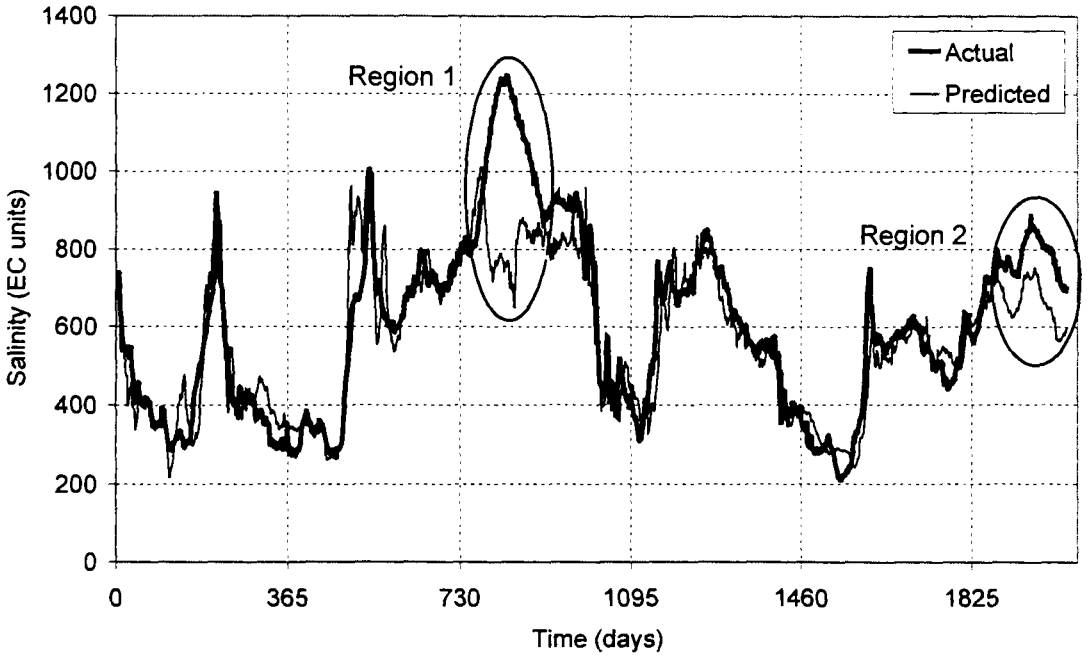

Figure 4.89 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by Method 3 (Model 3-2: August 1992 to March 1998)

Table 4.33 shows that Model 1-4 performed best out of the three input determination methods, closely followed by Model 3-2. Both Models 1-4 and 3-2 outperformed Model 2-4 by a significant margin. These results are in agreement with the results shown in Table 4.32. Based on these results, Model 1-4 was the most robust in the presence of uncharacteristic data.

Table 4.33 Second Validation Set RMSEs of the ANN Models Produced from Methods 1, 2 and 3 for the 14-Day Salinity Forecasts at Murray Bridge

Input Data Set	Model	Architecture	Second Validation Set	Second Validation Set - Regions 1 and 2 Removed
			RMSE (EC units)	RMSE (EC units)
Method 1	1-4	13-32-1	95.0	67.9
Method 2	2-4	16-30-1	135.8	125.8
Method 3	3-2	21-33-1	112.6	72.5

A trial was conducted to investigate whether the performance obtained by Model 2-4 could be improved. In this trial, rather than performing the PCA using a two step procedure (i.e univariate followed by multivariate PCA), the PCA was performed in

one step using the complete set of 960 inputs. The RMSEs for the models developed using the inputs obtained by PCA-GAGRNN #2 are given in Table 4.34. As with the initial implementation of PCA-GAGRNN, the models developed using these inputs were very sensitive to the network architecture. Model 6-4 exhibited the best generalisation ability but required a much larger number of hidden nodes in comparison to Model 2-4, however, it also used a larger number of inputs. The validation set RMSE for Model 6-4 (35.7 EC units) compares very favorably with the best validation set RMSEs obtained by Methods 1 and 3 in Table 4.32. However, Model 6-4 was not robust when applied to the second validation set real-time forecasting simulation. A plot of the 14-day forecasts obtained by the model developed using the inputs identified by PCA-GAGRNN #2 is shown in Figure 4.90. Like the model developed using the original implementation of PCA-GAGRNN, the forecasts exhibit a large degree of noise. As discussed previously, this is thought to be due to the inclusion of extraneous inputs within the weighted average of each PC. Another contributing reason may result from the fact that the second validation data are transformed into PCs using the eigenvectors obtained from the PCA performed on the calibration data (i.e. the data that comprise the training and testing sets). Since the second validation data are known to contain large amounts of data outside of the domain of the calibration data, the resulting PCs may not adequately capture all of the important information in the second validation set data. This problem could be ameliorated by periodically re-conducting the PCA and retraining the model.

Table 4.34 Results of the ANN Models Produced Using Inputs Identified by PCA-GAGRNN #2 for the 14-Day Salinity Forecasts at Murray Bridge. The Italicised Numbers Indicate the Best Model

Input Data Set	Model	Architecture	Training Set	Testing Set	Validation Set
			RMSE (EC units)	RMSE (EC units)	RMSE (EC units)
PCA-GAGRNN #2	6-1	22-45-1	38.1	48.3	43.0
	6-2	22-35-1	55.3	70.5	66.2
	6-3	22-25-1	80.1	94.0	87.4
	6-4	*22-55-1*	*28.2*	*39.0*	*35.7*

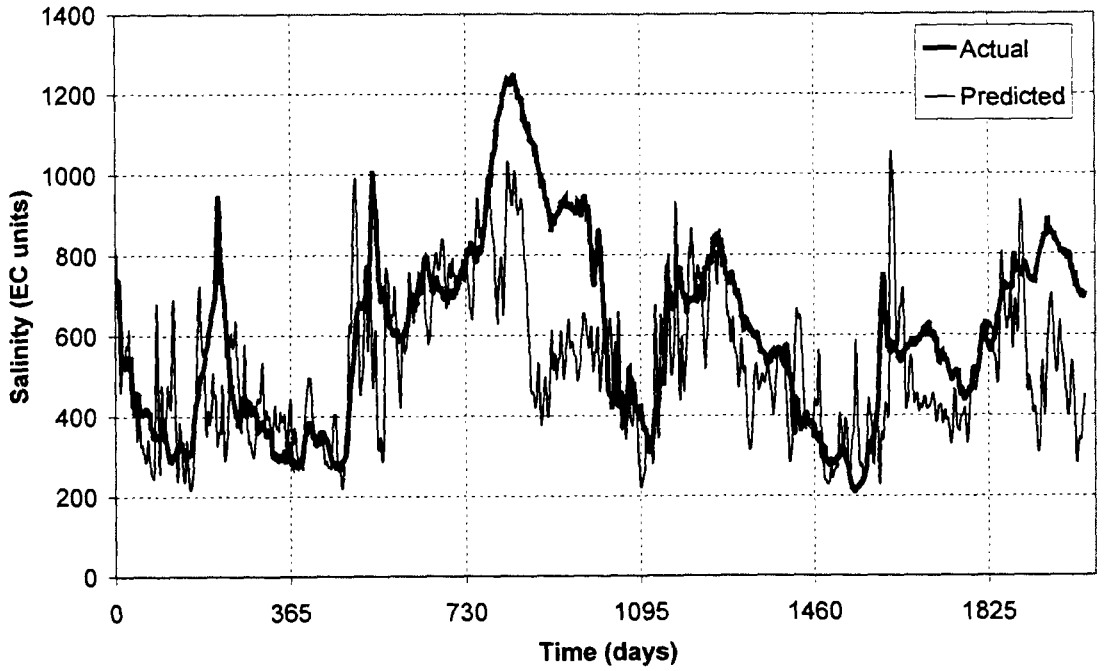

Figure 4.90 Second Validation Set 14-Day Forecasts for the Model Developed Using the Inputs Identified by the PCA-GAGRNN Method 2 (Model 6-4: August 1992 to March 1998)

4.9 Choice of Network Type and Architecture

The two feedforward ANN models proposed in Chapter 3, namely the EBMLP and the GRNN, were used to forecast salinity at Murray Bridge, 14 days in advance. The EBMLP is an evolutionary ANN, which utilises a genetic algorithm to determine suitable network architecture. The GRNN has a fixed architecture, which does not require optimisation. However, two variations of the original GRNN architecture were investigated. The first variation involved clustering the training data using a SOM. Each cluster is then represented by only one pattern layer node, which measures the distance of the input vectors from the cluster's centroid. The second variation involved assigning a separate sigma weight to each input in the GRNN.

4.9.1 Model Development

The data from 01-12-1986 to 30-06-1992 were used to create training, testing and validation data sets. In total, 1953 data records were available, from which 1562 records (80%) were used for calibration and 391 records (20%) were used for validation. The 1562 records in the calibration set were further divided into 1250 training records (80%) and 312 testing records (20%). The data were divided into statistically similar training, testing and validation sets using the GA data division technique (Section 3.3.2).

For the MLP models, the data were linearly transformed by using the original data range to rescale the series to a range that was commensurate with the output transfer function. The ranges used in conjunction with the positive output transfer functions (i.e. linear and sigmoid) were -1.0 to +1.0 for the network inputs and +0.2 to +0.8 for the network outputs. The ranges used in conjunction with the positive/negative output transfer function (i.e. hyperbolic tangent) were -1.0 to +1.0 for the network inputs and -0.8 to +0.8 for the network outputs. For the GRNN models, the data were also linearly transformed. The network inputs and output were scaled between 0.0 and +1.0.

In Section 4.8, the ANN models trained on the input set identified by the stepwise PMI algorithm (Section 4.8.1.1) were found to perform the best for this case study. Consequently, these 13 inputs (Table 4.27) were used in the development of the EBMLP and GRNN models. All other aspects of the modelling process were held constant in accordance with the default selections outlined in Section 4.4.

4.9.1.1 Determination of Network Architecture

Evolutionary backpropagation multilayer perceptron

By nature, the search space for an appropriate MLP architecture is infinite. However, upper limits were placed on the number of hidden layer nodes so that a reasonable size search space could be defined. To determine if the addition of a second hidden layer would enhance the generalisation ability of the ANN model, two hidden layers were considered. Using Equation 2.70 with 13 inputs, the theoretical upper limit for the number of hidden layer nodes is 27. However, the NGO software provides upper limits of 8, 16, 32, 64, 128, or 256 hidden nodes. Therefore, an upper limit of 32 hidden nodes was chosen, as this is the closest to 27. For consistency, an upper limit of 32 hidden layer nodes was also adopted for the second hidden layer.

Three transfer functions were considered for each of the hidden layer and output nodes. These were the hyperbolic tangent, sigmoid and linear transfer functions. The input layer is fixed, therefore, to calculate the total number of possible ANN configurations, it is necessary to consider the total number of configurations of each of the hidden layers and the output layer and to then multiply these figures together.

For example, consider a two node hidden layer with three possible transfer functions, including the hyperbolic tangent (H), sigmoid (S) and linear (L) transfer functions. In this case, there are six combinations of these three transfer functions taken two at a time with repetitions, namely HS, HL, SL, HH, SS and LL. From Kreyszig (1988), the general formula for the number of different combinations of N different things, k at a time, with repetitions, is given by:

$$\binom{N+k-1}{k} = \frac{(N+k-1)!}{k!\left[(N+k-1)-k\right]!} \tag{4.2}$$

Hence, for 32 hidden layer nodes, there are $\frac{(3+32-1)!}{32!\left[(3+32-1)-32\right]!} = 561$ different configurations of the hidden layer. The total number of possible configurations of the ANN architecture with two hidden layers, each with a maximum of 32 hidden layer nodes, 1 output node and three possible transfer functions is $561 \times 561 \times 3 = 9.4 \times 10^5$ different ANNs. Assuming a training time of 5 minutes for each ANN, a total enumeration of the search space would require a total run time of approximately 9 years. Consequently, optimisation algorithms, such as the GA, are clearly suited to this type of problem.

In considering the determination of an optimal ANN architecture using a GA, the makers of the NGO software stated that, "... with genetic algorithms, an excellent solution often appears in less than 1500 evaluations", which for the example in their investigation meant an evaluation of only 0.009% of the total number of possible configurations (BioComp Systems Inc., 1998). Therefore, in this study, 2000 evaluations were considered adequate, which is an evaluation of 0.2% of the total search space. This was achieved by using 100 generations with a population size of 20. The EBMLP model was implemented in NGO using the neural and genetic parameters/settings given in Table 3.11 and Table 3.12, respectively.

In the NGO software, the GA selection method discards the poor chromosome strings, and hence, the population needs to be refilled every generation. Initially the cloning method of refilling was used, whereby the survivors of the selection process are cloned to refill the population. Cloning can lead to faster convergence but may result in premature convergence if the population size is small (BioComp Systems Inc., 1998). Therefore, another refill strategy was also trialled, which randomly creates new population members, introducing fresh stock and thereby helping to avoid search stagnation. In addition, the cloning refill strategy was also trialled using a larger population size of 40. With this size population, the number of generations was reduced to 50 to maintain a total of 2000 network evaluations.

General regression neural network

The first modification to the architecture of the original GRNN model involved clustering the training data using a SOM. In this trial, four different size Kohonen layers were investigated, including layers of size 10 by 10, 20 by 20, 30 by 30 and 40 by 40. The SOM training algorithm utilises a learning rate and a neighbourhood size parameter. For each size Kohonen layer, the learning rate and neighbourhood size were adjusted to achieve a maximum number of clusters on the grid. The learning rate and neighbourhood size used for each Kohonen layer, along with the number of clusters obtained for each size are given in Table 4.35.

Table 4.35 Details of the Learning Rate and Neighbourhood Size Used for Each Size Kohonen Layer

Kohonen Layer Size	Learning Rate	Neighbourhood Size	Number of Clusters
10 by 10	0.1	0.3	94
20 by 20	0.3	0.1	282
30 by 30	0.1	0.3	462
40 by 40	0.1	0.3	604

Once the clusters were obtained, the centroid for each of the M clusters was calculated by taking an average of the N points for each of the p inputs in the cluster.

$$\bar{x}_{jk} = \frac{1}{N_{jk}} \sum_{i=1}^{N_{jk}} x_{jki}, \quad j = 1,...,M; \quad k = 1,...,p \tag{4.3}$$

The second adaptation to the original GRNN architecture involved assigning separate sigma weights to each input in the network. In this case, there were 13 inputs and a GA was used to find the 13-dimensional set of sigma weights to the network. The test set RMSE was used as the fitness function and the GA was run for 1000 generations with a population size of 200.

4.9.2 Results and Discussion

4.9.2.1 EBMLP

The first feedforward ANN investigated was the EBMLP model. Details of the ten best ANN models found during the first GA run are given in Table 4.36. It can be seen that there was a large degree of similarity between the ten best networks. This can be attributed to the method used for refilling the population, which involved cloning the ANNs that survived the selection process. Of particular interest was the fact that each of the best networks required the use of two hidden layers. This further reinforces the recommendations of Flood and Kartam (1994), who advise that even though one hidden layer is theoretically sufficient, in practice, the use of two hidden layers may provide a better approximation of complex functions using fewer connection points. Nine of the ten best networks made exclusive use of the hyperbolic tangent transfer function in the first hidden layer. The other network used a combination of all three transfer functions. The best network obtained from this run utilised the sigmoid transfer function in the second hidden layer and the remaining networks used a combination of both sigmoid and hyperbolic tangent transfer functions in the second

hidden layer. All networks made sole use of the linear transfer function in the output layer.

In Table 4.36, it is evident that the networks with the optimised architecture performed consistently better than the networks that have been developed in previous sections using the default architecture. This highlights the efficacy of using a GA to determine optimal network architecture.

Table 4.36 also shows that some networks with the same architecture had different training and testing set RMSEs. For example, the networks ranked 4, 6 and 8 each made use of 11 nodes with hyperbolic tangent transfer functions in the first hidden layer, 13 nodes with a sigmoid transfer function and 1 node with a hyperbolic tangent transfer function in the second layer and 1 node in the output layer using the linear transfer function. Despite having the same architecture, these three networks each had slightly different training and testing RMSEs. This is because the NGO software does not allow the user to set a random number seed for the generation of the ANN's initial weights. As a result of this feature, if an ANN is trained with the same architecture during the GA run, the network will be initialised with a different set of weights and small differences in the final training/testing set errors between the two networks with the same architecture are likely to result. This allows the effect of different weight initialisations to be taken into account during the evolutionary process.

Table 4.36 Details of the Ten Best Networks Found During GA Run #1

(Generations=100, Population Size=20, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
1	14H	14S	1L	28.6	27.1
2	14H	11S, 1H	1L	29.7	27.7
3	1S, 21H, 1L	13S, 1H	1L	28.8	27.7
4	11H	13S, 1H	1L	29.0	27.8
5	14H	12S, 1H	1L	28.8	27.8
6	11H	13S, 1H	1L	29.4	27.9
7	14H	12S, 1H	1L	29.3	27.9
8	11H	13S, 1H	1L	28.6	27.9
9	14H	12S, 1H	1L	28.7	28.0
10	14H	12S, 1H	1L	29.7	28.0

L = Linear Transfer Func.
S = Sigmoid Transfer Func.
H = Hyperbolic Tangent Transfer Func.

Details of the ten best ANN models found during the second GA run are given in Table 4.37. There was an increase in the diversity of the ten best networks found. This can be attributed to the method used for refilling the population, which involved randomly generating ANNs, thereby introducing fresh stock into the evolutionary process. The architecture of the best ANN found during this run had a similar second hidden layer and the same output layer as the best ANN found during the first GA run, however, the main difference was the configuration of the first hidden layer. In addition, the training set and testing set RMSEs in this run were higher than those obtained in the first run. This indicates that the GA had not yet converged on a region in the architecture search space that provided a level of fitness commensurate with that obtained during GA run #1. This is in agreement with the makers of the NGO software who claim that the cloning method of refill provides faster convergence (BioComp Systems Inc., 1998).

Table 4.37 Details of the Ten Best Networks Found During GA Run #2 (Generations=100, Population Size=20, refill strategy: random)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
1	1S, 23H, 1L	14S, 1H	1L	29.0	28.2
2	1S, 23H, 1L	13S, 1H	1L	30.1	28.7
3	1S, 23H, 1L	15S, 1H	1L	29.0	29.0
4	1S, 23H, 1L	9S, 1H	1L	30.2	29.3
5	12H	19L	1L	29.8	29.7
6	25H, 2L	7H	1H	29.7	29.9
7	1S, 23H, 1L	13S, 1H	1L	30.3	30.1
8	24H, 2L	12S, 1H	1L	29.9	30.2
9	24H, 2L	12S, 1H	1H	31.7	30.2
10	1S, 23H, 1L	14S, 1H	1L	30.3	30.2

Details of the ten best ANN models found during the third GA run are given in Table 4.38. It is apparent that increasing the population size and reducing the number of generations had a negative effect on the fitness of the best ANNs as shown by the higher testing set RMSEs. Nine of the ten best networks obtained during this run did not make use of a second hidden layer.

Table 4.38 Details of the Ten Best Networks Found During GA Run #3 (Generations=50, Population Size=40, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
1	24H, 2L	<none>	1L	31.7	30.0
2	23H, 3L	<none>	1L	31.4	30.0
3	24H, 2L	<none>	1L	31.6	30.1
4	24H, 2L	<none>	1L	31.4	30.3
5	24H, 2L	<none>	1L	31.7	30.4
6	24H, 2L	<none>	1L	31.7	30.5
7	24H, 2L	3S, 1H	1L	30.9	30.5
8	11H	<none>	1L	32.2	30.6
9	24H, 2L	<none>	1L	31.8	30.7
10	24H, 2L	<none>	1L	31.9	30.7

4.9.2.2 GRNN

The second feedforward ANN investigated was the GRNN. Before conducting the trials, it was considered important to determine whether the inverse Hessian method provided a suitable means for optimising the sigma weight (smoothing parameter). Since the inverse Hessian method is a local optimisation technique, the method may have difficulties with a multimodal error surface. Therefore, to evaluate the error surface, the sigma weight was incremented in the range [0.001, 10] using increments of 0.001. For each value of the sigma weight, a GRNN was constructed and the objective function (i.e. test set RMSE) was evaluated. The results of this investigation showed that the error surface was not multimodal, but rather, there was a clearly defined global minimum. The error surface in the vicinity of the global minimum is shown in Figure 4.91. It is apparent that the error curve has continuous first- and second-order derivatives and is readily amenable to optimisation by the inverse Hessian method.

The first GRNN trial involved investigating the effect of clustering the training data using a SOM. The training and test set RMSEs obtained with four different size Kohonen layers, and with no SOM clustering, are given in Table 4.39. The number of clusters obtained for each model are also given in Table 4.39. The GRNN model in which the training records were not clustered can be thought of as a special case, where the number of clusters is equal to the number of training set records. It was apparent that increasing the size of the Kohonen layer, and therefore, the number of clusters, resulted in a decrease in the training and testing set errors. This inverse relationship

can be seen in Figure 4.92, which shows a plot of the test set RMSE with the number of clusters expressed as a percentage of the total number of training set records. This result provides further evidence that the training and testing sets are representative of the same population, since using more of the information contained in the training set resulted in improved generalisation ability as measured by the test set RMSE. The best single-sigma GRNN model was obtained when all of the training set records were used in the pattern layer (i.e. no SOM clustering).

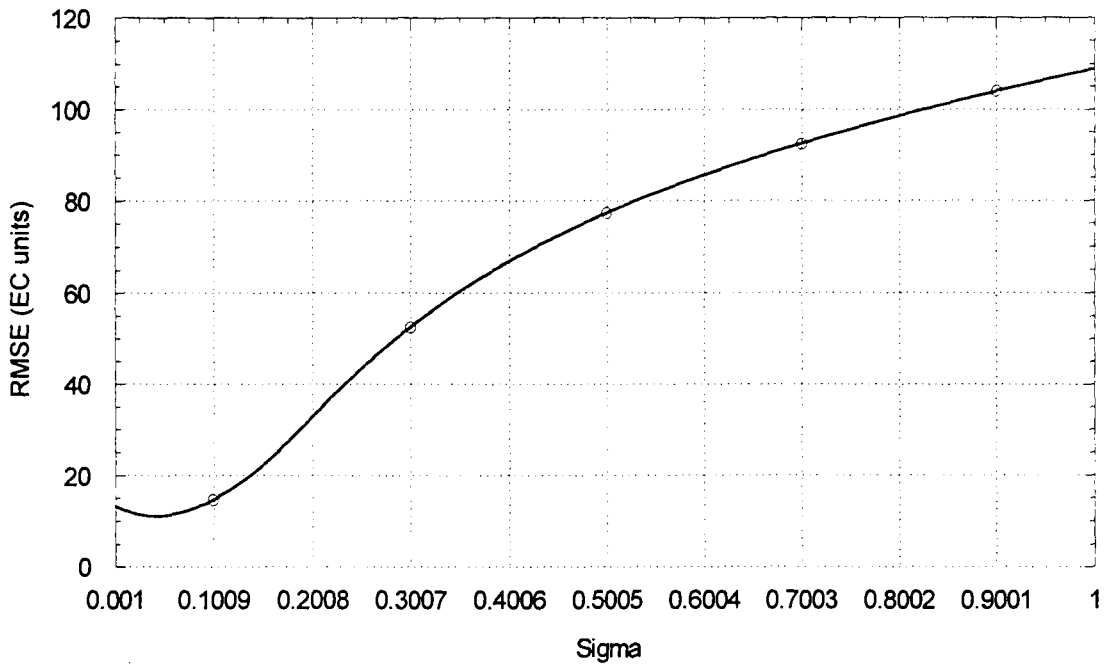

Figure 4.91 Single-Sigma GRNN Error Surface in the Vicinity of the Global Minimum

Table 4.39 Training and Testing Set Forecasting Errors for the GRNN With No SOM Clustering and the Four GRNNs that Utilised SOM Clustering

Kohonen Layer Size	Number of Clusters / Pattern Layer PEs	Training Set	Testing Set
		RMSE (EC units)	RMSE (EC units)
10 by 10	94	35.1	40.5
20 by 20	282	23.8	26.1
30 by 30	462	17.1	20.2
40 by 40	604	12.3	19.1
No SOM Clustering	1250	1.1	11.1

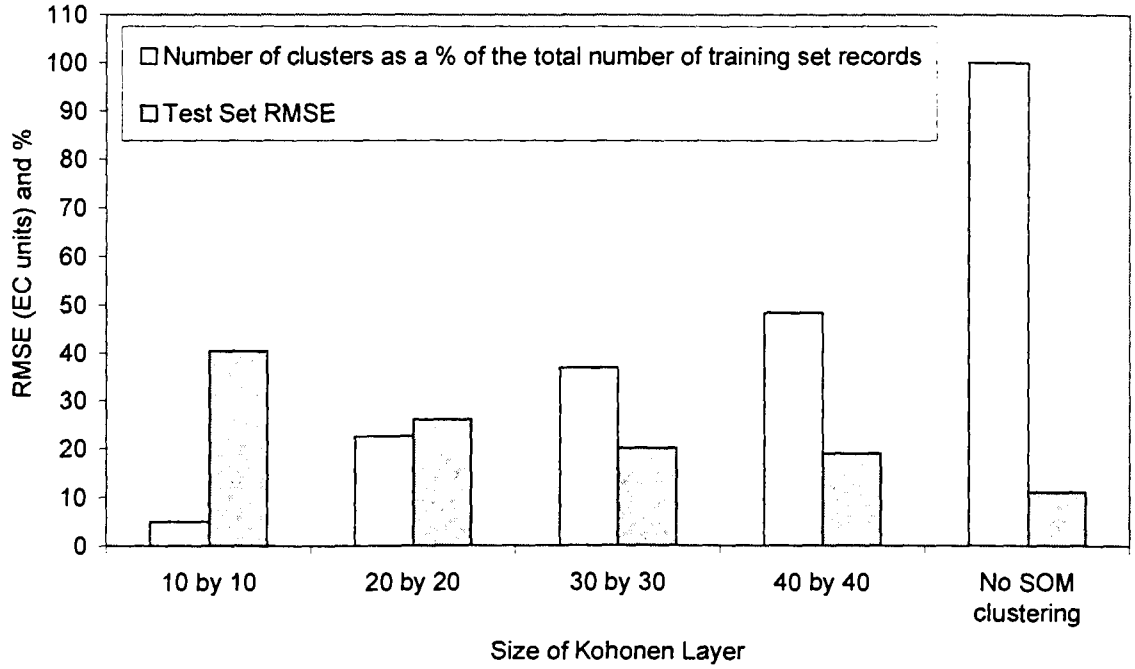

Figure 4.92 Test Set RMSEs and Number of Clusters as a Percentage of the Total Number of Training Set Records For Each Single-Sigma GRNN Investigated

In the second GRNN trial, the effect of using separate sigma weights for each input in the network was investigated. Table 4.40 provides a comparison of the training and testing set RMSEs for both the single- and multiple-sigma GRNNs. The multiple-sigma GRNN slightly outperformed the single-sigma GRNN, as measured by the test set RMSE, however, such a small difference in generalisation ability is not likely to be significant. The multiple-sigma GRNN was trained using a floating point GA and required a total training time of 7116 seconds (approximately 2 hours), as opposed to the single-sigma GRNN, which was trained using the inverse Hessian method and required a training time of only 12 seconds. Since the single-sigma GRNN provided comparable performance to the multiple-sigma GRNN but only required a fraction of the training time, it was selected as the best GRNN model.

Table 4.40 Training and Testing Set Forecasting Errors and Training Times for the Single-Sigma GRNN and the Multiple-Sigma GRNN

Model	Run Time (s)	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
Single-Sigma GRNN	12	1.1	11.1
Multiple-Sigma GRNN	7116	0.6	10.5

A comparison between the best GRNN and EBMLP models using the training, testing and validation set RMSEs is given in Table 4.41. The validation set was independent of the architecture determination phase of modelling and provides the most suitable means of comparing the two types of ANNs. Using the validation set, it can be seen that the GRNN model significantly outperforms the EBMLP, and provides a 53.0% reduction in the validation set RMSE. The EBMLP was also computationally intensive and took 39 hours to complete the GA run compared to 12 seconds for the single-sigma GRNN.

Table 4.41 Training, Testing and Validation Set Forecasting Errors for the Best GRNN and the Best EBMLP Models

Model	Training Set RMSE (EC units)	Testing Set RMSE (EC units)	Validation Set RMSE (EC units)
EBMLP	28.6	27.1	31.0
Single-Sigma GRNN	1.1	11.1	14.5

4.9.2.3 Real-time forecasting

In a real-time forecasting application, it is likely that the forecasting models would be subject to data that are outside of the domain of the calibration data. Therefore, to determine how robust the EBMLP and GRNN models are to the presence of such uncharacteristic data, they were applied to the second validation set data from 1992 to 1998. These data are known to contain uncharacteristic data that are outside of the domain of the training, testing and validation data (see Sections 4.6.2 and 4.7.2.1). As a consequence, validating each model on these data provides a realistic indication of how the model would perform when deployed in a real-world application with no model updating (i.e. no retraining). After accounting for the appropriate lags of each input variable, this data set consisted of 2028 samples from 29-08-1992 to 18-03-1998. Plots of the 14-day forecasts obtained from the EBMLP and the GRNN models are shown in Figure 4.93 to Figure 4.94, respectively. For the second validation set data, the most notable regions of poor performance identified in Section 4.6 were also found to be regions of poor performance for each of the models investigated here. These regions of poor performance have been identified in Figure 4.93 to Figure 4.94 as Regions 1 and 2. In Section 4.6, the SOM analysis technique was used to diagnose that the majority of the data in Regions 1 and 2 were uncharacteristic and outside of the domain of the calibration data. Therefore, it is expected that model performance would be adversely affected in these regions since the models were not calibrated for these events.

In Figure 4.93 and Figure 4.94, it can be seen that both models provided similar performance for the 14-day forecasts and were able to account for the major variations in salinity. However, the EBMLP was able to perform better at forecasting the salinity peaks and was more robust to the presence of the uncharacteristic data.

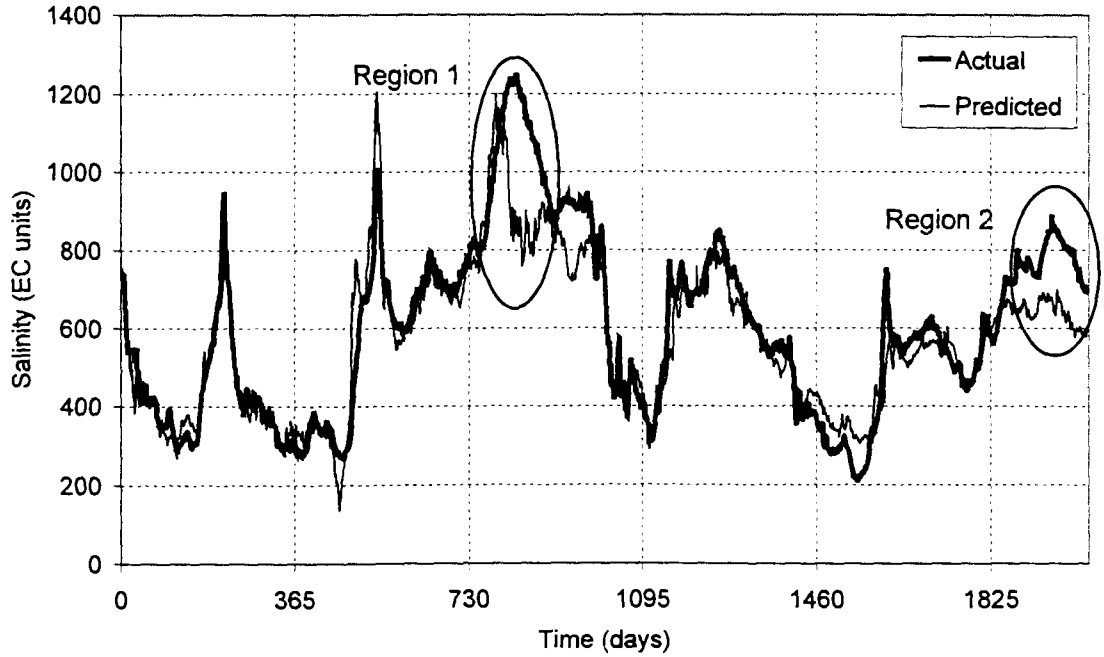

Figure 4.93 Second Validation Set 14-Day Forecasts for the EBMLP (August 1992 to March 1998)

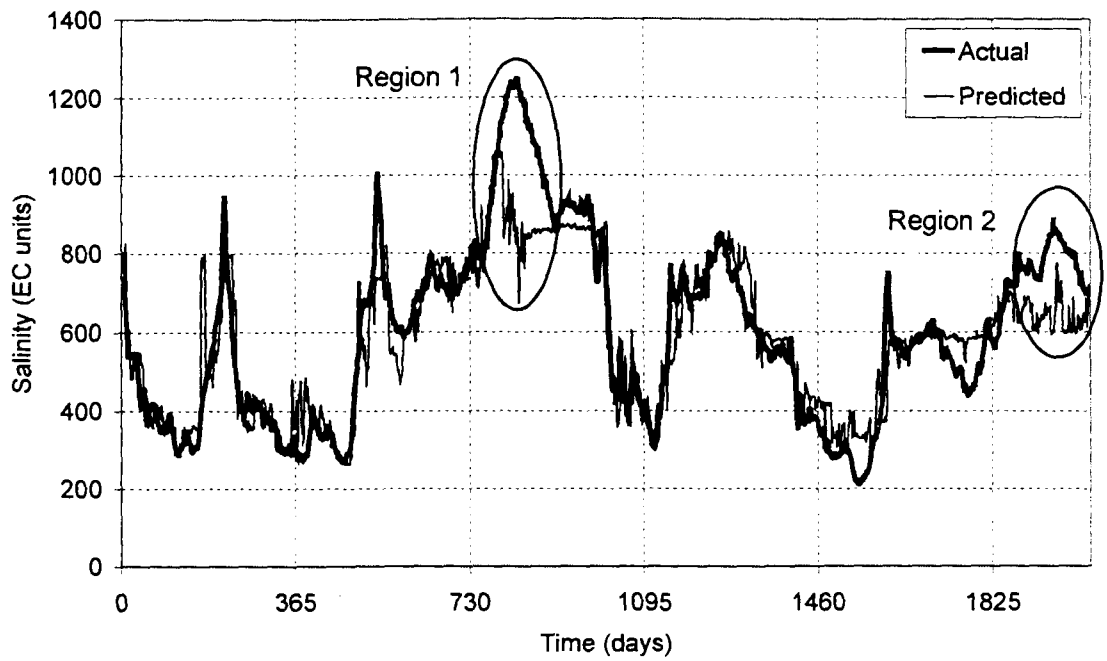

Figure 4.94 Second Validation Set 14-Day Forecasts for the Single-Sigma GRNN (August 1992 to March 1998)

The performances of the best EBMLP and GRNN models for the second validation set real-time forecasting simulation are summarised in Table 4.42. To obtain a fair representation of each model's performance, another calculation of the RMSEs was performed with the data from Regions 1 and 2 removed, since these were uncharacteristic data points and no data representative of these two regions had been used to train the models. Despite removing the uncharacteristic regions, the RMSEs were still high. Once again, this may be attributed to the very large size of the second validation data set, which comprised 2028 data records and provides further evidence of the need to periodically retrain the model.

From Table 4.42, it is evident that the EBMLP model performed better on the second validation set (Regions 1 and 2 removed) than the MLP in Table 4.33 (Model 1). This finding provides further confirmation of the advantages in optimising the MLP's architecture using a GA.

The GRNN provided superior performance when the validation data were known to have the same statistics as the calibration data (Table 4.41) but the EBMLP was slightly more robust as it provided better performance when uncharacteristic data were encountered in the second validation set (Table 4.42). Therefore, both models were retained for the subsequent steps in the ANN modelling methodology.

Table 4.42 Second Validation Set Forecasting Errors for the Best GRNN and the Best EBMLP Models

Model	Second Validation Set	Second Validation Set - Regions 1 and 2 Removed
	RMSE (EC units)	RMSE (EC units)
EBMLP	94.1	64.6
Single-Sigma GRNN	103.9	75.9

4.10 Training (Optimisation)

4.10.1 Choice of Performance Measure

The best EBMLP and GRNN models identified in Section 4.9 were used to determine an appropriate performance measure for each model. All other aspects of the model development process remained as specified in Section 4.9 and the procedure for the determination of a suitable performance measure, outlined in Section 3.7.1, was employed.

When training a MLP using the backpropagation algorithm, the derivative of the error function needs to be calculated analytically. Since a large number of the ten performance measures investigated in this research cannot be differentiated analytically, they were not incorporated directly into the backpropagation algorithm but were used in the assessment of the model's generalisation ability during the cross-validation training process. In so doing, each of the performance measures were used, in turn, to determine when to stop training the MLP.

To implement each of the performance measures for the MLP, the NeuralWorks Professional II/Plus software package was used because it has a capability known as UserIO programming for customising the performance measure. For each of the ten measures, a UserIO program was written in C source code and then compiled using the DOS batch files provided by the makers of the NeuralWorks Professional II/Plus software. This enabled the performance measure to be used in conjunction with the cross-validation (SaveBest) command in the software.

The GRNN identified as having the best performance in Section 4.9 was the single-sigma GRNN, trained using the inverse Hessian method. The inverse Hessian method employed in this research uses derivatives of the error function, which are calculated numerically, using the method of divided differences. Therefore, the ten performance measures were incorporated directly into the GRNN training algorithm. A test set was used to determine when the inverse Hessian method had converged on a suitable value for the sigma weight. The single-sigma GRNN Fortran source code was adapted to incorporate each of the ten performance measures.

The weights for the MLP and the GRNN were initialised to the same values for each of the ten performance measures investigated. After each set of near-optimal weights W_j were identified, these weights were then used to compute values P_{jk} for the nine

remaining performance measures. The results of this stage of the procedure are shown in Table 4.43 and Table 4.44 for the MLP and the GRNN, respectively.

Inspection of Table 4.43 reveals there was little difference in the model performance between each of the MLP models developed. However, the model developed using performance measure E_4 consistently produced low values in each of the other error measures and high values of the correlation-based measures, while the model developed using performance measure E_8 produced high values in each of the remaining error measures and lower values of the correlation-based measures. Table 4.44 also shows very little difference in the model performance between each of the GRNN models developed. The only exception is the model developed using performance measure E_3, which produced consistently higher values in each of the other error measures. It is not surprising that similar model performance was obtained for each GRNN developed, given the simple nature of the optimisation problem.

In the next stage of the analysis, the actual values of P_{jk} were replaced by ranks R_{jk}. The ranks were computed according to the magnitude of the items included in each group of P_{jk} values identified by a given value of the index k. In the case where two or more entries in Table 4.43 or Table 4.44 have the same rounded value, the ranks were determined according to the value of the next significant digit not shown in Table 4.43 or Table 4.44. The ranks for each performance measure are given in Table 4.45 and Table 4.46 for the MLP and the GRNN, respectively.

Table 4.43 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Sets of Weights W_j - MLP

j=	1	2	3	4	5	6	7	8	9	10
k= 1	19.9	20.7	19.9	19.7	21.7	20.7	20.5	24.6	21.7	21.3
2	26.4	27.0	25.9	26.6	28.8	27.0	27.1	31.8	28.8	27.7
3	0.485	0.417	0.411	0.408	0.445	0.417	0.453	0.456	0.445	0.436
4	3.657	3.871	3.820	3.551	3.816	3.871	3.713	4.721	3.816	3.934
5	0.00258	0.00265	0.00286	0.00239	0.00263	0.00265	0.00256	0.00407	0.00263	0.00272
6	0.982	0.981	0.982	0.981	0.978	0.981	0.981	0.973	0.978	0.980
7	0.982	0.983	0.984	0.982	0.982	0.983	0.982	0.980	0.982	0.982
8	0.0686	0.0778	0.0992	0.0414	0.1266	0.0778	0.0698	0.1228	0.1266	0.0613
9	0.000524	0.000561	0.000537	0.00053	0.000598	0.000561	0.000539	0.000774	0.000598	0.000588
10	0.00470	0.00441	0.00490	0.00409	0.00464	0.00441	0.00467	0.00613	0.00464	0.00469

Table 4.44 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Weight W_j - GRNN

j=	1	2	3	4	5	6	7	8	9	10
k= 1	7.3	7.3	8.5	7.3	7.3	7.3	7.3	7.5	7.3	7.3
2	11.1	11.1	13.1	11.1	11.1	11.1	11.1	11.4	11.1	11.1
3	0.219	0.221	0.224	0.217	0.219	0.221	0.221	0.211	0.219	0.222
4	1.286	1.288	1.537	1.285	1.286	1.288	1.288	1.314	1.286	1.291
5	0.00041	0.00041	0.00057	0.00041	0.00041	0.00041	0.00041	0.00043	0.00041	0.00041
6	0.997	0.997	0.996	0.997	0.997	0.997	0.997	0.997	0.997	0.997
7	0.997	0.997	0.996	0.997	0.997	0.997	0.997	0.997	0.997	0.997
8	0.0060	0.0064	0.0158	0.0055	0.0060	0.0064	0.0064	0.0038	0.0060	0.0068
9	0.000090	0.000090	0.000125	0.000090	0.000090	0.000090	0.000090	0.000093	0.000090	0.000090
10	0.00178	0.00178	0.00211	0.00179	0.00178	0.00178	0.00177	0.00187	0.00178	0.00177

Table 4.45 Ranks From P_{jk} Values Given in Table 4.43 - MLP

$j=$	1	2	3	4	5	6	7	8	9	10
$k=1$	4	6	3	2	9	7	5	10	8	1
2	3	5	2	4	9	6	7	1	10	8
3	10	3	2	1	6	4	8	9	7	5
4	2	7	6	1	4	8	3	10	5	9
5	3	6	9	1	4	7	2	10	5	8
6	2	4	1	3	8	5	6	10	9	7
7	5	2	1	4	8	3	6	10	9	7
8	3	5	7	1	9	6	4	8	10	2
9	1	5	3	2	8	6	4	10	9	7
10	8	2	9	1	4	3	6	10	5	7
Total	41	45	43	20	69	55	51	88	77	61

Table 4.46 Ranks From P_{jk} Values Given in Table 4.44 - GRNN

$j=$	1	2	3	4	5	6	7	8	9	10
$k=1$	3	5	10	7	1	4	6	9	2	8
2	4	1	10	8	6	2	3	9	5	7
3	5	7	10	2	3	6	8	1	4	9
4	4	6	10	1	2	5	7	9	3	8
5	5	2	10	8	7	3	1	9	6	4
6	4	1	10	8	6	2	3	9	5	7
7	5	2	10	8	7	3	1	9	6	4
8	5	7	10	2	3	6	8	1	4	9
9	4	2	10	8	6	1	3	9	5	7
10	5	3	10	8	7	4	2	9	6	1
Total	44	36	100	60	48	36	42	74	46	64

The sums of the ranks in the vertical lines of Table 4.45 and Table 4.46 are indicative of the performance measure's ability to produce low values of other performance measures, and as such, give some idea of the relative merits of each function. From Table 4.45 it is apparent that the lowest sum is obtained for the column $j = 4$, indicating that the set of weights W_4, which were found using the performance measure E_4 (AAPE, Equation 3.71), were also able to produce low values for many of the other performance measures. Based on the sums in Table 4.45, the next best performance measure was E_1 (AAE, Equation 3.66). This performance measure is also based on the sum of absolute relative deviations and is very similar to the performance measure that

Diskin and Simon (1977) found to be the best across a wide array of watersheds, models and applications. The next best performance measure was E_3 (Minimax objective function, Equation 3.68), followed by E_2 (RMSE, Equation 3.67).

Based on the summations (merit numbers) in Table 4.45, there was a large amount of difference separating the best performance measure (AAPE) from the remaining measures. It is important to note that these results are dependent on the case study under investigation, the data sets (training and testing) used in the analysis and the type of ANN model and optimisation algorithm used.

Repeating the procedure for the GRNN model led to a different ordering of the performance measures as shown in Table 4.46. The two performance measures with the equal lowest summation value were E_2 (RMSE, Equation 3.67) and E_6 (CE, Equation 3.73). These were followed by E_7 (r^2, Equation 3.74) and E_1 (AAE, Equation 3.66). The top four performance measures were all separated by a small margin. This is because finding a suitable value for a single sigma weight is a simple optimisation problem, exhibiting a single global minimum in most circumstances. As a result of this, a similar value for the sigma weight was found when using each of the performance measures, resulting in similar model performances for each GRNN model developed.

The merit numbers in Table 4.45 and Table 4.46 were ranked for each ANN model and these results are shown in Table 4.47. As mentioned above, the ranks assigned to each performance measure were not the same for each ANN model and this is evident in Table 4.47.

Table 4.47 Merit Numbers Assigned to Various Performance Measures For Both ANN Models

Performance Measure	MLP	GRNN
1	2	4
2	4	1
3	3	10
4	1	7
5	8	6
6	6	2
7	5	3
8	10	9
9	9	5
10	7	8

4.10.2 Choice of Optimisation Method

4.10.2.1 Feedforward MLP

The architecture determined using the EBMLP (Section 4.9) was also used to determine a suitable training (optimisation) method. In addition, the best performance measure identified in Section 4.10 for MLPs (i.e. AAPE) was used to assess the generalisation ability of each model developed. Generalisation ability was used as the sole criterion for the assessment of model performance because the architecture of each model was fixed, and hence, parsimony constraints were not an issue. The training speed was not considered to be important in this study. All other aspects of the model development process (i.e. data division, transformation and model inputs) remained as specified in Section 4.9.

The following methods for training feedforward MLPs were investigated: the generalized delta (GD) rule; the normalized cumulative delta (NCD) rule; the delta-bar-delta (DBD) algorithm; the extended delta-bar-delta (EDBD) algorithm; the QuickProp (QProp) algorithm; and, the MaxProp (MProp) algorithm. Each of these algorithms have been discussed in Section 3.7.2.2. For each training algorithm, 30 different sets of initial weights were used to provide a fair comparison of the training algorithms. This was achieved by using 30 different random seeds for each of the six optimisation algorithms, thereby giving rise to the development of 180 ANN models. The same 30 random number seeds were used for each method investigated and the initial weights were uniformly distributed in the range –0.1 to +0.1. In accordance with the study conducted by Maier and Dandy (1999), the mean, maximum and minimum of the AAPE was calculated for each of the 30 models developed. The values 2 standard deviations above and below the mean were used to measure variability.

Initially the epoch size was set at 16. The average, maximum, and minimum AAPEs obtained when the 30 random seeds were used with each of the six different optimisation methods are shown in Figure 4.95, and summarised in Table 4.48. It can be seen that there were large differences in the average forecasting errors for the four first-order methods. The average AAPEs ranged from 3.5% when the EDBD algorithm was used to 5.9% when the NCD rule was used. The generalisation ability of the second-order methods was inferior and this is in agreement with the findings obtained by Maier and Dandy (1999). The average AAPEs for the QProp and MProp algorithms were 6.1% and 8.7%, respectively. Maier and Dandy (1999) note that first-order methods are able to escape local minima in the error surface (depending on the size of

the steps taken in weight space), whereas second-order methods do not have this capability.

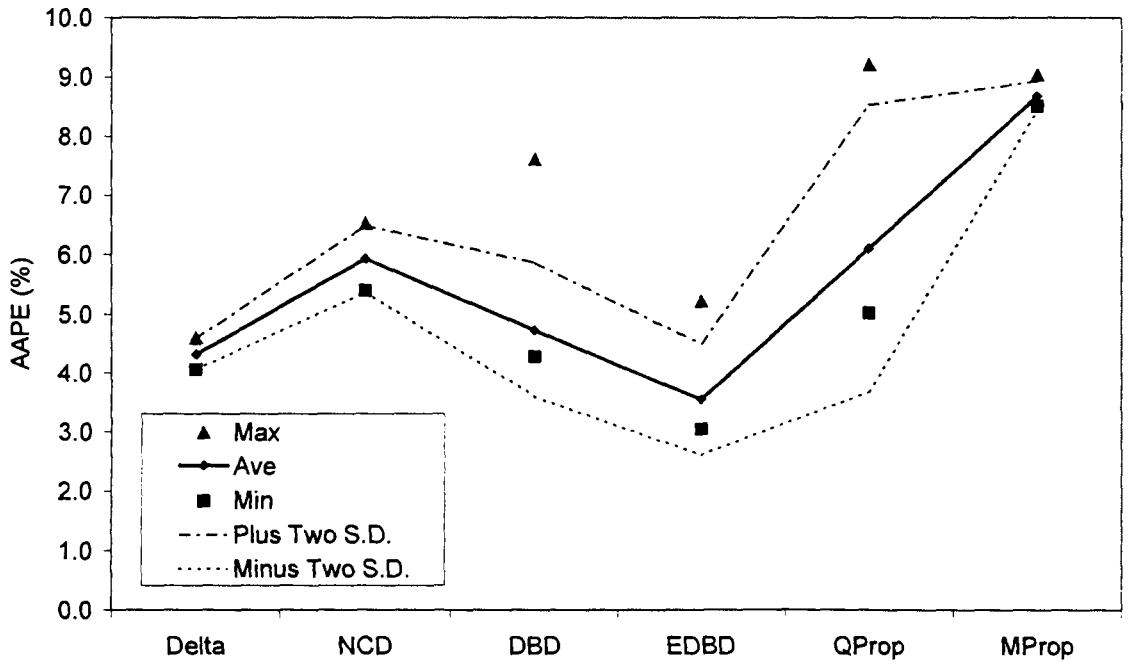

Figure 4.95 Test Set Performance of the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 16)

Table 4.48 Test Set AAPE Statistics for the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 16)

Optimisation Method	Maximum (%)	Average (%)	Minimum (%)	Plus Two Standard Deviations (%)	Minus Two Standard Deviations (%)
Delta	4.6	4.3	4.1	4.6	4.1
NCD	6.5	5.9	5.4	6.5	5.4
DBD	7.6	4.7	4.3	5.9	3.6
EDBD	5.2	3.5	3.0	4.5	2.6
QProp	9.2	6.1	5.0	8.5	3.7
MProp	9.0	8.7	8.5	8.9	8.4

Comparing the results obtained using the GD and NCD learning rules, it is apparent from Figure 4.95 that using an epoch size greater than one had a negative impact on generalisation ability. This indicates that the stochasticity introduced by updating the weights after the presentation of each training sample was successful in enabling the GD learning rule to escape local minima. When the GD rule was used, the results

exhibited a decreased variability in comparison to the NCD rule. This indicates that the GD rule was more consistent in converging to the same local minimum in the error surface. When the first-order algorithms that automatically adjust the size of the steps taken in weight space as learning progresses were used (i.e. DBD and EDBD), they helped to overcome the reduced generalisation ability that resulted from using an epoch size greater than one. However, the variability of the results obtained for the DBD and EDBD algorithms was larger than that obtained for the GD and NCD learning rules.

When the QProp algorithm was used, the variability obtained from starting the training from different regions of the weight space was greater than all of the other methods (Figure 4.95). This indicates that the models trained using this algorithm were unable to converge on the same local minimum in the error surface. The most likely reason for this finding is the inability of second-order methods to escape local minima. The models developed using the MProp algorithm exhibited much lower variability in the results produced. This may have resulted from the fact that the algorithm uses the maximum of the steps computed using the QProp algorithm and the GD learning rule. The model with the lowest forecasting error overall was trained using the EDBD algorithm and had an AAPE of 3.0% (Table 4.48).

Repeating the trials using an epoch size of 100 resulted in the average, maximum, and minimum AAPEs shown in Figure 4.96, and summarised in Table 4.49. It is important to note that the results presented in Figure 4.96, and Table 4.49 for the GD learning rule are the same as those presented in Figure 4.95 and Table 4.48, because this algorithm does not use an epoch size.

The results obtained using an epoch size of 100 reflect the general trends obtained when using an epoch size of 16 (Figure 4.95), with the exception of the results from the NCD learning rule. With an epoch size of 100, the NCD learning rule was unable to escape from local minima in the error surface and this resulted in the algorithm becoming trapped in a sub-optimal region of the weight space. The large degree of variability in the results obtained by the NCD learning rule, indicates that this algorithm was unable to converge on a single local minimum. By systematically reducing the epoch size, it was discovered that the epoch size needed to be below 40 for this case study, in order to yield acceptable generalisation ability with the NCD learning rule (i.e. AAPE $\approx$ 6.0%).

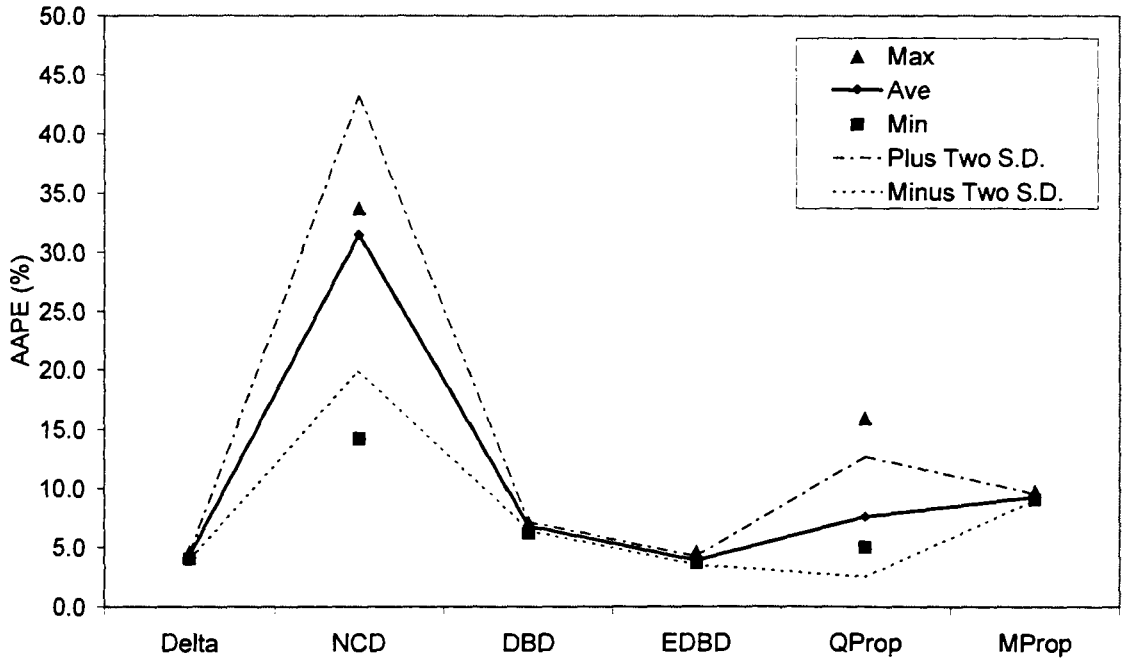

Figure 4.96 Test Set Performance of the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 100)

Table 4.49 Test Set AAPE Statistics for the Networks Trained With the Six Optimisation Methods Investigated (Epoch Size = 100)

Optimisation Method	Maximum (%)	Average (%)	Minimum (%)	Plus Two Standard Deviations (%)	Minus Two Standard Deviations (%)
Delta	4.6	4.3	4.1	4.6	4.1
NCD	33.7	31.5	14.2	43.1	19.9
DBD	7.1	6.9	6.2	7.2	6.5
EDBD	4.6	3.9	3.7	4.3	3.5
QProp	15.9	7.6	5.0	12.7	2.5
MProp	9.8	9.3	9.1	9.6	9.1

For each of the remaining training methods, Table 4.49 shows that the average AAPEs were also higher than those obtained when using an epoch size of 16 (Table 4.48). Contrary to the recommendations made by NeuralWare (1997), using a larger epoch size did not improve the results obtained when training the models with the QProp algorithm. With an epoch size of 100, the degree of variability in the results obtained using the DBD and EDBD algorithms was reduced. This is because the search space is less stochastic for larger epoch sizes.

4.10.2.2 Multiple-Sigma General Regression Neural Network

In Section 4.9, it was found that the multiple-sigma GRNN did not provide a significant improvement in generalisation ability when compared to the single-sigma GRNN (Table 4.40). However, to ensure that this finding was not a function of the training method used (i.e. GA), an alternative global optimisation algorithm, known as ant colony optimisation (ACO), was used to train a multiple-sigma GRNN. ACO is inspired by the behaviour of ant colonies and is based on the ability of ants to find the shortest route from their nest to a food source by depositing a trail of pheromone. The ACO methodology employed in this research is described in Section 3.7.2. The best performance measure identified for GRNNs in Section 4.10 (i.e. RMSE) was used to assess the generalisation ability of each of the models developed. All other aspects of the model development process (i.e. data division, transformation and model inputs) remained as specified in Section 4.9.

In the *Max-Min* Ant System (MMAS) used in this research, the following parameters need to be determined:

- Lower bound for all sigma weights - σ_j^-
- Upper bound for all sigma weights - σ_j^+
- Number of strata - b
- The minimum pheromone trail - τ_{min}
- Pheromone persistence coefficient - ρ
- Number of ants in population - m

The lower bound σ_j^- was set at 0 and the upper bound σ_j^+ was set at 1 for each of the 13 sigma weights in the network. To determine suitable values for the remaining parameters, the set of default parameters outlined in Table 4.50 were used and sensitivity analyses were conducted on each parameter in turn. 100 ant cycles (iterations) were used for the sensitivity analyses.

Table 4.50 Default MMAS Parameters

Parameter	Description	Value
b	Number of strata	100
τ_{min}	The minimum pheromone trail	20
ρ	Pheromone persistence coefficient	0.9
m	Number of ants in population	20

Number of strata

The number of strata determines how each of the sigma weights are to be discretised. Therefore, the number of strata controls the resolution of each sigma weight and hence, determines the size of the search space. Each of the sigma weights were discretised into the same number of strata. Three strata sizes were investigated, including 100, 1000 and 10000. Given the 13 inputs used in this case study, this produced corresponding search spaces of 10^{26}, 10^{39}, 10^{52}, respectively. The results obtained when using the different discretisation sizes are given in Table 4.51. It is apparent that increasing the number of strata did not have a large effect on the overall run time but did result in a decrease in generalisation ability as indicated by the RMSEs obtained. This result is expected, since the same number of ants (i.e. $m = 20$) are being used to search a much larger search space as the number of strata increases. Therefore, the effect of increasing the population of ants while discretising the sigma weights into 10000 strata was also investigated (Table 4.52).

Table 4.51 Run Times and Training and Testing Set Forecasting Errors For Each Number of Strata

Number of Strata	Run Time (s)	Training Set	Testing Set
		RMSE (EC units)	RMSE (EC units)
100	1597	1.1	10.9
1000	1598	5.6	14.9
10000	1661	6.4	17.0

When discretising the parameter space into 10000 strata, increasing the number of ants generally tended to lower the forecasting error (Table 4.52). However, increasing the number of ants to 200 caused the RMSE to increase again slightly. Using 100 ants gave the best performance overall, with a test set RMSE of 12.8 EC units. It was apparent that increasing the number of ants caused a significant increase in the computational time taken to train the models. Using 200 ants required over 2.5 hours of run time. Despite increasing the number of ants, the test set RMSEs of the models developed in Table 4.52 were still higher than those obtained when using 100 strata (Table 4.51). This may be a function of the larger search space when 10000 strata are used and may indicate that an even larger number of ants may be required for this search space. However, increasing the number of ants beyond 200 was not considered because of the large run times required. Therefore, it was decided to discretise each sigma weight into 100 strata for the subsequent GRNN models.

Table 4.52 Run Times and Training and Testing Set Forecasting Errors For Different Population Sizes - 10000 Strata

Number of Ants	Run Time (s)	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
20	1661	6.4	17.0
50	3106	3.3	16.2
100	4256	3.1	12.8
200	9504	5.5	14.1

The minimum pheromone trail

The minimum pheromone trail is a parameter introduced in the MMAS to help prevent search stagnation. The effect of varying the minimum pheromone trail was considered (Table 4.53). Using the test set RMSEs, it can be seen that the models were insensitive to values of the minimum pheromone less than or equal to 100. Increasing the minimum pheromone beyond this point caused the forecasting errors to increase. This indicates that values of the minimum pheromone trail less than 100 were sufficient to prevent the search from stagnating. Therefore, the initial default value of 20 was selected as the minimum pheromone trail to be used in the subsequent trials.

Table 4.53 Training and Testing Set Forecasting Errors For Different Values of The Minimum Pheromone Trail

Min. Pheromone Trail τ_{min}	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
5	1.2	10.9
10	1.1	10.9
15	1.1	10.9
20	1.1	10.9
25	1.1	10.9
50	1.1	10.9
100	0.7	10.9
200	1.5	11.0
500	3.6	13.6

Pheromone persistence coefficient

The pheromone persistence coefficient ρ controls the evaporation of pheromone from each of the paths at the end of each cycle. The pheromone is evaporated in order to prevent premature convergence of the MMAS to a sub-optimal region of the search space. The evaporation loss is $(1-\rho)$, therefore, the lower the value of ρ (i.e. higher evaporation) the slower the convergence to the solution. Table 4.54 shows the performance of the MMAS in training the GRNN when different values of the pheromone persistence coefficient were used. The best performance was obtained with intermediates values (i.e. ρ = 0.4 - 0.6). No evaporation (i.e. ρ = 1.0) caused the MMAS to prematurely converge on a sub-optimal region of the search space as indicated by the higher training and testing set RMSEs. The best test set performance was obtained when ρ = 0.6 was used. Therefore, this value was retained for the subsequent trials.

Table 4.54 Training and Testing Set Forecasting Errors for Different Values of the Pheromone Persistence Coefficient

Pheromone Persistence Coefficient ρ	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
0.1	1.2	11.1
0.2	1.3	11.1
0.3	1.3	11.1
0.4	1.7	10.9
0.5	1.0	10.7
0.6	1.3	10.5
0.7	0.7	11.1
0.8	1.2	11.1
0.9	1.1	10.9
1.0	7.0	16.8

Number of ants in the population

The pheromone updating is only carried out once each cycle of m ants have traversed the discretised parameter space. To investigate the effect that the number of ants has on the model's performance, a range of values were tested (Table 4.55). Using the model's generalisation ability on the test set, it was found that the MMAS was not very sensitive to the number of ants. However, as the number of ants increased, the run times required to train the networks increased linearly. Due to the large training times

required when using a large number of ants, a population size of 80 ants was selected for the final model. Using 80 ants required approximately one quarter of the time required when 300 ants were used but was still able to produce the same test set RMSE.

Table 4.55 Run Times and Training and Testing Set Forecasting Errors For Different Population Sizes

Number of Ants	Run Time (s)	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
20	788	1.3	10.5
25	1120	1.0	10.3
30	1387	1.4	10.4
40	1583	1.1	10.2
50	2154	0.9	10.6
70	2744	0.9	10.2
80	3220	1.2	10.1
100	4876	0.9	10.5
150	6191	1.2	10.1
200	7842	1.1	10.1
300	12742	1.0	10.1

MMAS-GRNN

The values of each parameter determined from the sensitivity analyses are given in Table 4.56. These values were used in the MMAS, which was then used to train the multiple-sigma GRNN model. The MMAS was run for 1000 ant cycles, which is equivalent to 1000 generations in a GA, so that a fair comparison could be made with the multiple-sigma GRNN trained using the floating point GA (Section 4.9.2).

Table 4.56 MMAS Parameters Determined Using Sensitivity Analyses

Parameter	Description	Value
b	Number of strata	100
τ_{min}	The minimum pheromone trail	20
ρ	Pheromone persistence coefficient	0.6
m	Number of ants in population	80

The results obtained when using the MMAS to train the multiple-sigma GRNN are given in Table 4.57. Also shown is a comparison with the results obtained when using the floating point GA. The MMAS provided a small improvement in the test set generalisation ability when compared to the GA, however, the small difference is

unlikely to be significant. As indicated by the run time to convergence in Table 4.57, the MMAS had the advantage that it was able to converge on a near-optimal set of sigma weights in a much shorter time than that required by the GA. This was because MMAS required considerably fewer iterations to reach convergence (Figure 4.97).

Table 4.57 Comparison of the Training and Testing Set Forecasting Errors and Training Times for the Multiple-Sigma GRNN Trained Using MMAS and the GA

Training Method	Total Run Time (s)	Run Time to Convergence (s)	Training Set RMSE (EC units)	Testing Set RMSE (EC units)
MMAS	31786	1589	1.0	10.1
GA	7116	3345	0.6	10.5

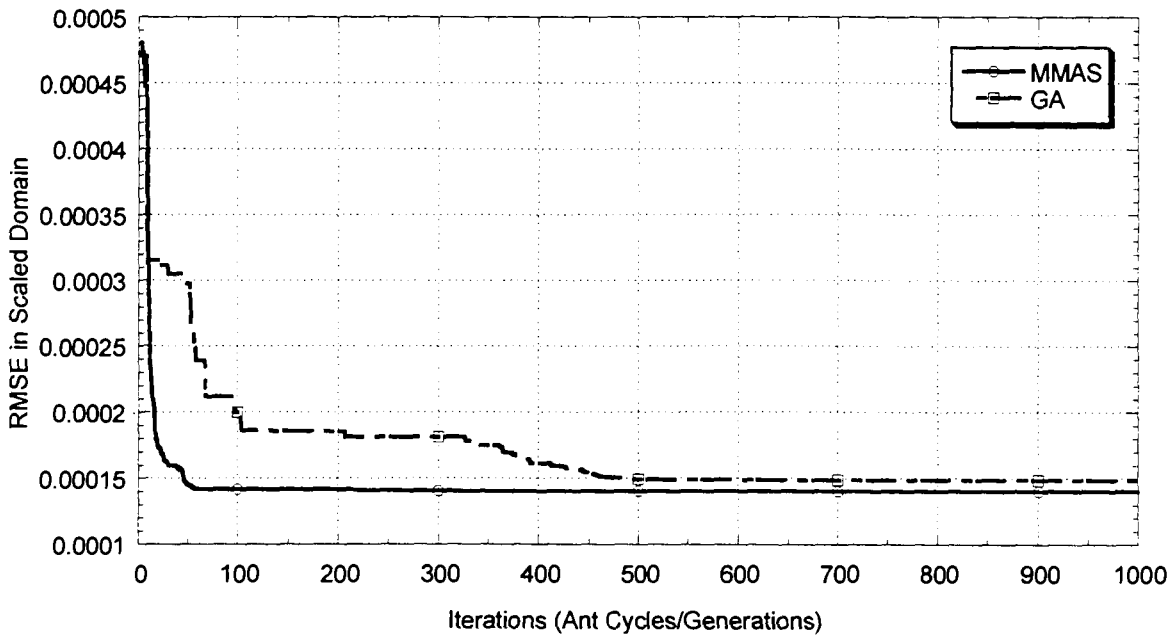

Figure 4.97 Comparison of the Number of Iterations Required by MMAS and the GA to Find the Best Set of Sigma Weights

4.11 Model Deployment

The optimised backpropagation MLP and GRNN models were used to investigate different retraining regimes that could be utilised once the models were deployed in an operational environment. The retraining scenarios investigated included: no retraining; retraining after each sample; and, selectively retraining given some information regarding the presence of an uncharacteristic or outlying data sample. The single-sigma GRNN was used for all GRNN experiments, since it was able to achieve comparable generalisation performance with the multiple-sigma GRNN but required only a fraction of the training time.

To evaluate each retraining strategy, a real-time forecasting simulation was conducted using the second validation set data. After the appropriate lags had been taken into account for each input variable, this data set consisted of 2028 samples spanning the range 29-08-1992 to 18-03-1998.

To detect uncharacteristic data, two new methods were developed in this research, including the hybrid SOM-MLP model and the GRNN outlier detector, both of which are discussed in Section 3.8.

4.11.1 Backpropagation MLP

The first scenario investigated was the effect of not retraining the MLP model. The MLP model was developed using the data from the period 01-12-1986 to 30-6-1992 and the train, test and validation AAPEs were 2.3%, 3.0% and 3.0%, respectively. These results were obtained using the best MLP from Section 4.10.2.1, i.e. the MLP trained using the EDBD algorithm with the minimum test set AAPE (Table 4.48). Figure 4.98 shows the forecasts obtained (without retraining the MLP model) for the real-time forecasting test period of August 1992 to March 1998. It can be seen that the model performs poorly at predicting the magnitude and duration of the major salinity peak that occurred in the third year at around day 900. In addition, the model also significantly underestimates the peak that occurred in the sixth year after day 1825. The poor model performance on these peaks is reflected by the high forecasting errors obtained for the forecasting test period (Table 4.58). In Section 4.7.2.1, it was identified that these salinity peaks corresponded to unseasonal low flow events. A Fourier series was fitted to the mean monthly salinity at Murray Bridge for the model development period (i.e. 01-12-1986 to 30-6-1992), which is shown in Figure 4.98 for the real-time forecasting test period. It is evident that the high salinity/low flow events that occurred in the third and sixth years are unseasonal and unlike any of the data used

in calibrating the MLP model. This has been shown in Section 4.6.2, where a SOM was used to diagnose that these data were outside of the training domain. Consequently, the model was unable to match these two large peaks and this resulted in large forecasting errors.

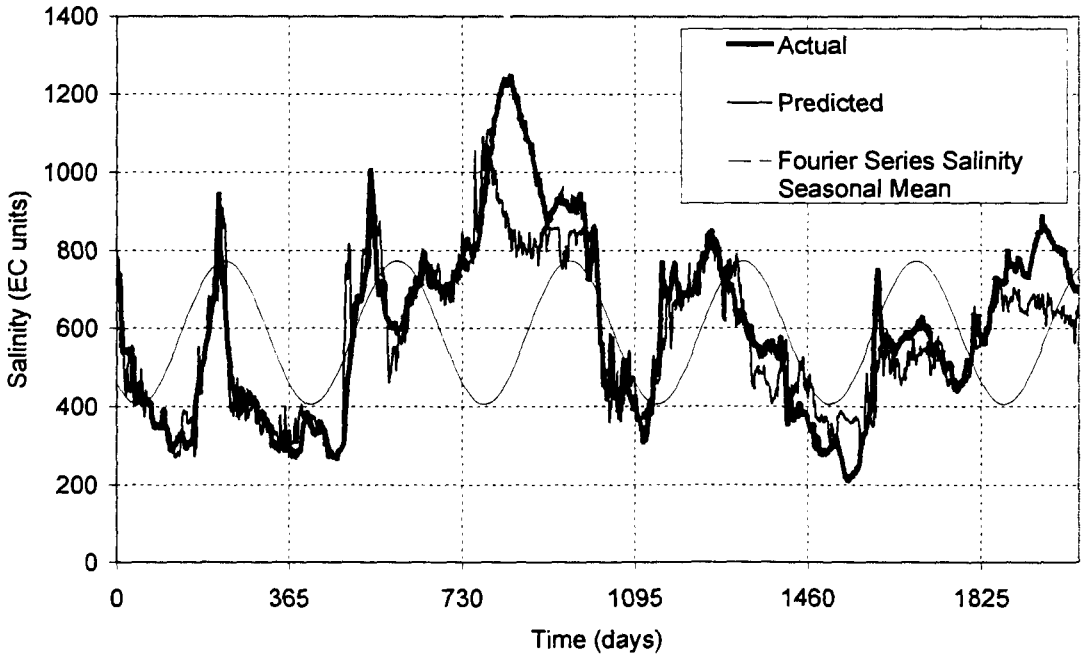

Figure 4.98 Results of Not Retraining the MLP Model for the Real-Time Forecasting Test Period (August 1992 to March 1998). The Fourier Series Seasonal Mean is also shown for the Salinity at Murray Bridge.

Table 4.58 Real-Time Forecasting Test Period Results for the Three MLP and GRNN Retraining Strategies

Model Type	Retraining Strategy	Real-Time Forecasting Test Period (Second Validation Set)	
		AAPE (%)	RMSE (EC units)
MLP	No Retraining	12.3	102.8
	Always Retrain	11.3	91.1
	Selective Retraining (Hybrid SOM-MLP)	11.3	86.4
GRNN	No Retraining	12.4	104.0
	Always Retrain	9.5	74.2
	Selective Retraining (Outlier Detector)	10.3	80.3

The second scenario investigated was the effect of retraining the MLP network after each new sample is obtained. It must be noted that to simulate an on-line operational

model, 14 days must elapse until a relevant output can be obtained before retraining of the model can commence. The results of this retraining scenario for the real-time forecasting test period are shown in Figure 4.99. As expected, this retraining scenario provided better predictions for the uncharacteristic salinity peaks in the third and sixth years and this is reflected by lower forecasting errors (Table 4.58). However, both of these uncharacteristic peaks were still underpredicted by the model.

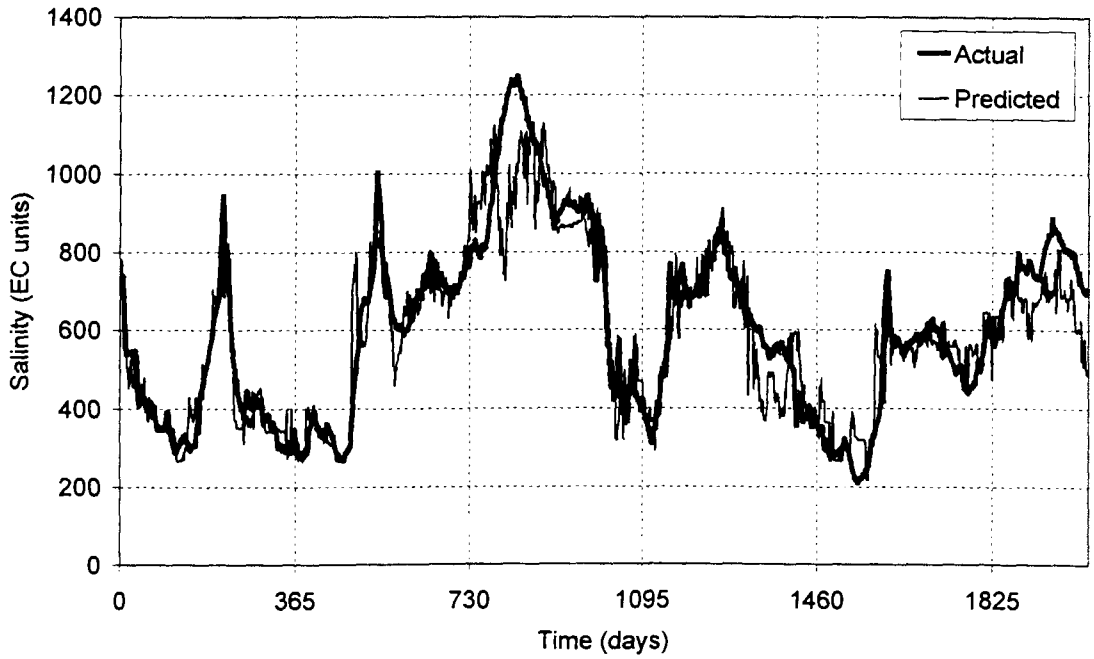

Figure 4.99 Results of Retraining the MLP Model After Each Sample for the Real-Time Forecasting Test Period (August 1992 to March 1998)

The third scenario investigated involved using the hybrid SOM-MLP model to selectively retrain based on the identification of uncharacteristic samples. The SOM-MLP model's results for the real-time forecasting test period are shown in Figure 4.100. The hybrid model still underestimated the major peak in the third year, but for this peak, the hybrid model improved upon the performance of the model that was retrained after every sample (Figure 4.99).

In total, there were 457 warnings issued in the forecasting test period, which indicated 22.5% of the patterns in this period were diagnosed as uncharacteristic. Despite only retraining 22.5% of the time, the SOM-MLP model was able to produce an AAPE of 11.3%, which equaled the AAPE obtained by the model that retrained 100% of the time. In addition, the results from the SOM-MLP represent an 8.1% improvement in the AAPE when compared with the no retraining scenario, thereby highlighting the advantage of selectively retraining the model.

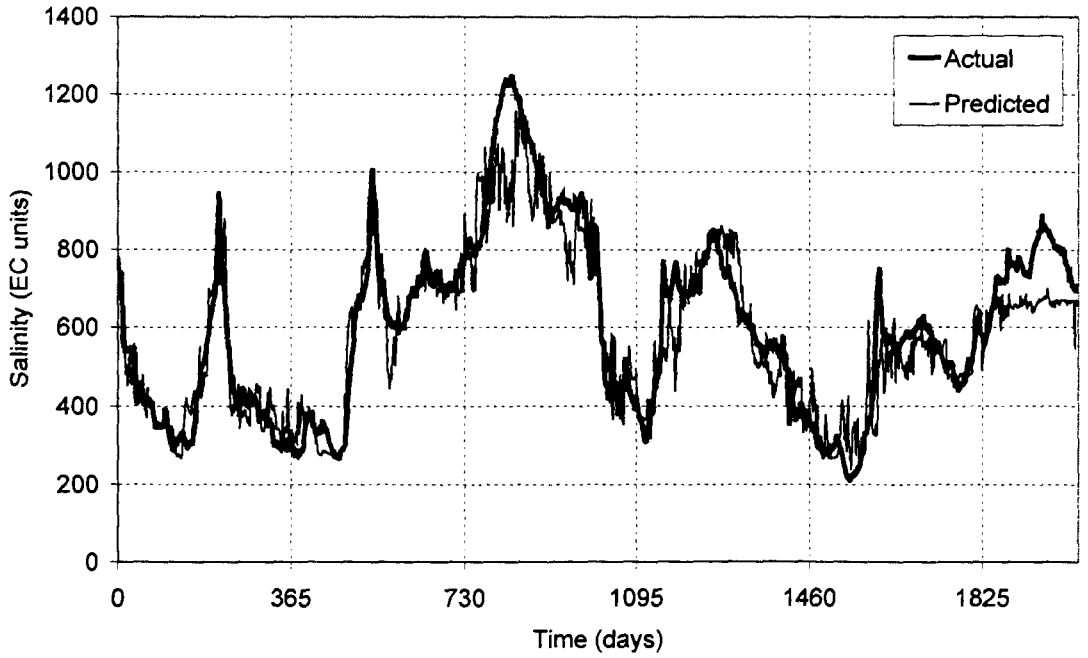

Figure 4.100 Results of Retraining the MLP Model Using the Hybrid SOM-MLP Model for the Real-Time Forecasting Test Period (August 1992 to March 1998)

4.11.2 GRNN

The effect of not retraining the GRNN model for the real-time forecasting test period was investigated and the results are shown in Figure 4.101. The results for the GRNN model were very similar to the results obtained by the MLP model with no retraining (Figure 4.98). The major regions of poor performance were the salinity peaks in the third and sixth years, with the model significantly underestimating both of these peaks. As discussed previously, both of these peaks correspond to uncharacteristic low flow events and it is expected that the model would perform poorly in these regions.

The effect of retraining the GRNN model after the presentation of each sample was also investigated. Once again, 14 days must elapse before updating the model since the salinity value at Murray Bridge 14 days in advance is needed to retrain the model. The results of continuously retraining the GRNN model are shown in Figure 4.102. It is apparent that the forecasts produced by this model represent a significant improvement over the GRNN that was not retrained. However, visual inspection of Figure 4.102 reveals that the forecast tends to lag the observed data by approximately 14 days. This delay time is unavoidable since the forecast horizon must elapse before retraining can commence. To investigate the model's performance with no delay in the model updating, the model was retrained after each sample without waiting the 14 days. It is

important to note that this does not represent a real-world simulation but was conducted to determine if the lag time of 14 days was largely responsible for the majority of the forecasting error evident in Figure 4.102. The results obtained from this investigation are shown in Figure 4.103. It can be seen that there was a dramatic reduction in the forecasting error and the AAPE dropped from 9.5%, when waiting the 14 days, to 0.7%, when retraining instantaneously. This provides evidence that the major source of the error in the forecasts obtained in Figure 4.102 was unavoidable and was due to the delay time required before retraining the model. It is also important to note that model performance is a function of the forecasting period and the model would perform better for shorter forecasting periods.

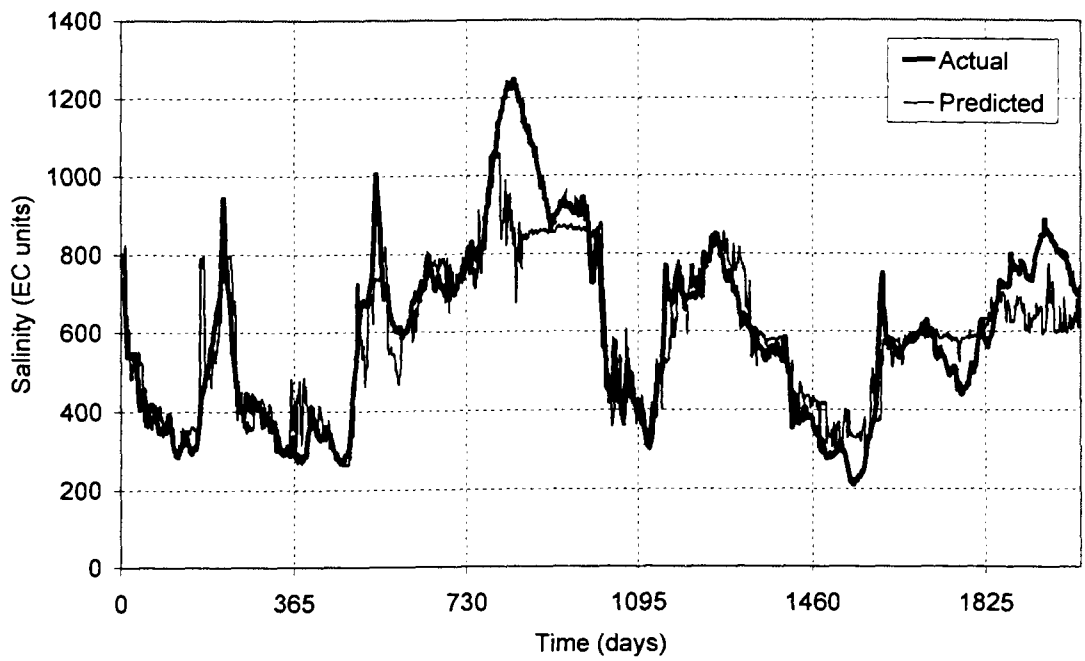

Figure 4.101 Results of Not Retraining the GRNN Model for the Real-Time Forecasting Test Period (August 1992 to March 1998)

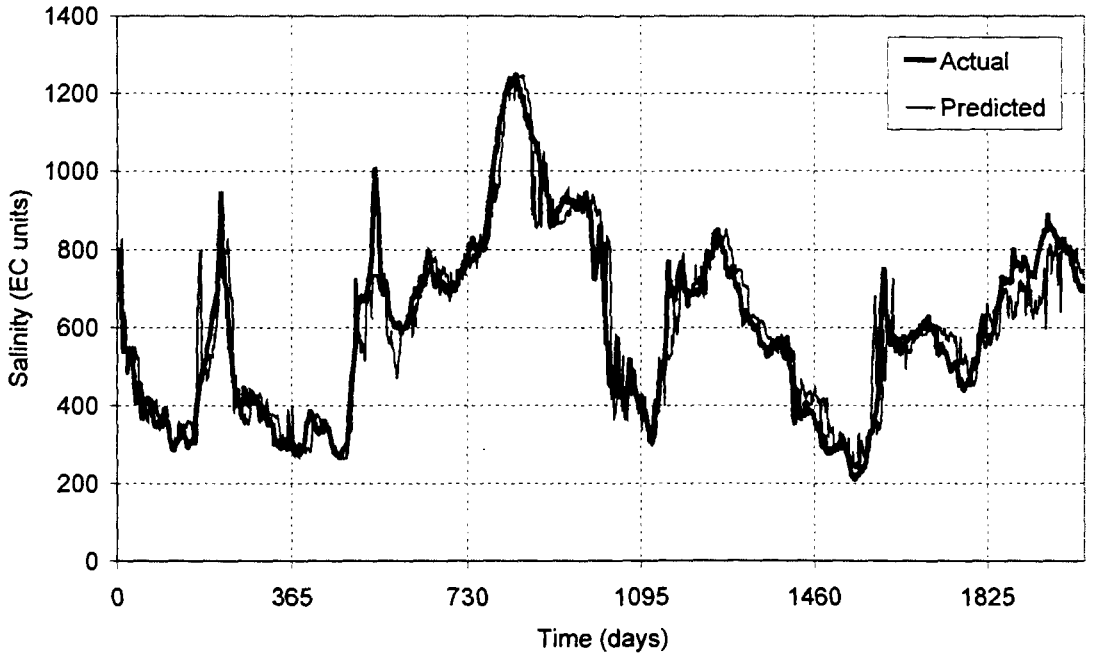

Figure 4.102 Results of Retraining the GRNN Model After Each Sample for the Real-Time Forecasting Test Period (August 1992 to March 1998)

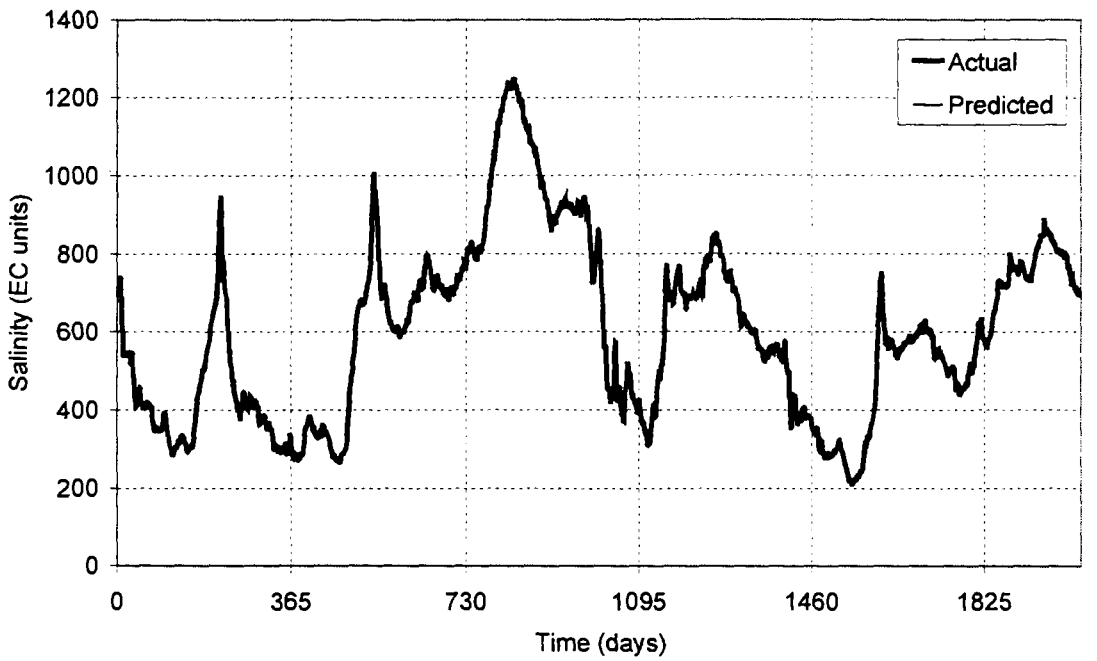

Figure 4.103 Results of Retraining the GRNN Model After Each Sample Without Waiting 14 Days to Update - Real-Time Forecasting Test Period (August 1992 to March 1998)

The final scenario investigated was selective retraining once the GRNN outlier detection method identified an uncharacteristic sample. The results of the selective retraining along with the regions identified as uncharacteristic are shown in Figure 4.104. The results obtained were similar but not quite as good as those obtained when retraining after each sample Figure 4.102. The majority of the uncharacteristic samples were identified early in the first three years of the forecasting test period. This is expected since the model becomes increasingly robust as it incorporates the uncharacteristic data into its training set. Therefore, the likelihood of encountering new uncharacteristic data decreases with time. In total, 469 warnings were issued, indicating that 23.1% of the patterns in the test period were diagnosed as uncharacteristic. Even though the model only retrained 23.1% of the time, it was still able to obtain an AAPE of 10.3%. This was not much larger than the AAPE obtained by the model that retrained 100% of the time i.e. 9.5%. In addition, the results from the GRNN outlier detection method represent a 16.9% improvement in the AAPE when compared with the no retraining scenario.

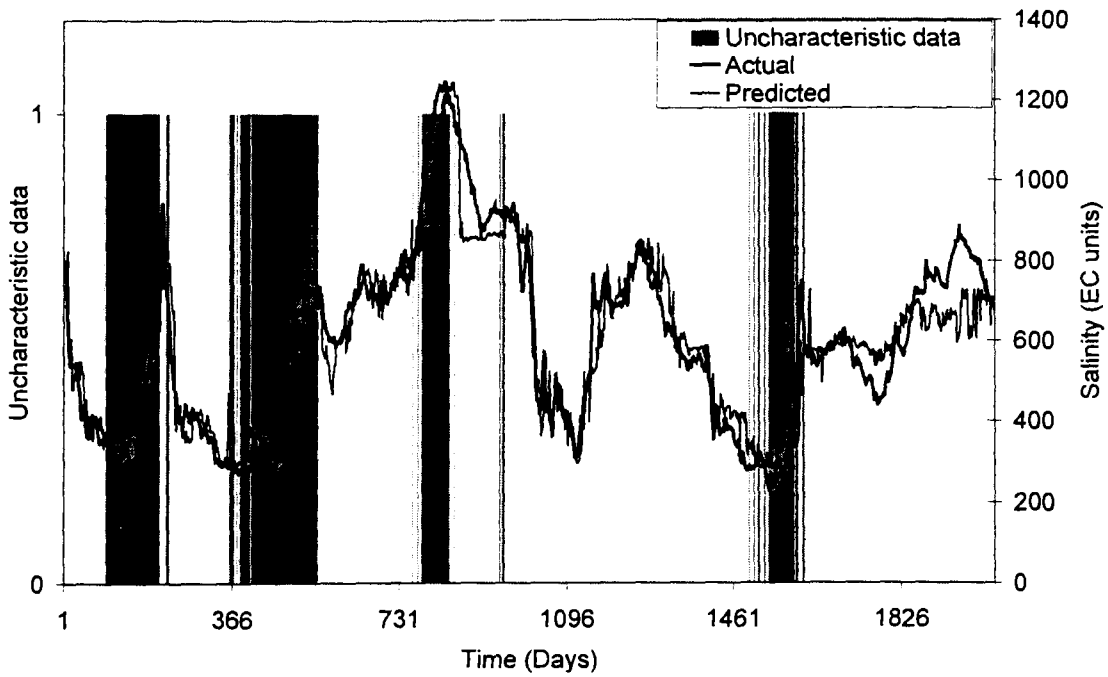

Figure 4.104 Results of Retraining the GRNN Model Using the GRNN Outlier Detection Method for the Real-Time Forecasting Test Period (August 1992 to March 1998)

A comparison of the forecasting errors obtained by each retraining scenario for both the MLP and GRNN models is given in Table 4.58. The best result obtained by the MLP was achieved when using the hybrid SOM-MLP model to selectively identify when retraining was necessary. This model produced an AAPE and RMSE for the

forecasting test period of 11.3% and 86.4 EC units, respectively. However, this result was inferior than the best result obtained by the GRNN, which was achieved when the GRNN model was retrained after each sample. This GRNN model produced an AAPE and RMSE for the forecasting test period of 9.5% and 74.2 EC units, respectively. In addition, the GRNN was much faster to train than the MLP, and therefore, was not computationally intensive to retrain after each sample. The forecast produced by this GRNN model was only limited by the forecast lead-time, which must elapse before retraining is conducted.

4.12 Summary and Conclusions

In this chapter, the ANN methodology outlined in Chapter 3 was applied to the forecasting of salinity in the River Murray at Murray Bridge, 14 days in advance. The steps in the ANN model development process that were considered in detail included: testing the data for evidence of nonlinearity; data division; data transformation; determination of model inputs, choice of network type and architecture; training (optimisation); and model deployment. For each step, numerical experiments were conducted to illustrate the proposed methods, to aid in the formulation of guidelines, and to develop optimal salinity forecasting models for this case study. The findings obtained for each of the steps investigated are summarised below.

4.12.1 Testing for Nonlinearity

Two nonlinearity tests (i.e. BDS test and Kaboudan's FCS test) were applied to the salinity time series at Murray Bridge. Both tests determined that the time series was significantly nonlinear. Kaboudan's FCS test provided additional information, and indicated that the time series contained a strongly linear component and a nonlinear component with a medium level of noise. This information justified the use of ANNs in modelling these data.

4.12.2 Data Division

In this research, two techniques for the optimal division of data for ANN models have been investigated and compared with the results obtained using the method most commonly employed in the literature, i.e. an arbitrary division. When tested on the second validation set, the GA and SOM data division techniques resulted in a reduction in RMSE of 24.2% and 9.9%, respectively, over the conventional data division method. However, it must be pointed out that a range of errors is possible when the conventional method is used because the data division is arbitrary (i.e. the error could have been larger or smaller and is likely to change each time the arbitrary division is performed). The main advantage of the GA and the SOM data division methods is the consistency in forecasting results that is obtained. This is because these methods ensure that the training, testing and validation sets are representative of the same population. Given a suitable amount of data, these techniques enable the development of a training set that extends to the edges of the modelling domain in all dimensions. In so doing, the ANN is able to find a generalised solution to the problem being investigated. It is important to note that overfitting only occurs when the training set is not totally representative of the population (Masters, 1993). Hence, the GA and SOM division techniques will

reduce the likelihood of overfitting the data in the training set. It was found that the ANN models developed using the SOM and GA data division techniques outperformed the ANN models developed using the conventional data division method. However, as mentioned above, the real advantage was the consistency that was obtained when using these data division methods.

When developing a training set, it is also important that the training samples are evenly distributed throughout the modelling domain, otherwise training will focus on densely clustered regions and neglect the sparsely represented regions. The SOM technique achieves an evenly distributed training set by firstly dividing all of the data into a number of clusters. By then sampling an equal number of records from each cluster, the training, testing and validation set records are evenly distributed throughout the problem domain. Another advantage is that the SOM division technique employs the minimum number of samples for achieving this objective and consequently, avoids the problem of selecting what proportion of the data should be used in the training, testing and validation subsets. While the GA data division technique does not automatically determine the proportion of data to use for each subset, it does have the advantage of ensuring that the extreme values are contained within the training set. This was achieved through the addition of penalty constraints.

Finally, when tested on the second validation set, the models developed using the GA- and SOM-based approaches outperformed the model developed using an arbitrary division of the data. However, this performance was masked by the fact that the second validation set contained new regions of data that were not representative of the data used in the training, testing and validation sets. This highlights the importance of periodic retraining of the model.

The two data division methods presented both proved to be very capable of dividing the data into three statistically similar and representative subsets. It is important to note that these techniques provide representative data sets that best approximate the modelling domain based on the available data. There can be no guarantee that the model will perform well when faced with new data. However, using the SOM analysis, it was shown in this research that it is possible to diagnose regions of poor performance resulting from uncharacteristic data (i.e. data that are unlike any of the data that have been used in developing the model).

4.12.3 Data Transformation

Six data transformation techniques for use with ANN models were investigated. As ANN models are data-driven, an "optimal" transformation can, if at all, only be defined relative to the specific application for which the ANN model is being developed. However, some general findings have come out of the data transformation study and it is hoped that in time, results from similar studies will begin to define when such transformations may or may not be useful in water resources applications of ANN models. Taking a logarithmic transformation of the data, removing the seasonality from the data and transforming the inputs and outputs to normality were all found to give significantly larger forecasting errors than a simple linear transformation of the data. It is believed that these transformations distorted the original relationships between variables in a way that was not beneficial to ANN learning. The models developed using data transformed by histogram equalization and kernel transformation were found to perform well on data within the training domain but were not robust when applied to new data patterns. The model developed using a linear transformation gave the best results overall as this model proved the most robust on new data patterns while still giving a relatively low RMSE on the training, testing and validation data sets. These findings reinforce the popular belief that when using ANNs (particularly feedforward MLPs, which fit the data without assuming any functional form), one does not have to say that the data should be distributed in any particular way for the approach to be used. Furthermore, an analysis of the residuals produced by the linear transformation ANN model showed that the hypothesis of normal (or at least symmetrical) random shocks was valid. Consequently, the use of the MSE as the objective function to calibrate this model is justified as it maximises the likelihood of the model under this assumption.

4.12.4 Determination of Model Inputs

Three input determination techniques were developed and applied to the salinity case study. The first method was a modified version of the stepwise PMI algorithm. This method is a supervised, *model-free* approach that is able to account for input redundancy. The second approach utilised PCA as the unsupervised technique to preprocess the input data. A hybrid GAGRNN model was then used as the supervised technique to further refine the input set based on forecasting ability. The third method was a variation of the second approach, and utilised a SOM as the unsupervised technique and a GAGRNN as the supervised input determination technique. The input subset obtained from each method was used to develop ANN models and the

performance of these models was compared with models developed using the inputs obtained in two previous studies: Maier and Dandy (1996b) and Maier and Dandy (1997a). Based on the validation set results, it was evident that the stepwise PMI algorithm and the SOM-GAGRNN provided the most effective means for selecting appropriate model inputs. The models developed using both of these methods significantly outperformed the model obtained using the PCA-GAGRNN approach and the models developed using the inputs obtained from the two previous studies. In addition, the models developed using the stepwise PMI algorithm and the SOM-GAGRNN were more parsimonious than the models developed using the inputs obtained from the two previous studies.

The models developed using each method were also applied to the second validation set. It was found that the model developed using the inputs identified by the stepwise PMI algorithm was the most robust in the presence of uncharacteristic data. The model developed using the PCA-GAGRNN approach performed poorly on the second validation data set. One possible reason for this may arise from the fact that the PCs are each a weighted average of all 960 inputs, and as such, may contain a large degree of irrelevant information. Even after the supervised procedure is applied (i.e. GAGRNN) to select the most important PCs, it is likely that these PCs will still contain irrelevant information. Another cause of poor performance may be the fact that the eigenvectors obtained using the calibration data are used to transform the second validation set data into the respective PCs. Since the seasonal variation is known to change in the second validation data set, this may have contributed to the poor performance that resulted from this model. This could be alleviated by periodically re-performing the PCA and retraining the model.

In an attempt to improve the performance of the model developed using the inputs identified by the PCA-GAGRNN model, the PCA was performed on all 960 inputs rather than in a two step procedure (i.e. univariate PCA followed by multivariate PCA). This approach improved the model's performance on the training, testing and validation sets, however, the model still performed poorly on the second validation set data. This was most likely due to the same factors discussed above.

It is well known that extraneous inputs increase the likelihood of local minima in the error surface and require more training data to efficiently optimise the connection weights. Therefore, when no *a priori* knowledge is available to suggest possible inputs, and for complex problems, where the number of potential inputs is large, it is a great advantage to use a method capable of selecting the significant inputs. When applied to the salinity data set, the stepwise PMI algorithm was able to reduce 960 candidate

inputs to just 13 significant model inputs and produced the most parsimonious model of all the methods considered. Despite requiring only 13 inputs, this model also exhibited the best generalisation ability. The stepwise PMI method was shown to have the added advantage that the magnitude of the PMI scores could be used to obtain valuable information about the system under investigation. The PMI score obtained for each input was found to provide similar information, in relation to the order of importance of each input, as would be gained from a sensitivity analysis.

4.12.5 Choice of Network Type and Architecture

Two types of feedforward ANNs were investigated in this research, namely the EBMLP and the GRNN. The EBMLP is a type of evolutionary ANN because it uses a GA to optimise the network architecture. The GRNN architecture is fixed, however, two variations were considered. These included: clustering the training data using a SOM to reduce the number of pattern layer PEs; and, using a separate sigma weight for each input in the network (multiple-sigma GRNN).

The EBMLP was found to be sensitive to the type of refill strategy used during the GA optimisation. The best models were obtained when cloning was used to refill the population. This refill strategy resulted in a decrease in the diversity of the networks obtained but allowed for faster convergence to a near-optimal network configuration. Increasing the population size did not improve the results.

The best network found used two hidden layers and used the hyperbolic tangent transfer function in the first hidden layer, the sigmoid transfer function in the second hidden layer and the linear transfer function in the output layer. The results indicated that there were many networks with different configurations that were capable of providing similar performance, and as such, there can be no guarantee that the true global minimum was found.

The single-sigma GRNN's error surface was shown to have a well-defined global minimum and therefore, was readily amenable to optimisation by the inverse Hessian method. When the single-sigma GRNN's pattern layer PEs were reduced by clustering the training data using a SOM, the GRNN's performance also decreased. However, increasing the size of the Kohonen layer, and consequently, the number of clusters, resulted in improved model performance as measured using the training and testing set RMSEs. The best result was obtained when there was no clustering i.e. a pattern layer PE was used for each sample in the training set.

Using separate sigma weights for each input in the GRNN only marginally improved the model's forecasting ability. However, to determine an optimal set of sigma weights, the multiple-sigma GRNN needed to be trained by the floating point GA. This training procedure required approximately 2 hours run time, as opposed to the single-sigma GRNN, which was trained using the inverse Hessian method and required only 12 seconds to train.

The independent validation set was used to compare the performance obtained by the single-sigma GRNN and the best network obtained by the EBMLP. Using this data set, the GRNN model significantly outperformed the EBMLP model, and resulted in a 53.0% reduction in the validation set RMSE. When both models were applied to the second validation set, the converse was true. The EBMLP was found to be slightly more robust in the presence of uncharacteristic data. For the salinity case study, it can be concluded that the GRNN provides superior performance when the validation data are known to have the same statistics as the calibration data, whereas the EBMLP is the more reliable option if uncharacteristic patterns are likely to be encountered.

4.12.6 Training (Optimisation)

Ten performance measures commonly used in hydrological modelling were investigated for use with ANN salinity forecasting models. The procedure formulated by Diskin and Simon (1977) for the determination of a suitable performance measure was applied to the MLP and GRNN models. Using the salinity case study, the most suitable performance measure for the MLP model was found to be the AAPE. For the GRNN model, the RMSE and the CE performance measures were both found to be the most suitable.

Six training methods (four first-order and two second-order methods) were evaluated for optimising the connection weights in a feedforward MLP. The first-order methods investigated were: the generalized delta rule; the normalized cumulative delta (NCD) rule; the delta-bar-delta (DBD) algorithm; and, the extended delta-bar-delta (EDBD) algorithm. The QuickProp (QProp) algorithm and the MaxProp (MProp) algorithm were the second-order methods investigated. Initially, an epoch size of 16 was used. It was found that there was a large degree of difference in the results obtained by each of the first-order methods. The EDBD algorithm provided the best results, as measured by the average error calculated using 30 different weight initialisations. In addition, the EDBD provided the best model overall, with an AAPE of 3.0%.

The generalisation ability of the second-order methods was found to be inferior to that of the first-order methods. The most likely cause of this is the inability of second-order methods to escape local minima. Increasing the epoch size to 100 resulted in a decrease in generalisation ability for each of the first- and second-order methods that utilised cumulative updating of weights i.e. NCD, DBD, EDBD, QProp and MProp.

An alternative optimisation algorithm, known as ant colony optimisation, was applied to the training of a multiple-sigma GRNN. The objective was to determine if a significant improvement in generalisation ability could be achieved by using an alternative to the GA. The *Max-Min* Ant System (MMAS) was used in this research, and requires the determination of a number of internal parameters. To determine suitable values for each of these, sensitivity analyses were conducted. The following general findings for the salinity case study were deduced from these analyses:

- For a given number of ants, increasing the number of strata that each sigma weight is discretised into caused a reduction in generalisation ability. When using a large number of strata, increasing the number of ants tended to improve the generalisation ability.
- The MMAS algorithm was insensitive to values of the minimum pheromone trail less than or equal to 100.
- The best performance was achieved using intermediate values of the pheromone persistence coefficient (i.e. $\rho = 0.4 - 0.6$).
- Increasing the number of ants in the population did not cause a significant improvement in the network's performance, but resulted in a linear increase in the training times required.

Training the GRNN using the MMAS resulted in a small improvement over the model trained using the floating point GA, however, this improvement is unlikely to be significant. Therefore, it can be concluded, with some degree of certainty, that the inability of the multiple-sigma GRNN to significantly outperform the single-sigma GRNN is not a function of the optimisation algorithm used. The MMAS did have the advantage that it required much fewer iterations than the GA, to converge on the best solution.

4.12.7 Model Deployment

Two new retraining methodologies for real-time forecasting models were introduced, including: a method based on a hybrid SOM-MLP model; and, a method for detecting outlying data patterns based on a statistical test of the GRNN's *B* summation unit. The

performance of the SOM-MLP and GRNN models were compared with two alternative retraining strategies, involving no retraining and continuous retraining, respectively. All three retraining methods were trialled on a real-time forecasting period from August 1992 to March 1998 (i.e. second validation set).

For the MLP model, the results indicate that the SOM-MLP provided an effective means of identifying when retraining was required. The front-end SOM component of the hybrid model was able to diagnose the data that were outside of the training domain and retrained 22.5% of the time during the real-time forecasting test period. This retraining strategy resulted in a significant improvement over the scenario in which the MLP model was not retrained. In addition, selectively retraining based on the presence of uncharacteristic data was found to produce an AAPE equal to that obtained when the MLP was continuously retrained after each new sample.

For the GRNN model, the results indicated that the outlier detector was also successful in identifying when retraining was required. 23.1% of the data patterns in the forecasting test period were diagnosed as uncharacteristic. Retraining the model when an outlying pattern was detected resulted in a large improvement over the GRNN in which no retraining was performed. Despite only retraining 23.1% of the time, the GRNN outlier detector was able to obtain a similar AAPE to that obtained by the GRNN, which was retrained 100% of the time. It was also discovered that the majority of the error obtained by the retrained GRNN, was caused by the delay time (i.e. 14 days). This delay time must elapse before retraining can commence in order to simulate the time that passes before the sample is collected in the real-world. This source of error was unavoidable and significantly inhibited the model's performance.

Chapter 5

Case Study II: Cyanobacteria in the River Murray

5.1 Introduction

Cyanobacteria (blue-green algae) are a major water quality problem in the Murray-Darling system and in other parts of the world. Cyanobacteria are photosynthetic bacteria that, under certain conditions, can multiply to large numbers and dominate otherwise healthy phytoplankton communities. Such excessive growth incidences are of major public concern because cyanobacteria are unsightly and capable of producing a variety of toxins and undesirable tastes and odours. The toxins produced by cyanobacteria can cause severe gastrointestinal illness and, in extreme cases, death in humans and animals. The presence of large numbers of cyanobacteria also significantly increases the cost of water treatment. The algae increase the suspended solids content and turbidity of the water, thereby reducing the efficiency of disinfection and blocking filters. Consequently, to prevent blooms of cyanobacteria, it is necessary to have an understanding of the complex relationship between cyanobacterial growth events and

environmental conditions, such as flow, irradiance, temperature, nutrients and competitive interactions with other taxa.

In Chapter 3, an ANN modelling methodology was formulated and outlined. In Chapter 4, this methodology was applied to the development of ANN models for forecasting salinity in the River Murray, 14 days in advance. The results of the numerical experiments conducted on the salinity case study showed that the methodology was able to lead to the development of ANN models capable of providing useful salinity forecasts at Murray Bridge. Using the findings obtained from the salinity case study, it was considered important to test the methodology on another case study to verify its suitability for a different problem. In this chapter, the ANN methodology outlined in Chapter 3 was applied to the development of a model capable of forecasting the incidence of a species group of the cyanobacterium *Anabaena spp.*, in the River Murray at Morgan. *Anabaena* is the most common genus of nuisance cyanobacteria in the lower reaches of the River Murray and also poses the most significant concern from a water supply perspective. This is because large blooms of *Anabaena* are capable of producing algal toxins and taste and odour problems in drinking water supplies. The removal of toxins and odour-producing compounds requires advanced water treatment processes such as powdered activated carbon (PAC) or oxidation by ozone or chlorine (Rositano and Nicholson, 1994). The ability to forecast impending blooms of *Anabaena* using ANN models would provide a significant aid in reducing water treatment costs. It would enable water treatment plant (WTP) operators to respond rapidly to changes in raw water quality and ensure that the use of expensive chemicals such as PAC are kept to a minimum. In addition, the forecasts would enable early warnings to prevent stock from drinking the affected water and recreational users from coming into contact with the toxins.

5.2 Background

Cyanobacteria are oxygenic photoautotrophs that are prokaryotic (i.e. whose cells have no nucleus). They are commonly referred to as blue-green algae due to the presence of blue and red accessory pigments as well as chlorophyll *a* (Maier and Dandy, 1994). Their prominent habitats are limnic and marine environments and their life processes require only water, carbon dioxide, inorganic substances and light (Chorus and Bartram, 1999).

Cyanobacteria were amongst the pioneer organisms of the early earth and were probably the chief primary producers of organic matter (Chorus and Bartram, 1999). Prokaryoate fossils 3,450 million years old have been identified in Warrawoona sedimentary rock in northwestern Australia. It is believed that these photosynthetic microorganisms were probably the first organisms to release elemental oxygen into the earth's primitive atmosphere (Chorus and Bartram, 1999). In so doing, they helped shift the balance of gases in the atmosphere "...imperceptibly but inexorably from the primitive atmosphere to the oxygen-rich atmosphere we know today...creating new opportunities for life" (Suzuki, 1997). In general, planktonic algae are credited with the generation of some 70% of the world's atmospheric oxygen supply and they form the base upon which the aquatic food chain is founded (Reynolds, 1984a).

Cyanobacteria are a naturally occurring component of the freshwater algal community (Maier et al., 1998). With respect to water quality, Sullivan (1990) has divided freshwater algae into the following four groups (Figure 5.1):

- Diatoms
- Cyanobacteria (Blue-green algae)
- Flagellates
- Green algae

Cyanobacteria are a concern because of the severe practical problems for water supplies caused by their abundant growth. Cyanobacteria are capable of producing a range of compounds, including toxins and odours, which have a deleterious effect upon drinking water quality (Burch and Steffensen, 1997). In polluted water systems all over the world, the development of toxic strains of cyanobacteria has become commonplace. Consequently, cyanobacterial toxins or "cyanotoxins" have become a significant concern for human health (Chorus and Bartram, 1999). In order to ensure safe water supplies in affected areas, it is essential that monitoring for cyanobacteria and their

toxins is conducted in the source waters and the distribution system, and that reliable management procedures are put in place (Steffensen et al., 1999).

When conditions are favourable, cyanobacteria can become dominant and form blooms (National Rivers Authority UK, 1990). A review of the literature has highlighted that there are a number of factors implicated in the development of cyanobacterial blooms.

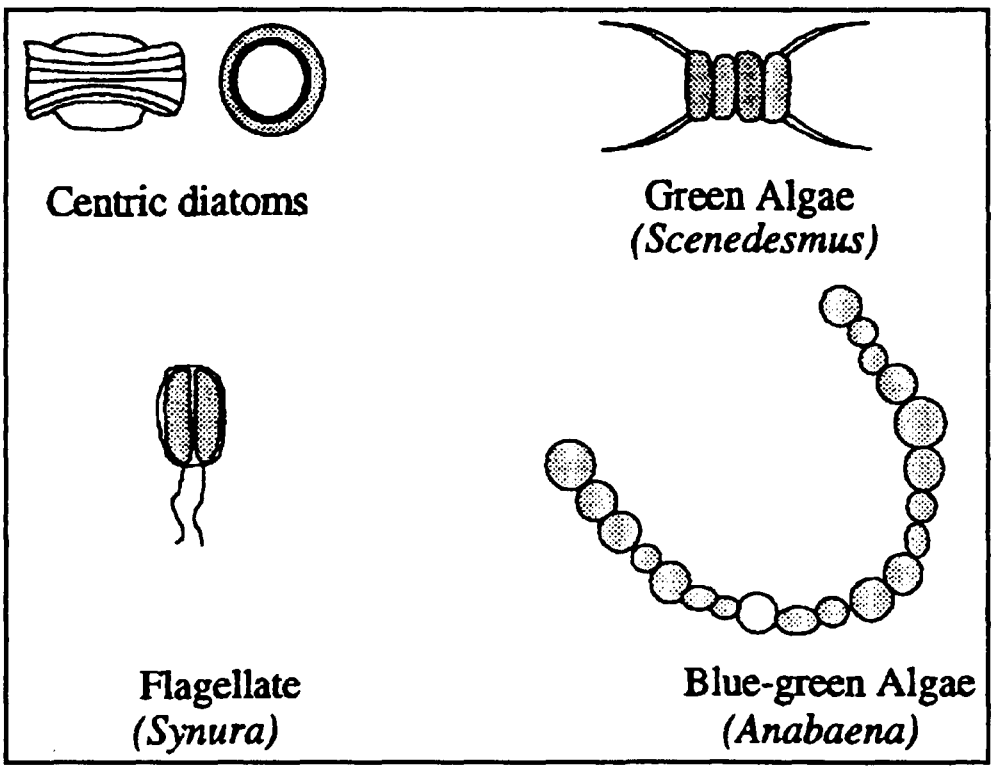

Figure 5.1 Examples of the Four Main Groups of Algae (source: adapted from Sullivan, 1990)

5.2.1 Factors Affecting the Incidence of Cyanobacterial Blooms

The increase in the frequency of blooms of cyanobacteria is largely the result of nutrient enrichment by human activity, although some blooms may be entirely natural in origin (Harris, 1994). Physical processes of water column stratification also play an important role. Algal blooms in rivers result from similar processes to those observed in water storages and reservoirs - the key parameters being nutrient loading and flow rate or residence time (Harris, 1994).

The interaction between the development of blooms and environmental variables is a complex one, however, there are factors that are common to all incidents of excessive blue-green algal growth. In a general sense, the size of the phytoplankton population can be regarded as a function of the import and export of algae, the algal growth rate

and the amount of algae lost through grazing and decomposition (Oliver, 1994). The factors which influence cyanobacterial growth from 'below' such as flow, light, temperature and nutrients can be referred to as 'bottom up' factors (Section 5.2.1.1), and the factors which arise 'higher' in the food chain, such as grazing pressure from zooplankton can be separated into 'top down' factors (Section 5.2.1.2) (Harris, 1994).

The adequate management of a body of water prone to blooms of cyanobacteria requires a good understanding of the response of cyanobacteria to environmental factors. It is also important to recognise that different cyanobacterial taxa will exhibit different behaviour in nature because their ecophysiological properties differ (Chorus and Bartram, 1999).

5.2.1.1 'Bottom Up' Factors

The 'bottom up' factors are factors which influence the cyanobacterial populations from below. These factors not only determine the growth of a population but also determine which genera and species become established and dominant in specific ecosystems. Table 5.1 summarises some of the factors potentially affecting cyanobacterial population ecology and dynamics.

Table 5.1 Physical-Chemical Factors Affecting Cyanobacterial Blooms (source: Paerl, 1996)

Factors	Effects and Impacts
Physical	
Flushing/altered water residence time	Potential removal mechanism for blooms, if flushing exceeds growth rates of bloom taxa
Large scale vertical mixing	Counteracts near surface accumulations of buoyant bloom populations. Forces competition for light and nutrients with more desirable, non-buoyant eukaryotic taxa
Small scale turbulence (shear)	May disrupt blue-green algal filaments, colonies, aggregates and mutualistic associations with other microflora and microfauna
Shading (reduced surface irradiance)	Can alter phytoplankton community composition and can negatively affect blue-green algal surface bloom taxa
Temperature	Generally temperatures in excess of 20°C accompanied by stratification and high nutrient loading can promote blooms

Continued.

Factors	Effects and Impacts
Chemical	
pH modifications	Can alter phytoplankton community composition: low pH (< 6.0) favours eukaryotes, and high pH (> 8.0) favours cyanobacteria
Nutrient (N and P) inputs	Long term (months/years) reductions in both N and P inputs are frequently effective in reducing blue-green algal bloom potentials; low N:P loading ratios (< 20) often caused by excessive P loading can enhance bloom potential
Salinity	Salinity in excess of a few percent (as NaCl) can be an effective barrier to the development and persistence of many nuisance species
Trace metals	Under high N and P loading conditions, restricted availability of Fe may control phytoplankton growth; cyanobacteria are able to compete effectively for low levels of Fe; no convincing evidence for other trace metal limitations

Flow

Residence times

Most cyanobacteria have a slower rate of growth than other planktonic algae (National Rivers Authority UK, 1990). This slow rate of growth requires relatively long hydraulic retention times in order for the population to reach a substantial size. In highly regulated river systems the reduction of flow and extraction of water increases the residence time so that it becomes equal to that of small lakes. This is particularly true for the weir pools that dominate the River Murray between Mildura and the Barrages in South Australia. Knowledge of cyanobacterial growth in reservoirs and storages can therefore be directly applied to growth in weir pools and regulated rivers (Harris, 1994).

Longer residence times allow suspended solids to settle out thereby reducing the turbidity and allowing greater light penetration. Better light penetration in the water column stimulates the benthic growth of cyanobacteria (MDBC, 1993).

Water column stability

The stability of the water column is a function of the turbulence induced by both flow and wind (Maier et al., 1998). Many noxious species of cyanobacteria, including all of the known toxic species in Australia, tend to proliferate in calm stable waters (Steffensen et al., 1999). They predominantly occur in lakes and reservoirs, particularly in the summer months, when the vertical mixing is reduced due to stratification. Stratification in water bodies is caused by density variations in the water column induced by differences in temperature and other water quality parameters. A thermocline, where the temperature gradient is greatest, is established between the warmer epilimnion and the cooler, denser water of the hypolimnion (Fischer et al., 1979).

Although lakes and reservoirs typically play host to noxious species of cyanobacteria, regulated river systems can also provide conditions conducive to cyanobacterial growth. Jones (1994) considered the weir pool conditions stimulating cyanobacterial blooms in the Murrumbidgee River, and found that *Anabaena* was dominant during periods of stratification associated with periods of low flow. Sherman and Webster (1997) also looked at the relationship between flow and cyanobacterial growth in the Murrumbidgee River, in particular, the field experiment focussed on the Maude weir pool. When the water column was well mixed, it was observed that diatoms dominated the phytoplankton community, however, after approximately a fortnight of persistent stratification, the toxic cyanobacterium *Anabaena circinalis* began to dominate.

In the Lower River Murray, studies have shown that wind, rather than flow, is the dominant variable affecting the degree of thermal stratification under low flow conditions (Bormans et al., 1997). Under entitlement flow conditions, the average velocity of the water in the River Murray, South Australia is very low (in general less than 0.1 m/sec) (MDBC, 1993). In the Lower River Murray, persistent stratification is only likely to occur when the velocities are very low (i.e. velocities in the range 0.04-0.06 m/s) combined with low wind speed (<1.2 m/s) (Burch et al., 1998).

The reason some species of cyanobacteria have an advantage over other forms of phytoplankton in stratified conditions is due to the presence of specialised intracellular gas vesicles. These minute, proteinaceous hollow cylinders are stacked together and maintain a gas filled space in the cell which, when considered cumulatively, may occupy 2 - 20% of the volume of the intact cell (National Rivers Authority UK, 1990). The gas vesicles have the effect of lowering the density of the organism below that of the surrounding water, thereby inducing a buoyant effect.

Buoyancy regulation is an ecologically significant mechanism as it allows the cyanobacteria to position themselves within the vertical gradients of physical and chemical factors in the water column. Environmental stimuli (e.g. photic, gravitational, chemical and thermal) are used by the cyanobacteria as clues in determining the optimal position (Chorus and Bartram, 1999).

In times of low turbulence, the cyanobacterial ecostrategists capable of actively regulating their buoyancy float to the surface layers, where light conditions are optimal. Cyanobacteria that are able to maintain themselves above the thermocline can succeed in thermally stratified water columns, while other phytoplankton settle out (Brookes et al., 1998; Burch and Steffensen, 1997).

If wind or turbulence destratifies the water column, the species of cyanobacteria capable of buoyancy regulation are pushed below the water surface and are randomly dispersed throughout the mixed layers. Wind tunnel studies have shown that wind speeds of more than 2 to 3 m/sec are required to break up surface scums of cyanobacteria (MDBC, 1993).

Remobilisation of sediment bound phosphorus

Under low flow conditions, stratification becomes more probable and it is also likely that anoxic conditions will prevail (MDBC, 1993). Anoxic conditions result when the rate of decomposition of organic matter and the total amount of oxygen demand exceeds the diffusion of oxygen into the bottom waters and depletes the stock of oxygen dissolved in the hypolimnion (Harris, 1994). Anoxic conditions near the riverbed have the effect of releasing phosphorus that is usually tied up in the oxidised sediments, thereby increasing the phosphorus available for cyanobacterial growth.

Irradiance

Like other forms of phytoplankton, cyanobacteria contain chlorophyll *a*, which is the major pigment used to harvest light and conduct photosynthesis for growth. However, cyanobacteria can also contain other pigments that allow them to harvest light in the green, yellow and orange part of the spectrum (500-650 nm), wavelengths which are hardly ever used by other phytoplankton species (Chorus and Bartram, 1999). The ability to utilise light intensities over a broad range of the visible spectrum gives cyanobacteria a competitive advantage in low-light environments (National Rivers Authority UK, 1990).

The euphotic zone, with depth (Z_{eu}), is the zone in which photosynthesis can occur. It is defined as the region extending from the surface to the depth at which 1 per cent of the surface light intensity can be detected (Chorus and Bartram, 1999). The mixed upper layer of water in a thermally stratified body of water is termed the epilimnion, with depth (Z_m). This layer of water entrains the planktonic algae or cyanobacteria that are capable of only weak active movement. The euphotic zone can be shallower or deeper than the epilimnion (Figure 2.1). Therefore, the algae capable of only weak active movement can only remain photosynthetically active when circulation maintains them in the euphotic zone (Chorus and Bartram, 1999; Reynolds, 1984b). Consequently, the ratio of euphotic depth to mixed depth, Z_{eu}/Z_m, provides a good description of the light conditions experienced by the algae. Numerous studies have shown the critical importance of this ratio to phytoplankton growth (see Ganf, 1982; Harris et al., 1980; Harris and Piccinin, 1980; Hergenrader and Hammer, 1973; Walmsley, 1978).

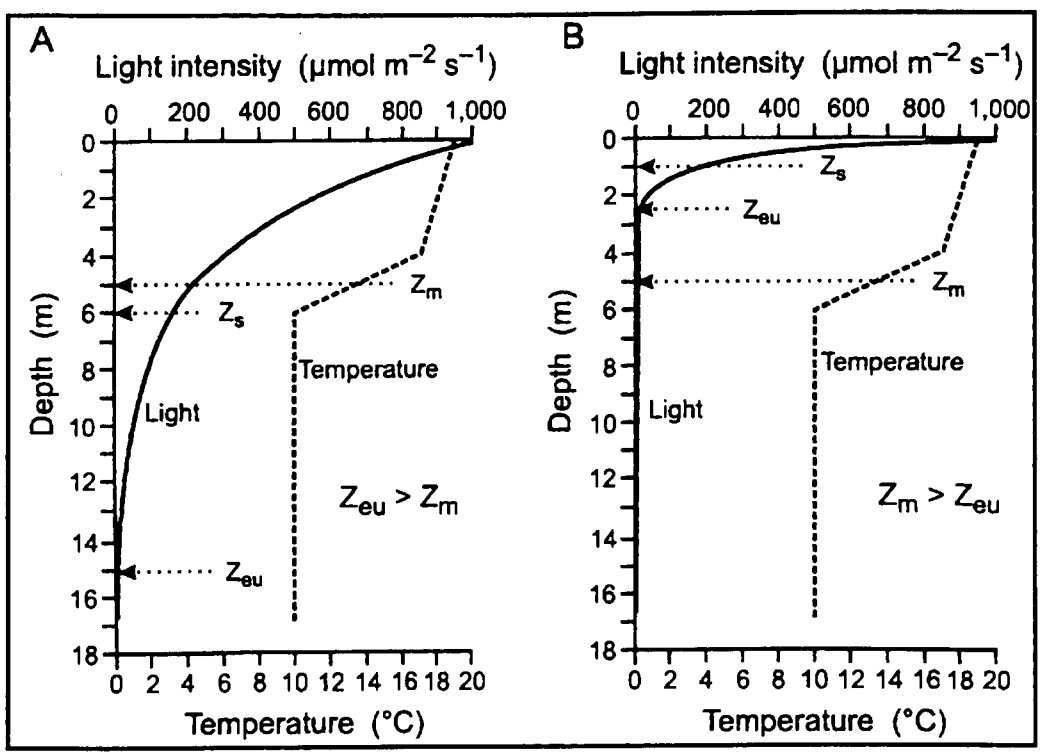

Figure 5.2 The Euphotic Zone (Z_{eu}) Relative to the Epilimnion (Z_m) in Situations With Different Turbidity. A. Euphotic zone is deeper than epilimnion; B. Euphotic zone is not as deep as epilimnion. Secchi depth (Z_s) is included as a rough measure of euphotic depth (Z_{eu}) ($Z_{eu} \cong Z_s \times 2.5$) (source: Chorus and Bartram, 1999)

The irradiance available for cyanobacteria to photosynthesize and grow is a function of the amount of incoming solar radiation and the optical properties of the water column, such as turbidity and colour (Maier et al., 1998).

Solar radiation

During the occurrence of a bloom, the concentration of cells in the water increases, light penetration decreases and the phytoplankton can become self-shading (Fogg and Thake, 1987). When this occurs an upper limit on the phytoplankton cell density is reached and this level can be expressed following Lambert-Beer's basic law of exponential extinction with increasing thickness of the water layer (Chorus and Bartram, 1999; Fogg and Thake, 1987). The law is expressed in Equation 5.1.

$$I_Z = I_o' e^{-Z\varepsilon} \tag{5.1}$$

where I_Z is the intensity at depth Z, and I_o' is the surface intensity. ε is the vertical extinction coefficient, which in turn is the sum of extinction by the water itself and the substances dissolved in it such as humic acids ε_W, the algae suspended in the water ε_a, and other particles suspended in the water ε_p.

Reynolds (1997) has formulated an expression to determine the average amount of light, I^*, which is available for phytoplankton entrained in the vertical mixing of the water body or (under thermal stratification conditions) in the epilimnion. In this expression, I^* is the square root of the product of the intensity at the surface, I_o, and the bottom of the mixed layer of water, I_m as shown in Equation 5.2.

$$I^* = \sqrt{I_o I_m} \tag{5.2}$$

Nutrient availability has an influence on the light determined carrying capacity of the phytoplankton population. When nutrients are excessive, phytoplankton will reach the biomass limit determined by light, whereas, if nutrients are limiting, phytoplankton cannot grow to density levels that reach the biomass limit determined by light (Chorus and Bartram, 1999).

Turbidity

Turbidity is a measure of the amount of light scattering that is produced by the suspended particles in a sample of water (Sullivan et al., 1988). High turbidity levels decrease the amount of light that can penetrate the water column, so that regardless of

nutrient levels and temperatures, the growth of phytoplankton is limited due to a lack of light (Sullivan, 1990). An inverse correlation between the turbidity and the concentration of cyanobacteria has been frequently observed (see Falter, 1978; Hötzel and Croome, 1994; Maier et al., 1998; Oliver, 1994; Sherman and Webster, 1997; Sullivan et al., 1988; Talling, 1971).

While the turbidity of a water body imposes a limit on phytoplankton growth, there are many species of cyanobacteria that are unable to withstand high light intensities over longer periods. A study conducted by Hobson et al. (1999) investigated the influence of three environmental parameters; irradiance, temperature and salinity, on biomass and toxin production by the cyanobacterium *N.spumigena* 001E isolated from Lake Alexandrina, South Australia. In this study it was found that biomass and toxin production was generally higher under light limited conditions. These results suggest that some species of cyanobacteria are better suited to lower levels of irradiance. Hence, this can limit their distribution to more turbid, eutrophic ecosystems (Chorus and Bartram, 1999).

In turbid waters it is likely that the euphotic zone will be more shallow than the epilimnion i.e. Z_{eu} / Z_m is < 1, hence, the cyanobacteria incapable of active movement are forced to spend part of the daylight in the dark. In these turbid waters, the extent of mixing and the corresponding depth of the epilimnion relative to the euphotic zone becomes a major primary production regulating factor (Geddes, 1984; Grobbelaar, 1985). As a consequence, in turbid and eutrophic waters, the dominant algae are the ecostrategists that are able to maintain themselves in the euphotic zone, especially cyanobacteria possessing gas vesicles, motile green algae, or low light adapted forms, such as diatoms (Wetzel, 1975).

Cyanobacteria require very little energy to maintain cell function and structure, hence, their maintenance constant is low (Gons, 1977, Van Liere et al., 1979 cited in Chorus and Bartram, 1999). When light intensities are low, this enables cyanobacteria to maintain a relatively higher growth rate than other forms of phytoplankton (Chorus and Bartram, 1999). It has been demonstrated in turbid waters, that, where light intensities are reduced (a frequent feature in Australian waters), cyanobacteria have a significant competitive advantage over other phytoplankton due to their low growth constant (Källqvist, 1981; Van Liere and Mur, 1979).

Colour

Like turbidity, colour also affects the optical properties of the water column. Low values of colour result in increased light penetration into the water column and a corresponding increase in phytoplankton growth.

Nutrients

Nitrogen and phosphorus are the nutrients that are most important for the growth of algae. The other nutrients that may play a role in algal growth are carbon, silica, iron, sulphur and molybdenum (MDBC, 1993).

The general response of aquatic ecosystems to additions of nitrogen and phosphorus is to increase the algal biomass (Conley et al., 1993; Watson et al., 1992). High levels of phosphorus coupled with a low nitrogen to phosphorus ratio can favour some nitrogen fixing cyanobacteria (MDBC, 1993; National Rivers Authority UK, 1990; Paerl, 1996). Many forms of cyanobacteria have a higher affinity for nitrogen or phosphorus when compared to other photosynthetic organisms. This enables them to out-compete other phytoplankton organisms during times of phosphorus or nitrogen limitation (Chorus and Bartram, 1999).

Nutrient enrichment, or eutrophication, of an aquatic ecosystem is a natural process that occurs over long periods of time (Maier and Dandy, 1994), and is largely due to the influence of climatic changes on the weathering and leaching of catchment soils, and on the nature of the vegetation they support (Sullivan, 1990). Human activity has sped up the process of eutrophication and has caused an increase in the amount of nitrogen and phosphorus entering the aquatic ecosystem. Clearance of vegetation and modern agricultural methods have changed the process of nutrient cycling in soils, resulting in nutrients being rapidly leached into water bodies. Point sources of nutrients such as sewage treatment plants, industrial plants, irrigation drainage, effluent from intensive animal industries, urban stormwater and groundwater have also contributed significantly to the process of eutrophication (MDBC, 1993).

At present, eutrophication is a common feature of many water bodies both in Australia and throughout the world (Falconer, 1994). The enrichment of aquatic ecosystems with dissolved nutrients has been identified as one of the world's major water quality problems (Vollenweider, 1968). The main elements involved in eutrophication are phosphorus, nitrogen (particularly the inorganic forms, such as phosphate, nitrate and ammonia) and to a lesser extent, carbon (Falconer, 1994).

Phosphorus

All living organisms are dependent on phosphorus. Phosphorus is an essential nutrient because it is involved in energy dynamics and protein synthesis (Falconer, 1994). The primary form of phosphorus used by cyanobacteria is inorganic phosphate (PO_4-P), however, some forms of organic phosphorus may be used when phosphate is scarce (Harris, 1994).

Phosphorus has the ability to adhere to reactive surfaces, both organic and inorganic, such as soil particles. This means that the majority of phosphorus in lakes and rivers is attached to particles, which then accumulate as sediments. Since a large percentage of the total phosphorus is associated with suspended particles, only a fraction becomes available to algae. Consequently, total phosphorus measurements overestimate the amount of bioavailable phosphorus. To avoid this problem, most phytoplankton monitoring programs target and measure soluble reactive phosphorus (SRP), or orthophosphate, because this is the portion of total phosphate that is available for algal growth.

Some phosphorus is also tied up in living pools, for example, in phytoplankton, zooplankton, fish and macrophytes. Hence, a water sample from the surface will only include some of these pools and will neglect the sedimentary pools of phosphorus (Harris, 1994).

Phosphate can be liberated via the degradation of organic matter and is taken up by bacteria and algae faster than it can be sampled and measured. In addition to this, some species of cyanobacteria are capable of storing large amounts of phosphorus. This enables them to perform up to two to four cell divisions corresponding to a 4-16 fold increase in biomass even if no SRP is measured (Chorus and Bartram, 1999). Therefore, it is the amount of phosphate that is bound within the cyanobacterial cells that largely determines the upper limit of cyanobacterial biomass and total phosphate is the variable that should be studied for managing biomass (Chorus and Bartram, 1999).

Even though biomass only requires about one seventh the amount of phosphorus as it needs of nitrogen, phosphorus is most often the limiting nutrient for phytoplankton growth in aquatic environments (Chorus and Bartram, 1999).

Nitrogen

Nitrogen is involved in the synthesis of amino acids and proteins and, therefore, is another essential constituent of all living organisms (Falconer, 1994). The atmosphere is the ultimate source of nitrogen with nitrogen gas (N_2) moving freely into and out of the aquatic ecosystem. Nitrogen can also enter the aquatic ecosystem in catchment runoff as breakdown products of organic substances or via untreated or biologically treated sewage, unless treatment includes nitrification and denitrification. Phytoplankton is able to utilise inorganic dissolved nitrogen in the form of nitrate, nitrite and ammonia (Chorus and Bartram, 1999).

Some cyanobacterial ecostrategists are capable of fixing atmospheric N_2 (Chorus and Bartram, 1999; Falconer, 1994; Harris, 1994; Maier and Dandy, 1994; National Rivers Authority UK, 1990; Paerl, 1996; Stewart et al., 1987; Varis, 1993; Wang et al., 1991). The major genera capable of nitrogen fixation include *Anabaena*, *Aphanizomenon*, *Cylindrospermopsis*, *Nodularia* and *Nostoc* (Chorus and Bartram, 1999). The above cyanobacteria possess the enzyme nitrogenase, which is used to convert N_2 to NH_4^+ so that, in turn, it can be used to generate the amino acids necessary for growth (Stewart et al., 1987). Under conditions of low dissolved inorganic nitrogen (nitrogen limitation), the cyanobacterial species capable of nitrogen fixation have a clear advantage over the other species of phytoplankton that are unable to fix nitrogen. Nitrogen fixation is a process that requires large amounts of light energy and as a consequence, is not as effective in very turbid waters (Chorus and Bartram, 1999).

Carbon

Carbon is a constituent of most cellular components and is important in the photosynthetic synthesis of sugars (Falconer, 1994; National Rivers Authority UK, 1990). Cyanobacteria are able to utilise dissolved carbon dioxide (CO_2) and bicarbonate which enters a water body largely by diffusion from the atmosphere (Falconer, 1994; Talling, 1976).

When phytoplankton growth is rapid, CO_2 gas will be withdrawn from solution faster than it is replaced, causing the water to lose its acidity and the pH to rise. The pH may rise substantially in soft waters, but in hard waters, buffered by bicarbonate, more CO_2 is released from the bicarbonate in solution which maintains the carbon supply and keeps the pH steady (National Rivers Authority UK, 1990).

On the basis of a variety of ecological observations, King (1970) suggested that cyanobacterial dominance in eutrophic waters is due to their ability to take up CO_2 more efficiently than green algae. This assertion was supported by Shapiro (1984), who conducted experiments in enclosed lakes and found that cyanobacteria become dominant when waters are enriched with phosphorus and nitrogen but CO_2 is limiting. These experiments suggest that cyanobacterial growth in nature is favoured over that of other phytoplankton, by low concentrations of dissolved inorganic carbon (Colman, 1989).

Micronutrients

Silica

Silica itself is not a direct requirement for blue-green algal growth. However, silica is an important nutrient for the growth of diatoms (Conley et al., 1993; Sullivan, 1990) and therefore, it is a key nutrient that determines phytoplankton succession.

Sullivan et al. (1988) investigated the relationship between silica and populations of diatoms, in particular, *Melosira*. The silica samples were filtered samples and only the dissolved silica was measured. An inverse relationship along the River Murray, for median numbers of *Melosira* and dissolved silica concentrations was observed (Figure 5.3). This suggests that large populations of diatoms deplete the water of its dissolved silica.

Blooms of cyanobacteria tend to be favoured by the silica depleted conditions which exist after the diatoms have exhausted their silica supply and ceased growth (Talling, 1986 cited in Harris, 1994).

Horn and Uhlmann (1995) studied the competitive growth of blue-green algae and diatoms in the Saidenbach Reservoir, Saxony. As the concentration of silica in the reservoir decreased, it was found that the diatoms became more and more repressed and finally were substituted by organisms with no silica requirements. Under natural conditions, cyanobacteria were often the best competitors.

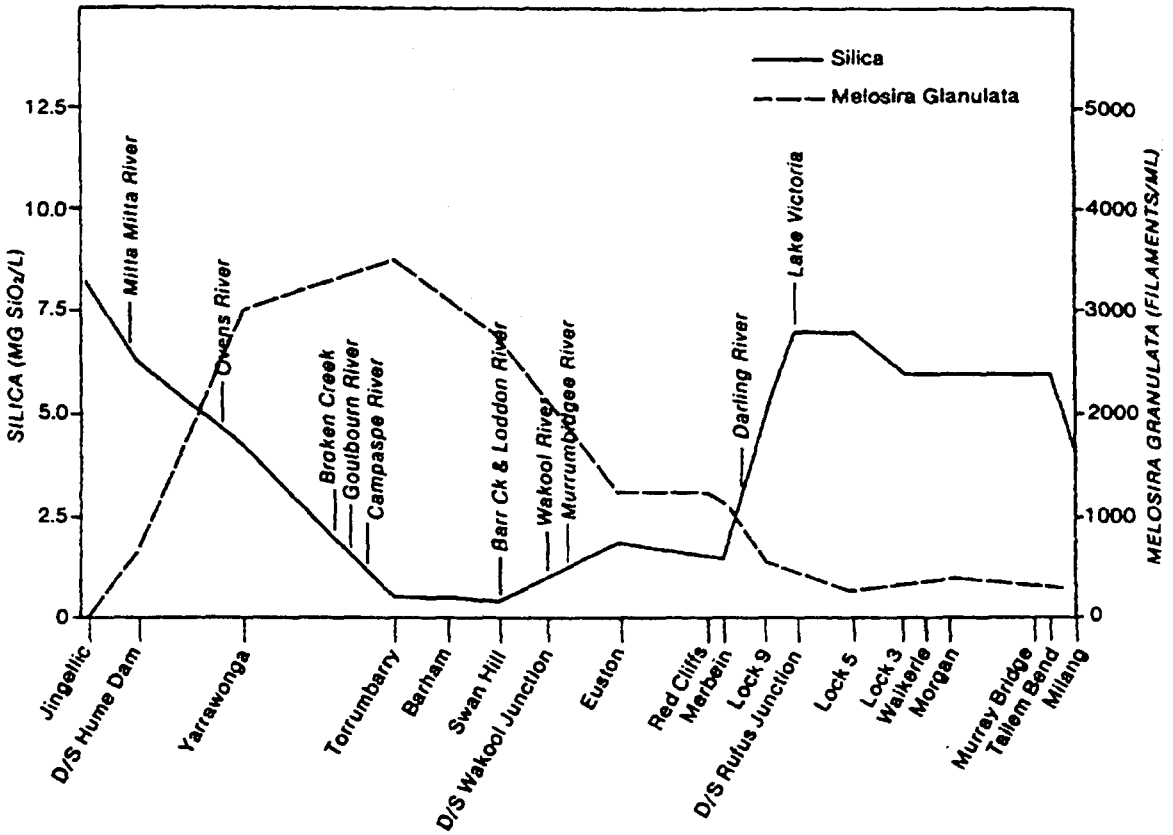

Figure 5.3 Median Silica (1978-1986) and *Melosira granulata* (1980-1985) Concentrations Along the River Murray (source: Sullivan, 1990)

Iron

Morton and Lee (1974) studied the effect of increasing iron concentrations on mixed populations of green algae (*Chlorella pyrenoidosa, Dictyospaerium pullchellum* and *Selenastrum capricornutum*) and blue-green algae (*Anabaena circinalis, Gleotrichia echinulata* and *Microcystis aeruginosa*). It was discovered that increases in the total iron concentration, even when already present in excess, favoured the growth of the blue-green algae over the green algae. Blue-green algae began to dominate over the green algae at concentrations of about 0.1-1.0 mg/L of total iron. No explanation for the mechanism that resulted in the stimulation of the blue-green algal growth was provided.

The presence of iron has been found to increase photosynthesis at low light intensities (Elder and Horn, 1977 cited in Falconer, 1994) and to stimulate nitrogen fixation (Murphy et al., 1976). Iron limitation can decrease the synthesis of photosynthetic pigments and can impair the organisational structure of the photosynthetic membranes (Reuter and Peterson, 1987). Reuter and Peterson (1987) suggested that iron is important in nitrogen fixation because it offers a competitive advantage for nitrogen

fixing cyanobacteria over other non-nitrogen fixing phytoplankton in situations of high iron concentration and low nitrate concentration.

Murphy et al. (1976) found that cyanobacteria are able to excrete iron-selective chelators which enable them to dominate other algae. It was observed that during blooms of blue-green algae the other algae were completely suppressed. The ability of the blue-green algae to suppress other algae was thought to be determined by the availability of iron. In conditions of iron deprivation, hydroxamate chelators were produced which appeared to be the agent suppressing the other algae. Murphy et al. (1976) concluded that the availability of iron may be an important factor determining the stability and composition of aquatic ecosystems.

Temperature

Temperature is an important variable influencing the growth of cyanobacteria. Very low temperatures will not necessarily kill cyanobacteria, but will slow their metabolic and biochemical processes (Sullivan, 1990).

Maier et al. (1998) observed that peak concentrations of *Anabaena spp.* occurred at water temperatures >20°C and usually in the range of 25-30°C. This concurs with Robarts and Zohary (1987), who found that maximum growth rates are attained by most cyanobacteria at temperatures above 25°C. These optimum temperatures are higher than for diatoms and green algae and they explain why cyanobacterial blooms usually always occur during the warmer summer months and usually in temperate climates (Chorus and Bartram, 1999).

While a general relationship of accelerated cyanobacterial growth with elevated temperature is often observed, temperature alone is unlikely to be the most important environmental variable. This point is highlighted by Hötzel and Croome (1994) who found that under certain conditions, other overriding factors were more important than temperature in bloom formation. Hötzel and Croome (1994) observed that the occurrence of peaks of cyanobacteria in the Darling River at Burtundy were not necessarily a summer phenomenon related to higher temperatures and irradiance, but rather an opportunistic response to conditions of low flow, turbulence and turbidity in a water rich with nutrients. Seventeen cyanobacterial peaks over 10 000 cells/mL occurred at Burtundy during 1980-92. Of these seventeen, five occurred at temperatures below 18°C, one peak occurred at 10-12°C, another occurred at 10-14°C, and an extended bloom from August to October 1985 was present at water temperatures of 10-22°C.

Robarts and Zohary (1987) found that direct temperature effects were secondary when compared to the indirect temperature effects. Hence, temperature effects are likely to be synergistic with other factors occurring within the scheme of an interacting ecosystem, that is, increasing temperature, decreasing nutrients and increasing stability of the water body.

5.2.1.2 'Top Down' Factors

The 'top down' factors are factors which arise higher up in the food chain. The planktonic communities of freshwaters include animals as well as plants, and many of the herbivores feed directly on the algae and bacteria present in the water body (Reynolds, 1984a). The activities of such herbivores may have a significant effect on the phytoplankton dynamics and population ecology.

It has been suggested that the degradation of aquatic ecosystems is one of a number of possible causes of the increasing frequency and intensity of algal blooms (MDBC, 1993). When environmental conditions are changed, the balance in the ecosystem may be disturbed, resulting in the removal of some components of the food chain and excessive, unchecked growth of others (Maier and Dandy, 1994).

Phytoplankton in eutrophic systems may not be as susceptible to "top down" factors because of the size of the phytoplankton cells and colonies. Larger celled species of phytoplankton are difficult for grazers to consume (Chorus and Bartram, 1999).

Grazing pressure

There are many species of planktonic algae that are grazed upon by fish, protozoa, daphnids and copepods (Chorus and Bartram, 1999; MDBC, 1993; Shapiro, 1990). However, the extent of grazing that occurs in Australian waters is not well understood. Studies conducted in the northern hemisphere cannot be uncritically accepted as typical of the Australian situation because their physical, chemical and biotic conditions are appreciably different (MDBC, 1993).

The interaction between fish and cyanobacteria can be direct in the case of fish grazing on cyanobacteria, or indirect in the case of fish preying on grazers of cyanobacteria (e.g. zooplankton). In Australia, interactions between fish and cyanobacteria have received little study and the capacity for fish to graze on algal blooms is not known (Gehrke and Harris, 1994). However, it can be assumed that grazing of cyanobacteria by native

Australian freshwater fish is probably trivial as they lack mechanisms to process blue-green algal cells effectively (Gehrke and Harris, 1994). The indirect effect of benthic and planktivorous fish preying on zooplankton, which feed on algae, is likely to be more significant.

In Australia, the introduction of non-native fish into the River Murray, such as *Cyprinus carpio L.* (European carp) and *Gambusia affinis* (Mosquito fish), has had detrimental effects on the ecosystem, including (MDBMC, 1994):

- The disturbance of sediments on the river bed.
- A decline in the number of water plants.
- A decline in the number of smaller native algal predators.

In Europe, it has been observed that the ability of the zooplankton community to graze on blue-green algae depends on the presence of large grazers, especially *Daphnia* spp. (Dawidowicz, 1990). However *Daphnia* spp. are not the most abundant zooplanktors in Australian reservoirs and rivers, and if present at all, are seasonal.

There have been only very few reports of zooplankton grazing on cyanobacteria in Australian fresh waters. Calenoid copepods and rotifers are the most common form of zooplankton in Australian waters and these have less potential for controlling cyanobacterial blooms than do large cladocerans (Boon, 1994).

Higher turbidity also reduces the capacity of the grazing community to control algal growth. Jack et al. (1993) found that high turbidities had a direct suppressive effect on grazing by interfering with the grazing ability of cladocerans, copepods and rotifers.

Pathogens

Viruses, fungi, bacteria, amoebae and actinomycetes attack cyanobacteria and have the potential to break down populations, however, the importance of these natural enemies is not well understood (Chorus and Bartram, 1999; Daft et al., 1985).

5.2.2 Cyanobacterial Toxins, Their Production and Effects

Examples of toxic blooms and scums of *Microcystis, Anabaena, Oscillatoria (Planktothrix) and Aphanizomenon* species have been reported in many countries (e.g. Baker and Humpage, 1994; Chorus and Bartram, 1999; Codd, 1995; Codd et al., 1994; National Rivers Authority UK, 1990; Sivonen et al., 1990; Steffensen et al., 1993).

With the identification of several toxic cyanobacterial metabolites present in these blooms, concern for the potential health effects experienced by consumers of this water has grown in recent years (Steffensen et al., 1999).

An increasing awareness and understanding of cyanotoxins is a priority at present because of (Codd, 1998):

- The increasing occurrence of cyanobacterial blooms in water resources.
- The increasing knowledge of the types, occurrence, levels and toxicity of the toxins.
- The emerging examples of human and animal acute and chronic health incidents and problems associated with the toxins.

5.2.2.1 Effects on Public Health

The range of compounds produced by cyanobacteria, which include toxins and odours, have a deleterious effect upon the quality of water in both drinking water supplies and in water bodies used for recreational purposes (Burch and Steffensen, 1997). The major routes of human exposure to cyanotoxins include oral and dermal routes through drinking water and recreational water use, however, other forms of exposure include (Chorus and Bartram, 1999):

- Inhalation whilst showering, water-skiing or during certain work practices.
- Oral ingestion via cyanobacterial dietary supplements, such as *Spirulina* (mainly if the cyanotoxin levels in the supplements are not controlled).
- Intravenous exposure via dialysis clinics.

There are three principal sources of evidence which support the adverse human health effects from cyanotoxins. These are epidemiological evidence including human poisonings, animal poisonings, and toxicological studies (Chorus and Bartram, 1999). A summary of the various classes of toxins produced in Australia and their significance on public health is given in Table 5.2.

Table 5.2 Major Classes of Toxins Produced by Cyanobacteria in Australia and Their Significance to Drinking Water Quality and Public Health (source: adapted from Burch, 1999)

Type	Compound	Organisms	Effects	Drinking Water Quality and Public Health Significance
TOXINS				
Hepato-toxins	Microcystin	*Microcystis aeruginosa Anabaena spp.*	Liver Damage Tumour promotion in animal studies	The introduction of guidelines for microcystin-LR will oblige operators to monitor for the chemical compound, and this will require access to appropriate monitoring techniques These guidelines may or may not include reference to cell numbers
	Cylindro-spermopsin	*Cylindrospermopsis raciborskii Aphanizomenon ovalisporum*	Cytotoxic Liver, kidney and other organ damage	Implications for monitoring - as for microcystin This toxin is a significant water quality issue for tropical Australia Cannot be disregarded as an issue in southern Australia
Neuro-toxins	Saxitoxin class of compounds (Paralytic Shellfish Poison – PSP's)	*Anabaena circinalis*	Sodium channel blocking agent – acute poisoning results in death by paralysis and respiratory failure	Implications for monitoring - as for microcystin Not likely to be as significant a drinking water quality issue as microcystins, although this is dependent upon the guidelines
Endo-toxins	Lipopoly-saccharides (LPS)	Most cyanobacteria (outer cell wall component similar to LPS in cell walls of gram negative bacteria)	Implicated in Gastro-intestinal disorders Skin, eye irritation Skin rashes Respiratory symptoms	May become an issue for health investigation for recreational and drinking water, but lack of information makes it difficult to assess their significance Poor information base makes guideline development unlikely at present

Cyanotoxins are largely contained within the cyanobacterial cells. The two mechanisms most commonly responsible for the mass release of cyanotoxins are the breakdown of natural populations of cyanobacteria and the artificial cell lysis of a bloom by the application of copper sulphate (Chorus and Bartram, 1999; Maier and Dandy, 1994). These two mechanisms give rise to most of the gastrointestinal and hepatic illness resulting from cyanotoxins in drinking water supplies.

While acute toxicity is the most common problem encountered with cyanobacterial poisonings, a long term risk may also be present (Chorus and Bartram, 1999). Although the majority of animal experiments have focussed on the short-term acute effects, there are some animal experiments that have shown chronic liver injury resulting from continuing oral exposure to microcystins (e.g. Falconer et al., 1988). In particular, the potential for tumour promotion and carcinogenesis still needs to be evaluated. Falconer and Humpage (1996) stated that whole animal studies and human epidemiology is needed to unravel the potential risks of cyanotoxins in promoting tumour growth.

Hepatotoxins

The known cyanobacterial hepatotoxins found in drinking water can be divided into two groups (Falconer, 1996):

1. The cyclic peptide toxins from *Microcystis* and *Nodularia*.
2. The alkaloid toxin from *Cylindrospermopsis raciborskii*.

The hepatotoxins produced by cyanobacteria in Australia are summarised in Table 5.2. and include microcystin and cylindrospermopsin.

Microcystin

The known freshwater cyanobacterial genera capable of producing microcystin are *Microcystis, Anabaena, Nodularia, Nostoc* and *Planktothrix (Oscillatoria)* (Kiviranta et al., 1992; Luukkainen et al., 1994; Namikoshi et al., 1992; Sivonen et al., 1992; Vasconcelos et al., 1993).

In Australia, the predominant cyanobacterium producing microcystin is *Microcystis aeruginosa*, however, microcystins can occasionally be produced by *Anabaena* spp. (Burch, 1999). Falconer et al. (1983) found that microcystins produced by blooms of

Microcystis were implicated in causing liver damage in a human population. As the algae passed through kilometres of pipe, and through pressure reducing valves, it was believed that cell lysis was likely to occur, thereby freeing the toxin into the water and exposing the population via the drinking water supply.

The primary target of microcystins is the liver, where disruption of the cytoskeleton, consequent on inhibition of protein phosphatases 1 and 2A, causes massive hepatic haemorrhage (Dawson, 1998). Microcystins have also been found to promote tumour growth in experimental animals (Falconer and Humpage, 1996), and the impact on humans subject to chronic exposure via drinking water supplies is unclear (Burch, 1999).

Cylindrospermopsin

In 1979, there was a major outbreak of hepatoenteritis among the children of an Aboriginal community at Palm Island in northern Queensland, Australia (Blyth, 1980). Further investigation determined that the cause of the hepatoenteritis was a bloom of *Cylindrospermopsis raciborskii* in the water supply reservoir and subsequently, a new cyanobacterial toxin, cylindrospermopsin, was discovered (Chorus and Bartram, 1999).

Recently, cylindrospermopsin has also been isolated in *Aphanizomenon ovalisporum* (Shaw et al., 1999). Cylindrospermopsin is cytotoxic and its biochemical mechanism of action has been shown to be inhibition of protein synthesis (Falconer, 1996). The experimental dosing of mice with extracts of *C. raciborskii* was observed to cause injury to the intestine, kidneys, adrenals and lungs, with the most damage occurring in the liver (Hawkins et al., 1985).

The clearance of cylindrospermopsin from the body is slower than that of microcystin (Maier and Dandy, 1994).

Neurotoxins

The neurotoxins produced by cyanobacteria include the Paralytic Shellfish Poisons (PSPs) anatoxin-a, anatoxin-a(s), homoanatoxin-a, saxitoxin, neosaxitoxin and gonyautotoxin (Falconer, 1996; Rapala et al., 1993; Sivonen, 1996). Neurotoxins are able to disrupt signaling between the nerves and muscles, thereby acting as neuromuscular blocking agents, and if given in high enough doses to humans and animals, these toxins can cause death by respiratory paralysis (Burch, 1999). The general class of PSPs have been responsible for a significant amount of human

morbidity and mortality, brought about via ingestion of contaminated shellfish. This is highlighted by the epidemic of shellfish poisonings that occurred throughout Western Europe in 1976 (Luthy, 1979).

The production of neurotoxins is most commonly associated with species of *Anabaena* and *Aphanizomenon,* and in Australia, *Anabaena circinalis* has been shown to produce the saxitoxin class of neurotoxins (Baker and Humpage, 1994). Neurotoxins can be cleared by the body quite rapidly (Maier and Dandy, 1994), and they may reservedly be considered less of a threat to public health via public water supply than hepatotoxins (Burch, 1999). However, the reservations are that there is little information on the occurrence of these highly toxic compounds in drinking water and these compounds have not yet been the subject of medium or long term chronic animal studies (Burch, 1999).

Endotoxins

Lipopolysaccharides (LPS) are components of the cell wall of cyanobacteria and these endotoxins are ubiquitously associated, in varying amounts, with all cyanobacteria, irrespective of whether they produce the better known hepato- or neurotoxins (Burch, 1999). Endotoxins can elicit irritant and allergic responses in human and animal tissues that come into contact with them resulting in gastroenteritis, skin irritation, eye irritation, hay-fever-like symptoms and asthma (Chorus and Bartram, 1999; Jones et al., 1993). Since skin irritation is the predominant reaction to endotoxin exposure, the most likely risk of being exposed via a water supply system occurs through bathing and showering in contaminated water. Other sources of exposure to these toxins include bathing or coming into contact with contaminated water in a recreational water body.

Case studies

For a history of suspected cyanobacterial poisonings in humans see Chorus and Bartram (1999); Maier (1995); National Rivers Authority UK (1990).

5.2.2.2 Effects on Animal Health

The toxins discussed in Section 3.1 also have a deleterious effect on animal health. These effects have been demonstrated in numerous animal experiments (e.g. Falconer et al., 1988; Galey et al., 1987; Hooser et al., 1990; Jackson et al., 1984). Cyanobacterial poisoning of animals can occur by two routes: through direct consumption of the cyanobacterial cells from the water, or indirectly by consuming

other animals that have themselves fed on cyanobacteria and have accumulated toxins (Chorus and Bartram, 1999).

The tendency of most of the species of cyanobacteria to form surface scums means that under the right conditions, accumulations of cyanobacterial scum-forming material may be stranded around the edge of a water body. It is the accumulation of these shoreline scums that can present animals, drinking from the edge, with a highly concentrated source of contaminated water, possibly containing cyanobacterial toxins (National Rivers Authority UK, 1990).

Reports extending back almost a century ago give accounts of animal deaths, both domestic and wild, that have been attributed to cyanobacterial blooms. Cases have included sheep, horses, cattle, pigs, dogs, wild fish, farmed fish, rodents, amphibians, waterfowl, bats, zebras and rhinoceros (National Rivers Authority UK, 1990).

5.2.2.3 Ecological Effects

The ecological effects of cyanotoxins include; the poisoning of wild animals (Section 5.2.2.2) and the accumulation of toxins in shellfish, zooplankton and fish (Chorus and Bartram, 1999; Jones et al., 1993; Luthy, 1979).

Toxic effects on stream biota

When algal matter decays there is an increase in the biological oxygen demand (BOD) of the water due to bacterial metabolism. Consequently, this can lead to very low levels of oxygen available for aquatic organisms, resulting in great stress and even death of aquatic biota, especially fish (Maier and Dandy, 1994). Therefore, it is difficult to unequivocally ascribe the death of natural aquatic biota to cyanotoxin poisoning, when some of the deaths may be due to the anoxia (Chorus and Bartram, 1999).

5.2.2.4 Effects on Recreation and Tourism

In some parts of the world, recreational water use is likely to be a major route of exposure to cyanotoxins. Some recreational activities such as swimming, fishing, windsurfing, canoeing and water skiing expose participants via intake through oral ingestion or inhalation, or by direct contact, which may result in allergic and toxic dermal reactions. When a cyanobacterial bloom occurs in a water body, the use of that water body for recreational purposes may be prohibited. This is dependent on the guidelines set by the relevant health authority and the nature of the bloom. Such

incidents may affect tourism, resulting in economic losses to tour operators, caravan park owners and local shopkeepers (Maier and Dandy, 1994).

5.2.3 Toxic Cyanobacteria in the Murray-Darling Basin

As discussed in Section 4.2.1, river regulation structures have been constructed along the rivers within the Murray-Darling Basin to overcome water shortages during periods of drought. In the River Murray, the structures have improved the use of the river for navigation, water supply purposes and flood control. At present, two-thirds of the water in the Murray-Darling river system that would have originally reached the sea is extracted for consumptive use (MDBMC, 1994).

The Murray-Darling Basin Commission (MDBC) has the responsibility of regulating the flow in the River Murray, its anabranches and the Darling River downstream of Menindee Lakes (Jacobs, 1990). The main aim of river regulation is to meet the water requirements of the three states (South Australia, New South Wales and Victoria), while ensuring minimum wastage. During the dry summer and autumn months at times of peak irrigation and water consumption, flow in the system can be reduced significantly (Figure 4.2). The periods of low flow are now more frequent and prolonged and this is believed to be a critical factor which has increased the frequency of algal blooms in the Murray-Darling river system (MDBMC, 1994).

Most of the basin is a naturally saline environment, turbidity levels are naturally high and a large quantity of the nutrients are naturally occurring, originating in the rocks and soils of the catchment (MDBC, 1999). It is clear that human activities have exacerbated the above features, which has resulted in deterioration of the water quality throughout the basin.

In recent years, blooms of cyanobacteria have been prevalent in the surface waters of the Murray-Darling Basin (MDBMC, 1994). Although cyanobacterial blooms are naturally occurring, it has been recognised that they are a symptom of water quality deterioration. Baker and Humpage (1994) surveyed the Murray-Darling Basin and found that a substantial proportion, (42%), of the blooms surveyed were toxic. The most common cyanobacteria in the rivers of the Murray-Darling Basin are *Anabaena, Aphanizomenon, Anabaenopsis* and *Microcystis* (Sullivan, 1990; Sullivan et al., 1988).

In response to the cyanobacterial problem, the Murray-Darling Basin Ministerial Council has prepared an Algal Management Strategy, with the intention of reducing the frequency and intensity of algal (cyanobacterial) blooms and other associated water

quality problems (MDBMC, 1994). The strategy has identified that the environmental processes involved in the formation of blooms are not very well understood and aims to address the problem with a strong emphasis on research and management. The strategy has also identified that a key objective is to develop a modelling capability for processes related to the development of cyanobacterial blooms.

The strategy aims to achieve its goals through a combination of actions in five areas, including:

- Improved flow regimes and flow management.
- Reduced nutrient concentrations in the streams and storages of the basin.
- Heightened community awareness.
- Improved scientific knowledge.
- Progressive refinement of the strategy.

5.2.3.1 History of Bloom Formation

The first report of toxic cyanobacteria in the scientific literature was made by a Government Analyst, George Francis (Francis, 1878). In this account, livestock deaths at Lake Alexandrina, South Australia were attributed to *Nodularia spumigena*. Also, late last century, reports of green scums along the River Murray were made by police officers to the Chief Inspector of Stock and thence to the Commissioner of Crown Lands (Anon. 1888 cited in Codd et al., 1994). Since then, sporadic accounts of blooms of cyanobacteria have been reported, and in most cases, the dominant genera implicated in toxic incidents have been *Microcystis*, *Anabaena* and *Nodularia* (Steffensen et al., 1993).

The occurrence of toxic blooms of cyanobacteria within the surface waters of the Murray-Darling Basin has become commonplace in recent years, with blooms being reported in rivers, lakes, water supply reservoirs, billabongs and farm dams (Steffensen et al., 1993). The most notable bloom of recent years occurred in the summer of 1991-92 when "the largest river bloom of blue-green algae recorded anywhere in the world emerged along the Darling River" (MDBMC, 1994). This toxic bloom of *Anabaena circinalis* extended for over 1000 km and occurred at a time of drought when flows in the Darling River were minimal. Deaths of approximately 1600 sheep and 40 cattle were attributed to the bloom and preventative measures were taken to avoid riverside communities consuming contaminated water (Codd et al., 1994). During the bloom the cost of emergency arrangements for stock and domestic water supplies exceeded

AUD$1 million (Anon. 1992 cited in Hötzel and Croome, 1994). It was the Darling River bloom that sparked recognition and focus on blooms in rivers.

It is argued that the greater number of algal bloom reports may be due to a heightened public awareness of the issue and that no conclusive evidence exists to indicate that the incidence of algal blooms has increased, especially since the 1970s (MDBC, 1999). While it may be difficult to prove that the incidence of blooms has increased since the 1970s, the influence of European settlement in the Murray-Darling Basin has undoubtedly increased the frequency of blooms of cyanobacteria through nutrient enrichment and alteration of the natural state of the aquatic ecosystem. A large body of literature supports the argument that indications of an increasing frequency and intensity of blooms in the Murray-Darling Basin arise from the eutrophication of source waters, due to intensive agriculture and the discharge of urban and industrial effluents (see Chorus and Bartram, 1999; Falconer, 1988; Harris, 1994; Maier and Dandy, 1994; MDBC, 1993; MDBMC, 1994; Sullivan, 1990).

Croome, (1980) found that the historical records indicate that blooms in the River Murray are predominantly confined to the lower reaches in South Australia, with the possible exception of Lake Hume and Lake Mulwala.

5.2.3.2 Long Term Methods of Controlling Cyanobacteria

Methods of controlling cyanobacteria usually exploit their preference for calm stable waters, the association with high nutrient loading, the isolation of seeding areas, and the promotion of grazing by zooplankton (Steffensen et al., 1999).

Flow management

The hydrodynamics of a river system are very important in determining the population ecology of river phytoplankton (Reynolds, 1984a). The past few decades have seen a steady increase in the amount of water diverted from the Murray-Darling River system, often leaving very little flow for 'environmental' purposes (MDBC, 1993). Low flow is often regarded as a stimulatory factor for the growth of cyanobacteria in rivers (Jones, 1994). An inverse correlation between flow and cyanobacteria concentrations is one of the only relationships that has been consistently observed (see Harris, 1994; Jones, 1994; Maier and Dandy, 1997b; Sullivan et al., 1988). As a consequence, the changed flow regime in the Murray-Darling River system has been identified as a significant component of the algal bloom problem (MDBC, 1993).

Burch et al. (1998) simulated various operational flow response strategies designed to disrupt a cyanobacterial bloom at several key locations along the River Murray. The strategies that were modelled involved changing weir levels to spill water between weir pools, thereby disrupting site specific thermal stratification by increasing velocities. Relatively large (0.5m) reductions in pool levels were found to be required to diurnally mix the water column and make conditions less favourable for the growth of cyanobacteria.

Jones (1994) investigated a novel control strategy that involved the installation of weir pool syphons at Maude Weir on the Murrumbidgee River. Syphons (0.3m diameter), capable of carrying up to 35 ML/day each, were positioned on top of the weir. Approximately half of the average daily summer flow through Maude Weir was transferred over the weir by the syphons. It was envisaged that the syphons would work in two ways to control cyanobacterial growth. Firstly, by reducing the surface water residence time, the amount of solar radiation absorbed by this top layer of water is reduced, thereby decreasing the surface temperature. Secondly, *Anabaena* are buoyant and even in mildly turbulent conditions will tend to concentrate in the surface layers of the weir pool. Hence, syphons would continually remove the *Anabaena* cells from the surface layer and prevent build up of a large population. During the period of use it was found that the syphons significantly reduced the abundance of *Anabaena*. However, it was not determined whether the low *Anabaena* abundance in Maude Weir was a chance occurrence or caused as a direct result of the use of syphons.

Nutrient control

In the lower reaches of the River Murray, the levels of oxidised nitrogen are usually low, whilst phosphorus levels are generally much greater than the levels required for phytoplankton growth (Maier and Dandy, 1994). This creates a situation that is more favourable for the growth of nitrogen fixing cyanobacteria. The Algal Management Strategy has highlighted the following measures that may be adopted to reduce the nutrient inputs to streams (MDBMC, 1994):

- Reducing the amount of phosphorus in detergents.
- Treating effluent to remove nutrients.
- Recycling water.
- Improving irrigation techniques.
- Using filtration systems such as pasture crops, buffer strips, grassed or vegetated drains, wetlands and retardation basins.

Biomanipulation

In eutrophic aquatic systems, control of phytoplankton biomass by food chain manipulations is less successful and in any case is unpredictable in timing or effect, due to the size of the phytoplankton cells and colonies (Shapiro, 1990). The impact of planktivorous fish on zooplankton and phytoplankton is still largely unknown for Australian conditions and more research is needed before the biomanipulation approach can be used successfully within the Murray-Darling Basin.

5.2.4 Modelling Cyanobacteria

Attempts to model phytoplankton communities have their roots in predictive limnology. Over 30 years ago, Vollenweider (1968) recognised that a new approach to limnology was necessary: one rooted in empiricism and directed to the prediction of salient properties of the whole system. This approach attracted other researchers and in the initial stages, the constructs used to predict phytoplankton were highly simplified models (e.g. Jorgensen, 1976) and regressions (e.g. Dillon and Rigler, 1974; Vollenweider, 1976), utilising only a few variables but based on extensive collections of data.

In general, there are four approaches that have been used to model the incidence of phytoplankton populations, these include: process-based (deterministic) models; statistically based models; rule-based (heuristic) models; and, more recently, ANN models. The former three techniques have not been successful at modelling phytoplankton populations (French and Recknagel, 1994b), however, recently, the ANN technique has shown that it is a flexible approach, giving significant improvement in the way phytoplankton populations can be modelled (Maier et al., 1998). For a review of the different techniques used to model the incidence of phytoplankton populations, see: Recknagel et al. (1997), French and Recknagel (1994b) and Maier et al. (1998).

5.2.4.1 Process-Based (Deterministic) Models

Process-based models are concerned with the underlying physical processes affecting the size of the phytoplankton populations and generally maximise the use of available scientific knowledge. In these models the rate of change in the size of phytoplankton populations is related to the physical, biological and chemical processes using ordinary or partial differential equations (e.g. Bormans and Condie, 1998; Di Toro et al., 1975; Jorgensen, 1976; Olsen et al., 1999; Varis, 1993). The main advantage of process-

based simulation models is that they have a wider domain of applicability than statistically based models because they are based on fundamental physical and biological relationships (Maier and Dandy, 2000a). Di Toro et al. (1975), Jorgensen (1976) and Varis (1993) developed process-based models that relate phytoplankton biomass to a number of causal variables (solar radiation levels, temperature and nutrient levels) as well as zooplankton grazing in lakes and reservoirs.

Bormans and Condie (1998) developed a process-based model to predict the stratification dynamics and the corresponding concentration of *Melosira* and *Anabaena.* The model took into account diurnal temperature stratification and vertical mixing due to surface evaporation and night time cooling experienced by the phytoplankton populations. Stratification was predicted from the net surface heat flux, wind speed, air temperature and relative humidity. A constant sinking velocity was assumed for *Melosira* and a constant floating velocity for *Anabaena* (due to the presence of gas vesicles). The algal growth rate was assumed to be strictly light limited, with no nutrient limitation. The model was successful in reproducing the time evolution of the concentrations of the two algal species, however, the model was only applied to low flow conditions where the effects of resuspension and advection were negligible.

Two key findings with respect to sampling and management were highlighted by the study. Firstly, the results collected showed that the much-used grab surface sample for algal counts was not at all representative of the water column under stratified conditions. Secondly, the time of day was found to be important in determining the vertical distribution of phytoplankton.

5.2.4.2 Statistically Based Models

Traditionally, regression analysis has been the most common statistical model, making use of historical data to relate the annual peak concentration (or biomass) of phytoplankton to variables of total phosphorus, total nitrogen, temperature, turbidity and water depth (Maier et al., 1998). The simplest of the aforementioned models are those that relate the concentration of chlorophyll-*a* to concentrations of total phosphorus (e.g. Dillon and Rigler, 1974; Jones and Bachmann, 1976; Vollenweider and Kerekes, 1982). The Organisation for Economic Cooperation and Development (OECD) model developed by Vollenweider (1982), is the most comprehensive of these models. Data for annual mean values of chlorophyll-a were compiled from a wide variety of phosphorus limited lakes (predominantly in North America and Europe) and related to annual mean concentrations of phosphorus. The OECD model has been widely used as a management tool to predict the average and the maximum

phytoplankton biomass range for a given phosphorus concentration. However, this approach must be used with caution as the model integrates the behaviour of a series of lakes rather than the response of any one lake to changes in the phosphorus concentration.

Smith (1985) developed regression models that relate summer mean blue-green algal biomass to total phosphorus concentrations and the mean depth of the lake. Smith et al. (1987) used stepwise multiple regression analysis to predict the summer peak biomass of four different species of cyanobacteria in a number of Swedish lakes. In addition to total phosphorus concentrations and the mean depth of the lake, the variables used in the study include water temperature, total nitrogen and total CO_2 concentration.

Varis (1991) used canonical analysis to study the associations between phytoplankton species and nine physical and chemical growth factors in a polyhumic Finnish lake. The nine variables used include water temperature, irradiance, total nitrogen, ammonia, nitrate, total phosphorus, orthophosphate and the ratios of total nitrogen to total phosphorus and dissolved inorganic nitrogen to dissolved inorganic phosphorus.

5.2.4.3 Rule-based (Heuristic) Models

Rule-based models (e.g. Recknagel et al., 1994; Reynolds, 1984a) make use of a knowledge base of human expertise on the factors that affect the population size of various species of phytoplankton in order to predict the species composition and phytoplankton population size. The accuracy of the predictions is based on the quality of the data and the rules obtained from the human expert.

Rule-based models derive predictions by running the knowledge base through an inference engine (a software program that interacts with the user and processes the results from the rules and data in the knowledge base). Recknagel et al. (1994) used fuzzy models in order to quantify the predictions from the rule-based approach.

5.2.4.4 Artificial Neural Network Models

The traditional statistical and process-based approaches have been unsuccessful at forecasting the magnitude and timing of particular species or species groups of phytoplankton (French and Recknagel, 1994b). ANN models are often perceived to overcome some of the difficulties that are associated with statistical models and, as a consequence, ANN models have been widely adopted by modellers in the fields of hydrology and water resources modelling. Recently, several researchers have applied

ANNs to water quality case studies aimed at predicting the size of phytoplankton populations and the timing of blooms (French and Recknagel, 1994a; French and Recknagel, 1994b; Maier, 1995; Maier and Dandy, 1997b; Maier et al., 1998; Recknagel, 1997; Recknagel et al., 1997; Recknagel et al., 1998; Whitehead et al., 1997; Yabunaka et al., 1997). The results of these studies have shown that ANN models are a very promising predictive tool for modelling phytoplankton populations.

5.2.5 Forecasting *Anabaena* spp. at Morgan

The case study in this chapter involves the development of an ANN model capable of forecasting concentrations of *Anabaena* spp. in the River Murray. It was considered important to model *Anabaena* because it is the most common genus of nuisance cyanobacteria in the lower reaches of the Murray. *Anabaena* is capable of producing all three types of toxins (i.e. hepatotoxins, neurotoxins and endotoxins) and is also significant from an aesthetic perspective, as it is capable of producing compounds associated with tastes and odours in drinking water. Morgan was selected as the modelling location as a WTP is located there and water abstracted at this point is delivered to the cities of Port Pirie, Port Augusta and Whyalla via the Morgan-Whyalla pipeline (Figure 4.3). In addition, Morgan is the only site in the South Australian reaches of the River Murray for which a reasonable set of data was available.

5.3 Available Data

The data set used in this study was supplied from a number of sources, including: the Murray-Darling Basin Commission (MDBC); the Australian Water Quality Centre (AWQC); the Bureau of Meteorology (MET); and the South Australian Department for Water Resources (DWR). All data were collected as part of routine monitoring and consequently, were not collected specifically for the purposes of this study. A summary of the available data is given in Table 5.3. The river level data used in this research are given above the Australian Height Datum (AHD).

The data used for training, testing and validation were available from 08-01-1980 to 20-11-1996 and were collected at Morgan; with the exception of wind run data which were collected at Blanchetown and flow data (flow into South Australia), which were recorded at the South Australian border with New South Wales. An additional set of data from 26-11-1997 to 13-02-2002 was collected for a real-time, independent validation period. All daily data (flows and river levels) were converted to weekly averages.

Table 5.3 Available Data

Variable	Units	Sampling Interval	Source
Anabaena spp.	cells/mL	weekly	AWQC
Flow	ML/day	daily	DWR
River Level	m	daily	DWR
Temperature	°C	weekly	MDBC
Colour	Hazen Units (HU)	weekly	MDBC
Turbidity	Nephelometric Turbidity Units (NTU)	weekly	MDBC
pH	-	weekly	MDBC
Silica	mg/L	weekly	MDBC
Total Kjedahl Nitrogen (TKN)	mg/L	weekly	MDBC
Total Phosphorus (TP)	mg/L	weekly	MDBC
Soluble Phosphorus (SP)	mg/L	weekly	MDBC
Wind Run Below 3m	km	daily	MET

The mean, standard deviation, maxima and minima of the available time series from 1980 to 1996 are shown in Table 5.4.

Table 5.4 Statistics of the Available Time Series

Variable	Mean	Standard Deviation	Maximum	Minimum
Anabaena spp.(cells/mL)	241	1105	25393	0
Flow (ML/day)	20296	23373	108923	1936
River Level (m)	3.93	1.07	7.64	3.18
Temperature (°C)	18.7	4.9	30.0	10.0
Colour (HU)	22.5	18.1	102.0	4.0
Turbidity (NTU)	68.3	55.9	410.0	4.4
pH	7.9	0.4	9.3	6.7
Silica (mg/L)	4.8	3.7	17.0	1.0
TKN (mg/L)	0.88	0.32	2.28	0.05
TP (mg/L)	0.134	0.091	1.500	0.005
SP (mg/L)	0.029	0.050	0.970	0.002
Wind Run Below 3m (km)	105	46	264	23

Anabaena spp. represents a species complex, and these counts comprise the following species: *Anabaena circinalis* Rabenhorst; *Anabaena flos-aquae* f. *flos-aquae* (Lyngb.) Komárek; *Anabaena spiroides* f. *spiroides* (Elenkin) Komárek; *Anabaena spiroides* f. *crassa* (Lemm.) Elenkin; *Anabaena solitaria* Klebahn and *Anabaena aphanizomenoides* Forti. The counts for individual species were combined because there was a change in the taxonomic recognition of species in the laboratory performing the counts part way during the time period of the data set.

Unfortunately, data for many other variables that are likely to influence the concentration of cyanobacteria were not available, including: solar irradiance; competitive interactions with other taxa such as algal predators (e.g. zooplankton and planktivorous fish), competing species of phytoplankton and pathogens which may attack cyanobacterial cells (e.g. viruses and actinomycetes); the thermal profile of the river and dissolved oxygen levels. It is also important to note that there is a large degree of uncertainty in the *Anabaena* spp. data. This uncertainty is derived from both sampling and counting errors. The causes of this uncertainty are described below.

Cyanobacteria exhibit patchiness in their distribution in all three spatial dimensions. Vertical patchiness results from a stratified water column, in which certain species of cyanobacteria can utilise their intracellular gas vesicles to control their position in the water column. Horizontal patchiness is particularly pronounced with cyanobacterial species because prevailing winds tend to cause accumulation downwind along bends in

reaches of the river (Jones et al., 2003). Unrepresentative sampling, and hence, sampling error, may result from not taking the patchiness of cyanobacteria into account in the sampling program. To overcome the problem of vertical patchiness, it is necessary to take an integrated depth sample of the water column. Averaging the cyanobacterial concentration over depth provides a better representation of the 'true' population. To overcome horizontal patchiness, it is necessary to take a number of samples across the cross-section of the river. Such open-water sampling requires the use of a boat. The samples taken across the cross-section can then be pooled and the resulting composite sample thoroughly mixed.

In this study, historical data were used and these data were grab sample data collected at a fixed depth and location in the river. No replicate samples were taken. The fixed sampling depth corresponded to the Morgan WTP off-take and was located at a depth of 0.3m below the water surface and approximately 10m from the river bank. The river at this point is approximately 150m wide. The rationale behind this sampling location was that it would provide an indication of the total load of cyanobacteria, and their toxins, entering the water treatment plant in the raw water. However, a fixed depth sample does not provide a good indication of the 'true' population of cyanobacteria in the river and such an unrepresentative sample is less useful for ecological studies.

The frequency and timing of the collection of samples is also an important consideration as these factors can also contribute to sampling uncertainty. Cyanobacteria in the field can exhibit growth rates from 0.1 – 0.4 d^{-1}, which equate to population doubling times of between two days and a week (Jones et al., 2003). Therefore, with favourable environmental conditions, growth rates of cyanobacteria may be such that a population doubling time of 2 days is achieved. To adequately detect an upward trend in growth and to capture the population dynamics, a sampling frequency of up to twice weekly may be required (Jones et al., 2003). However, the cyanobacteria data used in this study were only measured on a weekly basis, which may make it difficult to capture the dynamics of the algal growth process. The timing of the sampling is also important for species such as *Anabaena*, which are able to regulate their buoyancy during calm, stratified conditions. This is only an issue if integrated depth samples are not collected. Cyanobacteria in the water column exhibit diurnal variation and tend to accumulate near the surface overnight, thereby resulting in an over-estimation for samples collected in the early morning and under-estimation for samples collected at other times. During the timespan of the data collected in this study, the samples were taken early in the morning.

Errors can occur in the cell counting process due to a bias in technique (systematic error), which affects the accuracy of the results, and from random sources, which affects both the accuracy and the precision of the data (Jones et al., 2003). Random error results from the spatial distribution of cells or colonies of cells (trichomes) in a population and therefore, can never be eliminated from the counting process. As a result of this, cell counts should be expressed in terms of their variability. Jones et al. (2003) describe the counting precision as "... an indication of chance variability about the observed mean value when repeated measurements (counts) are made". Since the counting precision does not take into account other forms of error (e.g. unrepresentative sampling), it can be interpreted as the best possible error range for a given count. In reality, the overall error is usually higher than indicated by the counting precision.

The factors that affect the counting precision include: the number of organisms counted; their spatial distribution in the counting chamber; and the variability of cells within a colony or trichome of the population. A precision of ±30% or better is required in order to detect a cell doubling in subsequent samples as a statistically significant change (Jones et al., 2003). Laslett et al. (1997) defined the counting precision mathematically as the ratio of the standard error to the mean (expressed as a percentage) for replicated counts and assumed a Poisson distribution of counting units (cells, colonies or trichomes) in the counting chamber. In the simple formula derived by Laslett et al. (1997), the precision (counting error) can be calculated from the total number of units counted (N) using the following equation:

$$\text{Counting error } (\pm\,\%) = 100\sqrt{\frac{2}{N}} \tag{5.3}$$

Using the equation for counting error, the numbers of trichomes or colonies counted for a required error are given in Table 5.5. To achieve a linear increase in the counting precision, the number of trichomes that need to be counted (i.e. counting effort) increases logarithmically (Figure 5.4). The increased counting effort is also associated with an increased cost. From Figure 5.4, it is apparent that there is a strong "law of diminishing returns" for increased counting effort beyond approximately 50 trichomes.

During the period of the data collected in this study, the number of trichomes counted did not remain constant, and, hence, the counting precision varied. The precision of the cell count data used in this study ranged from ±20% to as high as ±50% (Baker, 2003 *pers. comm.*).

Table 5.5 Minimum Counting Errors Based on the Number of *Anabaena* Trichomes Counted (source: Laslett et al., 1997)

Total trichomes counted	Counting error (± %)
1	140
2	100
4	75
8	50
13	40
23	30
50	20
200	10

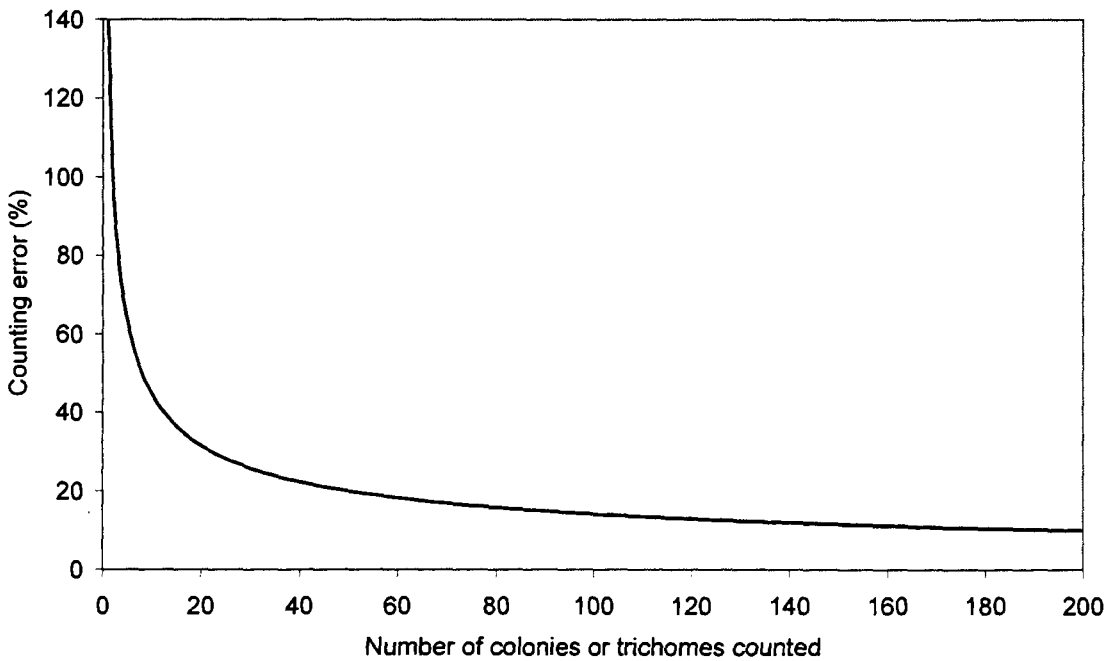

Figure 5.4 Relationship Between Counting Effort and Counting Error (Precision) For Trichomes of *Anabaena* (source: Laslett et al., 1997)

To gain familiarity with the data under investigation in this study, time series plots of the available data were inspected. Each of the variables listed in Table 5.4 is plotted and discussed below.

5.3.1 *Anabaena* spp.

A plot of the time series of *Anabaena* spp. at Morgan from 08-01-1980 to 20-11-1996 is shown in Figure 5.5. With the exception of 1983, 1984 and 1985, a definite seasonal variation is apparent, with growth events predominantly occurring in the summer of

each year. The absence of *Anabaena* spp. from 1983 to 1985 corresponded with very high and persistent turbidity levels (Figure 5.17). A linear regression with time was performed using the *Anabaena* spp. data and is shown in Figure 5.5. An increase in the frequency and duration of the growth events in the latter half of the period is reflected by an increase in the average *Anabaena* spp. concentration of approximately 4.0 cells/mL per year. To determine if a trend was present in the data, the slope of the regression equation was tested for statistical significance. The slope coefficient was divided by the estimated standard error of the slope coefficient to give an observed *t*-value of 3.73. For a one-sided test, with 876 degrees of freedom, the critical *t*-value at the $\alpha = 0.05$ significance level is 1.65. Since the absolute value of *t*-observed is greater than *t*-critical, the slope can be considered statistically significant and hence, a trend is present in the data.

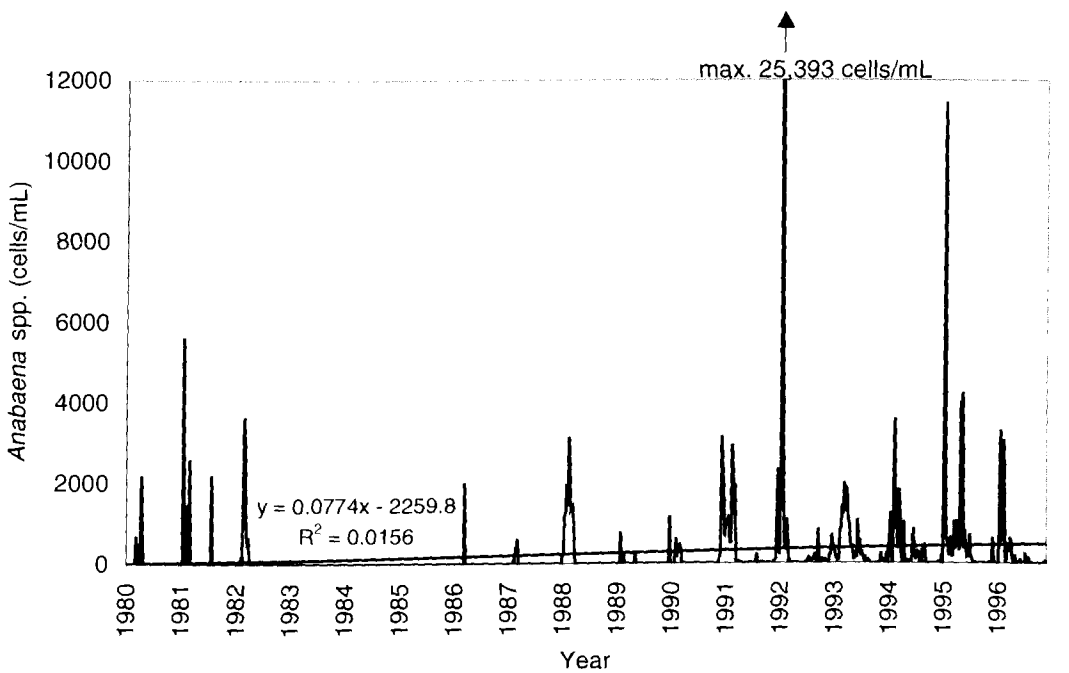

Figure 5.5 Concentrations of *Anabaena* spp. at Morgan (1980 to 1996)

A nationally accepted system of managerial response levels known as the National Alert Levels for Cyanobacteria in Drinking Water has been developed by Michael Burch of the Australian Water Quality Centre. The alert levels are given in Table 5.6 and show the cell density range for each level and the associated actions that are required to be taken by water treatment plant operators and managers. The time series of *Anabaena* spp. at Morgan and the Alert Levels are shown in Figure 5.6. During the time span considered in this study, Alert Level 1 was only triggered 77 times, Alert Level 2 was triggered 23 times and Alert Level 3 was only reached once during the 17

years (Figure 5.7). For the majority of this period (779 records), the concentration of *Anabaena* spp. was less than 500 cells/mL.

Table 5.6 ARMCANZ National Cyanobacterial Managerial Alert Level System For Drinking Water Supplies (source: Jones et al., 2003)

Alert Level	Cell Range (cells/mL)	General Actions
1	500 - 2,000	• increase monitoring frequency
2	2,000 - 15,000	• commence toxin testing • implement advanced water treatment • commence health risk assessment
3	>15,000	• notify health authorities • media releases • reach assessment regarding health risks – consider withdrawal of water from supply if no treatment for toxin removal

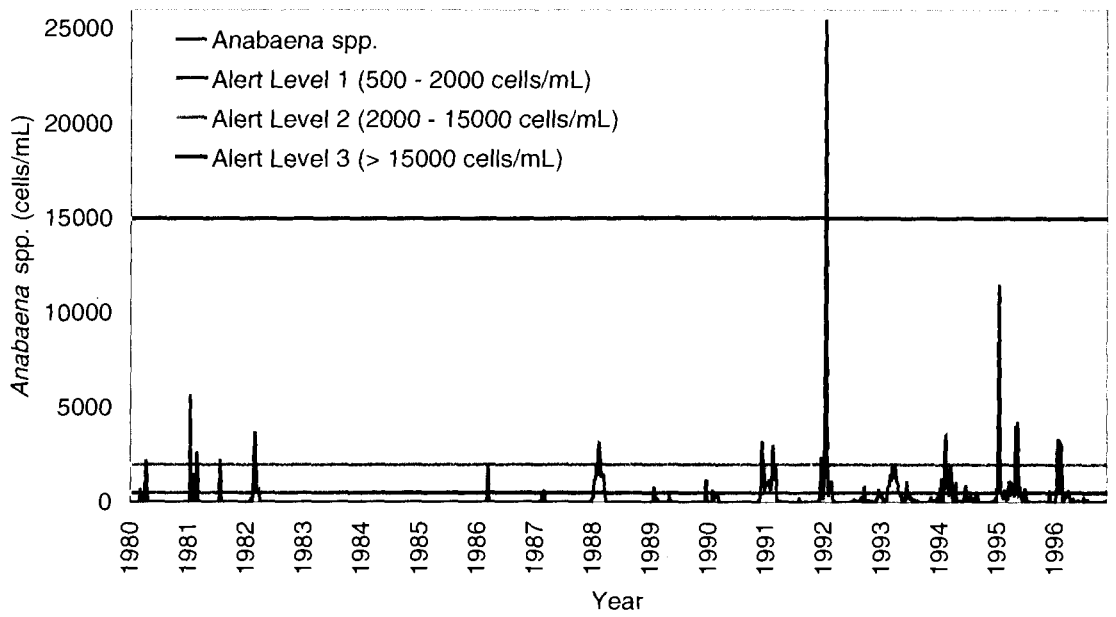

Figure 5.6 Concentrations of *Anabaena* spp. at Morgan (1980 to 1996) and the Alert Levels

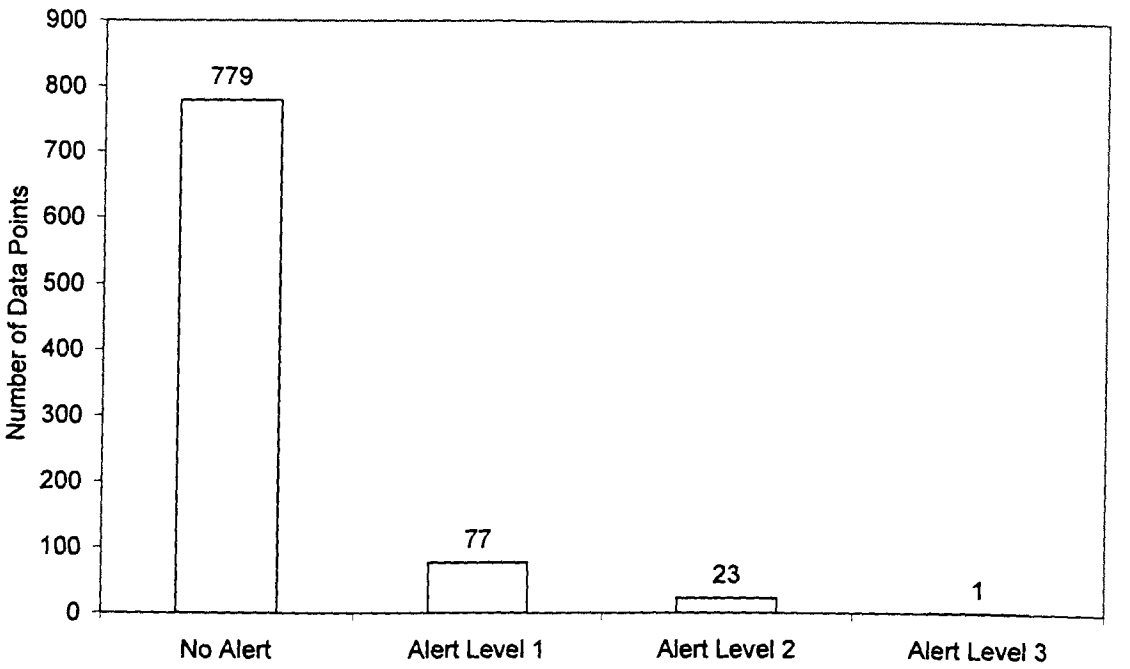

Figure 5.7 Number of Records in Each Alert Category For *Anabaena* spp. at Morgan (1980 to 1996)

As a preprocessing step, a logarithmic transformation of the *Anabaena* spp. data was performed (Figure 5.8). Such a transformation is commonly used in microbiological studies and is justified by the skewness of the data and by the potential for exponential growth and decay of cyanobacteria. A histogram plot of the raw *Anabaena* spp. data (i.e. before the logarithmic transformation) is shown in Figure 5.9. There are a large number of values in the first bin due to the large number of zero (absent) values recorded. The zero values were transformed by adding a small constant before taking the logarithm. After taking a logarithmic transformation, the distribution of the non-zero *Anabaena* spp. concentrations became more evenly spread over the range (Figure 5.10). Removing all zero values from the data set and plotting the Gaussian expectation for the non-zero values revealed that these data were also more normally distributed after being logarithmically transformed (Figure 5.11).

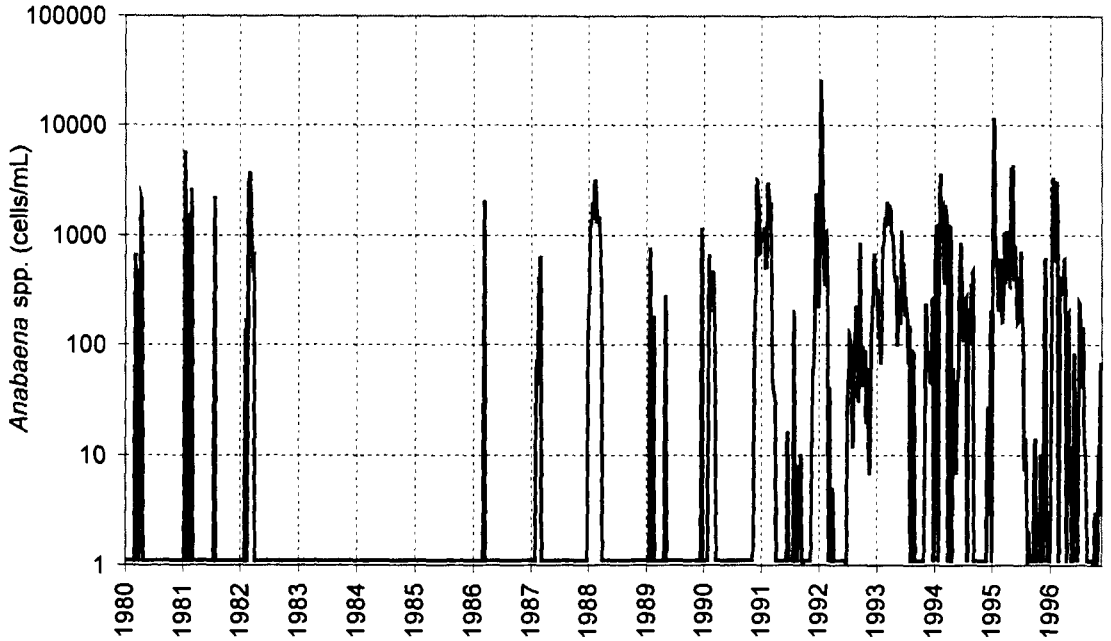

Figure 5.8 Concentrations of *Anabaena* spp. at Morgan on a Log Scale (1980 to 1996)

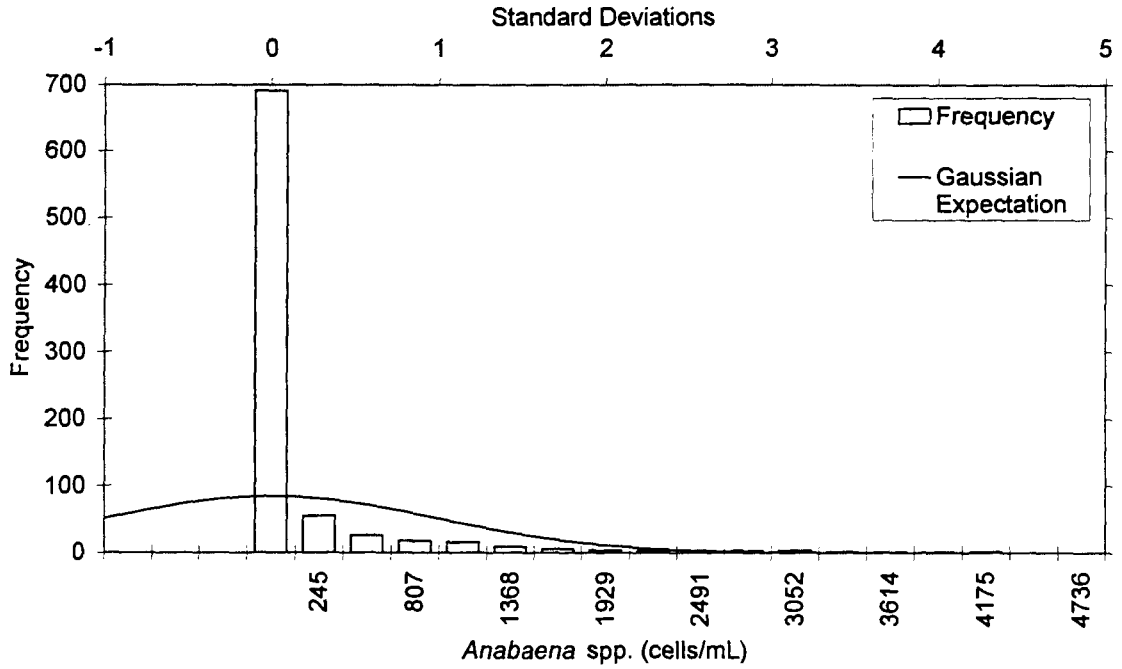

Figure 5.9 Histogram of *Anabaena* spp. at Morgan (Raw Data)

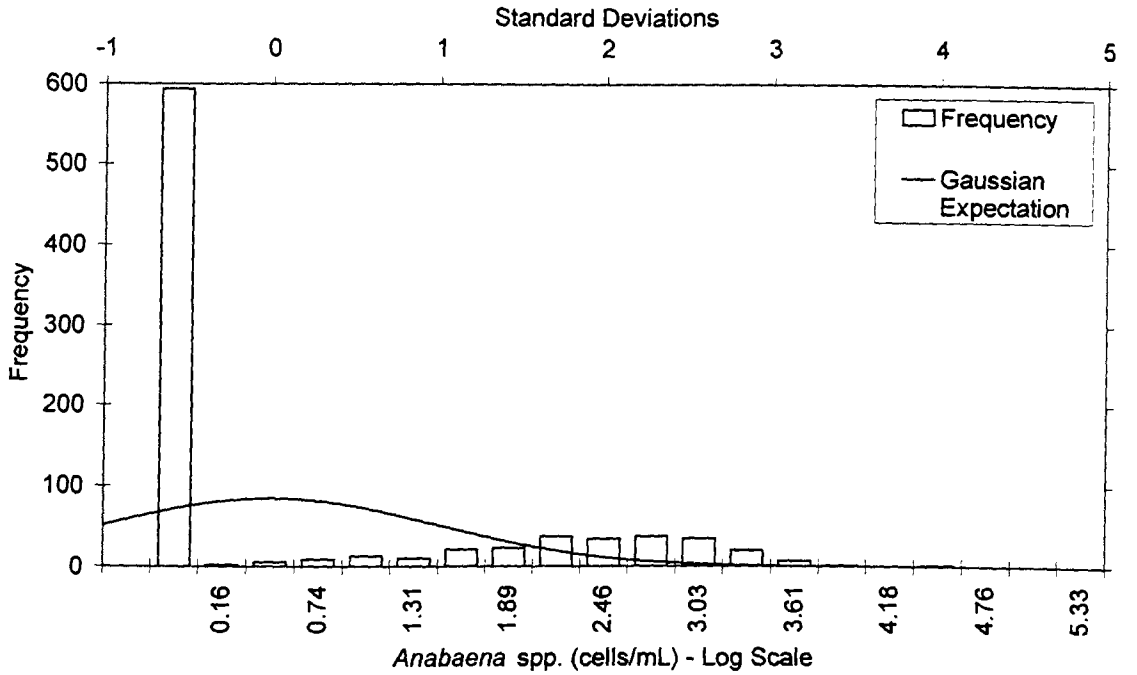

Figure 5.10 Histogram of *Anabaena* spp. at Morgan (Log Transformed)

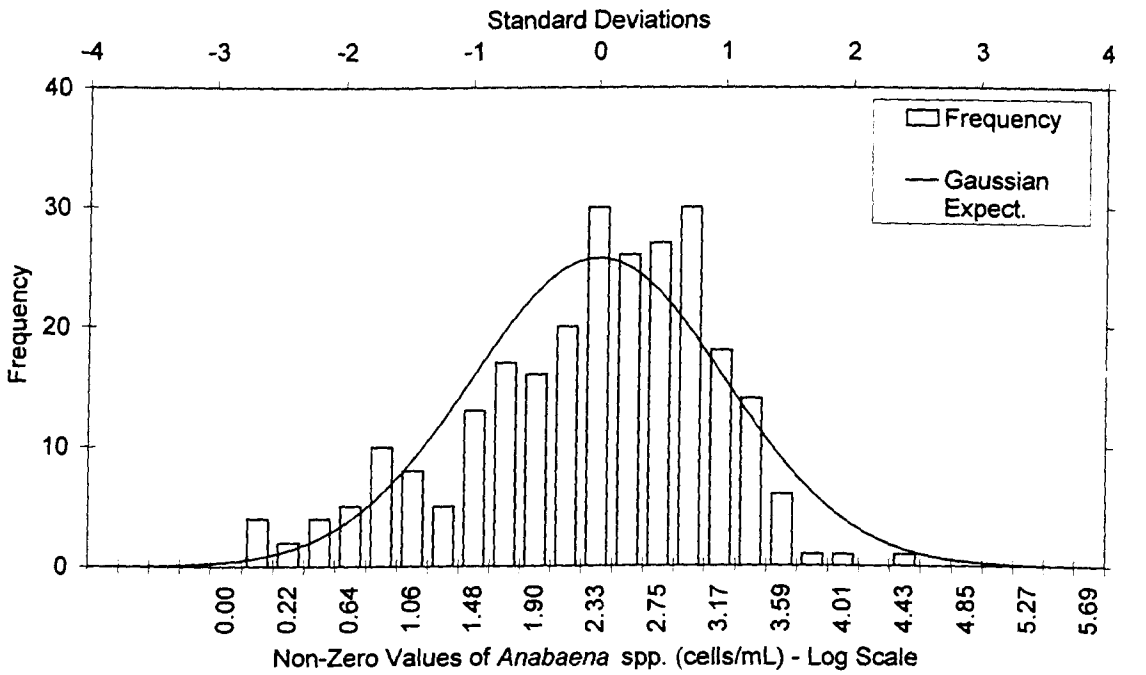

Figure 5.11 Histogram of the Non-Zero Values of *Anabaena* spp. at Morgan (Log Transformed)

5.3.2 Flow

A plot of the time series of *Anabaena* spp. at Morgan and flow into SA is shown in Figure 5.12. It is evident that there is a strong inverse relationship between *Anabaena* and flow. Persistent low flow conditions favour the development of thermal stratification and the growth of cyanobacteria, as discussed in Section 5.2.1.1. The average flow during significant growth events of *Anabaena* spp. was 10700 ML/day. A significant incidence of *Anabaena* spp. is defined in accordance with the Alert Levels Framework (Table 5.6) as a concentration greater than 500 cells/mL. During some years (e.g. 1990/1991, 1991/1992, 1992/1993, 1993/1994 and 1994/1995), relatively high concentrations of *Anabaena* spp. occurred on the falling limb of the flood hydrograph. This is believed to be caused by populations of cyanobacteria being advected downstream from shallow productive lagoons that have a hydraulic connection to the main river channel (Baker et al., 2000). As the flow recedes, the wetlands drain back into the river channel, thereby increasing the cyanobacterial load and providing the inoculum for populations to grow in the river.

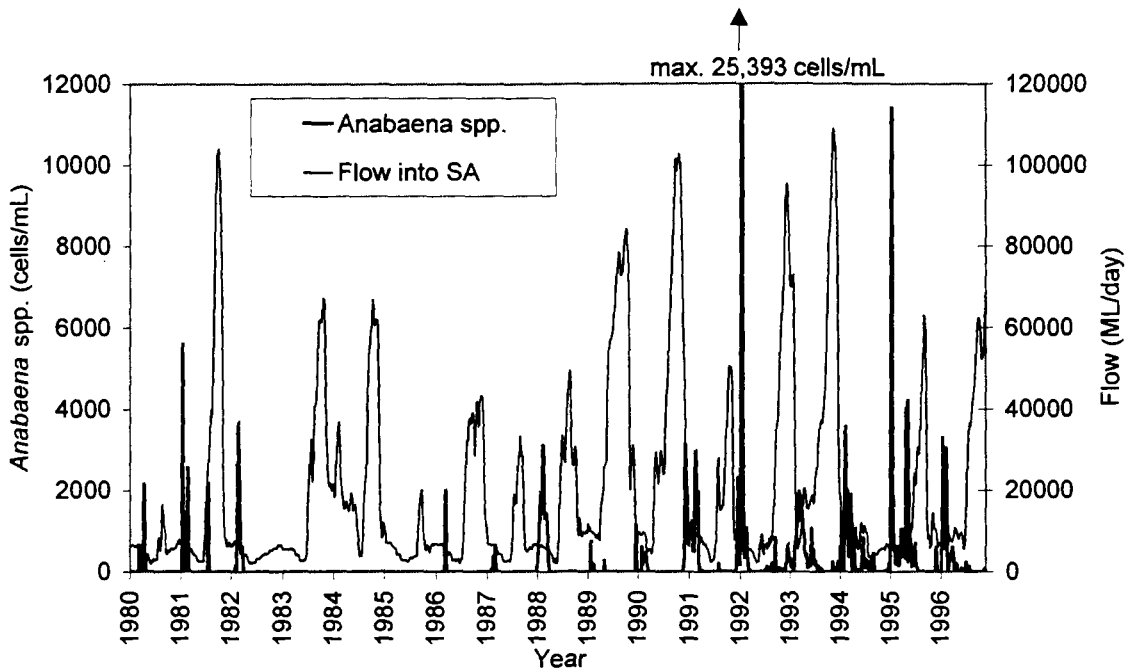

Figure 5.12 Concentrations of *Anabaena* spp. at Morgan and Flow into SA (1980 to 1996)

5.3.3 River Level

A plot of *Anabaena* spp. and the river level time series is shown in Figure 5.13. River level is highly correlated with flow (Figure 5.14), and therefore, also exhibits a strong

inverse relationship with *Anabaena* spp. cell densities. Significant growth events tended to occur when the river level was close to its minimum.

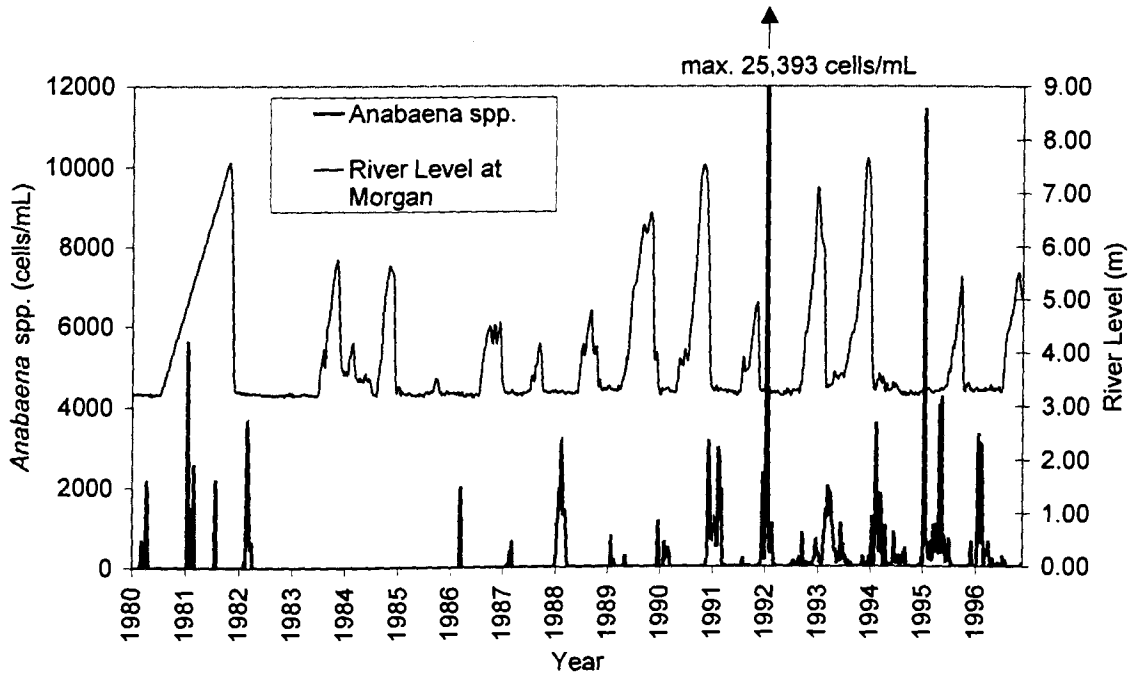

Figure 5.13 Concentrations of *Anabaena* spp. and River Level at Morgan (1980 to 1996)

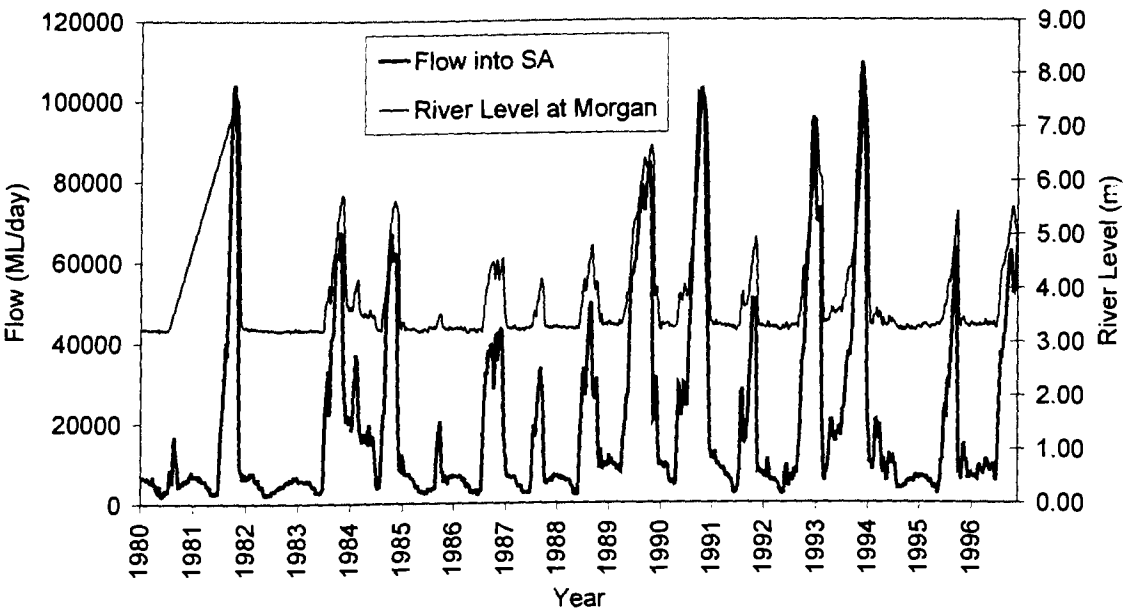

Figure 5.14 Flow into SA and River Level at Morgan (1980 to 1996)

5.3.4 Temperature

The *Anabaena* spp. and temperature time series are plotted in Figure 5.15. As expected, temperature is positively correlated with *Anabaena*. Higher temperatures can be associated with thermal stratification and also increase the growth rate of algae. In Figure 5.15, it is apparent that most incidences of *Anabaena* spp. occurred when the water temperature was greater than 20°C. Significant populations usually coincided with temperatures greater than 25°C. The most notable exceptions to this general observation were the growth events that occurred during 1981 and 1995. It should be noted that the annual temperature cycle approximately follows the cycle of incoming solar radiation and hence, temperature may also be a surrogate for this important variable.

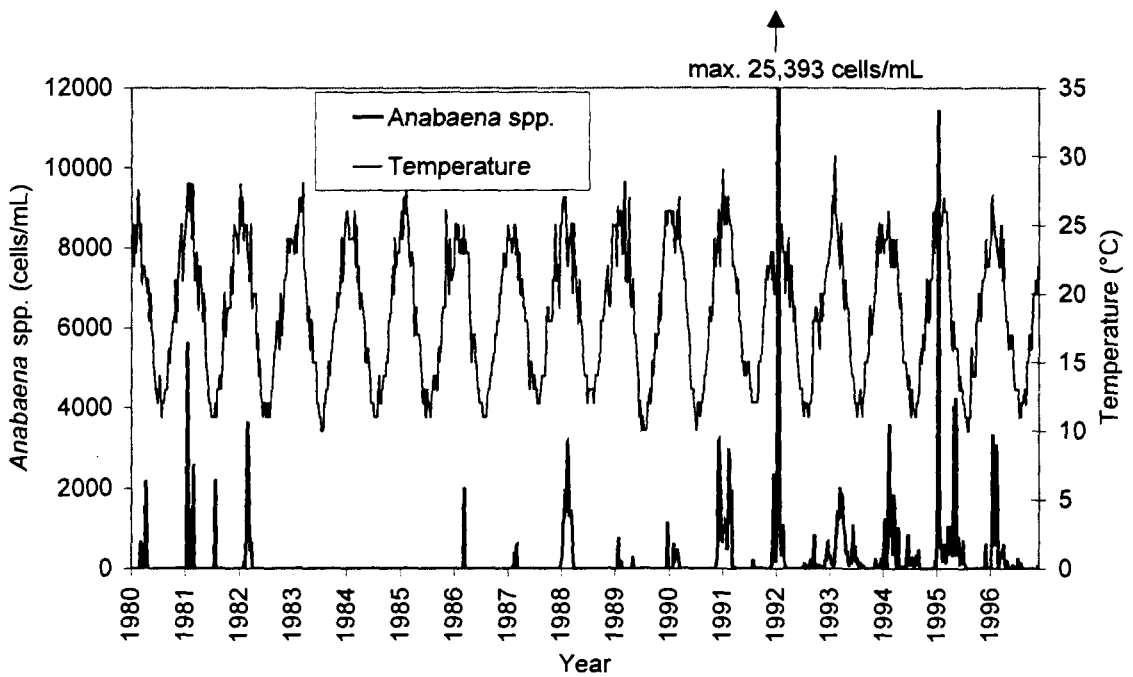

Figure 5.15 Concentrations of *Anabaena* spp. and Temperature at Morgan (1980 to 1996)

5.3.5 Colour

A plot showing the variation of *Anabaena* spp. and colour at Morgan is shown in Figure 5.16. An inverse relationship is apparent between *Anabaena* and colour. This is expected, since colour is a measure of the optical properties of the water. High levels of colour reduce the amount of sunlight that can penetrate the water column and hence, the growth of cyanobacteria. Colour is strongly correlated with flow and significant populations of *Anabaena* spp. tended to occur when the colour was less than 20 HU.

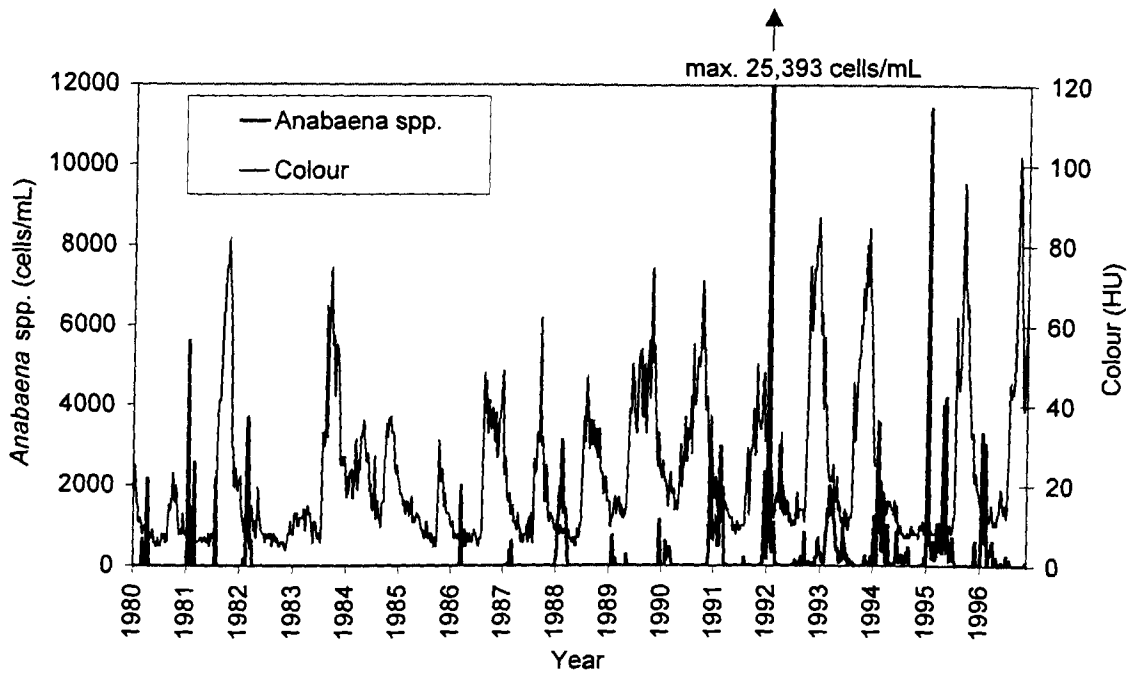

Figure 5.16 Concentrations of *Anabaena* spp. and Colour at Morgan (1980 to 1996)

5.3.6 Turbidity

A plot showing the variation of *Anabaena* spp. and turbidity at Morgan is shown in Figure 5.17. Turbidity is negatively correlated with *Anabaena*. Like high levels of colour, high turbidity levels reduce the amount of sunlight that can penetrate the water column and therefore, reduce the cyanobacterial growth rate. The turbidity levels during 1983, 1984 and 1985 were very high and this appears to have prevented the growth of *Anabaena*. During these years, most of the water entering the South Australian reaches of the River Murray was sourced from the Darling River, which is considerably more turbid than water which is from the upper reaches of the River Murray (Maier, 1995). In the 17-year period investigated in this study, populations of *Anabaena* spp. usually occurred at turbidity levels less than 50 NTU.

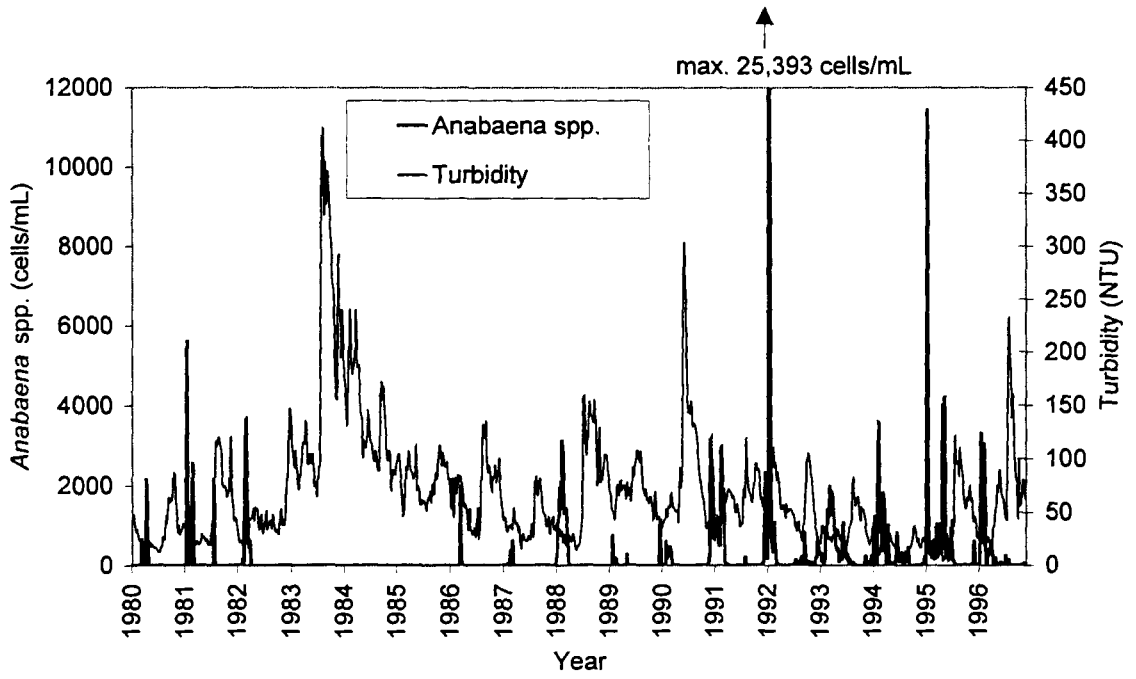

Figure 5.17 Concentrations of *Anabaena* spp. and Turbidity at Morgan (1980 to 1996)

5.3.7 pH

A time series plot of *Anabaena* spp. and pH at Morgan is shown in Figure 5.18. It can be seen that a positive correlation exists between *Anabaena* and pH. The pH is an important input variable as it can alter the composition of the phytoplankton community. Low pH (< 6.0) favours eukaryotes, whereas high pH (> 8.0) favours cyanobacteria. During the 17-year period, populations of *Anabaena* spp. tended to occur when the pH was high (> 8.0). The average pH during significant *Anabaena* spp. growth events was 8.2.

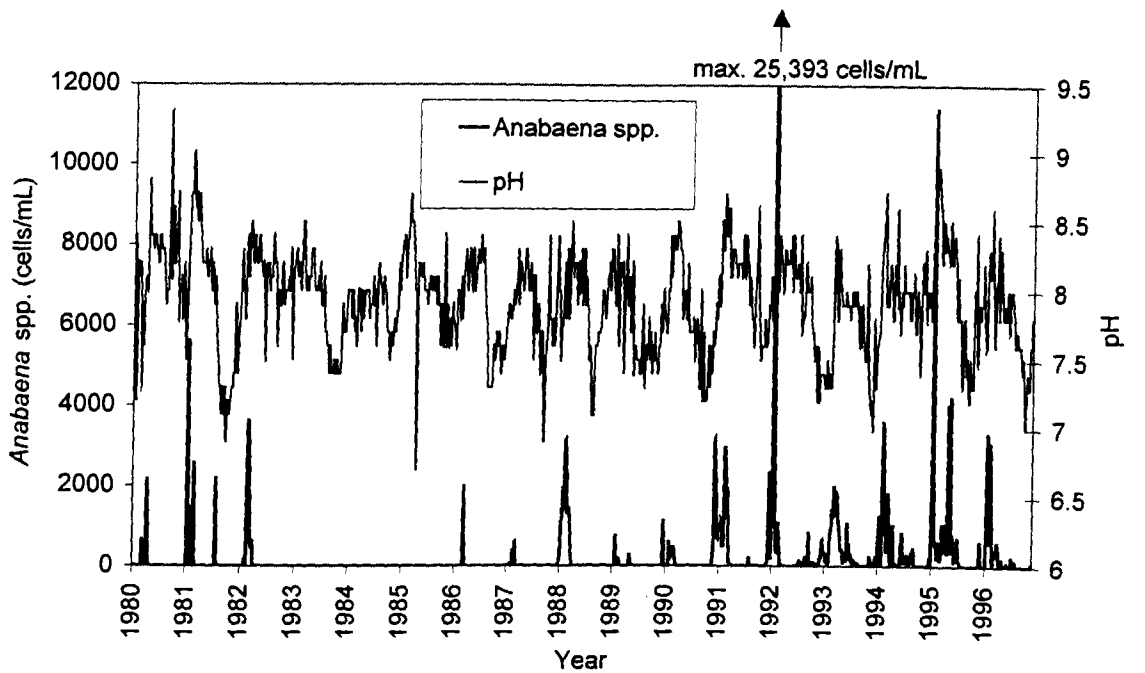

Figure 5.18 Concentrations of *Anabaena* spp. and pH at Morgan (1980 to 1996)

5.3.8 Silica

A plot showing the variation of *Anabaena* spp. and silica at Morgan is shown in Figure 5.19. It can be seen that the concentrations of silica and *Anabaena* spp. are inversely related. Silica is a micronutrient that is important in determining phytoplankton succession. Diatoms are dependent on silica for growth and when silica supplies become exhausted, cyanobacteria are favoured. The average concentration of silica during significant incidences of *Anabaena* spp. was only 2.0 mg/L. This is quite low compared to the average concentration over the 17-year period of 4.8 mg/L (Table 5.4).

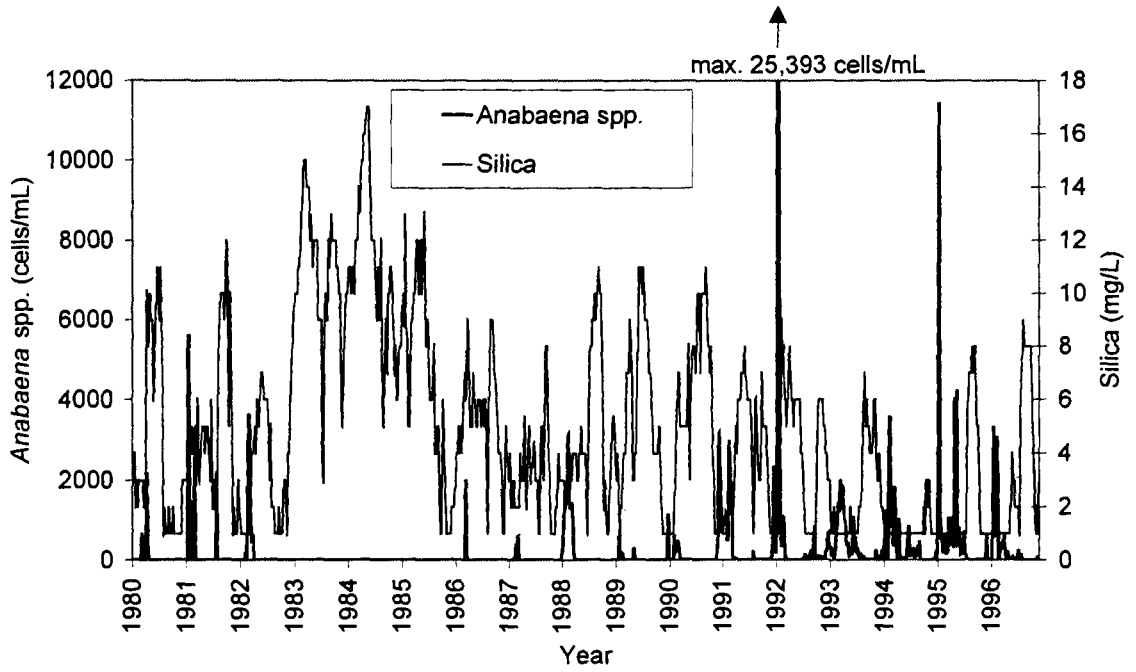

Figure 5.19 Concentrations of *Anabaena* spp. and Silica at Morgan (1980 to 1996)

5.3.9 TKN

A plot showing the variation of *Anabaena* spp. and TKN at Morgan is shown in Figure 5.20. Upon visual inspection, there does not appear to be any significant correlation between *Anabaena* and TKN. This may suggest that nitrogen levels were not limiting during this period. During times of low nitrogen availability, *Anabaena* have a competitive advantage over many other species of phytoplankton, since they are capable of fixing nitrogen from the atmosphere.

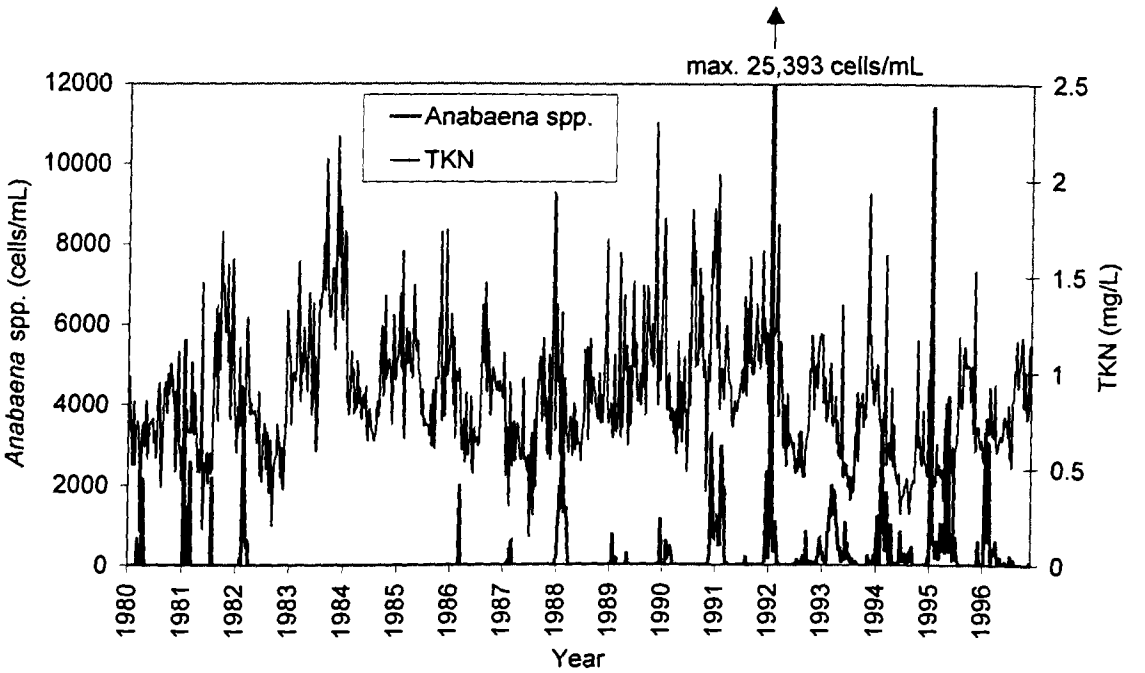

Figure 5.20 Concentrations of *Anabaena* spp. and TKN at Morgan (1980 to 1996)

5.3.10 Total Phosphorus

A plot showing the variation of *Anabaena* spp. and total phosphorus at Morgan is shown in Figure 5.21. It is apparent that an inverse relationship exists between concentrations of *Anabaena* spp. and total phosphorus. This is contrary to expectation, since phosphorus is an important nutrient for phytoplankton growth to occur. However, total phosphorus concentrations are highly correlated with turbidity (Figure 5.22). This is because phosphorus is generally bound to the clay particles responsible for turbidity (Maier, 1995). Therefore, the effect of turbidity on *Anabaena* populations may be more significant than any effect caused by a reduced availability of phosphorus. This observation tends to suggest that phosphorus concentrations may not be limiting in the lower reaches of the River Murray.

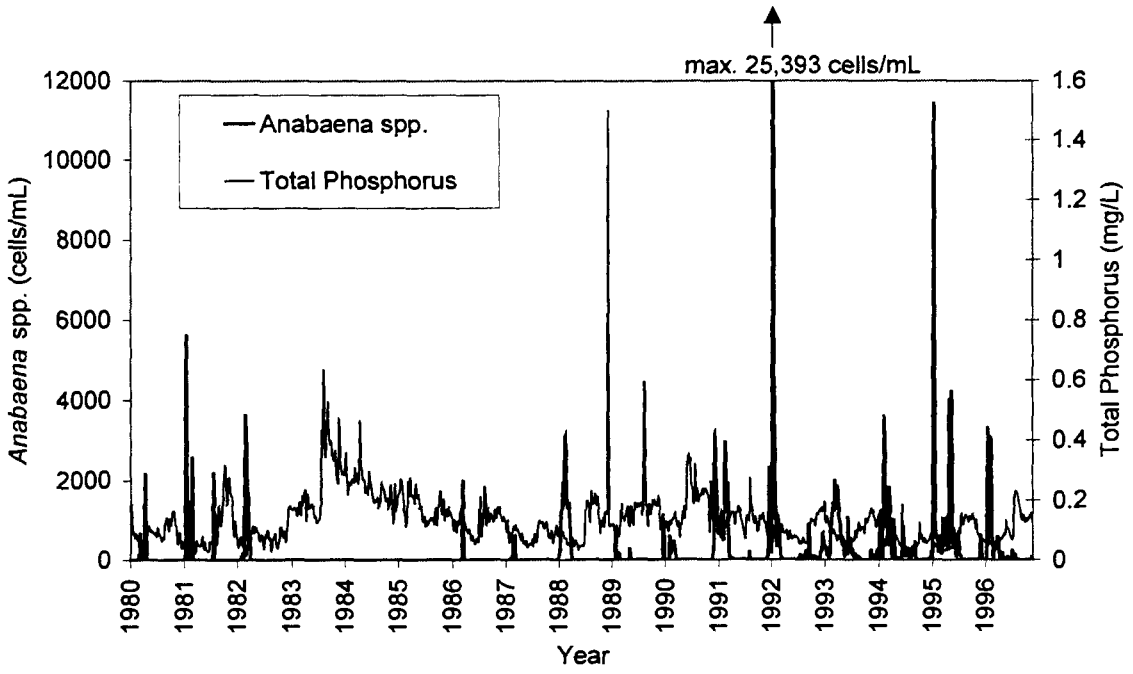

Figure 5.21 Concentrations of *Anabaena* spp. and Total Phosphorus at Morgan (1980 to 1996)

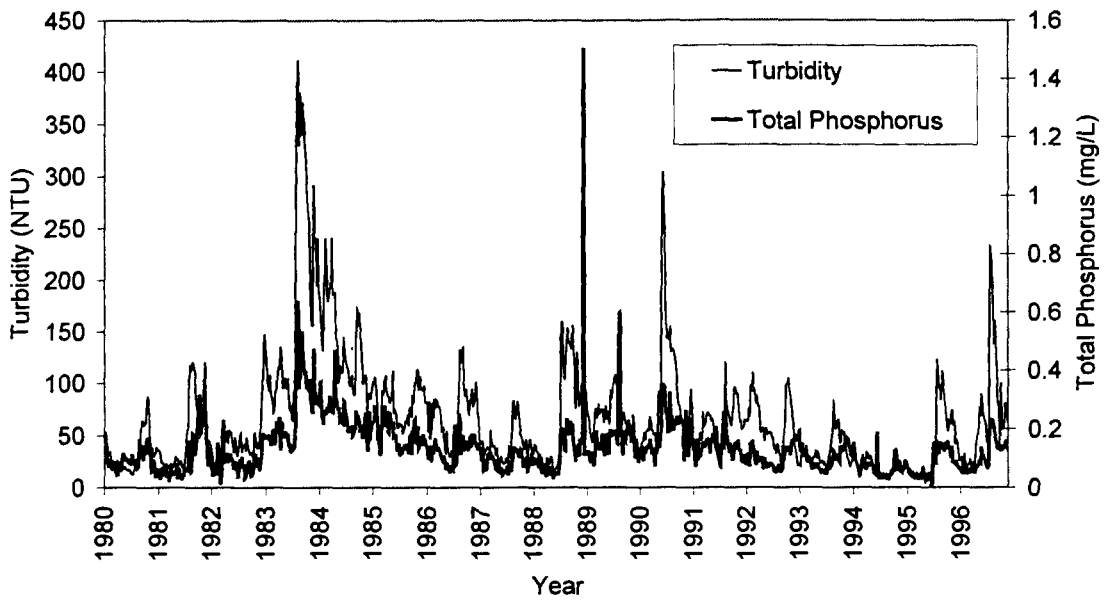

Figure 5.22 Concentrations of Total Phosphorus and Turbidity at Morgan (1980 to 1996)

5.3.11 Soluble Phosphorus

A plot showing the variation of *Anabaena* spp. and soluble phosphorus at Morgan is shown in Figure 5.23. *Anabaena* and soluble phosphorus appear to be negatively correlated. However, as discussed in Section 5.3.10, high levels of soluble phosphorus coincide with high turbidity levels and soluble phosphorus and turbidity have an opposing effect on *Anabaena* populations. During the period under investigation, turbidity appears to have been the dominant factor, reinforcing the fact that phosphorus may not be a limiting nutrient.

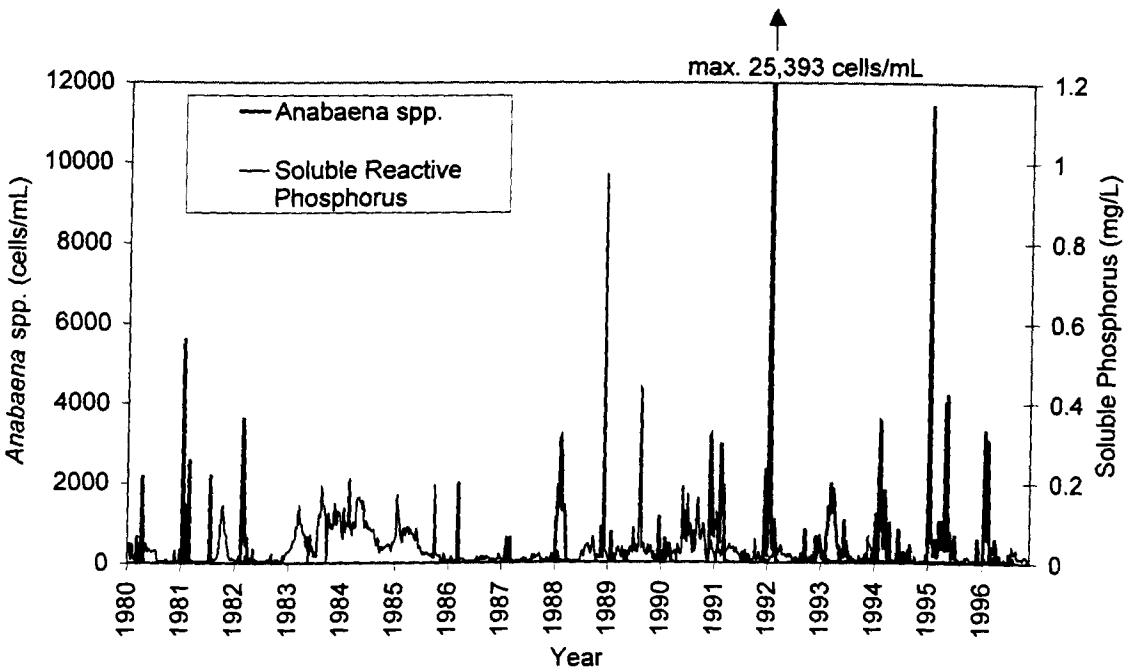

Figure 5.23 Concentrations of *Anabaena* spp. and Soluble Phosphorus at Morgan (1980 to 1996)

5.3.12 Wind Run

A time series plot of *Anabaena* spp. at Morgan and wind run at Blanchetown is shown in Figure 5.24. Blanchetown is located approximately 40 km downstream of Morgan (Figure 4.3) and was the closest available station along the River Murray measuring wind run. From Figure 5.24, it is evident that the wind data contain a trend and heteroskedasticity (change in the variance). Upon investigation it was discovered that the bearings in the anemometer were not replaced or properly maintained and consequently, the friction in the apparatus increased over the period of operation, thus causing the observed trend and heteroskedasticity. This example highlights the need to

check the consistency of the data and to investigate any uncharacteristic traits. Due to these factors, the wind run data were not used in this study.

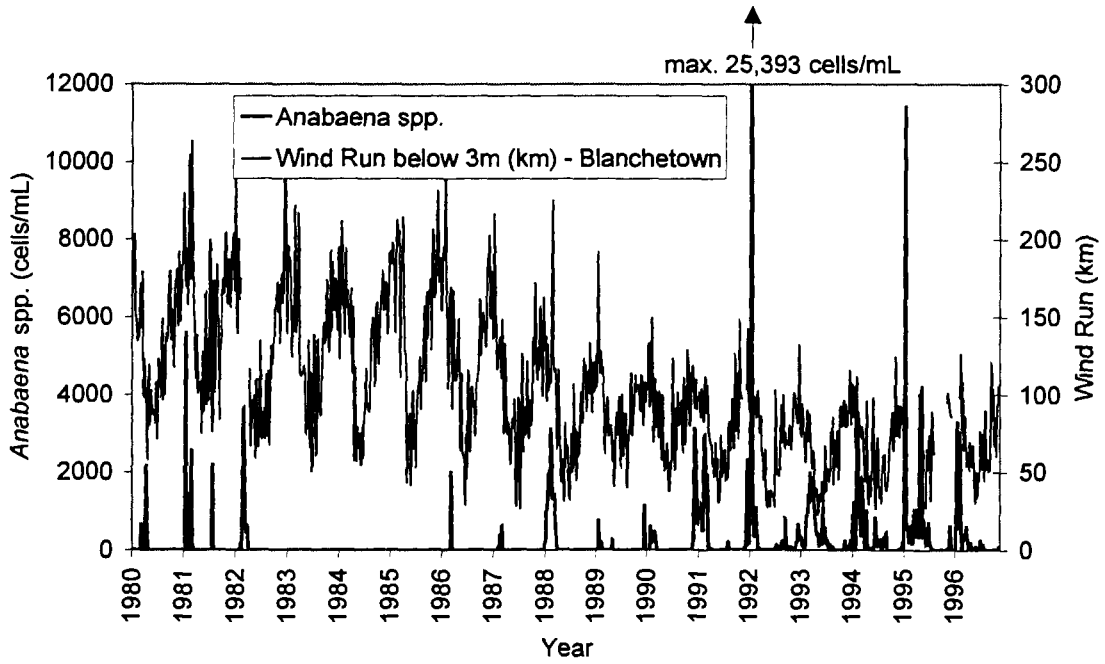

Figure 5.24 Concentrations of *Anabaena* spp. at Morgan and Wind Run Below 3m at Blanchetown (1980 to 1996)

5.4 ANN Model Development

The ANN methodology described in Chapter 3 was applied to the *Anabaena* spp. case study. The results from the numerical experiments conducted using the salinity case study (Chapter 4) were taken into consideration when applying the methodology to the *Anabaena* spp. case study. For the steps in the methodology that are not case study dependent, the method that provided the best results for the salinity data set was also used for the *Anabaena* spp. data set. For steps in the methodology that are case study dependent, the step was repeated using the *Anabaena* spp. data set. For example, the choice of an appropriate data transformation is dependent on the application under investigation. Therefore, both linear and histogram equalization transformations were tested for this case study.

A forecasting period of 4 weeks was adopted for the present study. This lead-time was identified by water treatment managers as the necessary time required to prepare suitable treatment processes, such as PAC dosing, in the case of an impending bloom. The onset, peak and duration of a bloom or growth event are the three most important characteristics describing the occurrence of *Anabaena* spp. Performance measures, such as the RMSE, are not ideal measures of fitness for this case study as they do not take into account the timing of a growth event. Even if the RMSEs of several forecasts are similar, the usefulness of the forecasts may differ. For example, two forecasts may have the same RMSE, but one may forecast increases that lead the actual event while the other may lag it, making the former more useful. Therefore, a visual inspection of the plots of actual and predicted results is important in addition to calculating the RMSE between them. In the absence of any better alternatives, the RMSE was adopted as the default performance measure as it places greater emphasis on the larger forecasting errors.

The ANN model development process adopted for the *Anabaena* spp. data set is summarised in Figure 5.25. After testing the data for evidence of nonlinearity (Step 1), the data are divided into training, testing and validation sets using a genetic algorithm (GA) (Step 2). Both linear and histogram equalization transformations are trialled in Step 3. Kernel transformation was not investigated as its objective is the same as histogram equalization and the two transformations produce similar results. Both the stepwise PMI algorithm and the SOM-GAGRNN input determination method are used in Step 4. In Step 5, both a GRNN and EBMLP model are developed for each combination of data transformation and input determination method, leading to the development of 8 ANN models. To determine the most suitable performance measure, the method of Diskin and Simon (1977) outlined in Section 3.7.1.3 was used in Step 6.

The 10 performance measures described in Section 3.7.1.2 were also investigated in this case study. The choice of the optimisation algorithm in Step 6 is dependent on whether the GRNN or MLP was found to give superior performance in Step 5. If the GRNN is found to outperform the MLP, then a multiple-sigma GRNN trained using ant colony optimisation (ACO) will be investigated as an alternative to the single-sigma GRNN trained using the inverse Hessian method. On the other hand, if the MLP is found to outperform the GRNN, then the generalized delta (GD) rule, the normalized cumulative delta (NCD) rule, the extended delta-bar-delta (EDBD) algorithm, the QuickProp (QProp) algorithm and the MaxProp (MProp) algorithm will be compared and contrasted as alternative methods for training an MLP for this case study. The model deployment stage in Step 7 is also dependent on the outcome of Step 5. In the case that the GRNN outperforms the MLP, then the GRNN outlier detector will be used for model deployment. Conversely, in the case that the MLP outperforms the GRNN, the hybrid SOM-MLP model will be used for deploying the final model.

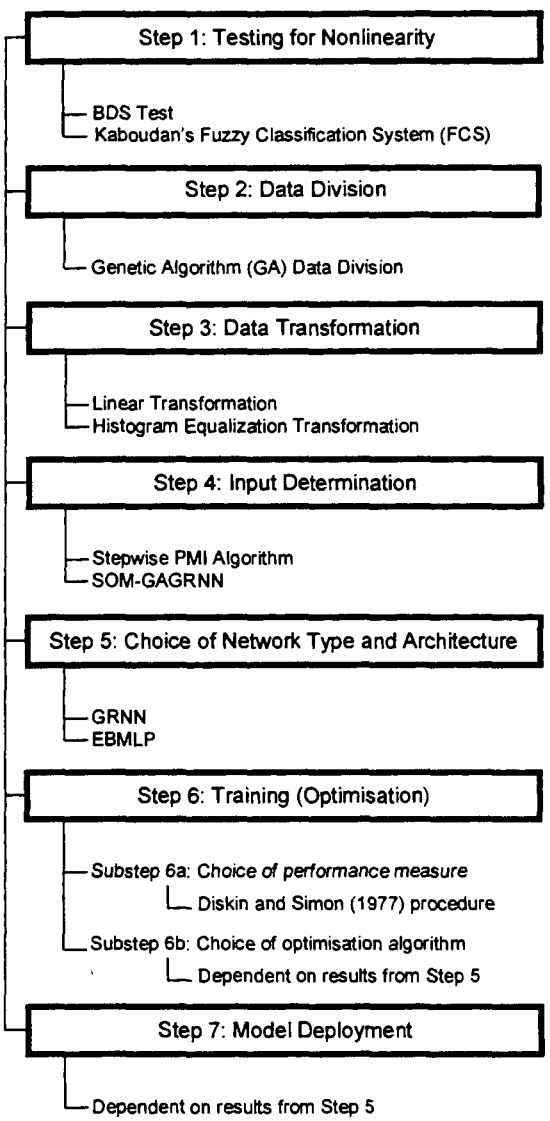

Figure 5.25 ANN Model Development Steps for the Anabaena spp. Case Study

A description of how each of the steps in the ANN modelling methodology were applied to the *Anabaena* spp. data set is given below, including the reasons for choosing the methods that were used. The results obtained for each step are also presented and discussed.

5.5 Testing For Nonlinearity

The nonlinearity tests described in Section 3.2 were applied to the log transformed *Anabaena* spp. time series at Morgan to gain insight into the nature of the data and, consequently, determine whether ANNs provide a suitable modelling methodology for these data. This data set consists of log transformed concentrations of the species complex of cyanobacteria, *Anabaena* spp., recorded weekly in the River Murray at Morgan, South Australia during the period 08-01-1980 to 20-11-1996. It was considered important to use both nonlinearity tests to determine if the results were in agreement. Kaboudan's FCS has an advantage over the BDS test as it is able to provide additional information about the time series structure.

5.5.1 Results and Discussion

5.5.1.1 BDS Test

Table 5.7 shows the results obtained when the BDS test was applied to the log transformed *Anabaena* spp. data set. Linearity was strongly rejected for this data set, suggesting that the log(*Anabaena* spp.) time series data have nonlinear serial dependence. This is not surprising given knowledge of the potential for exponential growth and decay of cyanobacteria. The inference was robust to the setting of an embedding dimension within the 2 to 8 range.

Table 5.7 BDS Test Z Statistics, Residuals from ARMA Fit to Log(*Anabaena* spp.) Data Set

Data Set	Fitted ARMA Order (p, q)	Embedding Dimension	BDS Z Statistic	Decision
Log(*Anabaena*) at Morgan	(1, 1)	2	15.33	Reject linearity (strongly)
		3	16.63	
		4	17.56	
		5	18.67	
		6	19.87	
		7	21.65	
		8	23.96	

5.5.1.2 Kaboudan's FCS

The results obtained when the FCS was applied to the log transformed *Anabaena* spp. data set are shown in Table 5.8. It can be seen that these results are in agreement with those obtained using the BDS test, however, some additional information has also been obtained. The FCS also diagnosed the nonlinear component of the data set but also identified a fairly linear component. It is interesting to note that the FCS did not identify any noise associated with the nonlinear component of this data set.

Table 5.8 Diagnosis Results of the FCS for the Log(*Anabaena* spp.) Data Set

Data Set	Fitted ARMA Order (p, q)	r^2	θ	Final Diagnosis
Log(*Anabaena*) at Morgan	(1, 1)	0.64	0.19	FL-NL[a]

[a]Fairly linear - nonlinear

5.5.2 Summary

When the BDS test was applied to the log(*Anabaena* spp.) data, these data were diagnosed as significantly nonlinear. When Kaboudan's FCS was applied to the log(*Anabaena* spp.) data, they were found to possess a fairly linear component and a nonlinear component. Since a multivariate model is being developed in this research, the results from the nonlinearity tests need to be interpreted carefully. The output time series (i.e. log(*Anabaena* spp.) at Morgan) was found to be nonlinear, and autoregressive (lagged) inputs of this time series are being used in the model. Therefore, an ANN model should be used in preference to a multivariate ARMA model because the underlying structure will need to be nonlinear, even with the addition of the other input variables.

5.6 Division of Data for ANN Models

In Chapter 4, two new methods for data division, namely the GA data division method and the SOM data division method, were evaluated using the salinity data. Both methods were found to be suitable for dividing data for ANN models. In addition, both methods represented an improvement over an arbitrary division of the data. The GA data division method has the disadvantage that the proportion of data assigned to the training, testing and validation sets must be chosen, whereas the SOM data division avoids this. However, the SOM data division method is slightly more difficult to implement as it requires the selection of the Kohonen layer (grid size) and the choice of which samples to discard from each cluster. In Chapter 4, it was found that the model developed using the data divided by the GA was more robust when presented with uncharacteristic data. For these reasons, the GA data division method was used to divide the *Anabaena* spp. data set into training, testing and validation sets.

In the *Anabaena* spp. case study, a total of 848 data records were available once appropriate lags had been taken. A lagging window of 26 weeks was chosen as it was deemed unlikely that the present concentrations of *Anabaena* spp. would be affected by conditions from more than six months earlier. From the 848 data records, 678 records (80%) were used for calibration and 170 records (20%) were used for validation. The 678 records in the calibration set were further divided into 542 training records (80%) and 136 testing records (20%).

5.6.1 Results and Discussion

The statistical parameters for the training, testing and validation sets are shown in Table 5.9. It can be seen that the statistics are in good agreement. Table 5.9 also shows that for each variable, the training set contains the maximum and minimum values. The only exception is the Morgan log(*Anabaena* spp.) input variable, in which the training set contains the minimum value but not the maximum value. Hypothesis tests were performed for the data sets obtained using the GA data division (Table 5.10). The *t*-test null hypotheses were not rejected for all variables at the 0.05 significance level. The *F*-test null hypotheses were only rejected at the 0.05 significance level for the two phosphorus variables. However, from Figure 5.21 and Figure 5.23, it is apparent that the total and soluble phosphorus time series both contain a large peak that occurred at the end of 1988. From Table 5.9, it is apparent that these peak values were 1.5 mg/L and 0.97 mg/L for total and soluble phosphorus, respectively. Both of these values were included in the training set, however, these values may possibly be outliers, since they are both more than 10 times the interquartile range above the third quartile. For

the phosphorus variables, the inclusion of these very large values in the training set was the most likely cause of the *F*-test being rejected for the testing and validation sets. With the exception of this one anomaly, the GA data division technique was able to produce three data sets (i.e. training, testing and validation sets) that were representative of the same statistical population.

Table 5.9 Statistics of the Training, Testing and Validation Data Sets (Data Divided Using the Genetic Algorithm)

Variable and Data Set	Mean	Standard Dev.	Max.	Min.	Inter-quartile Range (IQR)
Input Variable 1: Morgan log(*Anabaena* spp.) (cells/mL)					
Training	0.75	1.17	4.06	0.00	1.56
Testing	0.79	1.19	4.40	0.00	1.75
Validation	0.66	1.06	3.60	0.00	1.11
Input Variable 2: Temperature (° C)					
Training	18.6	5.0	30.0	10.0	9.0
Testing	18.7	4.6	28.0	10.0	8.0
Validation	18.5	5.0	28.0	10.0	9.0
Input Variable 3: Flow (ML/day)					
Training	21014	23637	108923	2128	22379
Testing	20164	23779	101393	2324	20742
Validation	19786	22841	100383	2212	19545
Input Variable 4: Colour (HU)					
Training	23.2	18.4	99.0	4.0	22.0
Testing	22.3	19.3	102.0	5.0	22.0
Validation	21.7	16.2	81.0	5.0	19.8
Input Variable 5: Turbidity (NTU)					
Training	70.9	56.0	380.0	6.6	55.8
Testing	70.0	60.6	410.0	10.0	58.8
Validation	66.3	53.9	400.0	4.4	56.0
Input Variable 6: pH					
Training	7.9	0.4	9.3	6.7	0.5
Testing	8.0	0.4	9.0	7.2	0.4
Validation	7.9	0.3	8.8	7.1	0.5

Continued.

Variable and Data Set	Mean	Standard Dev.	Max.	Min.	Inter-quartile Range (IQR)
Input Variable 7: Silica (mg/L)					
Training	4.9	3.6	17.0	1.0	6.0
Testing	4.8	3.5	16.5	1.0	6.0
Validation	4.6	3.9	15.0	1.0	5.4
Input Variable 8: TKN (mg/L)					
Training	0.89	0.32	2.22	0.05	0.37
Testing	0.91	0.35	2.28	0.34	0.42
Validation	0.85	0.33	2.03	0.21	0.41
Input Variable 9: Total Phosphorus (mg/L)					
Training	0.14	0.10	1.50	0.01	0.09
Testing	0.13	0.08	0.64	0.03	0.10
Validation	0.13	0.08	0.41	0.03	0.11
Input Variable 10: Soluble Phosphorus (mg/L)					
Training	0.03	0.06	0.97	0.00	0.03
Testing	0.03	0.03	0.16	0.00	0.03
Validation	0.03	0.04	0.21	0.00	0.03
Input Variable 11: River Level (m)					
Training	4.0	1.1	7.6	3.2	1.0
Testing	4.0	1.1	7.3	3.2	1.1
Validation	3.9	1.1	7.5	3.2	0.8
Output 1: Morgan log(*Anabaena* spp.) at (t + 4) weeks (cells/mL)					
Training	0.74	1.16	4.40	0.00	1.56
Testing	0.62	1.04	3.62	0.00	0.58
Validation	0.84	1.21	3.75	0.04	1.78

Table 5.10 Hypothesis Tests About a Difference Between the Means and Variances of the Testing and Validation Sets when Compared to the Training Set (Data Divided Using the Genetic Algorithm)

Variable and Data Set	*t*-value	*t*-crit. α=0.05	*t*-test H_o: $\mu_1 = \mu_2$	*F*-value	*F*-crit. α=0.05	*F*-test H_o: $\sigma_1^2 = \sigma_2^2$
Input Variable 1: Morgan log(*Anabaena* spp.) (cells/mL)						
Testing	0.36	1.96	don't reject	1.04	1.29	don't reject
Validation	0.93	1.96	don't reject	1.21	1.29	don't reject
Input Variable 2: Temperature (° C)						
Testing	0.20	1.96	don't reject	1.17	1.29	don't reject
Validation	0.27	1.96	don't reject	1.00	1.29	don't reject
Input Variable 3: Flow (ML/day)						
Testing	0.37	1.96	don't reject	1.01	1.29	don't reject
Validation	0.61	1.96	don't reject	1.07	1.29	don't reject
Input Variable 4: Colour (HU)						
Testing	0.47	1.96	don't reject	1.10	1.29	don't reject
Validation	1.05	1.96	don't reject	1.29	1.29	don't reject
Input Variable 5: Turbidity (NTU)						
Testing	0.16	1.96	don't reject	1.17	1.29	don't reject
Validation	0.96	1.96	don't reject	1.08	1.29	don't reject
Input Variable 6: pH						
Testing	1.85	1.96	don't reject	1.00	1.29	don't reject
Validation	0.99	1.96	don't reject	1.14	1.29	don't reject
Input Variable 7: Silica (mg/L)						
Testing	0.24	1.96	don't reject	1.09	1.29	don't reject
Validation	0.70	1.96	don't reject	1.16	1.29	don't reject
Input Variable 8: TKN (mg/L)						
Testing	0.36	1.96	don't reject	1.20	1.29	don't reject
Validation	1.38	1.96	don't reject	1.04	1.29	don't reject

Continued.

Variable and Data Set	*t*-value	*t*-crit. α=0.05	*t*-test H_o: $\mu_1 = \mu_2$	*F*-value	*F*-crit. α=0.05	*F*-test H_o: $\sigma_1^2 = \sigma_2^2$
Input Variable 9: Total Phosphorus (mg/L)						
Testing	0.65	1.96	don't reject	1.41	1.29	reject
Validation	1.33	1.96	don't reject	1.72	1.29	reject
Input Variable 10: Soluble Phosphorus (mg/L)						
Testing	1.49	1.96	don't reject	3.25	1.29	reject
Validation	0.42	1.96	don't reject	2.33	1.29	reject
Input Variable 11: River Level (m)						
Testing	0.50	1.96	don't reject	1.07	1.29	don't reject
Validation	0.63	1.96	don't reject	1.08	1.29	don't reject
Output 1: Morgan log(*Anabaena* spp.) at (t + 4) weeks (cells/mL)						
Testing	1.16	1.96	don't reject	1.23	1.29	don't reject
Validation	0.90	1.96	don't reject	1.09	1.29	don't reject

5.7 Data Transformation

In Chapter 4, the histogram equalization transformation was found to produce the best results for the salinity data set when measured using the validation set. However, when tested on the second validation data set, a linear transformation of the data was found to provide the best results, indicating that that this transform may be more robust when applied to uncharacteristic data. Since these results were somewhat inconclusive, and given the fact that the most suitable data transformation is a case study dependent attribute, it was decided to try both linear and histogram equalization transformations for the Morgan *Anabaena* spp. data set. It is important to note that the *Anabaena* spp. data were already logarithmically transformed as a preprocessing step (Section 5.3.1). This was conducted to compress the distribution of the *Anabaena* spp. data, thereby making it more suitable to model.

The BestFit statistical analysis software package (Palisade Corp., 1997) was used in an attempt to fit a distribution to each input variable. BestFit includes 28 distribution types, however, it was not possible to fit any of these distributions at an acceptable level of significance to any of the variables used in this case study.

Histogram plots for each variable used in this study are presented in Figure 5.26 to Figure 5.35. To provide a comparison with the normal distribution, the Gaussian expectation is also shown on each of the histogram plots. In Figure 5.26, it is apparent that the distribution of water temperature at Morgan approaches that of uniformity. Temperature data are commonly uniformly distributed, however, it is most likely that insufficient data were available to demonstrate this. The distribution of flow into South Australia (Figure 5.27) and river level at Morgan (Figure 5.35) are characteristic of these types of hydrological data and are both heavily skewed with an extended right hand tail. The distributions of colour (Figure 5.28), turbidity (Figure 5.29), silica (Figure 5.31), and soluble phosphorus (Figure 5.34) also exhibit positive skewness and have an extended right hand tail. The distribution of pH (Figure 5.30) is approximately Gaussian, however, there is a gap in the middle of the distribution. This occurred at a pH value of 7.9 and may possibly be due to 7.9 always being rounded to 8.0 by the person taking the reading. In Figure 5.32, the distribution of TKN is also approximately Gaussian, however, it does possess positive skewness. The distribution of total phosphorus (Figure 5.33) is non-Gaussian and positively skewed. The histogram of the log transformed *Anabaena* spp data is given in Figure 5.10 and was discussed in Section 5.3.1.

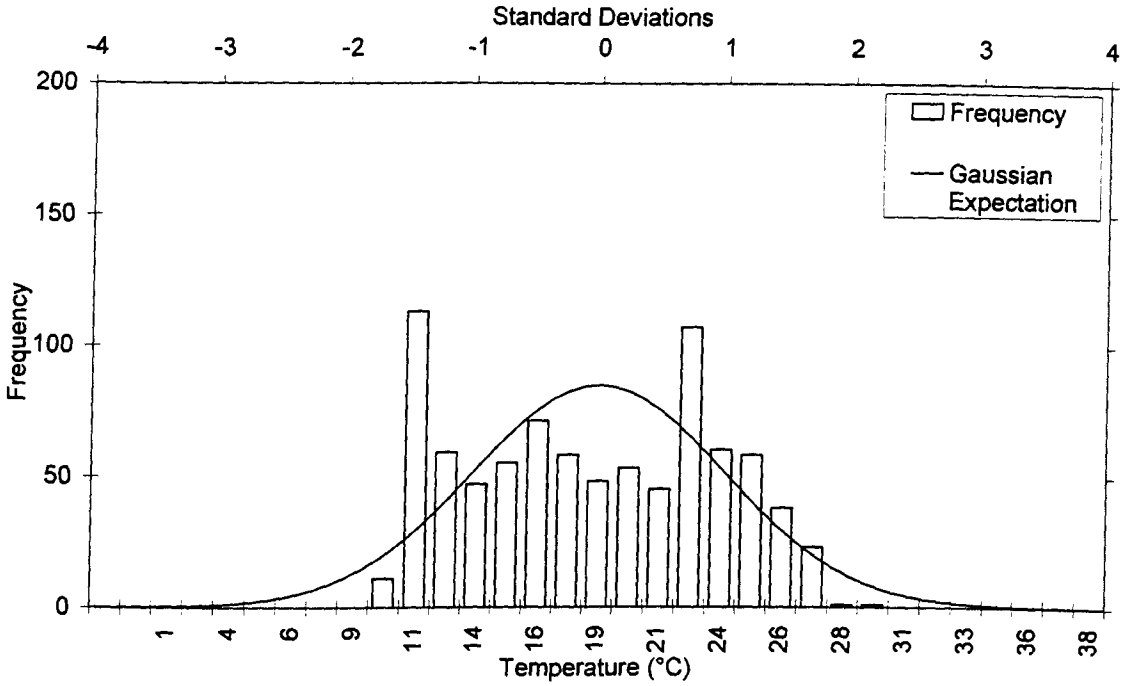

Figure 5.26 Histogram of Temperature at Morgan (Raw Data)

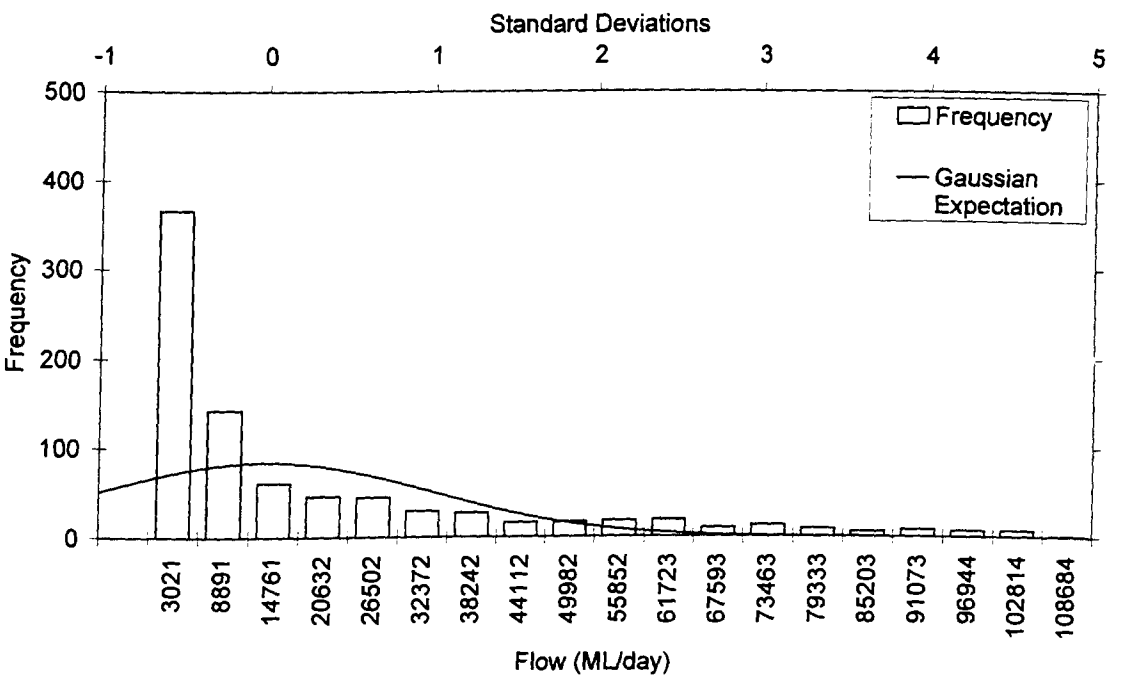

Figure 5.27 Histogram of Flow into SA (Raw Data)

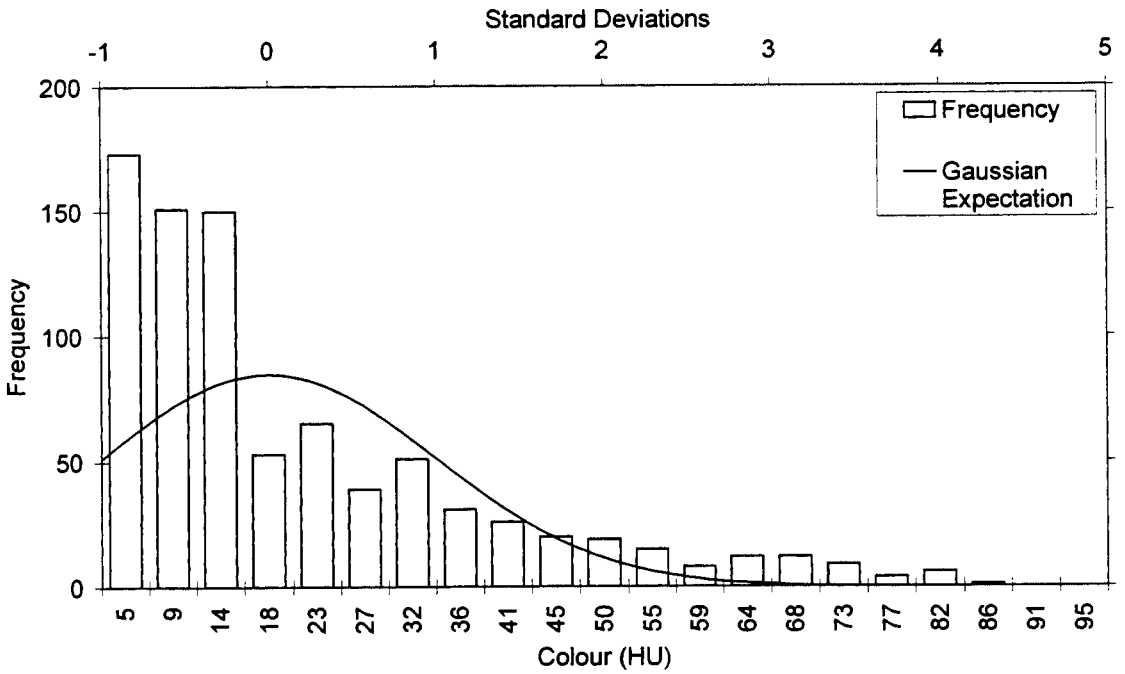

Figure 5.28 Histogram of Colour at Morgan (Raw Data)

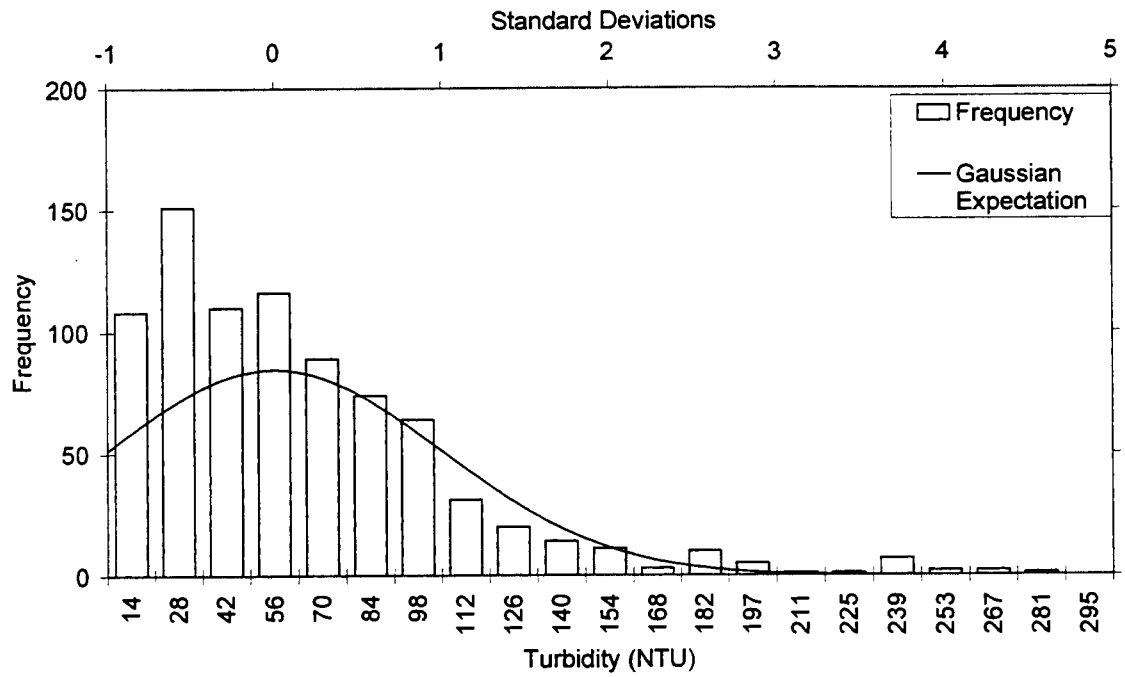

Figure 5.29 Histogram of Turbidity at Morgan (Raw Data)

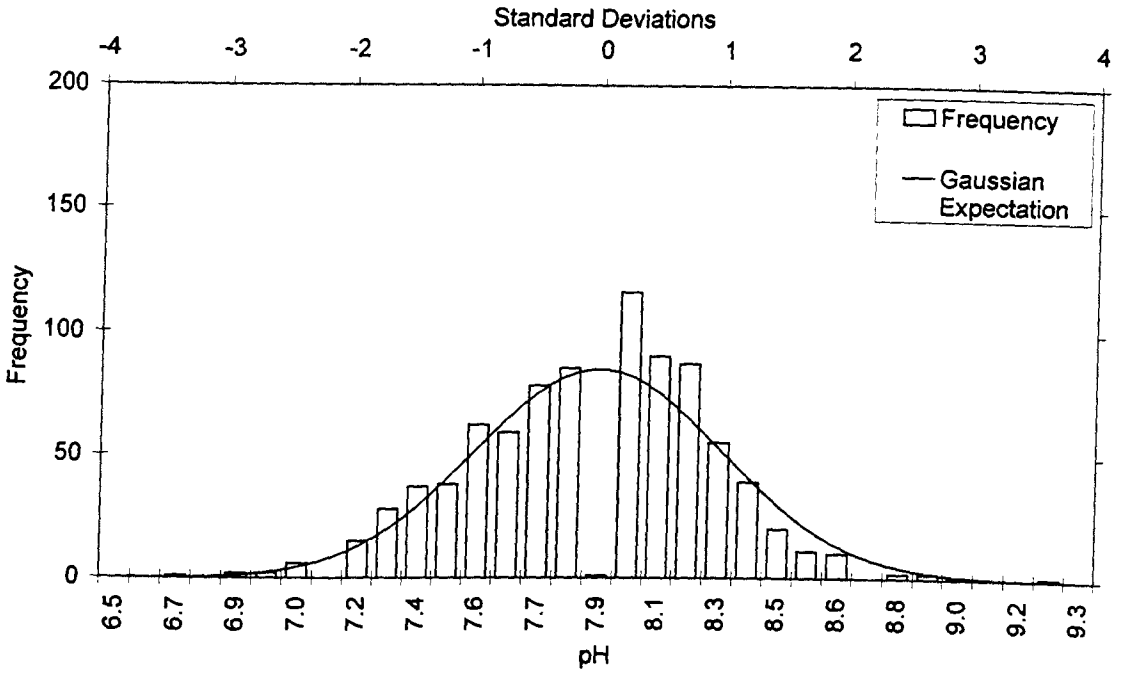

Figure 5.30 Histogram of pH at Morgan (Raw Data)

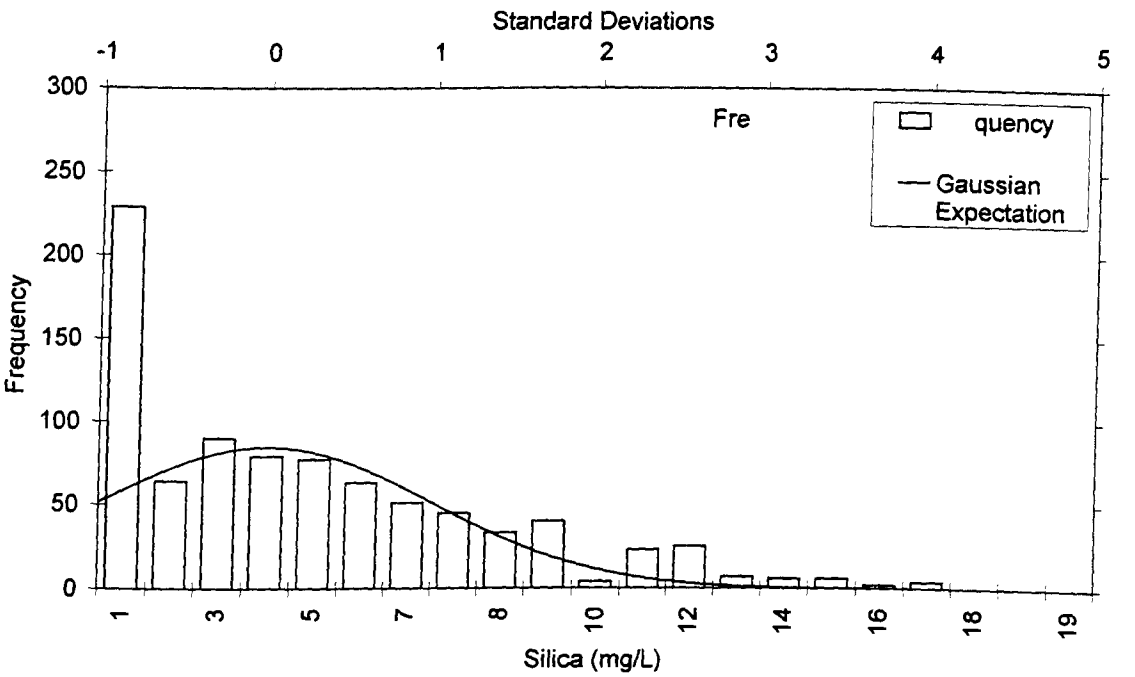

Figure 5.31 Histogram of Silica at Morgan (Raw Data)

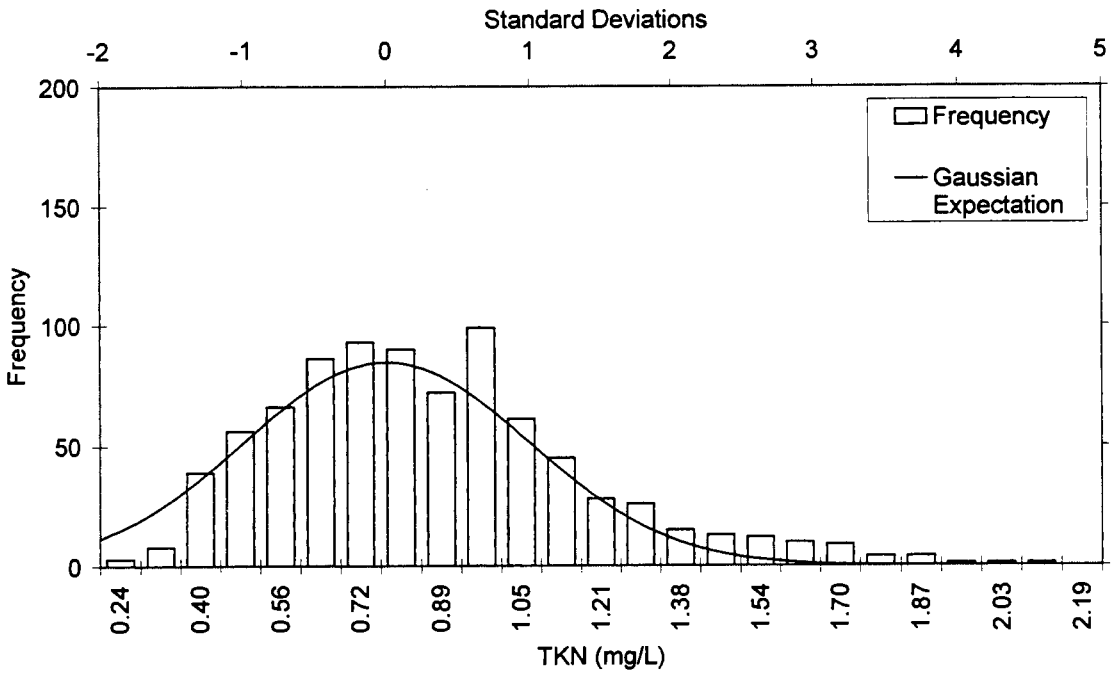

Figure 5.32 Histogram of TKN at Morgan (Raw Data)

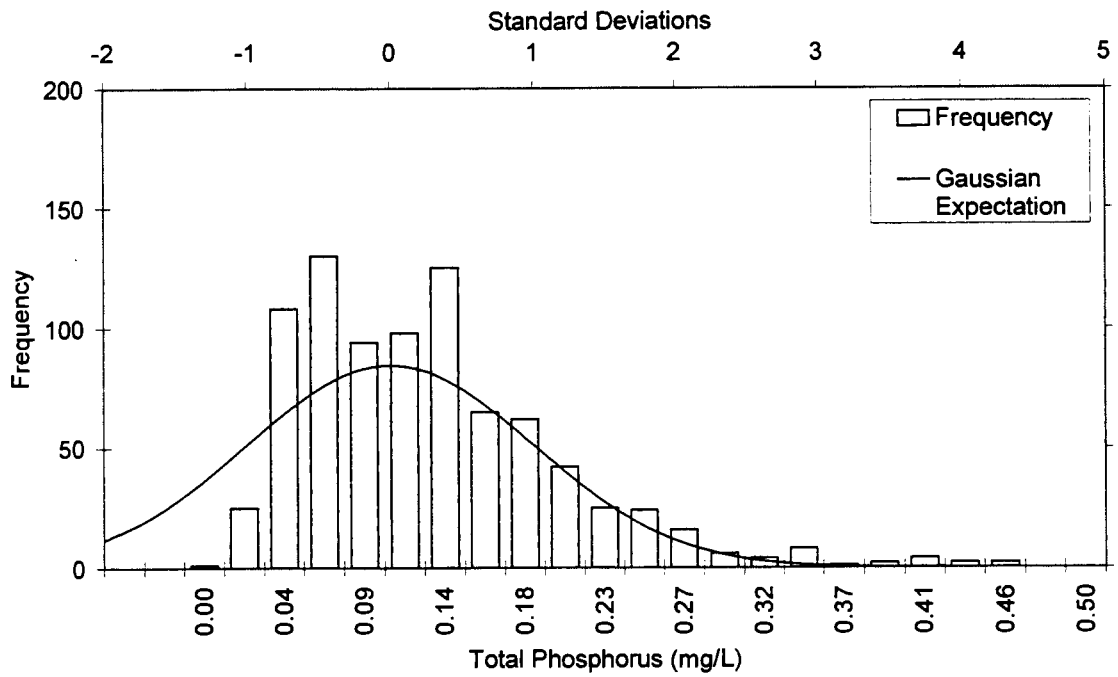

Figure 5.33 Histogram of Total Phosphorus at Morgan (Raw Data)

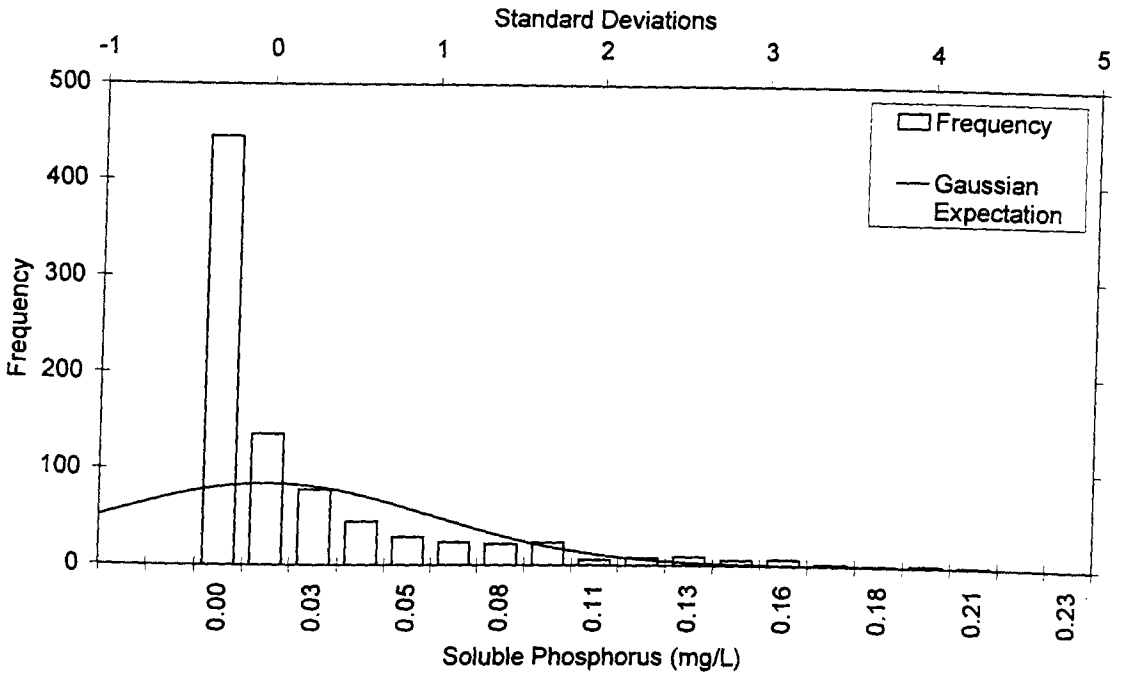

Figure 5.34 Histogram of Soluble Phosphorus at Morgan (Raw Data)

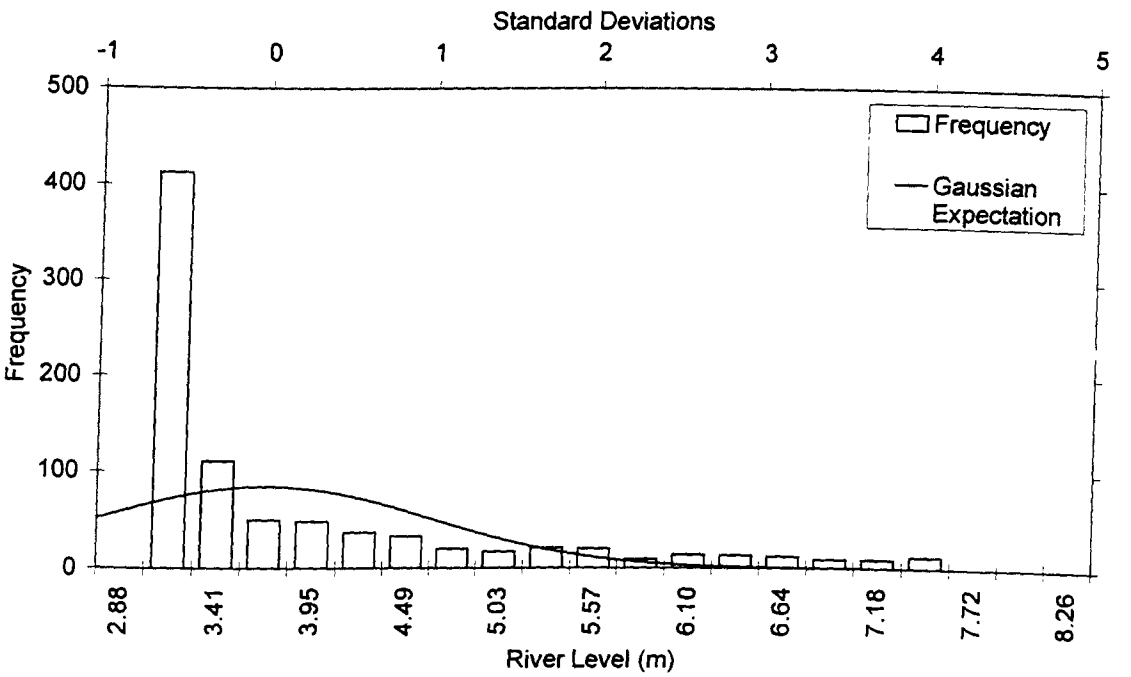

Figure 5.35 Histogram of River Level at Morgan (Raw Data)

5.7.1 Linear Transformation

To linearly transform the variables in the Morgan *Anabaena* spp. data set, the original data range is used to rescale the series to a "suitable" range. For GRNN models, a "suitable" range is one that places each variable on an equal basis, and as such, the

range [0,1] is sufficient. For MLP models, the "suitable" range is one that is commensurate with the transfer function used in the output layer. However, at this step, the transfer function to be used in the output layer is not known, since that is to be optimised in a subsequent step (Section 5.9). Since, linear transformation does not alter the distribution of the variable, it is not necessary to linearly transform the variables until an ANN model is to be developed. For example, the same results would be obtained by applying the stepwise PMI input determination algorithm to the raw data as would be obtained from applying it to the linearly transformed data.

5.7.2 Histogram Equalization

Histograms of each variable in the *Anabaena* spp. data set, after the histogram equalization transformation, are shown in Figure 5.36 to Figure 5.46. Histogram equalization was successful in transforming the distribution of flow into South Australia (Figure 5.37), turbidity (Figure 5.39), TKN (Figure 5.42), total phosphorus (Figure 5.43) and river level (Figure 5.45) to approximate uniformity. The remaining variables were less uniformly distributed but the transformation still was successful in obtaining a more even spread of values over the whole range of each variable. The inability of histogram equalization to produce uniformly transformed series for temperature (Figure 5.36), colour (Figure 5.38), pH (Figure 5.40), silica (Figure 5.41), soluble phosphorus (Figure 5.44) and *Anabaena* spp. (Figure 5.46) results from the large number of common values in each of these series. If the series contains common values, then each of the common values will have the same transformed value after histogram equalization. Consequently, techniques such as histogram equalization and kernel transformation are only able to transform a series to uniformity if there are not too many common values in the series. However, in the case of many common values, the transformation may still be useful as it obtains a more even spread of values across the range for the other data points in the series. This may, in turn, assist the ANN in producing an appropriate mapping of the inputs to the output.

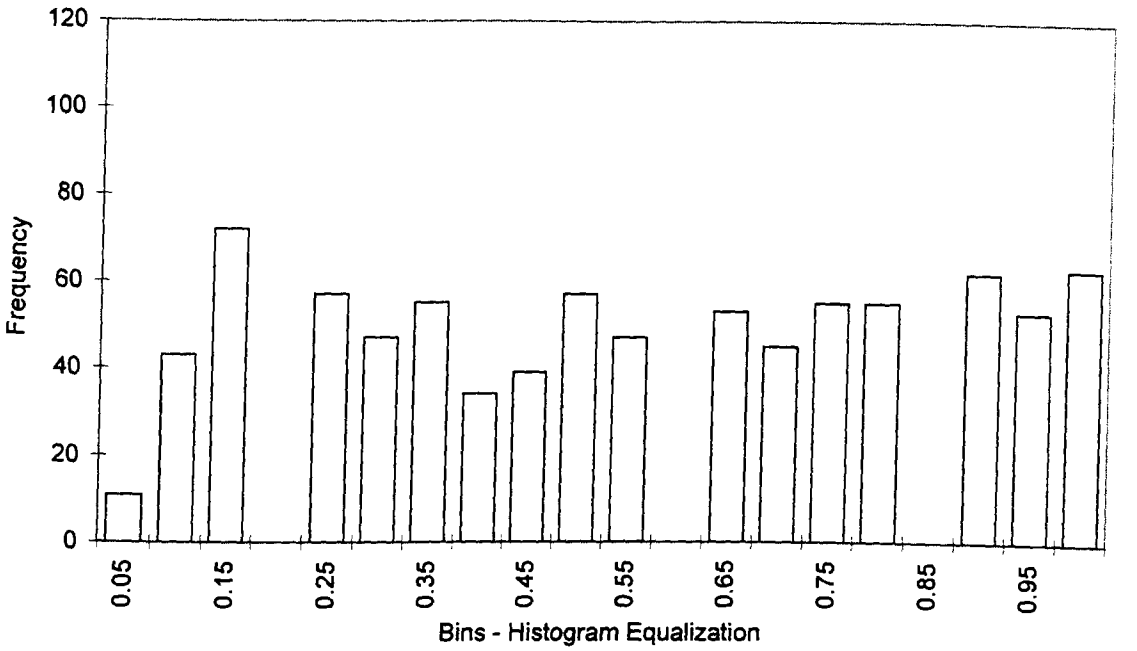

Figure 5.36 Histogram of Temperature at Morgan (Histogram Equalization Transformed Data)

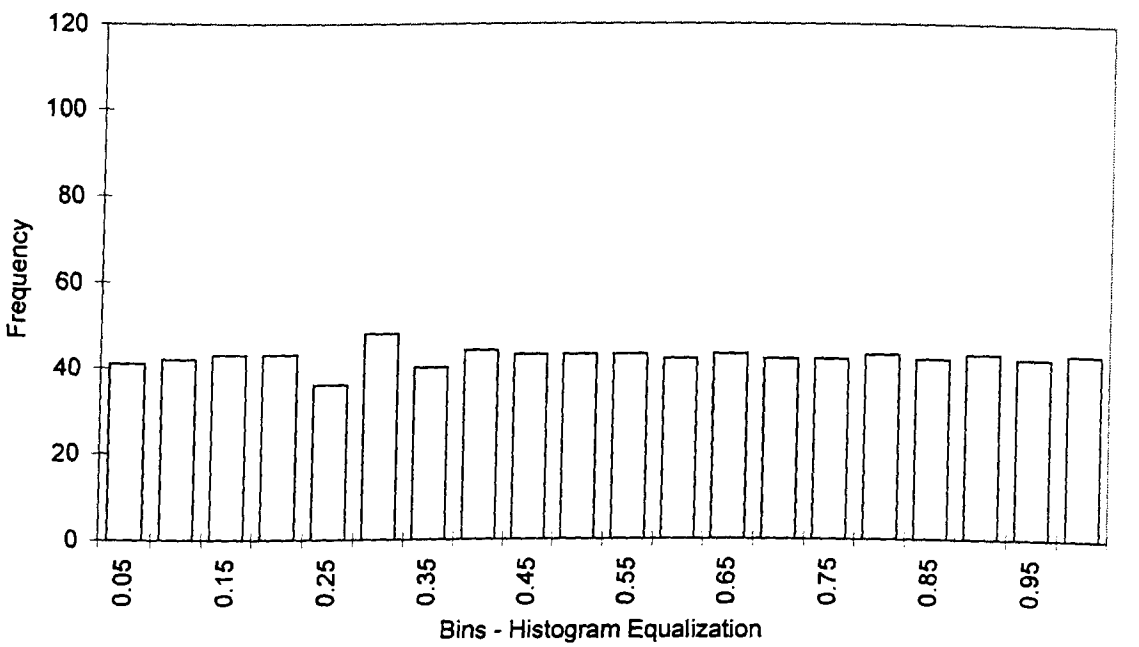

Figure 5.37 Histogram of Flow into SA (Histogram Equalization Transformed Data)

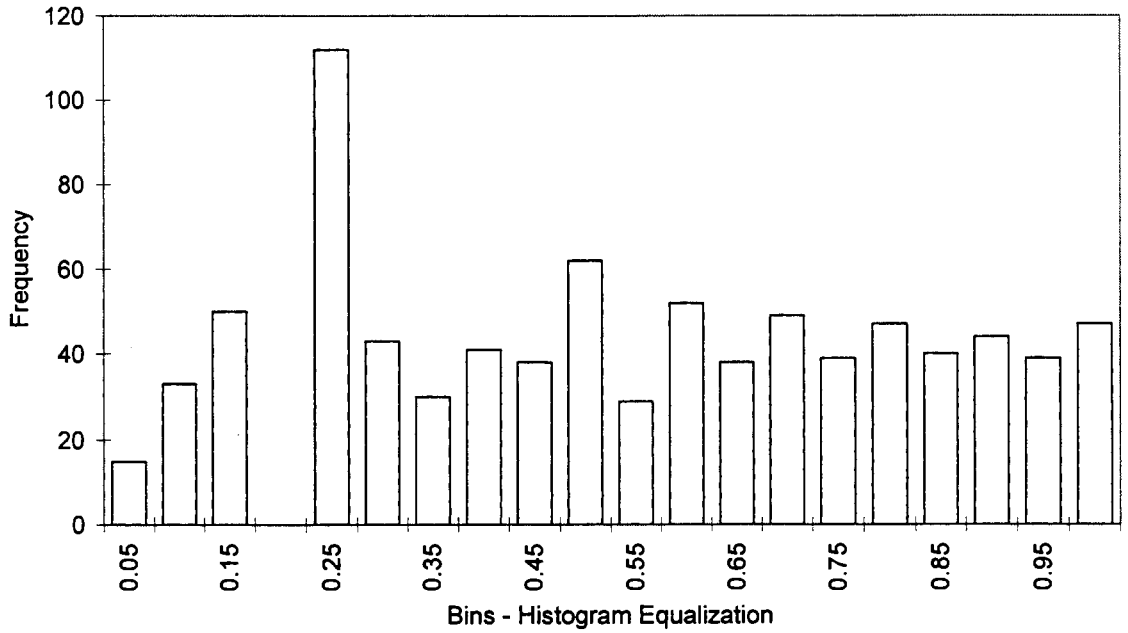

Figure 5.38 Histogram of Colour at Morgan (Histogram Equalization Transformed Data)

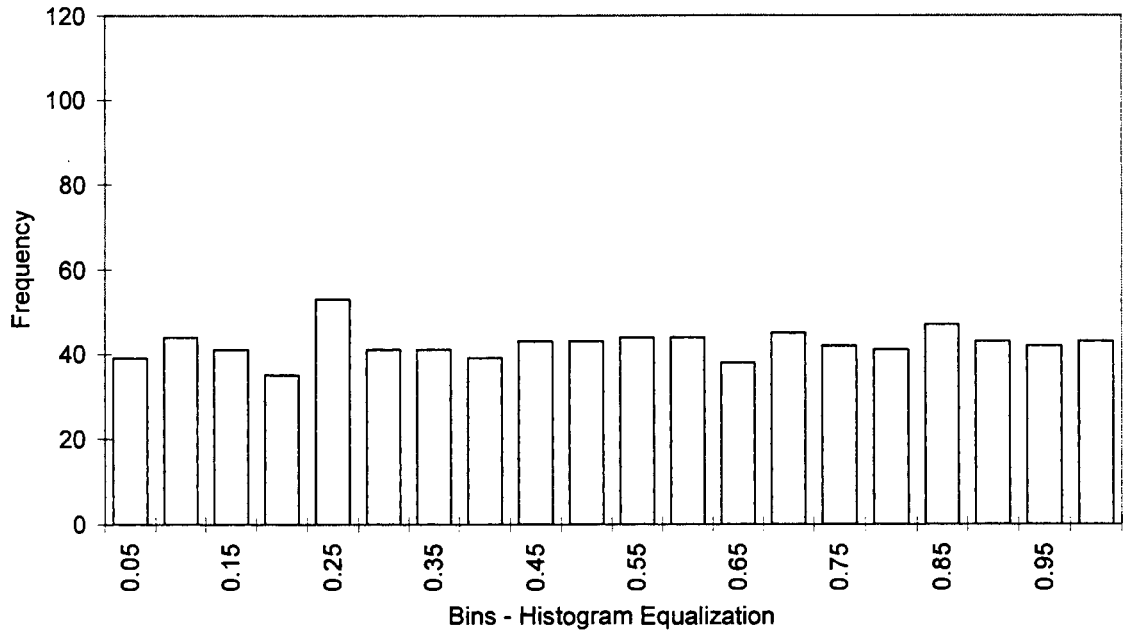

Figure 5.39 Histogram of Turbidity at Morgan (Histogram Equalization Transformed Data)

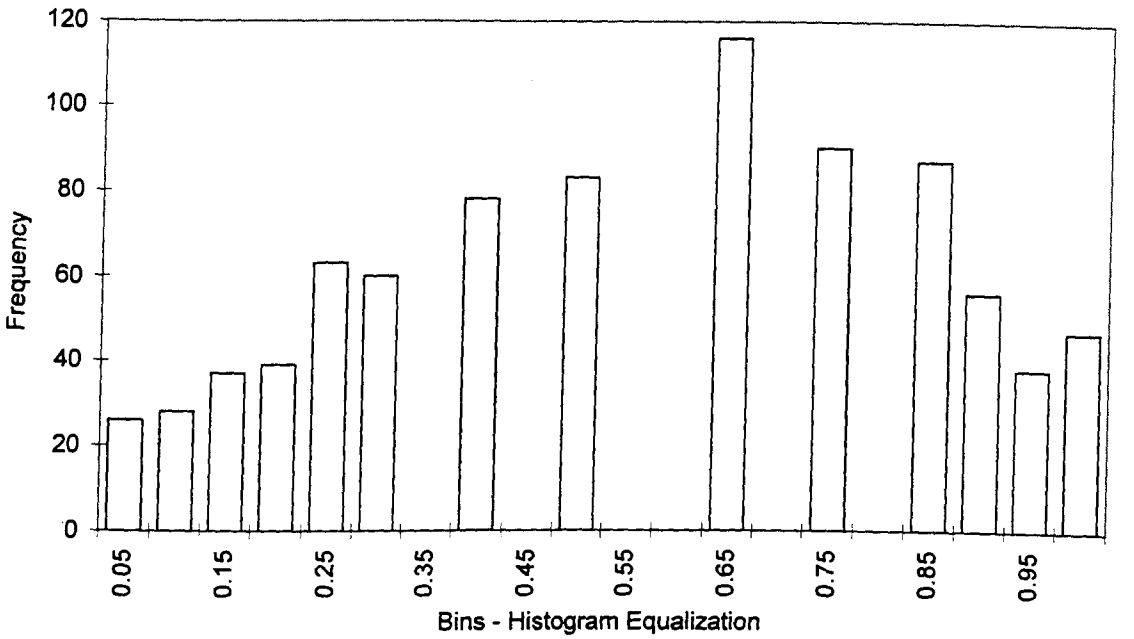

Figure 5.40 Histogram of pH at Morgan (Histogram Equalization Transformed Data)

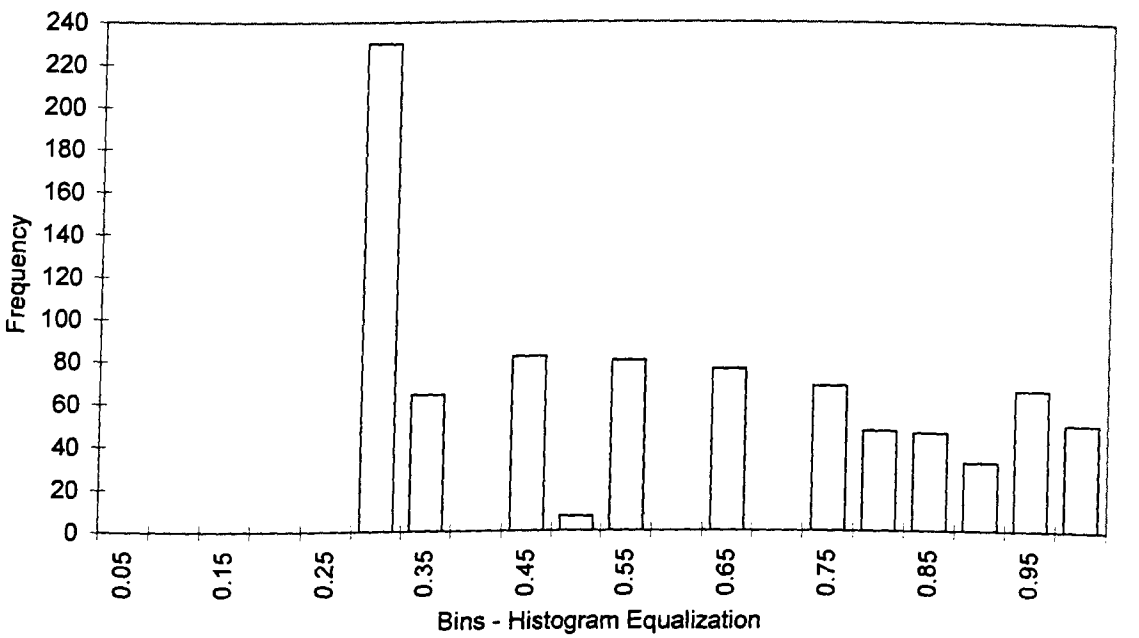

Figure 5.41 Histogram of Silica at Morgan (Histogram Equalization Transformed Data)

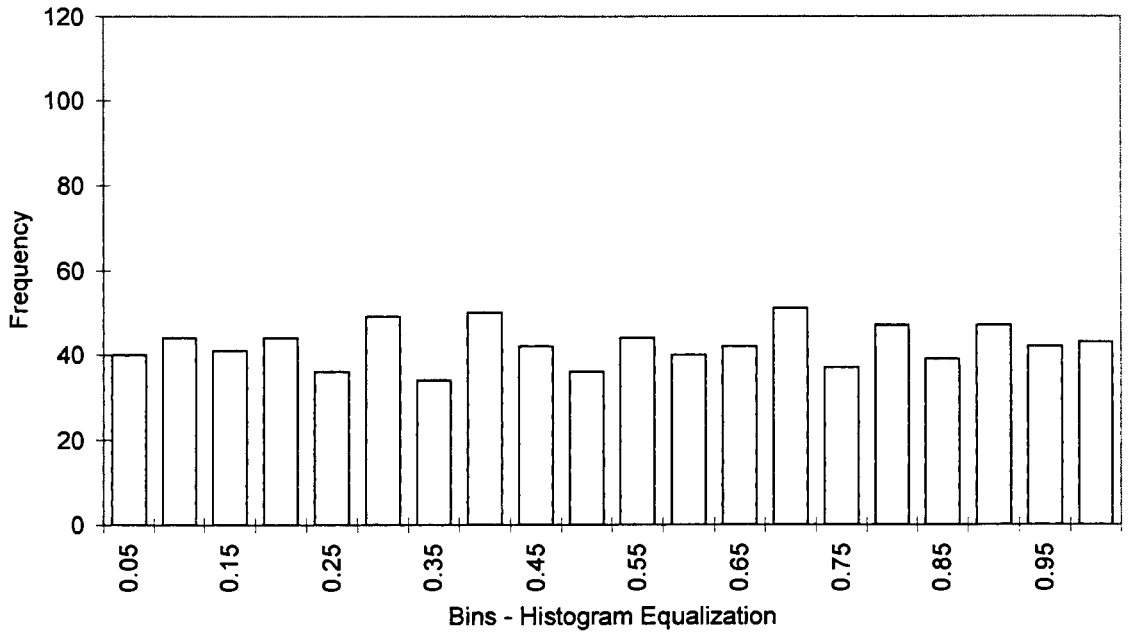

Figure 5.42 Histogram of TKN at Morgan (Histogram Equalization Transformed Data)

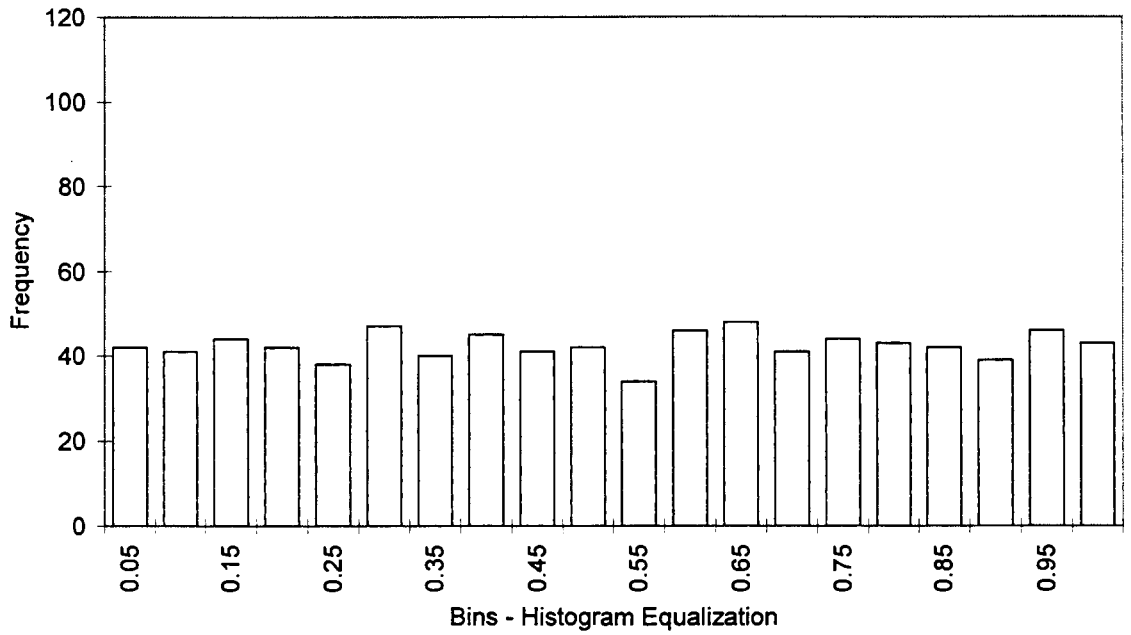

Figure 5.43 Histogram of Total Phosphorus at Morgan (Histogram Equalization Transformed Data)

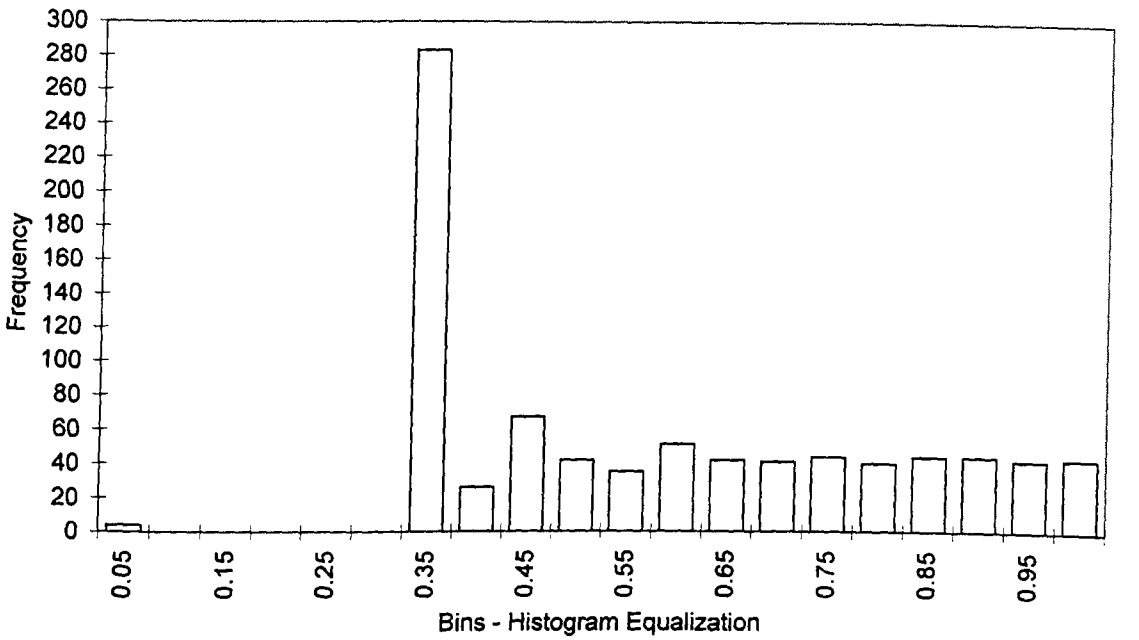

Figure 5.44 Histogram of Soluble Phosphorus at Morgan (Histogram Equalization Transformed Data)

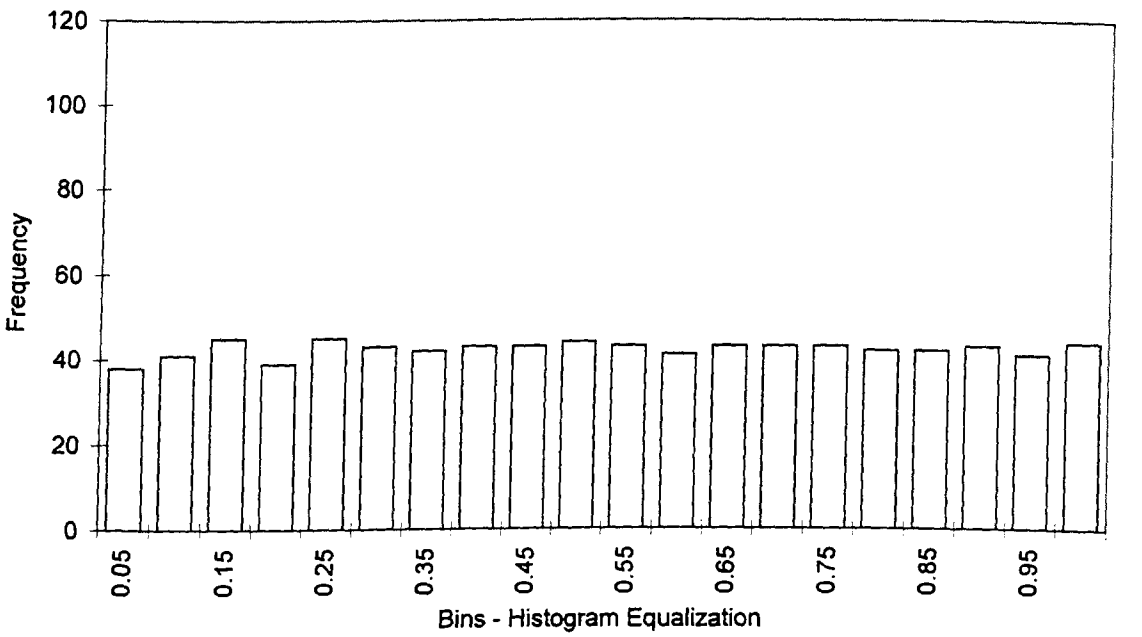

Figure 5.45 Histogram of River Level at Morgan (Histogram Equalization Transformed Data)

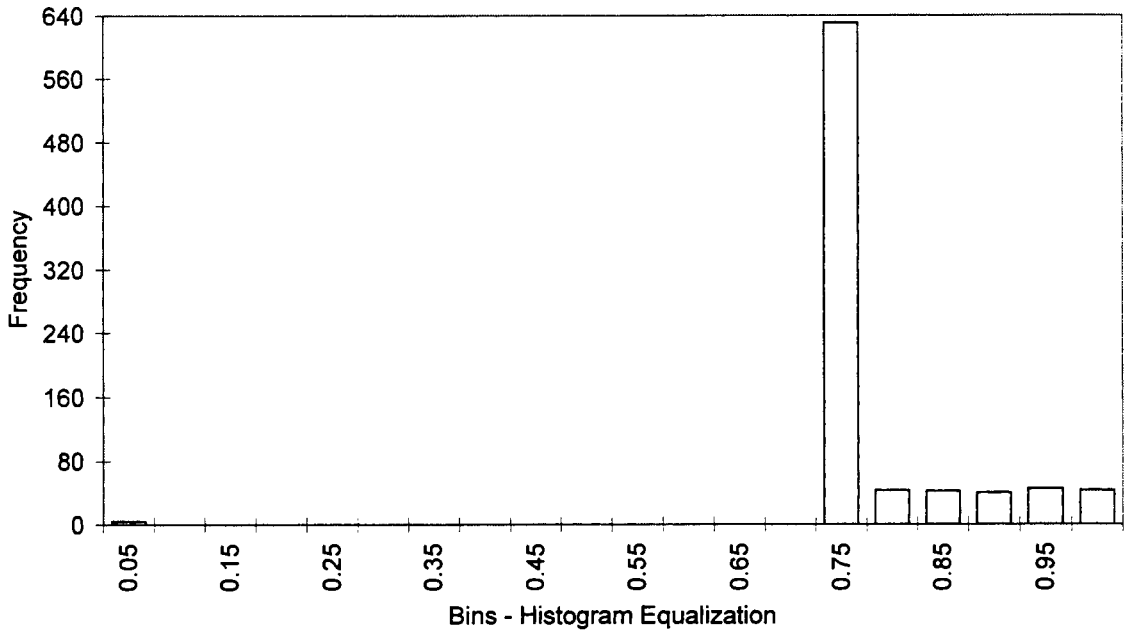

Figure 5.46 Histogram of log(*Anabaena* spp.) at Morgan (Histogram Equalization Transformed Data)

5.8 Determination of Model Inputs

The stepwise PMI input determination algorithm was found to be the most suitable method for selecting appropriate model inputs for the salinity case study in Chapter 4. One limitation that the stepwise PMI algorithm may experience is the selection of an appropriate bandwidth when the data are highly non-Gaussian. Since the *Anabaena* spp. data are highly non-Gaussian, it was considered important to also investigate the effect of using the SOM-GAGRNN input determination technique.

For this case study, the stepwise PMI algorithm and the SOM-GAGRNN were applied to the raw data (i.e. the data to be linearly transformed in a subsequent step) and to the data transformed using histogram equalization. This produced four sets of different inputs.

The stepwise PMI algorithm was applied in a two step procedure. In the first step (bivariate stage), the stepwise PMI algorithm was used to determine the significant lags for each variable. In the second step (multivariate stage), the significant lags selected for each variable were combined into one data set and the stepwise PMI algorithm was used to determine the final set of significant model predictors.

As a starting point for the *Anabaena* spp. data set, the first 26 lags of each of the eleven candidate variables were considered as potential model inputs (i.e. $k_{max} = 26$), giving a total of 286 possible inputs. As mentioned in Section 5.6, it was deemed unlikely that the present concentrations of *Anabaena* spp. would be affected by conditions from more than six months earlier. Therefore, in the bivariate stage, the stepwise PMI algorithm was applied to the first 26 lags of each of the eleven variables, in turn, for both the raw data (i.e. the data to be linearly transformed) and for the data transformed using histogram equalization.

The SOM-GAGRNN utilises the SOM to remove redundant (highly correlated) inputs and the GAGRNN to select the most suitable inputs based on forecasting error. For consistency, $k_{max} = 26$ was also used for each of the eleven variables considered in this input determination method. The SOM was implemented in a univariate step to determine the significant lags for each variable, and then in a multivariate step to ensure limited cross-dependence between the lags selected from each variable. The SOM-GAGRNN was applied to both the raw data (i.e. the data to be linearly transformed) and the data transformed using histogram equalization.

5.8.1 Linear Transformation (Raw) Data

5.8.1.1 Stepwise PMI Algorithm

The results of the bivariate stage of the stepwise PMI algorithm for the raw data are summarised in Table 5.11. The bold numbers in the table represent the higher of the PMI and the 95th percentile randomised sample. The last row in each model's results represents the cutoff point i.e. the input that was not included because the 95th percentile randomised sample was greater than the corresponding PMI score.

Table 5.11 Stepwise PMI Input Selection Algorithm Results on Bivariate log(*Anabaena* spp.) Data Sets (Lagging Window of 26 Weeks)

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
Temperature	26	**0.104**	0.029
	11	**0.095**	0.031
	5	0.022	**0.034**
Flow	18	**0.100**	0.034
	8	**0.088**	0.037
	2	**0.055**	0.039
	26	**0.058**	0.038
	14	0.023	**0.033**
Colour	16	**0.106**	0.036
	23	**0.085**	0.046
	3	**0.071**	0.045
	26	0.042	**0.043**
Turbidity	1	**0.129**	0.038
	15	**0.128**	0.047
	6	**0.071**	0.045
	25	**0.049**	0.046
	20	0.023	**0.046**
pH	16	**0.087**	0.038
	10	**0.066**	0.047
	24	**0.058**	0.042
	5	**0.048**	0.042
	18	0.036	**0.036**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
Silica	1	**0.163**	0.031
	8	**0.091**	0.033
	23	**0.063**	0.037
	4	**0.041**	0.036
	18	**0.045**	0.034
	26	**0.035**	0.033
	14	0.019	**0.030**
TKN	1	**0.066**	0.042
	26	**0.062**	0.046
	15	**0.061**	0.045
	7	**0.045**	0.045
	19	**0.042**	0.041
	11	0.024	**0.033**
TP	1	**0.100**	0.038
	7	**0.086**	0.046
	26	**0.057**	0.043
	16	**0.046**	0.041
	13	0.026	**0.040**
SP	1	**0.051**	0.033
	7	**0.064**	0.034
	23	**0.051**	0.036
	19	0.033	**0.035**
River Level	16	**0.080**	0.031
	6	**0.104**	0.034
	1	**0.051**	0.036
	26	**0.057**	0.038
	22	0.031	**0.035**
Log(*Anabaena*)	1	**0.193**	0.028
	7	**0.137**	0.035
	2	**0.102**	0.033
	21	**0.073**	0.031
	26	**0.055**	0.030
	14	**0.042**	0.030
	3	**0.034**	0.028
	11	0.025	**0.025**
Total	46		

From Table 5.11 it can be seen that the lags selected for most of the variables were not intuitive. A general observation is that the PMI scores for the most significant lag of each variable were generally considerably lower than those obtained for the synthetic test problems (Table 3.8) and the salinity case study (Table 4.26). This tends to reflect

the difficult nature of this forecasting problem. The input variable with the greatest PMI score was log(*Anabaena*) lag 1 with a PMI value of 0.193. This indicates that despite the rapidly varying nature of *Anabaena* spp. populations, the most recent concentration of *Anabaena* yields a significant amount of information about the future concentration of *Anabaena* spp., 4 weeks into the future. For the nutrient variables (i.e. silica, TKN, TP and SP), lag 1 was selected as the most significant lag, suggesting that the most recent nutrient conditions provides valuable information for the 4-week forecast. However, it must be noted that the most significant lags selected for TKN and SP had small PMI scores of 0.066 and 0.051, respectively. This indicates that although these inputs may be significant, the information provided is rather minor. In the bivariate stage, the 286 candidate inputs were reduced to a subset of 46 inputs.

In the multivariate stage of the PMI algorithm, the 46 lags from the bivariate stage were combined into one data set and the algorithm was used to determine the final set of model inputs for the raw (linear transformation) data. The results from the multivariate stage are given in Table 5.12. During this stage, the 46 inputs were further reduced to a final set of 8 inputs. There were three lags of log(*Anabaena*) (lags 1, 7 and 21) included in the final subset. Silica lag 1, temperature lag 26, flow lags 2 and 18 and pH lag 16 were also identified as significant inputs.

Table 5.12 Stepwise PMI Input Selection Algorithm Results on Multivariate Log(*Anabaena* spp.) Data Set (Lagging Window of 26 Weeks)

Variable	Lag	PMI	95th Percentile Randomised Sample PMI
Log(*Anabaena*)	1	***0.1930***	0.0278
Log(*Anabaena*)	7	**0.1372**	0.0350
Silica	1	**0.1328**	0.0432
Temperature	26	**0.0963**	0.0395
Log(*Anabaena*)	21	**0.0669**	0.0305
Flow	2	**0.0617**	0.0364
pH	16	**0.0562**	0.0392
Flow	18	**0.0337**	0.0242
Colour	16	0.0206	**0.0236**
Total	8		

5.8.1.2 SOM-GAGRNN

The results of the SOM-GAGRNN input determination procedure are given in Table 5.13. After applying the SOM in the univariate and multivariate stages, the 286

candidate inputs were reduced to a subset of 39 inputs. This subset was further reduced by the GAGRNN to a final set of 21 inputs. The only variables not represented in the final set were silica, TP and SP. It is interesting to note that the final set selected by the SOM-GAGRNN is quite different to the final set selected by the stepwise PMI algorithm (Table 5.12). However, this is consistent with the results obtained for the salinity case study.

Table 5.13 Input Subsets Selected using SOM-GAGRNN

Input	Unsupervised Technique		Supervised Technique
	Univariate SOM	Multivariate SOM	GAGRNN
Temperature	Lags 1, 3, 4, 5, 7, 8, 10, 11, 13, 14, 17, 18, 19, 21, 23, 24	Lags 1, 13	Lag 13
Flow	Lags 1, 3, 5, 7, 8, 10, 11, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25	Lags 1, 3, 5, 7, 8, 10, 11, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25	Lags 1, 3, 5, 8, 10, 14, 15, 20, 23, 25
Colour	Lags 1, 4, 5, 7, 8, 10, 12, 14, 15, 17, 18, 20, 21, 23, 25	Lags 1, 20	Lag 20
Turbidity	Lags 1, 4, 6, 8, 9, 11, 12, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25	Lags 1, 14, 20	Lags 1, 20
pH	Lags 1, 2, 3, 4, 5, 6, 7, 9, 10, 13, 16, 18, 19, 21, 23, 24	Lag 1	Lag 1
Silica	Lags 1, 3, 4, 6, 8, 9, 11, 13, 15, 16, 18, 20, 22, 24, 25	-	-
TKN	Lags 1, 4, 7, 10, 11, 13, 14, 16, 17, 20, 22, 24	Lag 1	Lag 1
TP	Lags 1, 3, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 23, 24	-	-
SP	1, 4, 7, 9, 10, 12, 13, 15, 18, 21, 24, 25	Lag 1	-
River Level	Lags 1, 3, 5, 7, 8, 10, 12, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25	Lag 1	Lag 1
Log(*Anabaena*)	Lags 1, 4, 5, 6, 8, 9, 12, 13, 15, 16, 18, 20, 21, 23, 24, 25	Lags 1, 5, 6, 9, 12, 13, 18, 20, 23, 24, 25	Lags 1, 6, 9, 13
Total	168	39	21

5.8.2 Histogram Equalization Transformation Data

5.8.2.1 Stepwise PMI Algorithm

The results of the bivariate stage of the stepwise PMI algorithm for the histogram equalization data are summarised in Table 5.14. Once again, the PMI scores for the most significant lag of each variable were relatively low. The input with the highest PMI score was log(*Anabaena*) lag 1 with a PMI value of 0.179, closely followed by silica lag 1 with a PMI value of 0.160. As was the case for the raw data in Table 5.11, the most significant lag for each of the nutrient variables was lag 1. During the bivariate stage for the data transformed using histogram equalization, the 286 candidate inputs were reduced to a subset of 55 inputs. This was larger than the 46 inputs selected for the raw data in the bivariate stage. There was a fair degree of similarity of the significant lags for the raw and histogram equalized data subsets, and, in fact, there were 23 inputs that were common to both the subsets after application of the bivariate stage of the stepwise PMI algorithm.

Table 5.14 Stepwise PMI Input Selection Algorithm Results on Bivariate Log(*Anabaena* spp.) Data Sets Transformed Using Histogram Equalization (Lagging Window of 26 Weeks)

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
Temperature	26	**0.100**	0.028
	11	**0.094**	0.028
	1	0.023	**0.031**
Flow	1	**0.087**	0.033
	22	**0.113**	0.031
	17	**0.068**	0.035
	26	**0.053**	0.036
	11	**0.034**	0.033
	7	0.020	**0.033**
Colour	16	**0.099**	0.031
	1	**0.097**	0.036
	8	**0.080**	0.034
	23	**0.067**	0.031
	26	0.023	**0.029**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
Turbidity	1	**0.151**	0.031
	9	**0.118**	0.039
	18	**0.072**	0.038
	26	**0.051**	0.036
	4	0.017	**0.031**
pH	15	**0.070**	0.027
	10	**0.071**	0.028
	24	**0.061**	0.026
	18	**0.049**	0.030
	5	**0.041**	0.025
	1	**0.023**	0.020
	20	0.011	**0.012**
Silica	1	**0.160**	0.024
	7	**0.119**	0.023
	15	**0.071**	0.026
	25	**0.071**	0.027
	3	**0.033**	0.026
	18	**0.027**	0.025
	10	0.018	**0.022**
TKN	1	**0.059**	0.033
	8	**0.057**	0.035
	26	**0.052**	0.033
	17	**0.051**	0.030
	4	**0.029**	0.028
	13	0.020	**0.021**
TP	1	**0.110**	0.033
	11	**0.095**	0.033
	2	**0.055**	0.035
	26	**0.039**	0.031
	20	**0.033**	0.031
	8	0.018	**0.028**

Continued.

Variable	Lags	PMI	95th Percentile Randomised Sample PMI
SP	1	**0.071**	0.028
	7	**0.099**	0.030
	16	**0.078**	0.030
	26	**0.057**	0.031
	23	**0.034**	0.029
	13	**0.029**	0.028
	4	0.026	**0.027**
River Level	1	**0.084**	0.030
	15	**0.084**	0.031
	25	**0.049**	0.034
	7	**0.034**	0.032
	18	0.018	**0.031**
Log(*Anabaena*)	1	**0.179**	0.029
	4	**0.120**	0.036
	7	**0.082**	0.037
	2	**0.061**	0.031
	22	**0.059**	0.030
	26	**0.041**	0.027
	19	**0.035**	0.026
	13	**0.025**	0.023
	3	0.020	**0.021**
Total	55		

In the multivariate stage of the PMI algorithm, the 55 lags from the bivariate stage were combined into one data set and the algorithm was used to determine the final set of model inputs for the data set transformed using histogram equalization. The results of the multivariate stage of the stepwise PMI algorithm are shown in Table 5.15. It can be seen that 12 inputs were selected in the final set, however, there were only 3 inputs in common with the final set chosen for the raw data in Table 5.12. Log(*Anabaena*) was the most common input with 5 lags selected, followed by flow with 2 lags, and silica, colour, TP, pH and SP each with 1 lag.

Table 5.15 Stepwise PMI Input Selection Algorithm Results on Multivariate Log(*Anabaena* spp.) Data Set Transformed Using Histogram Equalization (Lagging Window of 26 Weeks)

Variable	Lag	PMI	95th Percentile Randomised Sample PMI
Log(*Anabaena*)	1	**0.1790**	0.0287
Silica	1	**0.1326**	0.0351
Flow	1	**0.1060**	0.0415
Log(*Anabaena*)	2	**0.0776**	0.0380
Colour	16	**0.0633**	0.0328
Flow	26	**0.0471**	0.0329
TP	26	**0.0265**	0.0251
pH	5	**0.0145**	0.0124
Log(*Anabaena*)	26	**0.0053**	0.0049
Log(*Anabaena*)	7	**0.0042**	0.0041
SP	1	**0.0045**	0.0043
Log(*Anabaena*)	13	**0.0021**	0.0015
Log(*Anabaena*)	19	0.0012	**0.0014**
Total	12		

5.8.2.2 SOM-GAGRNN

The results of the SOM-GAGRNN input determination procedure for the histogram equalized data are given in Table 5.16. After applying the SOM in the univariate and multivariate stages, the 286 candidate inputs were reduced to a subset of 60 inputs. This subset was further reduced by the GAGRNN to a final set of 27 inputs. The final subset selected by the SOM-GAGRNN was very different to the final subset selected by the stepwise PMI algorithm and to that selected from the raw data (Table 5.13). The absence of log(*Anabaena*) at lag 1 is difficult to understand given that it was found to be significant in the 3 other procedures. It should be noted that it is difficult to extract some possible physical meaning from the final set of inputs selected.

Table 5.16 Input Subsets Selected using SOM-GAGRNN (Data Transformed Using Histogram Equalization)

Input	Unsupervised Technique		Supervised Technique
	Univariate SOM	Multivariate SOM	GAGRNN
Temperature	Lags 1, 4, 5, 6, 7, 8, 10, 11, 13, 14, 16, 18, 20, 21, 23, 24	Lags 1, 4, 10, 14, 18, 23	Lags 10, 14
Flow	Lags 1, 3, 4, 6, 8, 9, 11, 12, 14, 16, 18, 19, 21, 23, 25	Lags 8, 18, 21, 23, 25	Lags 8, 18, 21
Colour	Lags 1, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 21, 22, 24, 25	Lags 1, 7, 13, 16, 24	Lags 7, 13, 24
Turbidity	Lags 1, 4, 6, 8, 11, 13, 16, 18, 19, 21, 23, 25	Lags 16, 18, 23	Lag 18
pH	Lags 1, 4, 5, 7, 8, 10, 11, 13, 14, 15, 17, 18, 19, 21, 22, 24, 25	Lags 1, 7, 11, 14, 17, 21	Lag 1, 14
Silica	Lags 1, 4, 6, 8, 9, 11, 12, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25	Lags 1, 6, 11, 14, 20, 22	Lags 1, 6, 11, 20
TKN	Lags 1, 4, 5, 7, 8, 11, 12, 13, 14, 15, 17, 18, 20, 21, 23, 24, 25	Lags 1, 7, 12, 14, 17, 20, 23	Lags 12, 17, 20
TP	Lags 1, 4, 5, 7, 8, 9, 11, 12, 14, 15, 17, 18, 20, 21, 23, 24	Lags 4, 7, 9, 11, 14, 17, 20, 21, 23	Lags 4, 9, 11, 17
SP	Lags 1, 4, 5, 7, 8, 10, 12, 14, 15, 17, 18, 19, 21, 22, 24, 25	Lags 4, 10, 17, 21	Lag 17
River Level	Lags 1, 4, 5, 7, 8, 10, 12, 13, 14, 15, 17, 18, 20, 21, 22, 24, 25	Lags 1, 12, 17, 18, 20, 22	Lags 1, 20, 22
Log(*Anabaena*)	Lags 1, 3, 5, 8, 9, 11, 12, 14, 15, 18, 20, 21, 23	Lags 1, 11, 18	Lag 11
Total	173	60	27

5.9 Choice of Network Type and Architecture

The two feedforward ANN models proposed in Chapter 3, namely the GRNN and EBMLP were used to forecast *Anabaena* spp. at Morgan, 4 weeks in advance. It was considered necessary to try both ANN models because the choice of a suitable network type is case study dependent. A GRNN and EBMLP were developed for each of the 4 input sets identified in Section 5.8, resulting in the development of 8 ANN models.

For the GRNN models, the raw data were linearly transformed between 0.0 and +1.0. The input data transformed using histogram equalization were already scaled between 0.0 and +1.0 by virtue of the transformation. Therefore, only the output variable (i.e. *Anabaena* spp. at t+4 weeks) required linear transfornmation, since this had not been transformed using histogram equalization. For the MLP models, both the raw and histogram equalization data were linearly transformed to a range that was commensurate with the output transfer function. The ranges used in conjunction with the positive output transfer functions (i.e. linear and sigmoid) were -1.0 to +1.0 for the network inputs and +0.2 to +0.8 for the network output. The ranges used in conjunction with the positive/negative output transfer function (i.e. hyperbolic tangent) were -1.0 to +1.0 for the network inputs and -0.8 to +0.8 for the network output.

The GRNN models used in this section were single-sigma models trained using the inverse Hessian method, as discussed in Section 3.6.2.2 The EBMLP models implemented in this section used a GA to evolve MLP models with a suitable network architecture, as discussed in Section 3.6.2.1. In accordance with the results obtained for the salinity case study, 100 generations were used in the GA run, with a population size of 20 ANN models. The cloning refill strategy was used, as this strategy was found to lead to the development of near-optimal models.

5.9.1 Linear Transformation (Raw) Data

5.9.1.1 Stepwise PMI Algorithm

GRNN

A GRNN model was developed (Model 1) using the linearly transformed data and the inputs identified by the stepwise PMI algorithm. A plot of the 4-week forecasts obtained using this model for all data (training, testing and validation) is shown in Figure 5.47. It can be seen that Model 1 tended to underestimate the peak concentrations of *Anabaena* spp. for each year. The occurrence of a significant growth

event (i.e. concentration > 500 cells/mL) was only correctly predicted during 1991, 1992, 1993, 1994 and 1996, however, the actual peak concentration was still underestimated during each of these years. Model 1 performed well at forecasting the timing of growth events, although the predicted onset of many of the events lagged behind the actual onset. The durations of the growth events was well modelled.

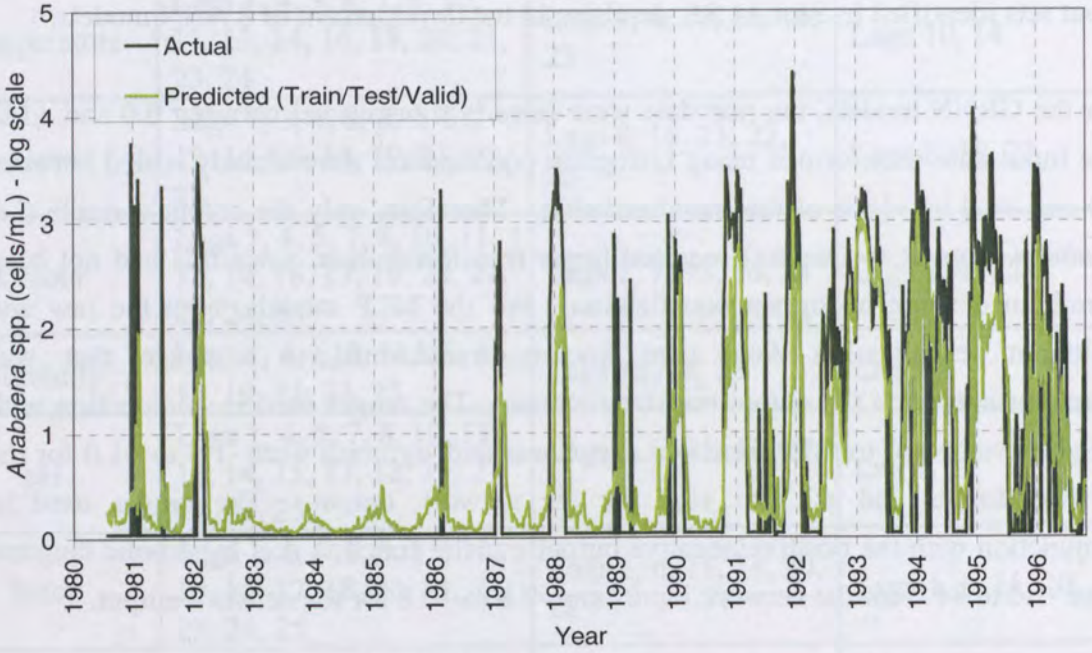

Figure 5.47 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 1)

The forecasting results plotted in Figure 5.47 also include predicted values for training and testing data. However, model performance must ultimately be assessed using the results on the independent validation set. To aid in the assessment of model performance, the validation set forecasts have been graphically presented in Figure 5.48. It can be seen that the forecasts obtained from Model 1 tended to underestimate the actual validation points. The timing of the validation set predictions also lagged the actual onset of growth events. For the validation points in 1983, 1984 and 1985, Model 1 correctly predicted that there were no significant growth events. This was quite surprising since turbidity was not included as an input for Model 1 and the absence of *Anabaena* spp. during these years was thought to be due to high turbidity levels (see Section 5.3.6). However, silica was included as an input in Model 1, and silica concentrations are correlated with turbidity. Hence, the model may have used this information to successfully forecast low *Anabaena* spp. concentrations during these years.

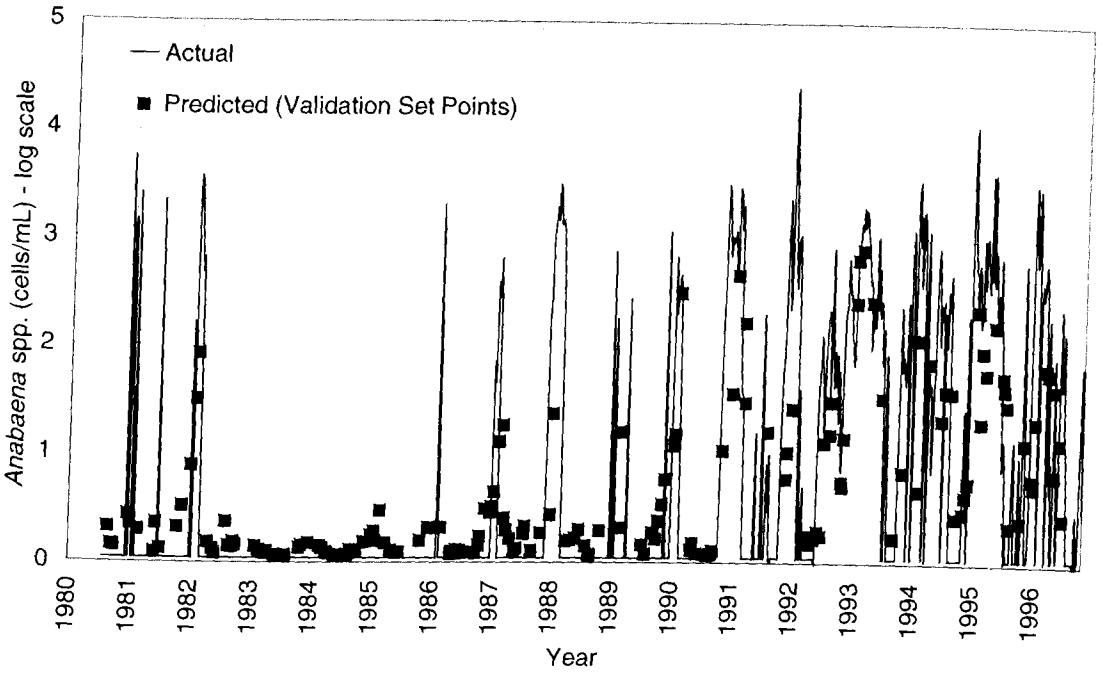

Figure 5.48 Validation Set GRNN 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 1)

EBMLP

Using the linearly transformed data and the inputs identified by the stepwise PMI algorithm, a number of EBMLP models were developed. Details of the ten best networks found during the GA run are given in Table 5.17. It is interesting to note that the ten best networks each made use of a second hidden layer, which appears to be indicative of the complexity of this problem and the relationships being modelled. However, the number of PEs in both hidden layers were relatively small. The best model only utilised 7 PEs in the first hidden layer and 3 PEs in the second hidden layer. As expected, the cloning refill strategy resulted in the ten best networks having very similar architectures.

The best network from the GA run was selected and is referred to as Model 2. A plot of the 4-week forecasts obtained by Model 2 is shown in Figure 5.49. The forecasts obtained by Model 2 were very similar to those obtained by Model 1. Model 2 was able to predict each of the significant growth events predicted by Model 1 and was able to predict an additional significant growth event that occurred during 1995. The validation set forecasts for Model 2 are shown in Figure 5.50. These results were also similar to Model 1, however, Model 2 performed better at predicting the growth events from 1992 to 1996. Model 2 was also successful at predicting that there were no

significant events during 1983, 1984 and 1985. It did more often forecast *Anabaena* numbers at times when they were absent (e.g. in 1980 and 1987).

Table 5.17 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (log scale)	Testing Set RMSE (log scale)
1	3S, 2H, 2L	1S, 1H, 1L	1L	0.733	0.699
2	2S, 2H, 2L	1S, 1H	1L	0.765	0.700
3	2S, 2H, 2L	1S, 1H	1L	0.780	0.707
4	2S, 2H, 3L	1S, 1H	1L	0.779	0.707
5	2S, 2H, 2L	1S, 1H	1L	0.754	0.708
6	2S, 2H, 2L	1S, 1H	1L	0.773	0.709
7	2S, 2H, 2L	1S, 1H	1L	0.771	0.709
8	2S, 2H, 2L	1S, 1H	1L	0.770	0.710
9	2S, 2H, 2L	1S, 1H	1L	0.762	0.716
10	2S, 2H, 2L	1S, 1H	1L	0.766	0.718

L = Linear Transfer Func.
S = Sigmoid Transfer Func.
H = Hyperbolic Tangent Transfer Func.

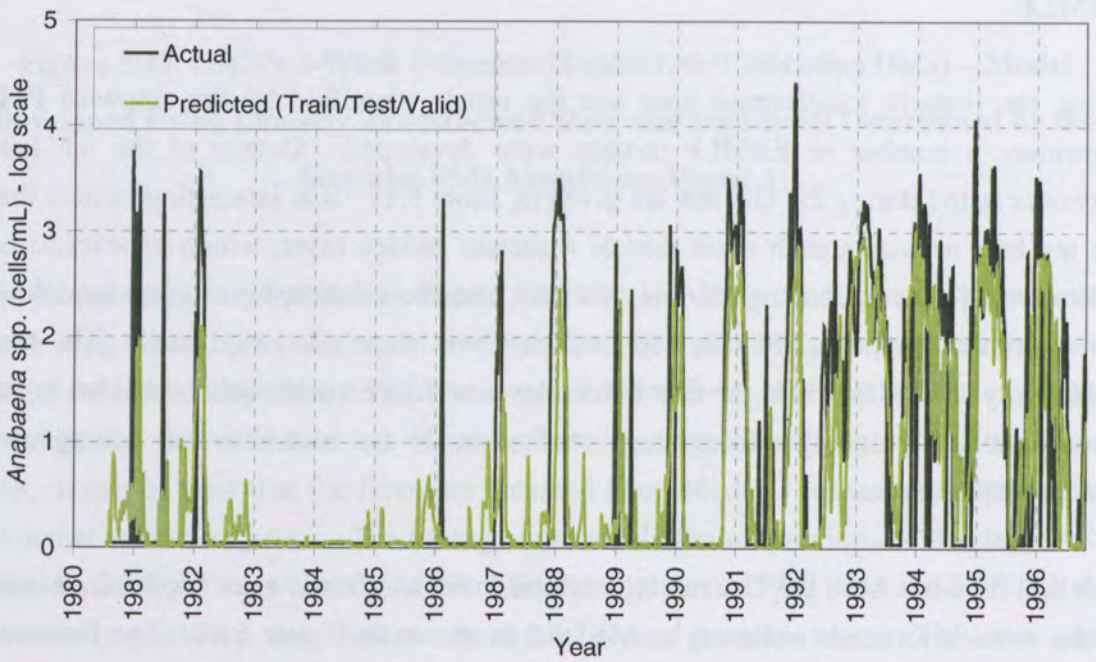

Figure 5.49 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 2)

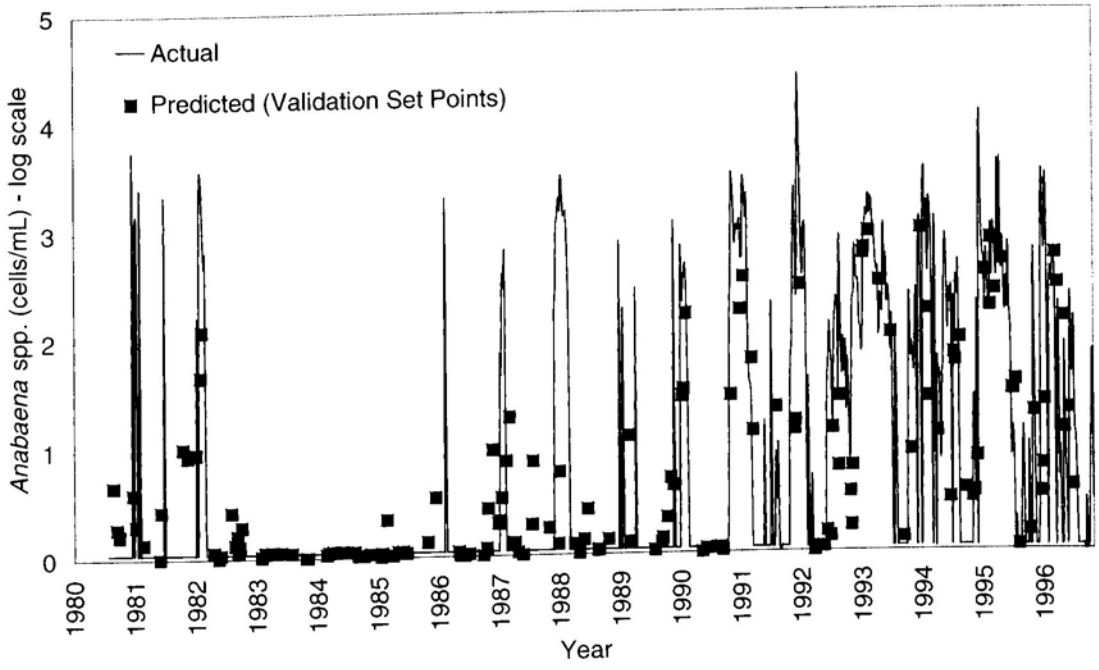

Figure 5.50 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the Stepwise PMI Algorithm (Model 2)

5.9.1.2 SOM-GAGRNN

GRNN

Using the linearly transformed data and the inputs identified by the SOM-GAGRNN procedure, a GRNN model was developed (Model 3). A plot of the 4-week forecasts obtained is shown in Figure 5.51. It is apparent that this model was able to predict nearly all of the significant growth events. The only events that were under-predicted were the events that occurred in 1981, 1986 and 1989. Model 3 also performed well at predicting the onset and duration of all growth events. The independent validation set forecasts for Model 3 are shown in Figure 5.52. These results indicate that this model was successful at learning the general relationship between the inputs and *Anabaena* spp. concentrations. The large peak that occurred towards the end of 1991 was successfully predicted by Model 3 and the timing of the validation set predictions was considerably better than that achieved by Models 1 and 2. The absence of *Anabaena* spp. during 1983, 1984 and 1985 was also well predicted.

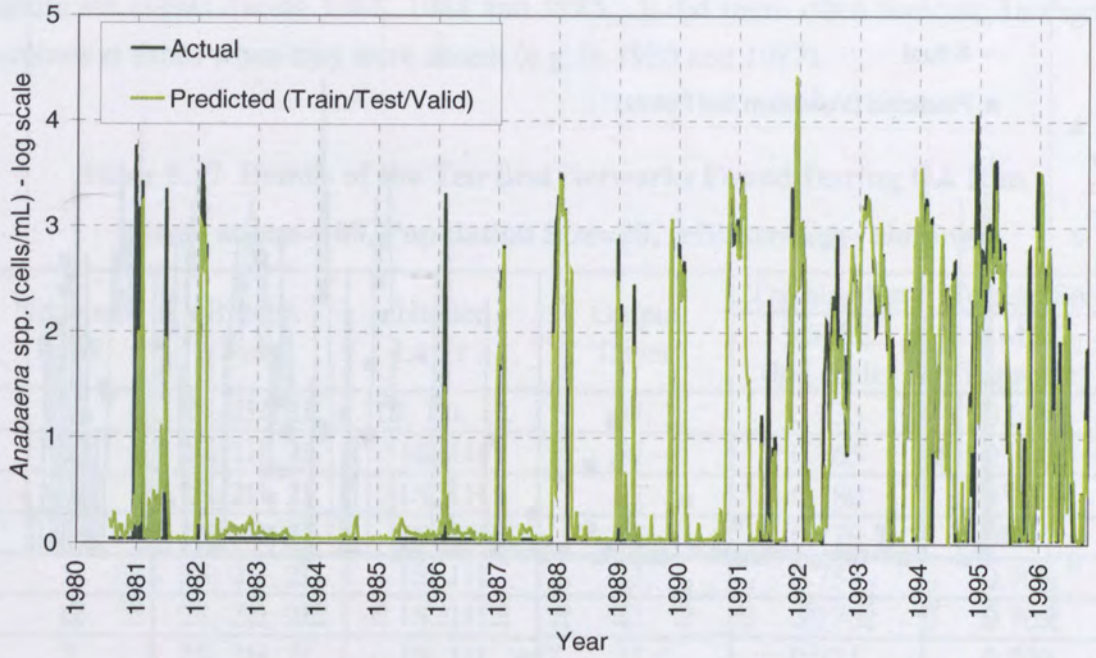

Figure 5.51 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 3)

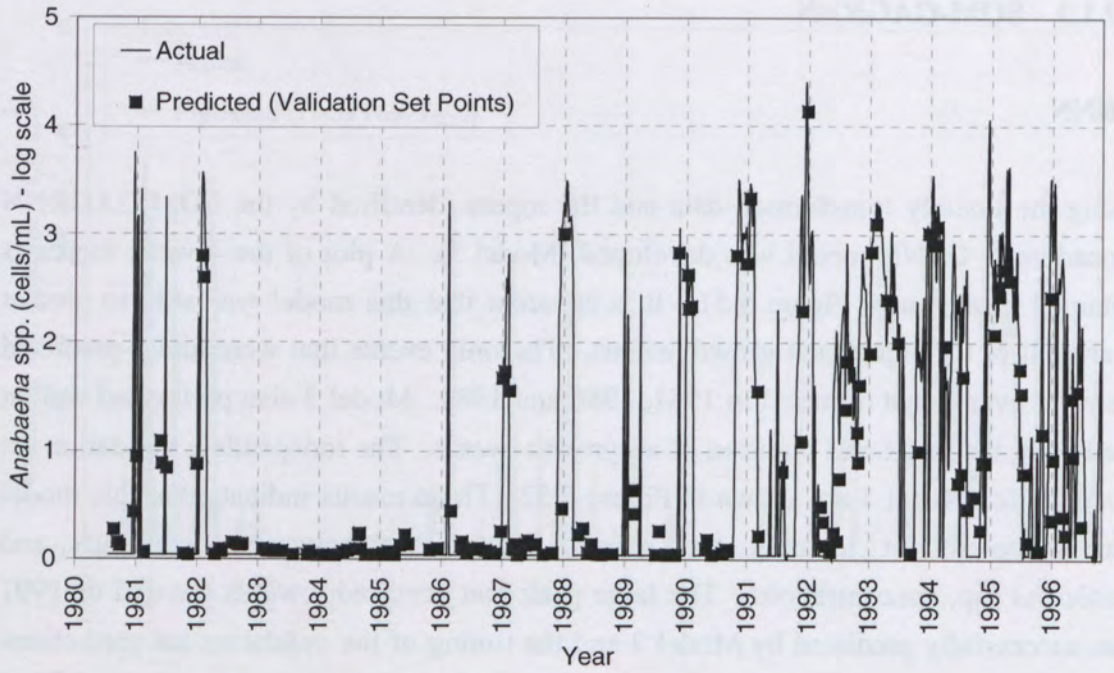

Figure 5.52 Validation Set GRNN 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 3)

EBMLP

Using the linearly transformed data and the inputs identified by the SOM-GAGRNN procedure, a number of EBMLP models were developed. Details of the ten best networks found during the GA run are shown in Table 5.18. Nine of the ten best networks required the use of a second hidden layer, which, as discussed previously, is indicative of the complexity of the relationships being modelled. Despite the fact that the cloning refill strategy was used, the ten best networks exhibited variation in their architecture. Each of the models made use of the sigmoidal, hyperbolic tangent and linear transfer functions in the hidden layers and only used the linear transfer function in the output layer. In comparison to the MLPs developed using the inputs identified by the stepwise PMI algorithm (Table 5.18), a much larger number of hidden layer PEs were required for the MLPs developed using the inputs identified by the SOM-GAGRNN. This is most likely due to the larger number of inputs identified by the SOM-GAGRNN procedure (21 inputs compared with 8 inputs for the stepwise PMI algorithm).

Table 5.18 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (log scale)	Testing Set RMSE (log scale)
1	10S, 19H, 9L	2S, 10H, 10L	1L	0.698	0.723
2	1S, 10H, 12L	<none>	1L	0.719	0.733
3	14S, 14H, 15L	2S, 2H, 2L	1L	0.733	0.741
4	13S, 12H, 13L	1S, 3H, 3L	1L	0.727	0.749
5	13S, 12H, 13L	1S, 4H, 3L	1L	0.745	0.755
6	16S, 15H, 7L	1S, 3H, 2L	1L	0.774	0.760
7	16S, 15H, 7L	1S, 3H, 3L	1L	0.756	0.761
8	16S, 15H, 7L	1S, 4H, 3L	1L	0.777	0.763
9	13S, 12H, 13L	1S, 3H, 3L	1L	0.781	0.771
10	10S, 19H, 9L	1S, 3H, 2L	1L	0.772	0.780

The best network from the GA run was selected and is referred to as Model 4. A plot of the 4-week forecasts obtained by Model 4 is shown in Figure 5.53. Model 4 was only able to predict 7 of the 14 significant growth events and tended to underestimate the peak concentrations. The forecasts produced by this model also tended to lag the actual onset of the growth events, however, the duration of most events was well modelled. The independent validation set forecasts produced by Model 4 are shown in Figure 5.54. These results are not as good as those obtained by Model 3 but still represent an improvement over Models 1 and 2. The peaks in 1988 and 1993 were well

predicted. In general, the timing of the validation set forecasts for Model 4 was not good, with the predicted onset of growth events lagging behind the actual onset.

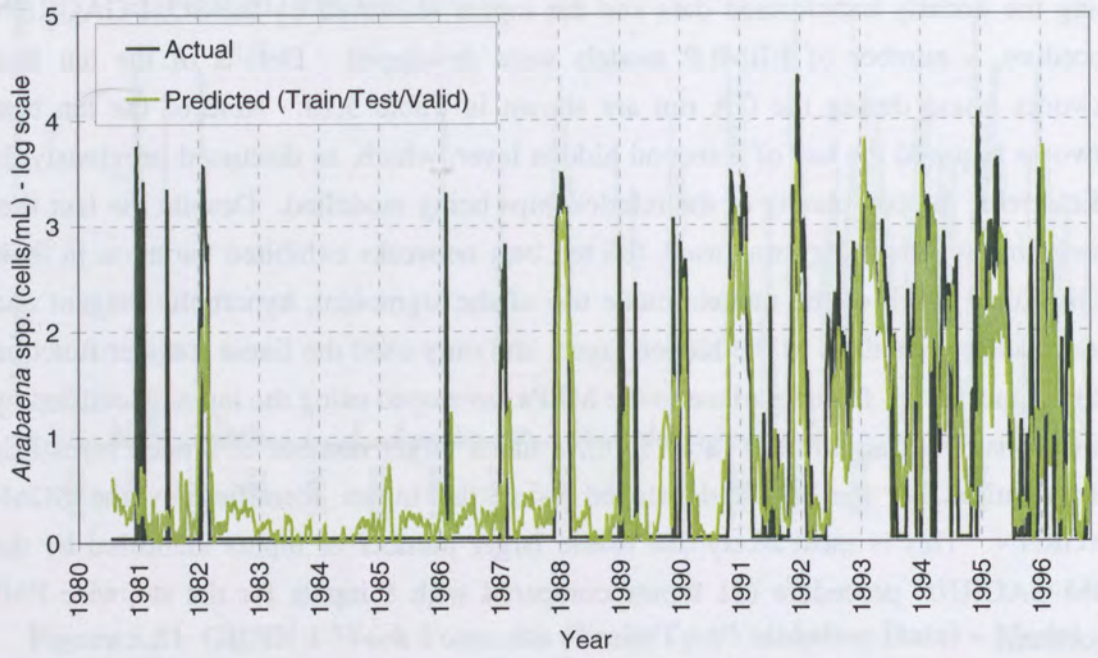

Figure 5.53 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 4)

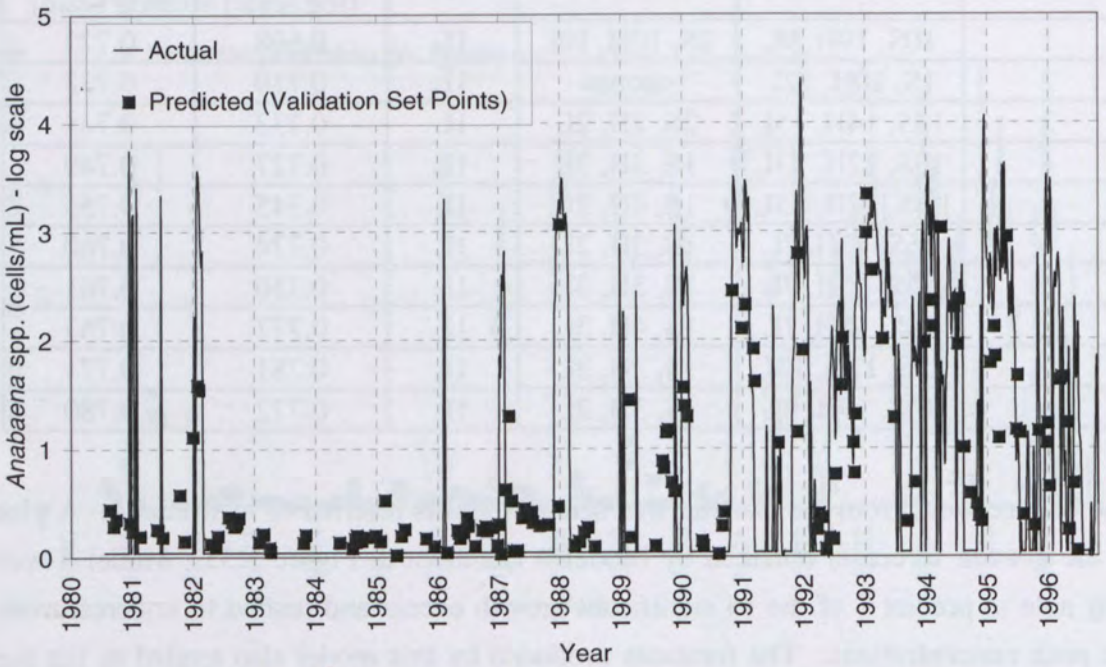

Figure 5.54 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Linearly Transformed Data and the Inputs Determined by the SOM-GAGRNN (Model 4)

5.9.2 Histogram Equalization Transformation Data

5.9.2.1 Stepwise PMI Algorithm

GRNN

Using the data transformed by histogram equalization and the inputs identified by the stepwise PMI algorithm, a GRNN model was developed (Model 5). A plot of the 4-week forecasts obtained using this model is shown in Figure 5.55. All significant growth events were predicted by Model 5 with the exception of the 1988/1989 growth event which was a significant event, but was slightly underestimated. The timing, including the onset and duration, of all growth events was well predicted.

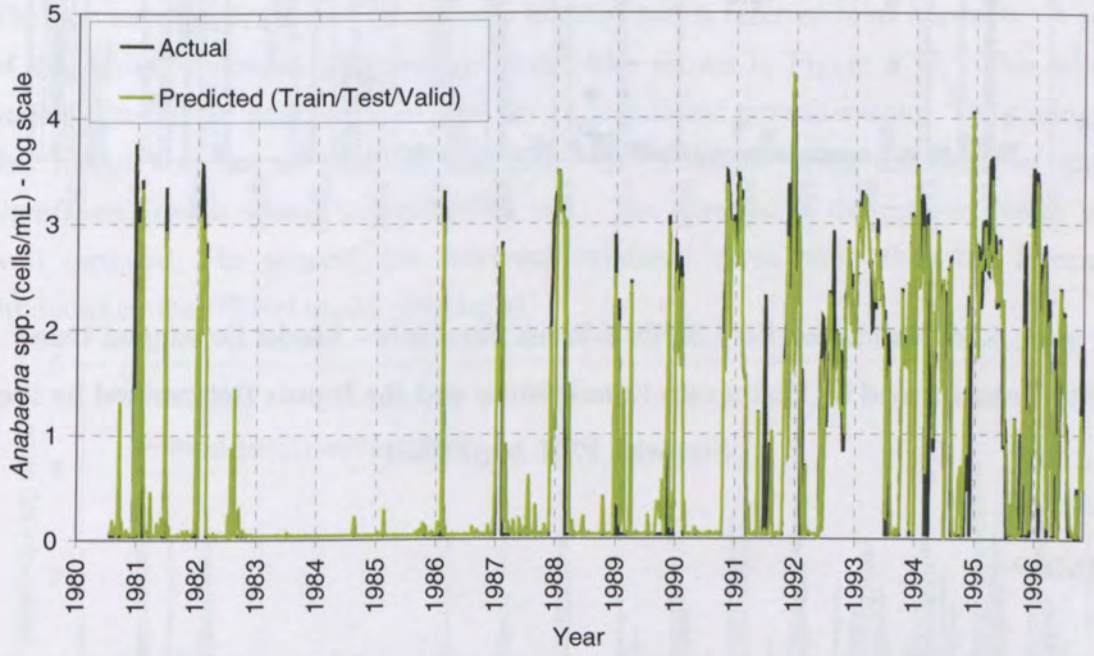

Figure 5.55 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm

The independent validation forecasts for Model 5 are presented in Figure 5.56. These results show that the growth events during 1992 to 1996 were particularly well predicted. A small growth event was erroneously predicted during the latter half of 1982 when the actual concentrations of *Anabaena* spp. were zero. This may in part be explained by the fact that Model 5 used silica lag 1 as an input. During the later half of 1982 there was a large decline in silica, which usually signals that the diatoms have

exhausted their silica supply and an *Anabaena* growth event is imminent (see Section 5.3.8). In addition, the flow in 1982 did not exhibit its usual seasonal cycle and remained low (< 8000 ML/day) for the entire year. Such low flows provide ideal conditions for growth to occur. Since the model also used flow as an input, it is not surprising that it erroneously predicted a growth event at this time.

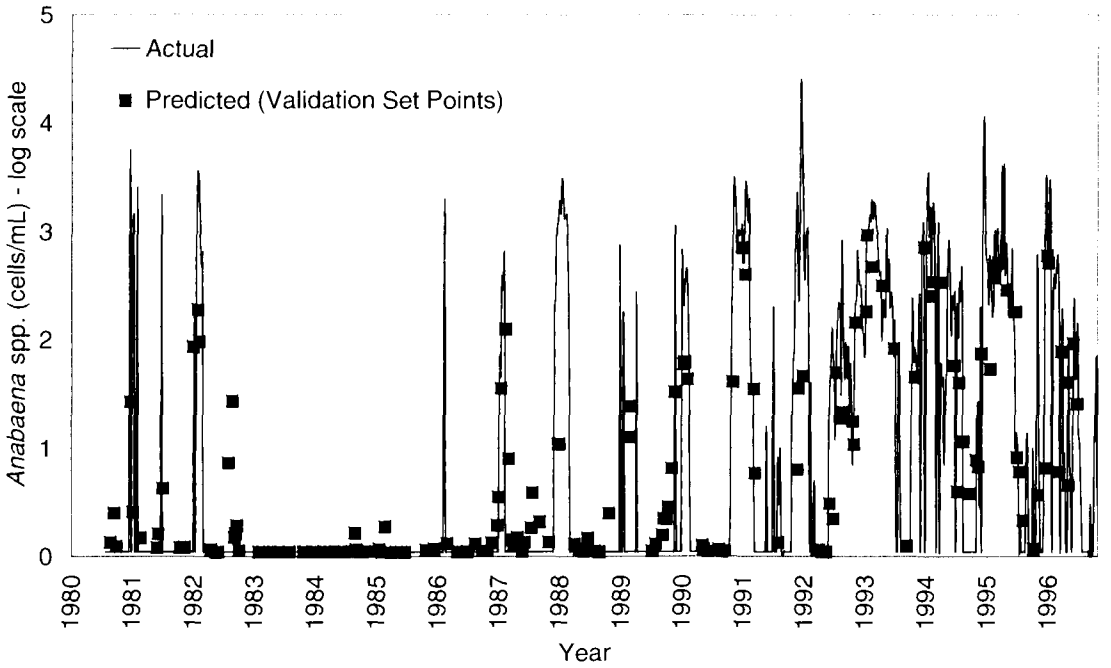

Figure 5.56 Validation Set GRNN 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm

EBMLP

Using the data transformed by histogram equalization and the inputs identified by the stepwise PMI algorithm, a number of EBMLP models were developed. Details of the ten best networks found during the GA run are shown in Table 5.19. Only two of the ten best networks utilised a second hidden layer, however, the remaining eight networks each used a large number of PEs in the first hidden layer. It is interesting to note that the aim of histogram equalization is to allow for a better mapping of the inputs to the output and this may have resulted in a second hidden layer not being required. The networks consisting of only one hidden layer each had similar network architecture.

Table 5.19 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (log scale)	Testing Set RMSE (log scale)
1	16S, 9H, 3L	<none>	1L	0.694	0.741
2	16S, 9H, 3L	<none>	1L	0.685	0.751
3	11S, 9H, 1L	<none>	1L	0.705	0.760
4	16S, 9H, 3L	<none>	1L	0.700	0.762
5	11S, 9H, 1L	<none>	1L	0.704	0.764
6	16S, 9H, 3L	<none>	1L	0.708	0.770
7	11S, 10H	4S, 2H	1L	0.731	0.771
8	11S, 9H, 1L	<none>	1L	0.687	0.777
9	11S, 9H, 1L	3S, 3H	1L	0.746	0.779
10	11S, 8H, 2L	<none>	1L	0.694	0.785

The best network from the GA run was selected and is referred to as Model 6. A plot of the 4-week forecasts obtained by Model 6 is shown in Figure 5.57. This model underestimated the peak value of 7 of the 14 significant growth events. The timing of this model was not as good as that achieved by Model 5, as the onset of many significant growth events was predicted late. The duration of the growth events was well captured. In general, the forecasts exhibited more noise than the forecasts produced by the GRNN model (Model 5)

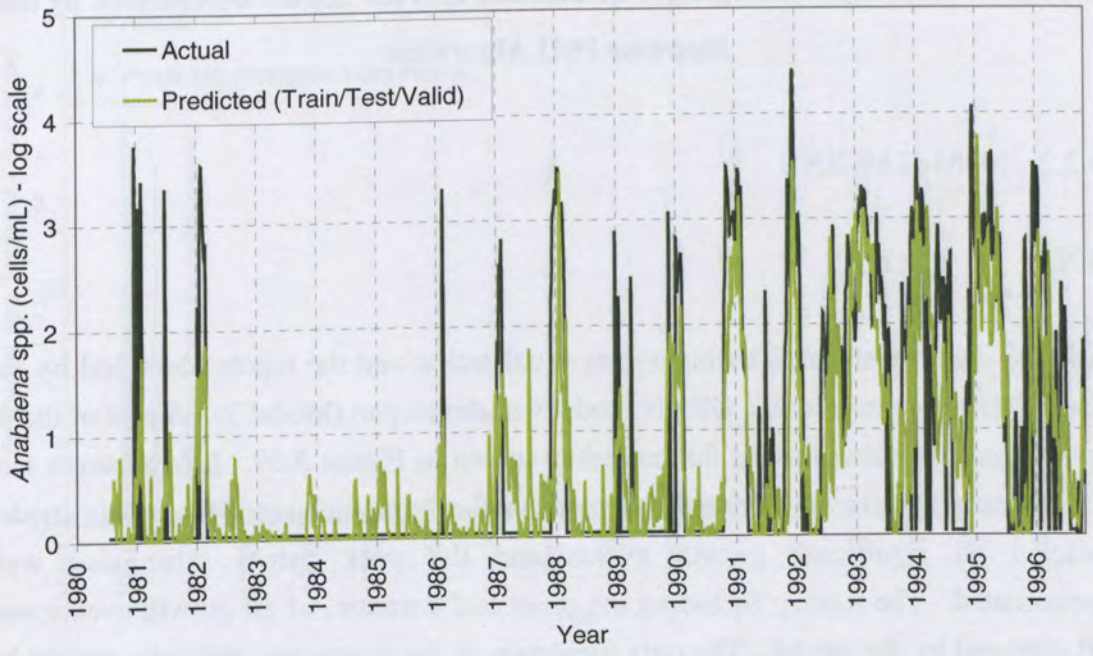

Figure 5.57 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm

The independent validation forecasts for Model 6 are plotted in Figure 5.58. Although the peak values at the end of 1991 and 1994 were better estimated by Model 6, the timing of the growth events was better predicted by Model 5.

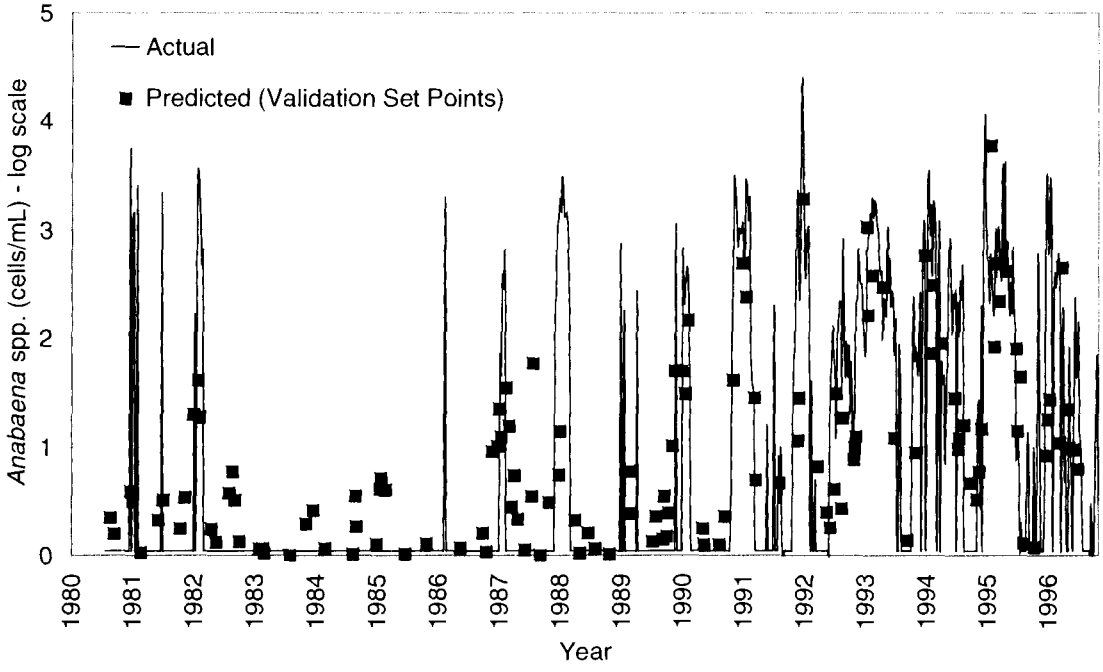

Figure 5.58 Validation Set EBMLP 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the Stepwise PMI Algorithm

5.9.2.2 SOM-GAGRNN

GRNN

Using the data transformed by histogram equalization and the inputs identified by the SOM-GAGRNN procedure, a GRNN model was developed (Model 7). A plot of the 4-week forecasts obtained using this model is shown in Figure 5.59. It is apparent that the forecasts achieved by Model 7 outperform all previous models. This model predicted all significant growth events and the peak values were also well approximated. The timing, including the onset and duration, of all growth events was well captured by the model. The only incidence of *Anabaena* spp. that was missed by the model occurred in mid 1991. A plot of the independent validation set forecasts is presented in Figure 5.60. From these results it can be observed that the model performed well on the validation data. The concentrations greater than 500 cells/mL

(i.e. 2.7 on a log scale) were well predicted and the absence of *Anabaena* spp. during 1983, 1984 and 1985 was also well predicted by this model.

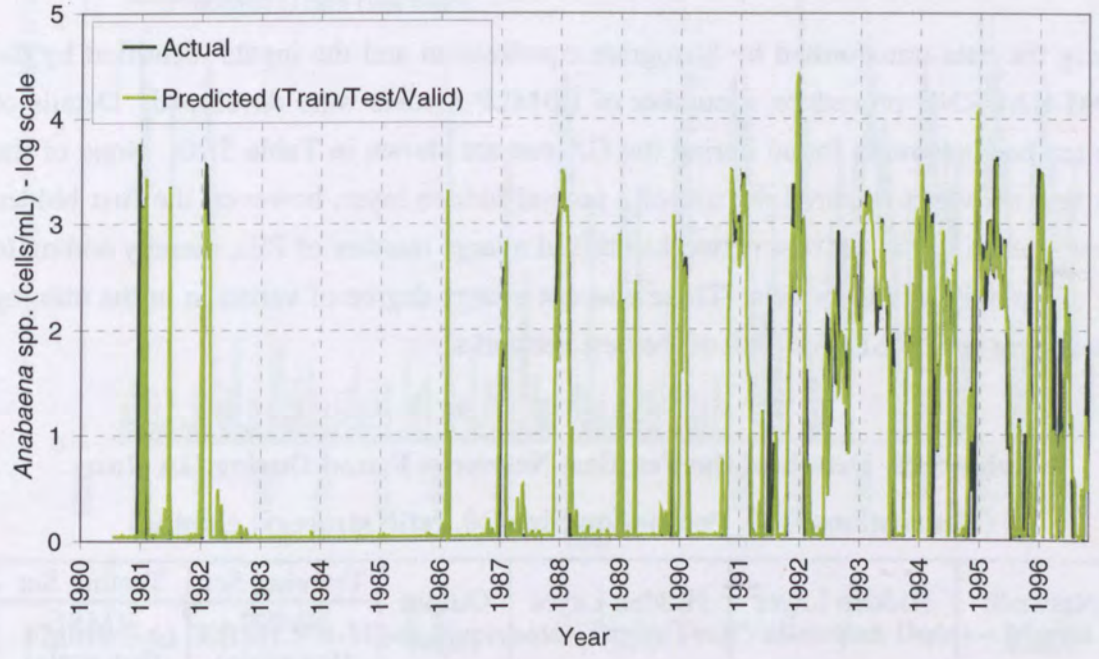

Figure 5.59 GRNN 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN

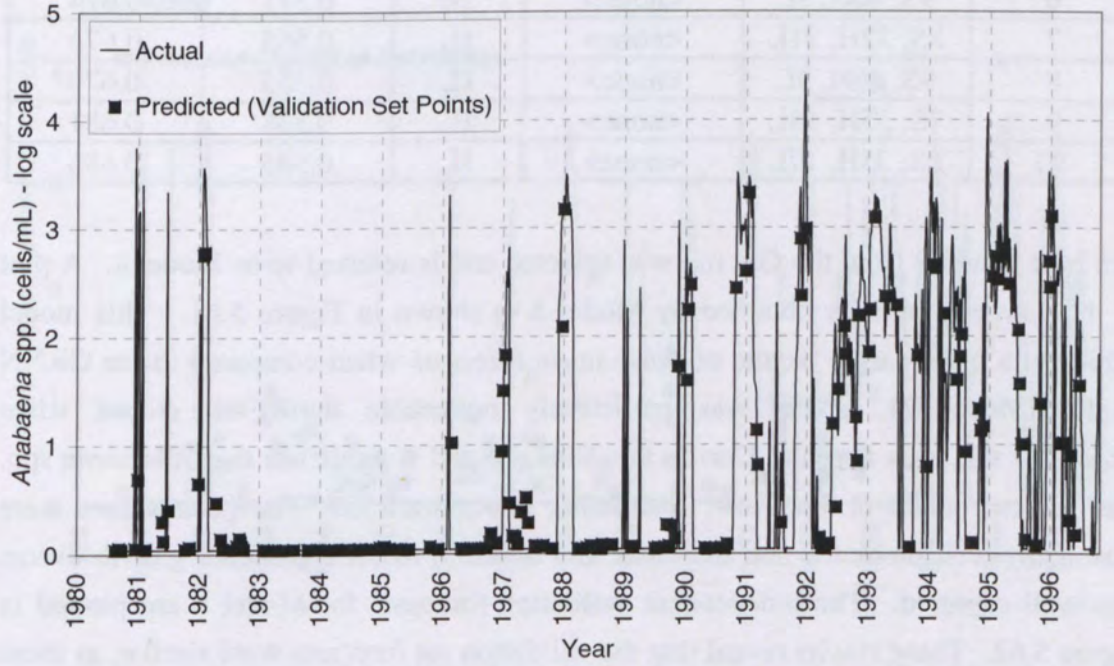

Figure 5.60 Validation Set GRNN 4-Week Forecasts – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN

EBMLP

Using the data transformed by histogram equalization and the inputs identified by the SOM-GAGRNN procedure, a number of EBMLP models were developed. Details of the ten best networks found during the GA run are shown in Table 5.20. None of the ten best networks required the use of a second hidden layer, however, the first hidden layer of each of the ten best networks utilised a large number of PEs, thereby adding to the complexity of the models. There was not a large degree of variation in the training and testing set RMSEs for each of the best networks.

Table 5.20 Details of the Ten Best Networks Found During GA Run (Generations=100, Population Size=20, refill strategy: cloning)

Network Rank	Hidden Layer 1	Hidden Layer 2	Output Layer	Training Set RMSE (log scale)	Testing Set RMSE (log scale)
1	8S, 32H, 21L	<none>	1L	0.532	0.660
2	7S, 35H, 18L	<none>	1L	0.525	0.662
3	8S, 32H, 21L	<none>	1L	0.595	0.669
4	9S, 46H, 3L	<none>	1L	0.549	0.671
5	7S, 35H, 17L	<none>	1L	0.610	0.671
6	9S, 46H, 3L	<none>	1L	0.571	0.674
7	8S, 32H, 21L	<none>	1L	0.565	0.676
8	9S, 46H, 3L	<none>	1L	0.553	0.679
9	7S, 35H, 18L	<none>	1L	0.538	0.681
10	7S, 35H, 17L	<none>	1L	0.545	0.686

The best network from the GA run was selected and is referred to as Model 8. A plot of the 4-week forecasts obtained by Model 8 is shown in Figure 5.61. This model exhibited a much larger degree of noise in its forecasts when compared to the GRNN model (Model 7). This was particularly noticeable during the times when *Anabaena* spp. was absent. During this time, Model 8 predicted that *Anabaena* spp. was present, albeit at very low, fluctuating concentrations. The peak values were reasonably well predicted and the onset and duration of all significant growth events was well captured. The independent validation forecasts for Model 8 are plotted in Figure 5.62. These results reveal that the validation set forecasts were similar, as these also displayed noise. Also evident are the over-predictions of the peak values that occurred during the growth events in 1993 and 1995.

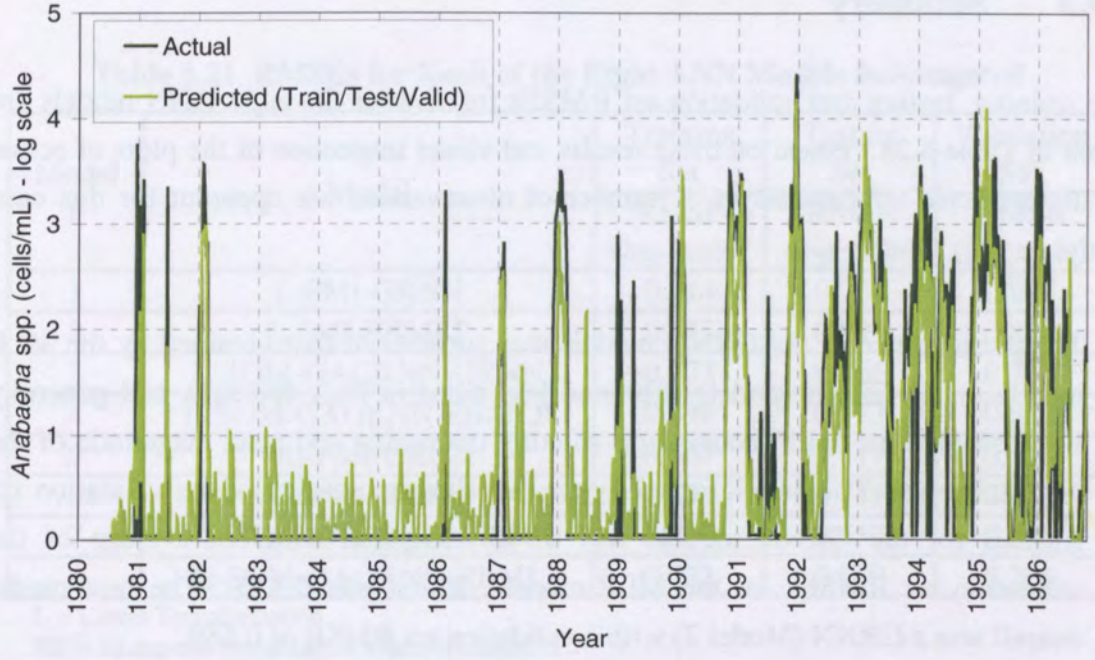

Figure 5.61 EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN

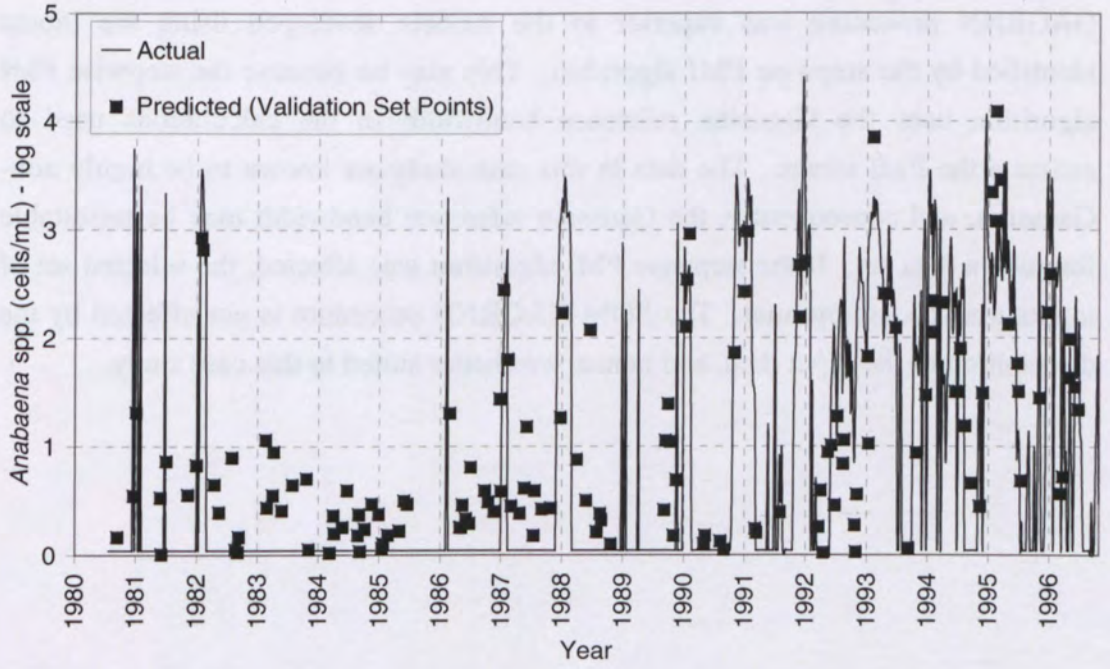

Figure 5.62 Validation Set EBMLP 4-Week Forecasts (Train/Test/Validation Data) – Model Developed Using Data Transformed by Histogram Equalization and the Inputs Determined by the SOM-GAGRNN

5.9.3 Summary

The training, testing and validation set RMSEs for each of the eight ANN models are given in Table 5.21. Based on these results and visual inspection of the plots of actual versus predicted concentrations, a number of observations are apparent for this case study:

- The performance of the GRNN models was superior to that obtained by the MLP models. The GRNN models exhibited less noise in their forecasts and generally outperformed the MLP models in predicting the timing and peak magnitude of the growth events. This is reinforced by the fact that the average of the validation set RMSEs for the GRNN models was 0.743 compared with the average of the validation set RMSEs for the MLP models, which was 0.835. The best model overall was a GRNN (Model 7) with a validation set RMSE of 0.589.
- The performance of the models developed using the data transformed using histogram equalization was superior to the models developed using the linearly transformed data. Therefore, it is apparent that this transformation was beneficial for this case study as it allowed the ANN models to better capture the underlying relationships in the data.
- The performance of the models developed using the inputs identified by the SOM-GAGRNN procedure was superior to the models developed using the inputs identified by the stepwise PMI algorithm. This may be because the stepwise PMI algorithm uses the Gaussian reference bandwidth in the calculations used to estimate the PMI scores. The data in this case study are known to be highly non-Gaussian, and consequently, the Gaussian reference bandwidth may be unsuitable for such a data set. If the stepwise PMI algorithm was affected, the selected set of inputs may be sub-optimal. The SOM-GAGRNN procedure is not affected by the distribution of the input data, and hence, was better suited to this case study.

Table 5.21 RMSEs for Each of the Eight ANN Models Investigated

Model #	Model	Training Set	Testing Set	Validation Set
		RMSE (log scale)	RMSE (log scale)	RMSE (log scale)
1	L-PMI-GRNN	0.614	0.766	0.871
2	L-PMI-EBMLP	0.733	0.699	0.852
3	L-SOM-GAGRNN-GRNN	0.177	0.666	0.737
4	L-SOM-GAGRNN-EBMLP	0.698	0.723	0.850
5	HE-PMI-GRNN	0.067	0.717	0.775
6	HE-PMI-EBMLP	0.694	0.741	0.872
7	HE-SOM-GAGRNN-GRNN	0.006	0.553	0.589
8	HE-SOM-GAGRNN-EBMLP	0.532	0.660	0.766

L = Linear Transformation
HE = Histogram Equalization Transformation

5.10 Training (Optimisation)

5.10.1 Choice of Performance Measure

In the previous section (Section 5.9), the single-sigma GRNN was found to outperform the MLP in producing forecasts of *Anabaena* spp. at Morgan, 4 weeks in advance. Therefore, the GRNN is used in this section to determine an appropriate performance measure. The data for the models developed in this section were divided into training, testing and validation sets using the GA. These data were transformed using histogram equalization and the inputs identified by the SOM-GAGRNN procedure were used (i.e. the data set used in the development of Model 7 in Section 5.9.2.2 was also used in this section).

The choice of an appropriate performance measure is case study dependent. Therefore, to determine an appropriate performance measure, the method of Diskin and Simon (1977) outlined in Section 3.7.1.3 was repeated for the *Anabaena* spp. case study. The 10 performance measures reviewed in Section 3.7.1.2 were considered as possible candidate measures. However, performance measures E_4 (average absolute percentage error) and E_5 (mean squared relative error) were excluded from this study because they are relative errors, and therefore, cannot be used when the data set has observed values equal to zero. The zero values arose in the *Anabaena* spp. data set as there were some cell counts of 1 cell/mL. Taking the logarithmic transformation of these data points gave rise to the zero values. Consequently, the remaining 8 performance measures were considered in the procedure.

Since the single-sigma GRNN trained using the inverse Hessian method was used, the 8 performance measures were incorporated directly into the GRNN training algorithm. A test set was used to determine when the inverse Hessian method had converged on a suitable value for the sigma weight. The GRNN's sigma weight was initialised to the same value for each of the 8 performance measures investigated. After each near-optimal sigma weight W_j was identified, this weight was then used to compute values P_{jk} for the 7 remaining performance measures. The results of this stage of the procedure are shown in Table 5.22.

Despite the simple nature of the optimisation problem, there was still considerable discrepancy in model performance for each performance measure. For example, for performance measure E_8, looking across the row of Table 5.22 (i.e. $k = 8$) reveals that the values ranged from 0.089, when E_8 itself was used in training the GRNN, up to 0.875, when E_3 was used in training the GRNN.

Table 5.22 Values of P_{jk} Computed By Various Performance Measures Identified By Index k Using Weight W_j - GRNN.

Note: Performance Measures 4 and 5 Were Excluded From This Study

j=	1	2	3	6	7	8	9	10
k= 1	0.240	0.276	0.426	0.276	0.267	0.240	0.248	0.270
2	0.601	0.553	0.625	0.553	0.554	0.593	0.567	0.554
3	0.025	0.022	0.017	0.022	0.023	0.025	0.025	0.023
6	0.665	0.715	0.637	0.715	0.715	0.673	0.701	0.715
7	0.696	0.723	0.653	0.723	0.724	0.701	0.718	0.724
8	0.104	0.317	0.875	0.317	0.274	0.089	0.169	0.289
9	0.180	0.187	0.283	0.187	0.182	0.177	0.174	0.184
10	0.219	0.222	0.251	0.222	0.222	0.220	0.221	0.222

In the next stage of the analysis, the actual values of P_{jk} were replaced by ranks R_{jk}. The ranks were computed according to the magnitude of the items included in each group of P_{jk} values identified by a given value of the index k. In the case where two or more entries in Table 5.22 have the same rounded value, the ranks were determined according to the value of the next significant digit not shown. The ranks for each performance measure are given in Table 5.23. Inspection of the sum of the ranks in Table 5.23 shows that the lowest sum is obtained by $j = 7$. This indicates that the sigma weight W_7, which was found using performance measure E_7 (coefficient of determination r^2), was able to produce consistently good values in many of the remaining error measures. Based on the summation values in Table 5.23, the next best performance measure was E_9 (DS#2), followed by E_8 (DS#1) and E_{10} (PLC), which both had summation values of 32. It is evident that when performance measure E_3 (minimax) was used in training the GRNN, the resulting model consistently performed poorly when measured using each of the other performance measures. This is expected due to the 'spikey' nature of the *Anabaena* spp. data. The minimax performance measure is a measure of the maximum discrepancy between actual and predicted values. With data such as *Anabaena* spp. concentrations, the maximum discrepancy between actual and predicted values is most likely to occur at one of the large peaks in the testing set. Attempting to model one peak well does not guarantee that the remaining data points are also well modelled. From these results it is apparent that attempting to model a particular peak in the testing set came at the expense of being able to model the remaining data to a suitable level of accuracy.

Table 5.23 Ranks From Pjk Values Given in Table 5.22 - GRNN. Note: Performance Measures 4 and 5 Were Excluded From This Study

j =	1	2	3	6	7	8	9	10
k = 1	1	7	8	6	4	2	3	5
2	7	1	8	2	4	6	5	3
3	8	2	1	3	5	7	6	4
6	7	2	8	1	4	6	5	3
7	7	4	8	3	1	6	5	2
8	2	7	8	6	4	1	3	5
9	3	7	8	6	4	2	1	5
10	1	7	8	6	4	2	3	5
Total	36	37	57	33	30	32	31	32

5.10.2 Choice of Optimisation Method

Training a single-sigma GRNN is relatively straightforward, therefore, it is not useful to compare different training algorithms for this type of network. However, an alternative network investigated in this study is the multiple-sigma GRNN. From the results obtained in Section 4.10.2.2, both the GA and the MMAS provided suitable methods for training multiple-sigma GRNN models. The MMAS is used in this study as it required fewer iterations to converge on the best solution. The MMAS uses a number of parameters that are case study dependent and need to be determined by sensitivity analyses.

The data set used in Section 5.10.1 (i.e. data divided by the GA, transformed using histogram equalization with the inputs selected using the SOM-GAGRNN) was also used in this investigation. The best performance measure identified in Section 5.10.1 was the coefficient of determination r^2, and this was used to assess the generalisation ability of all models. The sigma weight lower bound σ_j^- was set at 0 and the upper bound σ_j^+ was set at 1 for each of the 27 weights in the network. To determine suitable values for each parameter, it was necessary to start with a set of initial default values. The values of each parameter determined in the sensitivity analyses conducted on the salinity data set (Section 4.10.2.2) were used as a starting point in this study and are shown in Table 5.24. 100 ant cycles (iterations) were used for the sensitivity analyses.

Table 5.24 Default MMAS Parameters

Parameter	Description	Value
b	Number of strata	100
τ_{min}	The minimum pheromone trail	20
ρ	Pheromone persistence coefficient	0.6
m	Number of ants in population	80

5.10.2.1 Number of strata

Three discretisation sizes were investigated, including 100, 1000 and 10000. Given the 27 inputs in this case study, this produced corresponding search spaces of 10^{54}, 10^{81} and 10^{108}. The results obtained when using the different discretisation sizes are given in Table 5.25. In comparison with the salinity data set (Table 4.51), increasing the number of strata had a more significant impact on the overall runtime due to the larger number of inputs in this case study. Like the results obtained for the salinity data, using more strata resulted in a corresponding decrease in test set generalisation ability. As

discussed in Section 4.10.2.2, this is because the same number of ants are used to search a much larger search space as the number of strata increases. However, the reduction in generalisation ability was very small for this case study.

Table 5.25 Run Times and Training and Testing Set Performance For Each Number of Strata

Number of Strata	Run Time (s)	Training Set r^2	Testing Set r^2
100	1071	1.000	0.776
1000	1118	1.000	0.771
10000	1843	1.000	0.770

5.10.2.2 The minimum pheromone trail

The minimum pheromone trail is a parameter introduced in the MMAS to help prevent search stagnation. The effect of varying the minimum pheromone trail was considered in this study (Table 5.26). It can be seen that the best model performance was obtained when the minimum pheromone trail was set at 5. Increasing the value beyond 5 tended to result in a reduction in generalisation ability when measured using the test set.

Table 5.26 Training and Testing Set Performance For Different Values of The Minimum Pheromone Trail

Min. Pheromone Trail τ_{min}	Training Set r^2	Testing Set r^2
5	1.000	0.806
10	1.000	0.785
15	1.000	0.779
20	1.000	0.776
25	1.000	0.776
50	1.000	0.772
100	1.000	0.774
200	1.000	0.773
500	1.000	0.773

5.10.2.3 Pheromone persistence coefficient

Table 5.27 shows the performance of the MMAS in training the GRNN when different values of the pheromone persistence coefficient are used. This parameter controls how quickly the algorithm converges on a solution. It is evident that the best test set generalisation ability was achieved when the pheromone persistence coefficient was equal to 0.9. At such a high value, the amount of evaporation of pheromone is quite

small, therefore, convergence is likely to be faster. This indicates that for this case study, the MMAS had no difficulty in converging relatively quickly on a good solution. However, using no evaporation (i.e. $\rho = 1.0$), resulted in premature convergence on a solution that was not as good. Therefore, a small degree of evaporation was required to obtain good model performance for this case study.

Table 5.27 Training and Testing Set Performance for Different Values of the Pheromone Persistence Coefficient

Pheromone Persistence Coefficient ρ	Training Set r^2	Testing Set r^2
0.1	1.000	0.791
0.2	1.000	0.789
0.3	1.000	0.791
0.4	1.000	0.790
0.5	1.000	0.794
0.6	1.000	0.806
0.7	1.000	0.796
0.8	1.000	0.804
0.9	1.000	0.845
1.0	1.000	0.773

5.10.2.4 Number of ants in the population

To investigate the effect that the number of ants has on the model's performance, a range of values were tested (Table 5.28). The runtime was found to be linearly related to the number of ants used by the MMAS. Beyond a population size of 80, the extra computational effort was not rewarded with any significant improvement in generalisation ability. The best performance on the test set was achieved using a population size of 200 ants.

Table 5.28 Run Times and Training and Testing Set Performance For Different Population Sizes

Number of Ants	Run Time (s)	Training Set	Testing Set
		r^2	r^2
20	264	1.000	0.807
25	333	1.000	0.816
30	414	1.000	0.814
40	532	1.000	0.808
50	664	1.000	0.824
70	931	1.000	0.826
80	1082	1.000	0.845
100	1527	1.000	0.837
150	1984	1.000	0.844
200	2642	1.000	0.851
300	3982	1.000	0.845

5.10.2.5 Summary

The values of each parameter determined from the sensitivity analyses are given in Table 5.29. The multiple-sigma GRNN trained using MMAS is compared with the single-sigma GRNN trained using the inverse Hessian method in Table 5.30. It is evident that the multiple-sigma GRNN outperformed the single-sigma GRNN when measured by the test set r^2. However, in the next section various model deployment strategies are investigated and these require the model to be retrained many times. The improvement in test set performance obtained by the multiple-sigma GRNN was not enough to offset the extra computational effort that is required to train the model. Therefore, the single-sigma GRNN was adopted for the model deployment trials.

Table 5.29 MMAS Parameters Determined Using Sensitivity Analyses

Parameter	Description	Value
b	Number of strata	100
τ_{min}	The minimum pheromone trail	5
ρ	Pheromone persistence coefficient	0.9
m	Number of ants in population	200

Table 5.30 Comparison of the Training and Testing Set Performance for the Single-Sigma and Multiple-Sigma GRNNs

Training Method	Training Set	Testing Set	Validation Set
	r^2	r^2	r^2
Single-Sigma GRNN	1.000	0.723	0.767
Multiple-Sigma GRNN (MMAS)	1.000	0.851	0.711

Another approach for gaining insight into the relative importance of each of the input variables is to analyse each of the sigma weights in the multiple-sigma GRNN (Doan and Liong, 2002). A plot of the sigma weights obtained by the MMAS for each input in the GRNN is shown in Figure 5.63. From this plot it can be seen that the most significant input is temperature lag 14 with a sigma weight of 0.985. The important effect that temperature has on populations of *Anabaena* is well documented and has been confirmed in this study. This input was closely followed in its relative importance by flow lag 8 and pH lag 14. Flow and pH are also known to have a large effect on *Anabaena* populations and these relationships have also been confirmed. *Anabaena* and silica at lag 11 also had large sigma weights, indicating that these inputs also contribute significantly to the forecasts. Adopting an arbitrary level of significance of 0.1, there were only two inputs with sigma weights smaller than this level. These were colour lag 13 and river level lag 20. However, both the colour and river level variables had other lags that were found to be significant.

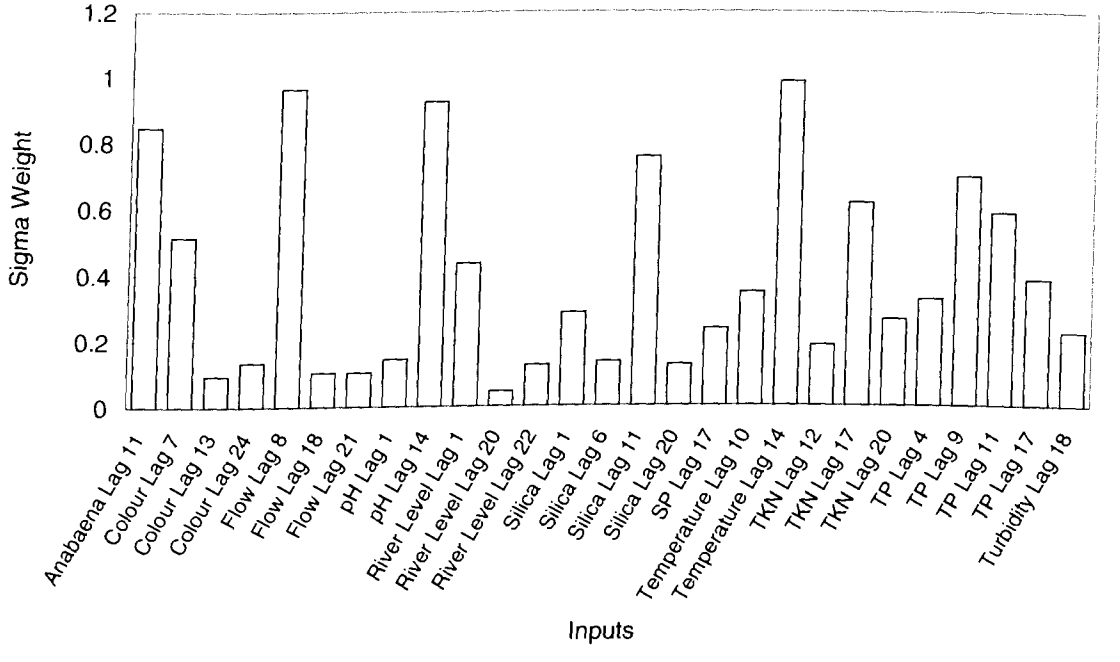

Figure 5.63 Sigma Weight For Each Input in the MMAS-GRNN

5.11 Model Deployment

The optimised single-sigma GRNN was used to investigate different retraining regimes for models deployed in an operational environment. The three retraining scenarios outlined in Section 3.8 were considered, including: no retraining; retraining after each sample; and selectively retraining the model when some information is available to suggest that a data point is uncharacteristic (i.e. lies outside of the domain of the training patterns). To evaluate the three retraining options, the second validation set data were used. After the appropriate lags had been taken into account for each input variable, this data set consisted of 192 samples spanning the period 20-05-1998 to 16-01-2002. These data were used to simulate a real-time forecasting scenario. The GRNN outlier detection method described in Section 3.8.2.2 was used to detect uncharacteristic data.

The first scenario investigated in this study was the effect of not retraining the GRNN model. The results for the real-time forecasting test period are presented in Figure 5.64. It can be seen that there were very few occasions when *Anabaena* spp. were absent during the second validation period. For most of this period, many of the growth events occurred in quick succession with only short intervals between each event. The GRNN developed using the data from 1980 to 1996 was not able to model this feature. Instead, the model predicted two main growth events: one occurring during late 1999 and the other spanning 2000/2001. However, these events did not coincide with the actual growth events. These results indicate that the model developed using the data from 1980 to 1996 was unable to model the underlying relationships in the second validation data.

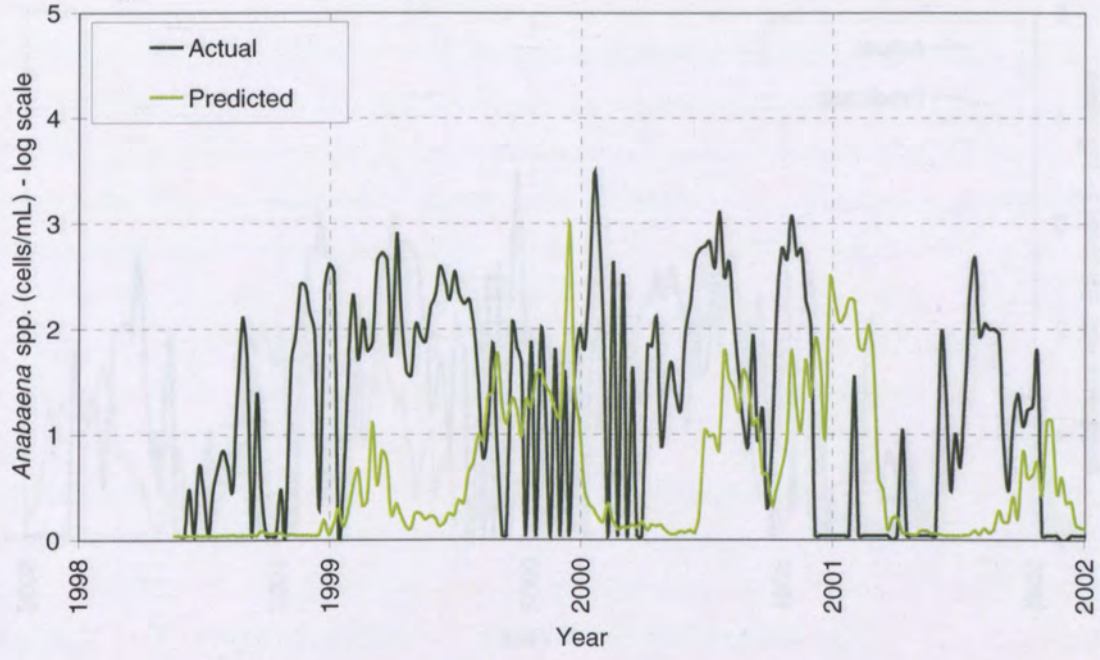

Figure 5.64 Results of Not Retraining the GRNN Model for the Real-Time Forecasting Test Period (May 1998 to January 2002)

The second scenario investigated was the effect of retraining the GRNN after each new sample is obtained. However, to simulate an on-line operational model, 4 weeks must elapse before retraining of the model can take place. This is because in reality, the output value (i.e. concentration of *Anabaena* spp. 4 weeks into the future) must be sampled and counted before it would be possible to use that value in retraining the ANN. The results of this retraining scenario for the real-time forecasting test period are shown in Figure 5.65. It is evident that retraining the model after each sample improved the predictions, however, the model was still incapable of predicting the relative magnitude and timing of the growth events in the second validation set.

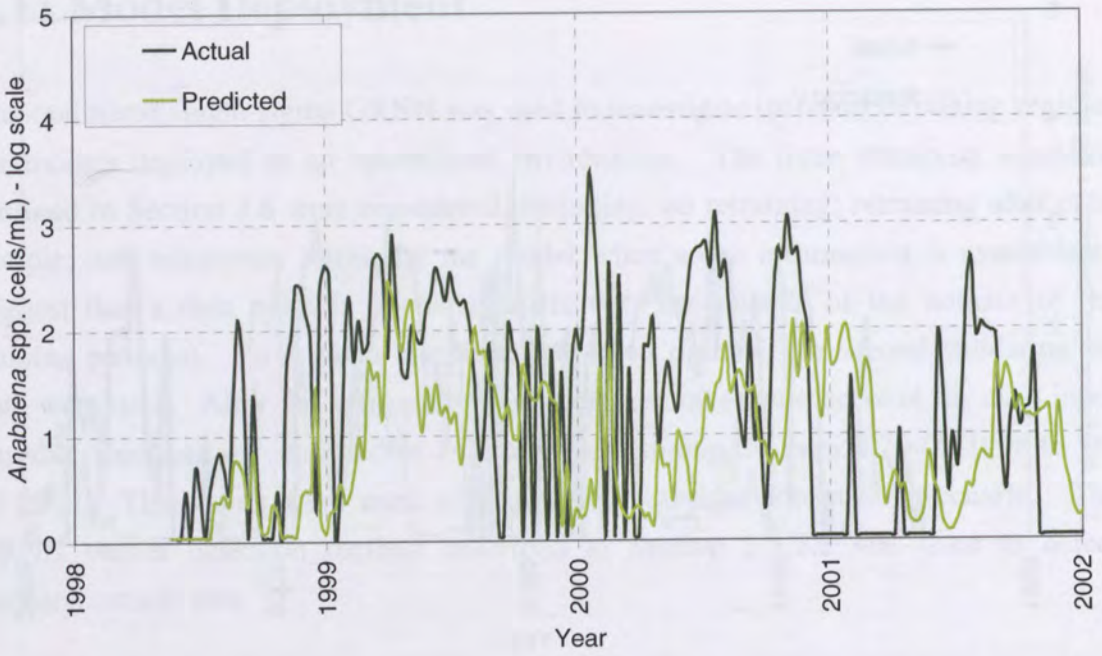

Figure 5.65 Results of Retraining the GRNN Model After Each Sample for the Real-Time Forecasting Test Period (May 1998 to January 2002)

To further investigate why retraining the model after each sample did not cause a more considerable improvement in the forecasts, an additional experiment was conducted. In this experiment, the GRNN was retrained after each sample without waiting the 4 weeks to update the model (Figure 5.66). The results of this experiment indicate that the GRNN was able to predict the relative magnitude and timing of each of the major growth events in the second validation set. This experiment is artificial as it is not possible in a real-world situation to retrain the model without waiting the 4 weeks, however, it shows that if these types of patterns were included in the calibration data, then the model has no difficulty in predicting these events. Even randomly selecting 50% of the samples (without replacement) from the second validation set, placing them in the calibration set, and retraining the GRNN produced very good forecasts of the remaining second validation patterns (Figure 5.67). This provides further evidence that the second validation patterns are not adequately represented in the calibration data, and that if they were, the forecasts would improve considerably. For the real-time forecasting test period, retraining is not sufficient to improve the forecasts markedly because the system was constantly shifting to a state that had not been seen previously by the model.

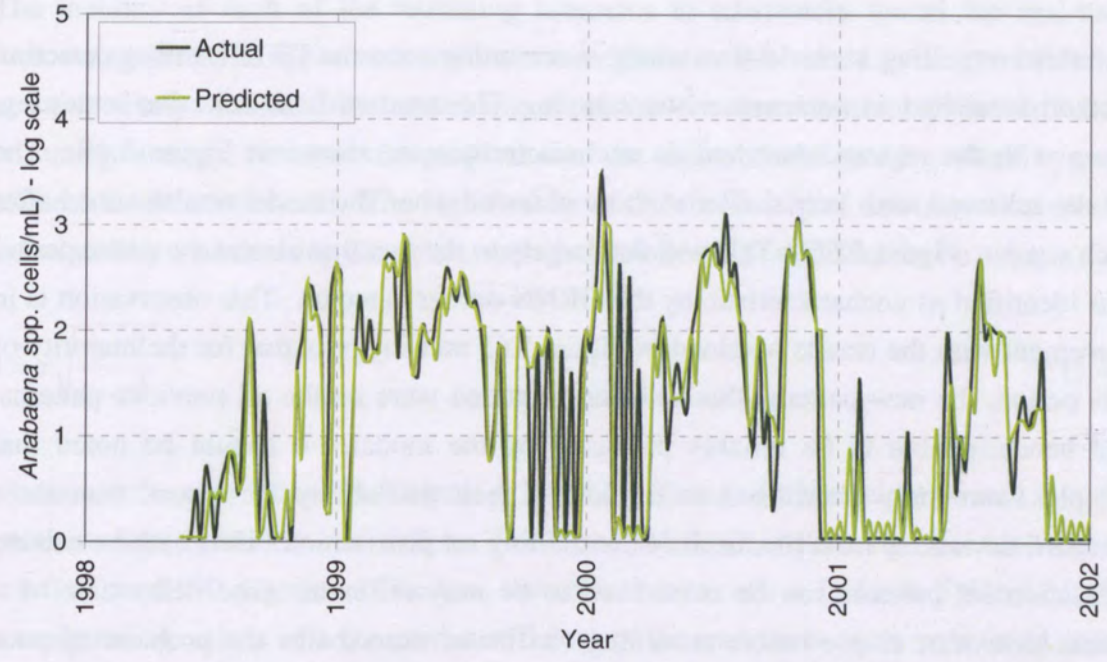

Figure 5.66 Results of Retraining the GRNN Model After Each Sample Without Waiting 4 Weeks to update - Real-Time Forecasting Test Period (May 1998 to January 2002)

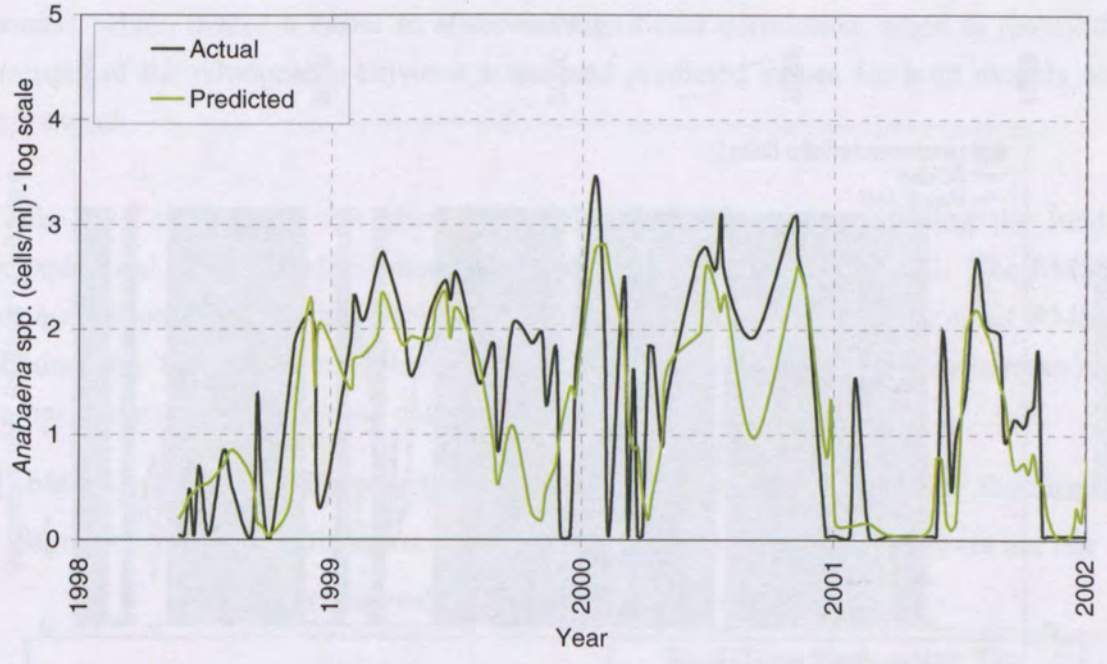

Figure 5.67 Results of Retraining the GRNN Model With 50 Per Cent of the Second Validation Set Samples Randomly Removed and Added to the Calibration Data (Remaining Patterns From May 1998 to January 2002)

The third retraining scenario was selective retraining once the GRNN outlier detection method identified an uncharacteristic sample. The results of the selective retraining, along with the regions identified as uncharacteristic are shown in Figure 5.68. The results achieved were very similar to those obtained when the model was retrained after each sample (Figure 5.65). This was due largely to the fact that almost the entire period was identified as uncharacteristic by the GRNN outlier detector. This observation is in agreement with the results obtained in Figure 5.65 and suggests that for the majority of this period, the new patterns that were encountered were unlike all previous patterns, and hence, unable to be reliably predicted by the model. It should be noted that samples were only considered as outliers if their probability was more than three standard deviations from the mean of the training set distribution. Hence, these second validation set patterns can be considered to be very different. The delay time of 4 weeks that must elapse before retraining can occur exacerbates the problem of poor prediction since the closest calibration data point to any new data point will be the sample from 4 weeks previous. Due to the exponential growth and decay of *Anabaena* populations, it is quite conceivable that the system has changed its state considerably in this intervening time. Consequently, for this data set, retraining does not help the model for the next forecast because of the delay time, as discussed previously.

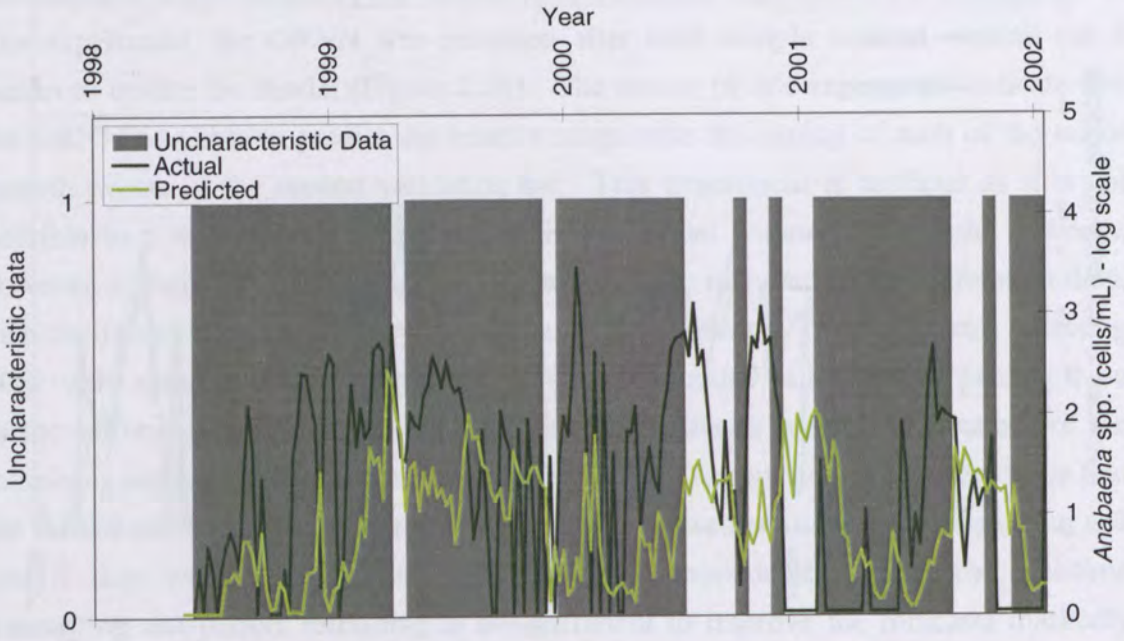

Figure 5.68 Results of Retraining the GRNN Model Using the GRNN Outlier Detection Method for the Real-Time Forecasting Test Period (May 1998 to January 2002)

The inability of each of the retraining scenarios to adequately model the real-time forecasting period is apparent from the results given in Table 5.31. Always retraining and selectively retraining the model improved the r^2 value when compared to the no retraining strategy, however, the proportion of the observed variance explained by each of these models was small. The significance of the coefficient of determination can be calculated by transforming the value, such that it follows a *t*-distribution by using:

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{5.4}$$

where r is the correlation coefficient, r^2 is the coefficient of determination and n is the number of data points in the sample. For a one-sided test, with a significance level of $\alpha = 0.05$, and 190 degrees of freedom, the critical value of $t = 1.66$. In Table 5.31, the corresponding values of t for each coefficient of determination were calculated using Equation 5.4 and are given in brackets. For the model developed using the no retraining strategy, it is apparent that the observed value of t was less than the critical t. Hence, the null hypothesis of no relationship between the actual and predicted values (i.e. H_0: $r^2 = 0.0$) cannot be rejected for this model. The continuous and selective retraining strategies produced models with a significant coefficient of determination. However, this was most likely due to the relatively large sample size (i.e. 192 data points), which makes it easier to observe a significant correlation, when in reality the strength of the relationship between actual and predicted values for both models was quite weak.

Table 5.31 also shows that always retraining and selectively retraining the model helped to reduced the RMSE when compared to not retraining the model. The RMSEs for each strategy were considerably larger than the testing and validation set RMSEs obtained for the GRNN models in Table 5.30. This was due to the uncharacteristic nature of the second validation data as discussed above.

Table 5.31 Real-Time Forecasting Test Period Results Obtained Using the Single-Sigma GRNN With Different Retraining Strategies (values in brackets are the corresponding *t*-values for each r^2 value)

Retraining Strategy	Real-Time Forecasting Test Period (Second Validation Set)	
	r^2	RMSE (log scale)
No Retraining	0.005 (0.98)	1.412
Always Retrain	0.032 (2.51)	1.136
Selective Retraining (Outlier Detector)	0.017 (1.81)	1.157

To further verify that the second validation data were uncharacteristic, a number of statistical tests and comparisons were performed. Firstly, the univariate time series of log(*Anabaena* spp.) was considered. A plot of the autocorrelation function for the training/testing/validation data and the second validation data is shown in Figure 5.69. As expected, there is a large peak in the training/testing/validation data autocorrelation function that occurs at a lag of 52 weeks and a smaller peak that occurs at lag of 104 weeks. This is due to the seasonality present in the data and the fact that during this period, growth events tended to occur at the same time each year (i.e. November through to April). A positive correlation at lag 52 and 104 weeks was not observed for the second validation data, which was indicative of the lack of seasonal variation in this data set.

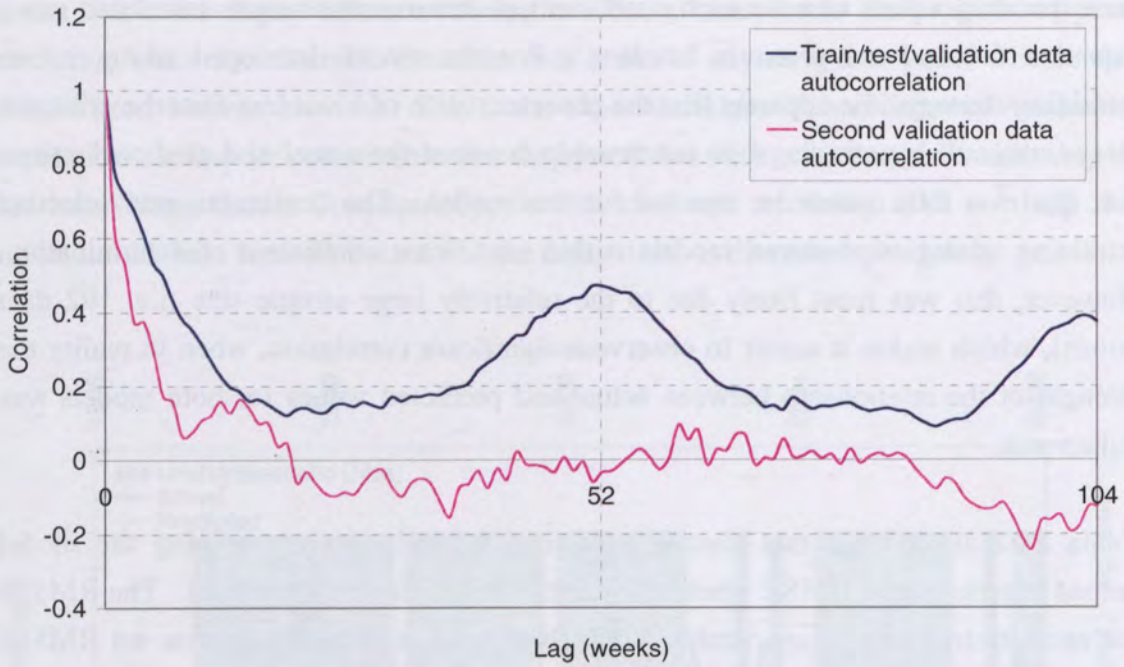

Figure 5.69 A Comparison of the Autocorrelation Function for the Train/Test/Validation Data and the Second Validation Data

A *t*-test was performed to test the null hypothesis that the means of the two data sets were equal (i.e. $\mu_1 = \mu_2$). For an equal-tails test, with a significance level $\alpha = 0.05$, and 1100 degrees of freedom, the critical value of $t = 1.96$. The calculated t statistic was 6.98 for these data sets, which indicates that the null hypothesis of equivalent means was rejected. A *F*-test was conducted to test the null hypothesis that the ratio of the variances σ_1^2 / σ_2^2 of the two data sets is 1 (i.e. $\sigma_1 = \sigma_2$). For an equal-tails test, with a significance level $\alpha = 0.05$, $f_1 = 876$ degrees of freedom and $f_2 = 191$ degrees of

freedom, the critical value of F = 1.26. The calculated F statistic was 1.38 for these data sets, which indicates that the null hypothesis of equivalent variances was rejected.

The log(*Anabaena* spp.) data used in this study are known to be highly non-Gaussian. Even for non-normally distributed data, the Central Limit Theorem suggests that the t-test and F-test will produce valid results when the sample size is large (e.g. >40 samples). However, as a precautionary measure, the Kolmogorov-Smirnov (K-S) test was also used to compare the two data sets because it is not based on any assumptions regarding the underlying distribution of the data being tested. The K-S test statistic is a measure of the maximum discrepancy between the empirical distributions of two samples. Comparing the log(*Anabaena* spp.) training/testing/validation data with the second validation data, the K-S statistic was calculated to be 0.42. Given this K-S statistic, the corresponding probability that the two data sets were drawn from the same population was calculated as 0.0. This result is in agreement with the results obtained for both the t-test and the F-test and suggests that the log(*Anabaena* spp.) data from the two time periods are statistically different.

The BDS test was also used to compare the log(*Anabaena* spp.) data from the two data sets (Table 5.32). It is apparent that the calculated BDS Z statistics for the second validation data were considerably smaller than those calculated for the training/testing/validation data at each setting of the embedding dimension. The null hypothesis of linearity was still rejected for the second validation set, indicating that like the training/testing/validation data, the second validation data also exhibit nonlinear serial dependence.

Table 5.32 A Comparison of the BDS Test Z Statistics Obtained for the Train/Test/Validation Data and the Second Validation Data

Training/testing/validation set data			Second validation set data		
Embedding Dimension	BDS Z Statistic	Decision	Embedding Dimension	BDS Z Statistic	Decision
2	15.33	Reject linearity (strongly)	2	6.48	Reject linearity (strongly)
3	16.63		3	7.09	
4	17.56		4	8.36	
5	18.67		5	8.76	
6	19.87		6	9.92	
7	21.65		7	10.71	
8	23.96		8	11.57	

Kaboudan's FCS was also applied to the second validation data and a comparison with the diagnosis results obtained for the training/testing/validation data is given in Table

5.33. Unlike the training/testing/validation data, the second validation data only possessed a weakly linear component and were filtered using an ARMA(1,0) model. The θ statistic obtained for the was significantly higher than that calculated for the training/testing/validation data and a medium level of noise was also found to be associated with these second validation data. This provides further evidence of the difference between the two time periods and offers another possible explanation of the model's poor performance for the second validation period.

Table 5.33 A Comparison of the FCS Diagnosis Results for the Train/Test Validation Data and the Second Validation Data

Data Set	Fitted ARMA Order (p, q)	r^2	θ	Final Diagnosis
Train/test/validation	(1, 1)	0.64	0.19	FL-NL[a]
Second Validation	(1, 0)	0.37	0.80	WL-NL-MN[b]

[a]Fairly linear - nonlinear
[b]Weakly linear - nonlinear - medium noise

After comparing and testing the univariate time series of *Anabaena* spp. for the training/testing/validation period and the second validation period, it was considered important to examine the multivariate relationships in the two periods. The first experiment involved clustering the data from the two time periods using a SOM. The SOM was implemented using the *NCS NeuFrame* software. A 10 by 10 Kohonen layer was used and the SOM was trained for 1000 epochs. The software default values for the learning rate and neighbourhood size were used. Inspection of the resulting clusters revealed that the second validation data clustered by themselves. Each cluster did not contain any samples from the training/testing/validation period. From this result, it can be concluded that the multivariate relationships were considerably different for the two time periods.

The second experiment involved calculating the cross-correlation function between *Anabaena* spp. and each input variable for the two time periods. The results of this investigation are presented in Figure 5.70. It is evident that the cross-correlation between *Anabaena* spp. and each of the ten input variables was considerably different for the two time periods. This concurs with the findings obtained using the SOM clustering and further explains why the model had difficulty in obtaining suitable forecasts for the second validation data. Since the relationship between each input variable and *Anabaena* spp. was different for the second validation data, it is only to be expected that the ANN would be unable to model the underlying relationships. As discussed previously, retraining the model was not sufficient to enable the model to

adjust to these new patterns, since the system was constantly shifting to a state previously unseen.

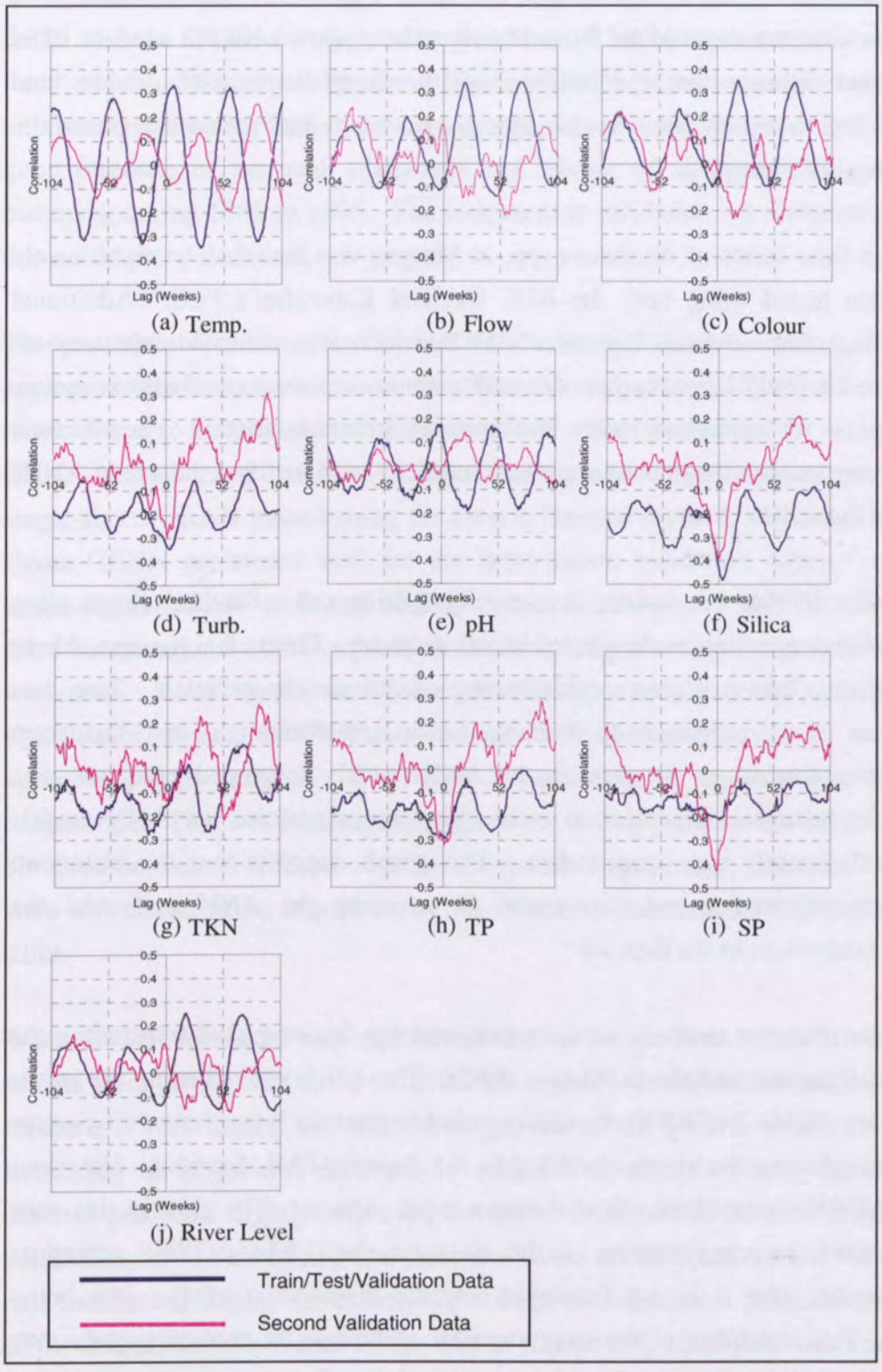

Figure 5.70 Comparison of Training/Testing/Validation Data Cross Correlation and Second Validation Data Cross Correlation Between *Anabaena* spp. and (a) Temperature, (b) Flow, (c) Colour, (d) Turbidity, (e) pH, (f) Silica, (g) TKN, (h) Total Phosphorus, (i) Soluble Phosphorus, and (j) River Level

5.12 Summary and Conclusions

In this chapter, the ANN methodology has been applied to the forecasting of *Anabaena* spp. concentrations in the River Murray at Morgan, 4 weeks in advance. The results obtained indicate that the relationships between the causal variables and cyanobacteria concentrations are very complex and not easily interpreted in terms of the underlying physical processes.

The univariate time series of *Anabaena* spp. at Morgan was found to be significantly nonlinear when tested using both the BDS test and Kaboudan's FCS. Additional information was obtained from Kaboudan's FCS, which also indicated that the time series contained a fairly linear component and a nonlinear component with no noise. This finding is in agreement with the general observation that cyanobacteria populations are capable of exponential growth and decay and justified the use of ANNs for modelling these data.

The multivariate data set was divided into training, testing and validation subsets using the GA data division technique developed in this research. This technique was able to produce three data sets that were representative of the same population. Two data transformations were investigated, namely, linear transformation and histogram equalization transformation. In general, the ANN models developed using the data transformed by histogram equalization were found to outperform the ANN models developed with linearly transformed data. This result suggests that the histogram equalization transformation was successful in assisting the ANN to model the underlying relationships in the data set.

Two input determination methods were investigated for this case study, including the stepwise PMI algorithm and the SOM-GAGRNN. The ANNs trained using the inputs identified by the SOM-GAGRNN procedure tended to produce lower forecasting errors than those trained using the inputs identified by the stepwise PMI algorithm. However, the SOM-GAGRNN procedure selected larger input subsets. The data in this case study are known to be non-Gaussian. In this situation, the SOM-GAGRNN procedure has the advantage that it is not dependent on the distribution of the data being investigated. The suitability of the stepwise PMI algorithm for this case study may have been impeded by the non-Gaussian nature of the data because the Gaussian reference bandwidth is used in the PMI calculations.

Both MLP and GRNN models were investigated for the *Anabaena* spp. case study. To determine appropriate network architecture for the MLP models, an EBMLP was used.

The GRNN models were found to outperform the EBMLP models in predicting the timing and peak magnitude of the growth events for this case study. The GRNN models also exhibited less noise in their forecasts. The best ANN model overall was found to be a GRNN using the data transformed by histogram equalization and the input subset determined by the SOM-GAGRNN procedure. Despite the large uncertainty associated with the *Anabaena* spp. data, this model was still able to provide good forecasts of the peak magnitude and timing of all significant growth events occurring during 1980 to 1996. The independent validation set forecasts produced by this model were also good.

The procedure formulated by Diskin and Simon (1977) was used to determine an appropriate performance measure. For this case study, the coefficient of determination r^2 was found to be the most suitable measure of model performance. A multiple-sigma GRNN trained using the MMAS algorithm was developed and compared with the single-sigma GRNN trained using the inverse Hessian method. Although the multiple-sigma GRNN performed well for the independent validation data ($r^2 = 0.711$), the single-sigma GRNN produced slightly better forecasts ($r^2 = 0.767$). Therefore, the extra computational effort required in implementing a multiple-sigma GRNN was not warranted. However, the multiple-sigma GRNN was able to offer insight into the significance of each of the input variables. This was achieved by examining the optimised sigma weights. The most important input variable was found to be temperature. This was not surprising given the strong positive relationship commonly observed between *Anabaena* concentrations and temperature. The other variables that were also found to be important for the forecasts included flow, pH, *Anabaena* and silica.

When the GRNN model was applied to a second independent validation set consisting of data from 20-05-1998 to 16-01-2002, the model was not able to produce acceptable forecasts of the timing and relative magnitude of the growth events in this period. Retraining the model after each sample or selectively retraining using the GRNN outlier detector produced slightly improved forecasts. Analysis of the data from the second validation period determined that these data were uncharacteristic and were unlike the data used in calibrating the model. To attempt to determine a potential physical cause of the uncharacteristic nature of the data in this period, an inquiry was conducted into the sampling procedure used to collect these data. The results of this inquiry revealed that there had been a change in the sampling procedure during the time the second validation data were collected (i.e. 1998 to 2002). Instead of surface grab samples being taken (as was the case for the training, testing and validation data), the samples had been pumped from depth. Since *Anabaena* spp. can control their position in the

water column via buoyancy regulation, this may have contributed to the samples in the second validation period being diagnosed as uncharacteristic. This further highlights some of the problems involved in using historical data.

For the second validation period, continuously retraining the GRNN only marginally improved model performance. This was due to the associated lag time that must elapse before retraining can commence and the fact that the system was constantly shifting to a new state. However, when the data from the second validation period were included in the training and testing sets, model performance increased markedly.

For the system to be modelled adequately, the ANN model needs to be trained on the full range of patterns that are likely to occur. Even for the relatively extensive data set used for training, testing and validation in this research (i.e. 17 years), the second validation set revealed that other patterns or events are possible that were not included in the original data collected from 1980 to 1996. The most likely cause of this is the complex manner in which the causal variables interact with concentrations of *Anabaena* and the numerous possible states for the system. The problem was further exacerbated by a change in the sampling procedure that may have affected the data from the second validation period.

Chapter 6

Conclusions and Recommendations

6.1 Contributions of the Research

The following contributions have been made in this thesis:

1. Methods and guidelines for determining when ANN models should be used.

   Before using nonlinear modelling methodologies, such as ANNs, it is first necessary to ask if such techniques are justified by the data. While many hydrological processes may seem unlikely to be linear *a priori*, their nonlinear nature may not manifest itself in specific aspects of their dynamics. In such instances, a model that preserves linear serial dependence may constitute an adequate approximation of the process. Recent advances in statistics have given rise to a number of algorithms suitable for addressing whether nonlinear modelling techniques are justified by the data. If a time series is found to be nonlinear and the particular model form is unknown, ANNs are suitable candidates as they are flexible nonlinear models, which avoid the need for any particular model form to be specified *a priori*. However, most ANN researchers simply apply ANNs to their

application without first considering the complexity of their time series data. In this research, two state-of-the-art nonlinearity tests from the statistics literature (the BDS test and Kaboudan's Fuzzy Classification System) were applied to the salinity and cyanobacteria case studies. These techniques were developed as tools for testing if a data set contains evidence of nonlinearity and hence, whether ANNs should be used in preference to conventional linear time series models for the data set under investigation.

2. Development of data division techniques.

ANNs (like other statistical and empirical models) cannot extrapolate beyond the range of the data used for calibration. Therefore, in order to develop the best ANN model, given the available data, all of the patterns that are contained in the data need to be included in the calibration set. Most of the papers reviewed arbitrarily divided the available data into subsets. If the statistical properties of the calibration and validation data are not taken into account, the quality of the model developed is "pot luck" and it is not possible to rigorously assess the generalisation ability of the model. Choosing the calibration data arbitrarily, without any knowledge of whether all extreme patterns have been included, will most likely lead to a sub-optimal model. If such a model is then validated using a data set that does contain the extreme data points that were excluded from the calibration data set, the model cannot be expected to perform well, as the validation data will test the model's extrapolation ability, and not its interpolation ability. In addition, by choosing calibration and validation data arbitrarily, without any knowledge of which types of patterns have been included in either set, the quality of the model developed, and hence the performance of the model on the validation data, has a large random component associated with it. It follows that if all of the patterns that are contained in the available data should be contained in the calibration set, then the toughest evaluation of the generalisation ability of the model is if all of the patterns (and not just a subset) are contained in the validation data. On this premise, two data division methods were developed in this research to address the lack of a suitable procedure for data division in the literature. One method that was developed utilised a genetic algorithm (GA) to divide the data into subsets so as to minimise the statistical difference (as measured by the mean and standard deviation) between training, testing and validation data sets and ensure that the extreme values were placed in the training set. The other method that was developed utilised a self-organizing map (SOM) to cluster the available data and then select representative points for each subset from each cluster. Both the GA and SOM data division

methods were applied to the salinity case study. The GA data division method was used for the cyanobacteria case study.

3. Guidelines for transforming data for ANN models and the development of new data transformation methods.

   Another area that has been largely overlooked in the literature is the effect of transforming an ANN's inputs and outputs. In this research, a transformation was developed for removing seasonality from time series data and the effect of removing the seasonal component of the data on the ANN's performance was investigated. A transformation was also developed consisting of a two-step numerical procedure to transform data distributed on any range to normality. The effect of using normally distributed inputs and outputs was also investigated. A new data transformation known as kernel transformation was developed in this research to transform data to uniformity. This transformation is based on kernel density estimation techniques. Estimating the densities using a kernel estimator has the advantage over a histogram in that it provides a smooth, continuous probability estimate. Linear, logarithmic, histogram equalization, kernel, seasonal and normality transformations were applied to the salinity data set. The linear and histogram equalization transformations were applied to the cyanobacteria case study. A comparison was made of the ANN's performance when using these various transformations.

4. Development of methods for selecting appropriate ANN model inputs.

   In this research three techniques were developed for determining an optimal subset of inputs for ANN models. The first technique was a variation of the stepwise PMI algorithm proposed by Sharma (2000). This technique is a *model-free* approach, as it does not require the construction of a model of the relationship between the two variables. In addition, it is able to capture nonlinear dependencies between two variables and can provide useful insights into the relative significance of each of the inputs identified. This research represents the first use of this algorithm for determining significant ANN inputs. Improvements were made to the original algorithm, including the use of the computationally efficient city block distance kernel in the density estimation technique and the use of general regression neural networks (GRNNs) to compute conditional expectations. GRNNs do not impose any restrictive assumptions on the form of the underlying regression model and are a flexible and easy to implement method for estimating the residuals. In the second input determination method developed in this research (PCA-GAGRNN), an

unsupervised technique (i.e principal component analysis) was used to reduce the dimensionality of the input space and a GAGRNN was used to select the final subset of important model inputs. The third method developed for input determination utilised a SOM to remove redundant inputs and a GAGRNN to select the final set of model inputs. All three input determination methods were applied to the salinity data set and the stepwise PMI algorithm and SOM-GAGRNN were also applied to the cyanobacteria data set.

5. The evaluation of two feedforward ANNs (GRNN and multilayer perceptron (MLP)) and methods for determining network architecture.

   In a review of 152 journal papers on the application of ANNs to water resources variables (published before the end of 2001), there were no applications that made use of the GRNN. In this research, the GRNN was considered as an alternative to the commonly used MLP model trained using the backpropagation algorithm and its use in water resources modelling was evaluated. The inverse Hessian method was developed as a technique for training single-sigma GRNNs. Variations of the standard single-sigma GRNN architecture were also investigated, including:

   - Clustering the training data to reduce the number of pattern layer PEs.
   - Using a separate sigma weight for each input in the network (i.e. multiple-sigma GRNN).

   An evolutionary method (i.e. GA) was used to optimise the architecture for the MLP model trained using the backpropagation algorithm (EBMLP). The effectiveness of the GRNN and the EBMLP were evaluated and compared for both the salinity and cyanobacteria case studies.

6. Guidelines for selecting suitable performance measures.

   Guidelines for the use of a number of commonly used performance measures in hydrological modelling were provided and the procedure outlined by Diskin and Simon (1977) was applied for the determination of a suitable performance measure. This represents the first use of this procedure in the context of determining an appropriate performance measure for ANN modelling. The procedure was applied to determine appropriate performance measures for both the salinity and cyanobacteria case studies.

7. Evaluation of training (optimisation) methods for MLP models.

   Six training methods (four first-order and two second-order methods) were evaluated for optimising the connection weights in a feedforward MLP. The first-order methods investigated were: the generalized delta rule; the normalized cumulative delta (NCD) rule; the delta-bar-delta (DBD) algorithm; and, the extended delta-bar-delta (EDBD) algorithm. The QuickProp (QProp) algorithm and the MaxProp (MProp) algorithm were the second-order methods investigated.

8. The application of ant colony optimisation for training multiple-sigma GRNN models.

   As an alternative to the GA, the *Max-Min* Ant System (MMAS) was used to train a multiple-sigma GRNN. A procedure consisting of a number of sensitivity analyses was developed for determining optimal values of the MMAS parameters. The multiple-sigma GRNN trained using MMAS was applied to both the salinity and cyanobacteria case studies.

9. Two new techniques for deploying an operational ANN model.

   A framework for deploying ANN models was developed in this research. When a real-world ANN model has been developed, it is usually deployed in an operational environment and new input patterns are collected in order to make real-time forecasts. However, because ANNs are unable to reliably extrapolate beyond the calibration range, there is a need to determine if these new input patterns are similar to the input data used in training the model. In order to address this problem, a novel hybrid forecasting model consisting of a SOM and a MLP trained using the backpropagation algorithm was developed (SOM-MLP). The hybrid model was successful at identifying when uncharacteristic input patterns were encountered. An additional method for detecting uncharacteristic data was developed for the GRNN. This method involved a statistical test based on the GRNN's B summation unit and was also successful in diagnosing uncharacteristic patterns. When uncharacteristic data were detected, both the MLP and GRNN models were then retrained to incorporate these patterns. In this way, the models were able to adapt to new information as it was encountered. The hybrid SOM-MLP and the GRNN outlier detector were both applied to the salinity data. The GRNN outlier detector was also applied to the cyanobacteria case study. For both case studies, the methods were compared to alternative retraining strategies, such as not retraining the model and retraining the model after each new sample.

10. The ANN modelling methodology developed in this research led to models that were successful in forecasting water quality parameters in the River Murray.

    The GRNN and MLP models were shown to produce very good 14-day forecasts of salinity in the River Murray at Murray Bridge. These forecasts can be used to reduce the salinity of Adelaide's water supply. The GRNN was shown to produce good 4-week forecasts of *Anabaena* spp. in the River Murray at Morgan, providing that representative data are provided in the calibration set. These forecasts are able to provide important information for water management authorities as they can be used to give warning of impending growth events.

6.2 General Conclusions

1. The two nonlinearity tests investigated in this research (i.e. Kaboudan's FCS and the BDS test) correctly detected the presence or absence of nonlinear dynamics in four synthetic data sets. Both nonlinearity tests were applied to the salinity and cyanobacteria case studies and were found to exhibit consistency in their inference for these data sets. The information obtained from these tests can be used to decide whether it is appropriate to use ANNs or linear methods as a modelling tool for the data set under investigation.

2. The GA and SOM data division methods were able to produce statistically representative subsets and consequently, the models developed using these subsets were able to produce consistent forecasting results. Both of these data division techniques reduce the likelihood of overfitting the data in the training set because overfitting only occurs when the training set is not totally representative of the population. The SOM data division method has the advantage that the proportion of samples to assign to the training, testing and validation sets does not need to be determined. The disadvantages of the SOM method include:

    - It is difficult to decide on a suitable size for the Kohonen layer and hence, the number of clusters that should be used.
    - A decision has to be made regarding which samples to discard.

    The GA data division method has the advantage that it can ensure that the extreme values are placed in the training set by using penalty constraints. The disadvantage of the GA method is that the proportion of samples to assign to the training, testing and validation sets needs to be chosen.

3. The most effective ANN data transformation was found to be case study dependent. The following findings were determined, however, these may not be indicative of all case studies:

   - Removing the deterministic seasonal component from data and transforming the inputs and outputs to normality were found to give significantly larger forecasting errors than a simple linear transformation. It is believed that these transformations distorted the original relationships between variables in a way that was not beneficial to ANN learning.
   - The models developed using the data transformed by histogram equalization and kernel transformation were found to provide superior performance when tested on data within the training domain, but were not as robust as the models developed using linearly transformed data when tested on data outside of the training domain.
   - The use of the mean squared error (MSE) as the objective function during training was found to be valid as it maximises the likelihood of the model under the assumption of normal random shocks. An analysis of the residuals produced by the ANN models trained using linearly transformed data showed that this assumption was satisfied.

4. It is well-known that extraneous inputs increase the likelihood of local minima in the error surface and require more training data to efficiently optimise the connection weights. Therefore, when no *a priori* knowledge is available to suggest possible inputs, and for complex problems where the number of potential inputs is large, it is a significant advantage to use an analytical method capable of selecting the important model inputs. The following three methods were found to be appropriate for determining inputs to ANN models:

   i. The stepwise PMI algorithm, which is a *model-free* approach and has demonstrated that it is able to identify the actual model inputs to a number of test problems. This method has the following advantages:

      - It is able to account for nonlinear relationships in the data.
      - It is able to account for redundant (highly correlated) inputs.
      - It can provide insight into the underlying process as the magnitude of the PMI scores give useful information regarding the relative importance of each input.

The disadvantages of this approach include:

- A significance level must be adopted for selecting inputs.
- An appropriate kernel bandwidth must be determined for this method. For highly non-Gaussian data, the Gaussian reference bandwidth adopted in this study may be unsuitable, and this can lead to a sub-optimal set of selected model inputs.

ii. The PCA-GAGRNN, which uses principal component analysis (PCA) as the unsupervised input reduction method and a GAGRNN as the supervised method of input determination. This method had the following advantages:

- A significance level does not need to be selected.
- PCA is easy to implement and many guidelines are in existence for this method.

The disadvantages of this method include:

- Each principal component (PC) is a weighted average of all candidate inputs and therefore, is likely to contain extraneous inputs.
- Only linear dependence between variables is considered in PCA.
- The number of PCs to retain must be selected.
- The use of a GA introduces additional parameters (e.g. number of generations and population size) and an appropriate objective function also needs to be selected.
- The supervised component of this method (i.e. GAGRNN) is *model-based* and hence, requires construction of a model. If the model does not provide a good representation of the relationships between the candidate inputs and the output, then the resulting set of selected inputs are likely to be sub-optimal.

iii. The SOM-GAGRNN, which is similar to the previous method but uses a SOM as the unsupervised input reduction method. This method had the following advantages:

- A significance level does not need to be selected.
- There are no limiting assumptions about the dependence in the data set (i.e. nonlinear relationships can be accounted for).

- There are no limiting restrictions about the way the variables must be distributed (i.e. non-Gaussian distributed inputs are acceptable).

The SOM-GAGRNN has the following disadvantages:

- The size of the Kohonen layer must be selected.
- The disadvantages of the GAGRNN approach listed above.

An additional disadvantage of the SOM-GAGRNN is that the unsupervised input reduction technique (i.e. SOM) does not guarantee the actual model inputs will be selected because highly correlated inputs may be selected if they are closest to the cluster centre. Despite this, when the SOM-GAGRNN was applied to synthetic test problems, it was still able to achieve similar predictive performance to the models developed using the inputs identified by the stepwise PMI algorithm. Consequently, both approaches are suitable when predictive performance is the primary aim. However, the stepwise PMI algorithm should be used when insight into the underlying process is of utmost importance, as this method has demonstrated that it is able to identify the actual model inputs to a number of test problems.

5. When tested on data within the calibration domain, the GRNN was found to be superior to the MLP. However, MLPs were found to be more robust when uncharacteristic patterns were encountered. No advantage was obtained by clustering the training data to reduce the number of pattern layer PEs in the GRNN. Likewise, there was no significant performance advantage in using a multiple-sigma GRNN when compared to the single-sigma GRNN. The weights obtained in the multiple-sigma GRNN can be used to gain insight into the relevant importance of each input in the network.

6. Using a GA to automatically evolve an MLP's architecture resulted in networks with improved generalisation ability when compared to trial-and-error methods. In some cases, two hidden layers were required despite the fact that MLPs with one hidden layer and sufficient hidden layer nodes have been theoretically shown to be universal approximators (Cybenko, 1989).

7. The choice of a suitable performance measure was found to be dependent on the type of ANN model used and the case study under investigation. Therefore, the procedure outlined by Diskin and Simon (1977) should be applied to each case study investigated and for each type of ANN model considered.

8. The generalisation ability of the MLPs trained using second-order algorithms was found to be inferior to the models trained using first-order algorithms. The second-order algorithms exhibited a reduced ability to escape local minima in the error surface.

9. The *Max-Min* Ant System (MMAS) was found to be an efficient method for training a multiple-sigma GRNN, requiring considerably fewer iterations to converge on an optimal solution compared to the multiple-sigma GRNN trained using a GA. The generalisation ability of the GRNNs trained using MMAS and the GA were found to be very similar.

10. The following two retraining methodologies were found to be appropriate for deploying ANN models in a real-world environment:

    i. The hybrid SOM-MLP model. The front-end SOM component of the hybrid model was successful at diagnosing uncharacteristic patterns and identifying when retraining was required.

    ii. The GRNN outlier detector. A statistical test was performed on the GRNN's *B* summation unit to detect patterns with densities that were more than three standard deviations from the mean of the training set densities. This approach was also found to be successful in detecting uncharacteristic patterns and identifying when retraining of the model was necessary.

    Both methods provided a significant improvement when compared to the scenario of not retraining the model.

6.3 Specific Conclusions

The specific conclusions of this research relate to the case studies used to illustrate the ANN modelling methodology.

6.3.1 Case Study I: Salinity in the River Murray

1. Kaboudan's FCS and the BDS test both detected nonlinear serial dependence in the time series of salinity levels in the River Murray at Murray Bridge, thereby justifying the use of ANNs for this case study.

2. A logarithmic transformation of the data did not improve the ANN's performance for this case study. This was believed to be caused by the compressing nature of the logarithmic transformation, which may have distorted the information contained in the flow and river level variables.

3. The stepwise PMI algorithm was found to be the best input determination method for the salinity case study. The 960 candidate inputs were reduced to 13 significant model inputs using this method. By examining the PMI scores obtained for each input, it was determined that the salinity at Mannum at a lag of 1 day was the most important input for the 14-day forecast at Murray Bridge. The second and third most significant inputs were Waikerie salinity at a lag of 1 day and Lock 7 flow at a lag of 1 day, respectively.

4. For this case study, the average absolute percentage error (AAPE) was found to be the most suitable performance measure for MLPs, while the RMSE was the most suitable for GRNNs.

5. The GRNN models were found to significantly outperform the MLP models. Although the multiple-sigma GRNN produced slightly better performance than the single-sigma GRNN, the latter was selected as the best model overall because it was much simpler to implement. For the training, testing and validation data, the single-sigma GRNN achieved RMSEs of 1.1 EC units, 11.1 EC units and 14.5 EC units, respectively.

6. The EDBD algorithm was found to be the most effective training algorithm for the MLP models developed in this case study. The training, testing and validation set AAPEs for the best MLP model trained using this algorithm were 2.3%, 3.0% and 3.0%, respectively.

7. When tested on the second validation data, the best result overall was achieved by the GRNN which retrained after each new sample. However, this result was only slightly better than the GRNN that was selectively retrained using the GRNN outlier detector. This was despite the fact that the GRNN outlier detector only needed to be retrained 23.1% of the time.

6.3.2 Case Study II: Cyanobacteria in the River Murray

1. Kaboudan's FCS and the BDS test both detected nonlinear serial dependence in the time series of log transformed *Anabaena* spp. concentrations in the River Murray at Morgan, thereby justifying the use of ANNs for this case study.

2. The histogram equalisation transformation was successful in improving the ANN's performance when compared to a linear transformation of the data.

3. The SOM-GAGRNN was the best input determination method for this case study. Using this method, the 286 candidate inputs were reduced to a subset of 21 final inputs.

4. For this case study, the coefficient of determination r^2 was found to be the most suitable performance measure for use with the GRNN.

5. The GRNN models developed for this case study outperformed the MLP models. As was the case for the salinity data set, the multiple-sigma GRNN provided slight improvement over the single-sigma GRNN, however, this improvement was not significant enough to warrant the extra computational effort required. The best single-sigma GRNN achieved training, testing and validation set r^2 values of 1.000, 0.723 and 0.767, respectively. Good performance on the independent validation set was obtained despite the fact that there is a large degree of uncertainty inherent in the *Anabaena* spp. data. The precision of the cell counts alone can vary from ±20% to as high as ±50%. The ability of the GRNN to obtain a r^2 of 0.767 on the independent validation set is testimony to the robustness of ANNs as a modelling tool.

6. Examining the sigma weights obtained in the multiple-sigma GRNN revealed that the most important input for this case study was temperature at a lag of 14 weeks, followed by flow at a lag of 8 weeks and pH at a lag of 14 weeks. Due to the complex relationships involved in this case study, physical interpretation of the results was very difficult.

7. The data used for the second validation data set (20-05-1998 to 16-01-2002) were found to be significantly different to the data used for model development (08-01-1980 to 20-11-1996). This was confirmed using a number of statistical tests. Further inquiry into a potential physical cause revealed that there had been a change in the sampling procedure at the time the second validation data were collected.

Instead of surface grab samples, the samples had been pumped from depth during this period. Since *Anabaena* spp. can control their position in the water column via buoyancy regulation, this may have contributed to the samples in the second validation period being diagnosed as uncharacteristic.

8. When the GRNN outlier detector was used, almost the entire second validation period was found to be uncharacteristic. Due to the associated lag time before retraining can commence and given the fact that the system can change rapidly to a new state in the intervening time, retraining the model only slightly improved the forecasts. However, when data from the second validation period were included in the calibration data, the GRNN's performance improved markedly.

6.4 Recommendations for Future Work

1. The performance of ANNs and linear time series models needs to be compared using data that belong to different time series classifications (e.g. linear and nonlinear time series data), as determined by the two nonlinearity tests.

2. In this research it was suggested that the θ statistic obtained in Kaboudan's FCS could be used to estimate the signal-to-noise ratio of a data set and hence, the optimal number of hidden layer nodes required by an ANN. The utility of this approach needs to be investigated for a number of synthetic and real-world data sets.

3. A difficulty noted in this research when using the SOM was the selection of an appropriate size for the Kohonen layer. Trial-and-error methods are the most commonly used approach for this task. Alternative methods for determining the size of the Kohonen layer should be investigated, such as the growing grid SOM proposed by Fritzke (1995), which is able to dynamically adapt its topology to suit the data distribution.

4. In this research, the Gaussian reference bandwidth was used in the stepwise PMI algorithm. If the input data are highly non-Gaussian, this may lead to a sub-optimal set of selected model inputs. Therefore, to determine the impact on the method's ability to select significant inputs, the stepwise PMI algorithm (with the Gaussian reference bandwidth) should be tested on highly non-Gaussian data, where the actual model inputs are known *a priori*. Alternative methods for selecting an appropriate bandwidth should also be considered for non-Gaussian data, such as those proposed by Harrold et al. (2001).

5. The ANNs that have been used in the past to model water resources variables have been almost entirely deterministic in nature. However, water resources models are subject to both model and parameter uncertainties. In addition, the data used in the development of such models are also subject to inherent uncertainties due to the stochastic nature of hydrological processes (Loucks et al., 1981) and any systematic error that may have occurred during the sampling of these data. Therefore, future research efforts should be directed towards dealing with these forms of uncertainty. Techniques such as Monte Carlo Simulation or First-Order Analysis should be considered in the ANN model development process. Bayesian learning (Neal, 1996) should also be considered as an approach for addressing uncertainty in ANN models.

6. This research has highlighted some of the difficulties associated with using historical data sets. In particular, the *Anabaena* spp. data used in this research possessed a high degree of uncertainty resulting from sampling and counting errors. These data were collected using surface grab samples at a fixed location, and as such, were never intended for use in an ecological study. Consequently, for future ecological studies, it is recommended that a sampling procedure be instituted that better captures the true population of cyanobacteria in the river. The spatial patchiness exhibited by cyanobacteria should be accounted for by taking an integrated depth sample at a number of points across the cross-section of the river. In each sample, enough trichomes (cell colonies) should be counted to ensure that the precision of the cell counting process is ±20% or less.

7. The ANN methodology developed in this research should be applied to other water resources case studies to further determine the generality of the guidelines and approaches presented in this thesis.

Appendix A

Notation

a_k	autoregressive coefficients
$A^i(k)$	A summation unit weights for cluster i after k observations
b	non uniform mutation system parameter determining the degree of dependency on the iteration number
b_m	moving average coefficients
$B^i(k)$	B summation unit weights for cluster i after k observations
$C(m, \varepsilon, n)$	correlation integral
d	dimension of the multivariate variable set
$\det(\cdot)$	determinant operation
d_k	desired output for output PE Y_k
D	maximum dimension of the Kohonen layer's columns and rows
D_j	distance computed for the *jth* pattern layer PE in a GRNN
e_k	error information term for kth output PE
e_t	random noise component of a time series at discrete time t
ein_j	backpropagated error for jth hidden PE
e_j	error information term for jth hidden PE
E	performance measure (objective function)
$E(k)$	error at iteration k

$E[\cdot]$	expectation operation
E_{min}	minimum error
$\hat{f}_X(\boldsymbol{x})$	multivariate kernel density estimate of the d-dimensional variable set $\boldsymbol{X}$ at coordinate location $\boldsymbol{x}$
$\hat{f}(x)$	univariate kernel density estimate at coordinate location x
$f_X(x)$	marginal PDF of X
$f_{X,Y}(x,y)$	joint (bivariate) PDF of X and Y
$f_Y(y)$	marginal PDF of Y
F_j	frequency at which PE j has historically won in a SOM
$g(\overline{\sigma}^{gb})$	solution cost of the global-best solution in MMAS
$g(\overline{\sigma}^{best})$	solution cost of the iteration-best solution in MMAS
g	performance measure
$g(x)$	original (raw) series
h	bandwidth (also called the window width or smoothing parameter)
$h(\cdot)$	link function in a regression model
$h^{(t)}$	gradient at time t
$H\|_{\sigma_0}$	Hessian evaluated at σ_0
H_N	histogram equalization transformation for the *Nth* input
$H(\cdot)$	Heaviside unit step function
I^*	average amount of light available for phytoplankton entrained in the vertical mixing of the water body or (under thermal stratification conditions) in the epilimnion
I_m	light intensity at the bottom of the mixed layer of water
I_P	the pth interval in the data series
I_Z	light intensity at depth Z
I_o	light intensity at the water surface
k_{max}	maximum lag of the inputs
$K(\cdot)$	kernel function centred at each data point
$L^{(t)}$	linear term at time t in the QProp algorithm weight update equation
L_k	low flow criterion
m	variable
M	number of Kohonen PEs
M_P	number of values that belong to the pth interval in the data series
n	number of observations in the data set
n_l	number of flows lower than one-third of the mean flow observed
n_p	number of peak flows greater than one-third of the mean peak flow observed
N	variable
N_c	SOM neighbourhood set

N^H	number of hidden layer nodes
N^I	number of input layer nodes
N_i	set of one-step neighbours of node i in MMAS
N^O	number of output layer nodes
N^{TR}	number of training samples
O_i	ith observed value (an element of a set of measured data)
$\overline{O}$	mean of the observed values
p	order of the autoregressive parameters
$p_{ij}^k(t)$	probability with which ant k should choose node $j \in N_i$ as the next node to move to at time t in MMAS
$\overline{P}$	mean of the predicted values
P_i	ith predicted model output
P_k	peak flow criterion
$P_X(x_i)$	univariate probability density of X
$P_{X,Y}(x_i,y_i)$	joint probability density of X and Y
$P_Y(y_i)$	univariate probability density of Y
q	order of the moving average parameters
Q_1	first quartile (25th percentile)
Q_3	third quartile (75th percentile)
$Q^{(t)}$	quadratic term at time t in the QProp algorithm weight update equation
r_k	sample autocorrelation at lag k
$r_{z_{i,t}z_{j,t}}$	sample cross-correlation between time series $z_{i,t}$ and $z_{j,t}$
r	Pearson correlation coefficient
r^2	coefficient of determination
$rlen$	running length of the training set
s_O	standard deviation of the observed values
s_P	standard deviation of the predicted values
$\boldsymbol{S}$	sample covariance matrix
$\boldsymbol{S}_{xz}$	sample cross-covariance between x and z
$\boldsymbol{S}_{zz}$	sample covariance of z
t	current generation in GA
T	maximum number of generations in GA
v_{oj}	bias weights from the input layer to the hidden layer
v_{ij}	connection weights from the input layer to the hidden layer
v_k	random number seed
v'_k	random number seed after mutation
$v_{ij}(new)$	new weight from input layer to the hidden layer
$v_{ij}(old)$	old weight from input layer to the hidden layer

V_O	variance of the observed values
V_P	variance of the predicted values
$w_{jk}(new)$	new weight from hidden layer to the output layer
$w_{jk}(old)$	old weight from hidden layer to the output layer
$w^{(t)}$	weight at time t
$w(k)$	connection weight at iteration k
w_{ok}	bias weights from the hidden layer to the output layer
w_{jk}	connection weights from the hidden layer to the output layer
$\boldsymbol{W}$	set of feasible model weights with $\boldsymbol{w} \in \boldsymbol{W}$ a vector of chosen weights
$x(t)$	raw data at time t
x_i	ith actual or observed value of a data series
X_i	input PE in MLP
x'_i	ith residuals of x
x_{Ni}	ith data point of the Nth original data series
$x_{\max}$	maximum value of the original data range
$\bar{x}$	mean of the actual values of a time series
$\tilde{x}$	mean of the predicted values of a time series
$x_{\min}$	minimum value of the original data range
x^T_{high}	new maximum value for the transformed data series
x^T_{low}	new minimum value for the transformed data series
$\hat{x}_i$	predicted values of a data series
$X^T(t)$	deseasonalised transformed variable
$\boldsymbol{X}^{tr}$	training set samples
X^T_{Ni}	ith data point of the Nth transformed data series
y'_i	ith residuals of y
yin_k	input signal to kth output PE
y_k	output signal from kth output PE
Y_k	output PE in MLP
zin_j	input signal to jth hidden PE
z_j	output signal from jth hidden PE
z	pre-existing set of inputs
Z_j	hidden PE in MLP
α	adaptation gain $0 < \alpha(t) < 1$ in SOM training algorithm
$\alpha(0)$	initial learning rate
$\alpha(i)$	learning rate at iteration i
β_0	bias weights from the constant input to the hidden layer / intercept in a regression model
β_i	coefficient
β_{jh}	weights from the input layer to the hidden layer

δ	decay rate
$\delta(k)$	gradient component of the weight change at iteration k
δ_{jk}	gradient of the error surface with respect to the weights
$\delta\sigma$	perturbation in the sigma weight
ε	separation scale in correlation integral / vertical light extinction coefficient
ε_a	light extinction by algae suspended in water
ε_p	light extinction by particles suspended in water
ε_W	light extinction by water
ϕ	constant
ϕ_h	transfer function used at the hidden layer
ϕ_o	transfer function used at the output layer
ϕ_η	constant learning rate decrement factor
ϕ_μ	constant momentum rate decrement factor
λ	smoothing parameter or bandwidth of the kernel density estimate / signal-to-noise ratio
λ_{ref}	Gaussian reference bandwidth
γ	SOM learning coefficient
γ_η	constant learning rate exponential factor
γ_μ	constant momentum rate exponential factor
η	learning rate
$\eta(k)$	variable learning rate at iteration k
κ	constant
κ_η	constant learning rate scale factor
κ_μ	constant momentum rate scale factor
$\mu_s(t)$	function for the seasonally varying mean
$\mu(k)$	adaptive momentum at iteration k
μ	mean / momentum parameter in backpropagation algorithm
θ	statistic used in Kaboudan's FCS
ρ	pheromone trail persistence in MMAS / correlation coefficient
σ	sigma weight (smoothing parameter) in GRNN
σ^2	variance
$\sigma_s(t)$	function for the seasonally varying standard deviation
$\overline{\sigma}^{gb}$	global-best set of sigma weights
$\overline{\sigma}^{ib}$	iteration-best set of sigma weights
σ_j^-	lower bound of the parameter σ_j
σ_j^+	upper bound of the parameter σ_j
$\overline{\sigma}$	vector of sigma weights
σ_x	standard deviation in parameter x

σ_{xy}	covariance between parameters x and y
σ_y	standard deviation in parameter y
τ	delay (lag) time
τ_{ij}	artificial pheromone trail for arc (i,j) of the search space in MMAS
τ_{max}	maximum pheromone trail in MMAS
τ_{min}	minimum pheromone trail in MMAS
ω_0	bias weights from the constant input to the output
ω_h	weights from the hidden layer to the output layer
$\Delta w^{(t)}$	weight change at time t
Δw_{jk}	weight correction term for weights from the hidden layer to the output layer
Δv_{ij}	weight correction term for weights from the input layer to the hidden layer
$\nabla E\|_{\sigma_0}$	gradient evaluated at σ_0
$\Delta\tau_{ij}^{best}$	amount of pheromone laid down on arc(i,j) by the iteration best ant in MMAS
$\Delta q^{(t)}$	inverse Hessian step at time t

Appendix B

Abbreviations

AAE	Average absolute error
AAPE	Average absolute percentage error
ACO	Ant colony optimisation
ADALINE	Adaptive linear element
AHD	Australian height datum
AIC	Akaike's information criterion
ANN	Artificial neural network
AR	Autoregressive
ARIMA	Integrated mixed autoregressive - moving average
ARMA	Autoregressive - moving average
ART	Adaptive resonance theory
AWQC	Australian Water Quality Centre
BAM	Bidirectional associative memory
BIC	Bayes information criterion
BOD	Biological oxygen demand
BSB	Brain state in a box
CE	Coefficient of efficiency
CHT	Chaotic

CMAC	Cerebellar model articulation controller
DBD	Delta-bar-delta
DCL	Differential competitive learning
DWR	Department for Water Resources
EANN	Evolutionary artificial neural network
EBMLP	Evolutionary backpropagation multilayer perceptron
EC	Electrical conductivity
EDBD	Extended delta-bar-delta
EWS	Engineering and water supply department
FCS	Fuzzy classification system
FIR	Finite impulse response
FL	Fairly linear
GA	Genetic algorithm
GD	Generalised delta learning rule
GL	Gigalitres
GRNN	General regression neural network
GTM	Generative topographic mapping
H	Hyperbolic tangent transfer function
HE	Histogram equalisation
HU	Hazen units
iid	Independent and identically distributed
L	Linear transfer function
L1LF	Lock 1 Lower flow
L1LL	Lock 1 lower river level
L1UL	Lock1 upper river level
L7F	Downstream of Lock 7 flow
LM	Levenberg-Marquardt
LOL	Loxton river level
LOS	Loxton salinity
LPS	Lipopolysaccharide
LVQ	Learning vector quantisation
MA	Moving average
MADALINE	Multiple adaptive linear element
MAE	Maximum absolute error
MAL	Mannum river level
MANN	Multivariate artificial neural network
MAS	Mannum salinity
MBL	Murray Bridge river level
MBS	Murray Bridge salinity

MDBC	Murray Darling Basin Commission
MET	Bureau of Meteorology
MI	Mutual information
Minimax	Minimisation of the maximum discrepancy
ML	Megalitres
MLP	Multilayer perceptron
MMAS	Max-Min Ant System
MOL	Morgan river level
MOS	Morgan salinity
MProp	MaxProp algorithm
MSE	Mean squared error
MSRE	Mean squared relative error
NCD	Normalised cumulative delta (learning rule)
NL	Nonlinear
NL-HN	Nonlinear – high noise
NL-LN	Nonlinear – low noise
NL-MN	Nonlinear - medium noise
NMSE	Normalised mean squared error
NotL	Not linear
NRMSE	Normalised root mean squared error
NTU	Nephelometric turbidity units
OCF	Overland Corner flow
OCL	Overland Corner river level
OLS	Orthogonal least squares
PAC	Powdered activated carbon
PC	Principal component
PCA	Principal component analysis
PE	Processing element
PLC	Peak and low flow criterion
PMI	Partial mutual information
PMSE	Pooled mean squared error
PNN	Probabilistic neural network
PSPs	Paralytic shellfish poisons
QProp	QuickProp algorithm
RBF	Radial basis function
RMSE	Root mean squared error
S	Sigmoid transfer function
SA	South Australia
SL	Strongly linear

SOM	Self-organizing map
SP	Soluble phosphorus
SRP	Soluble reactive phosphorus
SSE	Sum of squared errors
TDNN	Time delay neural network
TDS	Total dissolved solids
TKN	Total kjedahl nitrogen
TP	Total phosphorus
UANN	Univariate artificial neural network
VARIMA	Vector autoregressive integrated moving average
VARMA	Vector autoregressive - moving average
WAL	Waikerie river level
WAS	Waikerie salinity
WL	Weakly linear
WN	White noise

Bibliography

[1] Abrahart, R. J., See, L., and Kneale, P. E. (1999). "Using pruning algorithms and genetic algorithms to optimise network architectures and forecasting inputs in a neural network rainfall-runoff model." *Journal of Hydroinformatics*, 1(2), pp.103-114.

[2] Abrahart, R. J., See, L., and Kneale, P. E. (2001). "Investigating the role of saliency analysis with a neural network rainfall-runoff model." *Computers and Geosciences*, 27, pp.921-928.

[3] Abu Kiefa, M. A. (1998). "General regression neural networks for driven piles in cohesionless soils." *Journal of Geotechnical and Geoenvironmental Engineering*, 124(12), pp.1177-1185.

[4] Abu-Mostafa, Y. S. (1989). "The Vapnik-Chervonenkis dimension: Information versus complexity in learning." *Neural Computation*, 1(3), pp.312-317.

[5] Ackley, D. H., Hinton, G. F., and Sejnowski, T. J. (1985). "A learning algorithm for Boltzman machines." *Cognitive Sciences*, 9, pp.147-169.

[6] Adeli, H., and Hung, S. L. (1995). "Machine Learning Neural Networks, Genetic Algorithms, and Fuzzy Systems." John Wiley & Sons, New York, N.Y., 211p.

[7] Aguilera, P. A. (2001). "Application of the Kohonen neural network in coastal water management: Methodological development for the assessment and prediction of water quality." *Water Research*, 35(17), pp.4053-4062.

[8] Akaike, H. (1974). "A new look at the statistical model identification." *IEEE Transactions of Automatic Control*, 19, pp.716-723.

[9] Albus, J. S. (1975). "New approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC)." *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, September 1975, pp.220-227.

[10] Allen, G., and le Marshall, J. F. (1994). "An evaluation of neural networks and discriminant analysis methods for application in operational rain forecasting." *Australian Meteorological Magazine*, 43(1), pp.17-28.

[11] Alon, I., Qi, M., and Sadowski, R. (2001). "Forecasting aggregate retail sales: a comparison of artificial neural networks and traditional methods." *Journal of Retailing and Consumer Services*, 8, pp.147-156.

[12] Altendorf, C. T., Elliott, R. L., Stevens, E. W., and Stone, M. L. (1999). "Development and validation of a neural network model for soil water content prediction with comparison to regression techniques." *Transactions of the ASAE*, 42(3), pp.691-699.

[13] Amari, S.-I., Murata, N., Muller, K.-R., Finke, M., and Yang, H. H. (1997). "Asymptotic statistical theory of overtraining and cross-validation." *IEEE Transactions on Neural Networks*, 8(5), pp.985-996.

[14] Anderson, J. A., Silverstein, J. W., Ritz, S. A., and Jones, R. S. (1977). "Distinctive features, categorical perception and probability learning: Some applications of a neural model." *Psychological Review*, 84, pp.413-451.

[15] Angeline, P. J., Sauders, G. M., and Pollack, J. B. (1994). "An evolutionary algorithm that constructs recurrent neural networks." *IEEE Transactions on Neural Networks*, 5(1), pp.54-65.

[16] Anmala, J., Zhang, B., and Govindaraju, R. S. (2000). "Comparison of ANNs and empirical approaches for predicting watershed runoff." *Journal of Water Resources Planning and Management*, 126(3), pp.156-166.

[17] ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. (2000a). "Artificial neural networks in hydrology. I: Preliminary concepts." *Journal of Hydrologic Engineering, ASCE*, 5(2), pp.115-123.

[18] ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. (2000b). "Artificial neural networks in hydrology. II: Hydrologic applications." *Journal of Hydrologic Engineering, ASCE*, 5(2), pp.124-137.

[19] Ashley, R. A., and Patterson, D. M. (2001). "Nonlinear Model Specification/Diagnostics: Insights from a Battery of Nonlinearity Tests." *E99-05*, VirginiaTech, Blacksburg, VA, February.

[20] Atiya, A. F., and Shaheen, S. I. (1999). "A comparison between neural - network forecasting techniques- case study: river flow forecasting." *IEEE Transactions on Neural Networks*, 10(2), pp.402-409.

[21] Back, A. D., and Trappenberg, T. P. (1999). "Input variable selection using independent component analysis." *International Joint Conference on Neural Networks*, Washington.

[22] Baker, P., Brookes, J., Burch, M., Maier, H., and Ganf, G. (2000). "Advection, growth and nutrient status of phytoplankton in the lower River Murray, South Australia." *Regulated Rivers: Research and Management*, 16, pp.327-344.

[23] Baker, P. D., and Humpage, A. R. (1994). "Toxicity associated with commonly occurring cyanobacteria in surface waters of the Murray-Darling Basin, Australia." *Australian Journal of Marine and Freshwater Resources*, 45(5), pp.773-786.

[24] Barnett, W. A., Gallant, A. R., Hinich, M. J., Jungeilges, J. A., Kaplan, D. T., and Jensen, M. J. (1997). "A single-blind controlled competition among tests for nonlinearity and chaos." *Journal of Econometrics*, 82, pp.157-192.

[25] Bastarache, D., El-Jabi, N., Turkham, N., and Clair, T. A. (1997). "Predicting conductivity and acidity for small streams using neural networks." *Canadian Journal of Civil Engineering*, 24(6), pp.1030-1039.

[26] Battiti, R. (1992). "First- and second-order methods for learning: between steepest descent and Newton's method." *Neural Computation*, 4, pp.141-166.

[27] Beasley, J. D., and Springer, S. G. (1977). "Algorithm AS 111: The percentage points of the normal distribution." *Applied Statistics*, 26(1), pp.118-121.

[28] Bebis, G., and Georgiopoulos, M. (1994). "Feed-forward neural networks: Why network size is so important." *IEEE Potentials*, October/November, pp.27-31.

[29] BioComp Systems Inc. (1998). "NeuroGenetic Optimizer (NGO)." Version 2.6.120, Redmond, WA.

[30] Bishop, C. M. (1995). "Neural Networks for Pattern Recognition." Oxford University Press, Oxford, 504p.

[31] Bishop, C. M., Svensen, M., and Williams, C. K. I. (1997). "GTM: A Principled Alternative to the Self-Organizing Map." in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, eds., MIT Press, Cambridge, M.A., pp.354-360.

[32] Blyth, S. (1980). "Palm Island mystery disease." *Medical Journal of Australia*, 2, pp.40-42.

[33] Bodri, L., and Cermák. (2000). "Prediction of extreme precipitation using a neural network: application to summer flood occurrence in Moravia." *Advances in Engineering Software*, 31(5), pp.311-321.

[34] Boon, P. I. (1994). "Consumption of cyanobacteria by freshwater zooplankton: Implications of the success of 'top down' control of cyanobacterial blooms in Australia." *Australian Journal of Marine and Freshwater Research*, 45, pp.875-887.

[35] Boozarjomehry, R. B., and Svrcek, W. Y. (2001). "Automatic design of neural network structures." *Computers and Chemical Engineering*, 25, pp.1075-1088.

[36] Bormans, M., and Condie, S. A. (1998). "Modelling the distribution of *Anabaena* and *Melosira* in a stratified river weir pool." *Hydrobiologia*, 364, pp.3-13.

[37] Bormans, M., Maier, H., Burch, M., and Baker, P. (1997). "Temperature stratification in the lower River Murray, Australia: implication for cyanobacterial bloom develoment." *Mar. Freshwater Res.*, 48, pp.647-654.

[38] Bowden, G. J., Maier, H. R., and Dandy, G. C. (2002). "Optimal division of data for neural network models in water resources applications." *Water Resources Research*, 38(2), pp.2-1 - 2-11 (10.1029/2001WR000266).

[39] Box, G. E. P., and Jenkins, G. M. (1970). "Time Series Analysis, Forecasting and Control." Holden-Day Inc., San Francisco, CA, 553p.

[40] Braddock, R. D., Kremmer, M. L., and Sanzogni, L. (1998). "Feed-forward artificial neural network model for forecasting rainfall run-off." *Environmetrics*, 9, pp.419-432.

[41] Breiman, L. (1994). "comment on 'Neural networks: A review from a statistical perspective' by B.Cheng and D. M. Titterington." *Statistical Science*, 9(1), pp.38-42.

[42] Breiman, L. (1996). "Bagging predictors." *Machine Learning*, 24(2), pp.123-140.

[43] Breiman, L., Meisel, W., and Purcell, E. (1977). "Variable kernel estimates of multivariate densities." *Technomentrics*, 19(2), pp.135-144.

[44] Brion, G. M., and Lingireddy, S. (1999). "A neural network approach to identifying non-point sources of microbial contamination." *Water Research*, 33(14), pp.3099-3106.

[45] Brion, G. M., Neelakantan, T. R., and Lingireddy, S. (2001). "Using neural networks to predict peak cryptosporidium concentrations." *Journal of the American Water Resources Association*, 37(1), pp.99-105.

[46] Brock, W. A., Dechert, W. D., LeBaron, B., and Sheinkman, J. (1996). "A test for independence based on the correlation dimension." *Econometric Reviews*, 15, pp.197-235.

[47] Brock, W. A., Hsieh, D., and LeBaron, B. (1991). "Nonlinear Dynamics, Chaos, and Instability: Statistical Theory and Economic Evidence." MIT Press, Cambridge, MA, 346p.

[48] Brookes, J., Burch, M., Baker, P., Ganf, G., and Maier, H. (1998). "Critical Flow and Population Development of the Cyanobacteria *Anabaena* and *Microcystis* in the Murray-Darling River System." *ACW-1*, Environment Australia and Land and Water Resources Research and Development Corporation, January.

[49] Broomhead, D. S., and Lowe, D. (1988). "Multivariate functional interpolation and adaptive networks." *Complex Systems*, 2, pp.321-355.

[50] Brown, M., and Harris, C. (1994). "Neurofuzzy Adaptive Modelling and Control." Prentice Hall, New York, 508p.

[51] Bruton, J. M., McClendon, R. W., and Hoogenboom, G. (2000). "Estimating daily pan evaporation with artificial neural networks." *Transactions of the ASAE*, 43(2), pp.491-496.

[52] Burch, M. (1999). "Blue-green algal toxins - An international and local perspective on guideline development and the implications for management." *AWWA 18th Federal Convention*, Adelaide, 11-14th April.

[53] Burch, M., and Steffensen, D. (1997). "New developments in monitoring and management of toxic blue-green algae in water supply storages." *60th Annual Water Industry Engineers and Operators Conference*, Bairnsdale, Australia, pp.79-87.

[54] Burch, M. D., Maier, H. R., and Bormans, M. (1998). "Flow management strategies to control cyanobacterial blooms in the lower River Murray." *AWWA, WaterTECH*, pp.1-11.

[55] Burian, S. J., Durrans, S. R., and Nix, S. J. (2001). "Training artificial neural networks to perform rainfall disaggregation." *Journal of Hydrologic Engineering*, 6(1), pp.43-51.

[56] Burke, L. I. (1991). "Introduction to artificial neural systems for pattern recognition." *Comput. Oper. Res.*, 18(2), pp.211-220.

[57] Cacoullous, T. (1966). "Estimation of a multivariate density." *Annals of the Institute of Statistical Mathematics (Tokyo)*, 18(2), pp.179-189.

[58] Cai, S., Toral, H., Qiu, J., and Archer, J. S. (1994). "Neural network based objective flow regime identification in air-water two phase flow." *The Canadian Journal of Chemical Engineering*, 72, pp.440-445.

[59] Campolo, M., Soldati, A., and Andreussi, P. (1999). "Forecasting river flow rate during low-flow periods using neural networks." *Water Resources Research*, 35(11), pp.3547-3552.

[60] Cannon, A. J., and Whitfield, P. H. (2001). "Modeling transient pH depressions in coastal streams of British Columbia using neural networks." *Journal of the American Water Resources Association*, 37(1), pp.73-89.

[61] Carpenter, G. A., and Grossberg, S. (1987a). "ART 2: Self-organization of stable category recognition codes for analog input patterns." *Applied optics*, 26, pp.4919-4930.

[62] Carpenter, G A., and Grossberg, S. (1987b). "A massively parallel architecture for a self-organizing neural pattern recognition machine." *Computer Vision, Graphics, and Image Processing*, 37, pp.54-115.

[63] Carpenter, G. A., and Grossberg, S. (1990). "ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures." *Neural Networks*, 3, pp.129-152.

[64] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., and Rosen, D. B. (1992). "Fuzzy ARTMAP: A neural network architecture for incremental learning of analog multidimensional maps." *IEEE Transactions on Neural Networks*, 3, pp.698-713.

[65] Carpenter, G. A., Grossberg, S., and Reynolds, J. H. (1991a). "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network." *Neural Networks*, 4, pp.565-588.

[66] Carpenter, G. A., Grossberg, S., and Rosen, D. B. (1991b). "ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition." *Neural Networks*, 4, pp.493-504.

[67] Carpenter, G. A., Grossberg, S., and Rosen, D. B. (1991c). "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system." *Neural Networks*, 4, pp.759-771.

[68] Caudill, M. (1993). "GRNN and bear it." *AI Expert*, 8(5), pp.28-33.

[69] Chakraborty, K., Mehrotra, K., Mohan, C. K., and Ranka, S. (1992). "Forecasting the behavior of multivariate time series using neural networks." *Neural Networks*, 5, pp.961-970.

[70] Chang, F., and Hwang, Y. (1999). "A self-organization algorithm for real-time flood forecast." *Hydrological Processes*, 13, pp.123-138.

[71] Chang, F.-J., and Chen, Y.-C. (2001). "A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction." *Journal of Hydrology*, 245, pp.153-164.

[72] Chatfield, C. (1993). "Neural networks: forecasting breakthrough or just a passing fad?" *International Journal of Forecasting*, 9, pp.1-3.

[73] Chen, K., Yang, L. P., Yu, X., and Chi, H. S. (1997). "A self-generating modular neural network architecture for supervised learning." *Neurocomputing*, 16(1), pp.33-48.

[74] Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). "Orthogonal least squares learning for radial basis function networks." *IEEE Transactions on Neural Networks*, 2, pp.302-309.

[75] Cheng, B., and Titterington, D. M. (1994). "Neural networks: A review from a statistical perspective." *Statistical Science*, 9(1), pp.2-54.

[76] Chentouf, R., Jutten, C., Maignan, M., and Kanevsky, M. (1997). "Incremental neural networks for function approximation." *Nuclear Instruments and Methods in Physics Research A*, 389, pp.268-270.

[77] Chorus, I., and Bartram, J. (1999). "Toxic Cyanobacteria in Water - A Guide to Their Public Health Consequences, Monitoring and Management." E & FN Spon, London, 416p.

[78] Chow, T. W. S., and Cho, S. Y. (1997). "Development of a recurrent sigma-pi neural network rainfall forecasting system in Hong Kong." *Neural Computing & Applications*, 5(2), pp.66-75.

[79] Chtioui, Y., Panigrahi, S., and Francl, L. (1999). "A generalized regression neural network and its application for leaf wetness prediction to forecast plant disease." *Chemometrics and Intelligent Laboratory Systems*, 48, pp.47-58.

[80] Cichocki, A., and Unbehauen, R. (1993). "Neural Networks for Optimization and Signal Processing." John Wiley & Sons, New York, 544p.

[81] Clair, T. A., and Ehrman, J. M. (1996). "Variations in discharge and dissolved organic carbon and nitrogen export from terrestrial basins with changes in climate: a neural network approach." *Limnology and Oceanography*, 41(5), pp.921-927.

[82] Codd, G. (1998). "Toxic cyanobacteria - the global view." *WaterTECH*, Brisbane, Queensland, 26-29 April, pp.1-14.

[83] Codd, G. A. (1995). "Cyanobacterial toxins: occurrence, properties and biological significance." *Water Science and Technology*, 32(4), pp.149-156.

[84] Codd, G. A., Steffensen, D. A., Burch, M. D., and Baker, P. D. (1994). "Toxic blooms of cyanobacteria in Lake Alexandrina, South Australia: learning from history." *Australian Journal of Marine and Freshwater Research*, 45(5), pp.731-736.

[85] Colman, B. (1989). "Photosynthetic carbon assimilation and the suppression of photorespiration in the cyanobacteria." *Aquatic Botany*, 34, pp.211-231.

[86] Conley, D. J., Schelske, C. L., and Stoermer, E. F. (1993). "Modification of the biogeochemical cycle of silica with eutrophication." *Marine Ecology Progress Series*, 101, pp.179-192.

[87] Corchado, J. M., and Fyfe, C. (1999). "Unsupervised neural method for temperature forecasting." *Artificial Intelligence in Engineering*, 13(4), pp.351-357.

[88] Coulibaly, P., Anctil, F., aravena, R., and Bobée, B. (2001a). "Artificial neural network modeling of water table depth fluctuations." *Water Resources Research*, 37(4), pp.885-896.

[89] Coulibaly, P., Anctil, F., and Bobee, B. (2000a). "Daily reservoir inflow forecasting using artificial neural networks with stopped training approach." *Journal of Hydrology*, 230, pp.244-257.

[90] Coulibaly, P., Anctil, F., and Bobee, B. (2000b). "Neural network-based long term hydropower forecasting scheme." *Computer Aided Civil and Infrastructure Engineering*, 15, pp.335-364.

[91] Coulibaly, P., Anctil, F., and Bobée, B. (1999). "Prévision hydrologique par réseaux de neurones artificiels: État de l'art." *Canadian Journal of Civil Engineering*, 26(3), pp.293-304.

[92] Coulibaly, P., Anctil, F., and Bobée, B. (2001b). "Multivariate reservoir inflow forecasting using temporal neural networks." *Journal of Hydrology*, 6(5), pp.367-376.

[93] Coulibaly, P., Bobée, B., and Anctil, F. (2001c). "Improving extreme hydrologic events forecasting using a new criterion for artificial neural network selection." *Hydrological Processes*, 15, pp.1533-1536.

[94] Cover, T. M., and Thomas, J. A. (1991). "Elements of Information Theory." John Wiley & Sons, New York, 542p.

[95] Crespo, J. L., and Mora, E. (1993). "Drought estimation with neural networks." *Advances in Engineering Software*, 18(3), pp.167-170.

[96] Croome, R. (1980). "Turbidity, Colour, Nutrients and Plankton of the River Murray." A Report to the River Murray Commission Water Quality Committee, April.

[97] Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function." *Mathematics of Control Signals and Systems*, 2, pp.203-314.

[98] Daft, M. J., Burnham, J. C., and Yamamoto, Y. (1985). "Algal blooms: consequence and potential cures." *Journal of Applied Bacteriology Symposium Supplement*, pp.175S-185S.

[99] Dandy, G. C., and Crawley, P. D. (1992). "Optimisation of multiple reservoir systems including salinity effects." *Water Resources Research*, 28(4), pp.979-990.

[100] Dandy, G. C., Simpson, A. R., and Murphy, L. J. (1996). "An improved genetic algorithm for pipe network optimisation." *Water Resources Research*, 32(2), pp.449-458.

[101] Danh, N. T., Phien, H. N., and Gupta, A. D. (1999). "Neural networks for river flow forecasting." *Water SA*, 25, pp.33-39.

[102] Dawidowicz, P. (1990). "The effect of Daphnia on filament length of blue-green algae." *Hydrobiologia*, 191, pp.265-268.

[103] Dawson, C. W., Brown, M. R., and Wilby, R. L. (2000). "Inductive learning approaches for rainfall-runoff modelling." *International Journal of Neural Systems*, 10(1), pp.43-57.

[104] Dawson, C. W., and Wilby, R. (1998). "An artificial neural network approach to rainfall-runoff modelling." *Hydrological Sciences Journal*, 43(1), pp.47-66.

[105] Dawson, C. W., and Wilby, R. L. (1999). "A comparison of artificial neural networks used for river flow forecasting." *Hydrology & Earth System Sciences*, 3(4), pp.529-540.

[106] Dawson, C. W., and Wilby, R. L. (2001). "Hydrological modelling using artificial neural networks." *Progress in Physical Geography*, 25(1), pp.80-108.

[107] Dawson, R. M. (1998). "The toxicology of microcystins." *Toxicon*, 36(7), pp.953-962.

[108] De Falco, I. (1998). "Nonlinear System Identification by Means of Evolutionary Optimised Neural Networks." in *Genetic Algorithms in Engineering and Computer Science*, Quagliarella, ed., John Wiley & Sons Ltd., pp.367-391.

[109] Department for Water Resources. (2001). "South Australian River Murray Salinity Strategy 2001-2015." Government of South Australia, Adelaide.

[110] Desieno, D. (1988). "Adding a conscience to competitive learning." *International Conference on Neural Networks*, Vol. I, pp.117-124.

[111] DeSilets, L., Golden, B., Wang, Q., and Kumar, R. (1992). "Predicting salinity in the Chesapeake Bay using backpropagation." *Computer and Operations Research*, 19(3/4), pp.227-285.

[112] Di Toro, D. M., O'Connor, D. J., Thomann, R. V., and Mancini, J. L. (1975). "Phytoplankton-Zooplankton-Nutrient Interaction Model for Western Lake Erie." in *Systems Analysis and Simulation in Ecology*, B. C. Patten, ed., Academic Press, New York, pp.423-473.

[113] Diamantaras, K. I., and Kung, S. Y. (1996). "Principal Component Neural Networks: Theory and Applications." Wiley, New York, 272p.

[114] Dibike, Y. B., and Abbott, M. B. (1999). "Application of artificial neural networks to the simulation of a two dimensional flow." *Journal of Hydraulic Research*, 37(4), pp.435-446.

[115] Dibike, Y. B., Minns, A. W., and Abbott, M. B. (1999). "Applications of artificial neural networks to the generation of wave equations from hydraulic data." *Journal of Hydraulic Research*, 37(1), pp.81-97.

[116] Dillon, P. J., and Rigler, F. H. (1974). "The phosphorous-chlorophyll relationship in lakes." *Limnology and Oceanography*, 19(5), pp.767-773.

[117] Diskin, M. H., and Simon, E. (1977). "A procedure for the selection of objective functions for hydrologic simulation models." *Journal of Hydrology*, 34, pp.129-149.

[118] Doan, C. D., and Liong, S. Y. (2002). "Improving prediction accuracy and interpretability of fuzzy inference system with efficient and effective data set." *Fifth International Conference on Hydroinformatics*, Cardiff, UK, July, Vol. 1, pp.729-734.

[119] Doering, A., Galicki, M., and Witte, H. (1997). "Structure optimization of neural networks with the A* - algorithm." *IEEE Transactions on Neural Networks*, 8(6), pp.1434-1445.

[120] Dorigo, M., and Di Caro, G. (1999). "The Ant Colony Optimization Meta-Heuristic." in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, eds., McGraw-Hill.

[121] Dowla, F. U., and Rogers, L. L. (1995). "Solving Problems in Environmental Engineering and Geosciences with Artificial Neural Networks." MIT Press, Cambridge, MA, 310p.

[122] Dwyer Leslie Pty Ltd. (1984). "River Murray Water Quality Management Study." *Working Papers E and F*, Maunsell and Partners, Melbourne, Australia, November.

[123] Efron, B., and Tibshirani, R. (1993). "An Introduction to the Bootstrap." D.R. Cox, H. D.V, N.Reid, D. B. Rubin, and B. W. Silverman, eds., Chapman & Hall, 436p.

[124] Elman, J., L. (1990). "Finding structure in time." *Cognitive Science*, 14, pp.179-211.

[125] Elshorbagy, A., Simonovic, S. P., and Panu, U. S. (2000a). "Performance evaluation of artificial neural networks for runoff prediction." *Journal of Hydrologic Engineering*, 5(4), pp.424-427.

[126] Elshorbagy, A. A., Panu, U. S., and Simonovic, S. P. (2000b). "Group-based estimation of missing hydrological data: I. Approach and general methodology." *Hydrological Sciences Journal*, 45(6), pp.849-866.

[127] Elshorbagy, A. A., Panu, U. S., and Simonovic, S. P. (2000c). "Group-based estimation of missing hydrological data: II. Application to streamflows." *Hydrological Sciences Journal*, 45(6), pp.867-880.

[128] Engle, R. (1982). "Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation." *Econometrica*, 50, pp.987-1007.

[129] Evans. (1989). "Saline water disposal options in the Murray Basin." *BMR Journal of Australian Geology and Geophysics*, 11(2/3), pp.167-185.

[130] Fahlman, S. E. (1989). "Faster-learning variations on backpropagation: An empirical study." *1988 Connectionist Models Summer School*, San Mateo, C.A., pp.38-51.

[131] Fahlman, S. E., and Lebiere, C. (1990). "The Cascade-Correlation Learning Architecture." in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed., Morgan Kaufmann, San Mateo, California, pp.524-532.

[132] Falconer, I. (1994). "Health Effects of Toxic Cyanobacteria (Blue-Green Algae)." National Health and Medical Research Council (NHMRC), Canberra.

[133] Falconer, I. R. (1988). "Eutrophication by toxic blue-green algae: an increasing hazard in Australia." *Australian Biologist*, 1(4), pp.10-12.

[134] Falconer, I. R. (1996). "Potential impact on human health of toxic cyanobacteria." *Phycologia*, 35(6), pp.6-11.

[135] Falconer, I. R., Beresford, A. M., and Runnegar, M. T. C. (1983). "Evidence of liver damage from a bloom of the Cyanobacterium Microcystis aeruginosa." *Medical Journal of Australia.*

[136] Falconer, I. R., and Humpage, A. R. (1996). "Tumour promotion by cyanobacterial toxins." *Phycologia*, 35(6), pp.74-89.

[137] Falconer, I. R., Smith, J. V., Jackson, A. R. B., Jones, A., and Runnegar, M. T. C. (1988). "Oral toxicity of a bloom of the Cyanobacterium Microcystis aeruginosa administered to mice over periods of 1 year." *J. Toxicol. Environ. Health*, 24, pp.291-305.

[138] Falter, M. (1978). "Assessment of Expected Impact of Diversion of Saline Groundwater and Irrigation Drainwater on Phytoplankton of the River Murray, South Australia. In 'The South Australian River Murray Salinity Control Program'." *3*, Engineering and Water Supply Department, South Australia, Adelaide.

[139] Faraway, J., and Chatfield, C. (1998). "Time series forecasting with neural networks: a comparative study using the airline data." *Appl. Statist.*, 47(2), pp.231-250.

[140] Fausett, L. (1994). "Fundamentals of Neural Networks." Prentice Hall, Englewood Cliffs, N.J., 461p.

[141] Fernando, D. A. K., and Jayawardena, A. W. (1998). "Runoff forecasting using RBF networks with OLS algorithm." *Journal of Hydrologic Engineering*, 3(3), pp.203-209.

[142] Fischer, H. B., List, E. J., Koh, R. C. Y., Imberger, J., and Brooks, N. H. (1979). "Mixing in Inland and Coastal Waters." Academic Press, London, 483p.

[143] Fletcher, D. S., and Goss, E. (1993). "Forecasting with neural networks: an application using bankruptcy data." *Information and Management*, 24, pp.159-167.

[144] Flood, I., and Kartam, N. (1994). "Neural networks in civil engineering. I: Principles and understanding." *Journal of Computing in Civil Engineering*, 8(2), pp.131-148.

[145] Fogel, D. B. (1994). "An introduction to simulated evolutionary optimization." *IEEE Transactions on Neural Networks*, 5(1), pp.3-14.

[146] Fogel, D. B., and Fogel, L. J. (1994). "Guest editorial - Evolutionary computation." *IEEE Transactions on Neural Networks*, 5(1), pp.1-1.

[147] Fogel, D. B., Fogel, L. J., and Porto, V. W. (1990). "Evolving neural networks." *Biol. Cybern.*, 63, pp.487-493.

[148] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). "Artificial Intelligence Through Simulated Evolution." Wiley, New York.

[149] Fogg, G. E., and Thake, B. (1987). "Algal Cultures and Phytoplankton Ecology." The University of Wisconsin Press, 269p.

[150] Fortin, V., Ouarda, T. B. M. J., and Bobée, B. (1997). "Comment on "The use of artificial neural networks for the prediction of water quality parameters" by H.R. Maier and G. C. Dandy." *Water Resources Research*, 33(10), pp.2423-2424.

[151] Frakes, B., and Yu, Z. B. (1999). "An evaluation of two hydrologic models for climate change scenarios." *Journal of the American Water Resources Association*, 35(6), pp.1351-1363.

[152] Francis, G. (1878). "Poisonous Australian lake." *Nature*, 18, pp.11-12.

[153] Fraser, A. M., and Swinney, H. L. (1986). "Independent coordinates for strange attractors from mutual information." *Physical Review A*, 33(2), pp.1134-1140.

[154] French, M., and Recknagel, F. (1994a). "Modeling algal blooms in freshwaters using artificial neural networks." *Fifth International Conference ENVIROSOFT 94*, San Francisco Bay, USA, November 16-18, pp.1-8.

[155] French, M., and Recknagel, F. (1994b). "Modelling of Algal Blooms in Freshwaters Using Artificial Neural Networks." in *Computer Techniques in Environmental Studies V, Vol. II: Environmental Systems*, Computational Mechanics Publications, Southampton, Boston, pp.86-94.

[156] French, M. N., Krajewski, W. F., and Cuykendall, R. R. (1992). "Rainfall forecasting in space and time using a neural network." *Journal of Hydrology*, 137, pp.1-31.

[157] Fritzke, B. (1995). "Growing grid- a neural network with constant neighborhood range and adaptive strength." *Neural Processing Letters*, 2(5), pp.9-13.

[158] Fukushima, K. (1988). "Neocognitron: A hierarchical neural network capable of visual pattern recognition." *Neural Networks*, 1, pp.119-130.

[159] Fukushima, K., Miyake, S., and Ito, T. (1983). "Neocognitron: A neural network model for a mechanism of visual pattern recognition." *IEEE Transactions on Systems, Man, and Cybernetics*, 13, pp.826-834.

[160] Furundzic, D. (1998). "Application of neural networks for time series analysis: Rainfall-runoff modeling." *Signal Processing*, 64, pp.383-396.

[161] Gabrys, B., and Bargiela, A. (1999). "Neural networks based decision support in presence of uncertainties." *Journal of Water Resources Planning and Management*, 125(5), pp.272-280.

[162] Galey, F. D., Beasley, V. R., Carmichael, W. W., Kleppe, G., Hooser, S. B., and Hashek, W. M. (1987). "Blue-green algae (Microcystis aeruginosa) hepatoxicosis in dairy cows." *American Journal of Veterinary Research*, 48, pp.1415-1420.

[163] Ganf, G. G. (1982). "Influence of added nutrient on the seasonal variation of algal growth potential at Mount Bold reservoir, South Australia." *Australian Journal of Marine and Freshwater Resources*, 33, pp.475-490.

[164] Gautam, M. R., Watanabe, K., and Saegusa, H. (2000). "Runoff analysis in humid forest catchment with artificial neural network." *Journal of Hydrology*, 235, pp.117-136.

[165] Geddes, M. C. (1984). "Limnology of Lake Alexandrina, River Murray, South Australia, and the effects of nutrients and light on the phytoplankton." *Australian Journal of Marine and Freshwater Resources*, 35, pp.399-415.

[166] Gehrke, P. C., and Harris, J. H. (1994). "The role of fish in cyanobacteral blooms in Australia." *Australian Journal of Marine and Freshwater Research*, 45, pp.905-915.

[167] Gershenfeld, N., and Weigend, A. (1994). "Time Series Prediction: Forecasting the Future and Understanding the Past." A. Weigend and N. Gershenfeld, eds., Addison-Wesley, Reading, MA, 643p.

[168] Ghassemi, F., Jakeman, A. J., and Nix, H. A. (1995). "Salinisation of Land and Water Resources." University of New South Wales Press Ltd, Sydney, 526p.

[169] Goldberg, D. E. (1989). "Genetic Algorithms in Search, Optimization and Machine Learning." Addison-Wesley, Reading, MA, 412p.

[170] Goldberg, D. E., and Deb, K. (1991). "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms." in *Foundations of Genetic Algorithms*, J. E. Rawlins, ed., Morgan Kaufmann, San Mateo, California, pp.69-93.

[171] Golob, R., Stokelj, T., and Grgic, D. (1998). "Neural-network-based water inflow forecasting." *Control Engineering Practice*, 6(5), pp.593-600.

[172] Goswami, P., and Srividya. (1996). "A novel neural network design for long range prediction of rainfall pattern." *Current Science*, 70(6), pp.447-457.

[173] Govindaraju, R. S., and Rao, A. R. (2000). "Artificial Neural Networks in Hydrology." Kluwer Academic Publishers, Dordrecht, The Netherlands, 329p.

[174] Grassberger, P., and Procaccia, I. (1983). "Measuring the strangeness of strange attractors." *Physica D*, 9, pp.189-208.

[175] Grecu, M., and Krajewski, W. F. (2000). "A large-sample investigation of statistical procedures for radar-based short-term quantitative precipitation forecasting." *Journal of Hydrology*, 239, pp.69-84.

[176] Grobbelaar, J. U. (1985). "Phytoplankton productivity in turbid waters." *Journal of Plankton Research*, 7(5), pp.653-663.

[177] Grossberg, S. (1976). "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors." *Biological Cybernetics*, 23, pp.121-134.

[178] Guhathakurta, P., Rajeevan, M., and Thapliyal, V. (1999). "Long range forecasting India summer monsoon rainfall by a hybrid principal component neural network model." *Meteorology & Atmospheric Physics*, 71(3-4), pp.255-266.

[179] Guiot, J. (1981). "Analyse Mathématique de Données Geophysique. Applications á la Dendroclimatologie." Unpublished Ph.D. Dissertation, Université Catholique de Louvain.

[180] Gumrah, F., Oz, B., Guler, B., and Evin, S. (2000). "The application of artificial neural networks for the prediction of water quality of a polluted aquifer." *119*, 1-4, pp.275-294.

[181] Guo, J. C. Y. (2001). "A semivirtual watershed model by neural networks." *Computer-Aided Civil & Infrastructure Engineering*, 16(2), pp.106-111.

[182] Hakkarainen, J., Jumppanen, A., Kyngas, J., and Kyyro, J. (1996). "An Evolutionary Approach to Neural Network Design Applied to Sunspot Prediction." *A-1995-5*, Department of Computer Science, University of Joensuu, Joensuu, January.

[183] Hall, M. J., and Minns, A. W. (1999). "Classification of hydrologically homogeneous regions." *Hydrological Sciences Journal*, 44(5), pp.693-704.

[184] Hansen, J. V., and Meservy, R. D. (1996). "Learning experiments with genetic optimization of a general regression neural network." *Decision Support Systems*, 18, pp.317-325.

[185] Harris, G. P. (1994). "Nutrient Loadings and Algal Blooms in Australian Waters - a Discussion Paper." *Occasional Paper No. 12/94*, Land & Water Resources Research & Development Corporation, Canberra, August.

[186] Harris, G. P., Haffner, G. D., and Piccinin, B. B. (1980). "Physical variability and phytoplankton communities. II. Primary production by phytoplankton in a physically variable lake." *Archiv. fur Hydrobiologie*, 88, pp.393-425.

[187] Harris, G. P., and Piccinin, B. B. (1980). "Physical variability and phytoplankton communities. IV. Temporal changes in the phytoplankton community of a physically variable lake." *Archiv. fur Hydrobiologie*, 89, pp.447-473.

[188] Harrold, T. I., Sharma, A., and Sheather, S. (2001). "Selection of a kernel bandwidth for measuring dependence in hydrologic time series using the

mutual information criterion." *Stochastic Environmental Research and Risk Assessment*, 15, pp.310-324.

[189] Haugh, L. D., and Box, G. E. P. (1977). "Identification of dynamic regression (distributed lag) models connecting two time series." *Journal of the American Statistical Association*, 72(397), pp.121-130.

[190] Hawkins, P. R., Runnegar, M. T. C., Jackson, A. R. B., and Falconer, I. R. (1985). "Severe hepatotoxicity caused by the tropical cyanobacterium (blue-green alga) *Cylindrospermopsis raciborskii (Woloszynska) Seenaya* and *Subba Raju* isolated from a domestic water supply reservoir." *Applied Environmental Microbiology*, 50, pp.1292-1295.

[191] Haykin, S. (1994). "Neural Networks: A Comprehensive Foundation." MacMillan, New York, 696p.

[192] Hebb, D. (1949). "The Organization of Behavior." Wiley, New York.

[193] Hecht-Nielsen, R. (1987). "Counterpropagation networks." *Applied Optics*, 26, pp.4979-4984.

[194] Hecht-Nielsen, R. (1988). "Applications of counterpropagation networks." *Neural Networks*, 1, pp.131-139.

[195] Hecht-Nielsen, R. (1989). "Neurocomputing." Addison-Wesley Publishing Company, Inc., 433p.

[196] Hecht-Nielson, R. (1987). "Kolmogorov's mapping neural network existence theorem." *First IEEE International Joint Conference on Neural Networks*, San Diego, California, June 21-24.

[197] Hergenrader, G. L., and Hammer, M. (1973). "Eutrophication of small reservoirs in the Great Plains." *Geophysical Monograph*, 17, pp.560-566.

[198] Hertz, J., Krogh, A., and Palmer, R. (1991). "Introduction to the Theory of Neural Computation." Addison-Wesley, Redwood City, California, 327p.

[199] Hiew, M. F., and Green, G. W. (1992). "Beyond statistics. A forecasting system that learns." *The Forum*, 5(3), pp.1-6.

[200] Hill, T., O'Connor, M., and Remus, W. (1996). "Neural network models for time series forecasts." *Management Science*, 42, pp.1082-1092.

[201] Hinich, M. (1982). "Testing for gaussianity and linearity of a stationary time series." *Journal of Time Series Analysis*, 3, pp.169-176.

[202] Hinich, M. (1996). "Testing for dependence in the input to a linear time series model." *Journal of Nonparametric Statisitics*, 6, pp.205-221.

[203] Hobson, P., Fallowfield, H., and Burch, M. (1999). "Effect of irradiance, temperature and salinity on biomass and toxin production by the cyanobacterium *Nodularia Spumigena*." *AWWA 18th Federal Convention*, Adelaide, 11-14th April.

[204] Hochreiter, S., and Schmidhuber, J. (1997). "Long short-term memory." *Neural Computation*, 9, pp.1735-1780.

[205] Hooser, S. B., Beasley, V. R., Basgall, E. J., Carmichael, W. W., and Haschek, W. M. (1990). "Microcystin-LR induced ultrastructural changes in rats." *Veterinary Pathology*, 27, pp.9-15.

[206] Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the Natural Academy of Science USA*, Vol. 79, pp.2254-2558.

[207] Hopfield, J. J. (1984). "Neurons with graded response have collective computational properties like those of two-state neurons." *Proceedings of the Natural Academy of Science USA*, Vol. 81, pp.3088-3092.

[208] Horn, H., and Uhlmann, D. (1995). "Competitive growth of blue-greens and diatoms (Fragilaria) in the Saidenbach Reservoir, Saxony." *Water Science and Technology*, 32(4), pp.77-88.

[209] Hornik, K. (1991). "Approximation capabilities of multilayer feedforward networks." *Neural Networks*, 4, pp.2151-2157.

[210] Hornik, K., Stinchcombe, M., and White, H. (1989). "Multilayer feedforward networks are universal approximators." *Neural Networks*, 2, pp.359-366.

[211] Hötzel, G., and Croome, R. (1994). "Long-term phytoplankton monitoring of the Darling River at Burtundy, New South Wales: Incidence and significance of cyanobacterial blooms." *Australian Journal of Marine and Freshwater Research*, 45, pp.747-759.

[212] Hsu, K. L., Gao, X. G., Sorooshian, S., and Gupta, H. V. (1997). "Precipitation estimation from remotely sensed information using artificial neural networks." *Journal of Applied Meteorology*, 36(9), pp.1176-1190.

[213] Hsu, K. L., Gupta, H. V., Gao, X. G., and Sorooshian, S. (1999). "Estimation of physical variables from multichannel remotely sensed imagery using a neural network: Application to rainfall estimation." *Water Resources Research*, 35(5), pp.1605-1618.

[214] Hsu, K. L., Gupta, H. V., Gao, X. G., Sorooshian, S., and Imam, B. (2002a). "Hydrologic modelling and analysis using a self-organizing linear output network." *1st Biennial Meeting of the International Environmental Modelling and Software Society*, Lugano, Switzerland, 24-27 June, Vol. 2, pp.172-177.

[215] Hsu, K. L., Gupta, H. V., Gao, X. G., Sorooshian, S., and Imam, B. (2002b). "Self-organizing linear output map (SOLO): An artificial neural network suitable for hydrologic modeling and analysis." *Water Resources Research*, 38(12), pp.10.1029/2001WR000795.

[216] Hsu, K. L., Gupta, H. V., and Sorooshian, S. (1995). "Artificial neural network modeling of the rainfall-runoff process." *Water Resources Research*, 31(10), pp.2517-2530.

[217] Huang, W., and Foo, S. (2002). "Neural network modeling of salinity variation in Apalachicola River." *Water Research*, 36, pp.356-362.

[218] Imrie, C. E., Durucan, S., and Korre, A. (2000). "River flow prediction using artificial neural networks: generalisation beyond the calibration range." *Journal of Hydrology*, 233, pp.138-153.

[219] Islam, S., and Kothari, R. (1999). "Spatial characterization of remotely sensed soil moisture data using self organizing feature maps." *IEEE Transactions on Geoscience & Remote Sensing*, 37(2 Part 2), pp.1162-1165.

[220] Islam, S., and Kothari, R. (2000). "Artificial neural networks in remote sensing of hydrologic processes." *Journal of Hydrologic Engineering*, 5(2), pp.138-144.

[221] Jack, J. D., Wickham, S. A., Toalson, S., and Gilbert, J. J. (1993). "The effect of clays on a freshwater plankton community: an enclosure experiment." *Archiv. fur Hydrobiologie*, 127, pp.257-270.

[222] Jackson, A. R. B., McInnes, A., Falconer, I. R., and Runnegar, M. T. C. (1984). "Clinical and pathological changes in sheep experimentally poisoned by the Blue-green algae Microcystis aeruginosa." *Veterinary Pathology*, 21, pp.102-113.

[223] Jacobs, R. A. (1988). "Increased rates of convergence through learning rate adaptation." *Neural Networks*, 1(4), pp.295-308.

[224] Jacobs, T. (1990). "Regulation and Management of the River Murray." in *The River Murray*, N. Mackay and D. Eastburn, eds., MDBC, Canberra, Australia, pp.39-58.

[225] Jain, S. K., and Chalisgaonkar, D. (2000). "Setting up stage-discharge relations using ANN." *Journal of Hydrologic Engineering*, 5(4), pp.428-433.

[226] Jain, S. K., Das, A., and Srivastava, D. K. (1999). "Application of ANN for reservoir inflow prediction and operation." *Journal of Water Resources Planning and Management*, 125(5), pp.263-271.

[227] Jayawardena, A. W., and Fernando, D. A. K. (1998). "Use of radial basis function type artificial neural networks for runoff simulation." *Computer-Aided Civil and Infrastructure Engineering*, 13(2), pp.91-99.

[228] Jayawardena, A. W., Fernando, D. A. K., and Zhou, M. C. (1997). "Comparison of multilayer perceptron and radial basis function networks as tools for flood forecasting." *IAHS Publication (International Association of Hydrological Sciences)*, 239.

[229] Johnson, V., and Rogers, L. (2000). "Accuracy of neural network approximators in simulation-optimization." *Journal of Water Resources Planning and Management*, 126(2), pp.48-56.

[230] Jolliffe, I. T. (1986). "Principal Component Analysis." Springer-Verlag New York Inc., New York, 271p.

[231] Jones, G. (1994). "Weir pool conditions stimulating cyanobacterial blooms in the Murrumbidgee River." *The Murrumbidgee, Past and Present*, pp.70-82.

[232] Jones, G., Baker, P., Burch, M., and Harvey, F. (2003). "National Protocol for the Monitoring of Cyanobacteria in Surface Waters (in preparation)." Australian Water Quality Centre, Adelaide, January.

[233] Jones, G., Burch, M., Falconer, I., and Craig, K. (1993). "Cyanobacterial Toxicity." in *Technical Advisory Group Report, Murray Darling Basin Commission, Algal Management Strategy*, MDBC, Canberra, pp.17-32.

[234] Jones, J. R., and Bachmann, R. W. (1976). "Prediction of phosphorus and chlorophyll levels in lakes." *J. Water Pollut. Control Fed.*, 48, pp.2176-2182.

[235] Jones, P. D., Wigley, T. M. L., and Briffa, K. R. (1983). "Reconstructing surface pressure patterns using principal components regression on temperature and precipitation data." *Second International Meeting on Statistical Climatology*, Lisbon, Portugal, October, pp.4.2.1-4.2.8.

[236] Jordan, M. I. (1986). "Attractor dynamics and parallelism in a connectinist sequential machine." *Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, pp.531-546.

[237] Jorgensen, S. E. (1976). "A eutrophication model for a lake." *Ecological Modelling*, 2, pp.147-162.

[238] Jutten, C., and Chentouf, R. (1995). "A new scheme for incremental learning." *Neural Processing Letters*, 1, pp.1-4.

[239] Kaastra, I., and Boyd, M. (1996). "Designing a neural network for forecasting financial and economic time series." *Neurocomputing*, 10, pp.215-236.

[240] Kaboudan, M. A. (1995). "Measuring the Complexity of Nonlinearity By a Relative Index With Application to Financial Time Series." in *Chaos and Nonlinear Dynamics in the Financial markets: Theory, Evidence and*

Applications, R. Trippi, ed., Irwin Professional Publishing, Chicago, pp.417-438.

[241] Kaboudan, M. A. (1998). "Statistical properties of a time-series-complexity measure applied to stock returns." *Computational Economics*, 11, pp.167-187.

[242] Kaboudan, M. A. (1999). "Diagnosing chaos by a fuzzy classifier." *Fuzzy Sets and Systems*, 108, pp.1-10.

[243] Källqvist. (1981). "Hydroecological Field Experiment 1981. Incubation of Natural Phytoplankton in Lake Gjersjoen." *F-80402*, Norwegian Institute for Water Research, Oslo.

[244] Kang, K. W., Park, C. Y., and Kim, J. H. (1993). "Neural network and its application to rainfall-runoff forecasting." *Korean Journal of Hydroscience*, 4, pp.1-9.

[245] Karul, C., Soyapuk, S., and Germen, E. (1998). "A new approach to mathematical water quality modelling in reservoirs: Neural networks." *International Review of Hydrobiology*, 83, pp.689-696.

[246] Karunanithi, N., Grenney, W. J., Whitley, D., and Bovee, K. (1994). "Neural networks for river flow prediction." *Journal of Computing in Civil Engineering*, 1(2), pp.201-220.

[247] Kaski, S., Kangas, J., and Kohonen, T. (1998). "Bibliography of Self-Organizing Map (SOM) Papers: 1981-1997." *Neural Computing Surveys*, 1, pp.102-350.

[248] Kasuba, T. (1993). "Simplified Fuzzy ARTMAP." *AI Expert*, 8, pp.18-25.

[249] Khalil, M., Panu, U. S., and Lennox, W. C. (2001). "Groups and neural networks based streamflow data infilling procedures." *Journal of Hydrology*, 241, pp.153-176.

[250] Khotanzad, A., Afkhami-Rohani, R., Lu, T.-L., Abaye, A., Davis, M., and Muratukalem, D. J. (1997). "ANNSTLF - a neural network based electric load forecasting system." *IEEE Transactions on Neural Networks*, 8(4), pp.835-846.

[251] Kim, G., and Barros, A. P. (2001). "Quantitative flood forecasting using multisensor data and neural networks." *Journal of Hydrology*, 246, pp.45-62.

[252] King, D. L. (1970). "The role of carbon in eutrophication." *J. Wat. Pollut. Fed.*, 42, pp.2035-2051.

[253] Kitano, H. (1994). "Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms." *Physica D*, 75, pp.225-238.

[254] Kiviranta, J., Namikoshi, M., Sivonen, K., Evans, W. R., Carmichael, W., and Rinehart, K. L. (1992). "Structure determination and toxicity of a new microcystin from *Microcystis aeruginosa* Strain 205." *Toxicon*, 30(9), pp.1093-1098.

[255] Kocjancic, R., and Zupan, J. (2000). "Modelling of the river flowrate: the influence of the training set selection." *Chemometrics and Intelligent Laboratory Systems*, 54, pp.21-34.

[256] Koekkoek, E. J. W., and Booltink, H. (1999). "Neural network models to predict soil water retention." *European Journal of Soil Science*, 50(3), pp.489-495.

[257] Kohonen, T. (1982). "Self-organized formation of topologically correct feature maps." *Biological Cybernetics*, 43, pp.59-69.

[258] Kohonen, T. (1988). "Learning vector quantization." *Neural Networks*, 1(suppl. 1).

[259] Kohonen, T. (1990). "The self-organizing map." *Proc. IEEE*, 78(9), pp.1464-1480.

[260] Kosko, B. (1992). "Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence." Prentice-Hall, Englewood Cliffs, N.J., 449p.

[261] Kreyszig, E. (1988). "Advanced Engineering Mathematics." John Wiley & Sons, New York, 1294p.

[262] Kugiumtzis, D. (1999). "Lack of robustness of the surrogate data test." *submitted to Physica D.*

[263] Kuligowski, R. J., and Barros, A. P. (1998a). "Experiments in short-term precipitation forecasting using artificial neural networks." *Monthly Weather Review*, 126, pp.470-482.

[264] Kuligowski, R. J., and Barros, A. P. (1998b). "Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks." *Weather and Forecasting*, 13, pp.1194-1204.

[265] Kuligowski, R. J., and Barros, A. P. (1998c). "Using artificial neural networks to estimate missing rainfall data." *Journal of the American Water Resources Association*, 34(6), pp.1437-1448.

[266] Kwok, T.-Y., and Yeung, D.-Y. (1997a). "Constructive algorithms for structure learning in feedforward neural networks for regression problems." *IEEE Transactions on Neural Networks*, 8(3), pp.630-645.

[267] Kwok, T.-Y., and Yeung, D.-Y. (1997b). "Objective functions for training new hidden units in constructive neural networks." *IEEE Transactions on Neural Networks*, 8(5), pp.1131-1148.

[268] Laarhoven, P. J. M., and Aarts, E. H. L. (1987). "Simulated Annealing: Theory and Applications." D. Reidel Publishing Co., 186p.

[269] Lachtermacher, G., and Fuller, J. D. (1994). "Backpropagation in hydrological time series forecasting." in *Stochastic and Statistical Methods in Hydrology and Environmental Engineering*, K. W. Hipel, A. I. McLeod, U. S. Panu, and V. P. Singh, eds., Kluwer Academic Publishers, Dordrecht, pp.229-242.

[270] Lang, K. G., Waibel, A. H., and Hinton, G. (1990). "A time-delay neural network architecture for isolated word recognition." *Neural Networks*, 3, pp.23-44.

[271] Lange, N. T. (1999). "New mathematical approaches in hydrological modeling - An application of artificial neural networks." *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere*, 24(1-2), pp.31-35.

[272] Laslett, G. M., Clark, R. M., and Jones, G. J. (1997). "Estimating the precision of filamentous blue-green algae cell concentration from a single sample." *Environmetrics*, 8(4), pp.313-339.

[273] Lauzon, N., Rousselle, J., Birikundavyi, S., and Trung, H. T. (2000). "Real-time daily flow forecasting using black-box models, diffusion processes and neural networks." *Canadian Journal of Civil Engineering*, 27(4), pp.671-682.

[274] Le Cun, Y. (1991). "Second-Order Properties of Error Surfaces: Learning Time and Generalization." in *Advances in Neural Information Processing Systems*, Morgan Kaufmann, San Mateo, CA, pp.918-924.

[275] LeCun, Y. (1985). "Une procédure d'apprentissage pour réseau á seuil asymetrique." *Cognitiva*, 85(Paris: CESTA), pp.599-604.

[276] Legates, D. R., and McCabe, G. J. (1999). "Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation." *Water Resources Research*, 35(1), pp.233-241.

[277] Lekkas, D. F., Imrie, C. E., and Lees, M. J. (2001). "Improved non-linear transfer function and neural network methods of flow routing for real-time forecasting." *Journal of Hydroinformatics*, 3(3), pp.153-164.

[278] Lertpalangsunti, N., Chan, C. W., Mason, R., and Tontiwachwuthikul, P. (1999). "A toolset for construction of hybrid intelligent forecasting systems - application for water demand prediction." *Artificial Intelligence in Engineering*, 13(1), pp.21-42.

[279] Leshno, M., Lin, V. Y., Pinkus, A., and Schoken, S. (1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function." *Neural Networks*, 6, pp.861-867.

[280] Liong, S. Y., Khu, S. T., and Chan, W. T. (2001). "Derivation of pareto front with genetic algorithm and neural network." *Journal of Hydrologic Engineering*, 6(1), pp.52-61.

[281] Liong, S. Y., Lim, W. H., Kojiri, T., and Hori, T. (2000a). "Advance forecasting for flood stricken Bangladesh with a fuzzy reasoning model." *Hydrological Processes*, 14, pp.431-448.

[282] Liong, S. Y., Lim, W. H., and Paudyal, G. N. (2000b). "River stage forecasting in Bangladesh: Neural network approach." *Journal of Computing in Civil Engineering*, 14(1), pp.1-8.

[283] Lippmann, R. P. (1987). "An introduction to computing with neural nets." *IEEE Acoustics, Speech and Signal Processing Magazine*, 4, pp.4-22.

[284] Lischeid, G. (2001). "Investigating short-term dynamics and long-term trends of SO_4 in the runoff of a forested catchment using artificial neural networks." *Journal of Hydrology*, 243, pp.31-42.

[285] Lischeid, G., Lange, H., and Hauhs, M. (1998). "Neural network modelling of NO_3- time series from small headwater catchments." *IAHS Publication (International Association of Hydrological Sciences)*, 248.

[286] Loke, E., Warnaars, E. A., Jacobsen, P., Nelen, F., and Almeida, M. D. (1997). "Artificial neural networks as a tool in urban storm drainage." *Water Science & Technology*, 36(8-9), pp.101-109.

[287] Looney, C. G. (1997). "Pattern Recognition using Neural Networks." Oxford University Press, New York, 458p.

[288] Lorrai, M., and Sechi, G. M. (1995). "Neural nets for modelling rainfall-runoff transformations." *Water Resources Management*, 9(4), pp.299-313.

[289] Loucks, D. P., Stedinger, J. R., and Haith, D. A. (1981). "Water Resource Systems Planning and Analysis." Prentice-Hall Inc., Englewood Cliffs, N.J., 559p.

[290] Lu, R. S., Lai, J. L., and Lo, S. L. (1998). "Predicting solute transfer to surface runoff using neural networks." *Water Science & Technology*, 38(10), pp.173-180.

[291] Luk, K. C., Ball, J. E., and Sharma, A. (2000). "A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting." *Journal of Hydrology*, 227, pp.56-65.

[292] Luk, K. C., Ball, J. E., and Sharma, A. (2001). "An application of artificial neural networks for rainfall forecasting." *Mathematical and Computer Modelling*, 33(6-7), pp.683-693.

[293] Luthy, J. (1979). "Epidemic Paralytical Shellfish Poisoning in Western Europe 1976." in *Toxic Dinoflagellate Blooms*, D. L. Taylor and H. H. Seliger, eds., Elsevier/North Holland, New York and London, pp.23-28.

[294] Luukkainen, R., Namikoshi, M., Sivonen, K., Rinehart, K. L., and Niemela, S. I. (1994). "Isolation and identification of 12 Microcystins from four strains and two bloom samples of *Microcystis* spp.: Structure of new Hepatotoxins." *Toxicon*, 32, pp.133-139.

[295] MacLeod, C., and Maxwell, G. (2000). "Growing artificial neural networks using evolutionary algorithms." School of Electronic and Electrical Engineering, The Robert Gordon University, Aberdeen, Scotland, http://www.eee.rgu.ac.uk/research/neural/gannuea.html

[296] Maier, H. R. (1995). "Use of Artificial Neural Networks for Modelling Multivariate Water Quality Time Series." PhD Thesis, The University of Adelaide, Adelaide, 559p.

[297] Maier, H. R., and Dandy, G. C. (1994). "Blue-Green Algae in the River Murray." *G25*, University of Adelaide, Department of Civil and Environmental Engineering, Adelaide, December.

[298] Maier, H. R., and Dandy, G. C. (1996a). "Neural network models for forecasting univariate time series." *Neural Network World*, 6(5), pp.747-771.

[299] Maier, H. R., and Dandy, G. C. (1996b). "The use of artificial neural networks for the prediction of water quality parameters." *Water Resources Research*, 32(4), pp.1013-1022.

[300] Maier, H. R., and Dandy, G. C. (1997a). "Determining inputs for neural network models of multivariate time series." *Microcomputers in Civil Engineering*, 12(5), pp.353-368.

[301] Maier, H. R., and Dandy, G. C. (1997b). "Modelling cyanobacteria (blue-green algae) in the River Murray using artificial neural networks." *Mathematics and Computers in Simulation*, 43, pp.377-386.

[302] Maier, H. R., and Dandy, G. C. (1998a). "The effect of internal parameters and geometry on the performance of back-propagation neural networks: an empirical study." *Environmental Modelling and Software*, 13(2), pp.193-209.

[303] Maier, H. R., and Dandy, G. C. (1998b). "Understanding the behaviour and optimising the performance of back-propagation neural networks: an empirical study." *Environmental Modelling and Software*, 13(2), pp.179-191.

[304] Maier, H. R., and Dandy, G. C. (1999). "Empirical comparison of various methods for training feed-forward neural networks for salinity forecasting." *Water Resources Research*, 35(8), pp.2591-2596.

[305] Maier, H. R., and Dandy, G. C. (2000a). "Application of Artificial Neural Networks to Forecasting of Surface Water Quality Variables: Issues Applications and Challenges." in *Artificial Neural Networks in Hydrology*, R. S. Govindaraju and A. R. Rao, eds., Kluwer, Dordrecht, The Netherlands, pp.287-309.

[306] Maier, H. R., and Dandy, G. C. (2000b). "Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications." *Environmental Modelling and Software*, 15, pp.101-124.

[307] Maier, H. R., and Dandy, G. C. (2001). "Neural network based modelling of environmental variables: A systematic approach." *Mathematical and Computer Modelling*, 33, pp.669-682.

[308] Maier, H. R., Dandy, G. C., and Burch, M. D. (1998). "Use of artificial neural networks for modelling cyanobacteria *Anabaena* spp. in the River Murray, South Australia." *Ecological Modelling*, 105, pp.257-272.

[309] Maier, H. R., Sayed, T., and Lence, B. J. (2000). "Forecasting cyanobacterial concentrations using B-spline networks." *Journal of Computing in Civil Engineering, ASCE*, 14(3), pp.183-189.

[310] Maniezzo, V. (1994). "Genetic evolution of the topology and weight distribution of neural networks." *IEEE Transactions on Neural Networks*, 5(1), pp.39-53.

[311] Maren, A., Harston, C., and Pap, R. (1990). "Handbook of Neural Computing Applications." Academic Press Inc., San Diego, 448p.

[312] Mason, J. C., Tem'me, A., and Price, R. K. (1996). "A neural network model of rainfall-runoff using radial basis functions." *Journal of Hydraulic Research*, 34, pp.537-548.

[313] Masters, T. (1993). "Practical Neural Network Recipes in C++." Academic Press, San Diego, 493p.

[314] Masters, T. (1995a). "Advanced Algorithms for Neural Networks: A C++ Sourcebook." John Wiley and Sons, New York, 448p.

[315] Masters, T. (1995b). "Neural, Novel and Hybrid Algorithms for Time Series Prediction." John Wiley and Sons, New York, 514p.

[316] Masters, T., and Land, W. (1997). "New training algorithm for the general regression neural network." *IEEE International Conference on Systems, Man and Cybernetics*, Orlando, Florida, pp.1990-1994.

[317] McCullagh, P., and Nelder, J. A. (1989). "Generalized Linear Models." Chapman and Hall, London, 532p.

[318] McCulloch, W. S., and Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity." *Bulletin of Mathematical Biophysics*, 5, pp.115-133.

[319] McDonnell, J. R., and Waagen, D. (1994). "Evolving recurrent perceptrons for time series modeling." *IEEE Transactions on Neural Networks*, 5, pp.24-38.

[320] McLeod, A. I., and Li, W. K. (1983). "Diagnostic checking ARMA time series models using squared-residual autocorrelation tests." *Journal of Time Series and Analysis*, 4, pp.269-273.

[321] MDBC. (1993). "Algal Management Strategy - Technical Advisory Group Report." Murray-Darling Basin Commission, Canberra.

[322] MDBC. (1999). "Water quality." www.mdbc.gov.au/issues/water_quality/water_quality.html

[323] MDBMC. (1994). "Algal Management Strategy for the Murray-Darling Basin." Murray-Darling Basin Ministerial Council, Canberra.

[324] Medsker, L. R., and Jain, L. C. (2000). "Recurrent Neural Networks: Design and Applications." L. R. Medsker and L. C. Jain, eds., CRC Press, Boca Raton, FL, 416p.

[325] Mehra, P., and Wah, B. (1992). "Artificial Neural Networks: Concepts and Theory." IEEE Computer Society Press, California, 667p.

[326] Michaelides, S., Pattichis, C. S., and Kleovoulou, G. (2001). "Classification of rainfall variability by using artificial neural networks." *International Journal of Climatology*, 21(11), pp.1401-1414.

[327] Michalewicz, Z. (1994). "Genetic Algorithms + Data Structures = Evolution Programs." Springer-Verlag, Berlin, 340p.

[328] Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). "Designing neural networks using genetic algorithms." *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, pp.379-384.

[329] Miller, S. W. (1997). "Rain rate estimation using neural networks." *AI Applications*, 11(1), pp.95-98.

[330] Minai, A. A., and Williams, R. D. (1990). "Acceleration of back-propagation through learning rate and momentum adaptation." *International Joint Conference on Neural Networks*, January, Vol. Volume I, pp.676-679.

[331] Minns, A. W., and Hall, M. J. (1996). "Artificial neural networks as rainfall-runoff models." *Hydrological Sciences Journal*, 41(3), pp.399-417.

[332] Minsky, M., and Papert, S. (1969). "Perceptrons." MIT Press, Cambridge MA.

[333] Moatar, F., Fessant, F., and Poirel, A. (1999). "pH modelling by neural networks. Application of control and validation data series in the Middle Loire River." *Ecological Modelling*, 120(2-3), pp.141-156.

[334] Moody, J. E., and Darken, C. (1989). "Fast learning in networks of locally-tuned processing units." *Neural Computation*, 1, pp.281-294.

[335] Morton, S. D., and Lee, T. H. (1974). "Algal blooms - Possible effects of iron." *Environmental Monitoring and Assessment*, 8(7), pp.673-674.

[336] Mulier, F., and Cherkassy, V. (1995). "Self-organization as an iterative kernel smoothing process." *Neural Computation*, 7, pp.1165-1177.

[337] Murata, N., Yoshizawa, S., and Amari, S. (1994). "Network information criterion - Determining the number of hidden units for an artificial neural network model." *IEEE Transactions on Neural Networks*, 5, pp.865-872.

[338] Murphy, T. P., Lean, D. R. S., and Nalewajko, C. (1976). "Blue-green algae: Their excretion of iron selective chelators enables them to dominate other algae." *Science*, 192, pp.900-902.

[339] Murray-Darling Basin Commission. (1990). "The Murray." N. Mackay and D. Eastburn, eds., MDBC, Canberra, 363p.

[340] Murray-Darling Basin Commission. (1999). "The Salinity Audit of the Murray-Darling Basin." MDBC, Canberra.

[341] Murray-Darling Basin Ministerial Council. (1988). "Draft Salinity and Drainage Strategy." Murray-Darling Basin Ministerial Council, Canberra.

[342] Murray-Darling Basin Ministerial Council. (1989). "Murray-Darling Basin Natural Resources Management Strategy." Murray-Darling Basin Ministerial Council, Canberra.

[343] Muttiah, R. S., Srinivasan, R., and Allen, P. M. (1997). "Prediction of two-year peak stream discharges using neural networks." *Journal of the American Water Resources Association*, 33(3), pp.625-630.

[344] Namikoshi, M., Sivonen, K., Evans, W. R., Sun, F., Carmichael, W. W., and Rinehart, K. L. (1992). "Isolation and structures of Microcystins from a cyanobacterial water bloom (Finland)." *Toxicon*, 30, pp.1473-1479.

[345] National Rivers Authority UK. (1990). "Toxic Blue-Green Algae." Stanley L. Hunt (Printers) Ltd., 128p.

[346] Neal, R. M. (1996). "Bayesian Learning for Neural Networks." Springer-Verlag, New York, 183p.

[347] Neelakantan, T. R., Brion, G. M., and Lingireddy, S. (2001). "Neural network modeling of Cryptosporidium and Giardia concentrations in the Delaware River, USA." *Water Science and Technology*, 43(12), pp.125-132.

[348] Neelakantan, T. R., and Pundarikanthan, N. V. (2000). "Neural network-based simulation-optimization model for reservoir operation." *Journal of Water Resources Planning and Management*, 126(2), pp.57-64.

[349] NeuralWare. (1991). "Neural Computing, NeuralWorks Professional II/Plus and NeuralWorks Explorer." NeuralWare Inc.

[350] NeuralWare. (1997). "NeuralWorks Professional II/Plus Supplement." *Report*, Aspen Technology Inc., Pittsburg, Pa.

[351] NeuralWare. (1998a). "Advanced Reference Guide." Aspen Technology Inc., USA, 112p.

[352] NeuralWare. (1998b). "Neural Computing: A Technology Handbook for NeuralWorks Professional II/PLUS and NeuralWorks Explorer." Aspen Technology Inc., USA, 324p.

[353] NeuralWare. (1998c). "A Software Reference for Professional II/PLUS and NeuralWorks Explorer." Aspen Technology Inc., USA, 286p.

[354] Oja, E. (1989). "Neural networks, principal components, and subspaces." *International Journal of Neural Systems*, 1, pp.61-68.

[355] Oliver, R. (1994). "The Murrumbidgee River - Algal Growth Potential." in *The Murrumbidgee, Past and Present*, pp.109-117.

[356] Olsen, N. R. B., Hedger, R. D., and George, D. G. (1999). "Modelling the horizontal distribution of algae in a water supply reservoir." *28th Biennial Congress, IAHR*, Graz, Austria.

[357] Olsson, J., Uvo, C. B., and Jinno, K. (2001). "Statistical atmospheric downscaling of short-term extreme rainfall by neural networks." *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere*, 26(9), pp.695-700.

[358] Pachepsky, Y. A., Timlin, D. J., and Ahuja, L. T. (1999). "Estimating saturated soil hydraulic conductivity using water retention data and neural networks." *Soil Science*, 164(8), pp.552-560.

[359] Paerl, H. W. (1996). "A comparison of cyanobacterial bloom dynamics in freshwater estuarine and marine environments." *Phycologia*, 35(6), pp.25-35.

[360] Palisade Corp. (1997). "Decision making tools: @Risk, BestFit." Palisade Corp., New York, 84p.

[361] Pao, Y. H. (1989). "Adaptive Pattern Recognition and Neural Networks." Addison wesley Publishing Company, Reading, M.A., 309p.

[362] Parisi, R., Di Claudio, E. D., Orlandi, G., and Rao, B. D. (1996). "A generalized learning paradigm exploiting the structure of feedforward neural networks." *IEEE Transactions on Neural Networks*, 7(6), pp.1451-1460.

[363] Parker, D. B. (1985). "Learning Logic." *TR-47*, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA.

[364] Parzen, E. (1962). "On estimation of probability density function and mode." *Annals of Math. Statistics*, 33, pp.1065-1076.

[365] Pineda, F. J. (1989). "Recurrent back-propagation and the dynamical approach to neural computation." *Neural Computation*, 1, pp.161-172.

[366] Poff, N. L., Tokar, S., and Johnson, P. (1996). "Stream hydrological and ecological responses to climate change assessed with an artificial neural network." *Limnology and Oceanography*, 41(5), pp.857-863.

[367] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). "Numerical Recipes: The Art of Scientific Computing." Cambridge University Press, Cambridge, U.K., 933p.

[368] Previdi, F., Lovera, M., and Mambretti, S. (1999). "Identification of the rainfall-runoff relationship in urban drainage networks." *Control Engineering Practice*, 7, pp.1489-1504.

[369] Price, K., and Storn, R. (1997). "Differential evolution." *Dr Dobb's Journal*, pp.18-24.

[370] Rajeevan, M., Gahathakurta, P., and Thapliyal, V. (2000). "New models for long range forecasts of summer monsoon rainfall over North West and Peninsular India." *Meteorology and Atmospheric Physics*, 73(3-4), pp.211-225.

[371] Raman, H., and Sunilkumar, N. (1995). "Multivariate modelling of water resources time series using artificial neural networks." *Hydrological Sciences Journal*, 40(2), pp.145-163.

[372] Rapala, J., Sivonen, K., Luukkainen, R., and Niemela, S. I. (1993). "Anatoxin-a concentration in *Anabaena* and *Aphanizomenon* under different environmental conditions and comparison of growth by toxic and non-toxic *Anabaena* strains - a laboratory study." *Journal of Applied Phycology*, 5, pp.581-591.

[373] Ray, C., and Klindworth, K. K. (2000). "Neural networks for agrichemical vulnerability assessment of rural private wells." *Journal of Hydrologic Engineering*, 5(2), pp.162-171.

[374] Recknagel, F. (1997). "ANNA - Artificial neural network model for predicting species abundance and succession of blue-green algae." *Hydrobiologia*, 349, pp.47-57.

[375] Recknagel, F., French, M., Harkonen, P., and Yabunaka, K.-I. (1997). "Artificial neural network approach for modelling and prediction of algal blooms." *Ecological Modelling*, 96, pp.11-28.

[376] Recknagel, F., Fukishima, T., Hanazato, T., Takumura, N., and Wilson, H. (1998). "Modelling and prediction of phyto- and zooplankton dynamics in Lake Kasumigaura by artificial neural networks." *Lakes and Reservoirs: Research and Management*, 3, pp.123-133.

[377] Recknagel, F., Petzoldt, T., Jaeke, O., and Krusche, F. (1994). "Hybrid expert system DELAQUA - a toolkit for water quality control of lakes and reservoirs." *Ecological Modelling*, 71, pp.17-36.

[378] Reed, R. (1993). "Pruning algorithms - A review." *IEEE Transactions on Neural Networks*, 4(5), pp.740-747.

[379] Reed, R. D., and Marks, R. J. (1999). "Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks." MIT Press, Cambridge, M.A., 346p.

[380] Reuter, J. G., and Peterson, R. R. (1987). "Micronutrient effects on cyanobacterial growth and physiology." *New Zealand Journal of Marine and Freshwater Research*, 21, pp.435-445.

[381] Reynolds, C. S. (1984a). "The Ecology of Freshwater Phytoplankton." University Press, Cambridge, 384p.

[382] Reynolds, C. S. (1984b). "Phytoplankton periodicity: the interactions of form, function and environmental variability." *Freshwater Biology*, 14, pp.111-142.

[383] Reynolds, C. S. (1997). "Vegetative Processes in the Pelagic. A Model for Ecosystem Theory." *D-21385*, Ecology Institute, Oldendorf/Luhe.

[384] Riedmiller, M., and Braun, H. (1993). "A direct adaptive method for faster backpropagation learning: The RProp algorithm." *Proceedings of the ICNN 93*, San Francisco.

[385] Robarts, R. D., and Zohary, T. (1987). "Temperature effects on photosynthetic capacity, respiration, and growth rates of bloom forming cyanobacteria." *New Zealand Journal of Marine and Freshwater Research*, 21, pp.391-399.

[386] Rodriguez, M. J., Serodes, J. B., and Cote, P. A. (1997a). "Advanced chlorination control in drinking water systems using artificial neural networks." *Water Supply*, 15(2), pp.159-168.

[387] Rodriguez, M. J., West, J. R., Powell, J., and Serodes, J. B. (1997b). "Application of two approaches to model chlorine residuals in Severn Trent Water Ltd (STW) distribution systems." *Water Science and Technology*, 36(5), pp.317-324.

[388] Rogers, L. L., and Dowla, F. U. (1994). "Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling." *Water Resources Research*, 30(2), pp.457-481.

[389] Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological Review*, 65, pp.386-408.

[390] Rositano, J., and Nicholson, B. C. (1994). "Water Treatment Techniques for Removal of Cyanobacterial Toxins from Water." Australian Centre for Water Quality Research, Salisbury, South Australia.

[391] Ruck, B. M., Walley, W. J., and Hawkes, H. A. (1993). "Biological classification of river water quality using neural networks." *Applications of Artificial Intelligence in Engineering*, 2, pp.362-372.

[392] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning Internal Representations by Error Propagation." in *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, pp.318-362.

[393] Sahai, A. K., Soman, M. K., and Satyan, V. (2000). "All India summer monsoon rainfall prediction using an artificial neural network." *Climate Dynamics*, 16(4), pp.291-302.

[394] Sajikumar, N., and Thandaveswara, B. S. (1999). "A non-linear rainfall-runoff model using an artificial neural network." *Journal of Hydrology*, 216, pp.32-35.

[395] Sanger, T. D. (1989). "Optimal unsupervised learning in a single-layer linear feedforward neural network." *Neural Networks*, 2, pp.459-473.

[396] Sarle, W. S. (1994). "Neural networks and statistical models." *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, pp.1538-1550.

[397] Sarle, W. S. (1997). "Neural Network FAQ." periodic posting to the Usenet newsgroup comp.ai.neural-nets.

[398] Savic, D. A., Walters, G. A., and Davidson, J. W. (1999). "A genetic programming approach to rainfall-runoff modelling." *Water Resources Management*, 13(3), pp.219-231.

[399] Scardi, M. (1996). "Artificial neural networks as empirical models for estimating phytoplankton production." *Marine Ecology Progress Series*, 139, pp.289-299.

[400] Schleiter, I. M., Borchardt, D., Wagner, R., Dapper, T., Schmidt, K. D., Schmidt, H. H., and Werner, H. (1999). "Modelling water quality, bioindication and population dynamics in lotic ecosystems using neural networks." *Ecological Modelling*, 120(2-3), pp.271-286.

[401] Schoof, J. T., and Pryor, S. C. (2001). "Downscaling temperature and precipitation: A comparison of regression-based methods and artificial neural networks." *International Journal of Climatology*, 21(7), pp.773-790.

[402] Schwarz, G. (1978). "Estimating the dimension of a model." *Annals of Statistics*, 6, pp.461-464.

[403] Scott, D. W. (1992). "Multivariate Density Estimation: Theory, Practice and Visualization." Wiley, New York, 317p.

[404] See, L., and Abrahart, R. J. (2001). "Multi-model fusion for hydrological forecasting." *Computers and Geosciences*, 27, pp.987-994.

[405] See, L., and Openshaw, S. (1999). "Applying soft computing approaches to river level forecasting." *Hydrological Sciences Journal*, 44(5), pp.763-778.

[406] Setiono, R. (1997). "A penalty function approach for pruning feedforward neural networks." *Neural Computation*, 9(1), pp.185-204.

[407] Shafron, M., Croome, R., and Rolls, J. (1990). "Water Quality." in *The River Murray*, N. Mackay and D. Eastburn, eds., MDBC, Canberra, Australia, pp.147-165.

[408] Shamseldin, A. Y. (1997). "Application of a neural network technique to rainfall-runoff modelling." *Journal of Hydrology*, 199, pp.272-294.

[409] Shapiro, J. (1984). "Blue-green dominance in lakes: the role and management significane of pH and CO_2." *Int. Revue ges. Hydrobiol.*, 69(6), pp.765-780.

[410] Shapiro, J. (1990). "Biomanipulation: the next phase - making it stable." *Hydrobiologia*, 200/201, pp.13-27.

[411] Sharma, A. (2000). "Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: Part 1 - A strategy for system predictor identification." *Journal of Hydrology*, 239, pp.232-239.

[412] Sharma, A., Lall, U., and Tarboton, D. G. (1998). "Kernel bandwidth selection for a first order nonparametric streamflow simulation model." *Statistic Hydrology and Hydraulics*, 12, pp.33-52.

[413] Shaw, G. R., Sukenik, A., Livine, A., Chiswell, R. K., Smith, M. J., Seawright, R. L., Norris, R. L., Eaglesham, G. K., and Moore, M. R. (1999). "Blooms of the cylindrospermopsin containing cyanobacterium, *Aphanizomenon ovalisporum* (Forti) in newly constructed lakes, Queensland, Australia." *Environ. Toxicol. (in press)*, 14.

[414] Sherman, B., and Webster, I. (1997). "Flow and the control of cyanobacterial growth in river weir pools." *AWWA 17th Federal Convention*, Melbourne, 16-21 March, pp.423-428.

[415] Shi, J. J. (2000). "Reducing prediction error by transforming input data for neural networks." *Journal of Computing in Civil Engineering*, 14(2), pp.109-116.

[416] Shin, H.-S., and J.D., S. (2000). "Regional drought analysis based on neural networks." *Journal of Hydrologic Engineering*, 5(2), pp.145-155.

[417] Shukla, M. B., Kok, R., Prasher, S. O., Clark, G., and Lacroix, R. (1996). "Use of artificial neural networks in transient drainage design." *Transactions of the ASAE*, 39(1), pp.119-124.

[418] Silverman, B. W. (1986). "Density Estimation for Statistics and Data Analysis." Chapman and Hall, New York, 175p.

[419] Silverman, D., and Dracup, J. A. (2000). "Artificial neural networks and long-range precipitation in California." *Journal of Applied Meteorology*, 31(1), pp.57-66.

[420] Simon, E., and Diskin, M. H. (1975). "Objective function formulation and their effect on hydrologic simulation models." *Publ. No. 216*, Faculty of Civil Engineering, Technion - Israel Institute of Technology, Haifa.

[421] Simpson, A. R., Dandy, G. C., and Murphy, L. J. (1994). "Genetic algorithms compared to other techniques for pipeline optimisation." *Journal of Water Resources Planning and Management, ASCE*, 120(4), pp.423-443.

[422] Sivakumar, B., Liong, S. Y., Liaw, C. Y., and Phoon, K. K. (1999). "Singapore rainfall behavior: Chaotic?" *Journal of Hydrologic Engineering*, 4(1), pp.38-48.

[423] Sivonen, K. (1996). "Cyanobacterial toxins and toxin production." *Phycologia*, 35, pp.12-24.

[424] Sivonen, K., Namikoshi, M., Evans, W. R., Gromov, B., Carmichael, W. W., and Rinehart, K. L. (1992). "Isolation and structures of five Microcystins from a Russian *Microcystis Aeruginosa* strain Calu 972." *Toxicon*, 30, pp.1481-1485.

[425] Sivonen, K., Niemela, S. I., Niemi, R. M., Lepisto, L., and Luomo, T. H. (1990). "Toxic cyanobacteria (blue-green algae) in Finnish fresh and coastal waters." *Hydrobiologia*, 190, pp.267-275.

[426] Smith, J., and Eli, R. N. (1995). "Neural-network models of rainfall-runoff processes." *Journal of Water Resources Planning and Management*, 121(6), pp.499-508.

[427] Smith, M. (1993). "Neural Networks for Statistical Modeling." Van Nostrand Reinhold, New York, 235p.

[428] Smith, V. H. (1985). "Predictive models for the biomass of Blue-Green Algae in lakes." *Water Resources Bulletin*, 21(3), pp.433-439.

[429] Smith, V. H., Willen, E., and Karlsson, B. (1987). "Predicting the summer peak biomass of four species of Blue-Green Algae (Cyanophyta/Cyanobacteria) in Swedish lakes." *Water Resources Bulletin*, 23(3), pp.397-402.

[430] Sorooshian, S., Hsu, K. L., Gao, X., Gupta, H. V., Imam, B., and Braithwaite, D. (2000). "Evaluation of the PERSIANN system satellite-based estimates of tropical rainfall." *Bulletin of the American Meteorological Society*, 81(9), pp.2035-2046.

[431] Specht, D. F. (1990). "Probabilistic neural networks." *Neural Networks*, 3(6), pp.110-118.

[432] Specht, D. F. (1991). "A general regression neural network." *IEEE Transactions on Neural Networks*, 2(6), pp.568-576.

[433] Steffensen, D., Burch, M., Nicholson, B., Drikas, M., and Baker, P. (1999). "Management of toxic blue-green algae (cyanobacteria) in Australia." *Inc. Environ Toxicol*, 14, pp.183-195.

[434] Steffensen, D. S., Baker, P. D., and Humpage, A. R. (1993). "Cyanobacterial Blooms in the Murray-Darling Basin: Their Taxonomy and Toxicity." *8/93*, Australian Centre for Water Quality Research, December.

[435] Stewart, W. D. P., P. Rowell, F. R. S., Kerby, N. W., Reed, R. H., and Machray, G. C. (1987). "N_2-fixing cyanobacteria and their potential applications." *Phil. Trans. R. Soc. Lond. B*, 317, pp.245-258.

[436] Stone, M. (1974). "Cross-validatory choice and assessment of statistical predictions." *Journal of the Royal Statistical Society B*, 36, pp.111-147.

[437] Stützle, T., and Hoos, H. H. (2000). "*Max-Min* ant system." *Future Generation Computer Systems*, 16, pp.889-914.

[438] Sullivan, C. (1990). "Phytoplankton." in *The River Murray*, N. Mackay and D. Eastburn, eds., MDBC, Canberra, Australia, pp.251-261.

[439] Sullivan, C., Saunders, J., and Welsh, D. (1988). "Phytoplankton of The River Murray, 1980-1985." *2*, Murray-Darling Basin Commision, Canberra.

[440] Sureerattanan, S., and Phien, H. N. (1997). "Back-propagation networks for daily streamflow forecasting." *Water Resources Journal*, December, pp.1-7.

[441] Suzuki, D. (1997). "The Sacred Balance: rediscovering our place in nature." Greystone Books, Vancouver, British Columbia, 259p.

[442] Takagi, H. (1997). "Introduction to Fuzzy Systems, Neural Networks, and Genetic Algorithms." in *Intelligent Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms*, D. Ruan, ed., Kluwer Academic Publishers, Norwell, Massachusetts, USA, pp.1-33.

[443] Talling, J. F. (1971). "The underwater light climate as a controlling factor in the production ecology of freshwater phytoplankton." *Mitt. Int. Ver. Limnol.*, 19, pp.214-243.

[444] Talling, J. F. (1976). "The depletion of carbon dioxide from lake water by phytoplankton." *Journal of Ecology*, 64, pp.79-121.

[445] Tawfik, M., Ibrahim, A., and Fahmy, H. (1997). "Hysteresis sensitive neural network for modeling rating curves." *Journal of Computing in Civil Engineering*, 11(3), pp.206-211.

[446] Tax, D. M. J., and Duin, P. W. (1998). "Outlier Detection Using Classifier Instability." in *Advances in Pattern Recognition*, A. Amin, D. Dori, P. Pudil, and H. Freeman, eds., Springer, Sydney, pp.593-601.

[447] Taylor, J. G. (1997). "The Historical Background." in *Handbook of Neural Computation*, E. Fiesler and R. Beale, eds., Oxford University Press and the Institute of Physics Publishing, New York, pp.1-7.

[448] Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., and Farmer, J. D. (1992). "Testing for nonlinearity in time series: the method of surrogate data." *Physica D*, 58, pp.77-94.

[449] Thirumalaiah, K., and Deo, M. C. (1998a). "Real-time flood forecasting using neural networks." *Computer-Aided Civil and Infrastructure Engineering*, 13(2), pp.101-111.

[450] Thirumalaiah, K., and Deo, M. C. (1998b). "River stage forecasting using artificial neural networks." *Journal of Hydrologic Engineering*, 3(1), pp.26-32.

[451] Thirumalaiah, K., and Deo, M. C. (2000). "Hydrological forecasting using neural networks." *Journal of Hydrologic Engineering*, 5(2), pp.180-189.

[452] Tokar, A. S., and Johnson, P. A. (1999). "Rainfall-runoff modeling using artificial neural networks." *Journal of Hydrologic Engineering*, 4(3), pp.232-239.

[453] Tokar, S., and Markus, M. (2000). "Precipitation-runoff modeling using artificial neural networks and conceptual models." *Journal of Hydrologic Engineering*, 5(2), pp.156-161.

[454] Toth, E., Brath, A., and Montanari, A. (2000). "Comparison of short-term rainfall prediction models for real-time flood forecasting." *Journal of Hydrology*, 239, pp.132-147.

[455] Tsay, R. S. (1986). "Nonlinearity tests for time series." *Biometrika*, 73, pp.461-466.

[456] Tsintikidis, D., Haferman, J. L., Anagnostou, E. N., Krajewski, W. F., and Smith, T. F. (1997). "A neural network approach to estimating rainfall from spaceborne microwave data." *IEEE Transactions on Geoscience & Remote Sensing*, 35(5), pp.1079-1093.

[457] Van der Molen, D. T., and Pintér, J. (1993). "Environmental model calibration under different specifications: an application to the model SED." *Ecological Modelling*, 68, pp.1-19.

[458] Van Liere, L., and Mur, L. R. (1979). "Chapter 9. Some Experiments on the Competition Between a Green Alga and a Cyanobacterium." Ph.D. Thesis, University of Amsterdam.

[459] Varis, O. (1991). "Associations between lake phytoplankton community and growth factors - a canonical correlation analysis." *Hydrobiologia*, 210, pp.209-216.

[460] Varis, O. (1993). "Cyanobacteria dynamics in a restored Finnish lake: a long term simulation study." *Hydrobiologia*, 268, pp.129-145.

[461] Vasconcelos, V. M., Evans, W. R., Carmichael, W. W., and Namikoshi, M. (1993). "Isolation of microcystin-LR from a Microcystis (cyanobacteria) waterbloom collected in the drinking water reservoir for Porto, Portugal." *Journal of Environmental Science and Health, Part A: Environmental Science and Engineering*, 28(9), pp.2081-2094.

[462] Venkatesan, C., Raskar, S. D., Tambe, S. S., Kulkarni, B. D., and Keshavamurty, R. N. (1997). "Prediction of all India summer monsoon rainfall using error-back-propagation neural networks." *Meteorology & Atmospheric Physics*, 62(3-4), pp.225-240.

[463] Vila, J.-P., Wagner, V., Neveu, P., and Voltz, M. (1999). "Neural network architecture selection: new Bayesian perspectives in predictive modelling. Application to a soil hydrology problem." *Ecological Modelling*, 120, pp.119-130.

[464] Vollenweider, R., and Kerekes, J. (1982). "Eutrophication of Waters, Monitoring, Assessment, Control." Organisation for Economic Co-operation and Development, Paris.

[465] Vollenweider, R. A. (1968). "Scientific Fundamentals of the Eutrophication of Lakes and Flowing Waters, With Particular Reference to Nitrogen and Phosphorous as Factors in Eutrophication." *DAS/SCI/68.27*, OECD, Paris.

[466] Vollenweider, R. A. (1976). "Advances in defining critical loading levels for phosphorus in lake eutrophication." *Mem. Ist. Ital. Idrobiol.*, 33, pp.53-83.

[467] Walker, K. F. (1986). "The Murray-Darling River System." in *The Ecology of River Systems*, R. B. Davies and K. F. Walker, eds., Dr W. Junk Publishers, Dordrecht, pp.631-694.

[468] Walmsley, R. D. (1978). "Factors governing turbidity in a South African reservoir." *Verh. Int. Ver. Limnol.*, 20, pp.1684-1689.

[469] Wan, E. A. (1990). "Temporal backpropagation: An efficient algorithm for finite impulse response neural networks." *1990 Connectionist Models Summer School*, San Mateo, C.A., pp.131-140.

[470] Wang, Q. L., Liu, Y. D., Shen, Y. W., Jin, C. Y., Lu, J. S., Zhu, J. M., Li, S. H., and Ley, S. H. (1991). "Studies on mixed mass cultivation of *Anabaena* spp. (Nitrogen-fixing blue-green algae, cyanobacteria) on a large scale." *Bioresource Technology*, 38(2-3), pp.221-228.

[471] Warner, B., and Misra, M. (1996). "Understanding neural networks as statistical tools." *American Statistician*, 50(4), pp.284-293.

[472] Watson, S., McCauley, E., and Downing, J. A. (1992). "Sigmoid relationships between phosphorous, algal biomass, and algal community structure." *Canadian Journal of Fisheries and Aquatic Sciences*, 49, pp.2605-2610.

[473] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1990). "Predicting the future: A connectionist approach." *International Journal of Neural Systems*, 1(3), pp.193-209.

[474] Wen, C. G., and Lee, C. S. (1998). "A neural network approach to multiobjective optimization for water quality management in a river basin." *Water Resources Research*, 34(3), pp.427-436.

[475] Werbos, P. (1974). "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." PhD Dissertation, Harvard University, Cambridge, Mass.

[476] Werbos, P. J. (1990). "Backpropagation through time: What it is and how to do it." *Proceedings of the IEEE*, Vol. 78, pp.1550-1560.

[477] Wetzel, R. G. (1975). "Limnology." Saunders, Philadelphia.

[478] White, H. (1989). "Learning in artificial neural networks: A statistical perspective." *Neural Computation*, 1, pp.425-464.

[479] Whitehead, B., and Choate, T. D. (1994). "Evolving space-filling curves to distribute radial basis functions over an input space." *IEEE Transactions on Neural Networks*, 5(1), pp.15-23.

[480] Whitehead, B. A., and Choate, T. D. (1996). "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction." *IEEE Transactions on Neural Networks*, 7(4), pp.869-880.

[481] Whitehead, P. G., Howard, A., and Arulmani, C. (1997). "Modelling algal growth and transport in rivers - a comparison of time series analysis, dynamic mass balance and neural network techniques." *Hydrobiologia*, 349, pp.39-46.

[482] Widrow, B., and Hoff, M. E. (1960). "Adaptive switching circuits." *1960 IRE WESCON Convention Record*, New York, pp.96-104.

[483] Williams, R. J., and Zisper, D. (1989). "A learning algorithm for continually running fully recurrent neural networks." *Neural Computation*, 1, pp.270-280.

[484] Williamson, J. R. (1995). "Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps." *Tecnical Report CAS/CNS-95-003*, Boston University, Center of Adaptive Systems and Department of Cognitive and Neural Systems, Boston.

[485] World Health Organisation. (1984). "Guidelines for Drinking Water Quality.", Geneva.

[486] Wright, A. H. (1991). "Genetic Algorithms for Real Parameter Optimization." in *Foundations of Genetic Algorithms*, J. E. Rawlins, ed., Morgan Kaufmann, San Mateo, pp.205-218.

[487] Wu, X. D., Cao, H. X., Flitman, A., Wei, F. Y., and Feng, G. L. (2001). "Forecasting monsoon precipitation using artificial neural networks." *Advances in Atmospheric Sciences*, 18(5), pp.950-958.

[488] Xiao, R. R., and Chandrasekar, V. (1997). "Development of a neural network based algorithm for rainfall estimation from radar observations." *IEEE Transactions on Geoscience & Remote Sensing*, 35(1), pp.160-171.

[489] Yabunaka, K. i., Hosomi, M., and Murakami, A. (1997). "Novel application of a back-propagation artificial neural network model formulated to predict algal bloom." *Water Science and Technology*, 36(5), pp.89-97.

[490] Yang, C. C. (2000). "Artificial neural networks for subsurface drainage and subirrigation systems in Ontario, Canada." *Journal of the American Water Resources Association*, 36(3), pp.609-618.

[491] Yang, C. C., Lacroix, R., and Prasher, S. O. (1998). "The use of backpropagation neural networks for the simulation and analyses of time series data in subsurface drainage systems." *Transactions of the ASAE*, 41(4), pp.1181-1187.

[492] Yang, C. C., Prasher, C. O., and Lacroix, R. (1996). "Applications of artificial neural networks to land drainage engineering." *Transactions of the ASAE*, 39(2), pp.525-533.

[493] Yang, H. H., Van Vuuren, S., Sharma, S., and Hermansky, H. (2000). "Relevance of time-frequency features for phonetic and speaker-channel classification." *Speech Communication*, 31, pp.35-50.

[494] Yao, X. (1993). "A review of evolutionary artificial neural networks." *International Journal of Intelligent Systems*, 8(4), pp.539-567.

[495] Yao, X. (1995a). "Evolutionary Artificial Neural Networks." in *Encyclopaedia of Computer Science and Technology*, A. Kent and J. G. Williams, eds., Marcel Dekker Inc., New York, pp.137-170.

[496] Yao, X. (1995b). "A new simulated annealing algorithm." *International Journal of Computer Mathematics*, 56, pp.161-168.

[497] Yao, X., and Liu, Y. (1997a). "EPNet for chaotic time-series prediction." *Proc. of the First Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'96)*, Taejon, Korea, 9-12 November 1996, pp.331-342.

[498] Yao, X., and Liu, Y. (1997b). "A new evolutionary system for evolving artificial neural networks." *IEEE Transactions on Neural Networks*, 8(3), pp.694-713.

[499] Yao, X., and Liu, Y. (1998a). "Making use of population information in evolutionary artificial neural networks." *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 28(3), pp.417-425.

[500] Yao, X., and Liu, Y. (1998b). "Towards designing artificial neural networks by evolution." *Applied Mathematics and Computation*, 91(1), pp.83-90.

[501] Zealand, C. M., Burn, D. H., and Simonovic, S. P. (1999). "Short term streamflow forecasting using artificial neural networks." *Journal of Hydrology*, 214, pp.32-48.

[502] Zhang, B., and Govindaraju, R. S. (2000). "Prediction of watershed runoff using Bayesian concepts and modular neural networks." *Water Resources Research*, 36(3), pp.753-762.

[503] Zhang, M., Fulcher, J., and Scofield, R. A. (1997). "Rainfall estimation using artificial neural network group." *Neurocomputing*, 16, pp.97-115.

[504] Zhang, Q., and Stanley, S. (1999). "Real-time water treatment process control with artificial neural networks." *Journal of Environmental Engineering-ASCE*, 125(2), pp.153-160.

[505] Zhang, Q., and Stanley, S. J. (1997). "Forecasting raw-water quality parameters for the North Saskatchewan River by neural network modeling." *Water Research*, 31(9).

[506] Zheng, G. L., and Billings, S. A. (1996). "Radial basis function network configuration using mutual information and the orthogonal least squares algorithm." *Neural Networks*, 9(9), pp.1619-1637.

Corrigenda for

Forecasting Water Resources Variables Using Artificial Neural Networks

by

Gavin J. Bowden

On page 2, the sentence:
"Many researches in the field of water resources modelling..."

should be replaced with:
"Many researchers in the field of water resources modelling..."

On page 4, the sentence:
"A better understanding of salinity and cyanobacteria in the River Murray and the factors that affect their occurrence can be also be obtained from ANN modelling."

should be replaced with:
"A better understanding of salinity and cyanobacteria in the River Murray and the factors that affect their occurrence can also be obtained from ANN modelling."

On page 17, the figure caption:
"Figure 2.4 (a) A Feedforward Neural Network and (b) A Feedback Neural Network (source: Takagi, 1997)"

should be replaced with:
"Figure 2.4 (a) A Feedback Neural Network and (b) A Feedforward Neural Network (source: Takagi, 1997)"

On page 23, Equation 2.6:

$$f(x) = \frac{1}{1+\exp(-x)}$$

should be replaced with:

$$f(zin_j) = \frac{1}{1+\exp(-zin_j)}$$

On page 42, the sentence:
"The weights on the A summation layer PE keep count..."

should be replaced with:
"The weights on the *A* summation layer PE keep count..."

On page 79, the sentence:
"To simplify the calculation it is common to set $\tau = 1$."

should be replaced with:
"To simplify the calculation it is common to set $\tau = 1$ (Brock et al., 1996)."

On page 90, the sentence:
"In this way, an optimal number of hidden nodes..."

should be replaced with:
"In this way, an appropriate number of hidden nodes..."

On page 317, insert the following paragraph beneath the first paragraph:
"It is important to note that retraining the model using the GRNN outlier detection method enabled the model to better predict the uncharacteristic salinity peak that occurred during the third year. As expected, the prediction of this peak was significantly better than the model that utilised no retraining (Figure 4.101)."
