PARALLEL DISTRIBUTED PROCESSING IN ROCK ENGINEERING SYSTEMS

Dean Lee Millar
B.Eng. (Hons), ARSM

A thesis submitted for the award of the degree of Doctor of Philosophy (Ph.D.) of the University of London and the Diploma of Imperial College of Science, Technology & Medicine (DIC)

Department of Earth Science & Engineering
Royal School of Mines
Imperial College of Science, Technology & Medicine
London SW7 2BP

May 2008

Abstract

Rock Engineering Systems are a collection of ideas, mathematical tools and computer technology all of which are designed to solve problems in rock engineering with interacting components. The interactions between components can be complex and the rock engineering problems themselves contain a high degree of uncertainty. The research described in this thesis investigates the incorporation of computational techniques known as parallel distributed processing methods into the disciplines of rock mechanics and rock engineering. Two main applications of parallel distributed processing methods in rock engineering are investigated in this thesis.

1) Multilayered perceptron artificial neural networks are used successfully to encapsulate the laboratory behaviour of rocks under triaxial compression. Trained artificial neural networks are then used to replace conventional constitutive models within finite difference geomechanical numerical modelling codes.

2) Two multilayered perceptron artificial neural networks are developed to assist in the task of discrimination of rock fracture presence within digital imagery of rock exposures. The first is trained using samples of the image that contain fracture image content and samples that do not, and provides a probability-like measure of fracture presence. It was sufficiently successful to permit estimation of fracture intensity parameter $P_{2,2}$. The second was developed specifically to identify fracture termination condition by matching samples to a set of fracture termination condition templates.

Seven original contributions to the rock mechanics and rock engineering disciplines have resulted across the three application areas. These contributions are itemised, with details, at the beginning of the final Chapter of the thesis.

Declaration

I declare that all work presented in this thesis is my own, except where acknowledged.

Acknowledgements

This research was sponsored in part by a REMIT scholarship funded by the US Army Corps of Engineers via its European Research Office. Later parts of the work were supported by my employers, Imperial College and Camborne School of Mines.

I would like to acknowledge all those who have helped me during the time I have been engaged in the research. There are many people to whom I am indebted but the following individuals deserve special mention.

I am grateful to both Prof. John A. Hudson, my supervisor, who provided the inspiration, and organised the original funding, for the research and Dr John Harrison, my co-supervisor, with whom I have had many useful and supportive discussions. These individuals are also responsible for the provision of a research environment that I found both stimulating and exciting and will never forget throughout my life. My thanks also go to Prof. Bob Pine and my wife Dr Caterina Noula, who told me to get my thesis finished, or else!

Table Of Contents

Abstract i
Declaration ii
Acknowledgements ii
Table of Contents iii
List of Figures vii
List of Tables xii
Nomenclature xiii

1 Introduction 1
1.1 Purpose of this thesis 1
1.2 Parallel Distributed Processing 1
1.3 Rock Engineering Systems 2
1.4 Application Areas for Parallel Distributed Processing 5
1.4.1 Area A: Rock constitutive behaviour 5
1.4.2 Area B: Rock mass geometry modelling 5
1.5 Structure and Content of This Thesis 6

2 Artificial Neural Networks 9
2.1 Introduction 9
2.2 Artificial neural network overview 9
2.3 Structure of Artificial Neural Networks 10
2.3.1 Microstructure 10
2.3.2 Mesostructure 12
2.3.3 Macrostructure 15
2.4 Using Artificial Neural Networks 18
2.4.1 Suitability 18
2.4.2 Data required 19
2.4.3 Specification of Input and Output Quantities 20
2.4.4 Pre-processing 20
2.4.5 Information flow 20
2.4.6 Post-processing 22
2.5 Training Artificial Neural Networks 22
2.5.1 The Hard Learning Problem 22
2.5.2 Training Algorithms 23
2.5.3 Heuristic Aspects of Training Artificial Neural Networks 25
2.5.4 Termination of Training and Generalisation Issues 26
2.6 Multilayer Perceptron Using the Generalised Delta Rule and Back-Propagation of Error 26
2.6.1 Feedforward operation 27
2.6.2 Training using backpropagation of error 28
2.6.3 Practical training issues 31
2.6.4 Mesostructural complexity in pattern classification 31
2.7 Learning Vector Quantization 33
2.7.1 Topological correspondence 33
2.7.2 LVQ General description 33
2.7.3 LVQ Initialisation 35
2.7.4 LVQ Training 35
2.7.5 LVQ Validation 36
2.8 Sammon Mappings 36
2.9 Self-Organising Maps 38
2.9.1 SOM Training 39
2.9.2 SOM Initialisation and Validation 39
2.9.3 SOM Calibration 40
2.10 Conclusions – Types of ANN and their potential for rock mechanics modelling 40

3 Review of Applications of Artificial Neural Networks in Geotechnical Engineering 42
3.1 ANNs in general engineering practice 42
3.2 Survey of applications in geotechnical engineering 43
3.2.1 Constitutive Modelling 43
3.2.2 Constitutive behaviour and numerical methods 44
3.2.3 Rock Characterisation 44
3.2.4 Geological identifcation 45
3.2.5 Rock-Structure Performance 46
3.2.6 Plant control and performance 48
3.2.7 Blasting / Seismic Geophysics 49
3.2.8 Geostatistical Applications 50
3.2.9 Soil Classification 50
3.2.10 Soil hydrogeology 51
3.3 Survey of perceived benefits 51
3.4 Compound approaches 55
3.5 Recent developments 57
3.6 Summary 60

4 Training ANNs with data from simulated compression tests on linearly elastic rock 62
4.1 Introduction 62
4.2 General comments on the constitutive behaviour of rocks 63
4.3 Constitutive framework 67
4.3.1 Linear elasticity 67
4.3.2 Triaxial test procedure 69
4.3.3 Simulation of initial hydrostatic compression 69
4.3.4 Simulation of deviatoric compression 70
4.4 Triaxial Compression Test Simulation 70
4.4.1 Simulation procedure 70
4.4.2 Modelling experimental variance 71
4.4.3 Triaxial test schedule 72
4.4.4 Plane strain 'triaxial' test schedule 72
4.5 Data pre-processing 73
4.5.1 Normalisation procedure 73
4.5.2 Cyclic permutation of inputs and outputs 75
4.6 ANN Training Log 76
4.7 Training and Validation Results 77
4.7.1 Method of graphical presentation 77
4.7.2 Progression of the ANN training performance 77
4.7.3 Performance of the trained ANN. 80
4.7.4 Performance of the trained ANN with simulated data from different rock types 82
4.8 Effect of varying the normalisation parameters 84
4.9 Effect of varying the type and number of simulated tests used to synthesise data sets 86
4.10 Effect of improving the degree of experimental control simulated 88
4.11 Discussion 91

5 Numerical modelling with Artificial Neural Networks using *FLAC* 93
5.1 Introduction 93
5.1.1 Chapter Outline 94
5.2 Numerical Modelling & *FLAC* 94
5.2.1 Choice of Numerical Model 94
5.2.2 Characteristics of FLAC 96
5.2.3 Summary of FLAC calculation sequence 97
5.3 Encoding the User Defined Constitutive Model 99
5.3.1 Input and output quantities 100
5.3.2 Material properties 100

5.3.3 Structure of User Defined Constitutive Model 100
5.3.4 Tests to verify a UDM 102
5.3.5 Summary of benchmark validation test results defining criteria for acceptable performance of the ANN UDM 105
5.4 Verifying the ANN user defined constitutive model with FLAC 107
5.4.1 Need for tensor transformations arising from ANN data source 107
5.4.2 Preparation of the FISH data files encoding the ANN UDM 108
5.4.3 Performance of the ANN UDM 111
5.4.4 Further verification checks 121
5.5 Training an ANN to learn real constitutive behaviour 124
5.5.1 Crosslands Hill Sandstone – Conventional Treatment 124
5.5.2 The FLAC constitutive model for Crosslands Hill sandstone. 129
5.5.3 Preparing the Crosslands Hill sandstone triaxial test data for ANN training 130
5.5.4 Results of Training and Validation 138
5.6 Testing the ANNs trained on Crosslands Hill sandstone data 142
5.6.1 Network 1 ANN UDM performance under Verification Test Type 1, Case 1 142
5.6.2 Network 1 ANN UDM performance under Verification Test Type 1, Case 2 143
5.6.3 Network 1 ANN UDM performance under Verification Test Type 1, Case 3 144
5.6.4 Network 1 ANN UDM performance under a variant of Verification Test Type 2 146
5.6.5 General comments on the Crosslands Hill sandstone ANN UDM 149
5.7 Discussion 149

6. Fracture Identification from Digital Imagery with Artificial Neural Networks 153
6.1 Introduction 153
6.2 The Fracture Discriminator 155
6.2.1 Data set for the Fracture Discriminator 155
6.2.2 Fracture discriminator ANN specifics 157
6.2.3 Performance of the Fracture Discriminator 158
6.2.4 A fracture intensity measure based on fracture discriminator output 165
6.3 The Fracture Classifier 168
6.3.1 General concept of the Fracture Classifier 168
6.3.2 Data set for the Fracture Classifier - The Fracture Alphabet 169
6.3.3 Fracture classifier ANN details 173
6.3.4 Fracture classifier post-processing 174
6.3.5 Image noise: a clarification 175
6.3.6 The effect of additive noise on fracture alphabet characters 176
6.3.7 Performance of the Fracture Classifier 178
6.3.8 Recognition accuracy for individual fracture characters with varying brightness and contrast. 183
6.4 Sample image patch transformations 186
6.5 Testing the Fracture Classifier with real rock mass imagery 186
6.5.1 Image scanning procedure 186
6.5.2 Results of a Fracture Classifier scan of Figure 6.2 187
6.6 Effect of deliberately displacing 'stroke pixels' in the ANN training data on Fracture Classifier performance 191
6.6.1 Effect of mutation of Fracture Classifier performance 192
6.6.2 Options for fracture map assembly after post-processing 198
6.6.3 Further work for the Fracture Classifier 201
6.7 Conclusions 202

7 Contributions, Discussion and Conclusions 204
7.1 Original Contributions 204
7.2 Discussion 205
7.2.1 The Rock Engineering System 205
7.2.2 Parallel Distributed Processing in Rock Engineering Systems. 208

7.2.3 Review of Objectives: Rock Constitutive Behaviour 210
7.2.4 Review of Objectives: Rock Mass Geometry Modelling 211
7.3 Conclusions 211

8 References 214

List of Appendices

A VBA code used to simulate triaxial and polyaxial ('plane strain') compression tests 229

B1 Detailed worked example of FLAC calculation sequence 238
B2 FLAC commands data file for single zone verification example 252
B3 Spreadsheet calculations for single zone verification example 254
B4 FLAC data files for verification Test Types 1,2 and 3 260
B5 FLAC results from ANN UDM verification tests 273
B6 Procedures to facilitate use of ANN UDMs in FLAC 287
B7 FLAC/FISH code listings for ANN UDM modelling Crosslands Sandstone 301
C1 Image processing fundamentals 314
C2 Fracture Discriminator Weights and Thresholds 321
C3 Classification of edge detection image processing filters 329
C4 Visual Basic code to reproduce the Fracture Alphabet 330
C5 Weighted Image Reduction algorithm 333
C6 Contrast stretching algorithm 339
C7 Program listing of the Fracture Character mutation algorithm 342
C8 Fracture Classifier Weights and Thresholds 346

List of Figures

Figure 1.1: Rock mass system concept. Rock mass responds to, or processes, input perturbations producing output behaviour. Conservation rules are implied thus a dissipating mechanism introduced 3
Figure 1.2: Rock engineering system concept 4
Figure 1.3: Thesis 'road-map' 7

Figure 2.1: Common activation functions for artificial neurons 11
Figure 2.2: Transformation of sigmoidal activation function through manipulation of function parameters. See equation (2.2) in Section 2.6.1 for meaning of 'theta j' and 'theta o' 12
Figure 2.3: Multi-layered perceptron mesostructure 13
Figure 2.4: Diagram of the mesostructure of a Bi-directional Associative Memory ANN. 14
Figure 2.5: Diagram illustrating Hopfield network mesostructure 15
Figure 2.6: Diagram illustrating "Brain-state-in-a-box" ANN mesostructure. Note recurrent connections at the network nodes. 15
Figure 2.7: Diagram showing the relation between ANN microstructure, mesostructure and macrostructure. 16
Figure 2.8: Conceptual diagram of an ART 2 network developed by Carpenter & Grossberg (1987). .. 17
Figure 2.9: The role of hidden artificial neurons in multilayered perceptrons for two inputs (from Lippman, 1987). 32
Figure 2.10 : Diagrammatic illustration of the LVQ concept for a simple 2 dimensional case. Solid circles = 'IN'; Open circles = 'OUT' 34
Figure 2.11: Illustration of the Sammon Mapping (inter-vertex distance preservation) for a 3D polyhedron in comparison to orthographic projection (2 co-ordinates preserved) onto the 2D plane. 37
Figure 2.12 : Example Sammon Mapping of a 3D helix, projected onto the 2D plane. 38

Figure 4.1 : Scatter plots of target normalised $\Delta\sigma_z$ versus ANN output normalised $\Delta\sigma_z$ for the 3-6-3 ANN of Table 4.5. The x-axis of each plot is the target output; the y –axis of each plot is the actual output. The diagonal line presented in each plot shows the ideal 'locus' on which the crosses should lie, for perfect ANN training. The six plots illustrate the quality of ANN recall of the training data, as training iterations proceed. 78
Figure 4.2 : Performance of the 3-6-3 ANN of Table 4.5 during the training process. 80
Figure 4.3. Training and validation performance of the 3-6-3 ANN of Table 4.5. Top row: training results a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$. Middle row, validation results d) $=\Delta\sigma_x$, e) $=\Delta\sigma_y$, f) $=\Delta\sigma_z$. Bottom row, validation results with plane strain simulation, g) $=\Delta\sigma_x$, h) $=\Delta\sigma_y$, i) $=\Delta\sigma_z$. The x-axis of each plot is the target output; the y –axis of each plot is the actual output. The diagonal line presented in each plot shows the ideal 'locus' on which the crosses should lie, for perfect ANN training. 81
Figure 4.4 : Results of testing the 3-6-3 ANN of Table 4.5 with simulated triaxial compression test data for a rock with Young's modulus = 1GPa and Poisson ratio 0.45. a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$ (all normalised). Meaning of axes: as for Figures 4.1 and 4.3. 82
Figure 4.5: Results of testing the 3-6-3 ANN of Table 4.5 with simulated triaxial compression test data for a rock with Young's modulus = 90GPa and Poisson ratio 0.05. a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$ (all normalised). Meaning of axes: as for Figures 4.1 and 4.3. 83
Figure 4.6: Artificial neuron activation function, linear approximation and best fit. 85
Figure 4.7. Validation performance of a 3-6-3 ANN under plane strain constraint a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$. Normalised output values confined to the range –0.3 to 0.3. Meaning of axes: as for Figures 4.1 and 4.3. 86
Figure 4.8: Ideal, limited set of training exemplars for an ANN learning isotropic linear elastic behaviour of rock with $E = 40$ GPa and $\nu = 0.25$ 90

Figure 5.1 : Conceptual diagram illustrating the use of an ANN in place of an in-built constitutive model within a geomechanical code. (Modified from Itasca, 1993) 93
Figure 5.2 : Summary of calculation sequence for *FLAC* and *FLAC*-like finite difference numerical methods. 98

Figure 5.3: A UDM for isotropic linear elasticity coded in *FISH*..... 101
Figure 5.4: *FLAC* output showing model set up for Type 2 verification test. Rectangular quadrilateral zones using *FLAC* in-built constitutive model are in light grey on LHS of grid; zones using UDM constitutive model in darker grey on RHS. 104
Figure 5.5: *FLAC* output showing model set up for Type 3 verification test. Irregular quadrilateral zones using *FLAC* in-built constitutive model are in light grey on LHS of grid; zones using UDM constitutive model in darker grey on RHS..... 104
Figure 5.6: *FLAC* time step plot of differences in four stress tensor component magnitudes (mds**) monitored at two symmetrically located zones within the model, up to 500 time steps for the Test Type 2 UDM verification model. For example, mdsyy monitors the difference between σ_{yy} magnitudes for a zone of the LHS of the model in Figure 5.4 and a zone on the RHS of the model in Figure 5.4. 106
Figure 5.7: *FISH* code fragment showing implementation of isotropic, linear elastic constitutive law.109
Figure 5.8: *FISH* code fragment, showing ANN code within mode 2 invocation section of a *FLAC* user defined constitutive model..... 110
Figure 5.9: Results for Case 1 loading with ANN UDM: stress tensor components versus normal strain in y direction. Gradients of these lines are 48.5 GPa for σ_{yy}, 17 GPa for σ_{xx} and σ_{zz}. Corresponding values with E = 40 GPa and ν = 0.25 as prescribed are 48 GPa and 16 GPa. σ_{xy} = 0, as expected (horizontal line)..... 112
Figure 5.10: Results for Case 2 loading with ANN UDM: shear stress versus [tensorial] shear strain. Gradient of the line is 34.4 GPa against an expected value of $2G$ = 32 GPa..... 112
Figure 5.11: Results for Case 3 loading using ANN UDM: stress tensor components versus strain in y direction. Gradients of the lines are consistent with component loading cases (Figures 5.9 and 5.10) 113
Figure 5.12 : Results for Case 3 loading using ANN UDM: Shear stress versus [tensorial] shear strain. Gradients of the lines are consistent with component loading cases (Figures 5.9 and 5.10) 113
Figure 5.13: FLAC time step history plot of normal stress components at the two monitored locations in the model in Test Type 2 showing good, but not perfect agreement; the ANN UDM curves indicate a minor over-prediction of stiffness (consistent with the findings of Test Type 1 verification runs). 115
Figure 5.14: FLAC time step history plot of shear stress components at the two monitored locations in the model in Test Type 2. Although there is qualitatively similar behaviour presented, symmetry breaking is apparent from 50 time steps onward. 115
Figure 5.15: *FLAC* time step history plot of differences in four stress component magnitudes (mds**) monitored at two symmetrically located zones within the model up to 500 time steps for the Test Type 2 UDM verification model with the ANN UDM used on RHS of the model. Trends in disparities are consistent with over-prediction of stiffness identified in Test Type 1 verification. 116
Figure 5.16: Plot showing grid node displacement vectors after 500 time steps for Test Type 2 UDM verification model with the ANN UDM used on the RHS of the model. 116
Figure 5.17: Plot of normal stress in the y direction versus strain in the y direction for the symmetrically positioned monitoring points including unload and tension test regime. The results indicate good agreement and consistent behaviour, but ideally the curves should overlie one another..... 118
Figure 5.18: *FLAC* time step plot of differences in stress component magnitudes for loading to 500 time steps and unloading for a further 800 time steps for the Test Type 2 verification test. Crossing point of all six curves at around 1100 time steps indicates that the initial stress-free state is recovered on the unload cycle. Disparities between corresponding stress components also reduce as unloading progresses..... 119
Figure 5.19: *FLAC* time step plot of stress component magnitudes for loading to 500 time steps and unloading for a further 800 time steps for the Test Type 3 verification test with the ANN UDM used on the RHS of the model. This plot (and others not presented) indicates complete consistency with the results of Test Type 2 verification..... 120
Figure 5.20: Plot of *FLAC* grid and grid node displacement vectors at 500 time steps for the Test Type 3 UDM verification model with the ANN UDM used on the RHS of the model. Grid point displacements throughout the model have a coherent trend and the differences in displacement magnitudes and directions are small, across the model. 121
Figure 5.21: Plot showing values of major principal stress increment from ANN UDM ($zds1) versus elastic values ($zds1e), both computed within the ANN UDM, for zone (6,8) within the Test

Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value. 122
Figure 5.22: Plot showing values of intermediate principal stress increment from ANN UDM ($zds2) versus elastic values ($zds2e), both computed within the ANN UDM, for zone (6,8) within the Test Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value. 122
Figure 5.23: Plot showing values of minor principal stress increment from ANN UDM ($zds3) versus elastic values ($zda3e), both computed within the ANN UDM, for zone (6,8) within the Test Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value. 123
Figure 5.24: Axial stress versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve). It is evident that in the immediate post peak regions of the tests conducted with confining pressures 35 and 4.5 MPa, the testing system was not under full control. 125
Figure 5.25: Negative radial strain (i.e., extensile strain plotted as positive for convenience) versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve). 126
Figure 5.26: Volumetric strain versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve). Dilatant volumetric strain lies below the horizontal axis 127
Figure 5.27: Dependence of elastic parameters on confining pressure for Crosslands Hill sandstone rock 128
Figure 5.28: Plot of peak axial stress versus test confining pressure for Crosslands Hill sandstone (discrete points). Continuous line shows the Hoek-Brown failure criterion can be adequately fitted to the test data. 128
Figure 5.29: Determination of shear dilatancy angle from triaxial test results (after, Vermeer and de Borst, 1984) 129
Figure 5.30: Plot of change in axial displacement versus axial displacement for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. The plot shows that quantisation of experimentally observed values is an important factor for ANNs with incremental formulations 134
Figure 5.31: Plot of change in radial displacement versus radial displacement for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. The plot shows that quantisation of experimentally observed values is an important factor for ANNs with incremental formulations 134
Figure 5.32: Plot of change in axial load versus axial load for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. This plot shows that quantisation of axial load values is not as serious a problem for training ANNs with incremental formulations as displacement observations. 135
Figure 5.33: Piece-wise linear representation of constitutive behaviour for training 136
Figure 5.34: Graphical presentation of training error (normalised output) for Networks 1 to 4. For each plot, the x-axis represents target output and the y-axis represents actual output. These results indicate good training for NET3 and NET4 and poor training of NET1 and NET2. 141
Figure 5.35: Crosslands Hill sandstone ANN UDM results for Case 1 loading: Stress versus strain in y direction. Gradients of lines are $\lambda_1 = 14.6$ GPa ; $\lambda_2 = 5.7$ GPa, consistent with elastic properties determined in Section 5.5.1. 143
Figure 5.36: Crosslands Hill sandstone ANN UDM results for Case 2 loading: Shear stress versus tensorial shear strain. A surprising degree of linearity is obtained. Gradient of curve is 16 GPa implying a shear modulus of 8 GPa, around 2 GPa lower than obtained in Section 5.6.1 144
Figure 5.37: Crosslands Hill sandstone ANN UDM results for Case 3 loading (combination of Test Type 1 and 2 loading) : Stress versus normal strain in the y-direction. These results are as expected, that is, a combination of previous 2 cases. 145
Figure 5.38: Crosslands Hill sandstone ANN UDM results for Case 3 loading. (Shear stress versus tensorial shear strain). Gradient is lower than comparable case, Case 2 (Figure 5.36). 146
Figure 5.39: *FLAC* time step history plot of normal stress components versus average normal strain in the y-direction for the variant of Test Type 2. Confining pressure 35 MPa. *FLAC* sign convention produces the curve 'up-side-down' and 'back-to-front' in comparison to conventional means of presentation (Figure 5.24). σ_{yy} (Ave SYY) curve shows a clear peak at 30 $m\varepsilon$ axial

compression. Note: simulation does not present a 'real' triaxial compression test due to plane strain associated with 2D *FLAC* modelling. 147
Figure 5.40: *FLAC* grid plot showing grid node displacements after 8000 cycles (in the 'post-peak' region, fixity conditions and applied forces for the variant of verification Test Type 2. The diagram shows that the behaviour of the ensemble is consistent with that expected. 148

Figure 6.1: Illustration of the relative difficulty of an optical character recognition task (upper part) in comparison with a fracture recognition task (lower part). 154
Figure 6.2: Image sampling process for Fracture Discriminator. Solid squares: samples of image content containing a fracture. Dashed squares: samples of image content not containing a fracture. 155
Figure 6.3: Training data set for Fracture Discriminator, sampled from Figure 6.2. 156
Figure 6.4: ANN training concept for the Fracture Discriminator 157
Figure 6.5: Results of using the Fracture Discriminator to scan the original image, with other image processing operations for comparison. 159
Figure 6.6: Production of a fracture trace map using the Fracture Discriminator after post-processing with a generic raster to vector conversion utility. 162
Figure 6.7: Plot of fracture intensity $P_{2,2}(A)$ versus normalised sampling area, A'. The light continuous line, P22c(A), shows fracture intensity calculated at different scales, centred on the fracture probability map, Figure 6.5e. The heavy continuous line shows the mean $P_{2,2}(A)$ calculated from all possible sub-samples of the fracture probability map Figure 6.5e. 95% confidence limits are shown with dashed lines. The plot can be interpreted to yield a scale invariant estimate of mean fracture intensity $\overline{P}_{2,2} \approx 0.212$ m^2/m^2. 167
Figure 6.8: Illustration of expectation guided search with imagery for features matching defined 7 x 7 templates 168
Figure 6.9: Template for fracture characters of the fracture alphabet. 169
Figure 6.10: Four examples of characters from the fracture alphabet 170
Figure 6.11 The Fracture Alphabet 172
Figure 6.12: Distribution of the coefficient of variation of weights within six networks trained to reproduce the fracture alphabet. The average value of around 10 indicates that the magnitude of the standard deviation of the weight values is 10 times the mean of the values for network weights in the same 'location' within the ANN. 174
Figure 6.13: Distributions of ANN input values subject to additive random noise. Histograms taken over the complete fracture alphabet. Average frequencies taken over 5 simulations. 177
Figure 6.14: Showing the effect of increased additive noise on fracture character image content 178
Figure 6.15 Fracture character recognition performance with type (i) post-processing, plotted against Θ_2 varying between 0.05 and 0.45 for increasing amounts of additive noise. Error bars show range of recognition accuracy across 5 sets of simulations. With additive noise less than 50% (see Figure 6.14) and Θ_2 =0.45, recognition accuracy is maintained above 85%. 179
Figure 6.16: Fracture character recognition performance with type (i) post-processing and Θ_2 =0.45, which confirms that recognition accuracy is maintained above 85% even when additive noise is 50%. Numbers on the diagram report recognition accuracy (%). 180
Figure 6.17: Comparison between Fracture Classifier performance for type(i) [Θ_2 =0.45.] and type(ii) post-processing. 181
Figure 6.18: Average recognition accuracy for type (i) and type (ii) post-processing for (top) 30% additive noise with 95.7% agreement between the two methods and (bottom) 50% additive noise with 37.6% agreement. Where dark grey columns and light grey columns meet, 100% recognition accuracy is obtained by both type (i) and type (ii) post-processing methods. 182
Figure 6.19: Graph showing the percentage of 100% agreement between post-processing methods for increasing additive noise. Percentage agreement reported next to data points. 183
Figure 6.20: Image scanning procedure with fracture classifer. Illustration shows a patch size of 7 x 7 pixels and a gap width of 4 pixels. 187
Figure 6.21: Binary fracture maps resulting from application of Fracture Classifier to Figure 6.2, for varying patch sizes and gap widths 189
Figure 6.22: Fracture recognition performance for radiometric noise free input patches, but with displaced stroke pixels. [Θ_2 =0.30], illustrating one reason why the Fracture Classifier performance is poor (with the three lower examples) 190
Figure 6.23: Illustration of character mutation process for character stroke 8 191

Figure 6.24: Example mutation transformation of fracture characters 34 and 144. a) and c): mutation rate = 0.2; b) and d) : mutation rate = 0.5 192
Figure 6.25: Fracture recognition performance of Fracture Classifier trained with mutated fracture character training patterns on the same test patterns as shown in Figure 6.22. [Θ_2 =0.30.]. Improved performance compared with 6.22 should be noted 193
Figure 6.26: Results of application of Fracture Classifier trained with mutated fracture characters to Figure 6.2, for varying patch sizes and gap widths 194
Figure 6.27: Examples of using the Fracture Classifier on selected samples from Figure 6.2. Example 1 : 'horizontal' line segment; Example 2 : 'vertical' line segment; Example 3 : X-intersection of two line segments; Example 4: 'diagonal' line segment. 196
Figure 6.28: Further examples of using the Fracture Classifier on selected samples from Figure 6.2. Example 5 : 3 intersecting line segments; Example 6 : termination in intact rock; Example 7 : T-termination of a fracture against another fracture; Example 8: re-centred example 7. 197
Figure 6.29: Results of application of Fracture Classifier utilising five options for fracture map assembly. a) Original image (Synthetic). b) Single-pass direct replacement (Option 1). c) Two-pass direct replacement (Option 2). d) Four-pass direct replacement (Option 3). e) Single-pass cumulative replacement (Option 4). f) Single-pass activation replacement (Option 5). 200

Figure 7.1: The Rock Engineering System 207

List of Tables

Table 2.1: Common ANNs, their applications and advantages and disadvantages (adapted from Maren *et al.* ,1990). 18

Table 3.1: Summary of perceived benefits of using ANNs 54

Table 4.1 : Summary of (de) normalisation parameters for strain increments. 74
Table 4.2 : Summary of (de) normalisation parameters for stress increments. 74
Table 4.3: Extract of training data file illustrating permutation of input and output quantities in training vectors. 75
Table 4.4: Extract of training data file illustrating permutation of input and output quantities in training vectors after normalisation with (4.26) and (4.27). 76
Table 4.5. Training performance after 10000 iterations for 5 ANNs trained with simulated triaxial compression test data for a linear elastic rock with E=40GPa and ν=0.25 77
Table 4.6 : Performance of the 3-6-3 ANN of Table 4.5 during the training process 80
Table 4.7 ANN performance when presented with simulated triaxial compression test data for rock with varying Young's modulus and Poisson ratio. (Performance of 3-5-3 ANN of Table 4.5 shown in bold type). 84
Table 4.8 : Summary of ANN training and validation results when number and type of simulated tests used to train the ANN are varied. (UX = uniaxial, TX = triaxial, HS = hydrostatic) 88
Table 4.9 : ANN training and validation performance simulating varying quality of machine control in compression testing 89

Table 5.1: Summary of Triaxial test results on Crosslands Hill sandstone. 127
Table 5.2: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing incremental formulation with history and partial state parameters. (NET1) 131
Table 5.3: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing incremental formulation with no history parameters and complete state parameters. (NET2) 131
Table 5.4: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET3) 131
Table 5.5: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET4) 131
Table 5.6: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET5) 131
Table 5.7: Allowable ranges for quantities used in ANN training patterns and their remapped ranges 133
Table 5.8: Samples used for piece-wise linear representation of the triaxial test results on Crosslands Hill sandstone rock for the test conducted at confining pressure 4.5MPa. 137
Table 5.9: Summary of training and validation errors, at different stages in the learning process, for ANNs trained with the triaxial test data 138
Table 5.10: Summary of training and validation errors, at different stages in the learning process, for ANNs trained with piece-wise linear re-sampled triaxial test data. 140

Table 6.1 Recognition performance for fracture character 0 varying contrast and brightness. Values indicate the fracture character label found 184
Table 6.2 Recognition performance for fracture character 4 varying contrast and brightness. Values indicate the fracture character label found 184
Table 6.3 Recognition performance for fracture character 69 varying contrast and brightness. Values indicate the fracture character label found 184
Table 6.4 Recognition performance for fracture character 102 varying contrast and brightness. Values indicate the fracture character label found 185
Table 6.5 Recognition performance for fracture character 255 varying contrast and brightness. Values indicate the fracture character label found 185

Nomenclature

Latin Symbols

$\langle\ \rangle$	average value
A	area
A_f	area of fracture surface exposed
A'	normalised sub-sample area
$A(i,j)$	real-world area of pixel
C	propagation speed
C_p	p-wave velocity
$D_i^{(t)}$	damping force
e_{ij}	strain tensor component
$\dot{e}_{ij}$	strain rate tensor component
$\dot{e}_{vol}$	volumetric strain rate
$\dot{e}_d$	deviatoric strain rate
$\dot{e}_{xx}^{(A)}$	normal strain rate in x direction for triangle A
E_p, E	error for pattern p, total error
E	Young's modulus
f	quantity used in Gauss divergence theorem
F_i	force component acting in i direction
$g_\varepsilon, h_\varepsilon$	gain and offset parameters for strain normalisation mapping
g_σ', h_σ'	gain and offset parameters for stress denormalisation mapping
G	shear modulus
I_j	input value at neuron j
m	strength parameter in Hoek-Brown strength criterion
$m_c(t)$	closest codebook vector at learning step t
$m_a(t)$	activated node associated vector
m_n	grid point mass
n	number of pixels in sub-sample window
n_A	number of pixels in sample area
$\mathbf{n}^{(a,b)}$	outward unit normal vector between nodes a and b
$n_x^{(a,b)}$	x-component of outward unit normal vector between nodes a and b
O_j	output value at neuron j
p_i	i^{th} input value
P	percentage noise level
r	threshold radius around activated node
s	strength parameter in Hoek-Brown strength criterion

$s(t)$	1 if input codebook and input vector labels agree, -1 if not
$\mathbf{S}$	compliance matrix
$\Delta s^{(i,j)}$	length of triangle side between nodes i and j
t	time support variable
t_i	i^{th} target value
t_{pk}	target value for pattern p, at neuron k
$x(t)$	input vector at learning step t
$P_{2,2}(A)$	scale dependent fracture intensity parameter
$\overline{P}_{2,2}$	scale invariant fracture intensity parameter
P_f	probability of fracture presence assigned to a pixel
u_j	velocity in j direction
$u_j^{(a)}$	velocity in j direction of node a
$U_i^{(t)}$	sum of all vertex forces arising from all quadrilaterals around a node
w_{ij}	weight value between neuron i and neuron j
x_i	x coordinate of node i
y_i	y coordinate of node i

Greek Symbols

α	momentum parameter
α	damping parameter
α_c	individual learning rate of closest codebook vector at learning step t
$\alpha(i,t)$	SOM learning rate distance i from activated node at learning step t
δ_k	differental of pattern error with respect to input at neuron k
Δ	increment of
Δx	increment of quantity x
ε_i	principal strain component
ε_{ij}	strain tensor component
$\boldsymbol{\varepsilon}$	pseudo vector of strain tensor components
η	learning rate parameter
θ_j	threshold at neuron j
θ_0	bias at neuron 0
Θ_1	threshold parameter (high level)
Θ_2	threshold parameter (low level)
κ	history parameter
λ_1, λ_2	Lamé's constants
ν	Poisson's ratio
$\boldsymbol{\sigma}$	pseudo vector of stress tensor components
σ_c	uniaxial compressive strength

σ_d	deviatoric stress
σ_i, σ_{ii} $i = 1,2,3$	principal stress components
σ_{ij}	stress tensor component
$\sigma_{ij}^{(rot)}$	stress tensor component contributions due to finite rotations
$\sigma_{ij}^{(cm)}$	stress tensor component contributions due to constitutive behaviour.
σ_0	hydrostatic stress
ψ	dilation angle
ω_{ij}	rotation matrix component

1 Introduction

1.1 Purpose of this thesis

The research described in this thesis investigates the incorporation of parallel distributed processing methods into the disciplines of rock mechanics and rock engineering. In this Chapter these terms are described, central ideas of the research are introduced, and the outline of the thesis is explained. To this end, the remaining portion of this Chapter is composed of four Sections. The first explains what is meant by parallel distributed processing in rock engineering systems and the second introduces rock engineering systems. The third describes the specific purposes and objectives of the research and the fourth explains the structure of the thesis.

1.2 Parallel Distributed Processing

The term *parallel distributed processing* describes a manner in which some quantity, either physical, for example, energy, or abstract, for example, information, is manipulated or transformed. The act of manipulation or transformation of the quantity is termed *processing*. The entity that performs the manipulation or transformation of the quantity refers to the *processor*. The sense of the word *distributed* in this context means that the quantity is not transformed or manipulated within a single processor but is apportioned to an array or ensemble of processors. The sense of the word *parallel* in this context means that the transformations or manipulations of the quantity within the distributed processors are concurrent.

Much of the impetus for general development of parallel distributed processing methods had its source in physiological studies of nervous tissues of higher mammals. It became apparent that the parallel distributed processing paradigm was an appropriate model for information processing in the brains of animals. The computational intelligence community developed models of information processing within brains that employed parallel distributed processing architectures. Subsequent psychological studies of learning in higher mammals inspired computational methods that reproduce capabilities to learn. These latter strategies are known collectively as *machine learning* techniques.

Integration of parallel distributed processing architectures with machine learning strategies resulted in the emergence of a class of methods called artificial neural networks. The machine learning processes within artificial neural networks essentially adjusts the degree of interaction between individual processors within the parallel distributed processing environment on the basis of external observations of some

phenomenon. Through this training or learning mechanism, the artificial neural network becomes a model of the observed phenomenon. Clearly, this is a powerful capability and one that has been exploited extensively within this thesis.

In biology, it is straightforward to recognise living organisms as being composed of many millions of cells, each one associated with a process and its own behavioural repertoire. The overall process (of living) can then be readily identified as being parallel and distributed. In the context of rock behaviour discussed in this thesis, the idea is not that the rock be considered in any similar biological sense, but that the observable behaviour within rock and rock masses can be explained in terms of how individual parts of rock and rock masses, and their associated processes, interact when they are considered distributed, or spatially localised, and their processes occur in parallel.

Through the investigations reported in this thesis, insights into activities and processes central to rock mechanics and rock engineering were gained by the author, and it is hoped, will be gained by the reader. The insights arose through the formulation of improved models for rock mechanics behaviour and through the development of alternative methods comprising the rock characterisation task using parallel distributed processing methods.

1.3 Rock Engineering Systems

> Rock Engineering Systems are a collection of ideas, mathematical tools and computer technology all of which are designed to solve problems in rock engineering with interacting components. The interactions between components can be complex and the rock engineering problems themselves contain a high degree of uncertainty.

Collectively, the ideas, techniques and technology, can be viewed as a discipline orientated methodology. Rock Engineering Systems are a perspective on rock mechanics and rock engineering that applies the concepts of *systems thinking*. This is done by approaching the solution of rock mechanics and rock engineering problems with the premise that rock mechanics and rock engineering are *processes* and by focusing upon the way in which these processes *interact* with each other.

This kind of systems thinking applied to the rock mechanics context is conceptually illustrated in Figure 1.1 for a rock mass system. The rock mass system comprises: i) two objects, the mass of rock under consideration and the environment within which that mass of rock is found (separated by a boundary), and ii) the interactive mechanisms between these (shown crossing the boundary). For the greater part of the time, the rock mass system will be in an equilibrium condition, so it can be argued that a dynamic context is inappropriate. However when any particular rock

mass is being considered, it is normally because it is expected to undergo some kind of change in a comparatively short period of time as a result of engineering perturbations. Thus, a time support is necessary and a dynamic context is assumed. The clearest example of such change is the engineering process, but changes in the natural stress, thermal and hydrogeological field potentials are considered too.

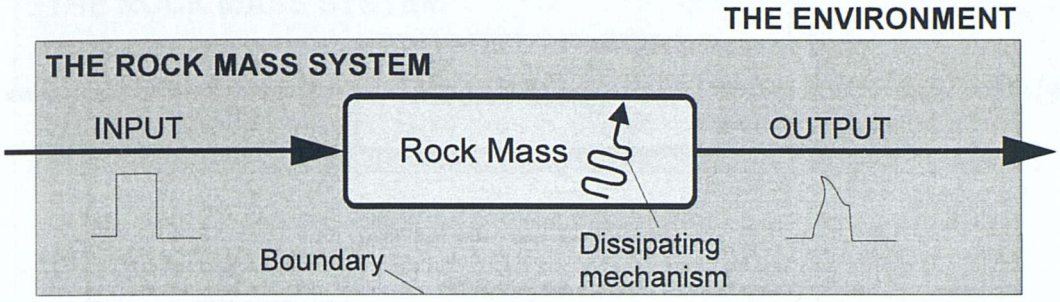


Figure 1.1: Rock mass system concept. Rock mass responds to, or processes, input perturbations producing output behaviour. Conservation rules are implied thus a dissipating mechanism introduced.

Systems thinking applied to rock mechanics focuses on the idea that the mechanical, time dependent, response of the rock mass to perturbations in its environmental field potentials is the process under investigation. Within the diagram (Figure 1.1), the process meaning is illustrated with the use of a labelled box with associated input and output arrows. The input arrow represents a channel from the environment to the rock mass that communicates environmental change. The output arrow represents a channel from the rock mass back to the environment that communicates the way that the rock mass has responded to environmental change.

At the outset of investigations, little of the detail may be known about how the rock 'processes' such change or how the changes are communicated between the objects. Consequently, the rock mass system is shown with grey components. Something that is known with confidence is the location of the rock mass, so its box is outlined with a bold, solid line. Explicit representation of the location of the rock mass is an important concept because it is known that rock mass properties are heterogeneous spatially and potentially subject to effects arising from considering the rock engineering at different scales.

Rock engineering attempts to gain control of the rock mass system, to enforce or induce specified rock mass response while the objective of the engineering process is attained. Conceptually, this can be shown by adding a third object to the system of Figure 1.1, an engineering component. Engineering will introduce change to both the rock mass and its environment. Normally, engineering measures are not introduced without some manner of monitoring or measurement of the natural processes of the

rock mass system. Hence, to accommodate the concept of induced change and need for observations, the engineering object is shown as modifying (hopefully controlling) the rock mass system response via a feedback loop (Figure 1.2).

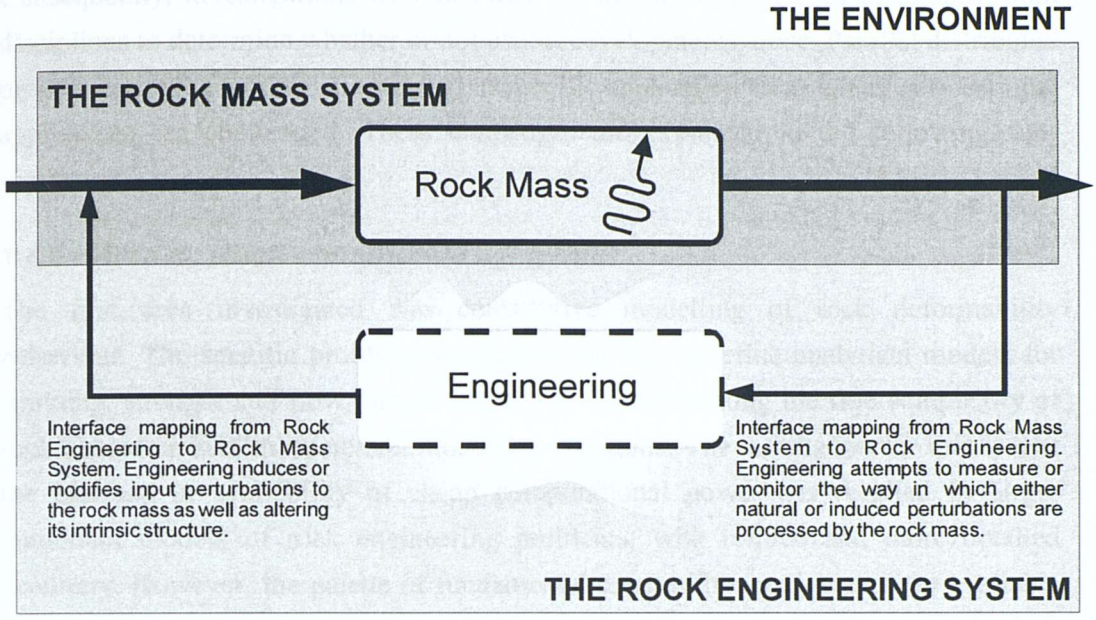


Figure 1.2: Rock engineering system concept

The engineering box is shown as white (or transparent) in Figure 1.2 indicating that the contents of this block are always known. However, the border of the engineering box is shown broken as it is not known whether the contents of the box apply to the rock mass until a full understanding of the rock mass character and behaviour is attained. Taken together, Figures 1.1 and 1.2 are a restatement of the Rock Engineering Systems concept, conceived by Hudson (1992) and discussed by Arnold (1993)

The contents of the engineering box comprise well understood, rock engineering processes, including (but not only):

Rock constitutive behaviour modelling,

Rock mass geometry modelling, and;

Rock mass performance assessment.

Later in this thesis, more detail on the basic layout of the rock engineering systems components found within the engineering box will be set out. At this stage, and in the context of this thesis, it is sufficient simply to identify the areas of rock engineering listed above as those areas where the parallel distributed processing methods have been applied.

1.4 Application Areas for Parallel Distributed Processing

At the outset of the work reported in this thesis, Rock Engineering Systems ideas had resonance with the already well established methods of parallel distributed processing. Consequently, investigations were directed towards hybridisation of ideas from both disciplines to determine whether or not useful developments arose. Parallel distributed processing methods were investigated in specific application areas where conventional approaches are challenged. These challenges are explained in the following sub-sections.

1.4.1 Area A: Rock constitutive behaviour

The first area investigated was constitutive modelling of rock deformability behaviour. The specific problem considered was not to refine analytical models for straining, strength and flow, but to find a way of representing the true complexity of rock behaviour within numerical modelling programs. The motivation for this is that the increase in availability of cheap computational power has resulted in larger numerical models of rock engineering problems, with refined and more detailed geometry. However, the palette of fundamental constitutive models that are available for use with these models has essentially remained the same over the same period. Consequently, the possibility of replacing the 'built-in' rock constitutive models with artificial neural networks was investigated. The criteria that any solution of this problem needs to meet are as follows:

i) The artificial neural network should be able to encapsulate the total rock behaviour and that this should be able to be done in a manner accessible to practitioners.

ii) The artificial neural network should also be able to be embedded in the numerical modelling codes in a straightforward way.

iii) The data source to calibrate the artificial neural network should be compatible with already well established methods.

1.4.2 Area B: Rock mass geometry modelling

The second area investigated was the analysis of discontinuities for rock engineering problems. Computerised methods for analysing rock mass geometry have become sophisticated in the last decade. The state of the art in this area is probably well represented by the computer program called FracMan (http:/www.golder.com). This program is able to assimilate and reconcile data from borehole logging, scanline surveys including observations of discontinuity geometry, and fracture trace maps on exposures, to produce a statistically conditioned rendering of the rock mass geometry called a *discrete fracture model*. It was not considered necessary to investigate the

application of parallel distributed processing methods in this already well developed area of the rock mechanics discipline. However, developments in data gathering techniques to support such programs have not kept pace. Gathering data on rock mass geometry remains a time consuming, possibly dangerous, essentially manual and thus subjective process. Consequently, parallel distributed processing methods were investigated for their potential in automating the discontinuity mapping process. The criteria that any potential solution had to meet were defined as follows:

i) Site presence for data gathering needs to be reduced to one or two hours rather than days.

ii) Personnel involved in the data gathering task needed to work from a safe location

iii) Interpreted result of a quality at least equal to that obtained from conventional means, obtained in a time at least equal to that expended using conventional techniques.

1.5 Structure and Content of This Thesis

In the first application area referred to above, Area A, it is the *process* of understanding and representation of rocks' mechanical behaviour that is the subject of concern. In the second application area, Area B, it is the *process* of measuring and characterising the geometry of fractures in rock masses that is investigated. The subject of this thesis, is to report on investigations into whether any of these can be usefully improved through the adoption of parallel distributed processing methods. The report of these investigations, which takes each application to the proof-of-concept stage or better, is laid out with the structure shown in Figure 1.3 and explained below. The diagram also shows that there is more than one way to read the thesis, depending on reader needs and preferences.

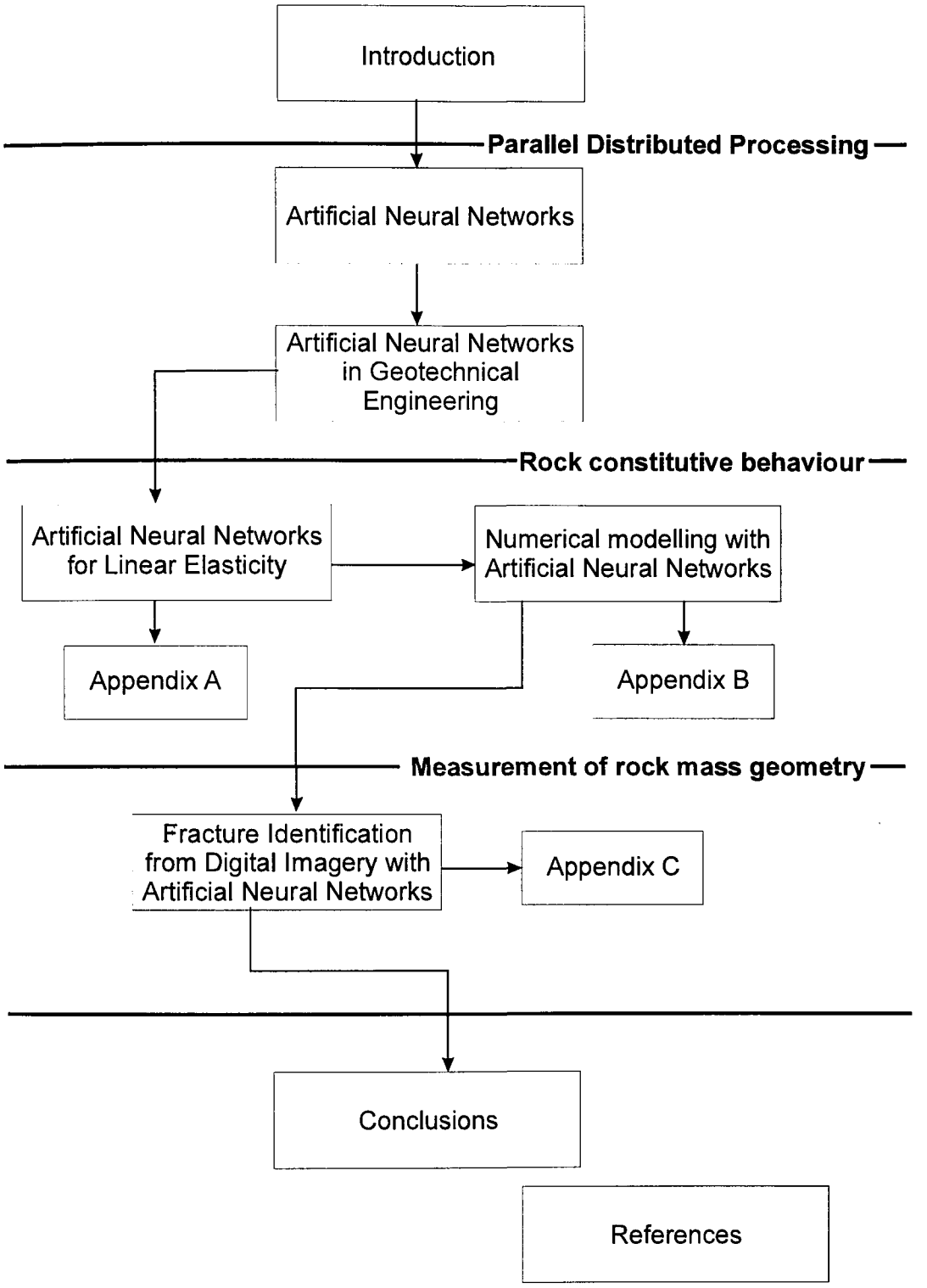


Figure 1.3: Thesis 'road-map'

While it is assumed that the reader is familiar with the main rock mechanics and rock engineering specialist areas referred to above, it is not assumed that the reader is familiar with parallel distributed processing methods that have been used in the work. Consequently, Chapter 2 provides an introduction to one class of parallel distributed

processing methods called *artificial neural networks*. The main categories of artificial neural networks are explained, and a resumé of the characteristics that make them special is provided, together with a review of important application areas in a general engineering context.

Chapter 3 continues the review of artificial neural networks but does so with the specific focus on their precedent use in the field of geotechnical engineering.

Chapter 4 and Chapter 5 should be considered together as relevant to Area A, referred to in Section 1.4.1: Rock constitutive behaviour modelling. In Chapter 4, investigations with isotropic linear elastic rock behaviour are presented to highlight some of the practical issues in implementation of an artificial neural network methodology, trained using data from rock triaxial testing. Chapter 5, is more advanced. Firstly, it demonstrates how some of the target numerical modelling codes work, using *FLAC* as an example. The way that the artificial neural network could form part of the *FLAC* calculation cycle is also explained in detail. Secondly, Chapter 5, provides an example using data gathered from rock triaxial testing to train an artificial neural network, which is then used in a series of *FLAC* numerical models.

Chapter 6 should be considered as relevant to Area B, referred to in Section 1.4.2: Rock mass geometry modelling. In this Chapter the development and performance of two artificial neural network image processing modules is explained. The first, called the *fracture discriminator*, determines the probability of discontinuity presence for every pixel in a digital image of a rock mass. The second, called the *fracture classifier*, systematically examines patches of an image and determines whether or not there is a match for any one of a palette of 255 rock discontinuity 'templates' in each patch.

A brief concluding chapter closes the thesis by listing the original contributions made by the author, bringing the main results of individual chapters together and making more general observations by revisiting the systems thinking mentioned earlier. Appendices containing listings of source code and data pertinent to the material in the main body of the report are presented for completeness.

2 Artificial Neural Networks

2.1 Introduction

As part of the research, artificial neural networks were explored with a view to application and development in rock mechanics and rock engineering. The applications are discussed in detail in subsequent chapters. In this Chapter, firstly the fundamental characteristics common to the various classes of artificial neural networks are reviewed and explained. Towards the end of this Chapter, attention is focussed on the three different types of artificial neural networks that were adopted for use within the work presented in the thesis: the mulilayered perceptron, learning vector quantization with Sammon mapping, and the Self-Organising Map. Readers already familiar with these artificial neural network methods may wish to skip this Chapter and proceed to Chapter 3 directly where applications of artificial neural networks in geotechnical engineering are reviewed.

From this point in the thesis onwards, the term Artificial Neural Network will be abbreviated as ANN. Occasionally, as is the case in the next sentence, the term will be expressed fully as a reminder to the reader.

2.2 Artificial neural network overview

Artificial neural networks (ANNs) are techniques that are used in data modelling in science and engineering. Their main characteristics of operation are that they are fault tolerant, can have excellent interpolative capabilities, and their quality of performance degrades gracefully when subjected to uncertain or imprecise input. As they are modelling techniques, they can be deployed in most situations where a 'crisp' model with a purely analytical basis is already being used in scientific and engineering applications.

ANNs can take the form of digital computer programs, analogue networks of electronic components or hybrid opto-electronic devices. Although analogue devices are faster, due to the widespread use of digital computers the majority of artificial neural networks take the form of computer programs.

In engineering, their characteristics lend themselves well to application areas where measurements are voluminous, noisy or imprecise and reliable, accurate performance of the model is still required. The obvious examples are control applications.

ANNs were originally designed as simulators of biological neural activity. Physiological investigations of brain tissue enabled insights to be made regarding the general topological relations of neuron connections for various types of brain tissue.

Research into electrochemical relations between neuron cell components led to simplified mathematical models for the synaptic interaction between neurons. ANN simulators mimick both structure and synaptic activation.

A simple, yet reasonably accurate, interpretation of artificial neural networks is that they are 'black box' information processing devices that are equipped with input and output channels. Typically, although not necessarily, they have large numbers of input and output channels.

Inside the box, input information is simultaneously relayed to a (typically large) set of densely interconnected artificial neuron processors that convert and transform the input according to predefined rules, before finally passing transformed information to the output channels. As the information processing within the artificial neural network is shared among a set of individual units, it is said to be processed in a *distributed* fashion.

Dense interconnectivity leads to appreciable redundancy in the number of possible routes for information presented to the input channels to reach the output channels. Redundancy in information paths and distributed processing lead to ANNs' robust, gracefully degrading, performance.

ANNs can store information by means of *training* or *learning* techniques. In essence, this consists of adjusting a set of parameters contained within the artificial neural network so that when information is fed into the input channels, desirable or target outputs are obtained. Training generally takes the form of optimisation based on an objective error function between target and actual output. Consequently, artificial neural network training is a process that can be thought of as a complicated form of curve fitting.

2.3 Structure of Artificial Neural Networks

There are many different types of artificial neural networks. One way of rationalising the huge variation in the types is through appreciation of their structure.

The structure of artificial neural networks can be broadly classified into three levels, here listed in order of increasing scale: *microstructure, mesostructure,* and *macrostructure*. In this Section, these three aspects are discussed in turn.

2.3.1 Microstructure

When the term *microstructure* of an artificial neural network is discussed, it is the processing rule of a single artificial neuron that is being referred to. A single neuron has at least one input channel, at least one output channel, (although typically, both input and output channels are numerous) and a processing function. There is considerable flexibility in the choice of processing function. In principle, it can be any

function that can be represented either in compiled computer code or in hard wired electronics.

Consequently, two key factors affect the overall operation of a single artificial neuron. The first factor is the way that the neuron manipulates the multiple inputs into a form suitable for input into the processing function. One way of dealing with multiple input quantities is to find a simple summation. More commonly, a 'connection strength' or weight is associated with each input quantity. Consequently, and most commonly, a weighted sum is calculated and input to the artificial neuron's processing function.

The second factor is the mathematical form of the processing function itself. In the main, the trend is towards simplicity as can be seen from the plots below showing the three most common processing functions that are found (Figure 2.1).

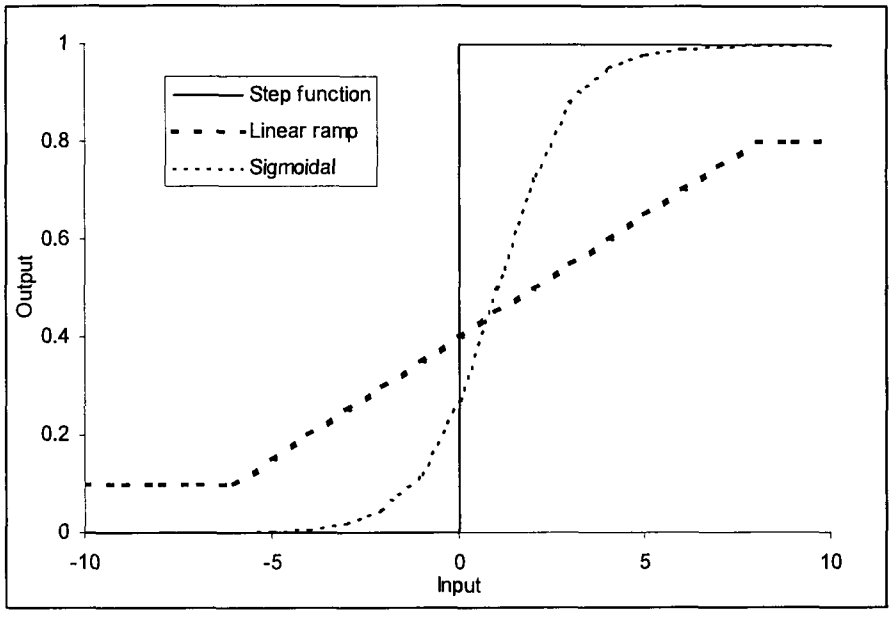


Figure 2.1: Common activation functions for artificial neurons

The simplest is the step function, where if the input is high enough to exceed a given threshold, a non-zero constant output results, otherwise a zero output is maintained. Artificial neurons of this type are also called 'threshold logic units'. The second type of processing function provides for a linear variation of output with increasing (or decreasing) magnitude of input, with the output being hard limited for extremes of input.

As both of these processing functions are not continuous over the possible range of input, they are non-differentiable. Differentiability is a necessary characteristic for certain training procedures that are to be discussed later. The third type of processing function has a sigmoidal form, meaning that it is continuous over all ranges of input. By adjustment of two parameters, the sigmoidal processing function can be made to approximate either of the former processing functions, as is shown in Figure 2.2. This

makes the sigmoidal processing function flexible and hence common in artificial neural network simulators.

Most commonly, the type of processing function found within a single network is homogeneous for reasons of simplicity in training tasks, although in principle, there is no reason why an artificial neural network cannot have hetrogeneous processing function types, providing the training task can still be accomplished.

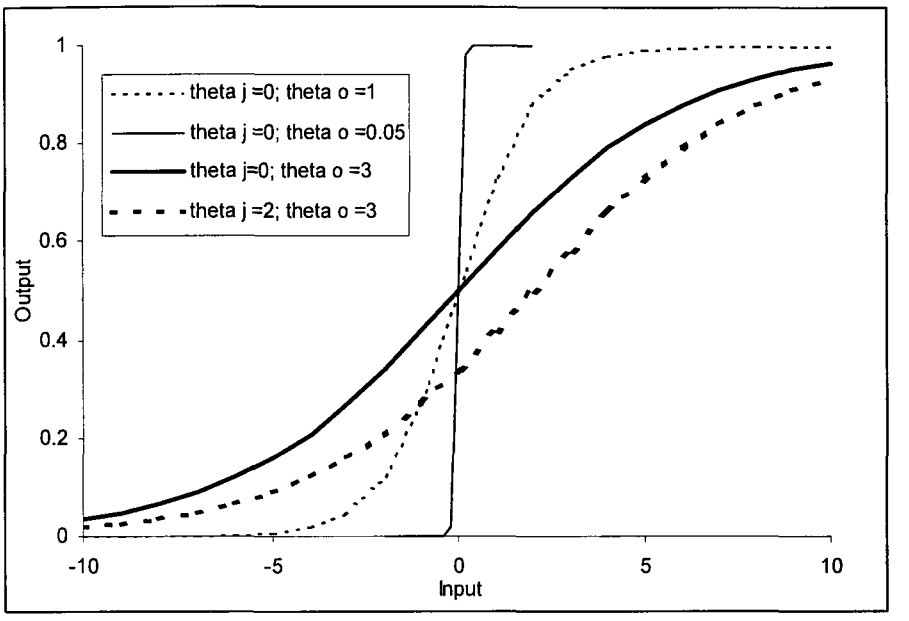


Figure 2.2: Transformation of sigmoidal activation function through manipulation of function parameters. See equation (2.2) in Section 2.6.1 for meaning of 'theta j' and 'theta o'.

2.3.2 Mesostructure

A classic conceptualisation of artificial neural networks is illustrated in Figure 2.3, which illustrates the internal architecture of a type of ANN called a multilayered perceptron. A perceptron ANN is a network of individual artificial neurons normally using only feedforward links. Neurons are often depicted as nodes, arranged in layers, with input nodes and output nodes at the extremities and layers of additional internal neurons between them. Connections are normally specified by means of directed graphs, their sense indicating the direction of information flow through the network. Processing functions (microstructure) are taken to act at the nodes.

The *mesostructure* of an artificial neural network refers to the number, and manner of interconnection of the artificial neurons. There is considerable flexibility in prescription of each of these parameters within ANNs and variations in the specification of these characteristics lead to the huge variety of different neural

networks described in the literature. Often a subtle difference in connectivity can lead to a different terminology when describing respective networks.

Figure 2.4 illustrates the mesostructure of a bi-directional associative memory (BAM) ANN (see for example, Chapter 6 of Pao, 1989). In comparison to the architecture of the multi-layered perceptron ANN, the BAM has no hidden layers and information flow between the two layers of the network can be forward or backward. A set of values representing some ANN input pattern is presented to either of the layers (but not both – the meaning of *xor* in Figure 2.4). A recall algorithm is used to compute values of ANN output at nodes on the 'other' layer. The output values are thus associated with the input values. A feature of this particular kind of ANN is that the process could be reversed. Values originally obtained at the output layer can be re-presented at the same layer. The values obtained at the other layer are those originally defined as inputs. Associations between pairs of 'input' and 'output' values are made with the network's training algorithm.

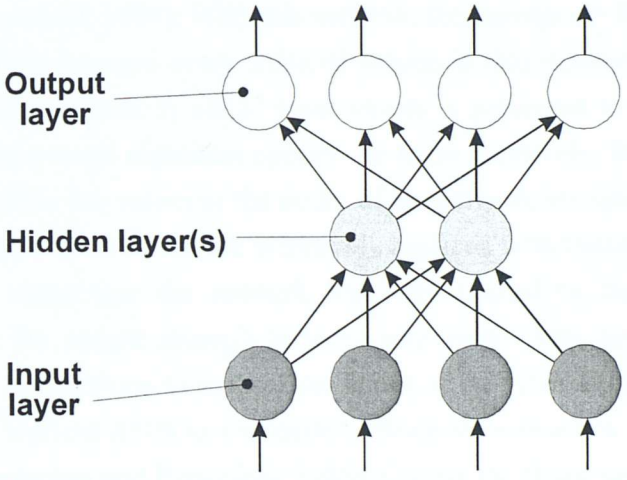


Figure 2.3: Multi-layered perceptron mesostructure

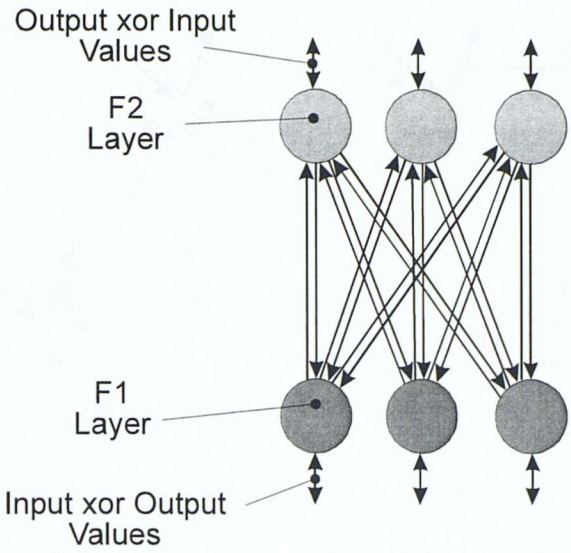


Figure 2.4: Diagram of the mesostructure of a Bi-directional Associative Memory ANN.

Figure 2.5 illustrates the mesostructure of a Hopfield ANN (Hopfield, 1982, in Anderson and Rosenfeld, 1988). With this network, the concept of 'layers' of artificial neurons breaks down because every artificial neuron in this network is connected to every other artificial neuron. A set of input values is presented to the nodes of the Hopfield ANN and a recall algorithm operates on these iteratively. With each iteration of the recall algorithm the values at the nodes of the network are updated. After many iterations, the values at the nodes are effectively replaced with those of one of the so-called attracting states that the network has been trained to store. The training algorithm adjusts the weight strength between neurons to store the attracting states. Adding recurrent connections to each of the nodes of the ANN (Figure 2.6), changes its type from an Hopfield ANN to a so-called, 'Brain-state-in-a-box' ANN (Anderson *et al.*, 1977, in Anderson and Rosenfeld, 1988). Clearly the Brain-state-in-a-box ANN will have similar dynamics to the Hopfield ANN, however input quantities are reinforced during each iteration of the recall algorithm.

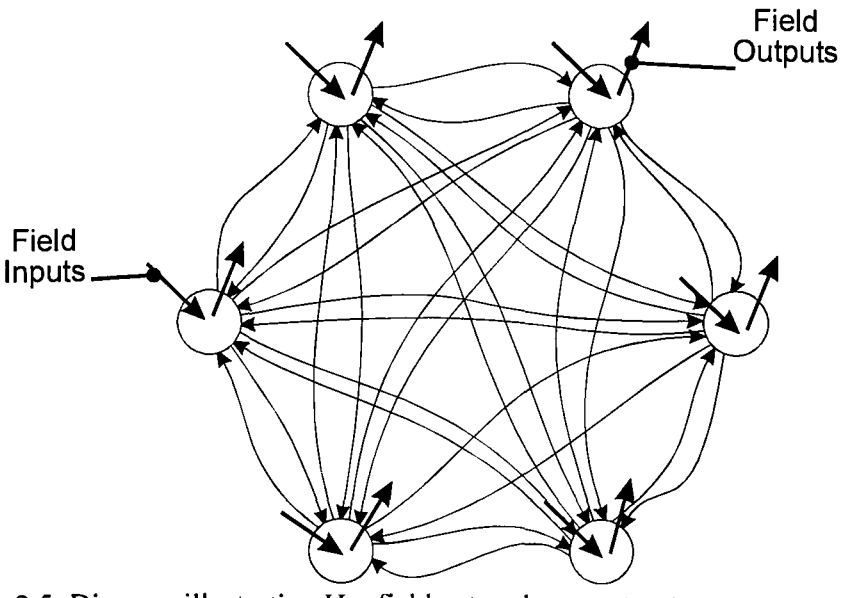


Figure 2.5: Diagram illustrating Hopfield network mesostructure.

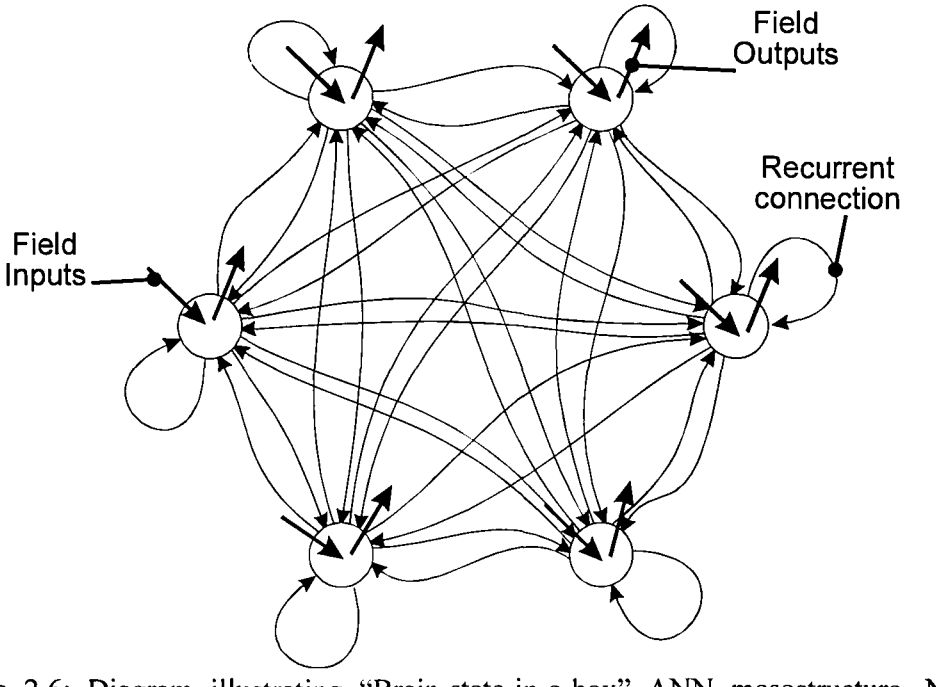


Figure 2.6: Diagram illustrating "Brain-state-in-a-box" ANN mesostructure. Note recurrent connections at the network nodes.

2.3.3 Macrostructure

When more than one artificial neural network is used within a modelling scheme, the connections indicating information flow are indicative of the *macrostructure* of an overall artificial neural network. Figure 2.7 shows the relation between ANN microstructure, mesostructure and macrostructure. The network types, defined by mesostructural characteristics, can be homogeneous (same mesostructures) or

heterogeneous (different mesostructures). As has been indicated in the previous Section, some classes of mesoscopic artificial neural networks are better suited to some tasks than others. It is rational in design for a specific modelling task to attempt to develop hybrid artificial neural networks where specialist 'skills' are not available from a single mesoscopic structure.

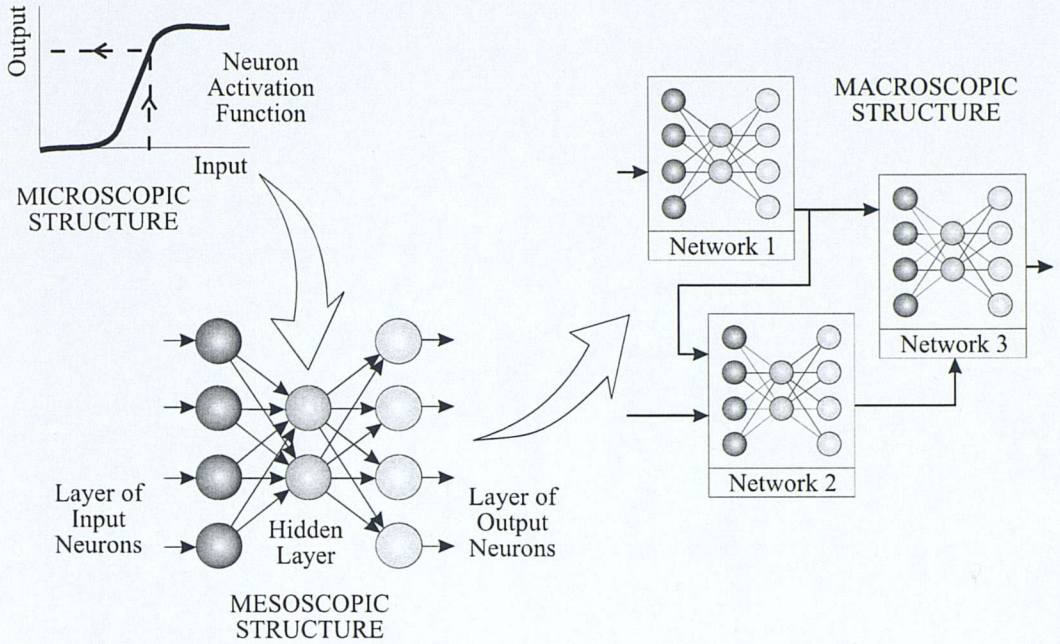
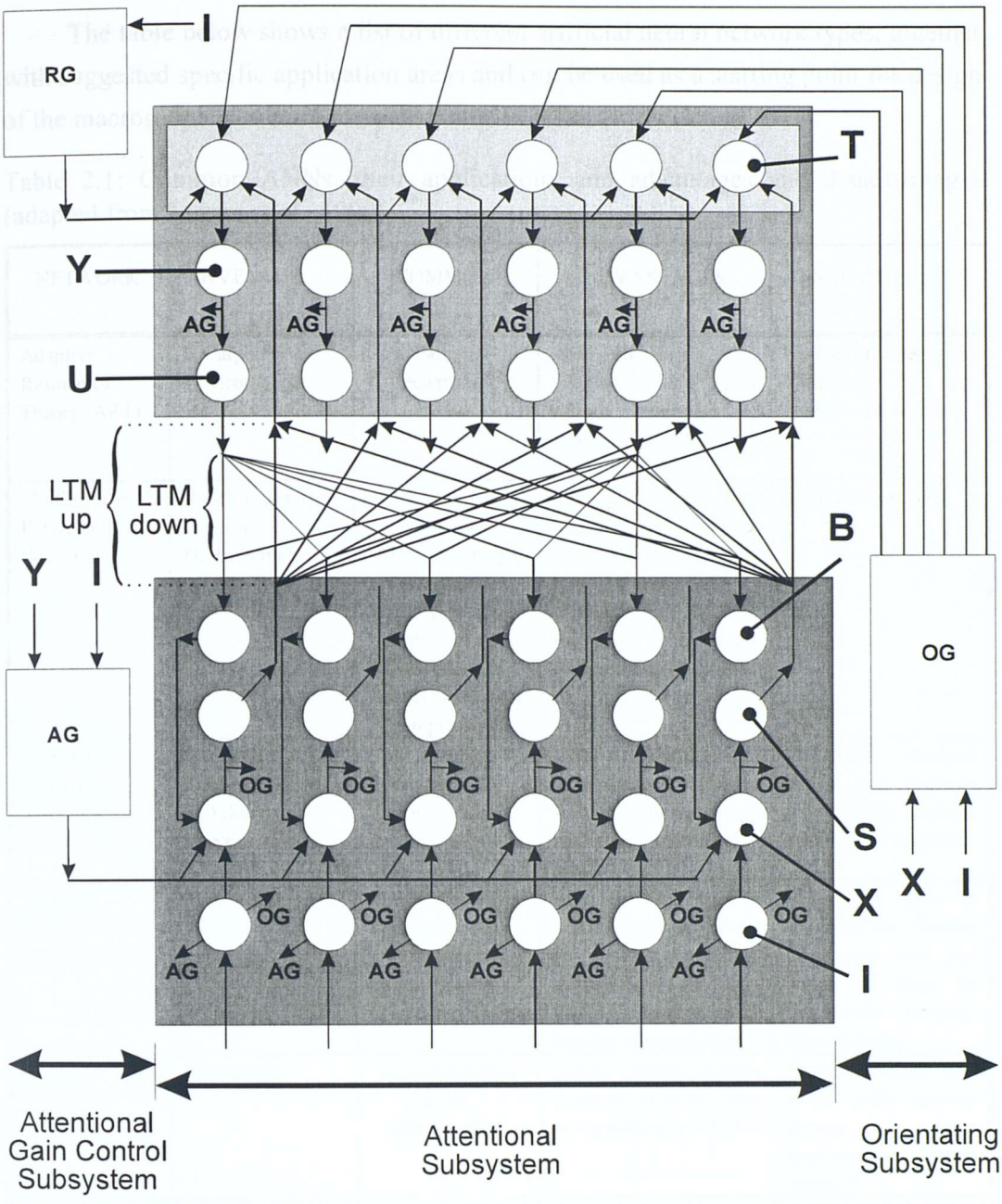


Figure 2.7: Diagram showing the relation between ANN microstructure, mesostructure and macrostructure.

Good examples of such a strategy are the artificial neural networks exploiting the so-called Adaptive Resonance Theory (Carpenter & Grossberg, 1987). A diagram of an ART network is presented in Figure 2.8. As will be clearly appreciated from the diagram, ART networks are complicated ANNs. Their use is in the discovery of clusters (like patterns) within data. The lower 'slab' of neurons, accepts raw input values which undergo a weighted sum to yield output values. The output values are then passed as input to the top 'slab', where one neuron becomes activated in a competitive learning layer based on a distance metric between the input pattern and a series of exemplar class defining vectors. The result of this, bottom up phase of processing, is reinforced by a top down stage, where the various gain parameters that condition the bottom up signal are adjusted. The lower slab thus behaves rather like a feed-forward multi-layered perceptron architecture; the top slab behaves like a self organising map. The adaptive tuning scheme means that the same ANN can be used with different data sets.



I	Input storage buffer	LTM up	Long term memory up
X	F1 pattern (STM)	OG	Orientating gain
S	F1 output to F2	RG	Recognition gain
B	Input from F2 to F1	AG	Attentive gain
T	Initial F2 pattern	STM	Short term memory
Y	Resulting F2 enhanced pattern		F1 Layer
U	F2 output to F1		F2 Layer
LTM down	Long term memory down		Lateral inhibition competitive network

Figure 2.8: Conceptual diagram of an ART 2 network developed by Carpenter & Grossberg (1987).

The table below shows a list of different artificial neural network types, together with suggested specific application areas and can be used as a starting point for design of the macroscopic structure for such complex tasks.

Table 2.1: Common ANNs, their applications and advantages and disadvantages (adapted from Maren *et al.* ,1990).

NETWORK	INVENTOR/ DEVELOPER	COMMON APPLICATION	ADVANTAGES	DISADVANTAGES
Adaptive Resonance Theory (ART)	G.Carpenter & S. Grossberg	Pattern recognition	Able to learn new patterns, form new pattern categories & retain learned categories.	Nature of categorical exemplars may change with learning.
Multilayered, Perceptrons (MLP)	P. J. Werbos, D.Parker, D. Rumelhart	Pattern recognition, Signal filtering, Noise removal, Signal/image segmentation, Classification, Association, Data compression	Fast operation. Good at forming internal representations of features in input data or classification. Well studied with many successful applications.	Long learning time.
Boltzmann/ Cauchy Machine	G.Hinton, T.Sejonwski, D.Ackley; H. Szu	Pattern recognition, Optimisation	Able to form optimal representation of pattern features. Follows energy surface optimisation minima.	Boltzman machine has a long learning time. Cauchy machine offers faster learning.
Learning Vector Quantization	T. Kohonen	Auto-associative recall, (pattern completion given partial pattern), Data compression	Able to self organise vector representations of probability distributions in data. Rapid execution after training is completed.	Unresolved issues in selecting the number of vectors used and length of time for appropriate training. Slow training.
Hopfield	J. Hopfield	Auto-associative recall, Optimisation	Simple concept, proven dynamic stability, easy to implement in VLSI.	Unable to learn new states, poor memory storage, many spurious states returned.
Self-Organising Topology Preserving Map	T. Kohonen	Complex mapping (involving neighbourhood relations), Data compression, Optimisation	Able to self organise vector representations of data with a meaning ordering among the representations.	Unresolved issues in selecting the number of vectors used and length of time for appropriate training. Slow training.

2.4 Using Artificial Neural Networks

2.4.1 Suitability

The main issue concerning the practical application of artificial neural networks in modelling for rock engineering is whether the problem being considered is inherently

suitable for an artificial neural network treatment. Resolution of this issue centres around gaining a good understanding of both the nature of the problem being tackled and the generic methodologies adopted with artificial neural networks. Indicators of suitability of an artificial neural network approach are as follows:

- The availability of (large) data sets.
- Detailed system models not being explicitly known.
- The complexity of interrelations between variables being such that establishment of a valid analytical model is unlikely.
- The data acquisition methods employed have yielded data sets of low reliability, for a variety of reasons.
- The variables defining the rock mechanics / rock engineering system do not change (although their values may change).

A pre-requisite is that the variables considered should be known to be related in some way.

2.4.2 Data required

As with any modelling method, the first consideration relates to the quantity, quality, and nature of the data that is to be used to produce the model. There is a common misconception that the quantity of data needed to produce working artificial neural network models must be large. This is not necessarily the case.

If some natural phenomenon is to be characterised as a model, then in general a large number of measurements have to be made in order to obtain a large enough sample of the variables characterising the phenomenon. However, if the set of values for the variables is finite and limited in number, then the data set can be formed from an exhaustive enumeration of all possible combinations. The resulting artificial neural network model then acts as an interpolator between these states, or is designed to deal with noisy input.

An example of this idea is an artificial neural network that is used in character recognition. If the character typeface and language used are known in advance, then it may be necessary only to train the network with 26 alphabetic, 10 numeric and (say 15) punctuation characters (for the English language). The total data set would consist of 51 suitably encoded entries. Alternatively, if the artificial neural network is to be used to recognise handwritten characters, then handwritten samples of characters from as many different people as possible would be required, producing a data set with cardinality many times greater than 51.

In both situations, but especially in the former, a training data set can be effectively expanded by the addition of a controlled amount of random 'noise'.

2.4.3 Specification of Input and Output Quantities

Input quantities are assumed to be the independent variables while output quantities are taken to be the dependent variables for the phenomenon being considered. Having specified the character of each of the variables, a database containing a large number of examples of input and corresponding output values is established and pre-processing and training proceeds. Inverse relations can be obtained through reassignment of input and output characters and developing a second artificial neural network. Alternatively, a network capable of specially dealing with inverse relations in data can be used, such as the bi-directional associative memory ANN.

2.4.4 Pre-processing

Pre-processing methods are often applied to these input and output variables prior to manipulation with the network algorithm. In many cases the pre-processing step involves a transformation of the raw data into some convenient representation that facilitates the ability of the network to learn the training data.

In the simplest cases, the pre-processing stage can be the normalisation of the training data set between a convenient range of values, for example, +1 to -1. In complex cases, the pre-processing step can include transformations or functions which represent what is known about the system characteristic. For example, if the data are temporal in nature, they may first be transformed into the frequency domain through use of a Fourier Transformation, or if the data have a spatial character, a cosine transform may be appropriate.

In many cases, pre-processing is done in order to map the input data into an interval in which the artificial neuron processing functions are most sensitive.

Certain types of artificial neural networks, called 'functional-link' neural networks pass raw input values to the artificial neural network and also the same values are processed through some additional problem specific mathematical functions. This can have the advantage of reducing the number of neurons that need to be included in the network to represent the data. Reducing neurons normally makes the training process faster.

2.4.5 Information flow

When the artificial neural network has been trained, it is used according to the following steps:

- A set of input values are presented at the input channels.
- The input channels implement any pre-processing required.
- The resulting output is passed to the first set of internal artificial neurons.

- Each of the artificial neurons processes its input according to its processing rule, thus producing an output.
- Additional phases of internal processing are performed by sets of internal neurons. This cascade of processing continues until information arrives at the output channels.
- The output channels post-process their incoming information.

The steps listed above represent a stylised description of the information flow throughout a general artificial neural network that can become a great deal more complex depending on network mesostructure and macrostructure.

In cases where there is total interconnectivity between processing neurons including recurrent links, the values presented at the input channels act as an initial condition for network processing. This initial condition leads to certain neurons being 'turned on' and others being 'turned off'. As the neurons are completely interconnected, this new state necessitates a second cycle of processing resulting in further changes in the activation states. Processing continues in cycles until the activation states are steady. At this stage, the network can be said to be in its *attracting state* and the output is obtained by sampling the state of the neurons in the network. The most widespread example of this type of artificial neural network operation is the Hopfield network, Hopfield (1982).

A finite number of attracting states can be contained in such artificial neural networks, which are called associative memories. The maximum number of attracting states that can be stored depends on the number of artificial neurons in the network. Training these networks consists of 'setting' their attracting states. Having been trained, if partial or noisy input is presented at the input channels as an initial condition, these networks will cycle to the 'closest' attracting state 'implanted' within the network during training. An energy function can be associated with each of the attracting states. By analogy with the minimisation of free energy in chemical systems, these states are associated with local minima within a vector space characterised by the activation ranges of the individual neurons and the number of neurons. Consequently, Hopfield networks have been applied in optimisation decision making problems (Hopfield and Tank, 1985), for example the Travelling Salesman Problem (TSP).

In artificial neural networks of this type that have semi-autonomic operation, the term *coherent behaviour* has been used to describe convergence to particular attracting states. This means that the engineering of the microscopic and mesoscopic rules of the neural network has led to co-ordination between individual processing units, observed at the scale of the whole network. The fact that this behaviour is not explicitly coded within the rules has also led to the use of the term *emergent property*

to describe this kind of phenomena. It is also one reason why these networks have generated great excitement in researchers in the artificial neural networks field. The dynamics of Hopfield type artificial neural networks, which have (non-direct) recurrent links and exhibit emergent properties, have been shown to be isomorphic to the dynamics of real paramagnetic materials (Hopfield, 1982). It is, perhaps, somewhat surprising to find that a biologically inspired parallel distributed information processing paradigm has contributed most to the study of the physics of paramagnetic materials, rather than to the understanding of thought processes in animals. Such issues are explored in detail by Muller and Reinhardt (1990) and Neelakanta and De Groff (1994).

2.4.6 Post-processing

Post processing consists of mapping the output of the artificial neural network into some real world domain so that the output can be used in the normal modelling sense.

2.5 Training Artificial Neural Networks

Training artificial neural networks involves the goal-directed establishment of a parameter set. Goal-directed means that an objective, or goal, to be achieved is defined and the search for suitable parameters is measured, and guided by, the closeness of the current state of the process to achieving the goal. For many network types, this parameter set is the set of connection weighting factors. The weighting factors apply to the input values along the connections between the artificial neurons.

Depending on its combined weighted input, an artificial neuron may become activated producing an activation value that is relayed to other neurons as (part of) their (weighted) input. A cascade of processing ensues that ultimately produces a predicted estimate of measured (target) outputs corresponding to the values of the input parameters. If the predicted estimates differ appreciably from the measured values, the weighting factors are adjusted in some systematic way until this error is minimised.

When the error at output is below some specified tolerance for all sets of input values in the training data set, the training process can be said to have been achieved satisfactorily.

Systematic adjustment of weighting factors is done according to predefined training algorithms, which often choose to adjust the weighting factors in proportion to the magnitude of error at output.

2.5.1 The Hard Learning Problem

McClulloch and Pitts (1943) developed the first mathematical model of a biological neuron. It has the form of a threshold logic unit with weighted input as discussed in

Section 2.3.1. The first model for training of biological neurons was developed by Hebb (1949). These breakthroughs lead ultimately to the earliest perceptron type artificial neural networks that were simply composed of two sets of artificial neurons, those composed of the input channels and those composed of the output neurons (Rosenblatt, 1959, 1962). Training algorithms could be relatively simple because all weights within these networks directly connected inputs to outputs. These networks aroused considerable interest in the early 1960's and could be trained to achieve useful and interesting tasks, but were generally disappointing in the range of problems they could tackle.

In 1969 in their book, "Perceptrons: An Introduction to Computational Geometry" Minsky and Papert (1969) showed that artificial neural networks of this type were unable to model exclusive-or situations. Exclusive-or situations mainly arise in logic systems, comprising multiple inputs and a single output. If one, and only one, of the inputs takes the value TRUE, the output value should also be TRUE. All other input values should produce FALSE. This issue was termed the *hard learning* problem. Multilayered networks containing 'hidden' artificial neurons (that do not constitute any part of input or output layers) were known to overcome the problem (Nielsen, 1965) but at that time an algorithm that could train these multilayered perceptrons had not been invented.

The basic problem was how to adjust the weights of connection between hidden neurons and input neurons. In 1986, Rumelhart, Hinton and Williams developed a training algorithm that could adjust 'internal weights' providing the processing functions of artificial neurons were differentiable. This rule was called the *generalised delta rule with back propagation of error*. The introduction of this training algorithm in connection with the introduction of hidden artificial neurons is largely responsible for the explosion of research and development in artificial neural networks since the mid 1980s. The generalised delta rule for multilayered perceptrons is discussed in detail in Section 2.6 of this Chapter. Next, as an introduction, the general characteristics of training algorithms are discussed.

2.5.2 Training Algorithms

Training algorithms vary in the manner in which a systematic variation of the network weight parameter set is conducted, such that the output estimates are as close as possible to the measured or target values. The currently most widely adopted artificial neural network training algorithm is the generalised delta rule with back-propagation of error (normally foreshortened to 'back-prop'), which adjusts the weight parameter set such that the sum-squared error between observed and estimated output values over all training data examples is minimised.

This is a gradient descent optimisation methodology which suffers from entrapment within local minima, although improvements to this training method to overcome the problem are many and varied. This includes the use of a technique called simulated annealing, where the early stages of training are subject to a large component of random variation of the weight parameter set which is gradually reduced as learning progresses.

When the target outputs of an artificial neural network are known and a training algorithm is used to force the output predicitions to match the output targets, the training method is termed *supervised.* However an alternative family of cluster discovery based methods exists and accomplishes *unsupervised learning* through comparison of similarity measures calculated from the training data examples. The Self-Organising Map (Kohonen, 1977, 1982, 1987, 1990) is an example of a method categorised as an ANN that uses this method of training.

Some sophisticated artificial neural networks are capable of *dynamic* learning, that is learning while information processing is underway. For example if an artificial neural network is used in a pattern recognition scheme, classification behaviour can be modified 'on-the-fly', and new classes introduced - should unrecognisable patterns be presented. In these cases, special mechanisms have to be introduced within the artificial neural network paradigm to allow these features. An example of such a network is Carpenter and Grossberg's ART1 & ART2 Networks (Carpenter and Grossberg, 1987; see also Pao, 1989).

An important characteristic of training algorithms is that they permit artificial neural networks to create their own organisation or representation of the information received during learning. These representations (referred to as *internal representations*) are created in an automatic fashion within the network, as a consequence of the actions governed by the learning algorithm. In most instances, the internal representation of learned information is of no direct significance to the problem at hand and that is why artificial neural networks have deservedly gained their 'black box' label.

Due to effective inaccessibility, superficially at least, internal representation can be considered a serious disadvantage of these methods. However, in many cases this disadvantage is heavily outweighed by important advantages: e.g. (i) that for extremely complex phenomena at least some kind of working model can be produced; and (ii) internal representations tend to be extremely compact.

Although training algorithms are being improved continually, and new strategies, for example genetic algorithms, are being deployed, consideration of the engineering tolerances typically acceptable to rock mechanics engineers, suggests that even rudimentary training algorithms can produce high calibre classifiers, system

models and pattern associators for use in rock engineering. Consequently, the simplest, the generalised delta rule with back propagation of error, will be discussed in detail.

2.5.3 Heuristic Aspects of Training Artificial Neural Networks

There are significant heuristic aspects associated with the use of artificial neural networks. These concern the detailed specification of micro- and meso-structures of the network for the paradigm used, as well as specification of values for parameters that control the training algorithms. Whilst these heuristics are a drawback of artificial neural networks, they do not prohibit the production of useful implementations. The trend is toward development of 'self-tuning' algorithms where such operational parameters adapt themselves during the course of learning. One example is that of dynamic node creation (Azimisadjadi *et al.*, 1993) where 'new' artificial neurons are added to the network as training progresses if their addition results in reduction of mean square error. A survey of similar dynamic mesostructural learning methods is provided by Kwok and Yeung (1997).

In practice, it is often necessary to investigate the effect of modifications to the artificial neural network meso-structure when attempting to optimise the training and generalisation characteristics, although methods are available that prune the meso-structure during training. In some cases, the performance of the resulting model is improved through selection of a different kind of network. It can be seen from Table 2.1, that a different ANNs can be used to implement similar core processes. The trend in rock engineering applications has been to adopt perceptron architectures due to their relative simplicity.

Within unsupervised learning algorithms, for example k-means (Kohonen *et al.* 1992a, 1992b) or fuzzy k-means (Harrison, 1995), a major heuristic element exists. These cluster discovery methods have the problem of selecting an appropriate value of k, the number of clusters. One way to overcome the problem is to choose a range of values for k, conducting unsupervised learning for each value, and selecting the best. Xu *et al.*, (1993) have attempted to overcome such problems by developing an algorithm that automatically chooses a value for an appropriate number of clusters. Pao, 1989 also presents a code for cluster discovery using a k-means approach with added nodes.

Artificial neural networks can operate in parallel with some other conventional mathematical model with an analytical basis, the latter having the role of producing estimates of target outputs, for shared input parameters or patterns. Training artificial neural networks in this way is called *distil supervised learning* and has the result that the artificial neural network can act as a direct replacement for the conventional

model. Such an approach can be applicable where determination of output using the conventional mathematical model is time consuming, for example, when it requires several matrix inversions.

2.5.4 *Termination of Training and Generalisation Issues*

The training process is assumed complete when the training error falls below some predetermined tolerance related to the nature of the application for which the network is being trained. The network weight parameter values converge and the trained network is ready for validation. Validation data sets have exactly the same structure as the training data sets; however validation data are never used in training.

Validation data sets are presented to the network at the end of each training phase and the *generalisation* capability of the network is evaluated. Essentially this refers to measuring the quality of the predictive capabilities of trained networks using data outside that used to train the network itself. For example, if the training data are representative of some mathematical (or other) function, the generalisation process is analogous to testing the interpolative or extrapolative capabilities of the network in estimating the function. Artificial neural networks are subject to the same dangers as traditional interpolation or extrapolation techniques when used outside of the range of the sampled data, but in the experience of this author, have been found to be much more robust.

2.6 Multilayer Perceptron Using the Generalised Delta Rule and Back-Propagation of Error

Thus far, the general characteristics of artificial neural network development and operation have been discussed. Now, details specific to the three artificial neural networks selected for use within these studies are presented. The first of these is described fully as being a multilayered, semilinear feedforward network trained with the generalised delta rule with backpropagation of error. The mesostructure of this network is as shown in Figure 2.3, which shows the artificial neurons being arranged in layers with information flow travelling forwards from input to output in normal operation.

Information on how this ANN works can be recovered from any mainstream textbook on the subject. The author's understanding of the formulation is presented in the next four Sub-Sections following that set out by Pao, (1989) and Aleksander and Morton (1990). Readers who know about ANNs can skip straight ahead to Section 2.7.

2.6.1 Feedforward operation

For the purposes of discussion, a three layered network is discussed. The input channels are taken to be in layer *i*; output artificial neurons are in layer *k*; and there is a single 'hidden' layer, labelled *j*. The input to a node in layer *j* , denoted I_j, is given by:

$$I_j = \sum w_{ji} O_i \tag{2.1}$$

where w_{ji} is the weight of the connection between the output of an artificial neuron in layer *i*, to the artificial neuron in layer *j* , and O_i is the output of the artificial neuron in layer *i*. The output of artificial neuron *j* is given by the processing function for this network:

$$O_j = \frac{1}{1+\exp-\left(\frac{I_j+\theta_j}{\theta_0}\right)} \tag{2.2}$$

where θ_j is a biasing parameter that shifts the output response curve along the I_j axis, and θ_0 is a shaping parameter. For the neurons in layer *k*, the inputs are:

$$I_k = \sum w_{kj} O_j . \tag{2.3}$$

The outputs of the output layer *k* are:

$$O_k = \frac{1}{1+\exp-\left(\frac{I_k+\theta_k}{\theta_0}\right)} . \tag{2.4}$$

As the mesostructure of the network essentially dictates that the values at output represent repeated superimposition of the various inputs, the network operation can be viewed as *linear*. However, the microstructural artificial neuron functions behave non-linearly with the consequence that the network as a whole is termed *semilinear*. Equations (2.2) to (2.4) essentially describe the recall operation of the multilayered perceptron ANN.

If instead of a simple weighted sum of inputs, a weighted sum of inputs raised to progressively higher order is made, the resulting network is termed a *higher order* neural network, and is *nonlinear*. Such networks are thought to be advantageous over simpler methods because, in general, it is found (Pao, 1989) that increasing the degree of non-linearity within a network, increases its capabilities to represent the phenomenon or data of interest. The use of higher order terms in the input connections to the activation functions of artificial neurons can lead to transformation invariant pattern recognition. Such a capability would be extremely useful in artificial neural

network applications, such as the handwritten character recognition problem mentioned earlier.

An important adaptation in operation of the basic scheme described above can be employed when the number of output neurons is large: conceptually they can be arranged as a 'slab'. Consequently, it becomes reasonable to assert that when a single neuron within the output slab becomes activated, through lateral connections, other neurons adjacent to this neuron will become activated too, but to a lesser extent. Such considerations are the basis of radial basis function artificial neural networks (see, for example, Tsoi, (1989) and Bottou and Vapnik, (1992)).

2.6.2 Training using backpropagation of error

In training the semilinear network, a *pattern*, $\mathbf{x}_p = \{I_{ip}\}$ of input values is presented to the input channels, in layer i, and the problem is to find a set of weights so that the output set $\{O_k\}$ matches a set of targets $\mathbf{t}_p = \{t_{pk}\}$, each within an acceptable tolerance of error. Furthermore, when remaining patterns are presented to the input channels of the network, the same set of weights should result in agreement between outputs and targets within specified tolerances.

For a given pattern $\mathbf{x}_p$, the error between targets and outputs is given as:

$$E_p = \frac{1}{2}\sum_k \left(t_{pk} - O_{pk}\right)^2 \tag{2.5}$$

where the factor one half is included for mathematical convenience. Over all P patterns, the average error is

$$E = \frac{1}{2P}\sum_p \sum_k \left(t_{pk} - O_{pk}\right)^2. \tag{2.6}$$

Using the former expression for a single pattern, adjustments must be made to the weights of the network to minimise this error. Considering the weights between layer j and k, the adjustments are given:

$$\begin{aligned}
\Delta w_{kj} &= -\eta \frac{\partial E_p}{\partial w_{kj}} \\
&= -\eta \frac{\partial E_p}{\partial I_k} \cdot \frac{\partial I_k}{\partial w_{kj}} \\
&= -\eta \frac{\partial E_p}{\partial I_k} \cdot \frac{\partial}{\partial w_{kj}} \sum w_{kj} O_j \\
&= -\eta O_j \cdot \frac{\partial E_p}{\partial I_k} \\
&= \eta O_j \delta_k
\end{aligned} \tag{2.7}$$

The last step defines

$$\delta_k = -\frac{\partial E_p}{\partial I_k} \tag{2.8}$$

which using the chain rule can be further expanded

$$\begin{aligned}\delta_k &= -\frac{\partial E_p}{\partial O_k}\cdot\frac{\partial O_k}{\partial I_k} \\ &= (t_k - O_k)\cdot\frac{\partial O_k}{\partial I_k}\end{aligned} \tag{2.9}$$

Subsuming the shaping parameter into the weights and thresholds, the activation function can be written

$$O_k = \frac{1}{1+\exp-\left(\sum_j w_{kj}O_j + \theta_k\right)} = \frac{1}{1+\exp-(I_k+\theta_k)} \tag{2.10}$$

from which its partial derivative can be written

$$\frac{\partial O_k}{\partial I_k} = O_k(1-O_k) \tag{2.11}$$

Finally,

$$\delta_k = (t_k - O_k)O_k(1-O_k) \tag{2.12}$$

and the weight adjustments for the final layer are

$$\Delta w_{kj} = \eta O_j(t_k - O_k)O_k(1-O_k). \tag{2.13}$$

For the weights found between layer i and j we start in a similar fashion as before.

$$\begin{aligned}\Delta w_{kj} &= -\eta\frac{\partial E_p}{\partial w_{ji}} \\ &= -\eta\frac{\partial E_p}{\partial I_j}\cdot\frac{\partial I_j}{\partial w_{ji}} \\ &= -\eta\frac{\partial E_p}{\partial I_j}\cdot\frac{\partial}{\partial w_{ji}}\sum w_{ji}O_i \\ &= -\eta O_i\cdot\frac{\partial E_p}{\partial I_j} \\ &= \eta O_i\delta_j\end{aligned} \tag{2.14}$$

Between the hidden and input layers the situation is more complicated in comparison to between the output and hidden layers, because changes in weights in the former do not affect the network outputs directly. We progress by expanding δ_j.

$$\begin{aligned}
\delta_j &= -\frac{\partial E_p}{\partial O_j}\cdot\frac{\partial O_j}{\partial I_j} \\
&= -O_j(1-O_j)\frac{\partial E_p}{\partial O_j} \\
&= -O_j(1-O_j)\sum_k \frac{\partial E_p}{\partial I_k}\cdot\frac{\partial I_k}{\partial O_j} \\
&= O_j(1-O_j)\sum_k\left(-\frac{\partial E_p}{\partial I_k}\right)\cdot\frac{\partial}{\partial O_j}\sum_j w_{kj}O_j \\
&= O_j(1-O_j)\sum_k\left(-\frac{\partial E_p}{\partial I_k}\right)w_{kj} \\
&= O_j(1-O_j)\sum_k \delta_k w_{kj}
\end{aligned} \tag{2.15}$$

and,

$$\Delta w_{ji} = \eta O_i O_j(1-O_j)\sum_k \delta_k w_{kj} \tag{2.16}$$

The last equation shows that δ_j is a function of δ_k as adjustments to weights between layer j and layer i, are found from the adjustments made to weights between layer k and layer j. This is the essence of the *backpropagation of error* method used to train this artificial neural network as the error between targets and outputs is propagated backwards through the network to hidden layers via the generalised δ terms. The δs are termed *generalised* because the form of the weight adjustment equation becomes general for hidden and output layers.

The parameter η is a *learning rate* parameter with values less than unity. The overall scheme can be seen to be similar to a gradient descent optimisation of the ANN weights. Too large a value of η yields fast learning but also can lead to weight oscillations. Consequently, a *momentum* term, α is introduced by writing the weight adjustment equation for the $(n+1)$th presentation of the input patterns

$$\Delta w_{ji}(n+1) = \eta\delta_j O_j + \alpha\Delta w_{ji}(n) \tag{2.17}$$

through which fluctuations in the rate of weight change are moderated but can prolong the training process.

2.6.3 Practical training issues

When training the multilayered perceptron in practice:

- The thresholds, $\theta_{j,k}$ are treated as weights with input unity and thus are found simulataneously in the back propagation of error algorithm.
- Weight and threshold values are initialised randomly prior to training.
- In a single training cycle, the weight adjustments are calculated for each of the input patterns presented to the network. The (algebraic) sum of the weight adjustments is applied.
- The learning rate is typically set to a value of 0.1, but can be much smaller.
- The momentum rate is typically set to a value of 0.9, but can be larger.
- The system error and maximum output error for a given pattern is recorded for each training cycle so that convergence of the algorithm can be monitored.

It is good practice to repeat the overall training exercise with different random seeds for the random number generator. The redundant interconnectivity of these networks means that the solution for the set of network weights is underconstrained. Thus, training an artificial neural network once simply recovers one possible set of weights from a family of sets of 'solution' weights. This characteristic of artificial neural networks essentially precludes any specific interpretation being made about a particular weight or subset of the weights as completely different 'solution' values can be found when the network is trained starting with a different random seed.

2.6.4 Mesostructural complexity in pattern classification

Multilayered feedforward artificial neural networks can be trained to work as pattern classifiers. The input pattern vector is presented to the input channels and the output channels then yield values that relate to the class or category of the input pattern.

Any single neuron in the network has an output that is a function of a simple linear weighted sum of its inputs. As the input changes, the neuron output can be envisaged as tracing a line, plane or hyperplane, for input pattern cardinalties of two, three and greater than three respectively. Continuing generally, a hyperplane can thus be thought to delineate a partition of the pattern vector space. Consequently, a group of hyperplane-defining artificial neurons together form a hypervolume partition which represents the membership requirements of a particular class. Each class specified as a hypervolume existing in a vector space of dimension N requires roughly one artificial neuron for each of its hyperfaces, that is roughly $2N$ neurons. If a single class is represented by two distinct hypervolumes, then an artificial neuron is required in the next layer of the network in order to conduct an exclusive-or operation to join both distinct regions.

It is this reasoning, illustrated in Figure 2.9 (taken from Lipmann, 1987), that suggests hidden layers are a prerequisite for perceptron type ANNs involved in arbitrary classification tasks. This line of reasoning also leads to guidelines for the number and arrangement of neurons within networks designed for a particular task.

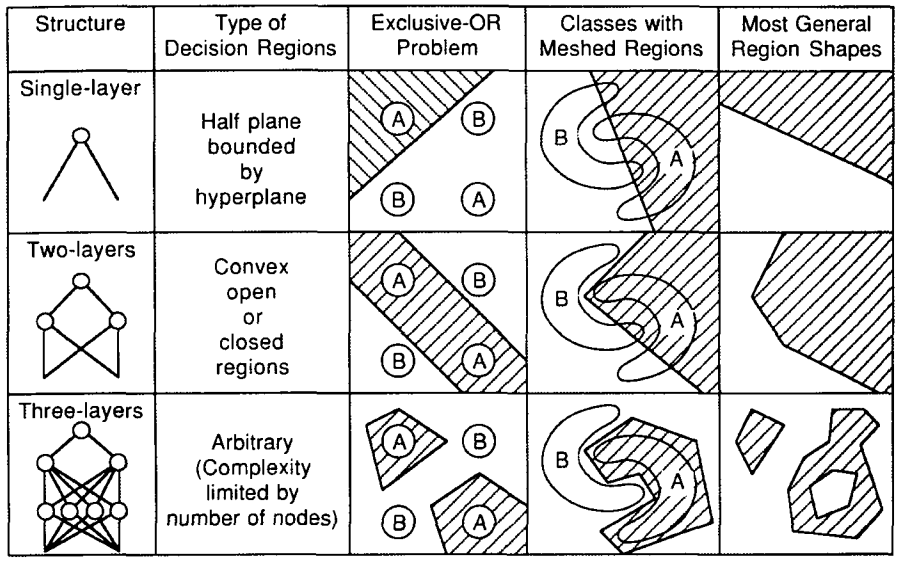


Figure 2.9: The role of hidden artificial neurons in multilayered perceptrons for two inputs (from Lippman, 1987).

From a practical viewpoint, increasing the number of artificial neurons in the network decreases overall generalisation capability but increases robustness and fault tolerance. Decreasing the number of artificial neurons speeds the training task. Increasing the degree of non-linearity in the network, for example by using higher order nets, results in the tracing out of hypersurfaces rather than hyperplanes. This means that hypervolumes can be specified more precisely in the training task and ultimately leads to improved classification. Thus, although increasing the number of artificial neurons in a network is the simplest manner through which classification accuracy can be improved, increasing the non-linearity tends to produce classification improvement that is obtained more efficiently as the overall network size is maintained at low levels.

These complexity issues are of prime concern when the output vector space is to be discretely partitioned for classification purposes. They are of lesser concern when the output vector space is known to be continuous. However, for certain phenomena where the output vector space can be envisioned to be complex (perhaps, for example, overfolded) they still remain an issue.

2.7 Learning Vector Quantization

2.7.1 Topological correspondence

One interpretation of the action of the artificial neural network presented in the previous Section is that it produces a mapping that transforms an N dimensional, normally Euclidean, input vector space into a vector space of different dimensionality that is non-Euclidean. Weight adjustments are caused by differences at output rather than differences between each of the input patterns. The latter differences are easily visualised as 'distances' between the input values. The question of whether there is topological correspondence between the input and output vector spaces, *i.e.* there is some relation between distances between input patterns and the corresponding distances between output patterns, is one that is usefully explored by means of learning vector quantization artificial neural networks (Kohonen *et al.* 1992b, 1995b).

A second general interpretation of the artificial neural network presented in the previous Section is that it makes a search to find a set of parameters that are in some manner representative of a phenomenon or a data set of interest. In the mulilayered feedforward artificial neural network, the parameters that are found (the weights) exist in an essentially abstract space.

The learning vector quantization (LVQ) method is not really a network as such, but a set of parameters that are optimised such that they represent a phenomenon or whole data set of interest. As learning vector quantization methodologies have the same purpose as multilayered feedforward artificial neural networks and use similar methods in optimisation and training respectively, the former have been classified as specialised ANNs, despite the fact that they have no explicit network mesostructure. In contrast to multilayered feedforward artificial neural networks, the parameter set that is optimised in learning vector quantization methods can have a physically meaningful interpretation: sets of co-ordinates in an Euclidean vector space.

In the following, the learning vector quantisation methodology is explained with reference to a specific sub-class: *optimised learning vector quantization* (OLVQ).

2.7.2 LVQ General description

The OLVQ method itself is a technique to produce a small number of trained vectors, each labelled with a class. These vectors are called *codebook vectors.* Learning or training processes consist of ensuring that the codebook vectors are representative of all possible vectors within a data set, each labelled with a class. This latter data set is called the training data set. The form of records within this data set is directly analogous to the input and output patterns discussed for the multilayered feedforward

artificial neural network, with the exception that the output numeric vector is replaced by a pattern label (although this pattern label itself can be numeric).

Figure 2.10 illustrates the LVQ concept which aims to approximate a complete partition of a vector space of a dimension matching that of the data with a finite set of code book vectors (the discrete points in the diagram). The positions of the code vectors may also approximate any probability density function that applies within the vector space. The continuous boundaries shown on the diagram and the grey scale indicating probability density function are ill defined at the outset of learning. Their approximation improves during the training process by 'tuning' the positions of the code book vectors to the larger set of labelled data vectors. In recall operations, new, unlabelled data vectors would become labelled by identifying their closest code book vector.

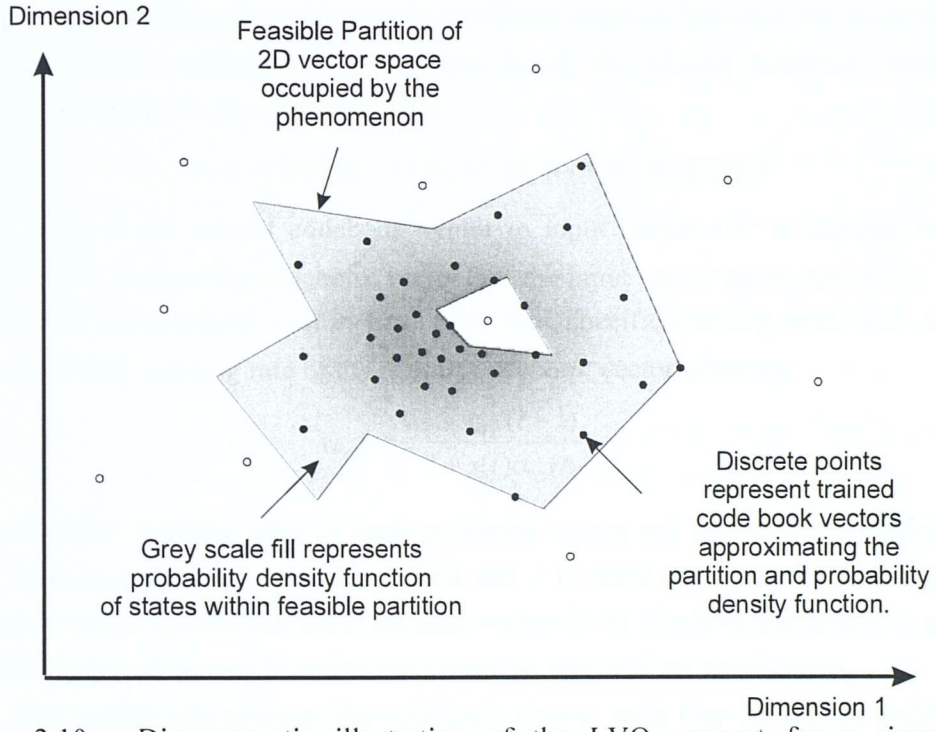


Figure 2.10 : Diagrammatic illustration of the LVQ concept for a simple 2 dimensional case. Solid circles = 'IN'; Open circles = 'OUT'

In use, the learning vector quantisation method aims to classify (new) input vectors that have played no part in the training process. Classifying a new unlabelled vector is accomplished by finding the 'nearest' trained codebook vector and returning its class label. The metric used to determine the nearest codebook vector is normally the Euclidean distance between the data vector and the codebook vector.

The OLVQ method is one of a family of Learning Vector Quantization methods that are special in the sense that the trained codebook vectors are optimised to reflect the boundaries between classes.

2.7.3 LVQ Initialisation

Once a training data set of labelled vectors has been set up, the first step in implementing the OLVQ methodology is to decide how many codebook vectors are to be produced and to initialise their values. Initialisation is achieved by extracting a subset of labelled vectors from the training data set such that these codebook vectors are either evenly distributed between the classes or in proportion to the frequency distribution of the classes. The *initial* values of the codebook vector components are then identical to those of the vectors that were extracted from the training data set.

2.7.4 LVQ Training

Training consists of comparing a vector within the training data set with the initialised codebook vectors and finding the closest. The closest codebook vector is found through determination of the minimum Euclidean distance between the input vector and each of the codebook vectors. When found, the closest codebook vector is modified according to the following rule:

$$m_c(t+1) = [(1 - s(t)\alpha_c(t)]m_c(t) + s(t)\alpha_c(t)x(t) \tag{2.18}$$

where $m_c(t)$ is the nearest codebook vector to input vector $x(t)$ at learning step t, $s(t) = 1$ if the label of the codebook vector and the input vector agree, and $s(t) = -1$ if not. Each of the codebook vectors have their own, specific learning rates, $\alpha(t)$. $\alpha_c(t)$ is the individual learning rate of the nearest codebook vector given by:

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)}. \tag{2.19}$$

The individual learning rates of each codebook vector are all initially assigned the same value, typically 0.3. Equations 2.18 and 2.19 have the effect of 'moving' the codebook vector toward the input training vector if the labels disagree and to move the codebook vector away from the input training vector if the labels agree.

This update rule arranges the codebook vectors such that the trained codebook vectors are representative of the complete distribution of the vector space. Codebook vectors that have already been trained with OLVQ can be fine tuned with the LVQ version 3. According to Kohonen *et al.* (1992b), the fine tuning adjustment in LVQ3 results in the boundaries between classes being better defined.

After these rules have been applied, the process is repeated with the next input vector from the training data set. Training continues until the classification accuracy of the trained codebook data set reaches an asymptotic value. Kohonen *et al.* (1992b) recommend, as a rule of thumb, that 40 n cycles should be performed where n is the number of codebook vectors. The trained codebook data set can then be considered as a *classifier*.

2.7.5 LVQ Validation

Classification testing consists of finding the class labels of the closest (as determined through the Euclidean distance) codebook vectors within a classifier for a set of labelled test input vectors that are independent of the training data set. This is normally described as a *k-nearest neighbour* classification, (see, for example, Pao, (1989) for explanation and source code). Classification testing is thus a specific form of validation and generalisation assessment discussed previously. Classification accuracy is expressed as the percentage of the number of test input vectors for which the labels of the closest codebook vectors agree.

The overall purpose of the OLVQ method is to effect a data compression in the sense that some large set of vectors is adequately (say 80-90% generalisation) represented by a much smaller set of vectors. The cardinality of the latter will also depend on the graininess of the input vector class labels: if there are many classes, more optimised vectors are required than otherwise.

The method also has the effect of distributing the codebook vectors within the input vector space in a manner that is in some way efficient with respect to the boundaries between classes. This distribution is interesting in its own right but difficult to visualise as the dimensions of typical input vectors are greater than the maximum three (or at most four) dimensions that can be physically rendered on paper.

2.8 Sammon Mappings

Accordingly, LVQ methods are often used in conjunction with an auxillary algorithm developed by Sammon Jr (1969), that can project the positions of trained codebook vectors of high dimension to vectors of dimension two or three which can be plotted for visualisation.

The Sammon mapping is not classified as an artificial neural network, although many elements of the Sammon mapping algorithm share common themes with them. The Sammon mapping algorithm starts by randomly initialising the locations of a set of two-dimensional points, where the number of points in the set is the same as the number of vectors in the higher dimensional space. The algorithm proceeds by computing the Euclidean distance between each vector in the higher dimensional data set and every other vector in the same set (a kind of outer product on distance). The distances between the corresponding two-dimensional points are then also computed and their positions adjusted such that the two dimensional-distances match the higher dimensional Euclidean distances. The position adjustments are made using a gradient descent iterative method that minimises the sum of the squared errors between the 2-D distances and n-D distances.

Perfect, that is zero error, representation of distances in high-dimensional vector spaces can only be achieved in that vector space itself. There is thus an inevitable residual error associated with the Sammon Mapping procedure that reflects the fact that *n*-D distances can never be perfectly preserved in 2-D space. This error is sometimes referred to as the Sammon stress.

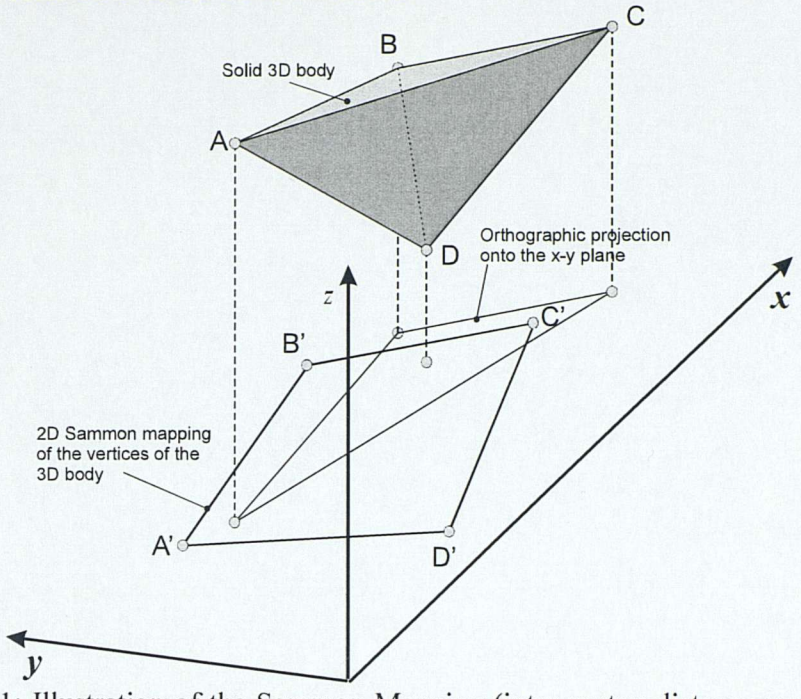


Figure 2.11: Illustration of the Sammon Mapping (inter-vertex distance preservation) for a 3D polyhedron in comparison to orthographic projection (2 co-ordinates preserved) onto the 2D plane.

Figure 2.11 demonstrates the concept of the Sammon Mapping. It is not a simple orthographic projection. Rather the distances between points A,B,C and D are preserved in the Sammon Mapping such that they are as close as possible to the distances between A',B',C' and D'. Figure 2.12 shows a simple example of the Sammon Mapping with real data. 301 vectors of a 3D helix were subjected to the algorithm for 600 iterations. The resulting 2D vectors were then imported into the AutoCAD package and plotted. The 3D vectors were also imported into the same AutoCAD drawing and plotted. The 3D vectors were displaced in the z-direction so that the two traces would not overlap in an isometric view.

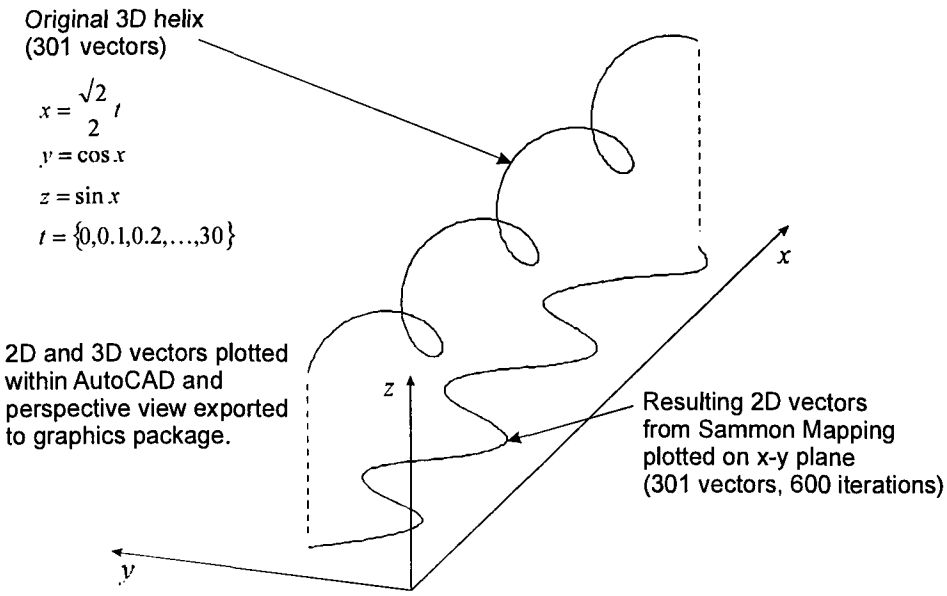


Figure 2.12 : Example Sammon Mapping of a 3D helix, projected onto the 2D plane.

The Sammon mapping procedure is computationally intensive. In studies conducted as part of the work reported herein, the Sammon mapping visualisation procedure normally took longer than the (O)LVQ process. Various measures have been proposed to speed up the algorithm including initialisation of 2D locations with principal component values of the data (Lerner *et al.*, 2000). The Sammon mapping also suffers from the problem that, if only one of the *n*-D vectors in the visualisation data set changes, or one point is added or removed, then the Sammon mapping has to be recomputed entirely.

2.9 Self-Organising Maps

Integration of the ideas of optimisation of representative vectors in LVQ with the visualisation technique used in Sammon mapping leads to a commonly utilised artificial neural network method called the Self-Organising Topological Map (SOM) which was developed by Kohonen (1982, 1990).

Self organising maps associate a vector with each of a set of nodes on a regular lattice. The lattice can be rectangular or hexagonal. The dimension of the node associated vector is the same as the dimension of the training data the SOM is to learn. A SOM is said to have completed training when individual nodes on the lattice are activated when presented with new data. The basis for 'activation' is a distance metric, normally the Euclidean distance, between the node associated vector and the new data vector (identical to the *k* nearest neighbour procedure mentioned earlier). Through the activation process, different nodes on the lattice are sensitive to new data vectors of specific character.

2.9.1 SOM Training

There are various details concerning the way in which the node associated vector components are adjusted during the training process but the general theme is the same. When a training data vector is presented to the SOM, a node is activated for adjustment when its associated vector is closest to the training vector (measured by the Euclidean distance). Suppose this is called the primary activated node associated vector. Nodes on the lattice that are adjacent to the activated node within a specified radius, r (of nodes) are additionally activated, denoting that their associated vectors should be adjusted too. Suppose these are called secondary activated node associated vectors. The adjustment to the activated associated vectors is specified as follows:

$$m_a(t+1) = m_a(t) + \alpha(i,t)[x(t) - m_a(t)] \tag{2.20}$$

where $m_a(t)$ denotes an activated node associated vector at some discrete time learning timestep, t, $x(t)$ is the training data vector, and $\alpha(i,t)$ is a so-called, learning rate. $\alpha(i,t)$ varies in time from an initial value prescribed at t=0, typically, 0.05, to zero at the time that the training process is completed. $\alpha(i,t)$ also varies with distance, i between the primary activated node associated vector and the secondary node associated vectors. The nature of this variation was confined to being a step function, that is:

$$\alpha(i,t) = \begin{cases} \alpha(t); & i \le r \\ 0; & i > r \end{cases}. \tag{2.21}$$

The effect of the training rule (2.20) and (2.21) is that data vectors that are topologically close in their own vector spaces tend to activate nodes that are close together on the SOM lattice. The SOM method aims to achieve preservation of topological order in the mapping between the input vector space and the two-dimensional lattice of SOM nodes. The overall result is a data cluster discovery method where, in a pathological case, one node on the SOM corresponds to a single cluster of data. More normally, a less distinct ordering is produced with the result that areas of nodes on the SOM correspond to clusters in the high dimensional vector space.

2.9.2 SOM Initialisation and Validation

The initial values of the components of the node associated vectors can be assigned using random numbers. The memory that the trained node associated vectors have of their initial values becomes exponentially small, as is indicated by the training rule (2.20). However, there are computational advantages in assigning initial values, that are in some way close, to the trained values. For this reason, node associated vectors are assigned using random selection from the training data set.

Validation of a SOM cannot be done in the same way as for either MLP back-propagation type ANNs or LVQ ANNs because the training procedure is unsupervised. However, some measure of how well the SOM represents the data on which it has been trained can be found by evaluating the sum of squared Euclidean distances between node associated vectors and the data vector that caused their activation. If this error is small, it can be said that the vector quantisation error of the SOM is low and its performance in 'recall' tasks is likely to be good. It should be noted that such a measure does not give any indication of the topological ordering quality of the SOM, despite the fact that it can be empirically observed that it does preserve order in a data set.

2.9.3 SOM Calibration

A subsequent stage in the SOM process is that of 'calibration'. Calibration is akin to a labelling process that can only be done if further information is available on what the values of the trained node associated vector components actually mean. As the SOM training procedure captures topological ordering in a *n*-D data set, and the SOM itself is a two dimensional entity, if the SOM nodes can be labelled then it allows *n*-D cluster visualisation. If this can be done, then an alternative to LVQ and Sammon mapping visualisation is available.

In practical uses indicated by Kohonen *et al* (1992a, 1995a), the classification process is exemplified by exposing a SOM to new data for which the input data vectors have some meaning. If the new data represents, for example, malfunction of a machine or system, then by observing which of the SOM lattice nodes become activated when exposed to this data, these nodes can be labelled as representing malfunctioning states. This is not much use without confirming, with additional information, that the same nodes do not become active when processing data which represents 'normal' operation.

2.10 Conclusions – Types of ANN and their potential for rock mechanics modelling.

In this Chapter artificial neural networks have been described in general terms. Their essential characteristics are that they should possess microstructure, mesostructure and training and recall rules, although there are exceptions, such as the LVQ methods which do not have explicit mesostructure. Three artificial neural networks have been discussed in more detail. These are the multilayered perceptron, learning vector quantization, and the self-organising topological map.

Issues of primary concern in development of an application of an ANN are its type, the synthesis or collection of appropriate data to train the network, and

consideration of whether typical information to be processed by the network is likely to change in character during processing operation.

Throughout the Chapter, several different interpretations of what artificial neural networks have been mentioned. These are summarised in the following list.

- Programmable information processing devices
- Complex mappings between vector spaces
- Empirically based modelling devices
- Cluster discovery methods
- Discriminants used in pattern classification / classifiers
- Data compression devices
- Associative memories
- Devices that create internal representations of data or phenomena
- Analogues of physical processes
- Expert systems

Through a series of examples, the research reported in later chapters investigates the extent to which these features can be exploited in various rock mechanics and rock engineering tasks.

It can be said that for simple rock mechanics or rock engineering problems there is no advantage in adopting an ANN solution, or the use of ANNs could be a disadvantage in the process of arriving at an optimal engineering solution. Use of ANNs may be advantageous for difficult problems, for example when a preliminary model for the relations between rock mechanics data are unknown, or complicated. In such cases, as are widespread within rock mechanics and rock engineering, the use of an artificial neural network may considerably simplify model formulation and implementation. These issues are to be explored in more detail in the next Chapter.

Partial destruction of an artificial neural network leads to a partial degradation in performance, although some network capabilities may be retained even with major network damage. Two major classes of fault tolerance exist. The first concerns the progressive degradation of network performance when it becomes 'damaged'. The second concerns their ability to perform robustly in the presence of noisy, corrupt, distorted or incomplete input information. In rock engineering it is typical that information of this type is obtained, and thus again artificial neural networks may have a role to play in this context.

ANNs are modular systems. This means that they can be readily integrated into existing or new processes and systems. If processing speed is of the essence, ANNs can be encoded in silicon chips to produce high speed capability.

3 Review of Applications of Artificial Neural Networks in Geotechnical Engineering.

Having explained the structure and operation of artificial neural networks (ANNs) in the previous Chapter, in this Chapter attention is directed towards how ANNs have already been used within geotechnical engineering. The review of geotechnical engineering applications then starts by classifying them by application. Subsequently the types of neural network models that have been adopted in their development are highlighted. Finally, the review closes by examining the perceived benefits of using ANNs.

3.1 ANNs in general engineering practice

Real world problems such as the mixing of fluids, turbulent flow, hydraulic systems, economic forecasting, human behaviour, image and speech recognition and failure of mechanical structures are examples where ANNs can produce effective solutions where other methods fail due to the inherent complexity of the core phenomena.

The following list gives the primary application areas of neural networks in which they have been found to be of great utility. The list is taken from Maren *et al.* (1990).

- Robust pattern detection (spatial, temporal and spatio-temporal)
- Signal filtering
- Data segmentation
- Data compression and sensor data fusion
- Database mining and associative search
- Adaptive control
- Optimisation, scheduling and routing
- Complex mapping and modelling of complex phenomena

Of these applications, probably the two most significant areas to date have been pattern recognition and adaptive control. This list is by no means exhaustive, and macrostructural networks, each with modular functionality, provide extended capabilities, for example, automatic process control.

A starting point in the assessment of whether ANN methods are applicable to a particular rock mechanics or rock engineering problem is to determine whether the task falls into one of the above categories. On initial appraisal, the problem being considered may *not* obviously fall into one of these categories. However where process choice is constrained to the palette of core processes listed above, the problem being considered can often be seen to be composed of a combination of such tasks.

3.2 Survey of applications in geotechnical engineering

In 1991, Zhang *et al.* provided one of the first publications detailing the potential of ANNs in a variety of roles specifically within rock mechanics and rock engineering. At the same time, Ghaboussi *et al.* (1991) first highlighted the capabilities of ANNs in modelling constitutive behaviour of general engineering materials, and later detailed the application of these methods to the field of geomechanics (Ghaboussi *et al.*, 1994). Review articles of the use of ANNs in mining engineering are provided by Hadjigeorgiou and Flament (1993), and Clarici *et al.* (1994).

3.2.1 Constitutive Modelling

The earliest work on constitutive modelling using ANNs dealt with concrete and reinforced concrete (Ghaboussi *et al.*, 1991) (Wu and Ghaboussi, 1992). Ghaboussi *et al.* (1994) subsequently extended his work to include geomechanical materials. Having drawn inspiration form Ghaboussi's earlier work with concrete, Millar and Clarici, (1994) reported on the use of ANNs to model the constitutive behaviour of intact rock specimens, directly from laboratory triaxial test data. Nie and Zhang, (1994) reported a similar application later that year. Calderbank, (1994) conducted extensive numerical simulations using the FLAC (Itasca, 1993, 1995) program, outlining how the ANN models could be integrated within the finite difference geomechanical code. Completion of what was 'work in progress' at the time, reported by Millar and Calderbank (1995) and Millar *et al.* (1995) is to be the subject of Chapter 5.

A specific aspect of rock mass characterisation is the appraisal of roughness of fractures found in rock masses, as this has a significant effect on the shear strength of these surfaces. Work on automating these currently subjective assessments was reported by Hadigeorgiou and Flament, (1993), and was extended later by Hadjigeorgiou *et al.*, (1995) to include their general constitutive behaviour.

Within the soil mechanics discipline, Ellis *et al.* (1995) were the first authors to report on the use of ANN constitutive models of various sands from triaxial test data in an implementation that also took account of the particle size distribution and varying stress histories. Dyminsky *et al.* (1996) used a similar method to model residual soil behaviour. Soft clays were the problem material for Logar and Turk, (1997) who used oedometer test curves to train an ANN. Using similar test data, Amorosi *et al.* (1996) show how an ANN can also be trained to model the constitutive behaviour of clay, taking its overconsolidation ratio into account. They also summarised several approaches to geomaterial deformability modelling and demonstrated that ANNs have emerged as comprehensive models for a wide variety of geomaterial constitutive behaviours, including that of discontinuities.

For the purposes of pavement design, Williams and Gucunski, (1995) developed an ANN that could automatically interpret measurements of dynamic surface waves to yield estimates of dynamic elastic modulii *in situ* and a neural network that can learn soil compaction behaviour has been reported by Basheer and Najjar (1995). Attohokine and Fekpe, (1996), took data from California Bearing Ratio tests and index tests on laterite soils and related these to the field strength of lateritic soils which are difficult to deal with due to their field scale heterogeneity.

3.2.2 Constitutive behaviour and numerical methods

Simply to show that ANNs can be trained to yield reliable estimators of constitutive behaviour is trivial, if the resulting networks cannot be readily integrated into engineering analyses. It is unlikely that ANN constitutive models would be used within elastic analyses; Hooke's law is compact and can be optimised for rapid execution in computer code. Where a rock mass is known to behave in a much more complex fashion, for example in the near field around an underground mine opening, ANNs could be used effectively as they can model the complex behaviour. For example, Millar and Calderbank (1995) showed how ANN models could be integrated into the finite difference code FLAC (Itasca, 1993) to model a strain softening mine pillar behaviour. Logar and Turk's (1997) networks, mentioned in the previous Section, were coupled to a finite element code and used to model the behaviour of an embankment with soft sub-soil. Mesh decomposition or partitioning is important in certain instances with numerical methods. Topping *et al.*, (1997) use an ANN to achieve this task efficiently when parallelising their finite element code to run on a distributed computing architecture.

3.2.3 Rock Characterisation

Although the characterisation of the constitutive behaviour of rock is probably most important for general rock engineering design, several additional sources have been published concerning the determination of more general rock properties with ANNs. In 1993, Bonifazi reported on the use of ANNs for mineral particle characterisation. Yi and Linqvist (1995) and Yang and Kim (1996) both predicted rock quality parameters by means of ANNs. For reservoir characterisation purposes, Wong *et al.*, (1995a,b) used an ANN to determine rock porosity and permeability accurately using data from wireline logs of petroleum wellbores. Mohaghegh *et al.* (1996) claim significant reservoir characterisation cost savings using a similar approach. Pezeshk *et al.* (1996), discuss the more general application of interpretation of geophysical logs using ANN for geological and hydrogeological classification with significant time savings.

3.2.4 Geological identifcation

A variety of ANN applications has been developed for the purposes of the geological identification of rock types and they can be broadly distinguished by means of the source data used.

Well log data is an obvious data source where there is a direct financial implication of improved classification. Applications that employ this data source are numerous. The first example is that of Huang and Williamson, (1996) who used rock cutting geochemical analyses coupled with well log data to identify oil source rocks, using an ANN that predicted total organic content. The implementation of Wong and Gedeon (1995) uses an ANN to detect outliers and remove noise in well log data in order to identify rock types by more conventional means. Chang *et al.* (1997) used a hybrid neural network / fuzzy classification approach to infer lithologies based on well log data and expert knowledge provided by site geologists.

If petrographical descriptions or thin sections of rock samples are available, then ANNs can operate directly on this data source Cardon *et al.* (1991). Carr and Hibbard (1991) argue that there is a significant bias in rock classification by means of thin section through the primary discrimination of the group (igneous, sedimentary, or metamorphic) to which the rock belongs. To overcome this bias they have produced an ANN based objective 'geologist's assistant' to aid in this primary discrimination.

With the increased prominence of deep sea engineering applications, more importance has been placed on identifying sea bed material type. Motooka *et al.* (1995) and Caiti and Parisini (1991) have used sub sea sonar and echo pulse data to discriminate rock, sand and silt materials using ANNs to interpret the raw data.

The voluminous nature of data from satellite based remote sensing and airborne geophysics is amenable to treatment using ANNs as there are significant time benefits in automating the interpretative / classification task. An *et al.* (1995) used small samples (typically 0.2-0.5% of the total available) of data from both sources to calibrate an ANN that enabled bedrock type classification with accuracy varying between 85-90%, showing that such an approach is effective. This is supported by Ninomiya (1995) who used thermal infrared imagery sourced from satellites to quantitatively estimate the percentage silica content of bedrock. After calibration of the ANN, errors between network predictions and ground geochemical analysis are quoted as being less than 7%. Of course, other methods of automatic classification of satellite imagery are available, the most prominent being Baysian classification. However, in his studies using multiple source satellite geophysical data, Gong (1996) shows that ANNs can be more reliable estimators than alternative, more conventional methods.

3.2.5 Rock-Structure Performance

The expression 'rock-structure performance' is taken to mean the behaviour of natural geomaterial masses when they are subjected to engineering. In this sense, a 'structure' is interpreted as any excavation as well as any support, reinforcement or foundation structure introduced.

There are many examples of ANNs being used to model or represent rock structure performance. In the following, the review is structured by the rock engineering system considered—underground excavations, rock/soil slopes, and rock/soil foundations.

Underground excavations

The first ANN application developed to assist in underground excavation design was reported by Lee and Sterling (1992a), (1992b). In this case, the network was used to associate sets of rock mechanics and geological parameters with failure modes, the training data being sourced from a data base of case studies. Their 'associator' shares a similar basis, and is an alternative to, standard rock mass classification schemes. A second 'pattern association' type application is presented by Feng *et al.* (1993). Their approach to underground excavation support design employs three distinct ANNs: the first undertakes a rock mass classification using geomechanical parameters; the second classifies the rock mass structure on the basis of structural geological features such as the average thickness of strata; and the third associates the output of these first two networks and excavation design parameters such as the shape and span of the opening with the opening support type and dimensions.

King *et al.* (1993) use data from routine condition monitoring of roofbolting equipment with an ANN trained in an unsupervised manner to develop a classifier for the geomechanical behavioural 'class' of the hanging wall in a coal mine roadway. In this treatment, no consideration of the dynamics of rock mass behaviour is made. In contrast, in a study by Zhang *et al.* (1995) that did attempt to characterise rock mass dynamic behaviour, coal mine roadways were also the subject. They employed spatio-temporal time series of convergence data to train an ANN to forecast future roadway convergence, although the quality of their predictions degraded appreciably just a few days ahead, and took no account of variability of near field geology or other *in situ* factors such as stress.

An approach that does attempt to account for all of the environmental factors and rock mass character, both of which influence the behaviour of underground excavations in rock is that due to Cai *et al.* (1996), and thus deserves detailed attention.

Hudson's (1992) rock engineering systems concept is employed as the template for the dynamics of the underground excavation process. The rock characterisation core is the Q-system structure developed by Barton, (e.g., Barton *et al.* (1974,1980)) and excavation performance assessment is made through observation of the induced tangential stress, excavation radial displacement and joint aperture. A case study database is composed according to this structure and for each record a perceptron network is trained. The weights of these trained networks are used as sets of interaction matrix coding parameters. Attractor states of dynamic, recurrent operation of the Interaction Matrix as discussed by Hudson and Hudson (1993), together with template parameters, are used as source for the training of a Hopfield network. The 'energy' of the Hopfield network is taken as an indicator of the potential energy contained within the underground excavation system and thus a stability indicator.

Foundations

Modelling the performance and behaviour of piles using ANNs has received appreciable attention. Stress wave data provided the basis for the ANN methods of Chow *et al.* (1995) (load bearing capacity of concrete bored piles) and Teh *et al.* (1997) (load bearing capacity of precast steel reinforced concrete piles). In an alternative approach specific to pile type, Goh (1996b) predicts the load bearing capacity of driven piles on the basis of hammer characteristics, and the properties of pile and soil.

In a slightly different application area, Tohmaz and Hassan (1995) produced an ANN simulator of tractive performance of rubber tyred plant on soft organic soils, relating pull-load and tyre inflation pressures for the purpose of plant selection. At a larger scale, the foundations of a concrete dam on rock were investigated by Ohnishi and Soliman (1995) who produced an *in situ* seepage estimator ANN.

Slopes

Coca-Stefaniak, (1993) developed an instability hazard indicator using an ANN for the Panama Canal slopes. Multilayered perceptron architectures were used with a database of slope morphology parameters, geological descriptions and index test results built up from case histories of the numerous and extensive slope instabilities that occurred during initial construction and various widening phases of the Canal. This is an example of an ANN model specifically tailored to particular site conditions, with the consequence, as with other empirical techniques, that, although the methodology is transferable to other sites, the actual network developed is not. A similar indicator methodology for debris flows in Japan is reported by Hirano *et al.* (1994).

When the parameters used in development of ANNs are non-site specific, the resulting networks are readily transferable, especially when they are trained using case studies from a wide variety of sites. This assertion is supported by the work on rubble mound breakwater stability assessment, developed by Mase *et al.* (1995). Their ANN involves 7 parameters: stability number (output), current damage level, number of attacking waves, a surf similarity parameter, a permeability parameter, the dimensionless water depth in front of the structure, and a wave spectral shape parameter. As the variables involved were of a generalised nature, predictions of the stability number from their ANN, trained on data from a given site, generalised well when tested with data from other sites.

The best 'portability' of a trained ANN is obtained when the variables it relates are of a generalised nature and 'contain' generalised mechanics. An approach of this type for rock slope stability was presented by Millar, Jiao, Arnold and Hudson (1994) which was the first example of full integration of Hudson's rock engineering systems methodology and ANNs.

3.2.6 Plant control and performance

Tunnelling

The main focus of the application of ANNs in tunnelling is in tunnel boring machines (TBMs), with ANNs taking the role of feedback controllers. Aoshima *et al.* (1993) have used a direct approach, allowing the neural network to feedback corrections to roll and alignment offset variables, to maintain and optimise the intended tunnel alignment. Their ANN learns the response characteristics of the TBM *in situ* to piloting corrections made by human operators and conventional automated controllers.

Takahashi and Hosaka (1996) presented a hybrid scheme comprising a conventional directional controller that is superseded by an ANN directional controller for a microtunnelling machine designed to excavate underground telecommunications cable conduits. Their system is an improvement on that of Aoshima (although it has only been assessed in simulations) as it also accounts for the ground reaction on the TBM. Takahashi and Yamada (1997) have modified the former system by deriving a radial basis function ANN from a trained perceptron network in order to estimate the TBM transfer function (Jacobian) resulting in a controller with improved reliability.

Mobile plant

An example of tunnelling equipment controlled by ANNs for applications not involving TBMs is provided by Lever *et al.* (1995). They use an ANN to identify one of nine fixed modes of operation of a front end loader / excavator, and to select the next required excavator behavioural program. In a completely automated application

for remote operation, the nine behavioural programs are specified by a fuzzy logic expert-system type controller derived from skilled human operators.

Static plant

The control of cone rock crusher height is of vital importance in maintaining an output grade that is within specification for finished aggregate. Parking and Calkin (1994) presented a neural network controller for crushing plant that is based on digital imaging of the crushed stone. Image processing algorithms extract size and shape parameters which are then used as input to a neural network that outputs a particle characterisation parameter. In pilot scale trials, this parameter is used to adjust the crusher settings until the crushed stone output is within specifications. Other example mineral processing applications of ANNs can be found in Aldrich and van Deventer (1993), Perez *et al.* (1993), and Cutmore *et al.* (1997).

3.2.7 Blasting / Seismic Geophysics

Seismologists have used ANNs to assist in inversion of raw geophysical data into a format suitable for intepretation of geological structures (faults, shear zones, etc.) of interest (see for example, Swininarski *et al.*, 1993). Huang and Wanstedt (1996) also showed how an ANN can assist in the delineation of ore bodies when operating upon geophysical logging data for mining applications. Musil and Plesinger (1996) discuss the use of an ANN to solve a small but interesting seismic problem. They considered an area prone to geological microseismic events that was also an area of extensive mining and quarrying, and hence blasting operations. Their ANN was trained using three spectral parameters and three amplitude parameters derived from velocigrams and produced a reliable discriminant between natural microseimic events and blasting operations that is now in operational use.

Production blasting in mines and quarries has to be carefully planned and explosive load and distribution carefully tailored to the *in situ* geomechanical conditions. This latter aspect is conventionally done with the use of crater or test blasts followed by heuristic adjustment. In order to mitigate such difficulties, Cai (1995) presents an ANN that classifies the rock for the purposes of blast design using case studies. A measure of efficiency of production blasting operations is the resulting fragment size distribution. Due to the volumes of rock involved, especially with open pit or quarry blasting, it is difficult to obtain reliable distribution curves. Many workers have attempted to use image processing algorithms to overcome this problem with varying degrees of success. An effective methodology coupling ANNs and image processing routines is that reported by Barron *et al.*, (1994). Their classifier was

consistently able to identify the fragment size classes, which were represented as percentages passing a certain screen size.

3.2.8 Geostatistical Applications

Many authors have employed ANNs in the field of mining geostatistical reserve estimation (see for example, Clarici *et al.* (1993a), (1993b), Denby and Burnett (1993), Wu and Zhou (1993)). This work is interesting from the rock mechanics / rock engineering point of view because mining geostatistical applications are specialised to deal with spatially distributed mineralisation figures within the ground. There is no difference if, for example, the underlying data reflect a spatially distributed value of, say, Young's modulus or some other geotechnical variable, such as rock head elevation as reported by Zhou and Wu (1994).

ANNs remove the statistics from geostatistics in the sense that their main purpose has been to replace variogram models that represent spatial correlation between values. ANNs learn the distribution directly from spatially indexed assay or other test data. Kriging, which is essentially a technique that interpolates between the test values using variogram models, has been described as the "best linear unbiased estimator" (Fookes, 1985). As it is reasonable to assume that spatial distributions of mineral value, Young's modulus, etc. are frequently nonlinear in nature, the consequence is that Kriging methods can only produce linear approximations to nonlinear distributions at best.

In contrast, semi-linear ANNs are estimators of any reasonable function, with no constraint on (non-)linearity. It is thus natural to expect that they will produce better models than contemporary geostatistical methods. This is confirmed by Clarici *et al.* (1993b), and consequently it is somewhat surprising that Dowd and Sarac, (1994) chose to use ANNs to learn variogram models.

3.2.9 Soil Classification

Although the primary concern of this thesis is rock mechanics and rock engineering, as has already been seen, many useful ANN applications have been developed within the soil mechanics field. In this Section, soil classification schemes are discussed. In the next, applications that have a strong soil hydrogeological content are covered.

Cal (1995) reported on an attempt to upgrade the current 'A-line' approaches (see for example, Smith, 1978 or Berry and Reid, 1987), of soil classification to a more quantitative basis using an ANN classifier. Index properties that are used with the contemporary methods (plastic index, liquid limit water capacity and clay content) are again used and associated with numerical categories to yield a more objective and sensitive classification. Levine *et al.* (1996) used percent organic carbon, silt and clay

as inputs into a perceptron type ANN that was used to discriminate between three classes of soil structure (granular, blocky and massive) with a quoted 79% accuracy.

Goh (1994, 1995) has conducted appreciable research into modelling soil correlations using ANNs, in particular with reference to assessment of seismic liquifaction potential. Perceptron architecture neural networks were trained with: the standard penetration test value, the fines content, the mean grain size, the equivalent dynamic shear stress, the total stress magnitude, the effective stress magnitude, earthquake magnitude and the maximum horizontal acceleration at ground surface as input parameters derived from field observations. Liquifaction indices were produced as output, of undoubted importance in assessment of slope stability. Goh (1996a) extends the capabilities of the scheme by associating soil liquifaction indices with cone penetration test data by means of an ANN.

3.2.10 Soil hydrogeology

Tamari *et al.* (1996) employed ANNs to correlate soil index properties to the hydraulic conductivity. They reported that their neural networks are more efficient in predicting hydraulic conductivity than multilinear regression techniques to which they were compared, especially when uncertainty superimposed on the data set of training exemplars was small. Consequently, they concluded that, when a large data base with accurate values is available, ANNs can be useful to predict hydraulic conductivity parameters over a broad spectrum of soils. Other workers that have investigated soil hydraulic conductivity using ANNs include Gribb and Gribb (1994) who overcame the problem of hydraulic conductivity determination when the soils considered are in the unsaturated condition, using a neural network.

Pachepsky *et al.* (1996) also found that ANNs were more reliable than statistical regression techniques when estimating soil water retention for the purposes of extension of the applicability of hydrological models. Soil texture, bulk density and hydraulic potential were the input variables and water content was the output variable in their scheme. Schaap and Bouten (1996), who also modelled soil water retention using ANNs, had a similar set of input variables but included soil organic matter content and replaced a texture parameter with the complete particle size distribution. The output of their neural networks was a series of points on the water retention curve. These predictions had low root mean square errors in comparison with points taken from measured curves for the same soil samples.

3.3 Survey of perceived benefits

The list below summarises the subject areas within geotechnical engineering in which ANNs have been used as discussed in the previous Section.

- Constitutive Modelling

- Numerical modelling
- Rock Characterisation
- Geological identifcation
- Underground excavations
- Foundations
- Slopes
- Tunnelling
- Mobile plant
- Static plant
- Blasting / Seismic Geophysics
- Soil Classification
- Soil hydrogeology

From the list it can be seen that there are few areas of geotechnical engineering that have not been 'touched' by ANNs. However, as no source detailing the application of ANNs within geotechnical engineering seems to exist prior to the seminal works in this field by Zhang *et al.* (1991) and Ghaboussi *et al.* (1991), this topic can be said to be relatively immature with a potential yet to be fully realised. Consequently, it seems appropriate to briefly summarise the perceived benefits of the use of ANNs reported by the authors of the articles reviewed in the previous Section. For brevity, this is done in Table 3.1, the benefits mentioned by each author being broadly classified into four categories.

(A) New capabilities. These are ANN applications that provide new capabilities driven by novel information sources, e.g. applications that allow the interpretation of data from conventional or contemporary sources and consequently provide new or significantly improved capability. ANNs represent a capability to provide a rational treatment of complex phenomena and to identify key or pivotal parameters or variables within such phenomena. As such, applications that have exploited either of these two characteristics have been included in section A.

(B) Automation. Applications in this category reflect benefits in being able to replace some task that until now has been undertaken by humans. The nature of automation generally means that these perceived benefits are normally associated with a given process. Sub-classes within section B are: automation applications that provide a replacement for a 'human operator' with the same performance; those that lead to improved performance; and those that lead or will lead to deployment with remote operation.

(C) Utilisation of data. One of the main benefits of the use of ANNs is an improvement in the utilisation of data. This is particularly the case when, for example, an incomplete data record would lead to the good entries within the record being

rejected in, say, the formulation of a new model. A second example is where a complete dataset exists, say as a by-product of some other necessary process, that is known to be somehow related to a phenomenon of interest, but in an indirect, not well understood manner. A third case arises when the signal to noise ratio in a dataset is low and as such trends or patterns within the data are unclear. ANNs offer clear benefits in this regard for almost all applications. Examples in each of these three categories are given in Table 3.1.

(D) Replacement of models / methods. Where existing models are present for a specific application area, the issue of in what way the ANN represents an improvement over the existing models arises. These issues are addressed in the sub-classes of section D. ANNs can replace existing models (however they have been derived / produced) because (i) they are available, (ii) they enable the inclusion of a model in a much simplified manner in comparison to the existing models, (iii) they are more robust and fault tolerant in the light of uncertainty or noise, or (iv) they produce better 'answers'. In the last case, it is frequently the case that there is a mismatch in the characteristics of the modelling method with the characteristics of the phenomenon or data being modelled. For example, if the data or phenomena are non-linear, a linear estimator should be expected to provide an inferior estimate in comparison to a non-linear estimator. ANNs are clearly beneficial in this regard.

Table 3.1: Summary of perceived benefits of using ANNs.

(A) New Capabilities	Subject Area
(1) applications that provide completely novel capability	
Cai, J., Zhao, J., Hudson, J.A., and Wu, X., 1996	Rock engineering systems
Coca-Stefaniak, J.A., 1993	Slope stability
Feng, X., Lin, Y. and Wang, L. 1993	Underground excavations
Lee, C.A., and Sterling, R.L., 1992a,b	Excavation stability
Mase, H., Sakamoto, M and Sakai, T, 1995	Breakwater stability
Millar, D.L., Jiao, Y., and Hudson, J.A., 1994	Rock mass classification
Motooka, S., Kitmai, M., Ogawa, K, and Takano, N., 1995	Materials characterisation
Topping, B.H.V., Khan, A.I., and Bahreininejad, A. 1997	Mesh decomposition
(2) interpretation of existing data sources in new ways	
An, P., Chung, C.F., and Rencz, A.N., 1995	Geological classification
Carr, J.R., and Hibbard, M.J., 1991	Geological classification
Chow, Y.K., Chan, W.T., Liu, L.F., and Lee, S.L., 1995	Concrete pile design
Ghaboussi, J. Garrett, J.H., & X. Wu, 1991	Constitutive behaviour
Millar, D.L., and Hudson, J.A., 1994	Performance monitoring
Musil, M., and Plesinger, A., 1996	Seismic discrimination
(3) improvements in existing capabilities (new methods)	
Barron, L., Smith, M.L., Prosbrey, K., 1994	Blast fragmentation
Cai, Y.D., 1995	Blast design
Caiti, A. and Parisini T., 1991	Seismic discrimination
Ninomiya, Y., 1995	Satellite petrographics
Schaap, M.G., and Bouten, W., 1996	Hydrogeological analysis
Takahashi, K., and Hosaka, H., 1996	Micro TBM controller
Wong, P.M., Taggart, I.J., and Gedeon, T.D., 1995	Reservoir characterisation
(4) rational treatment of complexity	
Cai, J., Zhao, J., Hudson, J.A., and Wu, X., 1996	Rock engineering systems
(5) identifcation of key variables	
Goh, A.T.C., 1994	Seismic liquifaction
Cai, J., Zhao, J., Hudson, J.A., and Wu, X., 1996	Rock engineering systems

(B) Automation	
(1) leading to improved speed and equalling human performance	
Willams, T.P., and Gucunski, N., 1995	Pavement design
(2) leading to improved control	
Cutmore, N.G., Liu, Y., and Middleton, A.G., 1997	Mineral processing
Parking, R.M., and Calkin, D.W., 1994	Rock crushing
Takahashi, K., and Yamada, T., 1997	TBM controller
(3) leading to remote operation	
Aoshima, S., Takeda, K., and Yabuta, T., 1993	TBM operation
Lever, P.J.A., Wang, F.Y., Chen, D.D., and Shi, X.B., 1995	Excavator operation
Takahashi, K., and Hosaka, H., 1996	Microtunnelling

(C) Utilisation of data	
(1) that is unreliable or uncertain	
Wong, P.M., and Gedeon, T.D., 1995	Well bore analysis
(2) utilisation of indirect data	
King, R.L., Hicks, M.A., Signer, S.P., 1993	Roof bolting
(3) maximisation of data utilisation	
Chang, H.C., Chen, H.C., Fang, J.H., 1997	Well bore analysis

Table 3.1: Summary of perceived benefits of using ANNs (contd).

(D) Replacement of existing models / methods	
(1) as straightforward alternatives	
Teh, C.I., Wong, K.S., Goh A.T.C., and Jaritngam, S., 1997	Pile design
(2) due to simplicity	
Goh, A.T.C., 1996a	Pile design
Wu, X.P., and Zhou, Y.X., 1993	Geostatistics
(3) due to increased reliability	
Basheer, I.A., and Najjar, Y.M., 1995	Pavement design
Dawson, M.S., Fung A.K., Manry, M.T., 1997	
Goh, A.T.C., 1996a	Pile design
Pachepsky, Y.A, Timlin, D., and Varallyay, G., 1996	Hydrological modelling
Tohmaz, A.S., and Hassan, A.E., 1995	Mobile plant selection
(4) due to improved estimation	
Attohokine, N.O., and Fekpe, E.S.K.,1996	Pavement design
Cal, Y., 1995	Soil classification
Clarici, E., Owen D. and Durucan S., 1993a	Geostatistics
Ellis, G.W., Yao, C., Zhao, R., and Penumadu, D., 1995	Constitutive modelling
Gong, P., 1996	Satellite geophysics
Huang, Z.H., and Williamson, M.A., 1996	Well log characterisation
Levine, E.R., Kimes, D.S., and Sigillito, V.G., 1996	Soil classification
Millar, D.L., and Calderbank, P.A., 1995	Numerical modelling
Schaap, M.G., and Bouten, W., 1996	Hydrogeological analysis
Tamari, S., Wosten, J.H.M., and RuizSuarez, J.C., 1996	Hydrological estimation
Wong, P.M., Gedeon, T.D., and Taggart, I.J., 1995	Well bore characterisation

To some rock engineering users of ANNs, they are black boxes that have been calibrated to produce a certain output given some input values and therein possibly lies their greatest disadvantage. Until they are accepted for use within geotechnical engineering disciplines as much as in other disciplines, acceptance of such 'black box' methods will be slow. However, it is perhaps useful to be reminded that currently most geotechnical engineers know relatively little about the detailed workings of numerical geomechanical codes, as they too lie within the domain of the computer scientist or programmer, and yet over the past decade they have made considerable gains in respectability and use.

3.4 Compound approaches

Chapter 2 discussed the fact that many different types of ANNs can be created through variations in microscopic and mesoscopic structure. Before leaving the review of ANNs in geotechnical engineering, it is useful to identify the ANN methods that have been used by workers in the geotechnical engineering field.

The majority of articles reviewed in Section 3.2 adopted a multilayered perceptron trained using backpropagation of error. This is due to its simplicity and a consequence of the good performance of this network architecture for most datasets of a geotechnical engineering nature. As the majority of geotechnical applications use perceptron architectures, the remainder of this Section will deal with variations of this method and the other methods that have been used.

As has already been discussed, Cai *et al.* (1996) used a combination of a multilayered perceptron, a Hopfield network, and an expert system. In a general sense, their scheme endows the perceptron architecture with a pre-processing role. In a contrasting approach, King *et al.* (1993) use a perceptron architecture as a post-processor. In their scheme to detect geomechanical features in coal mine roofs, an unsupervised *k*-nearest neighbour, self organising ANN is used to establish exemplar pattern vectors from abstract data from several sources. The perceptron is used to classify these exemplar patterns into meaningful geomechanical classes.

In their mine or quarry blast / microseismic activity discriminator, Musil and Plesinger (1996), used two different ANN methods simultaneously with similar, but distinct roles. The first, a multilayered perceptron, was used as the basic discriminator. The second, a Self Organising Map, was used to assess the reliability of the discriminative event. In verification of the multilayered perceptron, it correctly discriminated 99% of the events with which it was tested. It also incorrectly discriminated 1% of events. However the Self Organising Map correctly identified the same 1% as problematic, and a further 20% of the 99% correctly discriminated events as 'difficult'. Musil and Plesinger have thus demonstrated that use of ANNs in such complementary roles offers improved benefits over using a single method alone.

Ellis *et al.* (1995) described the use of a perceptron-type architecture with feedback or recurrent links to learn constitutive behaviour of unconsolidated sands. They concluded that recurrent architecture ANNs offer improved performance over non-recurrent architectures, when stress path history information is to be fully incorporated, including multiple load-unload loops.

Several authors have used the basic perceptron schemes with improvements in the training algorithm (Yang and Kim, 1996). For example, Huang and Williamson (1996) used the quickprop training algorithm which, as its name suggests, improves the time required to train perceptron schemes. In order to accelerate the learning process even further, Huang and Williamson also adopted a dynamic node creation / destruction algorithm, which results in network a mesostructure that is optimal with respect to the training process itself, not necessarily with respect to the data used to train the network. Further improvements can be made in parallelising the training algorithm (e.g., Topping *et al.*, 1997) so that it will operate on multiple computer processors concurrently. Speed up is not linear with increasing numbers of processors due to processor intercommunication overheads.

Other types of ANN models used include: radial basis functions, (Takahashi and Hosaka, 1996; Tamari *et al.*, 1996); learning vector quantization, (Millar and Hudson, 1994; Millar *et al.*, 1994); and fuzzy associative memories (Chang *et al.*, 1997).

3.5 Recent developments

Alvarez-Grima *at al.* (2000) reported on the use of both unsupervised and supervised learning ANN methods in the context of assessing tunnel boring machine performance. Initially, the unsupervised methods were used in a cluster discovery process in order to reveal order and structure in a database of over 640 case studies of tunnelling performance. Subsequently a supervised stage of learning is undertaken with a separate network that estimates likely TBM advance rate for new projects, based on geological, geotechnical and machine design parameters. Alvarez-Grima *et al* (2000) do not report trained ANN weights, but do show that their ANN approach is better than traditional statistical regression techniques by at least four times, this being measured in terms of the magnitude of RMS error on a validation data set.

In a sophisticated formulation, comprising three Adaptive Resonance Theory ANNs, use of fuzzy membership functions as inputs and an expert system, Chang *et al.* (2000) presented a tool for identifying lithofacies from petroleum well logs. An important aspect of their work was the fact that the three ANNs were each used to make predictions of lithofacies. The three predictions were reconciled through the application of expert system rules. The authors claim a more accurate well log classification with this scheme and importantly, the authors have built in output classifications that provide an indication of certainty of the predictions made. Marmo *et al.* (2005) have built on this idea, developing a carbonate rock texture classifer that operates on thin sections that they report as being 93% accurate. In their authoritative work, Duboisa *et al.* (2007) demonstrate that facies classifiers based on Bayesian inference, fuzzy logic, and *k*-nearest neighbour analysis are inferior to facies classifiers developed with ANNs when predicting rock types in gas well bores in Kansas, US.

In a later paper, Chang *et al.* (2002) recognise that self organising topological maps (SOM) are superior tools for the analysis of well logs in their application. This is because, not only is lithofacies identification accuracy comparable with, or better than, those of the back-propagating perceptrons and ART networks of their earlier study, the SOM imposes a spatial order on the training data patterns. The clusters produced by the SOM were found to be ordered with respect to paleo-environmental energy levels, an aspect which was found to be useful by geologists.

Hong and Rosen (2001) use a SOM to present the dynamics of chemical species concentration in a ground water aquifer in a fractured rock mass. Using the SOM, Hong and Rosen (2001) were able to identify correlations between rainfall events and migration of heavy metal species to specific locations within the aquifer. A similar theme is explored later by Samecka-Cymerman *et al.* (2007) who develop a SOM to

monitor and interpret the effect of chemical species in mosses transported by streams passing through various lithologies in Poland.

Huang and Wanstedt (1998) presented three short applications of ANN in rock mechanics contexts: i) delineation of an orebody, ii) aggregate quality prediction and iii) indentation depth prediction. In the last two of these applications, generic ANNs were developed that would be useful to workers in a broader range of disciplines and in applications not considered by the authors. Unfortunately, these authors do not present ANN weights or the training data set used.

Kim *et al.* (2001) present a valuable methodological contribution to the field of ANN in geotechnics by demonstrating how the so-called Relative Strength of Effects measure can be applied to a data mining of a trained ANN. The ANN inputs are systematically varied above and below nominal values in a similar manner to parameter 'flexing' in conventional sensitivity analysis. The outputs of the trained ANN are then used to determine the sensitivity of ANN predictions on variation in the input quantities. Sonmez *et al.* (2006) have also applied ANN data mining techniques to produce a useful chart allowing estimation of intact rock Young's modulus, based on intact rock strength and unit weight. It is interesting to note that these workers efforts to produce similar estimators for rock mass modulus appear to have reverted to more traditional techniques over neuro-fuzzy approaches this team reported earlier Gokceoglu *et al.* (2004).

Leu *et al.* (2001) used an ANN approach to establish a tunnel stability indicator. Despite their high learning rates with a back-propagation type ANN, they managed to train their ANN with approximately 1000 case study examples for tunnels in a specific sandstone formation. By data mining their ANN, the authors were able to establish that factors associated with tunnel support construction were a much more significant determinant on tunnel stability than geotechnical properties. They assert that this conclusion is specific only to their particular tunnel, yet do not publish trained ANN weights. Similarly, Bizjak and Petkovšek (2004) discuss a useful ANN formulation that uses observations of tunnel convergence taken at a given tunnel section after the first 7 days of excavation (as inputs) to predict the final expected convergence of the section (as output). Although details of the ANN meso-structure are provided, network weights and thresholds are not. Refreshingly, logs of geotechnical parameters from seismic inversion processes are present within the work reported by Klose (2006), and Klose *et al.* (2007). In this work, a SOM calibrated using seismic data from the Faido tunnel in Northern Italy was successfully used to predict condition and performance of the rock mass in the adjacent Gotthard base tunnel.

Jamialahmadi and Javadpour (2000) present an ANN approach to establishing correlations between porosity and permeability parameters with depth in an Iranian

oilfield. The paper contains no statement of the empirical relation found as no ANN weights are presented, nor the input training data file. The only real contribution here is re-affirmation that the methodological approach works. The work by Katz *et al.* (1999) in predicting porosity is disappointingly similar.

In a classic illustration of this problem, Meulenkamp and Alvarez-Grima (1999), present data on geotechnical index testing conducted to establish a data set to correlate Equotip hardness to UCS. An ANN was trained and comparative curve fitting techniques were explored. Firstly, a power law was fitted to the data, and the equation exponents and constants presented. Secondly, a multivariate regression analysis was conducted. Again, the equation was presented together with the equation's co-efficients. The neural network was then tested and a lengthy discussion follows about its merits in comparison to the first two methods but the ANN weights are not presented in the paper. Singh *et al.* (2001) having quoted the results of Meulenkamp and Alvarez (1999) proceed to make the same error, but this time in an even more useful ANN formulation to predict the strength of schistose rocks from their petrographic properties.

Other than state that a multilayer perceptron trained with back-propagation of error was used, Cizb and Rudajev (2007) decline to provide any further details on an otherwise useful ANN they developed to accomplish a complex data fusion task that apparently assimilated autocorrelation parameters, variogram parameters and the fractal dimension of acoustic emission data across a series of input channels, where the ANN output was used to assess rock sample stability. In their paper, Kahraman *et al.* (2006) provide the data used to train the ANN and clear statements of ANN microstructure, mesostructure and heuristics (e.g. learning rates, etc.) such that their results on sawability of carbonate rock can be reproduced, if necessary. Without providing trained weights, reconstruction of these results is subject to the vagaries of initialising the weights and thresholds with random numbers. The same is true of the work reported by Khandelwal and Singh (2006) on predicting the magnitude and frequency of blast induced vibrations. In contrast, Yang (2004) provides network structure and weight and threshold values in a report of ANN use on estimating grout take for dam foundations that is exemplary.

Sirat and Talbot (2001) used back-propagating perceptrons, learning vector quantization and self-organising topological maps to classify set membership of discontinuities. Lemy and Hadjigeorgiou (2003) use an ANN to post process vector fracture lineaments in the development of (semi-) automated methods for production of fracture trace maps. These contributions will be discussed in more detail later in the Thesis.

A valuable contribution to the field of ANN in rock engineering was made by Feng *et al* (2000). They used a back-propagating perceptron ANN to summarise the results of a short series of finite element method (FEM) analyses of a rock slope. Due to scale and detail, the FEM analyses were computationally expensive. Parameters used in these FEM runs had 'end-member' values. The trained ANN was used in place of the FEM analyses in a back-analysis search for optimal parameter values. The computational effort of the ANN is trivially small in comparison to that required for the FEM analysis. Having determined optimal parameter values, these were then checked by running a FEM analysis with them to reveal good agreement with observations.

Another example of slope analysis with ANNs is the formulation of Wang *et al.* (2006) which is more direct and more empirical than that of Feng *et al* (2000) as it relies on 26 case analyses of slope stability and instability to predict the slope stability of a 27^{th} case in the same slope sequence, without the intervention of a numerical modelling method. The approach is somewhat undermined by the fact that the ANN is used to predict a factor of safety for the slope considered, as supervised learning exemplars used limit equilibrium analysis to produce values for this quantity. The factor of safety for the new case was predicted using the ANN and compared to the limit equilibrium determined value, with the former appearing to take priority in reporting. In this instance the weight assigned to the ANN output appears counter-intuitive.

3.6 Summary

In this Chapter the basic application areas of ANNs in general engineering tasks and a detailed review of their use in geotechnical engineering have been presented. The benefits of using ANN have been identified as:

1) They provide a completely new capability.
2) They facilitate automation of equipment or methods.
3) They improve the utilisation of data.
4) They offer alternatives to existing modules or methods.

In recent years, there are some disturbing trends in the geotechnical ANN literature, particularly in papers where some correlation between rock mechanics parameters is established. In order for a reader of such papers to use the results that these authors find, either the training data set or the full ANN details must be fully specified. For transferable use of multi-layered perceptron ANNs:

i) the trained ANN weights and thresholds must be presented,
ii) the ANN microstructure must be stated,
iii) any normalising parameters or other pre-processing must be stated; and

iv) network connectivity must be specified.

Without all of these details being presented, the 'worth' of the paper reduces to a methodological statement. Some authors may argue that it is not necessary to present the ANN weights; however, the view of this author is that it is crucial. In these kinds of applications, ANNs are being used as regression tools to establish empirical relations.

Many relations within rock mechanics are empirical in nature. An example is the Hoek-Brown failure criterion. Without the publication of the empirically derived co-efficients for the Hoek-Brown strength criterion in their book, Hoek and Brown (1980), the criterion would have been of much less worth and be far less widely used. This situation is much improved with a full methodological statement coupled with presentation of the data with which the ANN was trained. In fact, where space allows, it is most useful to present both the data and the details of the trained network. If space cannot be afforded within the publication itself, a reasonably stable source to the data or trained network details should be referred to. The ISRM may have a role to play in providing an internet-based archive for authors to submit their networks. It is not surprising that many workers in rock mechanics and rock engineering have a rather disparaging or suspect view of ANNs given the less than comprehensive manner in which results have been published in recent years.

On a more positive note, the ANN literature contains developments that will have widespread impact on the rock mechanics and rock engineering discipline for many years to come. For example, ultimately all numerical analysis packages used in geotechnical engineering will be equipped with the methodology first presented by Feng *et al.* (2000) but also confirmed by the work of Deng and Lee (2001). This is the best application of ANNs in rock mechanics that this author has found, and will be proved a genuinely innovative step, if not widely known to be at present. Both of these sets of authors adopt a genetic algorithm as the optimising agent.

4 Training ANNs with data from simulated compression tests on linearly elastic rock

4.1 Introduction

The core techniques of parallel distributed processing that will be exploited within this thesis have been explained. At this point, we turn to the first application area that will be studied: using ANNs to model the stress-strain constitutive behaviour of rock. The application is described across two chapters. In this Chapter the development of an ANN that models linear elastic constitutive behaviour is detailed, based on the use of computer simulation techniques. In the next Chapter, the ANN techniques are developed to allow their application to compression test data acquired for real rocks. Importantly also, the next Chapter also explains how the resulting network is used in a finite difference formulation numerical code.

This Chapter also serves to illustrate the general procedure (described in Chapter 2) for training ANNs. This is done with reference to an application to the constitutive modelling of rock. The application uses synthetic data which simulates a triaxial compression test on an isotropic linear elastic rock. In the first Section of this Chapter, introductory comments on constitutive modelling in rock mechanics are presented in order to fully define the context for the work. Then the constitutive framework is described in the context of a triaxial compression test. In the third Section, the constitutive framework is used to develop a simple computer simulation procedure which generates the synthetic data use to train the network. The fourth Section deals with the specifics of pre-processing the synthetic data prior to the ANN training and validation process. The fifth Section reports on how effective the trained ANN is, both in representing the data on which it was trained and data that were not part of the training data set.

Various "what-if" scenarios are considered subsequently. Firstly, checks are conducted to determine that the ANNs learn data that are specific to the given rock type. Secondly, checks are made to determine whether the data pre-processing is suitable. Then checks are made to determine the best sources of test data to use to train the ANN and, finally, an examination of the effect of the degree of control in the laboratory equipment is made.

It is evident that an ANN is not an appropriate mathematical tool with which to model isotropic linear elasticity; Hooke's Law is much more compact, more computationally efficient by many orders of magnitude and is transparent analytically. However, training an ANN to reproduce the generically well understood behaviour of Hooke's Law provides insight into the final objective of using an ANN to model more complex constitutive behaviour in a compact and portable manner. More importantly,

it is to be expected that the primary data source for constitutive models embodied within ANNs will be conventional or near conventional laboratory testing techniques, such as triaxial compression testing.

By combining the concepts of triaxial compression testing and ANN training and validation procedures, the material presented in this Chapter investigates whether this is a realistic proposition before progressing onward to the more complex tasks associated with real constitutive behaviour.

4.2 General comments on the constitutive behaviour of rocks

Most, if not all, rock mechanics or rock engineering problems need some kind of assessment of how *in situ*, applied or induced stress fields lead to *in situ*, applied or induced strain fields, or *vice versa*. Conventionally, the tensor quantities, stress and strain are taken to be related through the deformability and strength properties of the medium concerned. In rock engineering the medium is rock or a rock mass depending on the scale of engineering and the characteristics of these media *in situ*.

The simplest conception of deformability behaviour is that some change in the state of stress within a rock body leads to a corresponding, although not necessarily directly proportional, change in the state of strain. The complementary relation also holds: a change in the state of strain will lead to a corresponding change in the state of stress. However only in the most special of cases should these two relations be considered the inverses of one another. The reason for this relates to the character of geological media explained in the next paragraph.

The inherent characteristics of the medium are the main reason for the difficulties: rock at engineering scales is frequently *discontinuous* due to the near ubiquitous presence of fractures, faults, fissures, joints and so forth; it is *heterogeneous* due to intrinsic spatial variability of rock mineralogical composition; and it is *anisotropic* as a result of the first two characteristics and perhaps geological processes such as sedimentation and contact metamorphism. Frequently these three characteristics of rock together result in non-linear deformability behaviour being observed. This idea is usefully summarised by the acronym DIANE, meaning Discontinuous, Inhomogeneous and Non Elastic, coined by Hudson & Harrison, (1997). Environmentally-based complicating factors exist: deformability behaviour of rock can be found to be appreciably stress, strain, temperature, and time dependent. Finally, the spatial scale at which one or more of the above factors is/are most prominent and the scale of the engineering can differ significantly. Resolution of such scale conflicts represents a further difficulty for the constitutive modeller of rock. Any

one of the factors mentioned in this paragraph can invalidate the inverse relation discussed in the preceding paragraph.

The most obvious example of stress dependent deformability behaviour of a rock or rock mass is the fact that above certain threshold levels of compressive stress the rock or rock mass will undergo state transitions from unbroken to broken states. The mechanism is complex, but the process is dominated by the initiation, growth and accumulation of tensile microcracks that erode the cohesive component of the strength of intact rock, and introduce a component of irrecoverable (*plastic*) strain. With increased loading or increased plastic strain, shear crack growth occurs and the frictional strength of existing and new fracture surfaces becomes mobilised as the cohesive strength component drops to its residual value (Martin and Chandler, 1994). At this stage, the frictional strength, of what is essentially broken material, is maximised with continuing (plastic) strain. Hajiabdolmajid, Kaiser and Martin (2002) present convincing evidence supporting this Cohesion-Weakening-Frictional-Strengthening (CWFS) model for the failure of intact rock. They draw upon i) results from micro-mechanical numerical modelling of laboratory tests, ii) large scale, accurate field observations from a mine-by experiment, iii) modelling and experimental test results from direct shear testing. This behaviour can vary considerably depending on the confining conditions of stress or strain on the body or rock under consideration.

The description above shows that the strength of rock is not something distinct from its deformability behaviour. Strength should be considered an integrated feature of rock deformability behaviour: it defines thresholds where the deformability processes change. It does not represent a threshold after which understanding and modelling of the rock is not required. Many rock mechanics applications in engineering practice have the deliberate objective of causing the rock to exceed such thresholds; rock fragmentation for excavation is an example, with the consequence that the new 'mode' of rock deformability behaviour must also be understood.

The notion of the stiffness of a body of rock itself implies two additional underlying factors: the extensive characteristics of scale and dimension, and the intensive property of a material modulus describing the relative proportions between increments of stress and strain of the material (for linear elastic materials this modulus is called the Young's modulus). Consequently, the idea of stiffness is distinct from the notions of strength or material modulii in that stiffness is a property that always applies to a specific (extensive) body, whereas strength and intrinsic, constitutive, behaviour applies to a specific (intensive) material. Stiffness can be interpreted as a parameter that describes the capacity of a body to bear forces, normalised by

displacements induced in the same body, under the same loads; stiff bodies produce less displacement than compliant bodies for the same applied loads.

Problems are always introduced when one attempts to measure the intrinsic deformability properties of a material because, paradoxically, it requires a finite body of the material to conduct the test. In practice this paradox is overcome by ensuring that the size and shape of every test body adheres to some standard ISRM Suggested Method (e.g. ISRM, 1979, 1983, 1999).

Other problems and paradoxes exist in rock testing. Although the quantities stress and strain are mathematically defined only at a point (inherently constitutive in character); whenever measurements of these same quantities are physically made, they are always conducted on bodies of rock of finite size.

In addition, there are state dependencies. In 'conventional' triaxial tests it is usual for the confining pressure to be held at a constant value whilst a constant axial velocity is imposed on the sample under servo-control. These conditions are not necessarily representative of those that an element of rock around a mine opening may be influenced by, following excavation (Brady and Brown, 1985).

The influence of loading and unloading is dependent on the physical state of the rock sample being tested, particularly the degree of damage it has already sustained throughout its geological history, expressed in terms of its *in-situ* microcrack density (see Homand-Etienne *et al.*, 1998 for a useful summary and constitutive formulation) or orientation (see Martin and Christiansson, 1990, for a practical example of effects). In the nearly linear, pre-peak portion of the complete stress strain curve, the stress path of unloading and loading will be close to the original loading path with negligible hysteresis. In the region of yield, the stress path deviates from its linear path and energy is dissipated as microfractures begin to form. Multiple load cycles in the yield region may cause migration of the hysteresis loops such that the sample fails without ever reaching peak strength.

Warwersik and Fairhurst (1970) examined the mechanisms of failure and fracture in the post–peak region, and as a result conducted a number of loading and unloading cycles in the post peak region. Brady and Brown (1985) noted several points about the behaviour of the samples.

i) On unloading and loading there is a degree of hysteresis.
ii) On reloading, the curve eventually joins that for a specimen in which the axial displacement continues monotonically with time.
iii) As displacement continues in the post peak region, the proportion of the total displacement that is irrecoverable increases.

iv) The apparent modulus of the rock which can be calculated from the slope of the reloading curve, decreases with post peak deformation and progressive fragmentation of the sample.

The rate of straining of rocks can also lead to additional complexities. In general, the greater the axial velocity that is applied to a specimen, the faster the load is applied and the peak strength becomes higher. The ISRM Commission (1979, 1983, 1999) suggests that for a standard uniaxial compressive strength test the "load on the specimen should be applied continuously at a constant stress rate such that failure will occur within 5–10 min of loading, alternatively the stress rate shall be within the limits of 0.5–1.0 MPa s^{-1}".

With the advent of servo-controlled testing machines, it is more common to use strain or displacement as the controlling variable. Brady and Brown (1985) suggest that the times to peak strength recommended by the ISRM Commission (1979,1983, 1999) correspond to axial strain rates in the order of 10^{-5}–$10^{-4}s^{-1}$.

When all the necessary considerations have been made, for the various paradoxes, the discontinuous, anisotropic, and inhomogenous character of rock, the difficulties in making *in situ* or laboratory measurements, and the scale effects, it is perhaps not surprising that rock engineering is a particularly challenging discipline. Irrespective of the complexities, understanding of the constitutive behaviour of a rock is the first step in finding a way to successfully engineer it in order to attain some engineering objective. This requires an appreciation of the points discussed above and is often manifest through the formulation of a mathematical model for the constitutive behaviour used in the engineering analyses. This prescription should be as complete as required by the objective of the rock engineering. It is suggested that in practice this requires knowing:

- the 'mode' of behaviour (as discussed above) of the rock before any engineering takes place and constitutive dependencies,
- the nature and magnitudes of any thresholds that modify or change the pre-existing mode of behaviour which are expected to be exceeded,
- the 'mode' of behaviour of the rock after any of the various thresholds identified have been exceeded.

The above list presents a deliberately general picture. In order to practically deal with complex rock constitutive behaviour for engineering projects, simplifications from this general picture are required. The most frequent simplification that is made in engineering practice (at least for 'first pass' or benchmark studies) is that of linear elasticity and thus the principle of superimposition can be taken to apply. In reality rock deforms non-linearly and non-elastically; however, over a restrictive range of

stress or strain magnitude, constitutive behaviour frequently approximates linear elasticity.

This is one reason why the subsequent treatment of linear elasticity with an ANN is useful. A second reason is that linear elasticity is well understood by all practitioners so that the results will be clear (hopefully). Finally, there is also a variety of benchmark analytical solutions with which the ANN constitutive model can be tested. However, ultimately the purpose of adopting the ANN approach to constitutive modelling of rock is:

- that they will be shown to accommodate both the general requirements of rock engineering listed above,
- they will be shown to respect the inherent complexity of the material, and;
- they do this within a simplifying conceptual framework.

4.3 Constitutive framework

4.3.1 Linear elasticity

In this short section, the main results of linear elasticity are presented. This is because the results will be used to synthesise a data set that simulates a triaxial compression test on a linearly elastic rock. ANNs will be trained with this data, in a first stage of development. The results here can be found in mainstream rock mechanics text books; readers familiar with linear elasticity can proceed to Section 4.3.2, if they wish.

For a general state of stress and an isotopic linear elastic material, the normal strain component in the x-direction within a Cartesian co-ordinate system is:

$$\varepsilon_x = \frac{1}{E}\left[\sigma_x - \nu\left(\sigma_y + \sigma_z\right)\right] \tag{4.1}$$

with

$$\varepsilon_y = \frac{1}{E}\left[\sigma_y - \nu\left(\sigma_z + \sigma_x\right)\right] \tag{4.2}$$

$$\varepsilon_z = \frac{1}{E}\left[\sigma_z - \nu\left(\sigma_x + \sigma_y\right)\right] \tag{4.3}$$

in the y and z directions respectively. E and ν are the Young's modulus and Poisson's ratio of the material respectively. The tensorial shear strain component in the xy plane is:

$$\varepsilon_{xy} = \frac{(1+\nu)}{E}\sigma_{xy} = \frac{1}{2G}\sigma_{xy} \tag{4.4}$$

with

$$\varepsilon_{yz} = \frac{(1+\nu)}{E}\sigma_{yz} = \frac{1}{2G}\sigma_{yz} \tag{4.5}$$

$$\varepsilon_{zx} = \frac{(1+\nu)}{E}\sigma_{zx} = \frac{1}{2G}\sigma_{zx}. \tag{4.6}$$

The shear modulus, G and the bulk modulus, K can both be expressed in terms of E and ν:

$$G = \frac{E}{2(1+\nu)} \tag{4.7}$$

$$K = \frac{E}{3(1-2\nu)}. \tag{4.8}$$

Adopting the pseudo-vector convention for arranging the stress and strain components then the constitutive relation expressing strain in terms of stress can be written using matrix notation:

$$\boldsymbol{\varepsilon} = \mathbf{S}\boldsymbol{\sigma} \tag{4.9}$$

where $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ are 6 x 1 strain and stress pseudo-vectors respectively and $\mathbf{S}$ is a 6 x 6 compliance matrix.

Lamé's constants, λ_1 and λ_2 are also useful in conveniently finding stress components from given strain components:

$$\begin{aligned}
\sigma_x &= \lambda_1\varepsilon_x + \lambda_2\varepsilon_y + \lambda_2\varepsilon_z \\
\sigma_y &= \lambda_2\varepsilon_x + \lambda_1\varepsilon_y + \lambda_2\varepsilon_z \\
\sigma_z &= \lambda_2\varepsilon_x + \lambda_2\varepsilon_y + \lambda_1\varepsilon_z
\end{aligned} \tag{4.10}$$

where:

$$\lambda_1 = K + \frac{4}{3}G = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \tag{4.11}$$

$$\lambda_2 = K - \frac{2}{3}G = \frac{\nu E}{(1+\nu)(1-2\nu)}. \tag{4.12}$$

Using matrix notation, the constitutive relation expressing stress in terms of strain is:

$$\boldsymbol{\sigma} = \mathbf{S}^{-1}\boldsymbol{\varepsilon} \tag{4.13}$$

where $\mathbf{S}^{-1}$, the inverse of the compliance matrix is known as the stiffness matrix.

Linear elastic constitutive behaviour also holds in incremental form, that is, either:

$$\Delta\boldsymbol{\varepsilon} = \mathbf{S}\,\Delta\boldsymbol{\sigma} \tag{4.14}$$

or

$$\Delta\boldsymbol{\sigma} = \mathbf{S}^{-1}\,\Delta\boldsymbol{\varepsilon}\,. \tag{4.15}$$

4.3.2 Triaxial test procedure

Specimens for rock triaxial compression tests are normally right cylinders with diameter around 54 mm and an aspect ratio in the range 2.5 to 3.0. Within the test cell, the flat ends of the specimen are subjected to axial loads via platens that are compliance matched so that the effect of shear stresses induced between platen and rock specimens are negligible. The curved face of the cylindrical specimen is subjected to a uniform pressure arising from a pressurised hydraulic fluid. The fluid does not wet the specimen due to the presence of a deformable membrane between the two.

In a typical triaxial compression test, after the specimen is prepared and mounted in the compression testing machine, firstly the specimen is loaded hydrostatically until the hydrostatic pressure reaches the planned confining pressure at which the test is to be conducted. After this stage has been reached, the specimen is then loaded deviatorically by increasing the axial load while keeping the confining pressure constant, via a servo-control mechanism.

When the objective of the compression test is to determine the deformability parameters of the specimen rather than its compressive strength value, the tests are normally conducted using strain or displacement control rather than stress or load control. The reason for this is so that the post-peak behaviour of the complete stress-strain curve can be obtained as this is important in many rock engineering applications such as rock breaking. Thus, in practice, both the initial hydrostatic and deviatoric loading phases are conducted using strain or displacement control: the strain or displacement rates are chosen carefully such that the rate of loading / application of stress lies within permitted ranges according to the ISRM Suggested Method.

It is frequently the case that the specimen will be taken through at least one deviatoric load-unload cycle before the sample reaches its peak strength in order that hysteresis effects can be identified.

As the specimen normally has compliance-matched platens, the flat ends of the specimen cylinder are principal stress planes. As the curved surface of the specimen is acted upon by a fluid (which by definition cannot sustain a shear stress) a state of principal stress also acts everywhere upon this surface. The loading conditions as well as the specimen geometry are axi-symmetric.

4.3.3 Simulation of initial hydrostatic compression

To simulate the case of hydrostatic compression the increments of strain in each of the Cartesian directions are set equal to one another, that is:

$$\Delta\varepsilon_x = \Delta\varepsilon_y = \Delta\varepsilon_z \tag{4.16}$$

The incremental form of equations (4.10) are then used to determine the corresponding increments of the stress components, which all have the same magnitude. Assuming that the corresponding rate of loading must also be maintained within 0.5 to 1.0 MPa/s then this places constraint on the magnitude of the strain increments during the hydrostatic loading phase because, for hydrostatic compression:

$$\Delta\varepsilon_z = \frac{\Delta\sigma_z}{3K}. \tag{4.17}$$

which implies:

$$\frac{0.5\times10^6}{3K} \le \Delta\varepsilon_z \le \frac{1.0\times10^6}{3K} \tag{4.18}$$

Within the simulator, the simulated specimen undergoes compression under constraints (4.16) and (4.18) until the hydrostatic stress becomes equal to the specified confining pressure.

4.3.4 Simulation of deviatoric compression

After this point, a period of deviatoric compression ensues, under the constraint that the confining pressure remains constant and the axial stress increases at a rate between 0.5 to 1.0 MPa/s. Taking the z-direction as the axial direction, then these constraints imply:

$$\frac{0.5\times10^6}{3E} \le \Delta\varepsilon_z \le \frac{1.0\times10^6}{3E} \tag{4.19}$$

If the axi-symmetric geometry is also considered then:

$$\Delta\varepsilon_x = \Delta\varepsilon_y = \frac{-\lambda_2}{\lambda_1+\lambda_2}\Delta\varepsilon_z. \tag{4.20}$$

4.4 Triaxial Compression Test Simulation

4.4.1 Simulation procedure

The initial state for the simulation is that all stress and strain components as well as their increments are set to zero (stress or strain free). In the initial hydrostatic loading, $\Delta\varepsilon_z$ is assigned a value under the constraint (4.18) that is used to define $\Delta\varepsilon_x$ and $\Delta\varepsilon_y$ using (4.16). With strain component increments defined, stress component increments can be computed using (4.10). Stress and strain components are found through accumulating the stress and strain component increments. Hydrostatic loading

ceases when the magnitude of the stress components becomes equal to the specified confining pressure.

During deviatoric loading $\Delta\varepsilon_z$ is assigned a value under the constraint (4.19) that is used to define $\Delta\varepsilon_x$ and $\Delta\varepsilon_y$ using (4.20). With strain component increments defined, stress component increments can be computed using (4.10). Stress and strain components are found through accumulating the stress and strain component increments. Deviatoric loading ceases when the magnitude of the axial stress component becomes equal to a specified threshold stress.

The simulated specimen is then unloaded, firstly deviatorically until the hydrostatic state of stress at the confining pressure is achieved, and then hydrostatically until the stress and strain free state is achieved.

4.4.2 Modelling experimental variance

In practice one would expect some variance in the data gathered for the strain rate as the control servo-loop continually readjusts the nominal strain rate to match the physical rate of strain of a real specimen under test (this process can have a frequency of 10 kHz in practice).

A beta distribution is used to model this variance within the simulation. The advantages of using this type of distribution are twofold. Firstly a beta probability distribution function does not have infinite tails to the distribution; thus, there is zero probability of a random value being selected that is outside defined limits for the distribution. Secondly, the beta distribution has two shaping parameters that enable the specific form of the distribution to be readily adjusted to any desired form. If the shaping parameters are set equal, the resulting distribution is symmetric. The magnitude of the shaping parameters set determine the spread of the distribution between its defined limits.

The simulation is prescribed using the Visual BASIC programming language within the MS Excel environment. This is advantageous as MS Excel already has a beta distribution implementation within it, as well as an inverse beta distribution function. Within the Visual BASIC code for this numerical experiment (which is presented in Appendix A) the syntax:

```
dezz = BetaInv(Rnd(), shape, shape, 500000 / (3 * K), 1000000 / (3 * K))
```

assigns a random value for $\Delta\varepsilon_z$ selected from the range (4.18) with the most probable value being $0.75\times10^6/3K$. A value of shape = 10000 produces the most peaked distribution such that a value of $\Delta\varepsilon_z = 0.75\times10^6/3K$ is almost certain. A value of shape = 1 produces a uniform distribution between the limits given in (4.18). A similar approach is used to set the axial strain rate for the deviatoric loading phase

where the limits are set using (4.19). In all the simulations reported in the following, the numerical values of the 2 shape parameters used in the calls to the BetaInv function were set equal.

Within the simulator, when this experimental variance is modelled, the random seed of the random number generator is controlled manually to guarantee that the same sequence of pseudo-random numbers is only ever used when specified.

4.4.3 Triaxial test schedule

In the example discussed, synthetic triaxial compression test data on a rock with Young's modulus 40 GPa and Poisson ratio 0.25 is produced for 4 distinct compression tests each 'conducted' at confining pressures of: 0, 5, 10 and 30 MPa. After pre-processing, these data are used to train an ANN. Further simulated triaxial compression tests have been 'conducted' at confining pressures of 20 and 50 MPa for use as validation data sets.

4.4.4 Plane strain 'triaxial' test schedule

It will be shown in Chapter 5 how the trained ANN model can be integrated into geomechanics numerical codes. The code chosen for this demonstration is *FLAC*^{2D} (Itasca Consulting Group, 1993; Itasca Consulting Corp., 1995). *FLAC*^{2D} is a 2D code subject to the additional constraints of either plane strain or plane stress. It is appropriate to check that the ANN trained with data representing triaxial compression (as obtained experimentally) can adequately reproduce the expected behaviour reflecting the additional constraint of compression subject to plane strain or plane stress (as required when used within *FLAC*). A third validation data set is produced that simulates a compression test on a specimen infinitely long in the direction perpendicular to the direction of deviatoric loading.

Considering the xz plane and again taking the z-direction as the direction of loading, plane strain requires the strain increment in the out-of-plane direction to be zero $\Delta\varepsilon_y = 0$; constant confining pressure (here, assumed to act only in the x direction) means that $\Delta\sigma_x = 0$. Thus:

$$\Delta\sigma_x = \lambda_1\Delta\varepsilon_x + \lambda_2\Delta\varepsilon_y + \lambda_2\Delta\varepsilon_z = 0 \tag{4.21}$$

yielding immediately:

$$\Delta\varepsilon_x = \frac{-\lambda_2}{\lambda_1}\Delta\varepsilon_z = \frac{-\nu}{1-\nu}\Delta\varepsilon_z \tag{4.22}$$

after substitution of Lamé's constants. Under these conditions:

$$\Delta\sigma_z = \frac{E}{(1+\nu)(1-\nu)}\Delta\varepsilon_z \tag{4.23}$$

and

$$\Delta\sigma_y = \frac{Ev}{(1+v)(1-v)}\Delta\varepsilon_z . \tag{4.24}$$

The ISRM constraint on stress rate is satisfied if:

$$\frac{0.5\times10^6\,(1+v)(1-v)}{E} \le \Delta\varepsilon_z \le \frac{1.0\times10^6\,(1+v)(1-v)}{E} \tag{4.25}$$

because $0 \le v \le 0.5$. The plane strain 'triaxial' compression test was simulated for confining pressures of 0, 7.5 and 40 MPa, that is: $\sigma_x = 0,7.5,40\,\text{MPa}$.

In a test schedule as prescribed within this section, the simulated sample will experience a monotonically increasing magnitude of stress in the z-direction, the stress component in the x-direction will be kept constant, and the stress component in the y-direction will also increase monotonically due to the condition of lateral constraint applying in this direction. The stress state this simulated sample experiences is thus more properly termed *polyaxial* rather than *triaxial*. All normal stress components are still principal stress components, but the non-major principal stress components are no longer equal.

4.5 Data pre-processing

4.5.1 Normalisation procedure

The simulated triaxial test data are to be used to train an ANN. The first stage in preparing the data for the training process is to normalise the input and output data into the range –0.45 to 0.45. A normalisation range of –0.5 to 0.5 is possible but makes for a rather challenging ANN training phase as the latter limits are at the extremes of activation values for the sigmoidal activation function of the network microstructure.

For a generally useful ANN constitutive model of rock, the range of stress and strain increment values should span those that are physically expected. The range of stress increments to be expected is defined by the ISRM Suggested Method to be between 0.5 and 1.0 MPa/s. The maximum corresponding range for the strain increments depends on the degree of compliance of the materials concerned. Very stiff rocks do not place any constraint on the range of maximum strain increments as comparatively small strain increments lead to stress increments within the target range. The most compliant material that would still be called a "rock" would have a Young's modulus around 1.0 GPa, a value also appropriate to stiff soils. For the case of deviatoric compression, Poisson's ratio has an effect on the maximum strain increments expected of a rock under test through equations (4.20) and (4.11) with

(4.12). Conservation of energy considerations (see Jaeger and Cook, 1979) mean that for isotropic solids (as considered in this case), $0 \leq \nu \leq 0.5$.

These considerations taken together provide a maximum strain increment value of 0.001 / s across all rocks with Young's moduli between 1 and 100 GPa and Poisson's ratio between 0 and 0.5. The minimum strain increment value is –0.001 / s, corresponding to unloading across the same theoretical specimens. These extreme values are used to normalise the strain increments of the synthesised data in preparation for the ANN training task.

The normalisation transformations are linear for both strain and stress increments. g_ε and h_ε are gain and offset parameters for the strain normalisation mapping:

$$p_i = g_\varepsilon \Delta\varepsilon_i + h_\varepsilon \tag{4.26}$$

where p_i is the ith input quantity to the ANN. $i = \{1,2,3\}$ corresponding to Cartesian directions x, y and z respectively. g_σ and h_σ are gain and offset parameters for the stress normalisation mapping:

$$t_i = g_\sigma \Delta\sigma_i + h_\sigma \tag{4.27}$$

Here, t_i denotes the i^{th} target output for the ANN. Denormalisation mappings are needed to examine the results of the training and validation processes of the ANN as well as for operational use:

$$\Delta\sigma_i = g_\sigma' o_i + h_\sigma' \tag{4.28}$$

where $'$ denotes a denormalisation parameter and o_i is the i^{th} output of the ANN. The transformation parameters are summarised in the following tables.

Table 4.1 : Summary of (de) normalisation parameters for strain increments.

$\Delta\varepsilon_i$	p_i	g_ε	h_ε	g_ε'	h_ε'
0.001	0.45	450	0	0.00222	0
-0.001	-0.45				

Table 4.2 : Summary of (de) normalisation parameters for stress increments.

$\Delta\sigma_i$	t_i	g_σ	h_σ	g_σ'	h_σ'
1.00×10^6	0.45	4.50×10^{-7}	0	2.22×10^6	0
-1.00×10^6	-0.45				

4.5.2 Cyclic permutation of inputs and outputs

Thus far, the stress and strain components in the z-direction have been taken as those acting axially. Without correction, the assumption implied in using the ANN constitutive model is that the direction of deviatoric compression is always in the z-direction, which is clearly unrealistic. To correct this and eliminate the assumption, in preparation of the data file ready for training, input and output values have been cyclically permuted twice:

Permutation 1:

$$\text{Inputs: } \Delta\varepsilon_x \rightarrow \Delta\varepsilon_y ; \Delta\varepsilon_y \rightarrow \Delta\varepsilon_z ; \Delta\varepsilon_z \rightarrow \Delta\varepsilon_x$$

$$\text{Outputs } \Delta\sigma_x \rightarrow \Delta\sigma_y ; \Delta\sigma_y \rightarrow \Delta\sigma_z ; \Delta\sigma_z \rightarrow \Delta\sigma_x$$

Permutation 2:

$$\text{Inputs: } \Delta\varepsilon_y \rightarrow \Delta\varepsilon_z ; \Delta\varepsilon_z \rightarrow \Delta\varepsilon_x ; \Delta\varepsilon_x \rightarrow \Delta\varepsilon_y$$

$$\text{Outputs } \Delta\sigma_y \rightarrow \Delta\sigma_z ; \Delta\sigma_z \rightarrow \Delta\sigma_x ; \Delta\sigma_x \rightarrow \Delta\sigma_y$$

Table 4.3 illustrates this concept numerically. It shows an extract from a training data file simulating the constitutive behaviour of rock under the deviatoric compression stage of a triaxial compression test (Young's modulus of 40 GPa and Poisson's ratio 0.25). Training vectors 45 through 47 are shown. Applying permutation 1 to vector 45 results in vector 46. Applying permutation 2 to vector 46 results in vector 47.

Table 4.3: Extract of training data file illustrating permutation of input and output quantities in training vectors.

Training Vector	Inputs dezz	dexx	deyy	Outputs dszz (MPa)	dsxx (MPa)	dsyy (MPa)
...	...	...	...	...	...	...
44	...	...	...	...	...	...
45	1.875×10^{-5}	-4.688×10^{-6}	-4.688×10^{-6}	7.500×10^{5}	0.000×10^{0}	0.000×10^{0}
46	-4.688×10^{-6}	1.875×10^{-5}	-4.688×10^{-6}	0.000×10^{0}	7.500×10^{5}	0.000×10^{0}
47	-4.688×10^{-6}	-4.688×10^{-6}	1.875×10^{-5}	0.000×10^{0}	0.000×10^{0}	7.500×10^{5}
48	...	...	...	...	...	...
...	...	...	...	...	...	...

Note that Table 4.3 shows input and target output quantities that have not been normalised. For clarity, the same information is reproduced after the normalisation process in the following table.

Table 4.4: Extract of training data file illustrating permutation of input and output quantities in training vectors after normalisation with (4.26) and (4.27).

Training Vector	Inputs dezz	dexx	deyy	Outputs dszz (MPa)	dsxx (MPa)	dsyy (MPa)
..	..	..	..	..	..	..
44	..	..	..	..	..	..
45	0.00844	-0.00211	-0.00211	0.33750	0.00000	0.00000
46	-0.00211	0.00844	-0.00211	0.00000	0.33750	0.00000
47	-0.00211	-0.00211	0.00844	0.00000	0.00000	0.33750
48	..	..	..	..	..	..
..	..	..	..	..	..	..

4.6 ANN Training Log

A multilayered perceptron (MLP) was used to learn the simulated data. The MLP comprised three layers: one input layer with three inputs, one hidden layer and one output layer with three outputs. Initially, the number of units in the hidden layer was set at five. The sigmoidal activation functions of the network scale to the range –0.5 to 0.5. Initial values of the network weights were assigned values between –0.3 and 0.3 in all cases. The ANSIM (SAIC, 1992) code was used to train the data.

Initial explorations in training the ANN determined that i) ANNs with two or more hidden layers did not converge within 10000 training iterations and ii) for ANNs with one hidden layer suitable values of the momentum parameter and the learning parameter were 0.97 and 0.00025 respectively. The initial explorations also revealed that all ANN weight changes between one presentation to the next were below 0.00001 after 10000 iterations of the training rule. A weight change of 0.00001 is below the precision level at which the ANN computations are conducted; thus, the ANN training can be said to have asymptotically converged at around 10000 presentations of the training data set. At this stage the maximum error at output was 0.045 and the RMS error across all training patterns was 0.014.

To determine the robustness of ANN training to changes in network structure, the training process was repeated using the same learning parameters and input data while increasing the number of units on the hidden layer of the network, to a maximum of 20. The table below summarises the training behaviour of each of these ANNs.

Table 4.5. Training performance after 10000 iterations for 5 ANNs trained with simulated triaxial compression test data for a linear elastic rock with E=40GPa and ν=0.25.

Perceptron Structure	RMS Error	Maximum error at output
3-3-3	0.015094	0.047737
3-5-3	0.015026	0.045610
3-6-3	0.015034	0.045866
3-7-3	0.014991	0.045653
3-8-3	0.015003	0.045804
3-9-3	0.014977	0.045525
3-20-3	0.014948	0.045290

Table 4.5 shows that for the data used, training performance was largely independent of the number of artificial neurons in the hidden layer of the ANN. The RMS error indicates the overall performance of the ANN in learning the data. A RMS error value of 0.015 means that, over the training patterns used, the typical error in estimation is around 1.5% of the true value because the output quantities are normalised into the range –0.5 to 0.5. A maximum error at output of 0.045 means that over all input patterns, the largest discrepancy between target output and ANN output is 4.5% of the target value.

4.7 Training and Validation Results

4.7.1 Method of graphical presentation

After the training process was complete, the performance of each trained ANN was investigated in detail. For each of the output quantities, a scatter plot is produced showing the target output against the actual output of the ANN. One point on the scatter plot corresponds to a single training record contained within the data set. The results of perfect training (with RMS error = 0 and maximum output error = 0) would lie on a straight line through the origin, inclined at 45° with respect to the axes of the plot. Poor training performance would be represented by significant deviations from the ideal line, with poorer performance having larger deviations.

4.7.2 Progression of the ANN training performance

Figure 4.1 shows the training performance of the 3-6-3 ANN, of Table 4.5, at various stages during the training process. In each of the six scatter plots, the horizontal axis is the value of the target normalised $\Delta\sigma_z$ and the vertical axis is the value of the ANN output normalised $\Delta\sigma_z$.

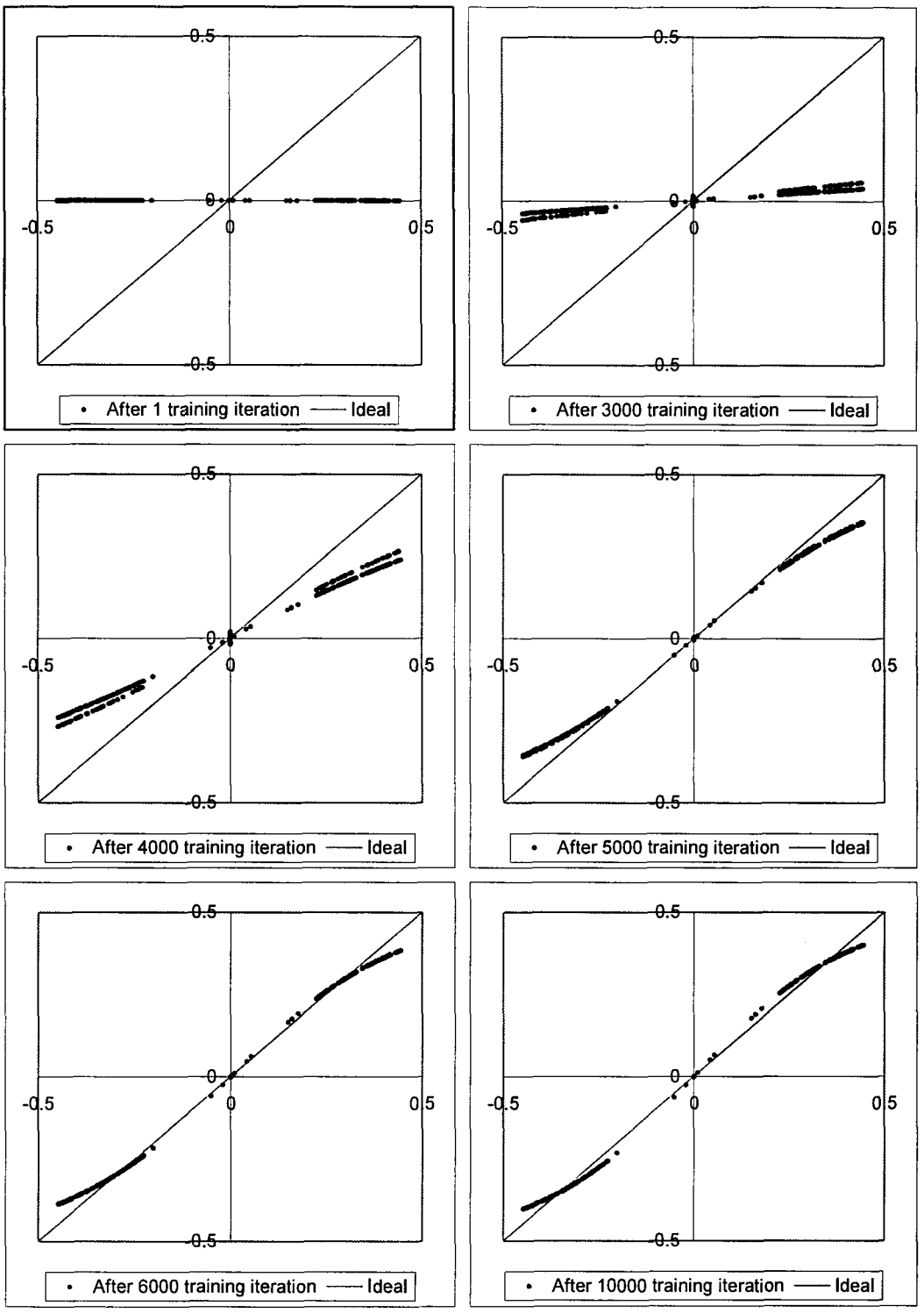


Figure 4.1 : Scatter plots of target normalised $\Delta\sigma_z$ versus ANN output normalised $\Delta\sigma_z$ for the 3-6-3 ANN of Table 4.5. The x-axis of each plot is the target output; the y –axis of each plot is the actual output. The diagonal line presented in each plot shows the ideal 'locus' on which the crosses should lie, for perfect ANN training. The six plots illustrate the quality of ANN recall of the training data, as training iterations proceed.

With randomly initialised ANN weights and 1 iteration of the training procedure, the output of the ANN is independent of the inputs to the ANN and consequently, the output values are constant. As the ANN is progressively trained so the ANN outputs become closer to their targets. From the plots, it appears that relatively early in the ANN training (3000 and 4000 iterations), the ANN cannot effectively discriminate between the cases of hydrostatic loading and unloading as is shown by the two linear clusters. After 5000 iterations, the ANN is able to discriminate between these two cases as the results have coalesced onto a single curve of roughly sigmoidal form. The linear portion of the sigmoid is approximately parallel to the ideal line; the largest errors correspond to situations with intermediate and large target normalised $\Delta\sigma_z$.

Between training iterations 5000 and 6000, the training algorithm appears to 'work on' the errors for intermediate and large values of normalised $\Delta\sigma_z$ and is successful in reducing the former to close to the ideal. From 6000 iterations to the stage of final convergence of the weights of the ANN (at 10000 iterations), the 'tactic' that the generalised delta rule appears to have exploited is to cause this curve to shift right slightly so that the average accuracy of the ANN predictions is improved. This assertion is supported by the fact that the two most densely clustered regions of results on the scatter point straddle the ideal line for the case when training iterations = 10000.

It is interesting to note that the ANN appears to have learned the data in phases. In terms of the scatter plot presented and for the data used, these phases could be described as: firstly, a rotation of linear clusters about the origin; secondly, a coalescence of behaviour (for the hydrostatic and triaxial training pairs in the data); thirdly, a transition to a sigmoidal form; and, finally, the sigmoid's translation to the lowest RMS error.

The magnitudes of the errors monitored during the training process are presented in Figure 4.2 and Table 4.6. For the first 2000 – 3000 iterations, both the RMS error and the maximum output error remain approximately constant, although, by 3000 iterations, both have dropped by a small amount. The period between 3000 to 4000 iterations is the period with the maximum rate of learning. From 4000 iterations onward, the rate of learning reduces gradually. Between 6000 to 10000 iterations, the learning rate is low and by 10000 iterations, weight changes are below the precision with which the ANN arithmetic is executed within the computer code (single precision). No better training performance can be obtained after 10000 iterations and the training process can be said to be complete.

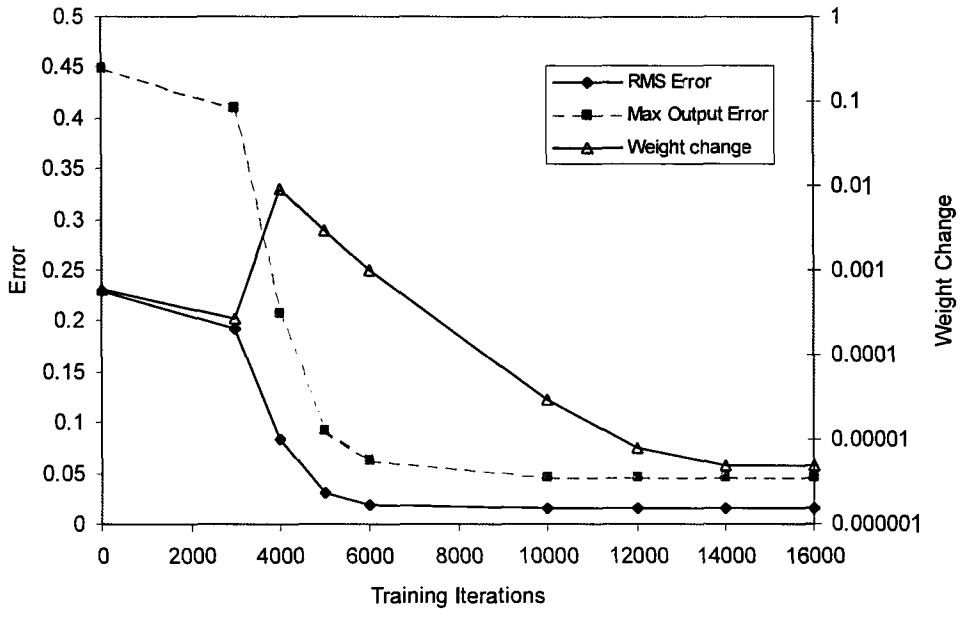


Figure 4.2 : Performance of the 3-6-3 ANN of Table 4.5 during the training process.

Table 4.6 : Performance of the 3-6-3 ANN of Table 4.5 during the training process.

Number of Training Iterations	Root mean square error	Maximum output error	Largest weight change
1	0.230012	0.448312	0.000588
3000	0.191563	0.409760	0.000271
4000	0.083292	0.207538	0.009014
5000	0.030963	0.092546	0.002919
6000	0.018139	0.062183	0.001003
10000	0.015034	0.045866	0.000030
12000	0.015029	0.045720	0.000008
14000	0.015022	0.045434	0.000005
16000	0.015018	0.045414	0.000005

4.7.3 Performance of the trained ANN.

The ANN has three output quantities: normalised $\Delta\sigma_x$, normalised $\Delta\sigma_y$ and normalised $\Delta\sigma_z$. Scatter plots for each of these output quantities are presented in Figure 4.3 a), b) and c) below, for the case after 10000 training iterations. These three plots show that the ANN has learned the linear elastic behaviour of a rock with Young's modulus of 40 GPa and Poisson's ratio 0.25 with data simulating 4 triaxial compression tests on that rock. Clearly, the degree to which this has been done is not perfect, but the mean error is 1.5% of the true value and the maximum error is less than 5%.

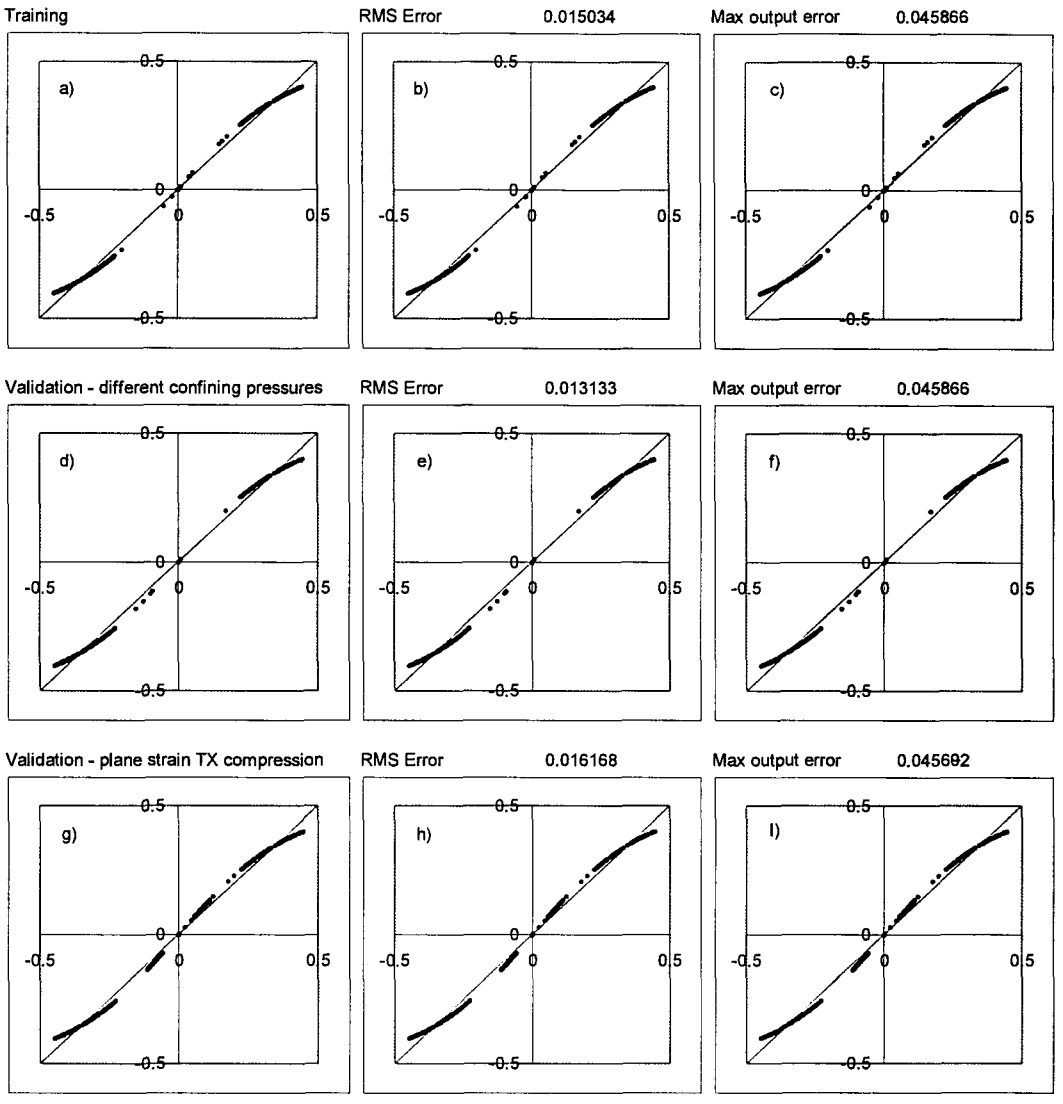


Figure 4.3. Training and validation performance of the 3-6-3 ANN of Table 4.5. Top row: training results a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$. Middle row, validation results d) $=\Delta\sigma_x$, e) $=\Delta\sigma_y$, f) $=\Delta\sigma_z$. Bottom row, validation results with plane strain simulation, g) $=\Delta\sigma_x$, h) $=\Delta\sigma_y$, i) $=\Delta\sigma_z$. The x-axis of each plot is the target output; the y –axis of each plot is the actual output. The diagonal line presented in each plot shows the ideal 'locus' on which the crosses should lie, for perfect ANN training.

Figure 4.3 d) to f) test the capability of the ANN to interpolate within, and extrapolate outside the data set with which it had been trained. The data used in this validation task were synthesised from simulated compression tests conducted at confining pressures of 20 MPa and 50 MPa that were absent from the training data set. It can be seen from Figure 4.3 d) to f) that there is no significant loss of accuracy in the interpolative or extrapolative task; RMS and maximum output errors are actually below those associated with the training data set.

As indicated earlier, a more robust test of the ANN capability is to assess its performance with simulated data from a compression test completely unlike the synthetic data from triaxial compression test simulations. The ANN was provided with input strain increments arising from the plane strain 'triaxial' (strictly, polyaxial) compression test discussed in Section 4.4.4. The correct outputs are plotted against the ANN estimates in Figures 4.3 g) to i). Again it can be seen that the quality of estimation demonstrated is at least equal to that of the ANN when it is tested with the data with which it was trained. Most importantly, the ANN performance under this test demonstrates that it has not only simply learnt the data with which it was trained, it has also managed to abstract general linear elastic behaviour. This is because the data with which the ANN was tested reflects additional constraint on linear elastic behaviour that was absent from the training data set.

4.7.4 Performance of the trained ANN with simulated data from different rock types

The trained ANN has produced valid results for all tests to which it has been subjected thus far. Triaxial compression test simulations for a rock with Young's modulus = 1 GPa, Poisson's ratio = 0.45 and confining pressures of 0, 5, 10 and 30 MPa have been conducted in order to produce another test data set for which the trained ANN should fail. The results of the trial are shown in Figure 4.4 .

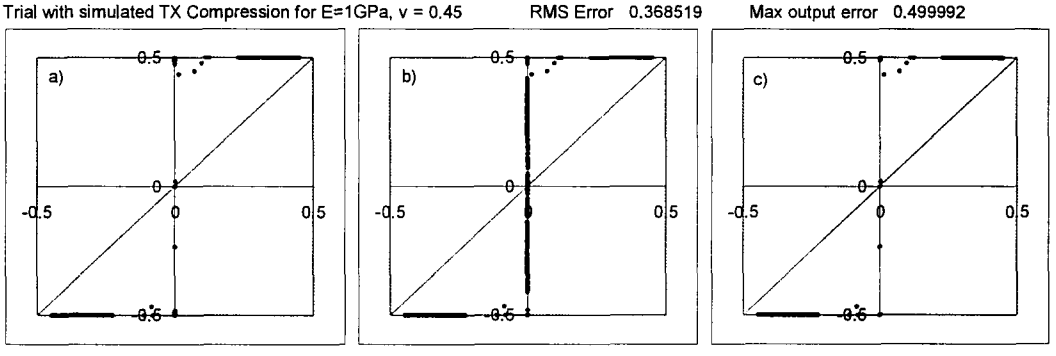


Figure 4.4 : Results of testing the 3-6-3 ANN of Table 4.5 with simulated triaxial compression test data for a rock with Young's modulus = 1GPa and Poisson ratio 0.45. a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$ (all normalised). Meaning of axes: as for Figures 4.1 and 4.3.

It can be seen from these scatter plots that for these data, the ANN has performed extremely poorly, which is to be expected. The elastic properties represented in the data set used to test the network are different from the elastic properties of the data used to train the ANN. In this test, the test data represent one of the extreme materials that the system presented thus far can accommodate (through the normalisation parameters). The test data represent a compliant material.

At the other extreme that can be accommodated is a material over twice as stiff as the E=40 GPa, ν=0.25 material used to train the ANN. Triaxial compression test simulations with a material having E=90 GPa and ν=0.05 were used to test the 3-6-3 ANN from Table 4.5. The results of the test are shown in Figure 4.5 below. Again, the expected result is that the ANN should fail the test because the data with which it has been tested apply to a different rock type than the data with which it has been trained. The scatter plots of Figure 4.5 indicate large errors, but the overall performance of the trained ANN in representing the behaviour of the 'foreign' data for this stiffer rock is better than for the case of the compliant rock.

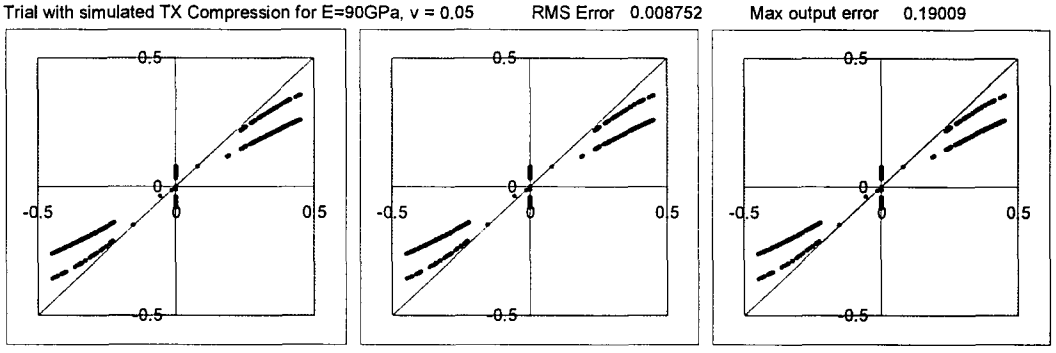


Figure 4.5: Results of testing the 3-6-3 ANN of Table 4.5 with simulated triaxial compression test data for a rock with Young's modulus = 90GPa and Poisson ratio 0.05. a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$ (all normalised). Meaning of axes: as for Figures 4.1 and 4.3.

The results presented above in Figures 4.4 and 4.5 represent only two from an exhaustive trial of the 3-6-3 ANN from Table 4.5. The performance of the ANN against simulated triaxial compression test data from a broad variety of rocks is shown in Table 4.7 below. Once again, the reader is reminded that there should only be good performance (low RMS error and low maximum output error) for the case where E=40 GPa, ν=0.25. From Table 4.7, several observations can be made:

- That only for the case of E=40 GPa, ν=0.25 are the RMS error and maximum output error both less than 5%.
- Rocks with higher Young's modulus than the target rock have lower RMS error than rocks with lower Young's modulus.
- For all rocks with E<90 GPa, the lowest RMS error arises when the Poisson's ratio of the test rock is 0.25 (the same as the 'target' rock).
- For rocks with higher Young's modulus than the target rock, the maximum output error reduces as the Poisson's ratio reduces.
- For rocks with lower Young's modulus than the target rock, the maximum output error increases as the Poisson's ratio reduces.

Table 4.7 ANN performance when presented with simulated triaxial compression test data for rock with varying Young's modulus and Poisson ratio. (Performance of 3-5-3 ANN of Table 4.5 shown in bold type).

Young's Modulus (GPa)	Poisson's Ratio	Shear Modulus (GPa)	Bulk Modulus (GPa)	RMS Error	Max output Error
90	0.45	31.0	300.0	0.157484	0.399111
90	0.25	36.0	60.0	0.107343	0.220802
90	0.05	42.9	33.3	0.087515	0.190090
60	0.45	20.7	200.0	0.145056	0.374690
60	0.25	24.0	40.0	0.059038	0.134566
60	0.05	28.6	22.2	0.077791	0.118312
40	0.45	13.8	133.3	0.142862	0.338754
40	**0.25**	**16.0**	**26.7**	**0.015028**	**0.045958**
40	0.05	19.0	14.8	0.112589	0.172792
20	0.45	6.9	66.7	0.203551	0.036962
20	0.25	8.0	13.3	0.084514	0.179056
20	0.05	9.5	7.4	0.211747	0.307194
1	0.45	0.3	3.3	0.368519	0.499992
1	0.25	0.4	0.7	0.305017	0.499949
1	0.05	0.5	0.4	0.391426	0.499990

4.8 Effect of varying the normalisation parameters

The overall performance of the ANN that has been investigated in the previous Section is good. The training errors are low and the ANN has demonstrated generalisation capability of a similar level to its performance with the training data set. The generalisation behaviour under conditions of simulated plane strain demonstrates that the ANN is not only able to reproduce the data on which it has been trained, but that it can also encapsulate linear elastic behaviour.

Despite these capabilities, the training and generalisation performance, measured by both RMS and maximum output errors, is still lower than that which may be considered acceptable for use in practical engineering situations.

There are two main reasons causing the level of inaccuracy of the asymptotically trained ANN. Firstly, at the point of asymptotic learning (from Table 4.6, around 14000 training iterations) the magnitude of the maximum weight change is low and of the same order as the precision at which the arithmetic of the ANN is computed. Subsequent training iterations cannot improve the training performance of the ANN. This has been checked by continuing the training iterations to 30000 for the 3-6-3 network of Table 4.5.

Secondly, it is apparent from Figure 4.3 a) to i), and Figures 4.4 and 4.5 that there is a degree of non-linearity in the response of the outputs of the ANN that is not expected given that the behaviour it is being trained to reproduce is linear. Figures 4.3 e) and f) show that in the final approach to asymptotic learning, the training algorithm

introduces a shift in the output response rather than causing the remaining non-linearity to be eliminated.

There are two reasons why the degree of non-linearity persists in the output behaviour of the ANN. The first reason is because the form of the output activation function producing the output is itself sigmoidal in form. Secondly, the normalisation parameters used cause the linear elastic behaviour of the simulated specimen to be scaled to span the range –0.45 to 0.45 when normalised. This output range, illustrated in Figure 4.6 with the artificial neuron activation function, covers a large proportion of the latter's full-scale range (-0.5 to 0.5). The 'solver' procedure available with MS Excel was used to find the gradient of the line that minimised the RMS error between the sigmoidal activation function and its linear approximation in the range where its output ranged between –0.45 and 0.45.

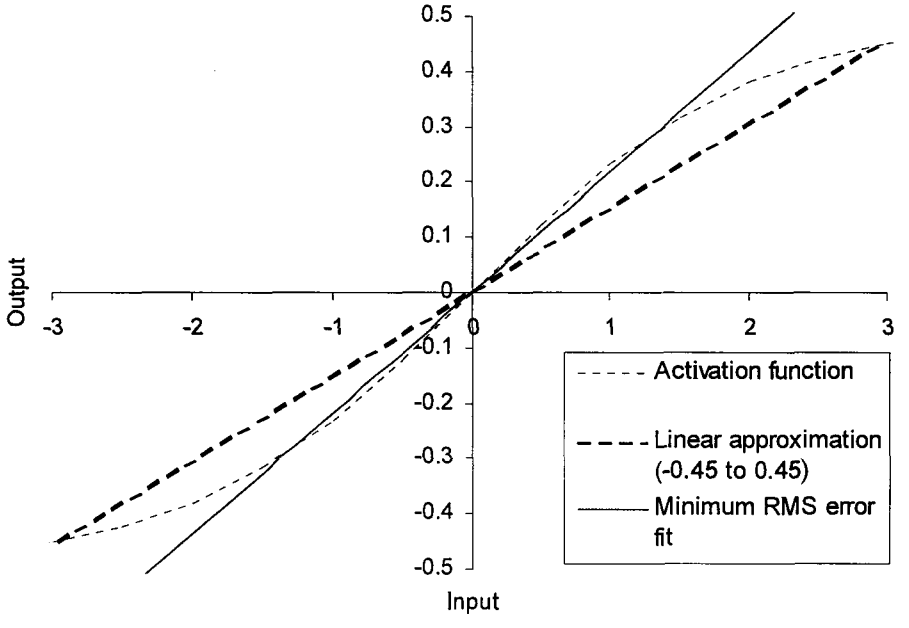


Figure 4.6: Artificial neuron activation function, linear approximation and best fit.

The minimum RMS error found was 0.0196 which can be compared to the RMS error of the ANN while training (0.015, asymptotic learning case). In making this comparison it is important to note that in the fitting procedure indicated immediately above, the samples on which the squared errors were computed were evenly distributed across the input range. In the case of the ANN in training, the RMS is calculated with samples that tend to cluster toward 'the ends' of the activation function.

Thus, to improve the performance of the ANN in learning linear data, the range of ANN outputs can be restricted so that the sigmoidal approximation of linearity improves. This is achieved through recalculation of the ANN normalisation parameters. Simply to demonstrate the effectiveness of this measure, which is only

appropriate to linear relations in training data, the training and validation procedures were repeated with the synthesised data sets with the normalised stress increment values constrained to the range (-0.3 to 0.3). The MS Solver best fit line indicated an RMS training error of around 0.005. After 10000 training iterations with the renormalised training data set, the RMS error of the ANN was 0.004359 and its maximum output error was 0.014196. Asymptotic error values found, when the maximum weight change was 0.000005, were 0.004313 and 0.013706 respectively.

Validation testing with simulated triaxial compression test data at intermediate confining pressures and confining pressures outside the training data set produced RMS error and maximum output error of 0.005371 and 0.014195 respectively. Validation using a plane strain 'triaxial' compression tests produced RMS error and maximum output error of 0.005090 and 0.014195 respectively. The performance under this test is illustrated in Figure 4.7 for comparison with Figures 4.3 g), h) and i) presented earlier.

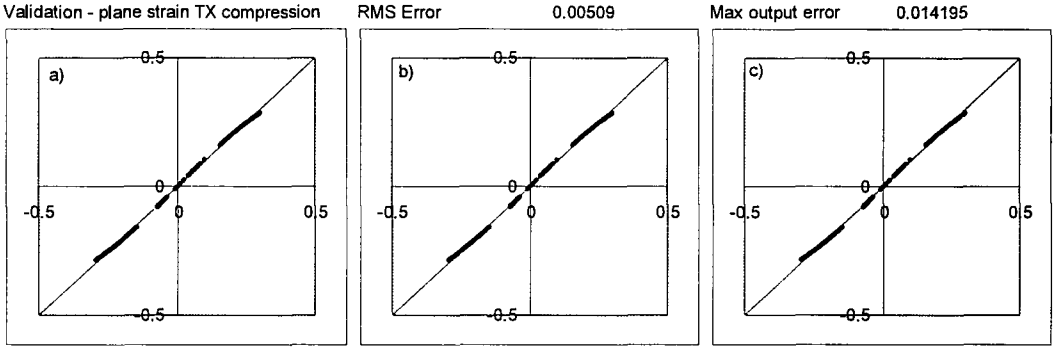


Figure 4.7. Validation performance of a 3-6-3 ANN under plane strain constraint a) $=\Delta\sigma_x$, b) $=\Delta\sigma_y$, c) $=\Delta\sigma_z$. Normalised output values confined to the range –0.3 to 0.3. Meaning of axes: as for Figures 4.1 and 4.3.

Clearly, applying the measure of reducing the allowable output range for normalised parameters had the desired effect in that ANN performance over all key measures improved approximately threefold. On average, the ANN produced is accurate to 0.5% and the largest error obtained no greater than 1.4%. Further improvements in training and validation performance could be obtained using an identical scheme with the ANN code being recompiled for double precision arithmetic.

4.9 Effect of varying the type and number of simulated tests used to synthesise data sets

The cases of uniaxial compression and hydrostatic compression are the two extreme cases of compression that an isotropic rock can be subjected to; triaxial compression and polyaxial compression can be viewed as intermediate cases. Before leaving the

topic of training an ANN to learn isotropic, linear elastic behaviour, it was considered useful to know whether the performance of a given rock type under uniaxial and hydrostatic compression alone was sufficient to allow the ANN to encapsulate the behaviour under more general conditions. The motivation behind this was that uniaxial and hydrostatic compression tests can be conducted in the laboratory with less specialised equipment than is necessary for a triaxial compression test.

An ANN was trained with synthesised data from one uniaxial compression test on a simulated specimen with E = 40 GPa and ν = 0.25 and one hydrostatic compression test. The maximum stress applied in the simulated tests was 75 MPa for uniaxial and hydrostatic cases. Learning and momentum parameters were specified at 0.00025 and 0.97 as for all previous ANNs with a 3-6-3 ANN mesostructure.

Under these conditions, the RMS error and maximum output error were 0.005293 and 0.013048 respectively, indicating training performance similar to the ANN in the previous Section trained with data from an increased number of tests. Validation testing of this ANN using the same triaxial compression validation data (which assesses interpolative and extrapolative capability) produced a RMS error of 0.005645 and a maximum output error of 0.013202. Validation with the plane strain compression test data also used in the previous Section produced a RMS error of 0.004590 and a maximum output error of 0.013202.

The results confirm the idea that there is sufficient information within the results of uniaxial and hydrostatic compression tests on linearly elastic rock specimens to train an ANN. The ANN effectively interpolates between the extreme loading conditions as well as encapsulates the linear elastic behaviour.

Following on from this, ANNs were trained using data from: one simulated uniaxial compression test only, one simulated hydrostatic compression test only, and one simulated triaxial compression test only (with confining pressure 10 MPa). After the ANN training phase, each of these networks were validated with unseen data as before. The results are summarised in Table 4.8 below which shows that all ANNs produced with the exception of that trained with simulated hydrostatic compression data, perform as robustly as the ANN produced in the base case scenario from Section 4.6.

This is perhaps surprising, but should not be unexpected. Conventional treatment of uniaxial compression test data from a single laboratory test allows the determination of all the elastic parameters of the material. For a perfectly linear elastic material, which is the case being modelled here, there is as much information on the deformability behaviour of the material in a triaxial compression test as is in a uniaxial compression test. The elastic parameters of a material can equally be determined from the results of a triaxial compression test, for an isotropic linear

elastic material. There is less information on the deformability behaviour of a material in a hydrostatic compression test due to symmetry between the input and output quantities.

Table 4.8 : Summary of ANN training and validation results when number and type of simulated tests used to train the ANN are varied. (UX = uniaxial, TX = triaxial, HS = hydrostatic)

Loading case represented in training data set	ANN Performance					
	Training		Triaxial validation		Plane strain validation	
	RMS Error	Max. Out.Error	RMS Error	Max. Out.Error	RMS Error	Max. Out.Error
TX, HS, UX	0.004359	0.014196	0.005371	0.014195	0.005090	0.014950
HS, UX	0.005293	0.013048	0.005277	0.013049	0.051580	0.013049
UX	0.003756	0.013181	0.005457	0.014226	0.005068	0.014226
HS	0.006455	0.031433	0.079099	0.193661	0.086914	0.167390
TX (10 MPa)	0.004276	0.014075	0.005306	0.015280	0.005043	0.015280

4.10 Effect of improving the degree of experimental control simulated

Thus far, all compression test simulations have been conducted with a value of the shape parameter modelling experimental variance equal to 1. A value of shape = 1 results in a compression test simulation where the controlling strain rate is sampled from a uniform distribution between its permissible limits. In comparison to physical laboratory test procedures, while the simulated tests have fallen within ISRM specifications, they represent a rather low level of test control. To illustrate, for the case of a rock with E = 40 GPa and ν = 0.25 and shape = 1, the axial strain of a uniaxial compression test will vary between 1.285 x 10^{-5} to 2.498 x 10^{-5} /second, corresponding to a range of loading rate between 0.513 and 0.999 MPa / second.

In this Section two further cases are investigated. In the first case examined, the value of the shape parameter is set at 10000. This will result in compression test simulations with a high level of control of the strain rate. To illustrate, for the case of a rock with E = 40 GPa and ν = 0.25 and shape = 10000, the axial strain of a uniaxial compression test will vary between 1.885 x 10^{-5} to +1.888 x 10^{-5} /second, corresponding to a range of loading rate between 0.753 and 0.755 MPa / second. This case is labelled "High" in Table 4.9 below.

The second case investigated eliminates any variance in the controlled strain rate. As the ANN is used to learn relations between strain component increments and stress component increments, this measure effectively reduces the number of different training patterns in the training data set to 9 (nine) despite the presence of approximately 2400 training exemplars in the training data set. This case is labelled "Perfect" in Table 4.9 below.

In both cases, a 3-6-3 ANN mesostructure was selected with learning and momentum parameters set at 0.00025 and 0.97 respectively. Simulated triaxial compression test data were produced for confining pressures 0, 5, 10 and 30 MPa and presented to each ANN for 10000 training iterations. Normalisation parameters were

chosen to map the stress increment output variables into the range –0.3 to 0.3. The trained networks are thus directly comparable with the network of Section 4.6, labelled "Low" in Table 4.9 below.

Table 4.9 : ANN training and validation performance simulating varying quality of machine control in compression testing

Level of control represented in training data set	ANN Performance after 10000 iterations (* 4000000 iterations)					
	Training		Triaxial validation		Plane strain validation	
	RMS Error	Max. Out.Error	RMS Error	Max. Out.Error	RMS Error	Max. Out.Error
Low	0.004359	0.014196	0.005371	0.014195	0.005090	0.014950
High	0.000763	0.006494	0.005999	0.017052	0.005418	0.017052
Perfect	0.000760	0.006467	0.005998	0.017050	0.005418	0.017050
Ideal limited set*	0.001715	0.003197	0.006683	0.019250	0.005949	0.018987

It is evident from Table 4.9, that the simulated degree of test control has little effect on the performance of an ANN trained with the data. When the degree of control is perfect, corresponding to a noiseless laboratory experiment, the performance of the resulting ANN is as good as or better than the case when appreciable lack of control (within ISRM limits) is introduced. This is because of the exactness of the relations between strain and stress component increments when linear elasticity according to Hooke's Law is assumed. Despite the fact that noise is artificially introduced into these training data sets, it is only introduced into one of the independent variables, $\Delta\varepsilon_z$. The remaining 'independent' variables are found from consideration of the loading conditions being applied (uniaxial, triaxial, hydrostatic) and the dependent variables determined using Hooke's Law. The 'randomness' introduced in the compression test simulations serves only to provide additional examples of the same constitutive relation.

This latter point is supported by ANN performance when trained with a data set comprising the 9 unique training exemplars present in the perfect data set (the perfect data set contains many repetitions). For this specific case, the number of training iterations had to be increased dramatically as the number of training exemplars was so small in comparison to the earlier 3 cases (9 exemplars vs 2400 exemplars). This training data set is presented in its entirety in Figure 4.8 below. The performance results of the trained ANN with this ideal limited set, can be seen to be satisfactory.

Many of the merits of ANNs relate to the manner in which they behave when imperfect or incomplete data is presented to them, either in training or recall tasks. The ANN performance when using the ideal limited set of Hooke's Law exemplars demonstrates that ANNs can have an effective role as interpolating or extrapolating agents, even when training data are perfect. The 9 exemplars of Figure 4.8 clearly provide sufficient information on the isotropic, linear elastic behaviour, such that the trained ANN can effectively interpolate and extrapolate to a loading condition foreign

to the training data set. This is evident when examining the validation results for the case of plane strain 'triaxial' compression test data.

```
TRAIN 3,1,3,1
  0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
 -0.00211  -0.00211   0.00844   0.00000   0.00000   0.22500
 -0.00211   0.00844  -0.00211   0.00000   0.22500   0.00000
  0.00844  -0.00211  -0.00211   0.22500   0.00000   0.00000
  0.00211   0.00211  -0.00844   0.00000   0.00000  -0.22500
  0.00211  -0.00844   0.00211   0.00000  -0.22500   0.00000
 -0.00844   0.00211   0.00211  -0.22500   0.00000   0.00000
  0.00422   0.00422   0.00422   0.22500   0.22500   0.22500
 -0.00422  -0.00422  -0.00422  -0.22500  -0.22500  -0.22500
END
```

Figure 4.8: Ideal, limited set of training exemplars for an ANN learning isotropic linear elastic behaviour of rock with $E = 40$ GPa and $\nu = 0.25$

It is also interesting to note that:

- For perfect conditions, the triaxial compression tests conducted at 10 and 30 MPa result in no new training exemplars being introduced into the training data set, which already contains information from a uniaxial compression test and a triaxial compression test conducted at 5 MPa.
- A single perfect triaxial test would contain all the exemplars of the ideal limited set shown in Figure 4.8.
- Results from a single perfect uniaxial test and a single hydrostatic compression test would contain all the exemplars of the ideal limited set shown in Figure 4.8.
- A perfect uniaxial compression test would not contain exemplars of hydrostatic compression or decompression shown in Figure 4.8 (the last two listed).
- A perfect hydrostatic compression test would only represent three of the exemplars shown in Figure 4.8 (the first and the last two).

With these observations, the results presented in the previous Section can be partially explained. The ANN trained with hydrostatic compression test data only (HS), does not contain any information about deviatoric loading. Hence its poor performance under validation tests is explained. The TX, HS, UX data set, UX HS training data set and the TX (10 MPa) data set, all contain all the exemplars, hence comparable performance of the corresponding ANNs can be explained.

The most surprising ANN performance under validation arises from that which was trained with uniaxial compression test data only. The reason for this is due to the fact that the simulated test data was permuted across the respective input and output variables. When not permuted, although training RMS and maximum output errors were 0.006776 and 0.023734 respectively, under validation with triaxial data the RMS

and maximum output errors were 0.168543 and 0.299644 respectively and under validation with plane strain 'triaxial' compression data the RMS and maximum output errors were 0.145291 and 0.299759 respectively. Thus, it is considered that permuting the input and output variables effectively provides the ANN with additional exemplars of 'end member' loading cases within which it can effectively interpolate when tested after training.

4.11 Discussion

This Chapter has reported on a limited set of numerical experiments conducted to investigate the ability of a multi-layered perceptron ANN to encapsulate linear elastic constitutive behaviour of rock. Conducting numerically simulated experiments allows issues associated with practical experimental error to be eliminated. A linear elastic constitutive model is also the simplest that can be applied to the behaviour of rock as an engineering material.

Chapters 2 and 3 highlighted that ANNs have been applied in a great many applications. One simple interpretation of ANNs that was presented there was that they are sophisticated curve fitting procedures that can be used to represent mathematical functions. In this Chapter, where perfect data and the simplest possible constitutive law have been adopted, the issue of whether the ANN is simply representing the data with which it is presented, or whether the procedures allow fundamental relations within the data to be abstracted and encapsulated has been investigated. On the basis of the work presented, neither option can be eliminated, but validation performance using simulated plane strain 'triaxial' (strictly: polyaxial) test data points toward the latter.

At the same time, care has been taken to make sure that the data with which the ANNs have been trained have a format compatible to that which is to be expected from data gathered from conventional physical laboratory experiments. This is because, ultimately, the final objective is to embody the constitutive behaviour of real specimens of rock within an ANN with data obtained from such tests. If it is evident that an ANN cannot learn the constitutive behaviour of a linearly elastic specimen with data simulating the laboratory tests from which such data would arise, then there is no need to continue investigations with data taken from physical laboratory experiments with real rocks.

Happily, the work conducted and presented in this Chapter has demonstrated that ANNs can perform this task easily. This should be unsurprising as, in effect, linear elasticity simply represents the state of stress as a weighted linear combination of the state of strain, and vice versa. Nevertheless, demonstrating that generalised Hookean behaviour can be effectively encapsulated in the weights of a trained ANN

provides a firm foundation for progression of the technique to learning the constitutive behaviour of real materials.

The work presented in this Chapter has also allowed identification of potential methodological pitfalls that can lead to a significant reduction in ANN performance in training and recall tasks, depending on circumstances. Consideration must be given over to the selection of appropriate normalisation parameters to maximise sensitivity of the normalised output quantities and to account for potentially undesirable effects associated with the artificial neuron activation function.

Isotropic materials can be dealt with by permuting the corresponding input and output quantities in each training exemplar that introduces symmetry into the training data set. Potentially, breaking this symmetry could provide an effective method to deal with anisotropic or orthotropic constitutive behaviour, providing data from laboratory testing can be obtained to support the ANN training process. The results from the training of an ANN with non-permuted simulated uniaxial compression test data support the approach. However additional complications would arise by the requirement to 'attach' a coordinate system to the intrinsic constitutive behaviour as well as a local coordinate system for the loading conditions and a global coordinate system for the problem domain.

5 Numerical modelling with Artificial Neural Networks using *FLAC*

5.1 Introduction

In the last Chapter, it was shown how an artificial neural network (ANN) could be readily trained to encapsulate linear elastic behaviour of a rock specimen within the values of the weight parameters. In this Chapter, it will be shown how the constitutive behaviour of real geologicial materials can be treated with exactly the same procedures and integrated within the numerical modelling code - *FLAC*.

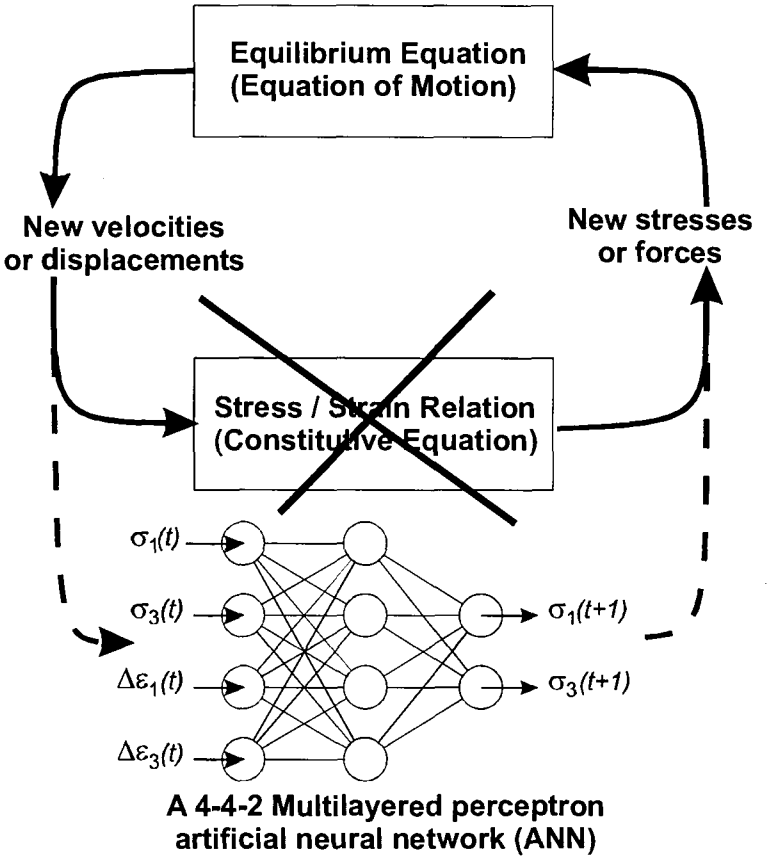


Figure 5.1 : Conceptual diagram illustrating the use of an ANN in place of an in-built constitutive model within a geomechanical code. (Modified from Itasca, 1993)

Finite element and finite difference techniques are already advanced computational tools. At present, their limitations arise from their use of simplified or stylised constitutive models. Numerical methods work well in cases when the selected constitutive model provides a reasonable approximation to the constitutive behaviour of the engineering medium concerned, particularly true for man-made materials used in construction and machine fabrication applications. For geological materials, the

suite of constitutive models typically available within numerical methods offers a rather limited choice if the physical behaviour of the material is to be represented accurately. Thus using an ANN to replace the constitutive model within a numerical method for stress and strain analysis (Figure 5.1) could offer significant advantages in geotechnical engineering applications and so is the subject of this Chapter.

In engineering analyses, such an ANN could be used in recall mode. Adhering to the formulation presented previously, for a known strain path, the strain increments are provided at the inputs to the ANN. The feedforward calculation procedure is then used to predict the increments in the stress tensors, and these increments accumulated along the strain path to determine the state of the body of rock under consideration.

5.1.1 Chapter Outline

In order to understand the detail of how the ANN is embedded within *FLAC*, understanding of the *FLAC* calculation sequence needs to be established. *FLAC* solves mechanical problems iteratively. In the first part of this Chapter, the general characteristics of *FLAC* and its explicit finite difference calculation cycle are explained. This is presented to render the *FLAC* calculation method transparent, and to ensure that the purpose and method of execution of the constitutive model is understood. An enhanced presentation, is deferred to Appendix B1, and is designed for readers who want to know the details.

The procedures used in coding the feedforward calculation of the multi-layered perceptron ANN in the *FISH* language and ANN use within *FLAC* are then explained. Following from this, the series of verification procedures adopted are presented. They are then used to test results of simulations using the in-built *FLAC* linear elastic model in parallel with the ANN UDM representing linear elastic behaviour (from the last Chapter). This is done so that the differences between the two, or rather, the lack of them, are highlighted.

In this way, the work presented in this Chapter will demonstrate that an ANN can be used within the *FLAC* numerical scheme in exactly the same way as any of the in-built *FLAC* constitutive models. Following on from this stage, triaxial test data from a real, rather than simulated, test will be used to demonstrate how the technique successfully encapsulates total rock constitutive behaviour with the *FLAC* code.

5.2 Numerical Modelling & *FLAC*

5.2.1 Choice of Numerical Model

In this Chapter, focus is retained on an explicit finite difference numerical method because:

- no iterations are necessary in the solution procedure to follow non-linear constitutive laws, and
- non-linear constitutive laws are always followed in a valid physical way.

Both reasons are particularly important in the current context of using of an ANN to embody constitutive behaviour.

Jing and Hudson (2002) and Jing (2003) provide an up-to-date summary of the state-of-the-art in numerical modelling, as applied to rock mechanics. All the numerical methods are implemented by dividing up the body under study, or problem domain, into a large number of discrete parts, commonly called *elements*, or sometimes called *particles*. These are frequently triangular or quadrilateral in shape for two-dimensional methods. In finite difference methods, mechanical quantities are defined only at the vertices of the element. With finite element methods, so called 'shape-functions' need to be selected so mechanical quantities have values any location within the problem domain. After this, the main distinction between the plethora of numerical methods available reduces to whether the formulation of the equations for the mechanical quantities within the problem domain is explicit or implicit.

Explicit methods are iterative methods where the state and dynamic performance of a body under investigation is modelled at discrete, closely spaced intervals in time. During each iteration, displacements, velocities, strains, forces and stresses are considered 'frozen' in time such that they do not vary over the iteration time step and all quantities in the dynamical and constitutive equations, used to predict the next state, are known. This means that the time step associated with an explicit method must be sufficiently small for the 'frozen' condition to hold true. Strain compatibility and (dynamic) equilibrium are automatically satisfied.

Implicit methods of solution involve communication of the state and performance of all parts of the body being modelled during the solution step. In the computational procedure, this implies that a large, so-called, stiffness matrix has to be constructed requiring large amounts of computer memory. An advantage with an implicit method is that the time step can be arbitrarily large. Iterations of implicit methods are also required but these are to find a solution that satisfies the requirements of strain compatibility or equilibrium, rather than iterations to update the state of the body. With large time steps, there is no guarantee that non-linear constitutive behaviour can be properly represented, although this problem is reduced when the time step chosen in the implicit method is small so that non-linearity can be properly modelled. In this case, implicit methods start to converge on the solution approach of explicit methods but with the disadvantage that large solution matrices still have to be constructed and inverted, requiring more memory and computation

time in comparison to an equivalent explicit method. Implicit approaches are a typical characteristic of many (but not all) finite element programs.

If faced with the problem of choosing a suitable numerical method in order to investigate the likely performance of an excavation in rock, at present there are few distinctions between the methods in practical use. All will have procedures available to define the geometry of the problem domain and to handle the spatial discretisation process. For straightforward constitutive models, all will require the same kind and quantity of information on material properties that define the material behaviour. The initial state (forces, stresses, displacements, velocities, strains, etc) of the problem domain will need to be specified to all of the programmes and the same is true of the constraints that are to apply to the problem domain, the so-called, boundary conditions. Each program / method will also have procedures and techniques available for accessing the results of the model.

In short, from the point of view of the user, practically there is little difference between one code and another. The distinctions occur in the method used to formulate the (same) physics, the accuracy of the formulation to the true physics and the solution procedure employed. Such things are known in detail by the programmer of the code but are hidden from, and assumed correct by, the typical user. Under these circumstances, that are prevalent at the time of writing, the choice of using one numerical method over another comes down to issues such as cost, availability, familiarity and ease-of-use; however, as Jing (2003) points out in his extended review, particular formulations are suited to investigations of a specific nature. For example, Boundary Element Method is considered to be the best tool to adopt to simulate fracturing processes.

5.2.2 *Characteristics of FLAC*

FLAC (standing for: *Fast Lagrangian Analysis of Continua*), is a computer program that uses a two dimensional explicit finite difference formulation to solving mechanical problems involving solid bodies (Itasca, 1993, 1995). As such it benefits from the two bulleted advantages indicated in the previous Section. It is widely used in the geotechnical engineering community and in geotechnical research; two international conferences have been held on geotechnical engineering and research using *FLAC*.

The version of the code adopted for this work was not particularly easy to use but was available and familiarity with it had developed with its use. The latest version of the code has increased its ease of use through the development of a graphical user interface (GUI) where program instructions are provided using menus. A significant

advantage of *FLAC* is that beneath the GUI is the same command-driven interface that prevailed with the earlier versions of the code.

The command-driven method of use of the code offers two main advantages. Firstly, any commands entered at the keyboard can also be written to a script file that can be submitted to the command interpreter and dealt with in batches. The script file then provides a precise and exact record of any simulation or engineering study conducted with the code. These script files also constitute the second main advantage of the *FLAC* code. Any simulation can be compactly transmitted to another scientist or engineer for verification and scrutiny. The command interpreter can also be programmed with a script programming language called *FISH* that allows automation of modelling procedures. *FISH* programming commands are held within the same files as those of the *FLAC* commands.

A special characteristic of the *FLAC* program that was important in its selection for the work presented here, is its ability to override use of any one of its available constitutive models and to use a user-defined constitutive model within its main calculation cycle. In the version of *FLAC* used in this work, user defined constitutive models (UDMs) are coded in the *FISH* scripting language. This makes them transparent and portable to any other user of *FLAC*. In later versions, UDMs can be coded in the language C++ and compiled into dynamic link libraries. This has the advantage that the *FLAC* code will execute more quickly than it would with a *FISH* coded UDM and retains the concept of portability, but has the simultaneous disadvantage that a compiled UDM cannot be scrutinised by another user before it is used in their simulation.

5.2.3 Summary of FLAC calculation sequence

Figure 5.2 summarises the calculation sequence of the *FLAC*-like explicit finite difference numerical method, that has been expanded upon as a 'worked example' in Appendix B1. For the worked example, involving only a single finite difference zone, the results of each calculation step are reported within that Appendix, following the process indicated in Figure 5.2. The complete set of calculations can be viewed in the print out of the Microsoft Excel worksheet that is presented in Appendix B3. The reader may also wish to verify that the calculation sequence reported herein is in fact identical to that conducted within *FLAC* by running the *FLAC* data file listed in Appendix B2.

In conducting and presenting the worked example calculations, the author gained sufficient understanding of *FLAC* in order to identify, precisely, how to couple the ANN to it. This level of understanding goes beyond that that would be expected of

a routine user of the program, but was necessary, not to implement the constitutive model replacement, but to verify that it operated as designed.

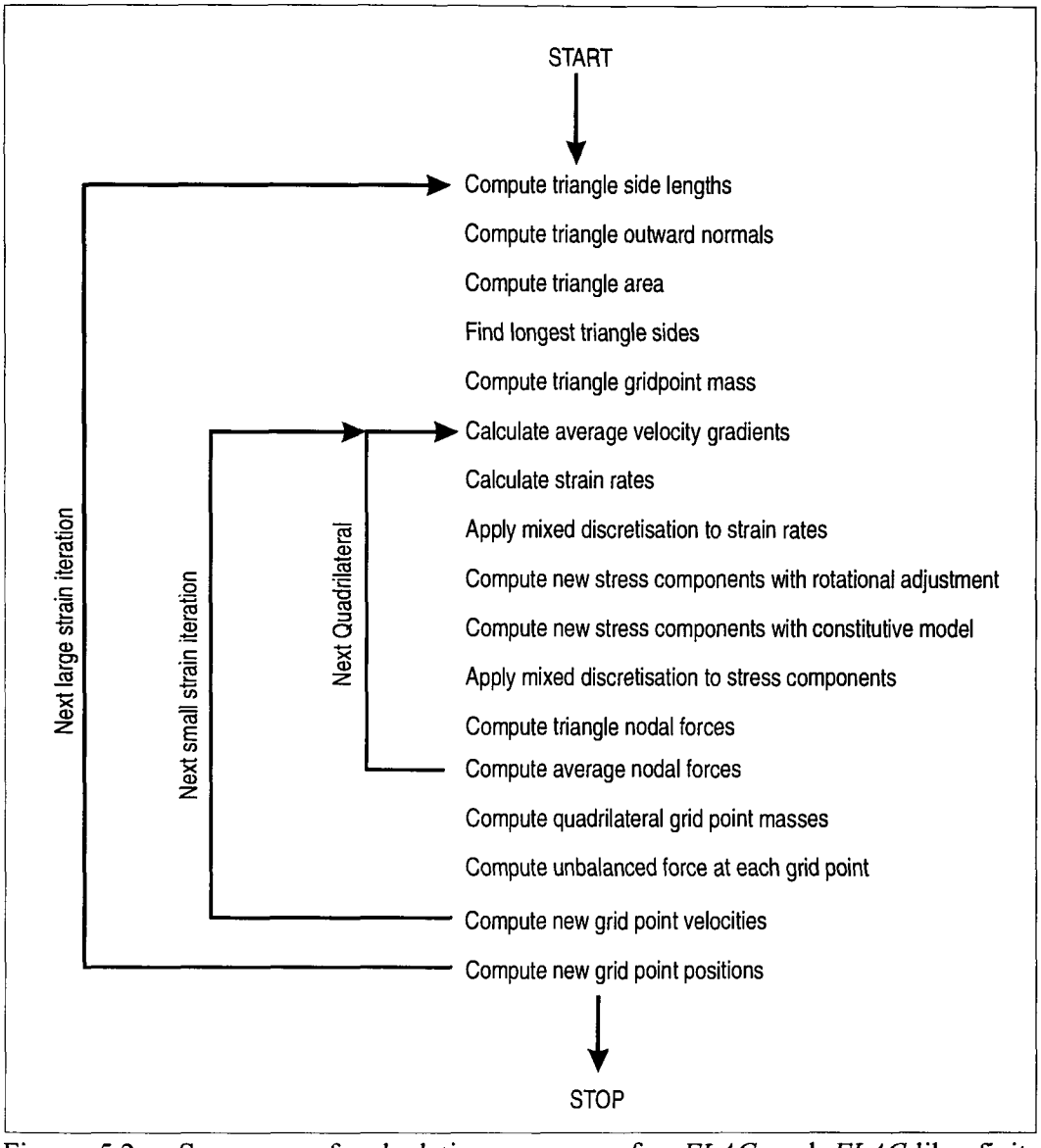


Figure 5.2 : Summary of calculation sequence for *FLAC* and *FLAC*-like finite difference numerical methods.

FLAC numerical models decompose the problem domain into a series of non-rectangular quadrilateral zones. Each quadrilateral is itself decomposed into two sets of two overlapping triangles. Each set of triangles 'covers' the quadrilateral. The formulation is designed like this to overcome a former limitation of finite difference methods: a limitation that only rectangular zones could be analysed (more detail on this issue is presented in Appendix B1).

In using a model involving many finite difference quadrilaterals, the effective force components applying at the vertices of every quadrilateral are calculated before

the vertex velocity components are recomputed. This is because for any one vertex within the whole finite difference grid, forces acting at this vertex arise from more than one quadrilateral (with the exception of zones logically found at the 'corners' of the deformed grid).

In small strain mode, updates to vertex co-ordinates are not implemented as they are assumed insignificant. Consequently, it is unnecessary to recalculate the triangle geometric quantities such as the outward normals, triangle side lengths and areas. The grid point masses will also be unchanged in this mode and so need calculation only once. Having computed the new grid point velocities, the next iteration of the method is conducted, starting with computation of the new velocity gradients.

In large strain mode, the grid point positions can be updated every time step and thus, geometric quantities associated with triangles need to be recomputed every iteration. The solution method depends upon a damping parameter dependent on grid point masses, in turn dependent on triangle geometry. Within *FLAC* the grid point masses are recomputed once every 10 iterations.

Various quantities can be logged, or recorded, while the main calculation cycle proceeds, to produce model results. One of the most interesting aspects of the *FLAC* code, from the point of view of this work, is that the constitutive model module of the code is relatively self-contained. This means that it is straightforward to replace the in-built constitutive model with one that is user-defined without any requirement to modify *FLAC*'s the main calculation cycle. One possible exception to this is the fact that the triangle grid point masses are used in calculating numerical damping parameters, that apply globally.

5.3 Encoding the User Defined Constitutive Model

Within *FLAC* the constitutive model is consulted four times per zone per time step during the main calculation cycle loop, that is, once for each triangle in the two overlaid pairs of triangles contained in one zone.

The main function of the constitutive model procedure within *FLAC* is to compute the new stress components for a triangle, given the strain increments (strain rates, for $\Delta t = 1$) for the triangle. This corresponds to mode 2 invocation of the constitutive model.

The constitutive model procedure has other ancillary functions corresponding to 3 other modes: mode 1, mode 3 and mode 4. Mode 1 invocation occurs prior to execution of the main calculation cycle for the purpose of initialisation of variables local to each zone. Mode 3 is invoked when the grid point masses of the finite difference grid need to be calculated. This is once before the main calculation cycle executes, and at least every 10 time steps in large strain mode where grid point co-

ordinates are updated. Mode 4 is invoked when a thermal analysis is being undertaken.

5.3.1 Input and output quantities

Four strain increment components are provided as input to the constitutive model during a mode 2 invocation. These are:

e_{11} Normal strain increment in the x-direction, variable zde11

e_{22} Normal strain increment in the y-direction, variable zde22

e_{33} Normal strain increment in the z-direction, variable zde33

e_{12} Shear strain increment in the xy plane, variable zde12.

Four stress components are provided as input to the constitutive model during a mode 2 invocation which must be updated as output quantities. These are:

σ_{11} Normal stress component in the x-direction, variable zs11

σ_{22} Normal stress component in the y-direction, variable zs22

σ_{33} Normal stress component in the z-direction, variable zs33

σ_{12} Shear stress component in the xy plane, variable zs12.

5.3.2 Material properties

Material properties are identified within the constitutive model procedure with use of the **f_prop** keyword. Variables defined with a **f_prop** keyword within the constitutive model procedure have values that are specific to particular zones. For example, the *FISH* statement:

```
f-prop YoungsMod
```

would allow for a different value of YoungsMod to be assigned to each zone within the finite difference grid.

5.3.3 Structure of User Defined Constitutive Model

A user-defined constitutive model (UDM) must be coded according to defined conventions if it is to work properly within *FLAC*. A UDM for isotropic linear elasticity is shown in Figure 5.3 below. This will be used to explain the general principles of operation and to explain the verification studies to be undertaken. Lines that are comments or notes in a *FLAC* or *FISH* data file are prefixed with a ';'.

```
;FISH version of standard elastic model
; based on FLAC user manual page 7-57.
set echo off

def m_elastic
 constitutive_model
 f_prop  m_g m_k
 f_prop  m_e1 m_e2 m_g2

 Case_of  mode
; ----------------------
; Initialisation section
; ----------------------
  Case 1
   m_e1   = m_k + 4.0 * m_g / 3.0
   m_e2   = m_k - 2.0 * m_g / 3.0
   m_g2  = 2.0 * m_g
; ---------------
; Running section
; ---------------
  Case 2
zs11=zs11 + zde11*m_e1 + (zde22 + zde33) * m_e2
zs22=zs22 + zde22*m_e1 + (zde33 + zde11) * m_e2
zs33=zs33 + zde33*m_e1 + (zde11 + zde22) * m_e2
zs12=zs12 + zde12 * m_g2
; ----------------------
; Return maximum modulus
; ----------------------
  Case 3
   cm_max = m_k + 4.0 * m_g / 3.0
   sm_max = m_g
  Case 4
; ----------------------
; Add thermal stresses
; ----------------------
    ztsa = ztea * m_k
    ztsb = zteb * m_k
    ztsc = ztec * m_k
    ztsd = zted * m_k
 End_case
end

opt m_elastic
set echo on
```

Figure 5.3: A UDM for isotropic linear elasticity coded in *FISH*.

The custom written *FISH* function is called **m_elastic** and is identified with this name elsewhere within *FLAC*. The function is identified as a UDM with the use of the **constitutive_model** statement. The next section of the UDM declares the material property parameters through the **f_prop** statement. Other parameters which are used in the calculations but that are global to every zone in the finite difference mesh can be declared with the **float** or **integer** keywords following the **f_prop** section (neither need to be illustrated in Figure 5.3).

A **case_of *variable*... end** compound *FISH* statement acts as a switch, allowing execution of the UDM to branch depending on the value of the variable *mode*. In a UDM the mode variable, which ranges from integer values 1 through 4, identifies the mode for which the UDM has been invoked. When *mode* = 1, this corresponds to invocation for initialisation. In the above UDM, initialisation involves calculation of values for Lamé's constants from values of the bulk and shear modulus as well as calculation of the variable *m_g2* which evaluates to twice the shear modulus. The latter variable is computed once only in the initialisation section for computational efficiency.

When *mode*=2, this corresponds to the main running section where new stresses are computed from strain increments. When *mode*=3, values for the maximum bulk and shear modulus are returned and used in the calculation of the grid point mass of the vertices of the quadrilateral for which the UDM has been invoked. When *mode*=4, the UDM is called to update the stress tensor to account for thermal strains.

The **end** statement terminates the definition of the UDM function. The two remaining lines shown in Figure 5.3, are *FLAC* statements rather than *FISH* statements (because the latter must appear inside a **def ... end** compound statement). The first of these attempts to optimise the UDM by interpreting it in order that when invoked, the UDM code runs faster. The **set echo on** statement reports each line of interpreted *FLAC* and *FISH* commands to the interactive console as it is read from the file. The first line in the file, **set echo off**, suppresses this while the UDM is being read.

5.3.4 Tests to verify a UDM

An isotropic linear elastic UDM is used first in order to demonstrate the methods through which the UDM can be tested for correctness. Listings of the *FLAC* data files used to conduct these tests are presented in Appendix B4. A narrative describing implementation of these tests is presented in Appendix B5. By comparing *FLAC* performance with its in-built constitutive model of linear elasticity, with *FLAC* performance using a linear elastic UDM, this narrative defines benchmark acceptability criteria for acceptance that a UDM formulation is 'correct'. Importantly, the tests proposed are independent of the constitutive behaviour being modelled, providing there is an available in-built *FLAC* constitutive model the UDM aims to replace. They will allow any UDM, including one embedding an ANN implementation of an in-built *FLAC* constitutive model, to be verified. In the context of the current work, this is important because these tests will verify that the ANN logic is working correctly within the *FLAC* environment. The verification strategy is summarised in the 2 steps below:

1. Compare linear elastic UDM with *FLAC* in-built linear elasticity. (Defines benchmark comparative performance thresholds)
2. Compare ANN linear elastic UDM with in-built linear elasticity. (Favourable comparison with benchmark thresholds verifies the ANN implementation and logic within *FLAC*)

If step 2 results in a favourable outcome, values of ANN weights can be replaced with values that represent realistic rock constitutive behaviour and thus the final objective is attained.

The comparisons referred to in steps 1 and 2 are between UDMs across three separate verification Test Types. The verification Test Types, which progressively become more demanding, are described immediately below.

Test Type 1

This involves specifying a single zone finite difference grid, fixing the four grid points in the x and y direction and taking the zone through known strain paths. The performance of the UDM is monitored to determine if the UDM is producing the required behaviour. Three cases of strain path are considered:

Case 1: Laterally confined, 'axial' compression

Case 2: Pure shear

Case 3: Cases 1 and 2 combined

Stress components are monitored to determine whether the valid estimates are reproduced within *FLAC* in comparison to analytical values for the same loading cases and strain paths. Gradients are calculated from the output performance curves and compared to values for materials properties that would normally be derived from them.

Test Type 2

This involves specifying a symmetric finite difference grid with a large number of zones and applying symmetric boundary conditions. On one side of the line of symmetry, the *FLAC* in-built constitutive model is used. On the other side, the UDM representing the same material properties is used (Figure 5.4). The symmetry of the performance of the whole finite difference grid is monitored to determine the effectiveness of the UDM. If the symmetry is preserved, the UDM can be said to be as effective as the in-built constitutive model. Importantly, this assesses interactions across the ensemble of *FLAC* finite difference quadrilaterals.

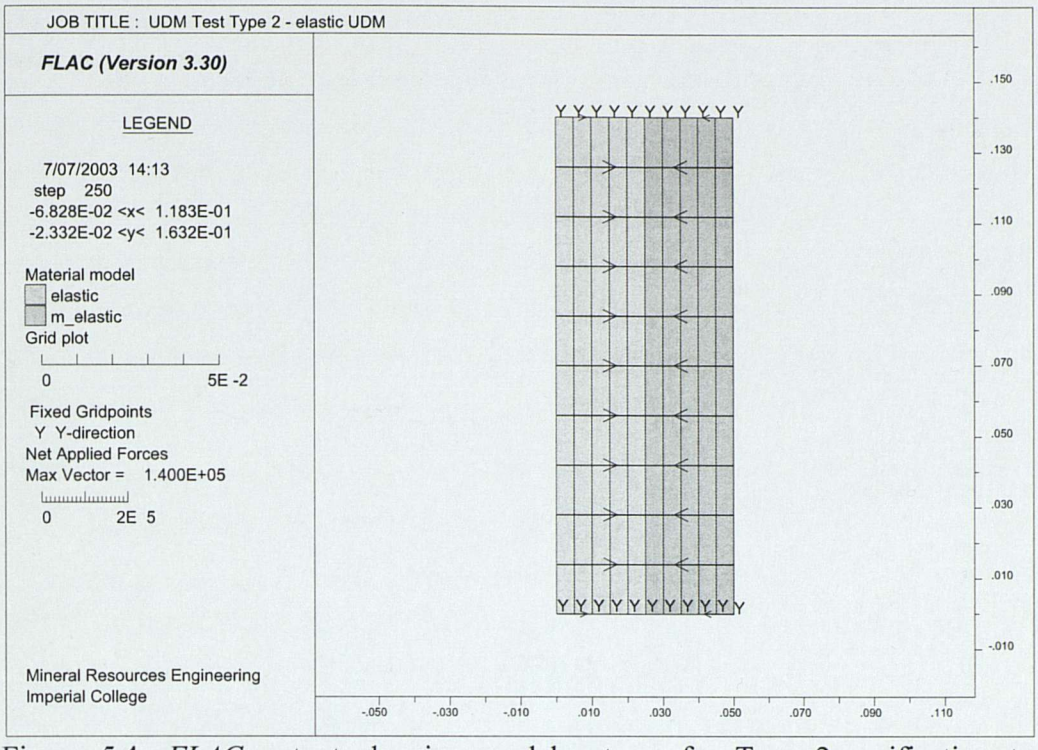


Figure 5.4: *FLAC* output showing model set up for Type 2 verification test. Rectangular quadrilateral zones using *FLAC* in-built constitutive model are in light grey on LHS of grid; zones using UDM constitutive model in darker grey on RHS.

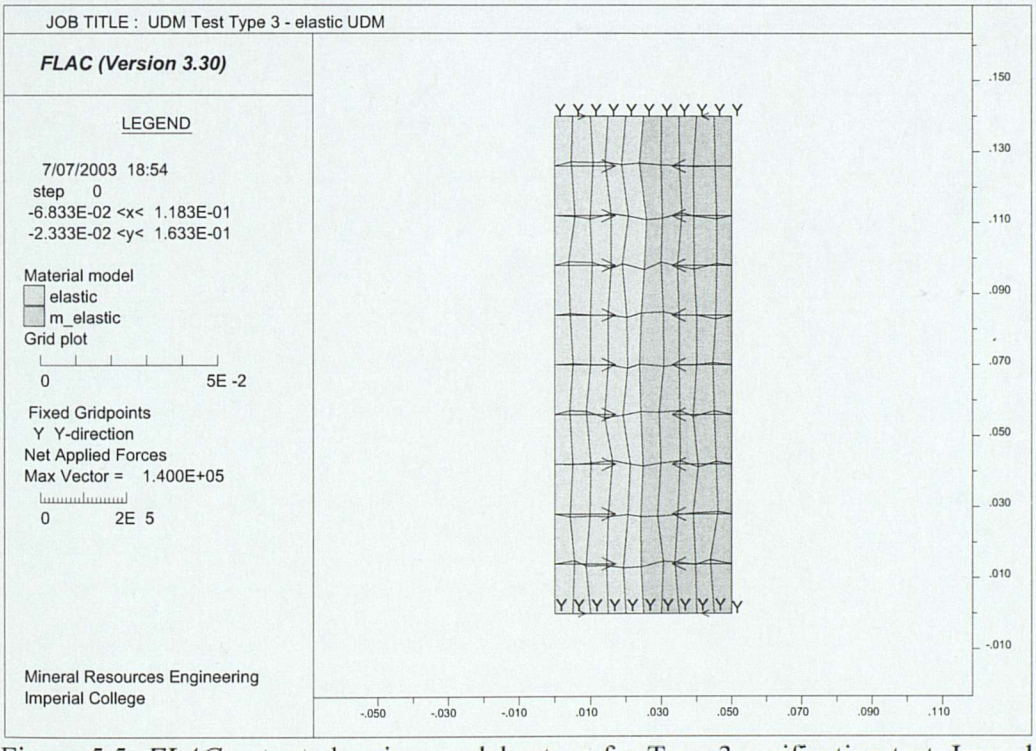


Figure 5.5: *FLAC* output showing model set up for Type 3 verification test. Irregular quadrilateral zones using *FLAC* in-built constitutive model are in light grey on LHS of grid; zones using UDM constitutive model in darker grey on RHS.

Test Type 3

This involves specifying a non-symmetric, irregular finite difference grid with a large number of quadrilaterals. Approximately one half of the model is assigned the *FLAC* in-built constitutive model; the remaining part is assigned the UDM (Figure 5.5). Asymmetry of finite difference zone geometry is deliberately introduced within the model to check whether errors in the UDM are latent under symmetric conditions. The results are assessed with the same tests of Test Type 2, with the distorted grid (i.e., degree of symmetry of performance in the face of asymmetry introduced in model set up).

5.3.5 Summary of benchmark validation test results defining criteria for acceptable performance of the ANN UDM

Test Type 1, Case 1. – Linear elastic UDM

FLAC output was analysed as if representing observed data. Calculated values of material properties were as specified in defining the simulation.

Test Type 1, Case 2 – Linear elastic UDM.

FLAC output was analysed as if representing observed data. Calculated values of material properties were as specified in defining the simulation.

Test Type 1, Case 3. – Linear elastic UDM

FLAC output was analysed as if representing observed data. Calculated values of material properties were as specified in defining the simulation.

Test Type 2 – Linear elastic on LHS of grid, Linear Elastic UDM on RHS of grid

Differences in stress tensor components for two symmetrically located quadrilateral zones (one in each UDM area) in a mesh comprising rectangular zones were insignificant up to around 250 time steps after the start of the simulation. In subsequent time steps, differences in the stress tensor component values grew, but at no time exceeded 1 part in 1000 of the tensor values themselves (Figure 5.6). Initially this was considered to be a result of minor variations in implementation of UDMs within *FLAC*. A subsequent check simulation, identical, but using the in-built, compiled linear elastic model in *FLAC* throughout, revealed similar behaviour - eliminating this explanation. The differences, mid-way through the simulations are thus ascribed to the cumulative effect of many rounding operations in arithmetic operations over many calculation cycles. These small discrepancies grow, to produce differences in stress tensor values ultimately constrained to the maximum values mentioned above.

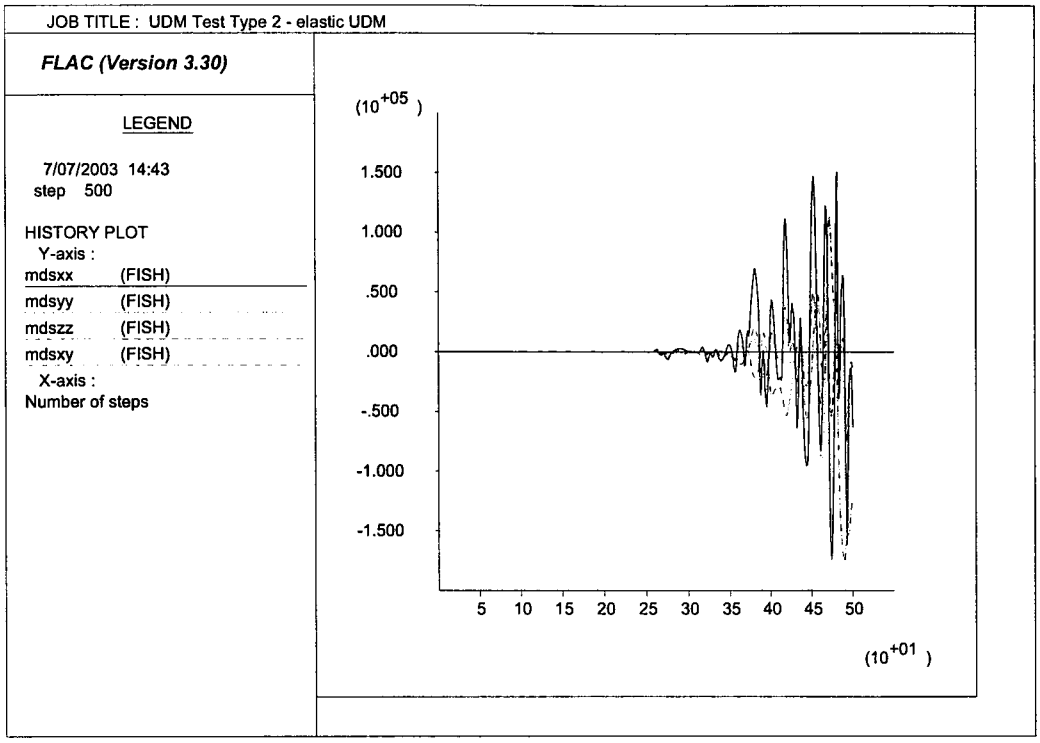


Figure 5.6: *FLAC* time step plot of differences in four stress tensor component magnitudes (mds**) monitored at two symmetrically located zones within the model, up to 500 time steps for the Test Type 2 UDM verification model. For example, mdsyy monitors the difference between σ_{yy} magnitudes for a zone of the LHS of the model in Figure 5.4 and a zone on the RHS of the model in Figure 5.4.

Test Type 3 - Linear elastic on LHS of grid, Linear Elastic UDM on RHS of grid

Differences in stress tensor components for two symmetrically located quadrilateral zones (one in each UDM area) in an irregular mesh were insignificant up to around 125 time steps after the start of the simulation. In subsequent time steps, differences in the stress tensor component values grew, but at no time exceeded 1 part in 100 of the tensor values themselves.

General observations

The simplest in-built constitutive model available was adopted for comparison against the simplest possible UDM in order that focus could be retained on the suitability of the tests as legitimate verification procedures. In effect, by implementing Step 1, the verification procedures themselves were being verified.

The true purpose of the procedures is not really to be able to investigate the suitability of an isotropic linear elastic UDM against the in-built elastic model, but to be able to make an assessment of the effectiveness of an ANN UDM representing linear elastic behaviour in comparison to the in-built elastic model. This is to be the subject of the next Section.

It is interesting to note that even when perfect, analytical constitutive models are used in the verification tests, significant 'imperfections' ultimately arise (e.g. Figure 5.6). This Step 1 of the verification strategy has identified that it is not reasonable to expect 'perfect' behaviour of an ANN UDM; numerical rounding and precision issues lead to imperfections in *FLAC* when it is benchmarked against itself.

The model performance results presented here thus represent the target level of performance of the ANN UDM. If under test, the ANN UDM produces performance comparable to that presented for the ideal situation, this will verify that ANN can be used effectively within finite difference procedures.

5.4 Verifying the ANN user defined constitutive model with FLAC

5.4.1 Need for tensor transformations arising from ANN data source

In integrating the ANN UDM into the *FLAC* code, it must conform to the conventions specified for the format of user-defined constitutive model procedures. It must also be remembered that although, thus far, no real experimental data has been used to train the ANN (idealised, numerical simulations have been used), due consideration must be given over to the likely data sources that will be used.

The main intent is to use data gathered from laboratory triaxial tests on rock specimens to train the ANNs, which then have their weight sets ported into the *FLAC* environment to define the ANN UDM. It has been shown that the general purpose of the UDM is to update the stress tensor components from strain increments derived from velocity gradients. Unless special conditions are specified, it should be expected that shear strain increments will be non-zero. The reason why non-zero shear strain increments are an issue is because, observations from a triaxial compression test on a rock specimen will always produce zero shear strain, as these tests only measure normal strains, in the axial and diametrical (or radial) directions. ANNs trained with data on principal strain and principal stress components will not perform well if asked to produce shear strain components.

The solution to this issue is to take strain increments provided by the *FLAC* main calculation cycle and to transform these to principal strain increments. The angle between the zone co-ordinate frame and the orientation of principal strain axes must be noted. The principal strain increments are used to compute principal stress increments, which is an arrangement compatible with the ANN data source. The principal stress increments are transformed back to the co-ordinate system of the (triangle of the) finite difference zone and added to the components of the 'old' stress tensor. The *FLAC* calculation cycle then proceeds as normal.

With the exception of isotropic linear elasticity, within the mode 2 section of the in-built constitutive models available within *FLAC*, the calculation sequence is as

follows. Firstly, elastic increments of stress are computed from the strain increments and these are added to the old stress tensor. This new 'trial' stress tensor is then transformed to its principal orientation. The principal stress tensor is then subjected to a series of tests to determine whether 'failure' has occurred. If not, the principal stress components are unchanged and the principal stress tensor is transformed back to the co-ordinate axes of the triangle of the zone that invoked the constitutive model. If so, the principal stress components are adjusted to reflect the identified failure mode. The adjusted principal stress tensor is then transformed to the co-ordinate axes of the triangle.

In the case of the ANN UDM, a stress transformation to principal axes is unnecessary as no tests for failure are necessary. All pre-peak, peak and post-peak behaviour of specimens under test is already encoded within the weights of the trained ANN. The purpose of the strain and stress transformation used with the ANN UDM, referred to earlier, have different purposes. These transformations are necessary for compatibility between data source and data use.

It is appropriate to note that stress or strain transformations have no physical effect on the state of stress or strain respectively. The transformation procedure simply permits examination of the same state of stress or strain within co-ordinate axes chosen such that the values for shear strain components are zero. Thus, stress or strain transformation has no effect on the constitutive response of the material under consideration.

In the development of the ANN UDM, firstly the stress and strain transformation scheme described above was implemented in the *FISH* coded UDM for isotopic linear elasticity and verification Test Types 1, 2 and 3 successfully executed with the revisions.

5.4.2 Preparation of the FISH data files encoding the ANN UDM

The ANN UDM is implemented within the *FLAC* environment by means of two files. The first file is an initialisation file that uploads ANN weights, thresholds and normalisation and de-normalisation data into *FLAC* tables. The second file contains the *FISH* code for the UDM.

Automatic procedures have been written to convert an ANSIM .net file containing trained network weights into a *FLAC* command script file in order that there is little overhead required for the initialisation of the ANN. Within the automatically generated file, there are 'placeholders' for the normalisation and denormalisation parameters which are entered manually. The MS Excel Visual Basic for Applications code which produces this initialisation script file is presented in Appendix B6.

The ANN feedforward multilayered perceptron code replaces the linear elastic law within the *FISH* UDM. The latter is shown in Figure 5.7 below. Figure 5.8 shows the statements that replace those presented in Figure 5.7 in order to implement the ANN UDM within the mode 2 invocation section.

```
; --- Compute principal stress increments from principal strain increments
;
--- Isotropic linear elastic model
;;
    $zds1 = $e1 * m_e1 + $e2 * m_e2 + $e3 * m_e2
    $zds2 = $e1 * m_e2 + $e2 * m_e1 + $e3 * m_e2
    $zds3 = $e1 * m_e2 + $e2 * m_e2 + $e3 * m_e1
```

Figure 5.7: *FISH* code fragment showing implementation of isotropic, linear elastic constitutive law.

Although there are clearly more *FISH* statements and computations required with the ANN UDM, with one exception, the ANN UDM will not change irrespective of the complexity of the constitutive behaviour being modelled. The exception relates to situations when history or state parameters are also used in the ANN training process. If the constitutive behaviour is known to be modified by the current state of stress, then stress component values would also be used on the input side of training vectors used to train the ANN. In use within the ANN UDM, the current stress component values would also need to be supplied as input. The only modification needed within the ANN UDM would be to add a further 3 lines which load these variable values into the input layer of the ANN. Using principal stress components to describe the current state of stress, the three lines are:

```
ytable(act,4)=-$s1
ytable(act,5)=-$s2
ytable(act,6)=-$s3
```

Referring to Figure 5.8, these three lines would be placed immediately following the first three 'ytable' statements.

```
;
; --- Compute principal stress increments from principal strain increments
;
; --- Using ANN code
;
;--- Load strain increments into ANN input layer
;--- Strain increments trained in the ANN adopted the geomechanics convention,
;--- whereas in FLAC tensile strains are positive.
    ytable(act,1)=-$e1
    ytable(act,2)=-$e2
    ytable(act,3)=-$e3
;--- Insert the ANN Feedforward Code Here. Uses principal stresses and strains ---

    last_in_prevlayer=0
    loop layer (2, niayers)
;
; Remap the input values to interval -0.5 to 0.5

;       if layer = 2 then
          loop neuron (1,ytable(struct,layer-1))
            ytable(act,neuron)=xtable(neuron,1)*ytable(act,neuron)
            ytable(act,neuron)=ytable(act,neuron)+ytable(neuron,1)
          end_loop
       end_if
;
; Main multilayer perceptron feedforward calculation
       last_in_prevlayer=last_in_prevlayer+ytable(struct,layer-1)
       first_in_layer=last_in_prevlayer+1
       last_in_layer=first_in_layer+ytable(struct,layer)-1

      loop neuron (first_in_layer,last_in_layer)
         sum=0.0
         weight=0
         loop input(first_in_layer-ytable(struct,layer-1),first_in_layer-1)
            weight=weight+1
            sum=sum+ytable(neuron,weight)*ytable(act,input)
         end_loop
         sum=sum+ytable(thresh,neuron)
         Ytable(act,neuron)=(1/(1+exp(-sum)))-0.5
      end_loop
   end_loop
;
; Remap the outputs of the feedforward calculation into stress space   ;
    count=0
    loop i (nunits-ytable(struct,nlayers)+1,nunits)
      count=count+1
      ytable(act,i)=ytable(act,i)*xtable(nunits+count,1)
      ytable(act,i)=ytable(act,i)+ytable(nunits+count,1)
    end_loop    ;
; End of ANN code
;
; Output the new principal stresses from the ANN
    $zds1 = -ytable(act,nunits-2)
    $zds2 = -ytable(act,nunits-1)
    $zds3 = -ytable(act,nunits)
```

Figure 5.8: *FISH* code fragment, showing ANN code within mode 2 invocation section of a *FLAC* user defined constitutive model.

No attempt has been made here to write *FISH* code that is in any way optimal. Rather, the *FISH* statements have been written so that the procedure reads as pseudo-code and thus is understandable to the non-ANN specialist, familiar with generic structured programming techniques. UDMs run considerably more slowly than their *FLAC* in-built counterparts which are compiled. The additional computations required for the ANN UDM, slow the computations further. However, at this stage, the overriding design consideration is to provide proof-of-concept in a format that can be reproduced by others. Execution speed is an aspect that can be addressed after the concept is proved.

Nevertheless, several tactics were investigated to try to improve execution speed with the ANN UDM. The *FLAC* optimise command (which invokes a pre-interpreter) does not work with the ANN UDM as it uses mixed mode arithmetic operations. Use could have been made of *FLAC*'s internal memory structures to store the weights, thresholds and normalisation parameters and would have considerably speeded up execution of the code. However, this would have led to implementation with the use of pointer arithmetic that would have been more difficult to understand for programmers with only limited experience.

The most efficient and compact way of implementing the ANN UDM would be to write a compiled version in a native programming language such as C or C++ and compile this code to a dynamic linked library. The latest version of *FLAC* has such a facility but was not available to this author.

5.4.3 Performance of the ANN UDM

The trained ANN used in the verification tests within *FLAC* was that described as having a 'perfect' degree of experimental control in Table 5.9 of Section 5.10. This network was trained for 10,000 iterations with the back propagation of error training algorithm using data simulating uniaxial and triaxial compression tests on a rock with E=40 GPa and ν=0.25.

After training, the binary ANN .net file was converted to a text file using the ANSIM 'dumpnet.exe' utility. The text formatted .ANN file was then imported into the MS Excel environment and exported as a *FLAC* initialisation file using custom written Visual Basic procedures (Appendix B6).

Test Type 1

The performance of the ANN UDM under verification Test Type 1 is presented in Figures 5.9 to 5.12.

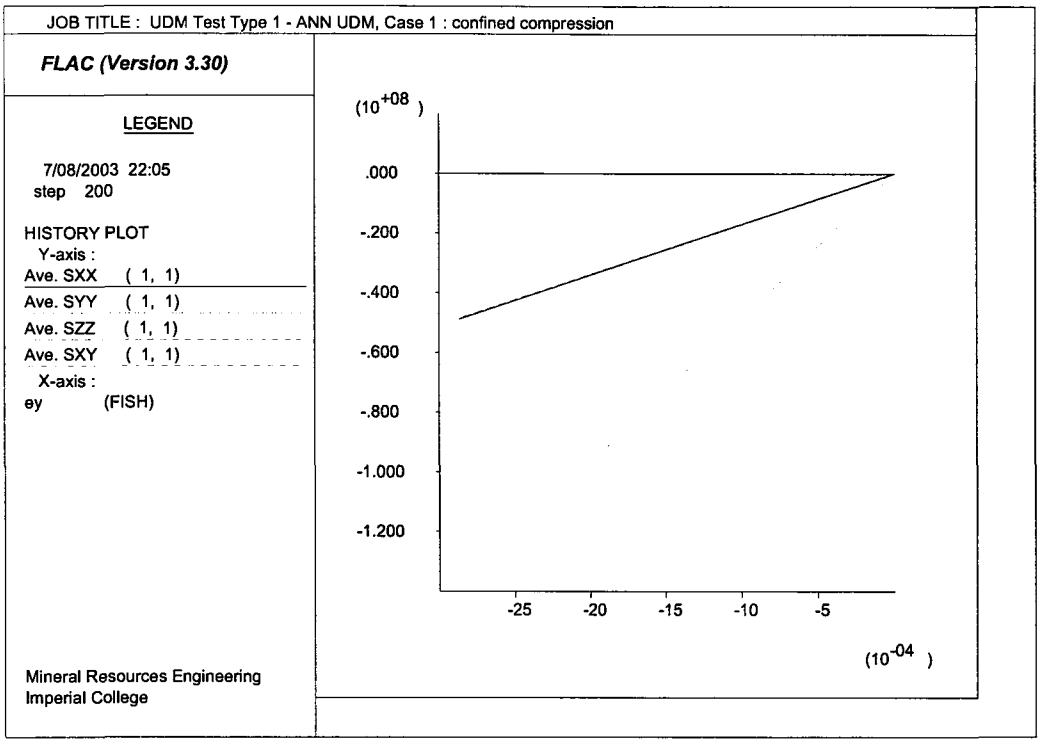


Figure 5.9: Results for Case 1 loading with ANN UDM: stress tensor components versus normal strain in y direction. Gradients of these lines are 48.5 GPa for σ_{yy}, 17 GPa for σ_{xx} and σ_{zz}. Corresponding values with E = 40 GPa and ν = 0.25 as prescribed are 48 GPa and 16 GPa. $\sigma_{xy} = 0$, as expected (horizontal line).

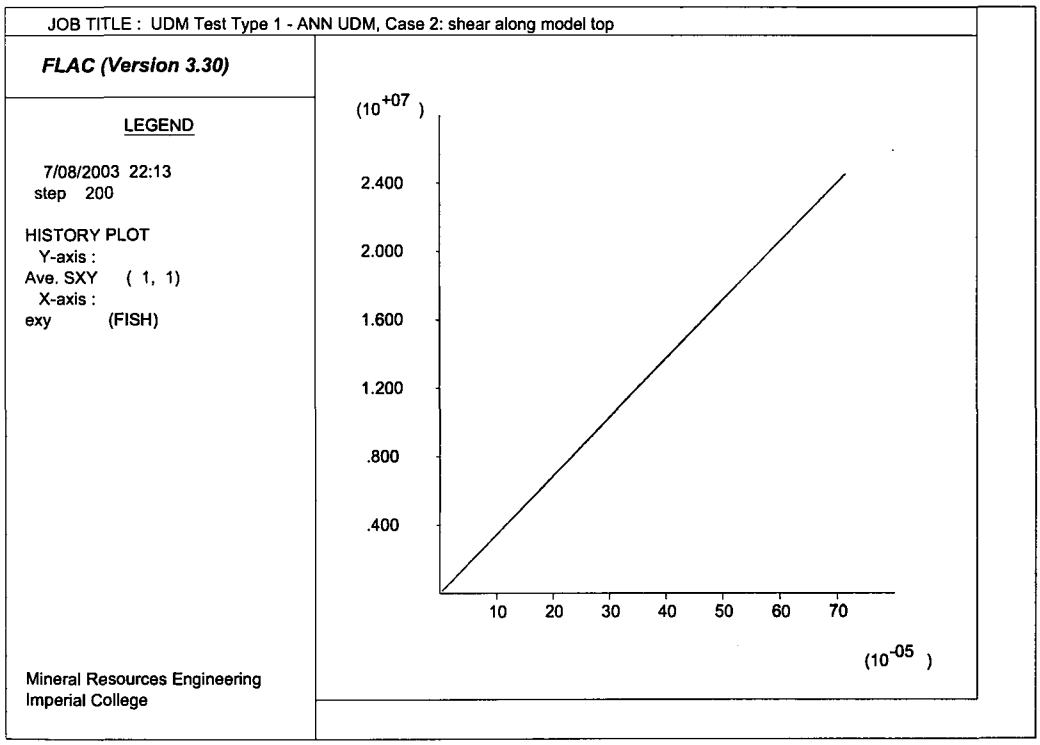


Figure 5.10: Results for Case 2 loading with ANN UDM: shear stress versus [tensorial] shear strain. Gradient of the line is 34.4 GPa against an expected value of $2G = 32$ GPa.

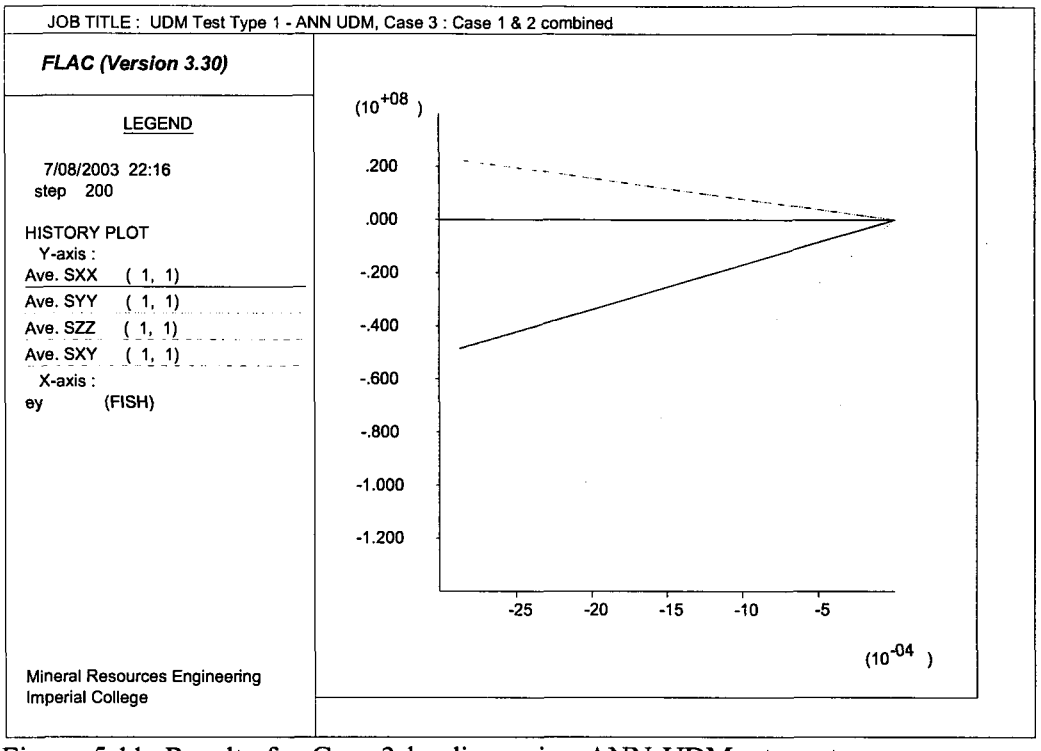


Figure 5.11: Results for Case 3 loading using ANN UDM: stress tensor components versus strain in y direction. Gradients of the lines are consistent with component loading cases (Figures 5.9 and 5.10)

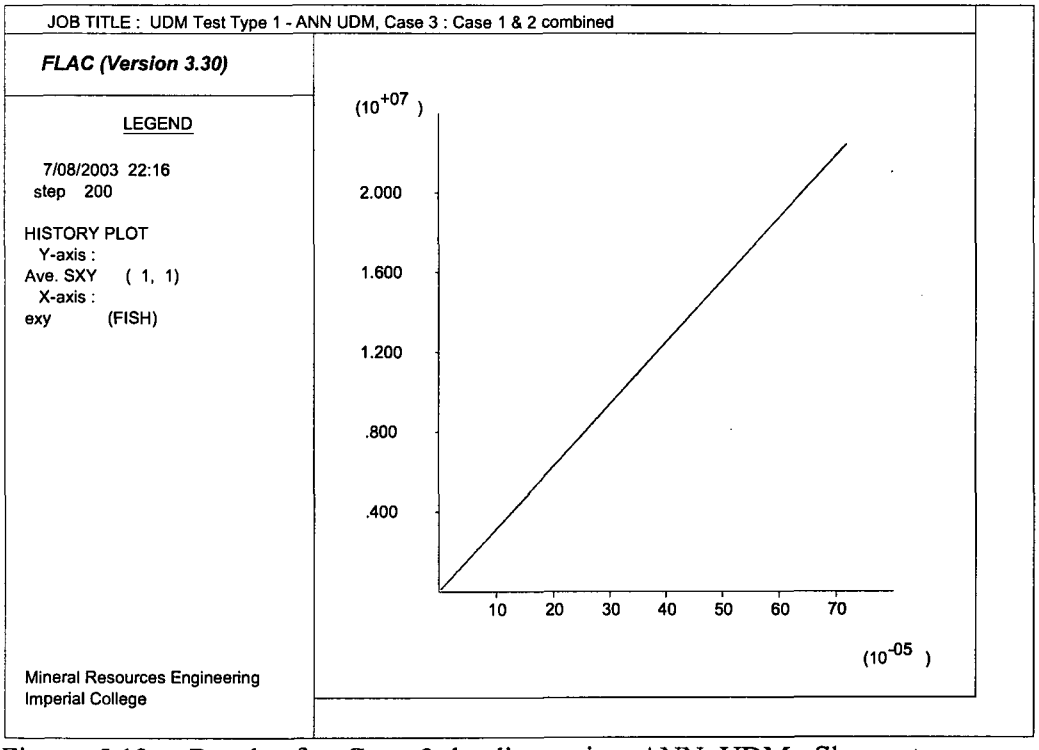


Figure 5.12 : Results for Case 3 loading using ANN UDM: Shear stress versus [tensorial] shear strain. Gradients of the lines are consistent with component loading cases (Figures 5.9 and 5.10)

In the case of Figure 5.9, the gradients of the curves are 48 GPa for σ_{yy}, 16 GPa for σ_{xx} and σ_{zz} (which are values consistent with the expected values of Lamé's constants with E=40 and ν=0.25) or zero for σ_{xy}. In the case of Figure 5.10, the gradient of the plot is 2G which, again, is expected behaviour. The gradients of the curves for Figures 5.11 and 5.12 for the case of the combined strain paths of Cases 1 and 2, are consistent with their component cases.

From the results of the ANN UDM performance in verification Test Type 1 the first observation that can be made is that for all cases, the response is linear – as expected. In comparison with the isotropic linear elastic model defined and on which it was trained, the ANN UDM performance demonstrates that there is good agreement. However, there are some differences when the results are examined in detail. In case 1 loading, the value of λ_1 that can be back-calculated from the performance curves is over-predicted by 0.5 GPa (against a target value of 48 GPa), and the value of λ_2 is over-predicted by 1.0 GPa (against a target value of 16 GPa). For case 2 loading, the ANN UDM leads to over-prediction of the shear modulus by approximately 1.2 GPa. The target value is 16 GPa, meaning a 7.4% over-prediction in shear stiffness. Overall these are good results under verification Test Type 1.

Test Type 2

Test Type 2 uses a 10 x 10 finite difference zone grid, with confining pressure of 10 MPa applied at the sides of the model. The performance of the model is measured by the degree of symmetry breaking observable in the performance curves for these two locations. It is to be remembered that even when using the in-built constitutive model in both the LHS and RHS portions of the grid, symmetry breaking performance will be observable eventually, as explained in earlier sections.

The results for the ANN UDM under Test Type 2 are presented in Figures 5.13 to 5.16. Figure 5.13 shows the magnitudes of normal stress components over a 500 time step simulation. For the first 50 time steps of the simulation, there is no discernible difference in the performance curves for the two constitutive models. After 50 time steps, divergence of the respective performance of the constitutive models emerges, and becomes greater as the number of time steps increases for σ_{zz} and σ_{yy}. The curves for σ_{xx} show good agreement over all 500 time steps, however the magnitude of σ_{xx} is constrained at the sides of the grid to model the confining pressure for this simulated triaxial test. The general trend of the difference between the two sets of curves is that the ANN UDM generally produces higher values of the stress components, indicating, as in verification Test Type 1, minor over-prediction of stiffness.

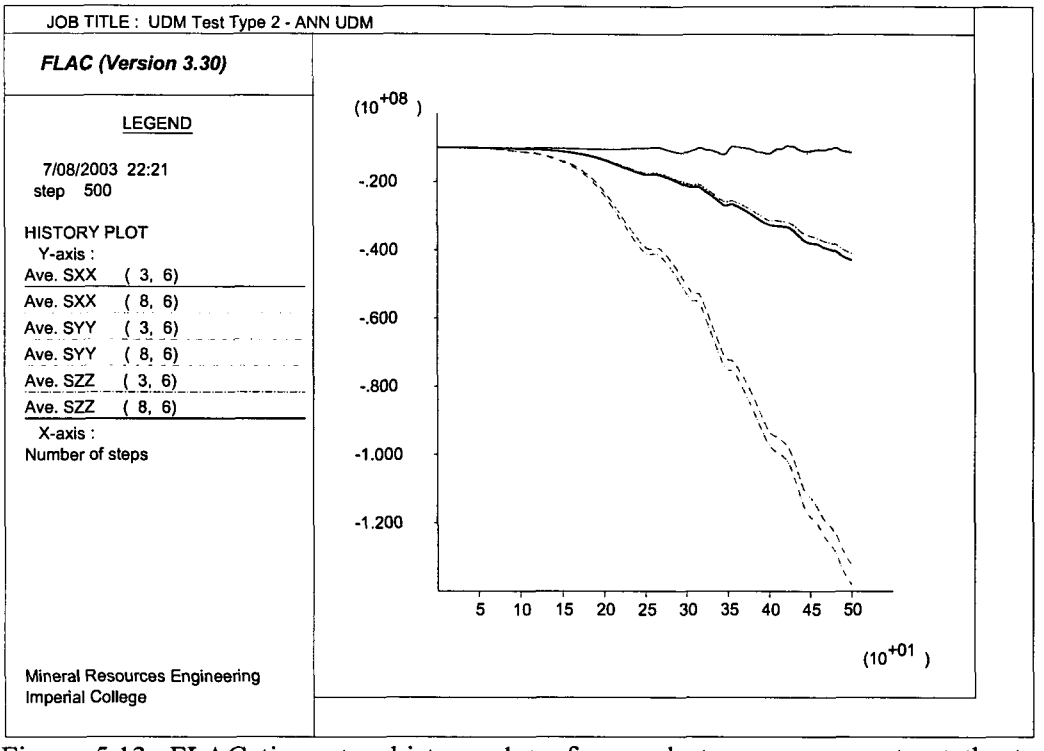


Figure 5.13: FLAC time step history plot of normal stress components at the two monitored locations in the model in Test Type 2 showing good, but not perfect agreement; the ANN UDM curves indicate a minor over-prediction of stiffness (consistent with the findings of Test Type 1 verification runs).

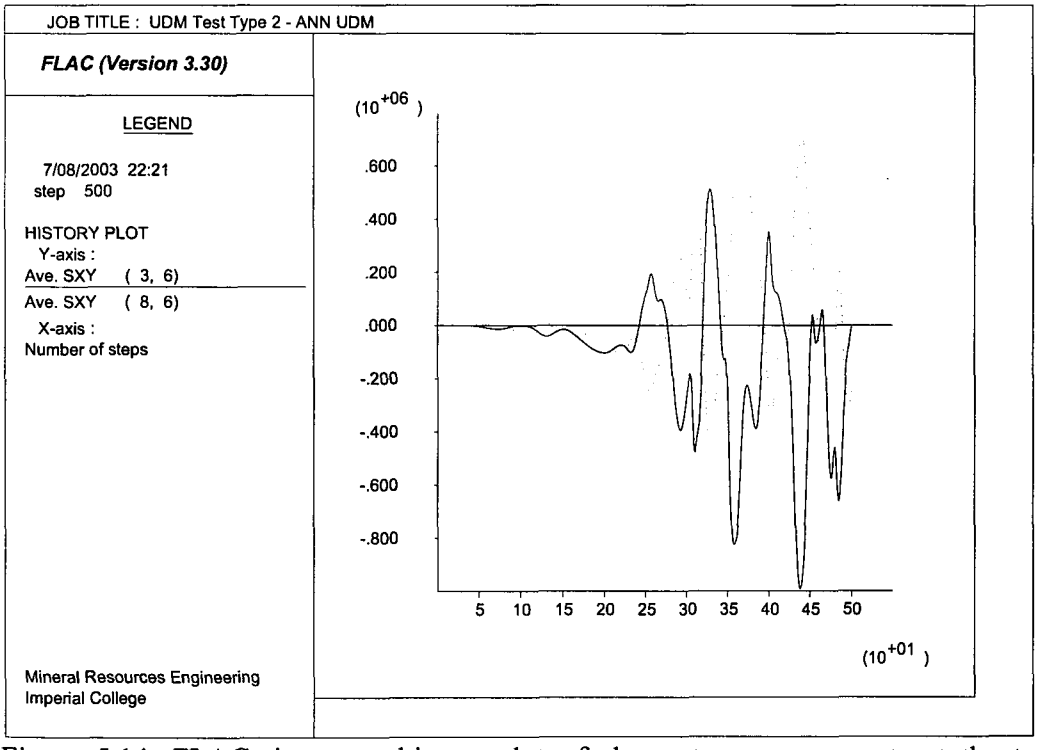


Figure 5.14: FLAC time step history plot of shear stress components at the two monitored locations in the model in Test Type 2. Although there is qualitatively similar behaviour presented, symmetry breaking is apparent from 50 time steps onward.

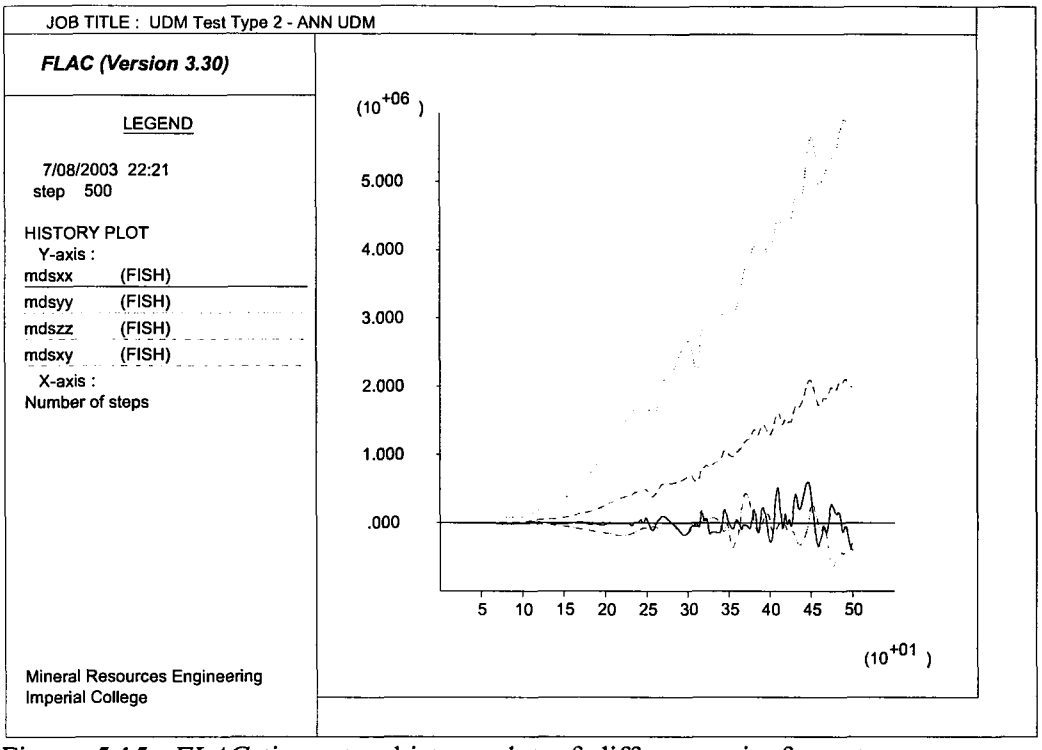


Figure 5.15: *FLAC* time step history plot of differences in four stress component magnitudes (mds**) monitored at two symmetrically located zones within the model up to 500 time steps for the Test Type 2 UDM verification model with the ANN UDM used on RHS of the model. Trends in disparities are consistent with over-prediction of stiffness identified in Test Type 1 verification.

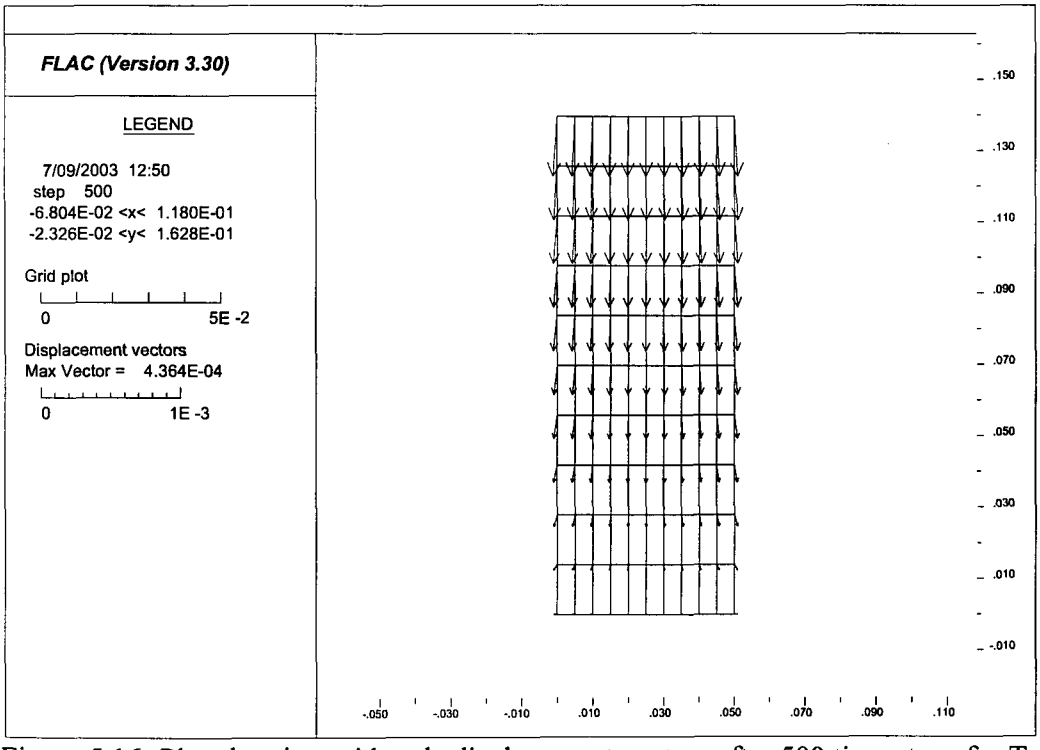


Figure 5.16: Plot showing grid node displacement vectors after 500 time steps for Test Type 2 UDM verification model with the ANN UDM used on the RHS of the model.

Figure 5.14 compares shear stress magnitudes between the two, symmetrically positioned locations within the model. Ideally, these curves should be mirror images of each other about the $\sigma_{xy} = 0$ axis. It can be seen that the while there is a 'kind of' symmetrical response between the two curves, the line if symmetry shifts downward as the number of time steps increases. The peaks of the curve for Ave. SXY (8,6) (for the ANN UDM) are lower below $\sigma_{xy} = 0$ than the peaks of Ave. SXY (3,6) (for the in-built constitutive model) are higher above it. This means, again, that the ANN UDM leads to over-prediction of shear stress, a result of a slightly higher 'learnt' shear modulus than the target value.

Figure 5.15 plots the differences between the stress component values at the two monitored locations. These curves show that, for σ_{zz} and σ_{yy} there is a minor problem with accuracy with the ANN UDM, although the magnitudes of the maximum errors are below 5% for normal stress components. The maximum error for the shear stress components is 7%.

Figure 5.16 shows the displacement at each of the grid points of the model at 500 time steps. In the y direction, the largest difference in displacement between symmetrically located grid points throughout the model is 4×10^{-7} metres. This means that differences in displacement due to the ANN UDM are less than 0.1% of the maximum displacement, around 0.0003% of the longest dimension of the model.

Extension to Test Type 2

It is instructive to examine how the ANN-UDM performs during a regime where the specimen is unloaded. This was simulated by reverse of the sense of the boundary velocity of the grid nodes at the top of the simulated specimen. The model was then stepped for a further 800 cycles. For these extended conditions of Test Type 2, Figure 5.17 shows a plot of normal stress in the y direction (at monitoring points) versus normal strain in the y direction, the latter being computed from displacement over the full height of the simulated specimen.

Ideally, the unload portion of each of the curves should lie over their loading portion. It can be seen that this is not the case for either the in-built linear elastic constitutive model nor the ANN UDM, but in both cases, the unload portions oscillate around their loading portions. The higher apparent stiffness of the ANN UDM is also apparent in this plot, the average gradient for the curve labelled Ave SYY (8,6) being higher than that for the curve labelled Ave SYY (3,6). The ANN UDM also exhibits a little hysteresis, or permanent 'axial' strain when passing back across the σ_{yy}=0 line. However, overall the performance of the ANN UDM on unload is broadly as should be expected, subject to the higher apparent stiffness issues discussed for Test Type 1.

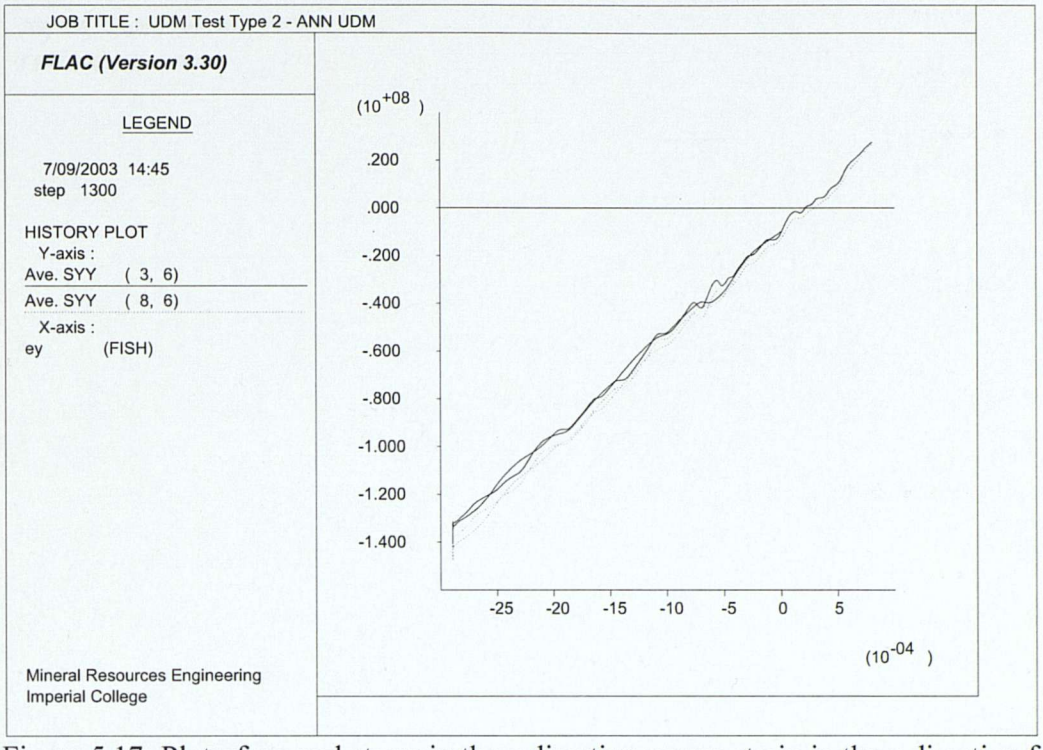


Figure 5.17: Plot of normal stress in the *y* direction versus strain in the *y* direction for the symmetrically positioned monitoring points including unload and tension test regime. The results indicate good agreement and consistent behaviour, but ideally the curves should overlie one another.

Figure 5.18 shows that, as should be expected, when the model is unloaded, it is unloaded broadly symmetrically. It also shows that as the loads are reduced, so the disparity between values for the LHS of the model and the RHS of the model reduce.

Another convincing aspect to Figure 5.18 is that all six curves cross at the point corresponding to the initial, stress free, state of the specimen, when the simulation 'started'. This confirms that the ANN UDM is capable of following conventionally adopted stress paths in an expected manner.

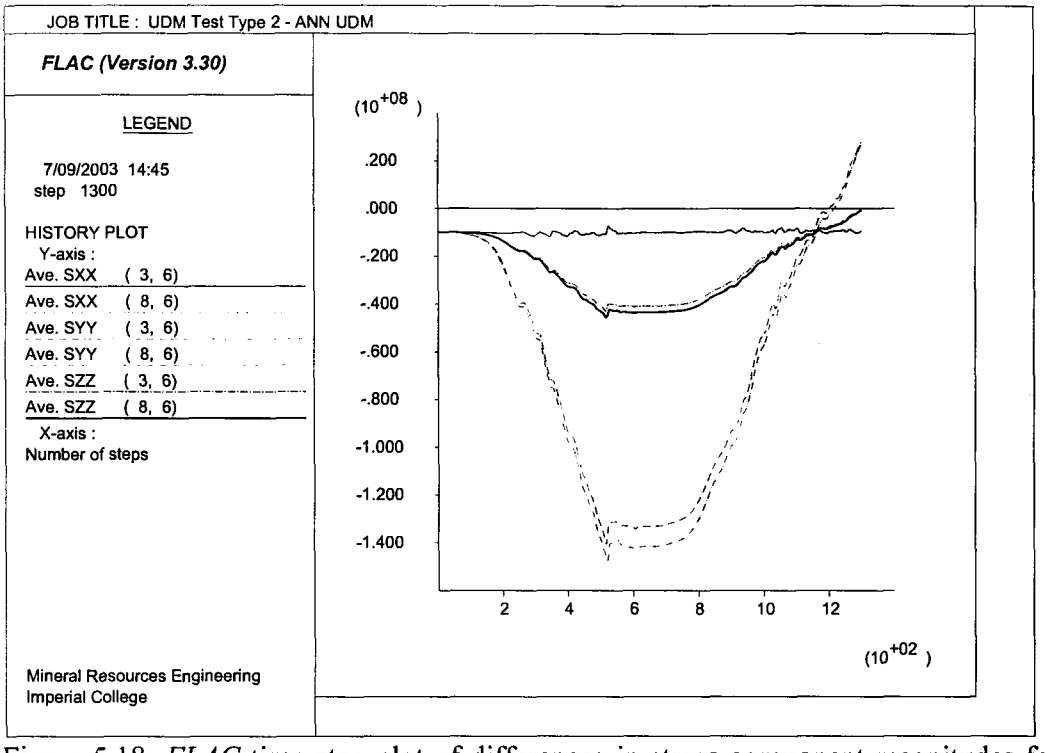


Figure 5.18: *FLAC* time step plot of differences in stress component magnitudes for loading to 500 time steps and unloading for a further 800 time steps for the Test Type 2 verification test. Crossing point of all six curves at around 1100 time steps indicates that the initial stress-free state is recovered on the unload cycle. Disparities between corresponding stress components also reduce as unloading progresses.

The fact that the ANN UDM exhibits just as much symmetry in its unloading behaviour as the in-built constitutive model, running under exactly the same conditions, suggests that the small inaccuracies associated with the ANN UDM are a result of the quality of the ANN training process, rather than a result of a numerical error in implementation of the UDM. This is supported by the reduction in disparities between the stress components, as the simulated specimen is unloaded.

The best available explanation for the higher apparent stiffness of the material represented by the ANN UDM at the time of writing is that the ANN training could be improved. The inaccuracies, expressed as percentages, for the normal stress components are smaller than those for the shear stress components. This suggests that perhaps, in ideal circumstances, more exemplars of the behaviour of the material represented by the ANN UDM need to be included in the ANN's training data set.

The results of verification Test Type 2 strongly support the concept that an ANN UDM is competent as a constitutive model, for the material its weight set represents. Importantly, verification Test Type 2 also demonstrates that ANNs can indeed be reliably incorporated within finite difference numerical codes.

Test Type 3

The results of Test Type 2 are confirmed by the results of Test Type 3, which is essentially a re-run of Test Type 2 with an asymmetric mesh.

Again, the magnitudes of differences between stress component values at the symmetrically positioned monitoring locations are larger than if the ANN had not been used (Figure 5.19). The magnitudes of maximum discrepancies are, again, around 5% for normal stress components and 7% for shear stress components. The magnitudes of discrepancies become greater, as the imposed displacement in the y direction becomes greater, and become smaller as the boundary velocities are reversed in the y-direction. Hysteresis effects associated with the ANN UDM appear to be lesser than in the case of verification Test Type 2 which has a regular grid.

Grid point displacements throughout the model have a coherent trend in magnitude and direction after 500 loading time steps (Figure 5.20). Differences in displacements for approximately symmetric locations across the grid are still small.

In comparison with Test Type 2 verification, the results of verification Test Type 3 show no significant deviation. This suggests that the coding of the ANN UDM does not introduce errors that would cancel out over a symmetric, regular grid. As such, the results of this test can be said to be successful.

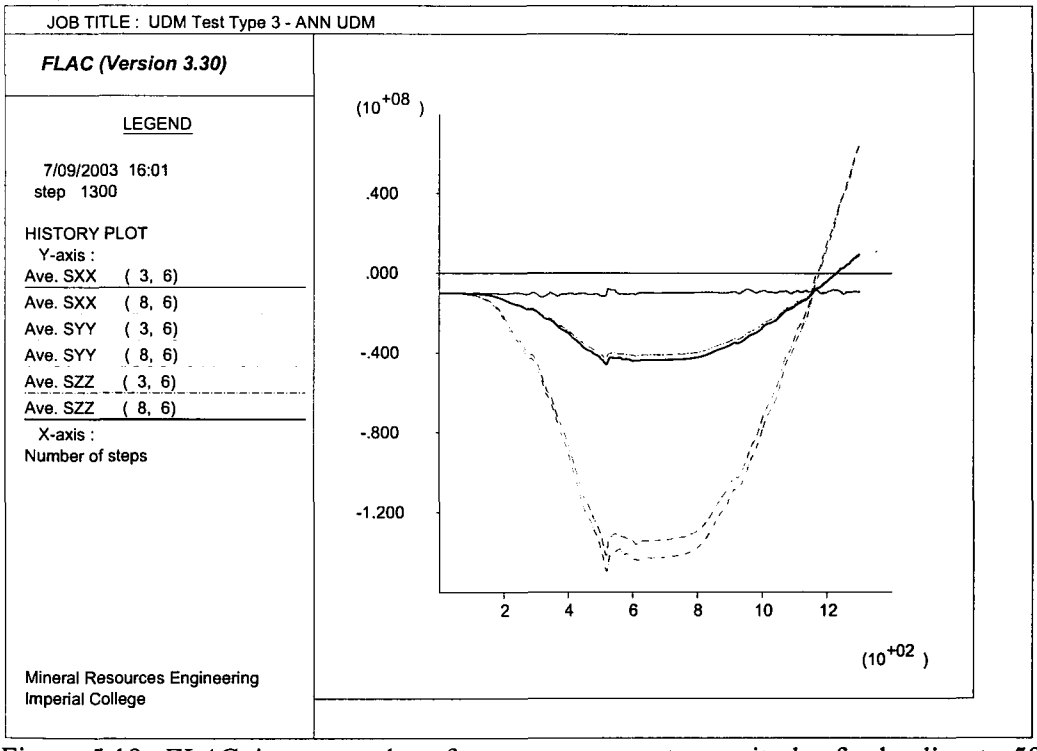


Figure 5.19: *FLAC* time step plot of stress component magnitudes for loading to 500 time steps and unloading for a further 800 time steps for the Test Type 3 verification test with the ANN UDM used on the RHS of the model. This plot (and others not presented) indicates complete consistency with the results of Test Type 2 verification.

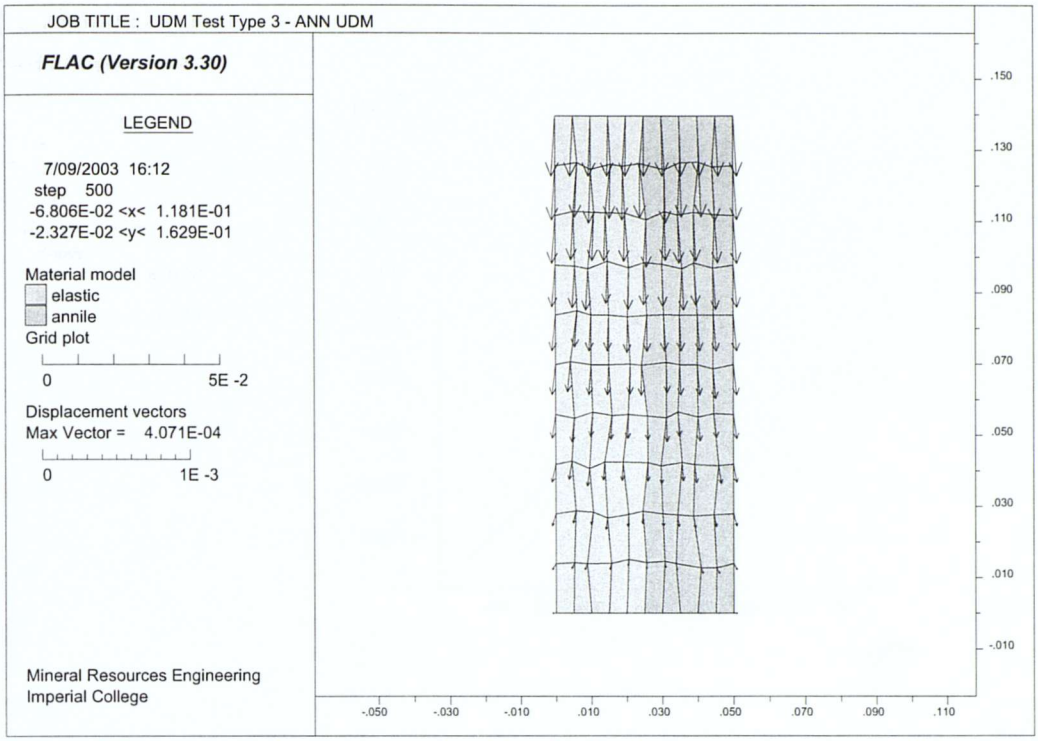


Figure 5.20: Plot of *FLAC* grid and grid node displacement vectors at 500 time steps for the Test Type 3 UDM verification model with the ANN UDM used on the RHS of the model. Grid point displacements throughout the model have a coherent trend and the differences in displacement magnitudes and directions are small, across the model.

5.4.4 Further verification checks

In order to investigate the reasons behind the over-prediction of stiffness of the ANN UDM over the in-built isotropic linear elastic model, several additional investigations were made, based on Test Type 2 of the previous Section.

Firstly, the ANN UDM was modified to also calculate the principal stress component increments that would result using isotropic, linear elastic equations. The principal stress components were then defined as **f_prop** variables within the ANN UDM so that they effectively became grid variables (As four Mode 2 invocations of a UDM are made for each zone, the values stored in these grid variables actually apply to the fourth triangle of each finite difference zone). Then both the ANN and elastic principal stress components were monitored as history variables for the fourth triangle in the ANN UDM monitored zone (8 zones along from the LHS of the model, 6 zones from the base of the model). Test Type 2 was then re-run and then history plots of the elastic stress component values versus the ANN predicted principal stress components were made. These are presented in Figures 5.21, 5.22 and 5.23 for $\Delta\sigma_1$, $\Delta\sigma_2$ and $\Delta\sigma_3$ respectively.

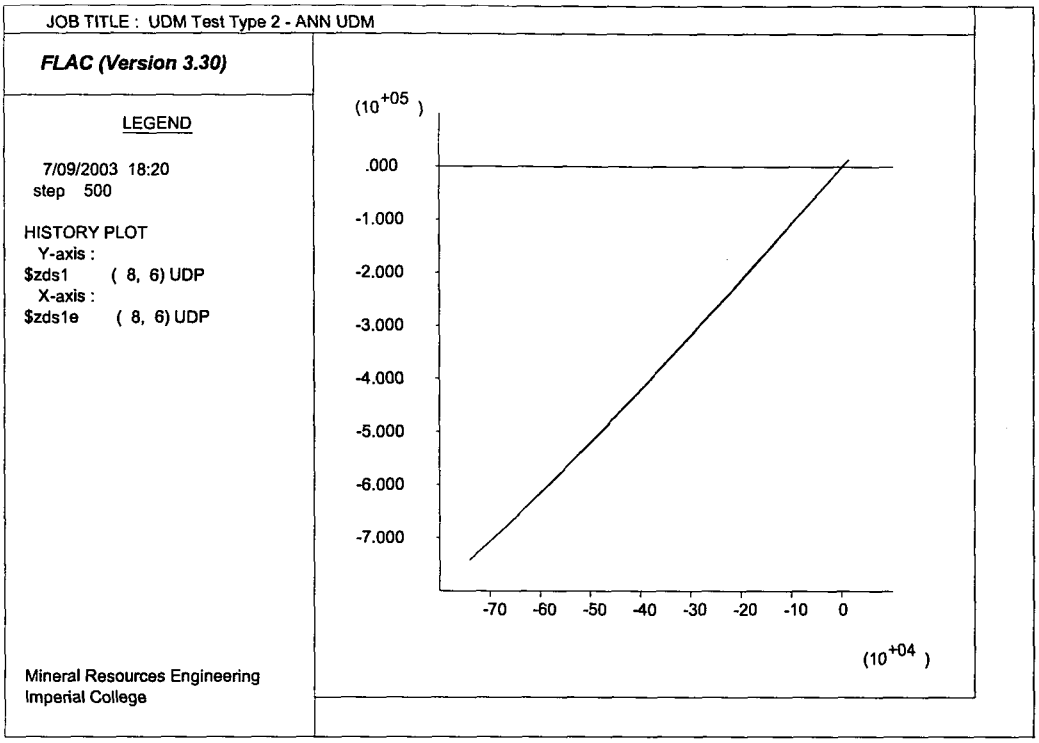


Figure 5.21: Plot showing values of major principal stress increment from ANN UDM ($zds1) versus elastic values ($zds1e), both computed within the ANN UDM, for zone (6,8) within the Test Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value.

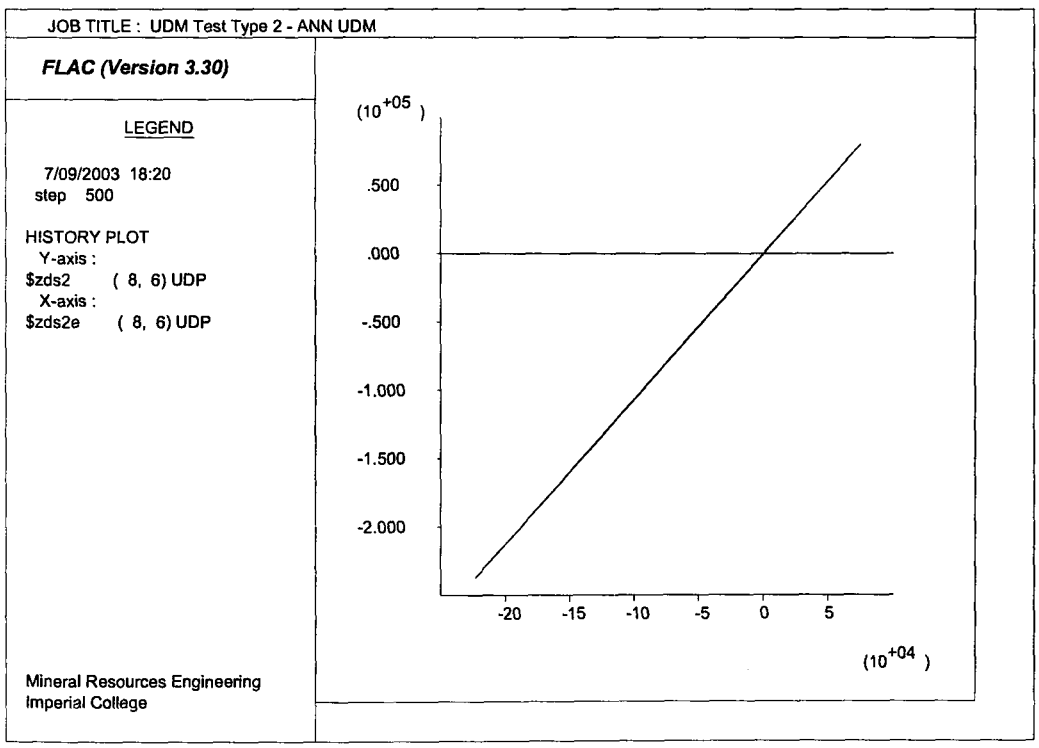


Figure 5.22: Plot showing values of intermediate principal stress increment from ANN UDM ($zds2) versus elastic values ($zds2e), both computed within the ANN UDM, for zone (6,8) within the Test Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value.

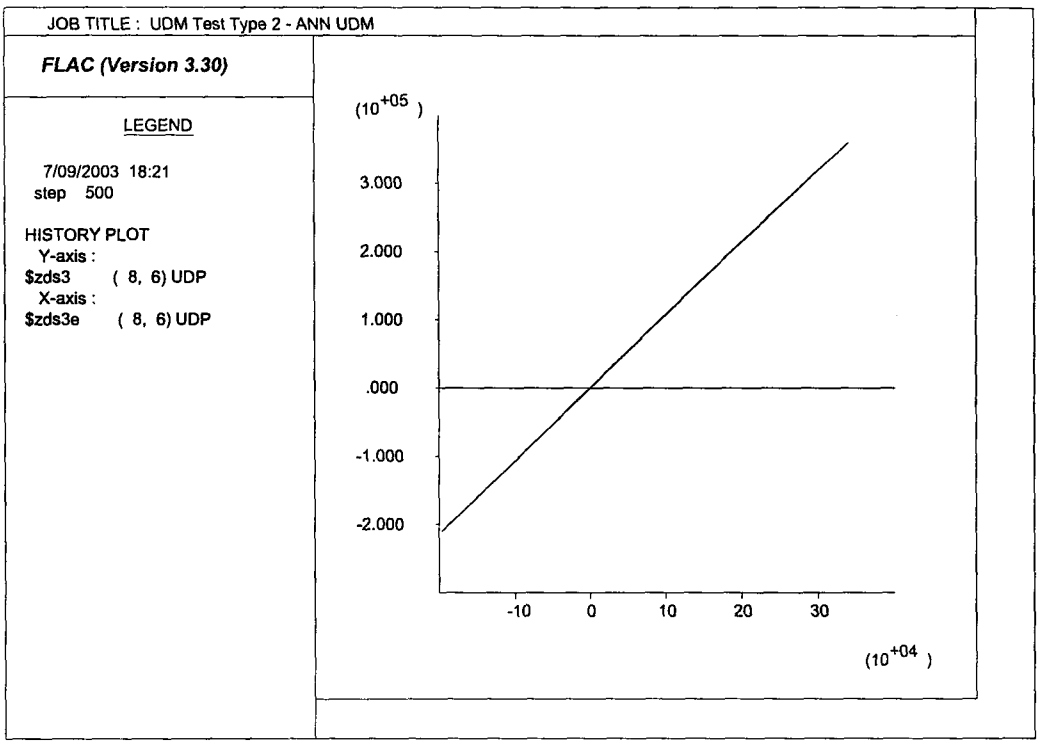


Figure 5.23: Plot showing values of minor principal stress increment from ANN UDM ($zds3) versus elastic values ($zda3e), both computed within the ANN UDM, for zone (6,8) within the Test Type 2 verification model. Bringing a straight edge up to the curve presented highlights the minor non-linearity in the ANN estimated value.

These plots demonstrate, again, that the ANN UDM makes sensible predictions of the principal stress increments in comparison to those that would be expected with the elastic model it represents. However, if the curves are examined in closer detail, for example, by bringing a straight edge up to each of the curves in turn, a small degree of non-linearity is evident. This is especially noticeable in Figure 5.21 and to lesser extents in Figures 5.22 and 5.23.

In the case of Figure 5.21, for the major principal stress increment, it is clear that, at extremes of stress increment magnitude, that is, towards the lower LHS of the plot, the ANN UDM prediction of the stress increment is slightly higher than would be expected. The source of this non-linearity is the activation function of the neurons within the ANN. A similar occurrence discussed and solved in Section 5.8 suggests that it is not sensible to pursue elimination of this non-linearity; Section 5.8 explains that encoding pure linear behaviour into an ANN that has non-linear activation functions is actually quite difficult to achieve.

The over-prediction of stress component increments, and thus of the stress components themselves, will be difficult, perhaps impossible, to eliminate while non-linear neuron activation functions are used within the ANN. It is at this stage that the reader is reminded that that the purpose of the verification tests is to establish whether

or not the ANN can be effectively used as a constitutive model within a geomechanical code. Adopting linear elastic constitutive behaviour forms a useful, and well understood, benchmark where the suitability of the ANN can be isolated and identified.

After consideration of the evidence presented thus far, the suitability of the ANN in this new role has been established. However, encoding linear elastic constitutive behaviour within the weight set of an ANN is far removed from its ultimate planned purpose. This will require embodying more complex, non-linear constitutive behaviour, closer to that observed for real geotechnical materials. The advantages in the ANN constitutive modelling approach arise when the non-trivial constitutive behaviour of real geological materials are to be embedded within the ANN structure – as is demonstrated in the next Section.

5.5 Training an ANN to learn real constitutive behaviour

The material presented thus far within this Chapter has extended and considerably improved from that reported by Millar and Calderbank (1995) and Millar *et al.* (1995). In this Section, the practicalities of training an ANN to embody the constitutive behaviour of Crosslands Sandstone rock (the data set also used in the earlier studies) are reviewed and verification of the resulting ANN UDM within the *FLAC* environment is demonstrated.

ANNs with various formulations have been found to be capable of learning the constitutive behaviour of a wide variety of natural geotechnical materials, as is reported by Amorosi and Millar (1996). Demonstration that the ANN works for Crosslands Hill sandstone rock thus provides support for its adoption for the much wider range of geomaterials discussed by Amorosi and Millar.

5.5.1 Crosslands Hill Sandstone – Conventional Treatment

Alemenara-Chau (1992) conducted six triaxial compression tests on Crosslands Hill sandstone rock, adopting the ISRM Suggested Method (Brown, 1985). The tests were conducted in a servo-controlled compression testing machine, equipped with a Hoek cell. Linear variable displacement transducers were used to measure the axial deformation of the specimens under test by setting them against the machine platens. Deformation in the radial direction was measured using a four-armed cantilever assembly. Observations of diametrical deformation were taken across each pair of opposing cantilever arms and averaged. The cantilever assembly was immersed in the hydraulic fluid providing the confining pressure on the specimen. The measuring points of the cantilever arms were set at the mid height of the specimen, in specially designed seats located on the outside of the sheath isolating the hydraulic fluid from the specimen. The velocity of convergence of the platens of the testing machine was

set nominally at 0.084 mm/min for each of the tests, this quantity being controlled using the servo-control mechanism. Each of the six tests was conducted with constant Hoek cell pressures, in each case, this also being servo-controlled.

The results of each of the triaxial tests are reported in Figures 5.24 and 5.25. Within Figure 5.24, the curves of axial stress versus axial strain exhibit a linear portion, varying between 18% and 79% of peak strength for the test conducted at the lowest confining pressure to 7% and 58% of peak strength for the test conducted at the highest confining pressure. As expected, Figure 5.24 shows that the peak strength increases as the confining pressure increases.

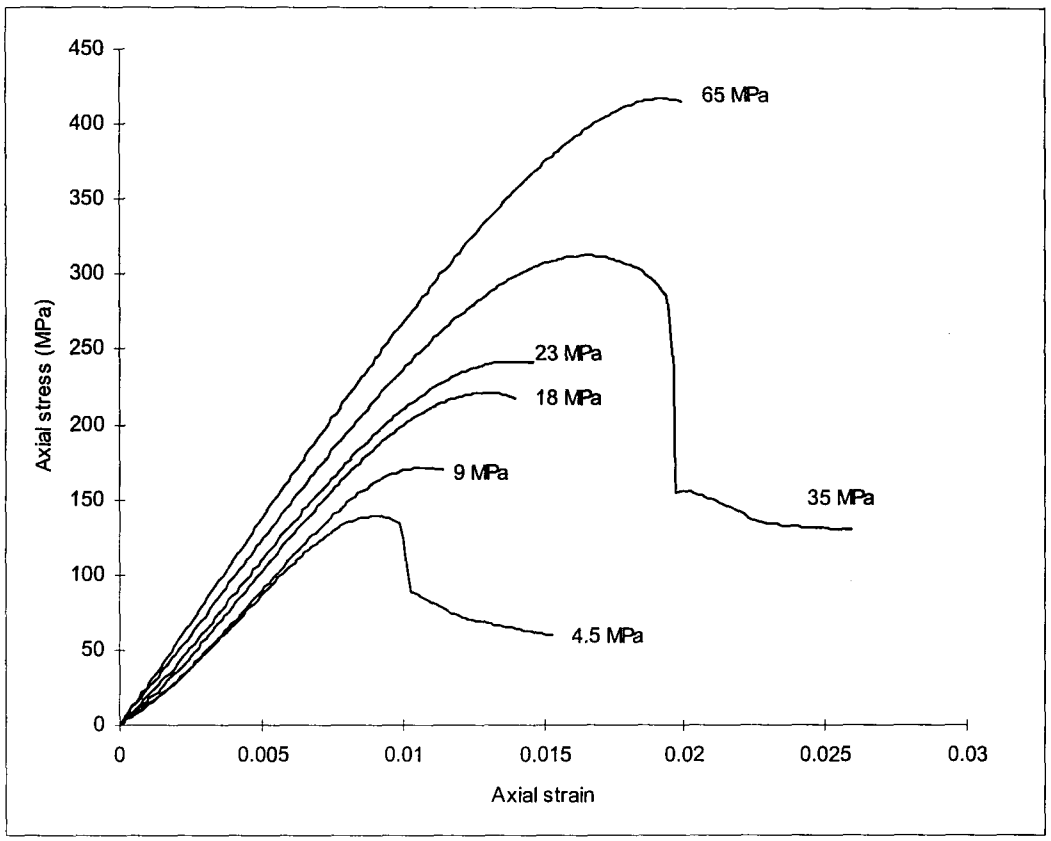


Figure 5.24: Axial stress versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve). It is evident that in the immediate post peak regions of the tests conducted with confining pressures 35 and 4.5 MPa, the testing system was not under full control.

It is also evident that the stiffness of Crosslands Hill sandstone rock increases as the confining pressure increases. Two of the tests were continued into the post peak region of the complete stress-strain curve (those conducted at 4.5 and 35 MPa). From these two curves, it can be said that the rock exhibits type I post peak deformation behaviour, i.e. the strain increases monotonically, and that it fails in a brittle manner, as the gradient of the post peak curves become noticeably steeper, a little after the peak strength has been surpassed. The stress strain curves for confining pressures of

4.5 and 35 MPa also show a distinct transition to shear deformation along newly created macro-fractures.

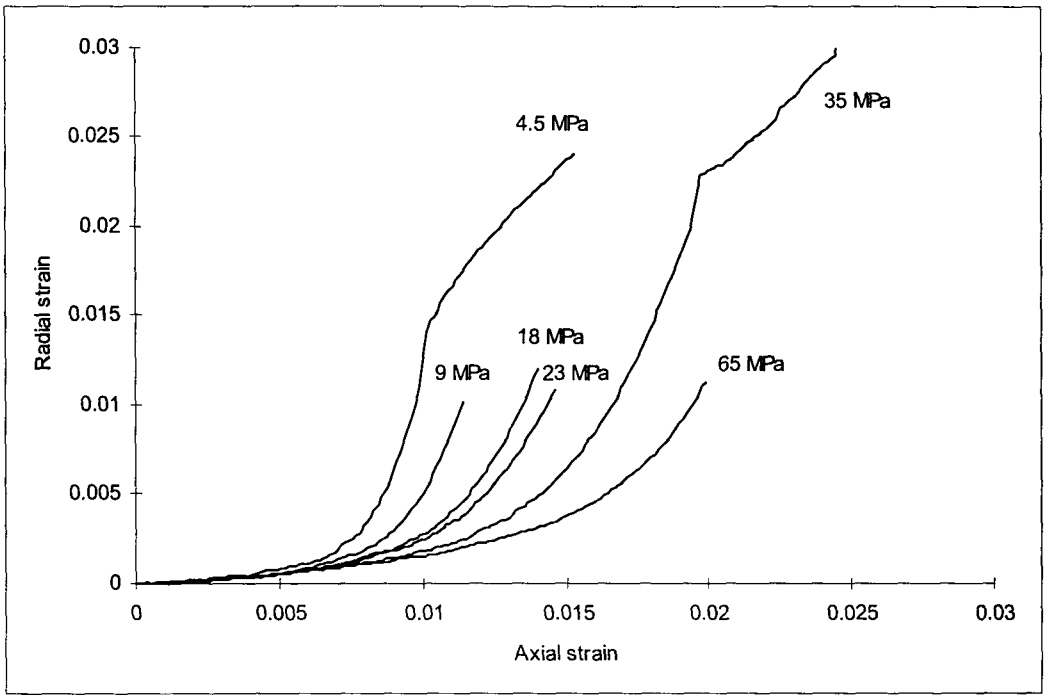


Figure 5.25: Negative radial strain (i.e., extensile strain plotted as positive for convenience) versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve).

The radial strain versus axial strain curves plotted in Figure 5.25 show that, as the confining pressure increases, the rock becomes less compliant laterally. This is true of both the linear region of the curve, close to the origin, but also as the specimens approach and pass the point of yield. Each of the specimens taken alone exhibits more compliant deformation behaviour laterally, as the axial strain increases. This leads to dilation, or expansion of the specimen volume as each test proceeds toward and after the point of peak strength. Figure 5.26 confirms that the magnitude of dilatant volumetric strain reduces as the confining pressure increases, as expected. For each individual curve of Figure 5.26, the volumetric strain is initially positive meaning that specimen volume is reduced in comparison to the reference (unloaded) state. At the point of peak strength, the volumetric strain is close to zero, and thereafter, the specimens expand with further straining in the axial direction.

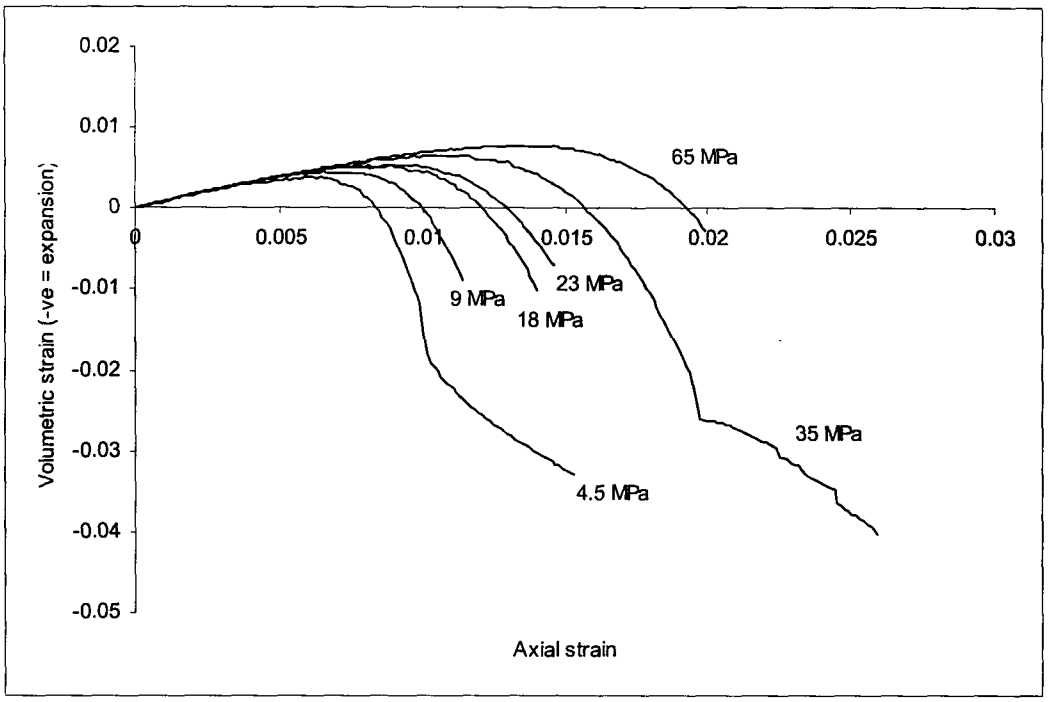


Figure 5.26: Volumetric strain versus axial strain plots for the 6 triaxial compression tests on Crosslands Hill sandstone rock (Confining pressure shown next to curve). Dilatant volumetric strain lies below the horizontal axis.

Crosslands Hill sandstone is a relatively 'well behaved' rock type in that its experimental behaviour conforms to the classic model of rock constitutive behaviour in many respects. In the terms explained by Hoek (2003), the rock may be described as elastic–brittle–strain softening. However, even in the pre-peak region of the experimental curves describing the constitutive behaviour, the rock cannot be said to behave in a linear elastic manner. Most obviously, none of the experimental curves can be seen to lie perfectly on a straight line. If we were to confine ourselves to description of the 'elastic' portion of the rock behaviour, a stress dependence of the elastic parameters would need to be taken into consideration, as Figure 5.27 and Table 5.1 show.

Table 5.1: Summary of Triaxial test results on Crosslands Hill sandstone.

Test	Confining pressure (MPa)	Axial velocity (mm/min)	Ax. Strain rate (/s)	Peak strength (MPa)	Young's Modulus (GPa)	Poisson's Ratio	Dilation Angle (o)
1	4.5	0.082	0.77	139	18.9	0.24	51.2
2	9	0.082	0.77	171	20.2	0.19	50.0
3	18	0.075	0.71	221	21.7	0.16	46.5
4	23	0.080	0.75	242	22.6	0.14	42.9
5	35	0.082	0.76	313	23.8	0.13	47.8
6	65	0.077	0.77	418	26.9	0.12	44.5

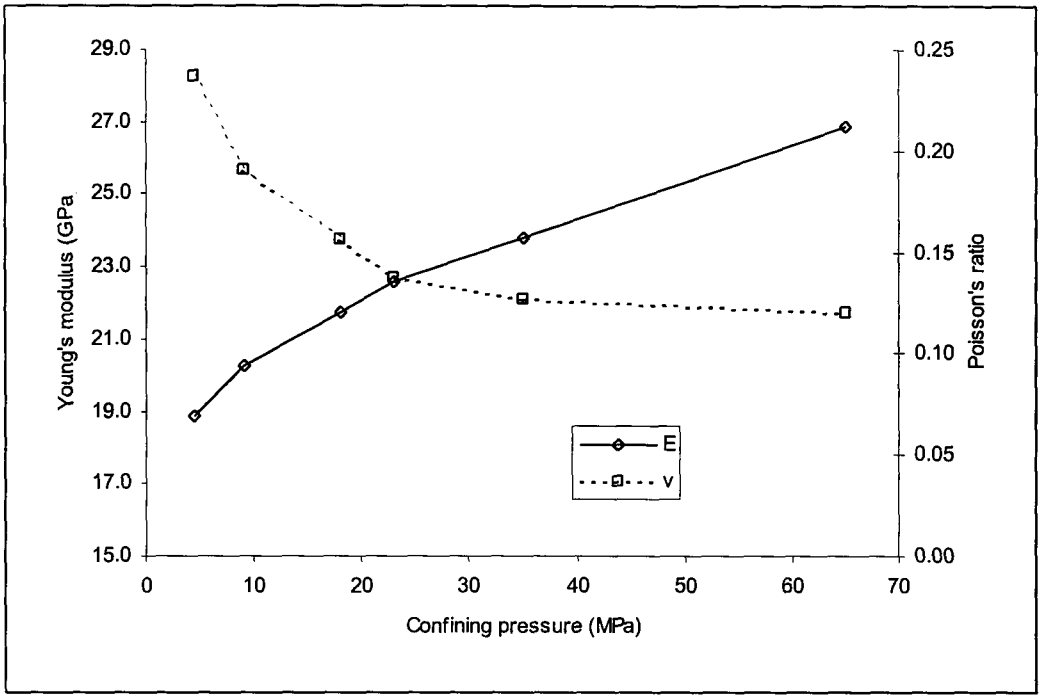


Figure 5.27: Dependence of elastic parameters on confining pressure for Crosslands Hill sandstone rock.

The experimental curves demonstrate that the peak strength of the specimens under test increase as the confining pressure increases. However, the relation is not linear.

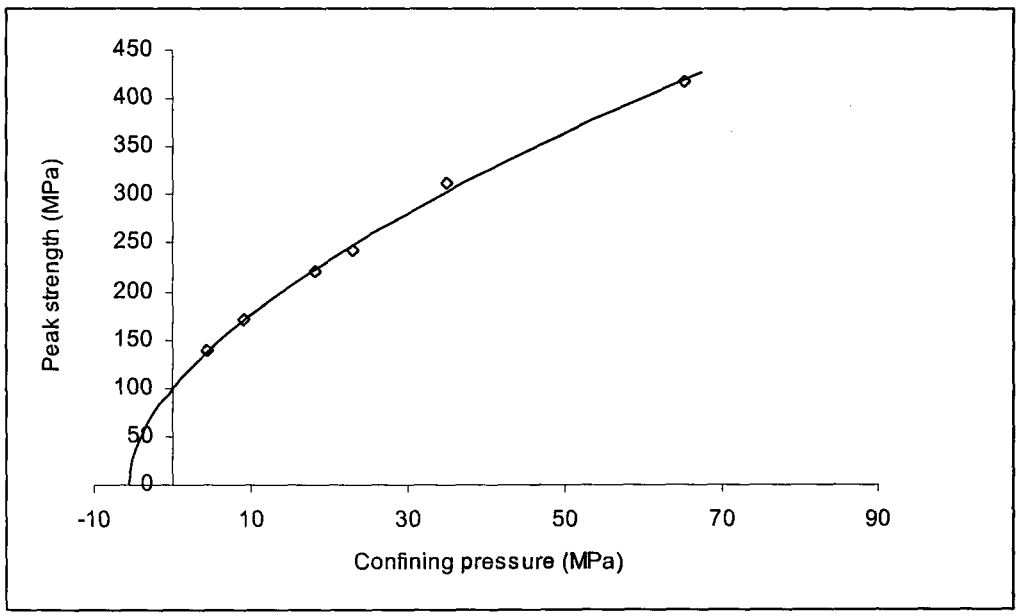


Figure 5.28: Plot of peak axial stress versus test confining pressure for Crosslands Hill sandstone (discrete points). Continuous line shows the Hoek-Brown failure criterion can be adequately fitted to the test data.

Figure 5.28 shows a good fit (R^2 = 0.9956) for the peak strength data using the Hoek-Brown failure criterion (Hoek & Brown, 1980) with m = 17.8, s = 1 and σ_c=100 MPa. The curve has the form:

$$\sigma_1 = \sigma_3 + \sigma_c\left(m\frac{\sigma_3}{\sigma_c} + s\right) \tag{5.1}$$

and is clearly non-linear.

An important parameter in characterising the post peak shear dilatancy behaviour of rock is the so-called dilation angle, ψ, ideally determined as shown in Figure 5.29 using a plot of volumetric strain versus axial strain. When comparing the method of ideally determining the parameter, with the experimentally determined curves, of volumetric strain versus axial strain (Figure 5.26), the non-linearity in the latter makes it difficult to choose a representative value. Table 5.1 shows values of the dilation angle determined along the linear portion of the volumetric strain curves, starting from the point of zero volumetric strain.

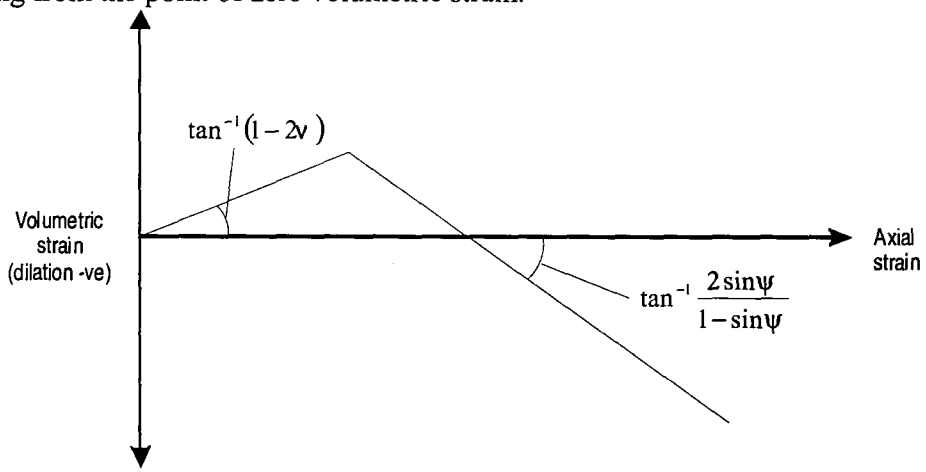


Figure 5.29: Determination of shear dilatancy angle from triaxial test results (after, Vermeer and de Borst, 1984)

5.5.2 The FLAC constitutive model for Crosslands Hill sandstone.

The conventional approach in geomechanical numerical modelling is to choose a constitutive model, from those available in the software, that best follows the conceptual constitutive behaviour that is identifiable within the experimental data. In the case of *FLAC*, the strain-softening constitutive model is probably the best to adopt for Crosslands Hill sandstone. However, the strain-softening constitutive model within *FLAC*

- uses isotropic linear elasticity before the peak strength is reached,
- ignores any stress dependence of elastic parameters,
- ignores any non-linearity in the criterion describing when peak strength is reached, and
- allows for reduction in Mohr-Coulomb strength parameters and the dilation angle after the peak strength is reached, but these do not accommodate stress dependence.

Thus the constitutive model within *FLAC* will, at best, only be able to use a stylised representation of the constitutive behaviour of Crosslands Hill sandstone.

5.5.3 Preparing the Crosslands Hill sandstone triaxial test data for ANN training

The key design constraint in selecting a pattern format for the triaxial test data was that the resulting ANN had to provide for estimates of the new stress tensor components given the old stress tensor components and strain increments, that is: it had to work within *FLAC*. Consequently, in formulating the training pairs, consideration was given to the general form of constitutive models within *FLAC* as it was recognised that the straightforward format used with the elastic ANN (of Section 5.4) would be insufficient to capture the complexity of actual triaxial compression test data. *FLAC* represents general constitutive behaviour as:

$$\Delta\sigma_{ij} = M\left(\Delta e_{ij}, \sigma_{ij}, e_{ij}, \kappa\right). \tag{5.2}$$

The new stress state is found by adding the incremental stress tensor to the input stress tensor, that is:

$$\sigma_{ij}^{(t+1)} = \sigma_{ij}^{(t)} + \Delta\sigma_{ij}^{(t)} \tag{5.3}$$

For the case of an ANN modelling isotropic linear elastic constitutive behaviour, there was no need to include state parameters $\left(\sigma_{ij}, e_{ij}\right)$ nor a history parameter, κ. In the case of Crosslands Hill sandstone rock, state and history parameters are required to assist in the description of the non-linear behaviour. While the incremental formulation (5.2) and (5.3) is the one adopted in *FLAC*, there is no reason why the new components of the stress tensor cannot be estimated directly:

$$\sigma_{ij}^{(t+1)} = M\left(\Delta e_{ij}^{(t)}, \sigma_{ij}^{(t)}, e_{ij}^{(t)}\right). \tag{5.4}$$

Then, if necessary, $\Delta\sigma_{ij}$ can be recovered from:

$$\Delta\sigma_{ij} = \sigma_{ij}^{(t+1)} - \sigma_{ij}^{(t)} \tag{5.5}$$

Thus, a variety of formats for the ANN training exemplars were possible and an investigation was conducted to determine the most suitable. Five options were considered which varied in the type and number of input quantities and the type of output quantities. The formats can be distinguished primarily through determination of whether stress components or stress component increments are estimated with the ANN. These have been called direct and incremental formulations respectively. If strain increments exist within the input quantities as well as output quantities, the format has been called a fully incremental formulation. If not, the pattern format is referred to as a hybrid incremental formulation, with similar terms for patterns

estimating the stress components directly. Five formats were selected, labelled NET1 to NET5. These are listed in Tables 5.2 to 5.6.

Table 5.2: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing incremental formulation with history and partial state parameters. (NET1)

Input quantities							Output quantities		
e_v	σ_1	σ_2	σ_3	Δe_1	Δe_2	Δe_3	$\Delta\sigma_1$	$\Delta\sigma_2$	$\Delta\sigma_3$

Table 5.3: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing incremental formulation with no history parameters and complete state parameters. (NET2)

Input Quantities									Output quantities		
σ_1	σ_2	σ_3	e_1	e_2	e_3	Δe_1	Δe_2	Δe_3	$\Delta\sigma_1$	$\Delta\sigma_2$	$\Delta\sigma_3$

Table 5.4: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET3)

Input Quantities									Output quantities		
σ_1	σ_2	σ_3	e_1	e_2	e_3	Δe_1	Δe_2	Δe_3	$\sigma_1^{(t+1)}$	$\sigma_2^{(t+1)}$	$\sigma_3^{(t+1)}$

Table 5.5: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET4)

Input Quantities				Output quantities		
σ_0	e_1	e_2	e_3	σ_1	σ_2	σ_3

Table 5.6: Input and output quantities for the training patterns for the Crosslands Hill sandstone data set employing non-incremental formulation (NET5)

Input Quantities					Output quantities		
e_v	σ_0	Δe_1	Δe_2	Δe_3	$\Delta\sigma_1$	$\Delta\sigma_2$	$\Delta\sigma_3$

Within Tables 5.2 to 5.6, e_v is the volumetric strain and σ_0 is the hydrostatic (isotropic) stress. NET1, NET2 and NET5 are fully incremental formulations. NET4 is a fully direct formulation and NET3 is a hybrid direct formulation.

Within *FLAC*, incremental values are in fact rates with time step $\Delta t = 1$. Thus all of the incremental values listed in Tables 5.2 to 5.6 were determined using the time stamp of each data record of the log file of the triaxial test:

$$\Delta x = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \tag{5.6}$$

where Δx represents any of the incremental quantities and x_i is the numerical value of the quantity value at timestamp t_i.

There is some redundancy in the training data allied to the fact that the data used arise from triaxial compression testing. Where three stress components are used in describing the current state of material in the zone for which the ANN UDM is called, two of these will be identical. Where three strain increment components are used as state update parameters within the pattern format, two of these will be identical. Where, stress increment components are estimated, two of these will be zero. In direct formulations, two of the target stress components in the training pattern will be the same.

As none of the triaxial compression tests included an unload loop prior to the peak strength being attained, unloading training patterns were artificially added using:

$$\Delta x = \frac{x_{i-1} - x_i}{t_i - t_{i-1}} \tag{5.7}$$

for the incremental quantities, for records over the portion of the triaxial curves where linear behaviour was evident. The addition of artificial unload patterns was not required for the NET4 format. In the case of the NET3 format, the output stress components targets in the training pattern were taken from the previous record in the triaxial test data rather than the subsequent record.

Further patterns were added to the ANN training data to reflect the 'no-change' condition, when all strain increments have the value zero. For incremental formulations, those predicting $\Delta\sigma_{ii}$, the values for these output quantities were also set at zero. For direct formulations the output stress tensor was the same as the input stress tensor.

As the current state of strain is not normally available for use within a *FLAC* UDM, special programming arrangements were implemented so that the strain state could be used within the training data. The strain components were defined as zone property variables using the **f_prop** statement. Strain component values were found by accumulating the average zone strain increments through each *FLAC* calculation cycle. The strain component values were used in NET2, NET3 and NET4. The zone volumetric strain was derived from the sum of the normal strains values; principal strain components were found through strain transformation within the UDM.

The normalisation parameters were chosen such that the corresponding allowable ranges matched the expected ranges during use of the ANN UDM within *FLAC* and also the ranges of the data gathered during the triaxial testing. The allowable intervals used are indicated in Table 5.7. All training patterns, whatever their format, were normalised prior to the ANN training process. The output ranges of the normalised parameters are also shown in Table 5.7.

Table 5.7: Allowable ranges for quantities used in ANN training patterns and their remapped ranges

Quantity	Allowable Interval	Remapped Interval
e_v	[-0.05,0.05]	[-0.45,0.45]
$\sigma_1, \sigma_2, \sigma_3, \sigma_0$	[-500,500] MPa	[-0.45,0.45]
e_1, e_2, e_3	[-0.05,0.05]	[-0.45,0.45]
$\Delta e_1, \Delta e_2, \Delta e_3$	[-1 x 10^{-4}, 1 x 10^{-4}]	[-0.45,0.45]
$\Delta\sigma_1, \Delta\sigma_2, \Delta\sigma_3$	[-1,1] MPa	[-0.45,0.45]

Six triaxial compression tests were conducted in the laboratory. In the ANN training procedures, half of the data were used to train the ANN and the remainder were used in validation. The training data set comprised records produced from the tests conducted at confining pressures 4.5, 18 and 35 MPa. Records from the tests conducted at confining pressures 9, 23 and 65 MPa were reserved for validation of the trained ANN.

The experience gained from training ANNs to encapsulate isotropic linear elastic behaviour from simulated triaxial test data indicated that a viable tactic to ensure a broader range of stress states is embodied in the ANN than is present within the available training data is to cyclically permute the input and output quantities (see Section 5.5.2 and end of Section 5.10). Consequently, in forming training and validation data files, directionally dependent quantities were subjected to the cyclic permutation process. Isotropic quantities, for example volumetric strain and hydrostatic stress, within training pattern formats were not subjected to the cyclic permutation process.

In preparing the data for fully incremental formulations and conducting preliminary training runs, training of the ANNs, irrespective of format proved difficult. The reason for these problems was identified as being the precision at which the analogue signals from the displacement transducers were converted into digital values. Figure 5.30 illustrates the point for the axial displacement observations; Figure 5.31 shows a similar curve for the radial displacement. In Figure 5.30, the quantisation level is 0.001 mm exactly, yielding a maximum of 18 different states for the change in axial displacement; in Figure 5.31, it is 0.0047 mm exactly with 17 different states.

While this level of precision is suited to the production of the standard suite of curves (Figures 5.26 to 5.28), a higher level of precision is necessary if the data from the test are to be used to train an ANN with incremental formulation. The reason for this is that the effect of the quantisation precision carries through to the calculation of strain in both the axial and radial direction, and ultimately to the calculation of strain increments. 18 distinct states of change of displacement, ultimately leads to 18 distinct

states of strain increment. Figure 5.32 shows that the quantisation is not as serious an issue in the measurement of axial load, which is used in the calculation of stress and stress increments.

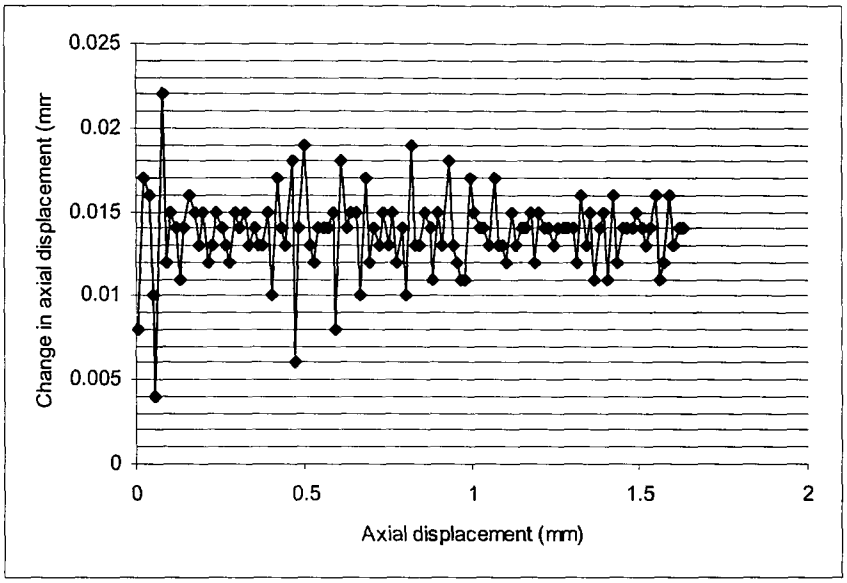


Figure 5.30: Plot of change in axial displacement versus axial displacement for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. The plot shows that quantisation of experimentally observed values is an important factor for ANNs with incremental formulations.

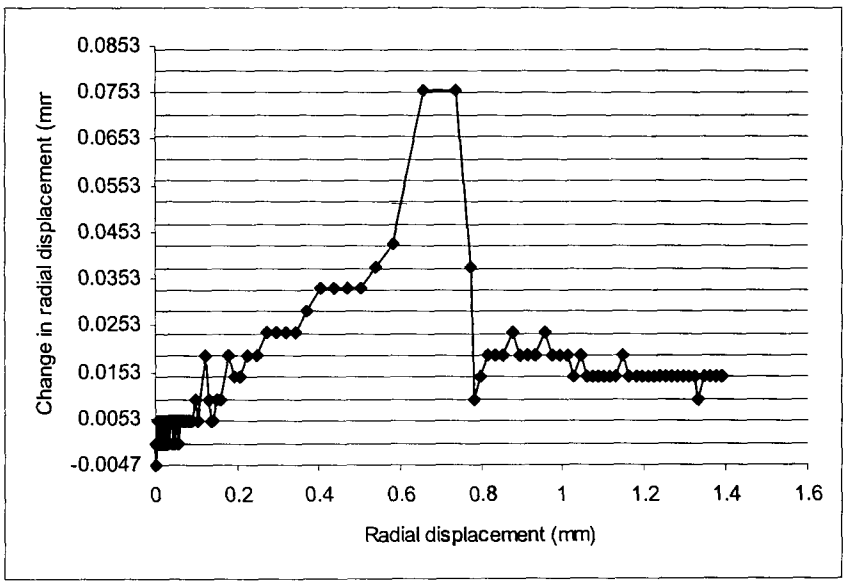


Figure 5.31: Plot of change in radial displacement versus radial displacement for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. The plot shows that quantisation of experimentally observed values is an important factor for ANNs with incremental formulations.

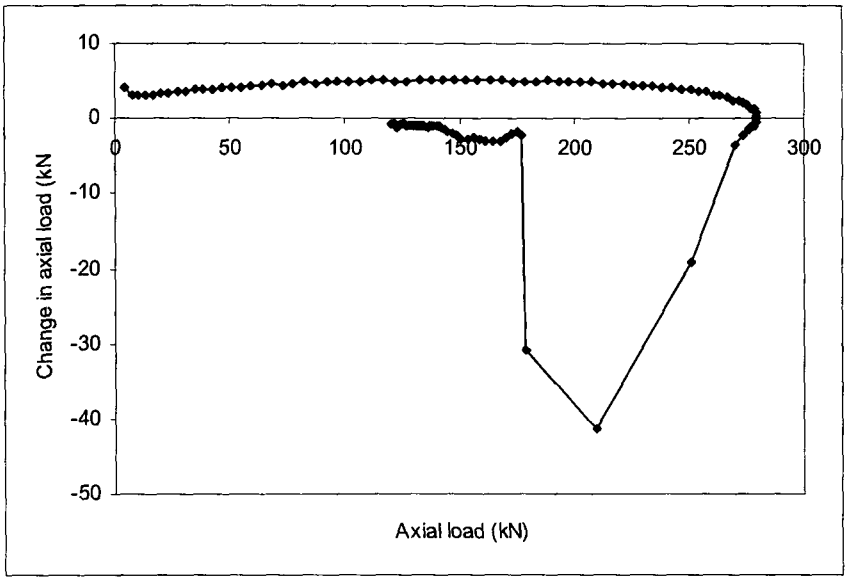


Figure 5.32: Plot of change in axial load versus axial load for the triaxial test on Crosslands Hill sandstone conducted at confining pressure of 4.5 MPa. This plot shows that quantisation of axial load values is not as serious a problem for training ANNs with incremental formulations as displacement observations.

In incremental formats of the ANN training patterns, strain increments form (part of) the input and stress increments form the output. The net result of low precision displacement analogue to digital (A to D) conversion and high precision axial load A to D conversion is that the same set of strain increments can be associated with different sets of stress increments. Consequently, in the back-propagation training process, two different sets of errors at output are associated with the same input. As ANN weight adjustments are made using errors across all training patterns, for these conflicting cases, the errors back propagated are more representative of the measurement (discretisation) process rather than the rock's constitutive behaviour.

In order to overcome this problem, a series of 15 to 20 samples were taken of the time stamped triaxial data from each triaxial test. These samples were used as the co-ordinates of a piece-wise linear representation of the constitutive behaviour of the specimen tested. Figure 5.33 shows the scheme for the triaxial test conducted at a confining pressure of 4.5 MPa. The curves are the actual data gathered, the crosses are the samples taken. Table 5.8 shows the numerical values of samples represented in the curves.

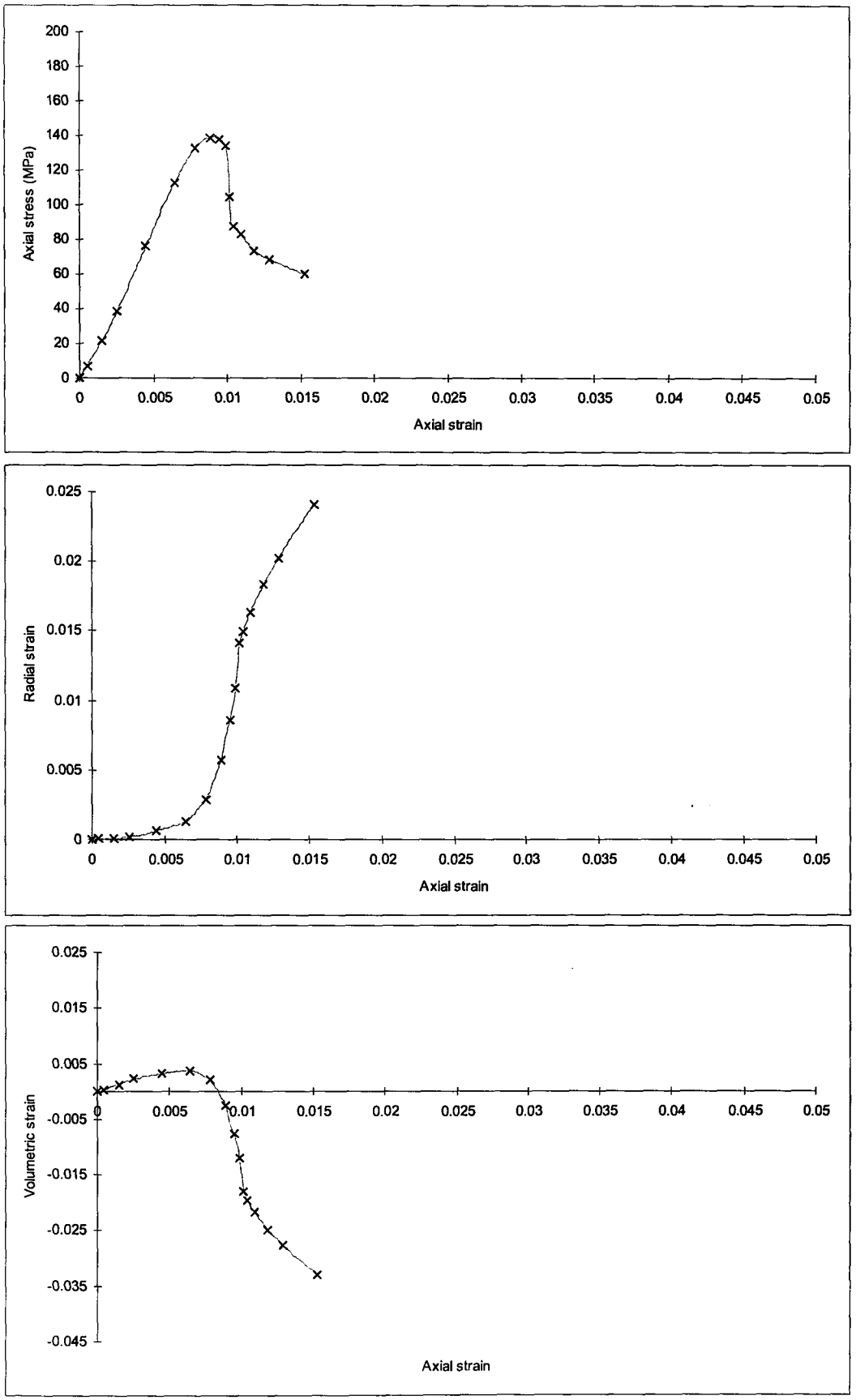


Figure 5.33: Piece-wise linear representation of constitutive behaviour for training.

The piece-wise linear representation was resampled at 10 second intervals and these interpolated curves were used to produce a second set of training and validation data files in an identical manner to that described earlier in this Section. While the discretisation precision issue was solved with this approach, a 'side effect' is that the linearity of the segments representing the constitutive behaviour means that the cardinality of the set of unique patterns contained within each training data set is reduced to the number of samples used to represent the curves.

Table 5.8: Samples used for piece-wise linear representation of the triaxial test results on Crosslands Hill sandstone rock for the test conducted at confining pressure 4.5MPa.

Sample	Time (s)	Axial Displ. (mm)	Radial Displ. 1 (mm)	Radial Disp. 2 (mm)	Axial Load (kN)	Axial Strain	Radial Strain	Axial Stress (MPa)
1	0	0.000	0.0000	0.0000	0	0.000000	0.000000	0.00
2	60	0.078	0.0047	0.0047	42.1	0.000771	0.000094	21.33
3	130	0.169	0.0095	0.0047	90.1	0.001670	0.000142	45.65
4	200	0.259	0.0190	0.0094	139.1	0.002560	0.000283	70.48
5	300	0.392	0.0237	0.0188	209.7	0.003875	0.000424	106.25
6	400	0.521	0.0332	0.0235	280	0.005150	0.000566	141.86
7	550	0.718	0.0521	0.0376	382.9	0.007097	0.000895	194.00
8	650	0.851	0.0616	0.0518	449.3	0.008412	0.001131	227.64
9	820	1.073	0.0948	0.0800	556.4	0.010606	0.001743	281.90
10	1000	1.310	0.1422	0.1224	660.2	0.012949	0.002639	334.50
11	1150	1.504	0.1991	0.1694	734.8	0.014866	0.003675	372.29
12	1250	1.634	0.2559	0.2212	774.9	0.016151	0.004759	392.61
13	1400	1.831	0.3839	0.3435	815.8	0.018098	0.007255	413.33
14	1450	1.896	0.4455	0.4000	822.7	0.018741	0.008433	416.83
15	1540	2.013	0.5972	0.5365	821.1	0.019897	0.011308	416.02

The overall result is that rather than having (at least two) patterns within the training data set with the same inputs and different outputs, the piece-wise linear representation and resampling processes produce the same number of patterns, but with many patterns being repeated (at least once). A second issue of whether or not the number of unique patterns within the training data sets is sufficient to represent the complete behaviour then arises.

It is stressed that the 'solution' to the low precision quantisation of displacement discussed is not an approach that would be adopted in normal circumstances. The correct solution to this problem, which is specific to ANN applications of the data, is to increase the number of bits used to quantise the analogue signal of the displacement transducers.

The procedures followed in the preparation of data to train the ANN have followed those discussed earlier for the linear elastic ANN, with variation on the training pattern format adopted. The adopted mesostructure of the ANN was determined heuristically, subject to requirements of minimisation of the number of

hidden layers and minimisation of the number of artificial neurons in hidden layers in order to keep training and recall times of the ANN as low as possible. The precision of training data has been found to be an important factor in both training ANNs and also, subsequently, their use within *FLAC*. The importance of the latter factor will become even more apparent when the results of the ANN training process are presented and discussed in the next Section.

5.5.4 Results of Training and Validation

The results of training five ANNs with the raw, 'low precision' triaxial data, using the formats for NET 1 through NET 4, are presented in Table 5.9. The results for networks 1 to 4 are graphically presented in Figure 5.34. A corresponding set of training and validation results for the 'piece-wise' linear laboratory data are presented in Table 5.10.

Table 5.9: Summary of training and validation errors, at different stages in the learning process, for ANNs trained with the triaxial test data.

Network	ANN Training log for Crosslands Sandstone : raw data		Structure		Error	Training Cycles				
	Format				Type	10K	40K	60K	80K	120K
1	NET1	ev,s1,s2,s3,de1,de2,de3,ds1,ds2,ds3	7-7-3	Train	RMS	0.037		0.018		
					Max	0.179		0.174		
				Verify	RMS	0.043		0.021		
					Max	0.152		0.118		
2	NET2	s1,s2,s3,e1,e2,e3,de1,de2,de3,ds1,ds2,ds3	9-9-3	Train	RMS	0.037	0.019		0.018	0.017
					Max	0.289	0.141		0.151	0.155
				Verify	RMS	0.042	0.021		0.021	0.021
					Max	0.148	0.242		0.255	0.260
3	NET3	s1,s2,s3,e1,e2,e3,de1,de2,de3,s1,s2,s3	9-9-3	Train	RMS	0.003	0.002			
					Max	0.075	0.076			
				Verify	RMS	0.007	0.005			
					Max	0.045	0.039			
4	NET4	s0,e1,e2,e3,s1,s2,s3	4-6-3	Train	RMS	0.036	0.007		0.006	
					Max	0.100	0.037		0.033	
				Verify	RMS	0.041	0.008		0.008	
					Max	0.120	0.036		0.037	
5	NET4	s0,e1,e2,e3,s1,s2,s3	4-12-3	Train	RMS		0.007		0.006	
					Max		0.036		0.031	
				Verify	RMS		0.009		0.009	
					Max		0.042		0.042	

Observations from the results presented in Table 5.9

1) Overall, the best network produced was Network 3, (format NET3) with a RMS training error of 0.002 and maximum training error of 0.076. Under verification tests with triaxial test data absent from the training data set, the RMS verification error was 0.005 and the maximum error at output was 0.039. Although NET4 (either version) has lower maximum output error, it also has higher RMS error.
2) Network 4 and 5 (format NET4) (trained with either 6 or 12 hidden artificial neurons), are reasonable seconds to Network 3, although it takes twice the number of training iterations to achieve comparable accuracy.

3) ANNs predicting stress increments were harder to train than those predicting the stress components directly. This can be seen by comparing RMS and maximum output error values for Network 1 and Network 2 with the remaining ANNs.
4) All networks are able to generalise (interpolate and extrapolate) with an accuracy comparable with the training error. This is seen by comparing RMS and maximum output errors for the same network between training and verification phases.
5) In Network 2, the RMS error reduces with increasing training cycles while the maximum output error increases. This is a sign that this network had not completed a full training phase. However, the training time required was lengthy so after 120,000 iterations, training was terminated in any event.
6) None of the trained ANNs demonstrate accuracy (either in RMS or maximum output error) comparable with that of the linear elastic ANN used in Section 4.4. Considering Network 3 after 40,000 training cycles, a RMS training error of 0.002 means that, on average, predictions of stress components will be 0.2% in error; the maximum output errors will be between 3.9% and 7.6%. Comparable figures for the elastic net were 0.07% RMS, and 0.6% maximum output error. While Networks 3 to 5 can be said to have learned the triaxial compression test data, these error magnitudes become significant when the ANN is used within *FLAC*.
7) Two training runs with a NET4 format are presented to demonstrate that increasing the number of artificial neurons in the hidden layer of the perceptron had negligible effect on the training or validation performance.

The training results for Networks 1 to 4 are presented graphically in Figure 5.34. Actual network output has been plotted against Target network output for the 3 outputs of each of the ANNs. The line on which the results should lie is also plotted. From the graphs, it can be seen that all of the networks have isolated the trends in the data; however, those using formats with incremental stress components at output do not perform well. For Network 1, vertical 'columns' formed by the around the origin indicate that there are multiple outputs for similar inputs. As mentioned earlier, the presence of these 'columns' for ANNs with incremental formulations is ultimately due to the precision of data acquisition used in the test.

Table 5.10: Summary of training and validation errors, at different stages in the learning process, for ANNs trained with piece-wise linear re-sampled triaxial test data.

Network	ANN Training Log: Crosslands Sandstone - re-sampled		Structure		Error	Training Cycles				
	Format				Type	5K	10K	40K	80K	120K
6	NET1	ev,s1,s2,s3,de1,de2,de3,ds1,ds2,ds3	7-7-3	Train	RMS		0.039		0.016	
					Max		0.388		0.311	
				Verify	RMS		0.043		0.016	
					Max		0.147		0.071	
7	NET2	s1,s2,s3,e1,e2,e3,de1,de2,de3,ds1,ds2,ds3	9-9-3	Train	RMS		0.037	0.017		0.015
					Max		0.388	0.303		0.306
				Verify	RMS		0.041	0.016		0.015
					Max		0.142	0.256		0.257
8	NET3	s1,s2,s3,e1,e2,e3,de1,de2,de3,s1,s2,s3	9-9-3	Train	RMS		0.028	0.002		
					Max		0.076	0.078		
				Verify	RMS		0.007	0.005		
					Max		0.045	0.039		
9	NET4	s0,e1,e2,e3,s1,s2,s3	4-6-3	Train	RMS		0.038	0.007	0.006	0.006
					Max		0.098	0.042	0.036	0.034
				Verify	RMS		0.046	0.010	0.093	0.009
					Max		0.140	0.046	0.044	0.042
10	NET5	e0,s0,de1,de2,de3,ds1,ds2,ds3	5-6-3	Train	RMS	0.038		0.021		0.017
					Max	0.386		0.300		0.298
				Verify	RMS	0.042		0.022		0.016
					Max	0.149		0.092		0.072

ANN training results with resampled experimental curves are presented in Table 5.10. Comparison of results in Table 5.9 with corresponding results in Table 5.10 demonstrate no improvement in training accuracy (in fact deterioration in most cases). It is reiterated that the resampling procedure was attempted to try to overcome the data precision issue and would not be part of routine application of the technique.

Network 10 shows verification errors that are much lower than training errors. Analysis of errors identified large stress component changes in the post peak regions of the stress strain curves (only in training data set) as the reason for this disparity.

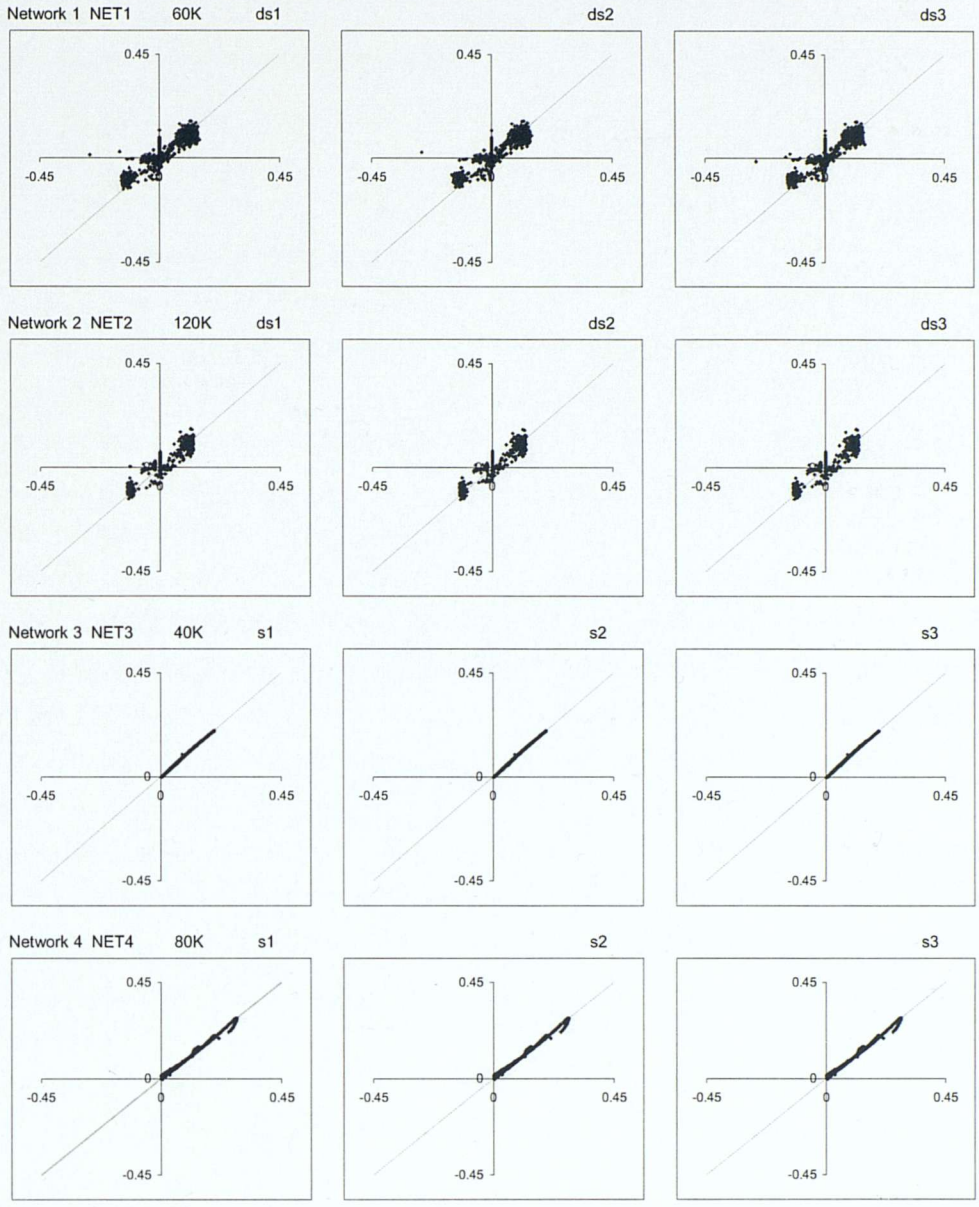


Figure 5.34: Graphical presentation of training error (normalised output) for Networks 1 to 4. For each plot, the x-axis represents target output and the y-axis represents actual output. These results indicate good training for NET3 and NET4 and poor training of NET1 and NET2.

5.6 Testing the ANNs trained on Crosslands Hill sandstone data

Two trained ANNs from those reported in the last Section carried forward to verification testing within *FLAC*. The first ANN tested was Network 3, the ANN with the lowest training error, which uses data format NET3. The NET3 formulation predicts values of the new stress components directly. The weights, thresholds and normalising and denormalising parameters for Network 3 were exported as a *FISH* data file, the NET3 format ANN UDM was invoked for the *FLAC* model for Verification Test 1, Cases 1, 2 and 3. The results of the performance of this network used within *FLAC* were so poor that they are not worth presentation here. The reason for its poor behaviour was that predictions of the stress components were insufficiently accurate to provide a stable calculation cycle. Initialising the model with zero velocities and then instructing *FLAC* iterations led to dynamic behaviour of the model, whereas it should not have responded at all with a null perturbation.

The second ANN tested within *FLAC* was Network 1. This Network employs the NET1 data format; it predicts stress component increments which are added to stress components from the previous *FLAC* cycle, rather than predicting the new stress components directly. Under training, this network had a RMS error of 1.8%, and maximum output error of 17.4%. Under validation with unseen data, the RMS error was 2.1% and the maximum output error of 11.8%. Errors under training and validation are thus rather poor. Nevertheless, the weights, thresholds, normalising and denormalising parameters were exported as a *FISH* data file, the NET1 format ANN UDM was invoked for the *FLAC* model and initialised with the Network 1 data. The results of testing this ANN within *FLAC* are presented in the following sub-sections.

Without prejudice to the remaining content of this Chapter, it is appropriate to highlight—at this stage—that the reason why the ANN with the lowest training and validation error behaved so poorly under test within the *FLAC* calculation cycle was that the formulation of the data format NET3 gave errors in estimation that were still sufficiently large to destabilise the *FLAC* calculation cycle. The poor performance of this network (Network 3) is not attributable to ANN training or verification issues.

5.6.1 Network 1 ANN UDM performance under Verification Test Type 1, Case 1

This tests the model by applying a constant velocity in the negative y-direction (toward the bottom) to the top boundary of a one zone model of a right cylinder. The details of the numerical procedure are described in Section 5.3.4 and Appendix B5. Appendix B7 provides *FLAC* data files that document the test. Figure 5.35 shows the variation of the normal stress components with increasing strain in the direction in which velocities are applied (the y-direction).

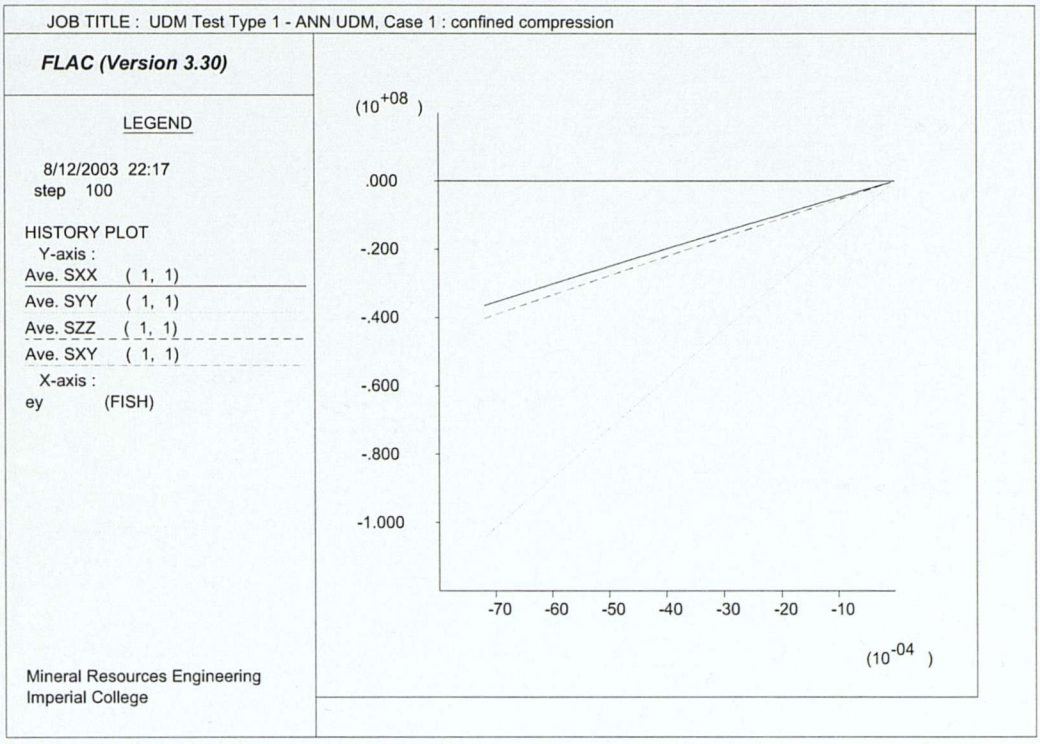


Figure 5.35: Crosslands Hill sandstone ANN UDM results for Case 1 loading: Stress versus strain in y direction. Gradients of lines are $\lambda_1 = 14.6$ GPa ; $\lambda_2 = 5.7$ GPa, consistent with elastic properties determined in Section 5.5.1.

The curves presented in Figure 5.35 are remarkably linear. Comparison with Figure 5.9 for linear elastic rock shows that generally, the behaviour of the specimen under this test has been recovered correctly. Analysis of the gradients of the curves, assuming elastic behaviour, yields values:

$\lambda_1 = 14.6$ GPa ; $\lambda_2 = 5.7$ GPa

$G = 10.2$ GPa ; $K = 12.4$ GPa

$E = 23.9$ GPa ; $\nu = 0.18$

which, after examination of Figure 5.27, can be seen to be broadly in agreement with the data gathered in the laboratory tests analysed by conventional means, i.e. for E and ν.

5.6.2 Network 1 ANN UDM performance under Verification Test Type 1, Case 2

This tests the model by applying a constant velocity in the positive x-direction (toward the right) to the top boundary of a one zone model of a right cylinder. The details of the numerical procedure are described in Section 5.3.4 and Appendix B5. Appendix B7 provides a *FLAC* data files that document the test. Figure 5.36 shows the variation of the shear stress with increasing shear strain.

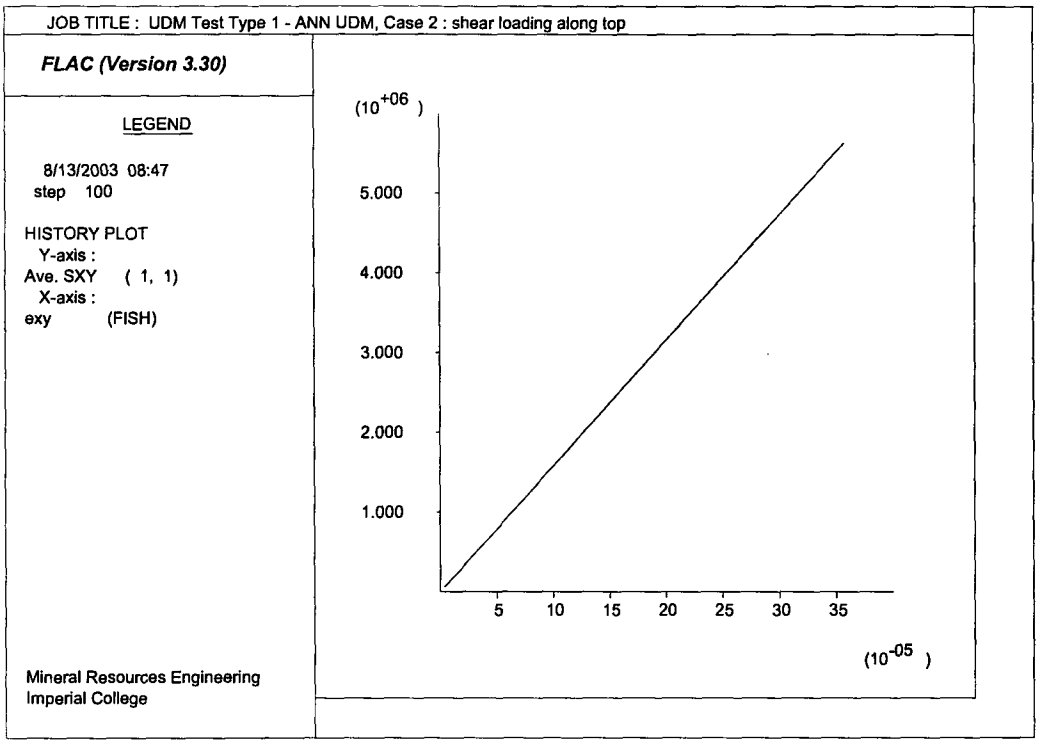


Figure 5.36: Crosslands Hill sandstone ANN UDM results for Case 2 loading: Shear stress versus tensorial shear strain. A surprising degree of linearity is obtained. Gradient of curve is 16 GPa implying a shear modulus of 8 GPa, around 2 GPa lower than obtained in Section 5.6.1.

Again, surprisingly linear behaviour has been obtained with the model using the Crosslands Hill sandstone ANN UDM. As the simulated specimen is sheared, the gradient of the curve presented in Figure 5.36 is 16 GPa. If it is assumed that the behaviour is elastic, the effective value of the shear modulus applying under these loading conditions is 8 GPa. This value of the shear modulus should be compared with the value obtained by analysing the curves of the chart in the previous Section. The shear modulus applying here is around 2 GPa lower than that obtained earlier. It should be noted that Figure 5.27 clearly showed stress dependence of the elastic parameters for Crosslands Hill sandstone rock, so it is not too surprising to find that different values of the parameters can be obtained when the simulated specimen is loaded in different ways.

5.6.3 Network 1 ANN UDM performance under Verification Test Type 1, Case 3

This tests the model by applying both the constant velocity conditions of Verification Test Types 1 and 2. The details of the numerical procedure are described in Section 5.3.4 and Appendix B5. Appendix B7 provides *FLAC* data files that document the test. Figure 5.37 shows the variation of normal and shear stress components with increasing compressive strain in the y-direction.

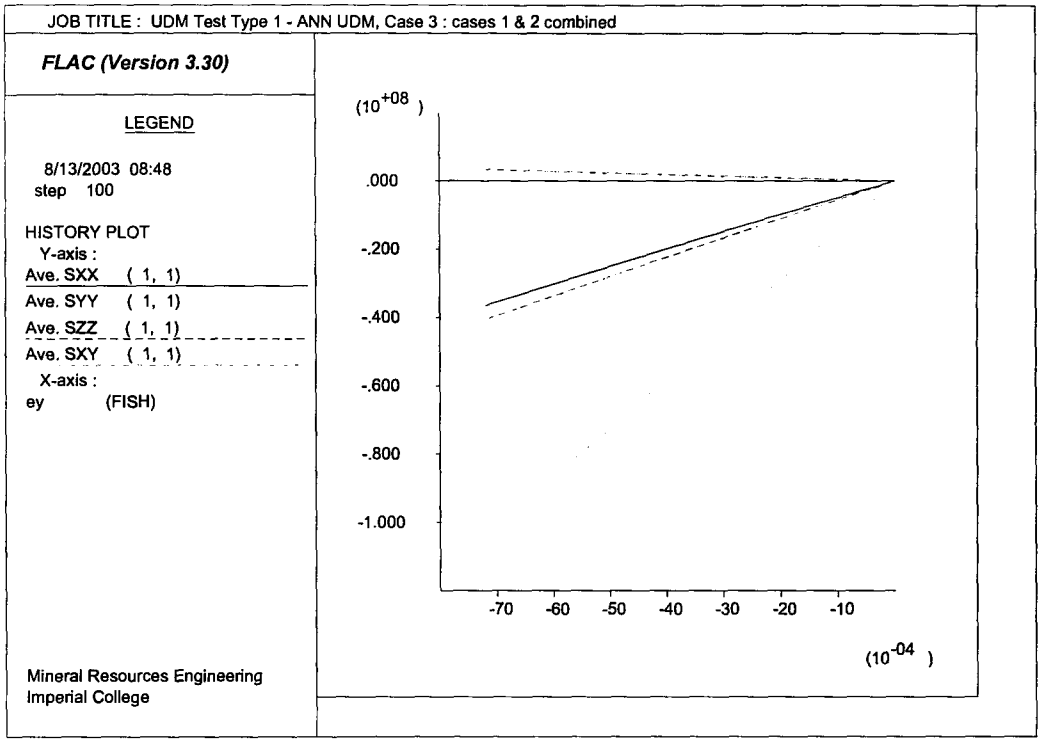


Figure 5.37: Crosslands Hill sandstone ANN UDM results for Case 3 loading (combination of Test Type 1 and 2 loading) : Stress versus normal strain in the *y*-direction. These results are as expected, that is, a combination of previous 2 cases.

Again, in comparison with the verification tests for linear elasticity or the linear elastic ANN UDM presented earlier, the performance of the Crosslands Hill sandstone rock is broadly comparable. Under applied normal and shear deformations, the model here performs in a manner consistent with the behaviour of the same deformations applied independently — with one exception. The shear modulus determined from Figure 5.38 below is approximately 5 GPa, compared with 8 GPa for the earlier test. The reasons for this are subject to further work.

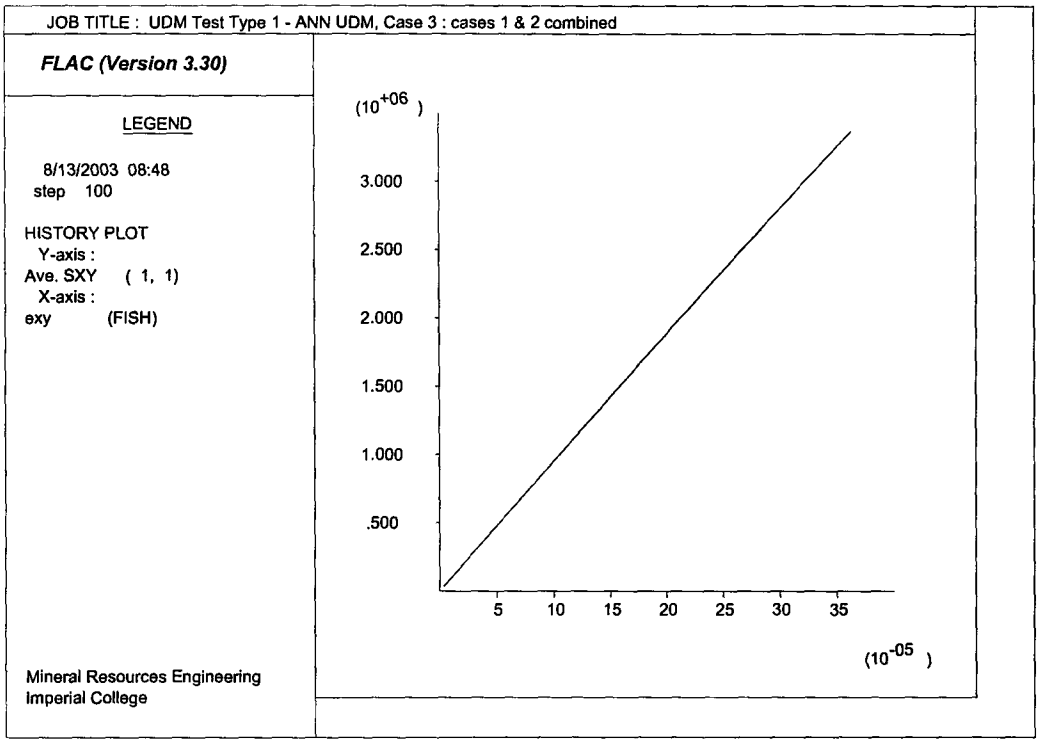


Figure 5.38: Crosslands Hill sandstone ANN UDM results for Case 3 loading. (Shear stress versus tensorial shear strain). Gradient is lower than comparable case, Case 2 (Figure 5.36).

5.6.4 Network 1 ANN UDM performance under a variant of Verification Test Type 2

Verification Test Type 2 cannot be applied to the *FLAC* model using the Crosslands Hill sandstone ANN UDM; this is because, verification Test Type 2 measures the performance of an elastic ANN UDM against the elastic constitutive model available within *FLAC* (see Section 5.3.5). The variant used here involves assigning the Crosslands Hill sandstone ANN UDM to all zones in a 10 x 10 zone model of a right cylinder. Boundary conditions simulating the triaxial compression test that produced data to train the ANN are applied to the model (the test conducted at 35 MPa confining pressure). The test is successful if the results of the *FLAC* model are consistent with the results recorded from the laboratory compression test.

Appendix B7 provides a *FLAC* data file that documents the test. Figure 5.39 shows the variation of normal and shear stress components at sample points within the simulated specimen with increasing compressive strain in the y-direction. Figure 5.40 shows a plot of the model after 8000 *FLAC* cycles with grid node displacement vectors superimposed.

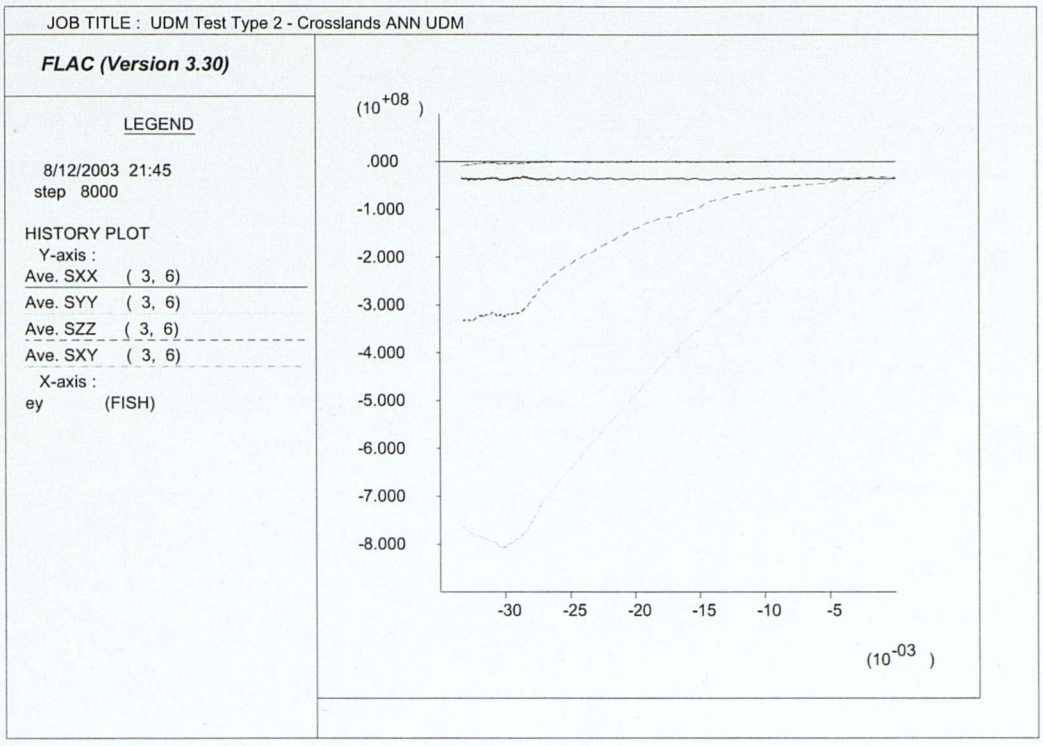


Figure 5.39: *FLAC* time step history plot of normal stress components versus average normal strain in the *y*-direction for the variant of Test Type 2. Confining pressure 35 MPa. *FLAC* sign convention produces the curve 'up-side-down' and 'back-to-front' in comparison to conventional means of presentation (Figure 5.24). σ_{yy} (Ave SYY) curve shows a clear peak at 30 $m\varepsilon$ axial compression. Note: simulation does not present a 'real' triaxial compression test due to plane strain associated with 2D *FLAC* modelling.

The first point to note from Figure 5.39 is that the 'axial stress' versus 'axial strain' curve shows that the simulated specimen has been taken through a peak stress point into the post-peak region, where strain softening behaviour is apparent. It is stressed that no conventional information on yield or peak strength has been encoded into the *FLAC* model, for example, a Mohr-Coulomb or Hoek-Brown failure criterion. The only information on peak strength available within the model is that contained within the weights of the ANN encoded in the ANN UDM.

The second point to note is that the simulated specimen has fully obeyed the boundary conditions applied within the model throughout the simulation. A constant confining pressure of 35 MPa was assigned to left and right boundaries of the model. The simulated test was initialised with hydrostatic stress of 35 MPa throughout all zones. It can be seen from the plot that σ_{xx} has remained constant (at 35 MPa) throughout the simulated test schedule. Also, the shear stress component σ_{xy} at the sampling point has also remained constant up to the point at which the peak strength is attained. Thereafter, it becomes non-zero, as would be expected in a real compression test.

σ_z is the so-called 'out-of-plane' stress component and is dependent not only on the loading of the specimen in the *y*-direction, but also on the requirement of zero strain in the out-of-plane direction. Thus σ_z rises as the compression of the specimen progresses. In a 'real' laboratory triaxial compression test this would not occur. The fact that it occurs in the simulation is a result of the plane strain formulation of the 2D *FLAC* code. Thus the simulated 'triaxial' specimen is subject to a greater degree of confinement than the equivalent physical specimen in a triaxial rig. This is one of the main reasons why, clearly, the peak strength apparent in the plot of Figure 5.39 is greater than the expected compressive strength of 313 MPa with a confining pressure of 35 MPa. If the average of σ_x and σ_z is used to represent the confining pressure applying at the point of failure, the result is (330 + 35) /2 = 183 MPa. Substitution of a confining pressure of 183 MPa into the Hoek-Brown criterion fitted to the Crosslands Hill sandstone triaxial test results (Equation 5.1) indicates a peak strength of 760 MPa. Thus the 805 MPa peak strength apparent in Figure 5.39 is not unreasonable.

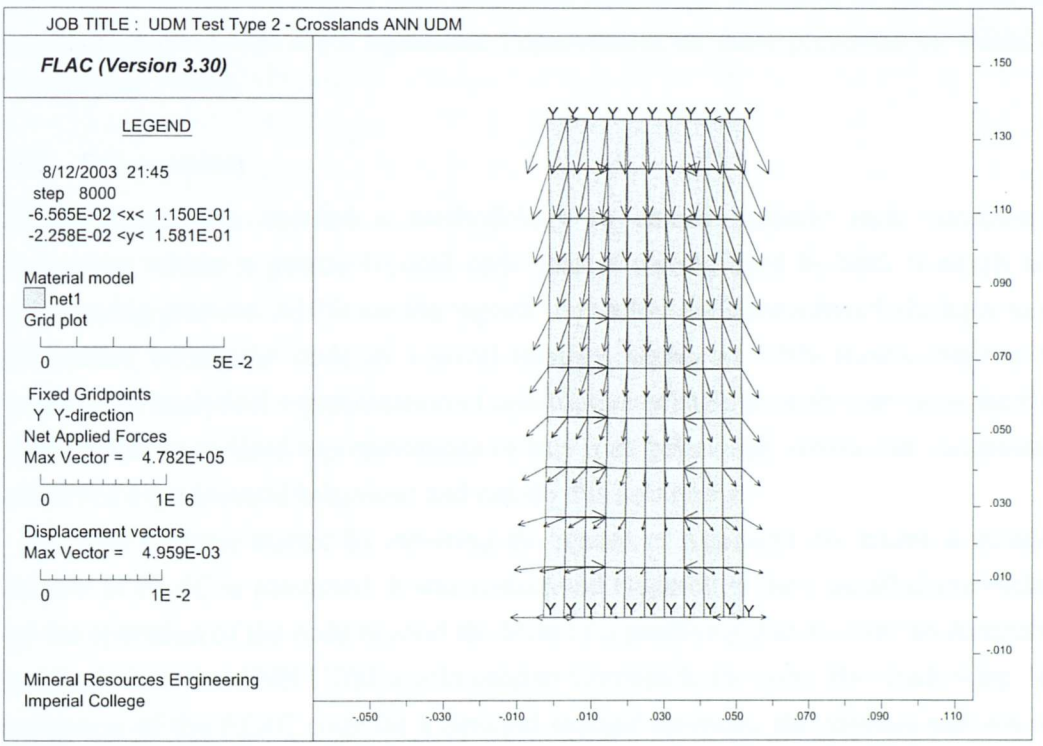


Figure 5.40: *FLAC* grid plot showing grid node displacements after 8000 cycles (in the 'post-peak' region, fixity conditions and applied forces for the variant of verification Test Type 2. The diagram shows that the behaviour of the ensemble is consistent with that expected.

Figure 5.40 shows the grid node displacements of the specimen after 8000 cycles, when the specimen is behaving in its strain softening mode, past the point of peak strength.

5.6.5 General comments on the Crosslands Hill sandstone ANN UDM

Given the poor level of training of Network 1 reported earlier, the performance of its ANN UDM under the loading cases of verification Test Type 1 and under modified verification Test Type 2 is surprising. RMS errors in training and verification of 2% are unsatisfactory and, from the experience of the author with ANNs, will be improved upon. The verification tests indicate there are still issues that need attention. For example, the issue of whether the change of value of the shear modulus between Case 1, 2 and 3 of verification Test Type 1, is correct behaviour still needs to be determined. Another important issue that is the subject of further work is how to deal with Mode 3 invocations of the ANN UDM, so that grid point masses are suitably tuned to the model time step increment. This latter point is likely to require the full co-operation of the code developers, Itasca, in order to be resolved.

Despite these and similar issues, the good performance of what would normally be regarded as a poorly trained ANN, is perhaps an indication of the robustness of the concept of integrating ANNs with numerical modelling. It should be noted that the results presented here are a significant improvement on those presented by Millar & Calderbank (1995)

5.7 Discussion

This Chapter has detailed a methodology to embed realistic rock constitutive behaviour within a geomechanical code that is widely used in both research and engineering practice. ANNs are the 'agents' that allow the constitutive behaviour to be embedded within the code in a novel manner. Using an ANN means there is no recourse to analytical representation of constitutive behaviour, with ever more detailed refinements to stylised representations of true rock behaviour. ANNs can encapsulate observed experimental behaviour and can do this accurately.

The Chapter started by referring the reader to Appendix B1 where a detailed review of *FLAC* is presented. It was considered essential to have detailed knowledge of the operation of the code beyond the level of a practising user in order to determine precisely how the ANN UDM would need to interface to the code. By 'shadowing' the operation of the *FLAC* code for a detailed worked example, the obvious and not-so-obvious interfaces were identified. It is true to say that in defining a user-defined constitutive model (UDM) focus can be retained on a single modular portion of the *FLAC* calculation cycle. Mode 2 invocation of a UDM must return new stresses from old stresses and strain increments. The user is 'free' to decide how this specific task is achieved, and an ANN is a suitable agent to achieve it.

However, the worked example demonstrated that it is crucial to recognise the role of mode 3 invocation of a UDM. The mode 3 section of the UDM needs to return

the maximum modulus of a zone in order to compute the grid point mass of triangles within the zone. In turn, the triangle grid point masses are used to determine the nodal grid point masses that are the 'tuning' parameters in *FLAC*'s numerical damping scheme. Thus there is an issue of ensuring compatibility between the mode 2 section of an ANN UDM and the mode 3 section. This issue is subject to further work beyond the scope of this thesis.

The isotropic linear elastic ANN trained as described in the previous Chapter was encoded into a *FLAC* user defined constitutive model (UDM) using *FLAC*'s *FISH* macro language. Despite its propriety nature, *FISH* is well understood by a large number of practising engineers and scientists. Thus it is hoped that the presentation of the specific implementation details in this Chapter, as well as listing of the ANN-UDM code, will prove useful.

Much of the material in this Chapter and the supporting appendices has concerned the presentation of evidence that the techniques work as stated. Demanding UDM verification tests were explained. These were developed not only to allow the absolute assessment of ANN UDM (i.e. did it work?) but also to allow performance assessment of the ANN UDM relative to the available, 'in-built' models (i.e. did it work as well?). Under these tests, the linear elastic ANN-UDM worked well and was shown to be as capable as the isotropic linear elastic constitutive model, subject to training issues.

These training issues related to the accuracy with which that the trained ANN 'learnt' the data with which it was trained. Generalised perceptrons with sigmoidal activation functions are semi-linear in nature (as explained in Chapter 2) and as such can never have output that exhibits perfect linearity. Isotropic, linear elasticity demands perfect linearity. In comparison to 'perfect' behaviour, the ANN trained with simulated triaxial compression test data with elastic behaviour led to minor over prediction of stress increments that could be directly attributed to the degree of training. Thus, as well as providing proof of concept (that ANNs can be used as UDMs in constitutive models) the ANN-UDM performance under the verification tests within *FLAC* highlighted a need for accuracy.

Surprisingly, this conclusion is at odds with the results obtained when the ANN was trained with real rock triaxial data. The laboratory results, when analysed conventionally revealed that Crosslands Hill sandstone rock was comparatively 'well behaved' in the sense that its behaviour fitted well with the main characterising concepts. Nevertheless, it also exhibited many aspects of non-linear behaviour. ANNs using the triaxial data in several different formats were able to 'learn' the data, to varying extents. In the training phase, ANNs predicting the new stress components directly had lower RMS and maximum output errors than ANNs predicting stress

component increments. However, the best ANN, that with the lowest RMS and Maximum Output errors, could not be trained with errors lower than for the isotropic linear elastic ANN of the previous Chapter.

Despite the difficulties in training the 'incremental' ANNs, they still 'learned' the required behaviour. It has been recognised that some of the difficulties arise as a result of the low precision of observed quantities, in particular, the strain rates. Displacement transducers, and their A to D conversion equipment, need to have sufficient sensitivity to accommodate incremental formulations of the ANN training data patterns. Sampling the triaxial compression test data to try to overcome these precision issues led to no improvement in ANN training. Overall, the training errors were comparable to the untreated data. In effect, one deficiency substituted for another.

Despite the rather poor training errors, the best 'direct formulation' ANN and the best 'incremental formulation' ANN were imported into the *FLAC* environment for testing as UDMs. Although the direct formulation ANN had lower training and validation errors, it proved hopeless in the *FLAC* environment. Its predictions of stress components were sufficiently inaccurate to lead to numerically unstable results, even when there was no perturbation to the model (no imposed loads or velocities).

In contrast, the incremental formulation ANN UDM worked well under the verification tests presented. At this stage, it is thought that the reason why this is so is because the errors that the ANN UDM inevitably introduces in its estimates of stress component increments are sufficiently small that they do not have an inherently destabilising influence. Moreover, through the *FLAC* cycling sequence, these errors could cancel themselves out (if they are small and random).

Here, it is important to recall one of the most important results of Chapter 4: The ANN was able to reproduce output consistent with 'plane strain' formulations, when no information reflecting this constraint was presented to the ANN in training. This is a non-trivial result and in the context of the current Chapter dealing with real rock constitutive behaviour and numerical modelling demonstrates a potential enhanced capability of ANNs as constitutive models, not present in conventional, analytical, approaches.

This Chapter has detailed a proof of concept study that has been successful in achieving its aim. Further work needs to be done to convert the work into a tool that can be adopted in a more widespread fashion for practical use. It is envisaged that in the future the tool may use the procedures presented here to demonstrate that numerical models themselves could be used to synthesise training and validation data sets for small but complicated engineering situations in up-scaling studies.

During the last 15 years, there have been many developments coupling ANN methodologies with those of numerical modelling in rock mechanics. Virtually all of these have comprised back-analysis formulations aiming to recover i) geomechanical parameter values, ii) structure of the rock constitutive model, iii) variability of geomechanical parameter values, iv) dynamic updates to structure and parameter values of constitutive models. A variety of parallel distributed processing methods have been adopted as optimising tools in these tasks including radial basis function ANNs, Genetic Algorithms and Particle Swarm codes. For the specific instance of finite difference codes such as *FLAC*, none report the use of the optimised agent directly replacing the mechanics of the constitutive model in the geomechanical code as has been demonstrated in the work reported in this Chapter. Rather, these alternative methodologies have loosely coupled the parallel distributed processing method to the geomechanical code, with similar or identical methodology to that proposed by Feng *et al.* (2000), reviewed in more detail in Chapter 3. There are examples where ANNs fully displace the constitutive model within geomechanical codes (Shin & Pande, 2000; Javadi, Tan & Zhang, 2003; Hashhash, Jung & Ghaboussi, 2004;) but these all concern finite element methods rather than finite difference methods.

6. Fracture Identification from Digital Imagery with Artificial Neural Networks

6.1 Introduction

The motivation behind this work is to speed up the acquisition of reliable data concerning fractures within rock masses. There are many examples of photographic, video or digital stills cameras being used to obtain imagery of rock mass exposures, for the purpose of mapping the fractures contained therein. The key step in these processes has been human interpretation of the imagery to produce discontinuity trace maps. In comparison to the data gathering phase (the taking of the photograph), this step of the process is rather slow and subject to human bias.

The purpose of the material presented in this Chapter is twofold.

1. To present further examples of the use of ANNs within rock engineering systems.
2. To develop two ANN classifier modules that operate upon digital imagery to semi-automatically produce trace maps.

Together, these two modules form an essential part of a rock engineering system that will ultimately be able to automatically extract geometrical information concerning rock fractures from digital imagery. Industry standard programs such as *FracMan* (http://fracman.golder.com/) are able to analyse fracture trace maps to compute useful measures of rock mass structure using robust algorithms such as those developed by Dershowitz and Einstein, (1988). A semi-automatic fracture trace map generator will complement programs such as *FracMan* to allow robust analysis of rock mass geometry at moderate cost. Currently, for serious work, there is no alternative to mapping the fractures using scan line methods as set out by the ISRM (1981). Workers such as Moore (1974), Chandler and Moore (1989) and Harrison (1995) have made significant progress in semi-automated approaches employing photogrammetric methods for discontinuity characterisation. Kamewada *et al.* (1990), Tanimoto (1991), Mimuro *et al.* (1991) and Reid (1998) have tackled the problem using digital image processing techniques. Cabello *et al.* (2002) has tacked the similar problem of identifying individual rock blocks, outlining them using image processing techniques coupled with ANNs in a real-time application for rock crusher protection.

The computational objectives of a fracture trace mapper are comparable with image processing modules for optical character recognition (OCR). However, in comparison with automatically identifying typeset or hand-written characters with an OCR module, trying to classify fractures apparent in digital imagery of rock masses with a fracture recognition module is a more difficult task. Firstly, an OCR module generally 'knows' where the characters it has to identify are, or at least where the next

character is, given the first one. A fracture recognition module has no such advantage and so must first identify locations likely to contain fracture image content. Secondly, OCR modules are generally optimised to recognise characters within digital images that have white, or at least uniformly coloured, backgrounds. A fracture recognition module may have to contend with significant variations in background colour and appreciable image 'noise'. This 'noise' is not only associated with the process used to obtain the digital images of rock masses. As can be seen in Figure 6.1 below, the 'noise' is also an inherent characteristic of their rock mass digital image feature content.

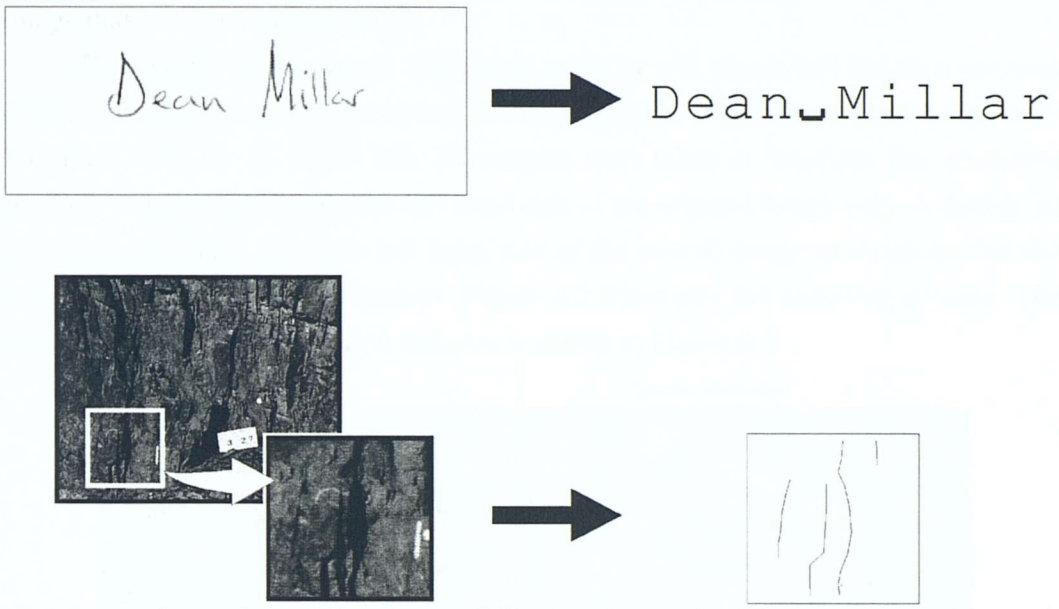


Figure 6.1: Illustration of the relative difficulty of an optical character recognition task (upper part) in comparison with a fracture recognition task (lower part).

The first of the two modules mentioned above is used to identify areas within a digital image that contain fracture, or more generally discontinuity, image content. It does this by partitioning the digital image into a set of image patches. Each patch is passed to the ANN classifier that determines whether or not it contains discontinuity image content. This first ANN module is termed the 'Fracture Discriminator' and is discussed in detail in the following section.

The second of the two modules operates upon samples passed to it to determine the orientation and termination condition of discontinuity segments contained within it. It does this by associating the image content of the sub-sample with one of an idealised set of representations of fracture content for which the orientation, persistence and termination conditions are known. This idealised set is termed a 'fracture alphabet' and this second ANN module is termed the 'Fracture Classifier'. It is introduced in Section 6.3 and developed in Sections 6.4 to 6.6.

The material in this Chapter requires some understanding of digital image processing fundamentals. As these can be found by reading any mainstream text on the subject, this material is not presented as part of the main body of the thesis but is deferred to Appendix C1 – for those readers who require this additional support.

6.2 The Fracture Discriminator

6.2.1 Data set for the Fracture Discriminator

The training data set used for the Fracture Discriminator is sampled from the digital image that is shown in Figure 6.2.

The overall image size is 256 pixels wide by 192 pixels tall and thus contains 49152 different pixels. Sub-samples, or image patches, from this original image were 16 pixels wide by 16 pixels tall. 30 samples were taken at locations that contained fracture image content from the left hand side of the original image only. A further 30 samples were taken from the left hand side of the overall image at locations that did not contain fracture image content. Figure 6.2 illustrates the sampling process. The complete training data set of 60 samples is shown in Figure 6.3.

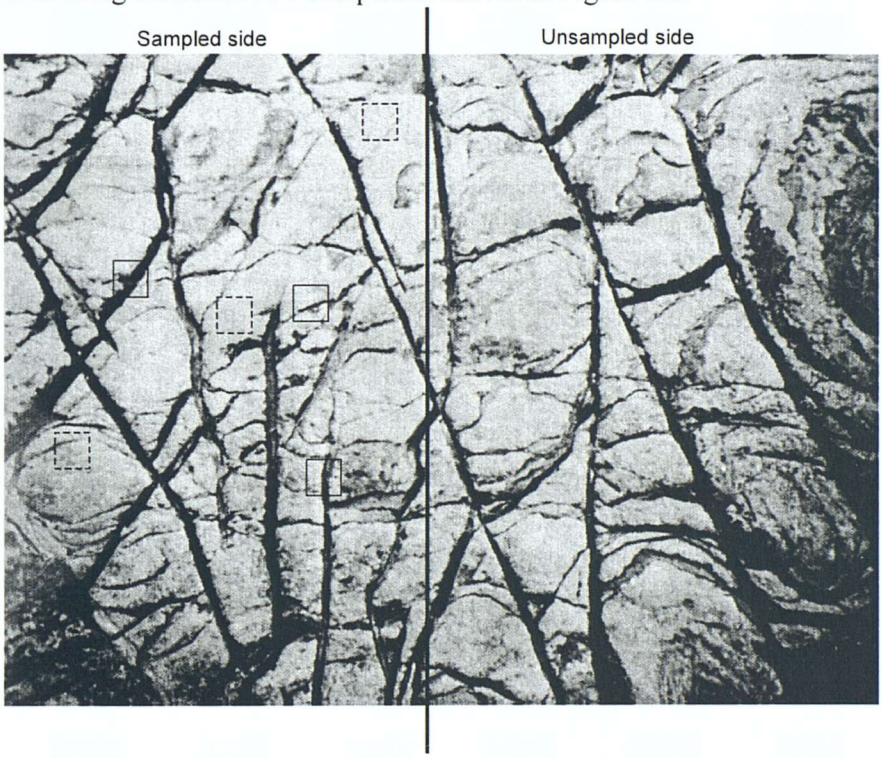


Figure 6.2: Image sampling process for Fracture Discriminator. Solid squares: samples of image content containing a fracture. Dashed squares: samples of image content not containing a fracture.

It should be noted that the sampling process should be viewed as a routine part of the methodology. For every set of imagery to be traced, samples of patches with fractures and patches without will be required. ANN training will also need to be

completed too. Sampling is a process that took no longer than 10 minutes for the data set presented in Figure 6.3. Fast, PCs enable the training process to be completed in a matter of minutes rather than hours (see below).

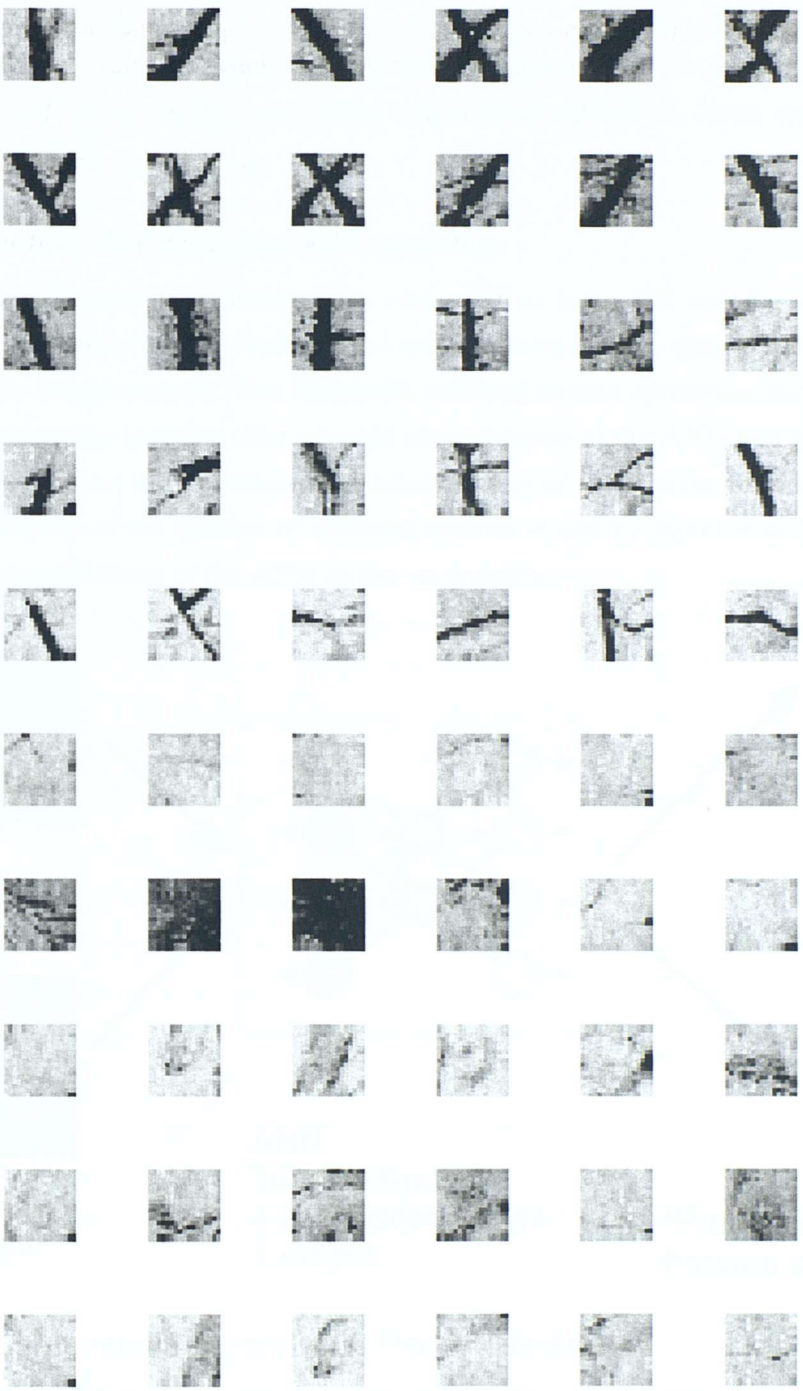


Figure 6.3: Training data set for Fracture Discriminator, sampled from Figure 6.2.

Each sample in the training data set was labelled to discriminate between those that contained fracture image content and those that did not. Those that did were assigned a label value 1, and those that did not were assigned a label value 0. The

ANN was trained to reproduce the correct label value for each of the 60 sample image patches. The output label values within the training data set were chosen so that the output of the ANN could be loosely interpreted as the *probability* that a given image patch contains fracture image content.

The pre-processing stage for the training data set consisted of normalisation of every pixel value, (dividing each of them by 255) and mapping the result, I_j into the range $-0.5 \leq I_j \leq 0.5$. In the case of the output label values O_k, these were simply mapped into the range $-0.5 \leq O_k \leq 0.5$.

6.2.2 *Fracture discriminator ANN specifics*

A multilayered perceptron architecture was used to learn the training data. This perceptron had 16 x 16 input nodes, 4 x 4 artificial neurons in a single 'hidden' layer and a single output neuron. The sigmoidal artificial neuron activation function was used throughout the network. The adopted mesostructure of the ANN was determined heuristically, subject to requirements of minimisation of the number of hidden layers and minimisation of the number of artificial neurons in hidden layers in order to keep training and recall times of the ANN as low as possible.

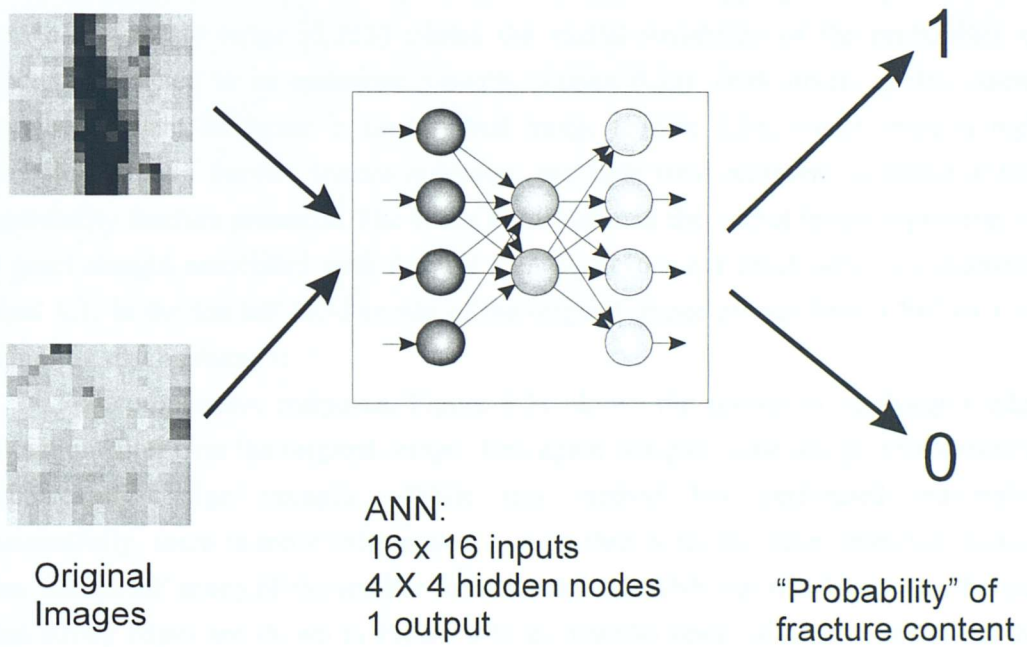


Figure 6.4: ANN training concept for the Fracture Discriminator

The generalised delta rule with back-propagation of error was used to train the network for 100 cycles. The momentum and learning rate training parameters were set at 0.9 and 0.1 respectively throughout all neurons within the network. Weights were initialised randomly with values in the interval [-0.3,0.3]. The training time was

approximately 5 minutes using the ANSIM ANNs training package on a IBM compatible PC equipped with a Pentium processor running at 120 MHz. The trained weights and thresholds, which represent a complete specification of the Fracture Discriminator for repeatability studies, are presented in Appendix C2.

6.2.3 Performance of the Fracture Discriminator

After 100 cycles the root mean square training error was 0.0049 (0.5%) and the maximum error at output was 0.0170, that is 1.7%.

The Fracture Discriminator module was then used to scan the whole input image, starting from its top left hand corner, row by row, to its bottom right hand corner. 69% of the pixels within the whole image did not form any part of the training process and thus the performance of the Fracture Discriminator module on the whole image serves also as module validation. To reinforce this point it should be recalled that training samples were only taken from the left hand side of the whole image. Performance of the Fracture Discriminator on the right hand side demonstrates how well the trained network is able to generalise to new data, unseen during training.

The raw output of the Fracture Discriminator is a value between 0 and 1 which was earlier likened to a probability of fracture presence. Mapping these 'probabilities' into a grey scale range [0,255] allows the spatial variability of the probability of fracture presence to be examined visually (Figure 6.5e). Dark pixels in this output image represent locations in the original image (Figure 6.5a) where there is high probability that a fracture feature is present, and light areas represent locations of low probability fracture presence. The white border around the output image represents an 8 pixel margin associated with the use of a 16 x 16 input patch area: for example, pixel 1,1, in the top left hand corner of the original image cannot form a full 16 x 16 sampling patch around it.

For comparative purposes, Figure 6.5c shows the results of applying a edge detection filter over the original image. This again is a grey scale image with darkness representing 'edge strength'. While this method has performed reasonably successfully, there is more information in 6.5c than 6.5e; the edge detection routine has 'identified' many of the smaller fractures that the ANN has not. However, the fact that strong edges are shown in Figure 6.5c as 'parallel lines' of pixels shows that the edge detector has detected changes in image contrast rather than detected fractures in the image content. Nevertheless, it is evident that such images of edge detection filters may be useful in the interpretation to determine fracture aperture.

Most edge detection algorithms used in digital image processing are principally designed to extract contrast edges and vary in implementation (that is, whether the frequency or spatial domain is used), and number and size of the filters used. Millar

(1992) conducted a comparative study of the edge detection methods applied to digital images of rock masses, none of which could be considered completely suitable to the task of fracture recognition. This classification is presented in Appendix C3.

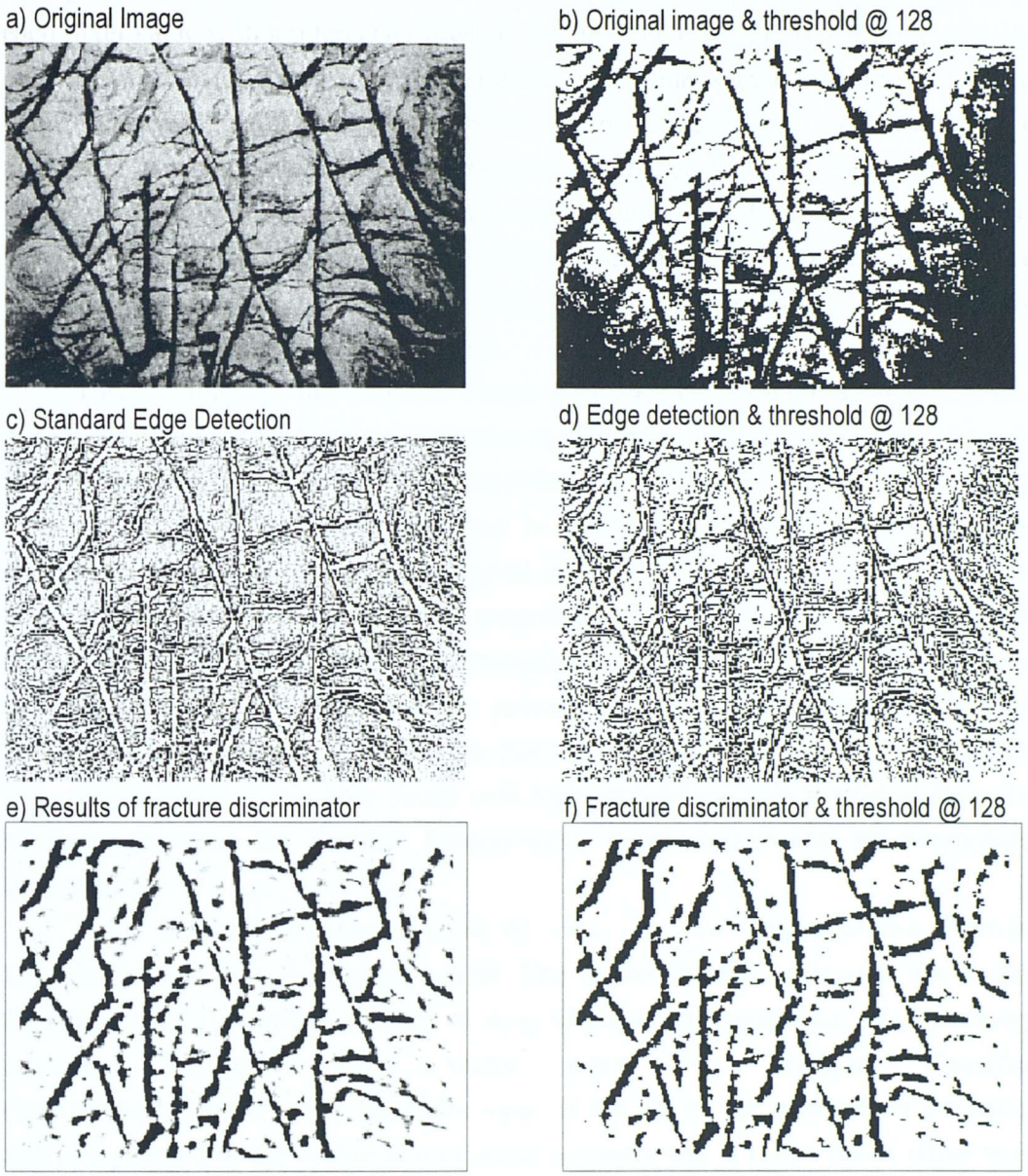


Figure 6.5: Results of using the Fracture Discriminator to scan the original image, with other image processing operations for comparison.

Qualitatively, the Fracture Discriminator has performed well when the fracture image content is found within a high contrast region of the image. However, it has also performed well in low contrast portions of the image (for example in the lower left hand corner of the original.

It is difficult to quantitatively assess the results of the Fracture Discriminator module and perhaps comparison with other more widespread methods (such as, edge

detection) is the best measure, but clearly it has managed to detect most of the fractures present within the original image.

The fracture probability map that Figure 6.5e represents can be rendered 'crisp' through the application of a threshold operation. Thresholding involves comparison of each pixel value with a reference (threshold) value. The pixel value is reassigned 1 (or equivalently 255 for a grey scale image) if the pixel value is greater than the threshold value, and reassigned 0 (or equivalently 0 for a grey scale image) if it is less than the threshold. Thresholding with a threshold value of 127 (or equivalently 0.5 for a probability map) has been applied to each of the three images discussed thus far in this Section: the original Figure 6.5a, the edge detection results, Figure 6.5c, and the Fracture Discriminator results, Figure 6.5e. The results are shown in Figure 6.5b, Figure 6.5d, and Figure 6.5f, respectively.

Thresholding of the original image produces encouraging results for this particular image but performance degrades rapidly when the image contrast is low. In imagery with many shadows, the thresholding would highlight shadow boundaries. Consequently, thresholding alone cannot be utilised as a general purpose tool for fracture extraction from rock mass digitial imagery. Thresholding of the grey scale edge detection results removes a large proportion of the background noise but cannot be said to be in a form suitable for determination of data and parameters of importance in rock mechanics without further processing. Thresholding of the Fracture Discriminator probability map confirms that, where a fracture has been found within the original image, it has been found with high probability. This is evident from the similarity between the Fracture Discriminator results image and its thresholded counterpart.

From the rock mechanics point of view, the raw output of the Fracture Discriminator is only moderately useful. The reason for this is because the output format is raster (i.e., still comprises an array of pixels) and most rock mass geometry analysis programmes need vector input. For example, *FracMan* (http://fracman.golder.com/) integrates most, if not all, of the most robust statistical analysis procedures applicable to rock mass geometry and is vector based rather than raster based. To render the data generated by the Fracture Discriminator suitable for manipulation by this off-the-shelve software, it must first undergo a raster to vector conversion. The basic idea behind these modules is that clusters of pixel positions that denote fracture locations are aggregated and then replaced by a vector equivalent for example, linear segments, Bezier curves or cubic splines.

The output of the Fracture Discriminator is similar to the 'ravine pixel' map reported by Reid and Harrison, (2000) in that it is a type of pre-processing stage involved in the automatic fracture identification process, before vectorisation. Reid's

ravine labelling involves the assembly of a neighbourhood matrix of grey level derivatives and the application of expert rules based on the eigenvalues of the matrix. It is thus a completely automatic process in comparison to the ANN method presented here. Reid also developed a vectorisation procedure which could be applied to the thresholded Fracture Discriminator results. Hadjigeorgiou *et al*, 2003 adopt an ANN methodology to facilitate the 'linking' together of vector features based on radiometric image content and information from discontinuity and edge vector primitives arising from a preliminary raster to vector conversion.

However, various commercially available computer programs are available that can also implement the raster to vector conversion process with suitably pre-processed input imagery. One computer program that the author has tested that can do this in a robust manner for Fracture Discriminator output is Scan2CAD (Softcover International Ltd, 2001). The results of post-processing the Fracture Discriminator output with Scan2CAD are presented in Figure 6.6 below.

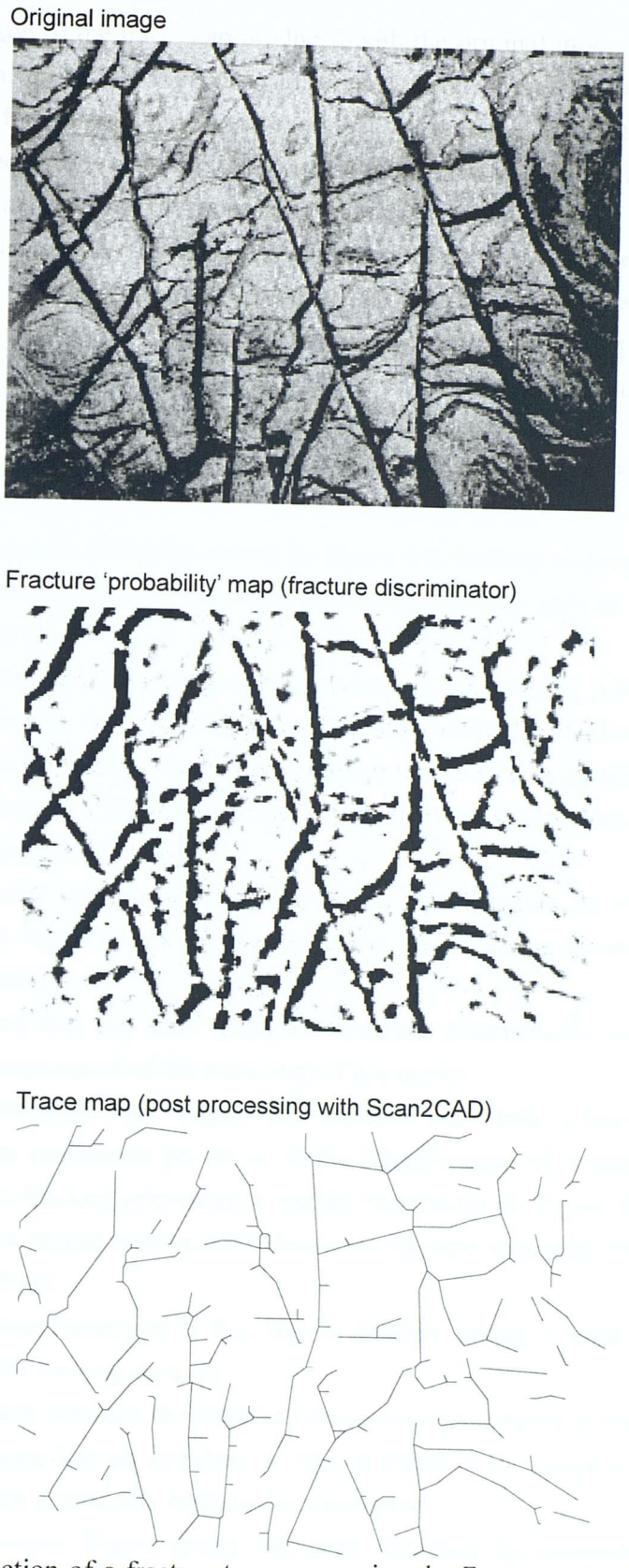


Figure 6.6: Production of a fracture trace map using the Fracture Discriminator after post-processing with a generic raster to vector conversion utility.

Comparison of the trace map produced with the original image shows that there are some significant, discrepancies in logical connectivity (information on the 'importance' of individual fractures, expressed in terms of their aperture, has also been 'lost'). Fractures that can be clearly seen to be persistent in the original image are disconnected in the vector representation. However, comparison of the vector trace map with the Fracture Discriminator output shows that the raster to vector process has been conducted faithfully. Thus it is evident that the 'problem' lies with the Fracture Discriminator output rather than the raster to vector conversion process.

Quantitative comparative studies between Fracture Discriminator trace maps and manually interpreted trace maps have not featured in the current work. However, general observations on the basis of the mapping experience of the author are as follows:

1) The fracture trace map shown in Figure 6.6 is likely to produce reasonably accurate statistics for measures of fracture intensity such as RQD and mean spacing.
2) A fracture persistence distribution derived from Figure 6.6 is likely to be truncated at both ends compared to the 'real' distribution because high persistence fractures have been 'broken' and low persistence fractures have been 'missed'. However, the difference between interpreted and 'real' mean persistence may be small.
3) Block size statistics utilising 'stochastic ray' measures on Figure 6.6 (such as that used by the *Fracman* code) may also produce representative estimates.
4) Analyses that use trace maps to condition discontinuity set identification procedures may find the trace map of 6.6 useful.

Detailed inspection of Figure 6.6 reveals one area where the Fracture Discriminator has performed poorly is in the identification of fracture termination conditions. Some fracture intersections readily observable in Figure 6.5a have been missed in Figure 6.5e and consequently the vector fracture trace map. Possible reasons for this are as follows.

(i) Insufficient examples of this type of fracture image content were included within the training data set.
(ii) The match between the fracture intersection geometries of examples within the training data set and those in other portions of the image is insufficient to yield high probability fracture discrimination.
(iii) The Fracture Discriminator has been confused by contrast based image content that is not fracture related, for example, shadows.

(iv) Rotated copies of the samples of fracture image content need to be added to the training data set to introduce some rotation invariance into the Fracture Discriminator ANN weights.

The motivation for reason (iv) is that in Figure 6.5a, where two 'bold' fractures intersect, the Fracture Discriminator has identified one of them effectively, but has missed the second that forms the X intersection. As four X intersection exemplars feature in the training data set it is thought that reason (i) above is unlikely.

It can be argued that if the fracture trace map is to be used to condition a statistical analysis of rock mass discontinuity geometry, then missing a few intersections of a certain type is not particularly important. It is more important that there are a sufficient number of examples of the various termination types present within the trace map such that it can be considered to be statistically representative.

However, the deficiency identified will have an important impact on some rock mechanics analyses that may follow the creation of the fracture trace map. Fracture termination conditions are of high importance for analyses that that rely on fracture connectivity. Computerised block theory, such as the implementation of Ikegawa, (1992), will clearly suffer if termination or persistence conditions are not properly represented. However, for some, simpler kinematic feasibility analyses relying on statistical analysis of fracture geometry, for example, those presented by Priest, (1985, 1992), it is the presence of a fracture of a given orientation that is most important as fracture persistence is assumed to be effectively infinite.

A second example where a deficiency in the representation of termination conditions is problematical concerns analyses involving fluid flow through the joints and fractures of the rock mass. For these types of analysis, any fracture that intersects with only one other fracture should also be discounted from the analysis as it does not contribute to flow through the fracture network. A good description of fracture termination condition, describing how fractures are linked to each other, is thus crucial for the success of such analyses.

In order to tackle the deficiencies, a second ANN module is developed within this Chapter—the 'Fracture Classifier'. This module is not only concerned with where a fracture is found but also with classifying fractures according to their termination condition. However, before this is addressed, the next sub-section examines the usefulness of the Fracture Discriminator output in defining a measure of fracture intensity.

6.2.4 A fracture intensity measure based on fracture discriminator output

The Fracture Discriminator ANN has been trained to produce an output value ranging between 0 and 1 which can be interpreted as a probability of fracture occurrence within the 16 x 16 sampling window. As the sampling window scans the image, so the Fracture Discriminator reports its output, pixel by pixel, producing a raster field of occurrence probabilities. The last sub-section considered a raster-to-vector conversion process as a possible post-processing step to allow quantities characterising the discontinuity geometry to be determined subsequently. However, having established a measure of probability occurrence, a useful measure of fracture intensity can be calculated from it, without the need for vectorisation.

Various measures of fracture intensity are available to rock engineers. The most widely known is the measure $P_{1,0}$, the number of fractures occurring per metre along a scanline. In this expression, the first sub-script refers to the number of spatial dimensions for the sampling region and the second refers to the dimension of the occurrence feature. For a simple count, the value of this second subscript is zero. If the fracture intensity were to be described as the total (or average) length of discontinuity traces intersecting a unit length of scanline, then the appropriate symbol for this quantity would be $P_{1,1}$.

A fracture intensity measure $P_{2,2}$ is the area of a fracture exposed within a two dimensional sampling window, A_f, per unit area of that sampling window, area A, that is:

$$P_{2,2} = \frac{A_f}{A} \tag{6.1}$$

A value of $P_{2,2} = 1$ implies all of the area of the sampling window is occupied by a fracture; a value of $P_{2,2} = 0$ implies none of the area of the sampling window is occupied by a fracture. A value of $P_{2,2} = 1$ could occur if the sampling window area is orientated parallel to the exposed surface, and of a scale such that it encompasses the fracture area completely. A value of $P_{2,2} = 1$ could also arise if the orientation of the pole of the sampling window is parallel to the strike of the fracture surface and its scale is chosen to be smaller than the fracture aperture. As with other measures of fracture intensity, $P_{2,2}$ is scale and orientation dependent. If orientation dependence is ignored, then the expression $P_{2,2}(A)$ recognises that the measure is a function of scale.

As the Fracture Discriminator output preserves fracture aperture reasonably faithfully (see Figure 6.5), an estimate of the fracture intensity measure $P_{2,2}(A)$ can be determined. Suppose $P_f(i,j)$ is the probability of fracture presence assigned to a pixel. The pixel has area within the image, $A(i,j)$, and is located i rows and j columns,

from the origin of the fracture probability map that has total area, A. Then, over the whole fracture probability map:

$$P_{2,2}(A) = \frac{\sum_{i,j} P_f(i,j) \cdot A(i,j)}{A}. \tag{6.2}$$

where:

$$A = \sum_{i,j} A(i,j) \tag{6.3}$$

As the area of each pixel in the fracture probability map is identical, expression (6.2) becomes:

$$P_{2,2}(A) = \frac{\sum_{i,j} P_f(i,j)}{n_A}. \tag{6.4}$$

where n_A is the number of pixels in the fracture probability map. To examine scale effects, the fracture probability map can be sub-sampled to produce values of fracture intensity applicable at smaller sampling window sizes, applying equation 6.4 directly. The applicable scales of estimates are tracked by definition of variable A', which is the normalised area of the sub-sample, that is:

$$A' = \frac{n}{n_A}. \tag{6.5}$$

where n is the number of pixels in the sub-sampled window. A plot of $P_{2,2}(A)$ versus A' is shown in Figure 6.7.

Observations from Figure 6.7 are as follows:

1) When the sampling window size is small, there is appreciable variance in estimates of $P_{2,2}(A)$.
2) As the sampling window size increases, the variance in $P_{2,2}(A)$ reduces.
3) The mean value of fracture intensity over the range $0.25 \leq A' \leq 0.9$ slowly varies around 0.212 m^2/m^2.
4) The variance in $P_{2,2}(A)$ is relatively constant in the range $0.55 \leq A' \leq 0.9$.
5) Variance in $P_{2,2}(A)$ increases when $A' \geq 0.9$.
6) The values $P_{2,2}(A)$ determined about the centre pixel of the image, labelled P22c(A) in Figure 6.7, lie with the 95% confidence interval envelope about the mean values of $P_{2,2}(A)$.

The measure $P_{2,2}$ appears robust when computed in this manner from the fracture probability map output by the Fracture Discriminator. The reason why the variance increases when $A' \geq 0.9$ is the relative absence of fractures in the periphery

of the original image (especially the lower LHS and the extreme RHS of Figure 6.2). If these areas are considered to belong to a different structural domain than the fractures evident within the bulk of the image, then as the sampling window area approaches the total area of the image, the $P_{2,2}$ metric effectively averages across two structural domains.

A method of estimating a scale invariant $P_{2,2}$ is thus proposed by the author for use in conjunction with output produced using the Fracture Discriminator methodology. It is defined as the value of $P_{2,2}$ that occurs over scales where the variance in $P_{2,2}$ is constant, practically determined by preparing a graph such as Figure 6.7 and reading off.

The fracture intensity estimates computed from the Fracture Discriminator output rely on the assumption that the ratio of pixel area to 'real world' area of rock exposure is a constant for every pixel in the fracture probability map. If this ratio does not remain constant, for example due to the perspective geometry imposed by the camera that acquired the original image, a correction to the basic procedure proposed is required. A 'real world' area is assigned to each pixel in the fracture probability map. Equation (6.4) does not hold, but equations (6.2) and (6.3) do hold if the $A(i,j)$ are taken to be the real world areas assigned to each pixel.

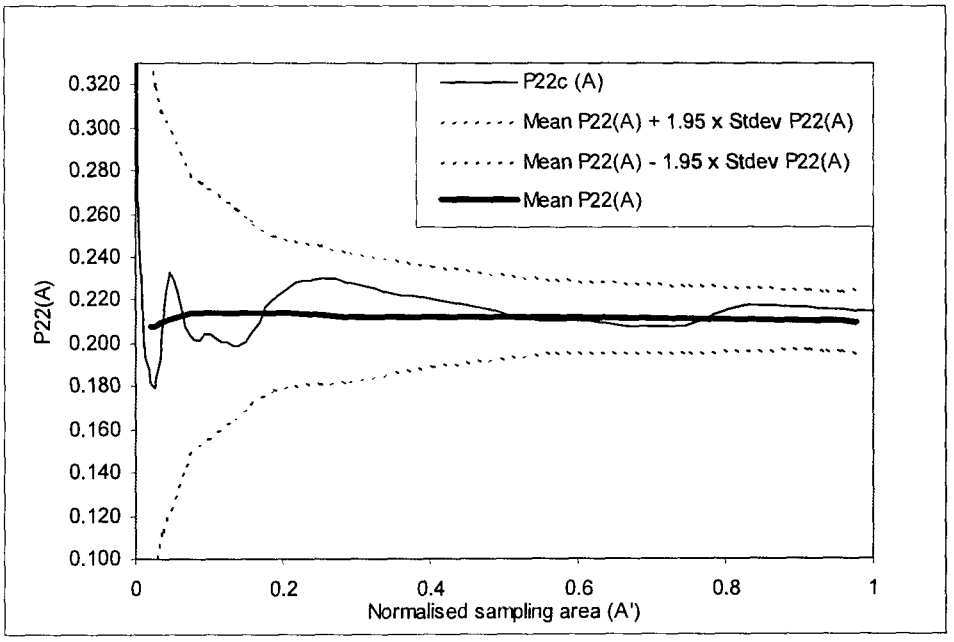


Figure 6.7: Plot of fracture intensity $P_{2,2}(A)$ versus normalised sampling area, A'. The light continuous line, P22c(A), shows fracture intensity calculated at different scales, centred on the fracture probability map, Figure 6.5e. The heavy continuous line shows the mean $P_{2,2}(A)$ calculated from all possible sub-samples of the fracture probability map Figure 6.5e. 95% confidence limits are shown with dashed lines. The plot can be interpreted to yield a scale invariant estimate of mean fracture intensity $\overline{P}_{2,2} \approx 0.212$ m^2/m^2.

6.3 The Fracture Classifier

The development of the Fracture Discriminator was *data-driven* as sampling was used to provide examples for the ANN to learn. In contrast, the Fracture Classifier, developed by the author, uses an *expectation-guided* approach where the examples are synthesised and the noise handling capabilities of ANNs are exploited.

6.3.1 General concept of the Fracture Classifier

An expectation guided approach to fracture identification with an ANN was investigated because it was thought that this may lead to improved performance in identification of fracture termination conditions, in comparison to the Fracture Discriminator's results of the last Section. With an expectation guided approach, it was recognised that there may have to be compromises made in terms of how well other geometric measures are faithfully represented, in favour of determination of fracture connectivity.

The image is scanned in a search for features that are similar to a defined set of templates. When a match for one of the templates is found within the image, this location is labelled. In the approach taken here, labelling takes the form of replacement with the template. This idea is presented in Figure 6.8 below which illustrates an image undergoing the process. While connectivity is broadly preserved, the locations of fractures that have been replaced with the template recognised may not be. However, these binary maps could represent the logical topology between fractures and be used in conditioning output from other fracture identification routines, for example the Fracture Discriminator.

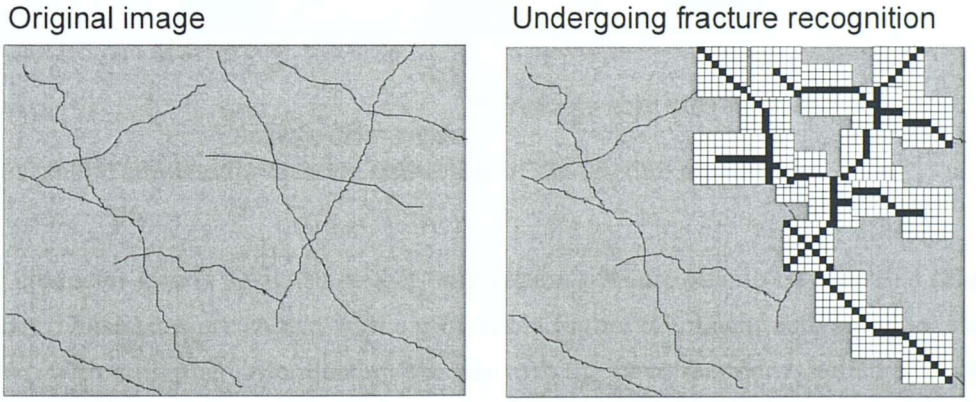


Figure 6.8: Illustration of expectation guided search with imagery for features matching defined 7 x 7 templates.

Figure 6.8 shows that for the best agreement between a series of identified templates and the disposition of the features they describe in the imagery, a degree of overlap between the templates may be required. For rock mass imagery in particular, the idealised templates would have to perform robustly in the presence of noise

introduced into the image from the acquisition process and also that due to the roughness of terrain and rock texture.

The set of templates that are used in the search by the Fracture Classifier are discussed next.

6.3.2 Data set for the Fracture Classifier - The Fracture Alphabet

The fracture alphabet is a set of bitmap image patches each of which is 7 pixels wide by 7 pixels tall. The fracture alphabet contains 256 different 'characters'. Each character is based on a 7 x 7 grid, with the fracture feature centred on it (Figure 6.9).

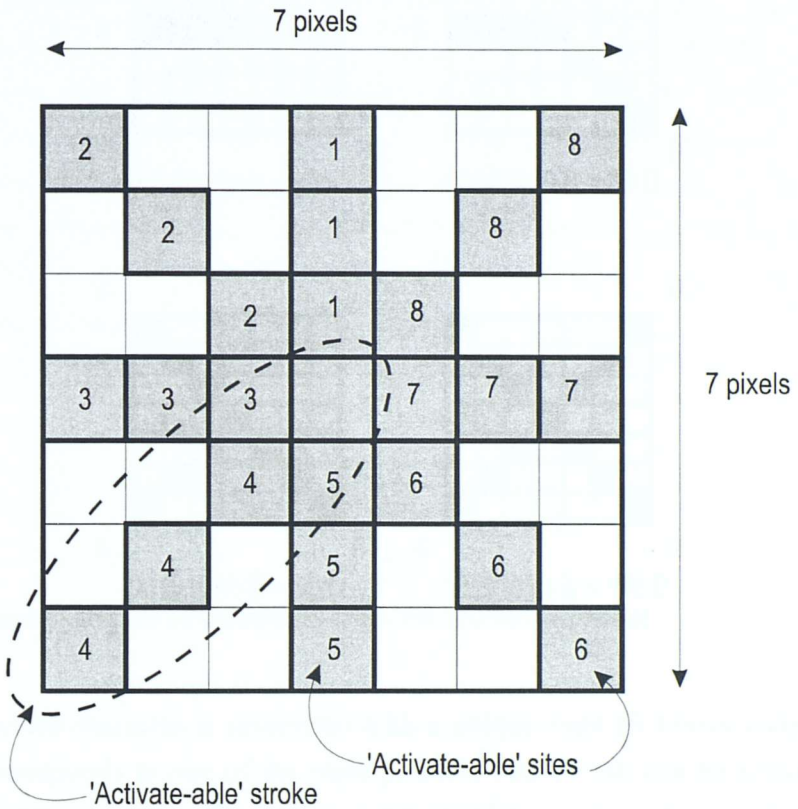


Figure 6.9: Template for fracture characters of the fracture alphabet.

The character grid is designed so that only certain sites are able to be activated (turned black). These sites are shown with a grey fill in Figure 6.9. Each 'activate-able' site is shown with a number. A collection of sites with the same number is called a 'stroke'. A fracture character consists of one or more activated strokes. The central site of the grid is always activated unless strokes 1 to 8 are all deactivated. Fracture characters with, one, two, three and four active strokes are shown in the following figure, Figure 6.10.

The upper left character in Figure 6.10 has stroke 3 activated, and represents a segment of a discontinuity that terminates in solid rock at its right end. The upper

right character has two strokes activated, stroke 3 and stroke 6. This character is taken to represent a segment of a discontinuity that spans the sampling window but which also contains an slope-break point. The lower left character represents two fractures, one that spans the sampling window diagonally from upper left to bottom right and a second that terminates against the first from the lower left hand corner. The lower right character represents the intersection of two discontinuities both of which span the sampling window diagonally forming a 'X' intersection.

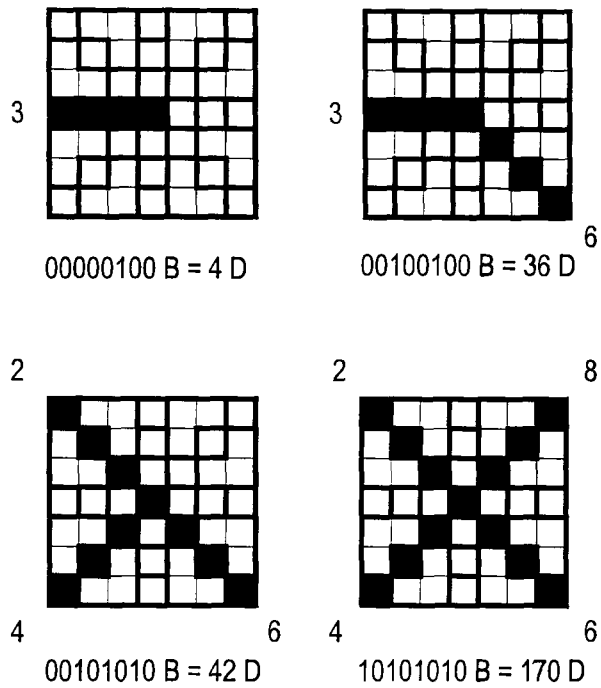


Figure 6.10: Four examples of characters from the fracture alphabet

Each fracture character is associated with a unique eight bit binary code. Each bit position corresponds to one of the eight possible strokes that can be activated to represent a fracture character. The left most bit position corresponds to stroke 1, the rightmost, stroke 8 with intermediate bit positions representing strokes from the upper left diagonal anticlockwise (see Figure 6.9). Each bit assumes a value of either 1 or 0 depending on whether for a specific fracture character the corresponding stroke is activated or not, respectively.

In Figure 6.10 the eight bit binary codes are also shown for the four fracture character examples. When these eight bit binary numbers are expressed as decimals, a number between 0 and 255 results, which is taken as the label or 'name' of a fracture character. For example, it is thus sensible to refer to character 42 of the fracture alphabet meaning (uniquely) the bottom left character shown in Figure 6.10. The 256 different fracture characters in the fracture alphabet thus correspond to all possible combinations activated strokes. In proceeding in this way, all possible orientations of

the four example characters (and more) are accounted for, and all termination conditions. A Visual BASIC for Applications program to reproduce the entire fracture alphabet shown in Figure 6.11 is presented in Appendix C4.

Note that use of a 7 x 7 pixel 'template' for fracture characters limits the sensitivity of fracture orientation to 45°. Increasing the size of the template or sub-sampling by interpolation onto a circular 'grid', would allow the orientation sensitivity into increase.

The fracture alphabet is the data set used to train the second ANN module. Instead of generating training exemplars from real rock mass digital imagery (as is the case for the training data set for the Fracture Discriminator) here, synthesised exemplars have been created. For training with the neural network training software, the input patterns were scaled to map into the interval [-0.5,0.5] (by subtracting 0.5 from the stroke activation values). The binary target output values were also mapped into the interval [-0.5,0.5] (by subtracting 0.5 from each bit value). The network output was then subject to a preliminary post-processing step by mapping back to the interval [0,1]. The network outputs are thus not binary but real, and ideally close to the 'crisp' target bit values (that is: 0 or 1).

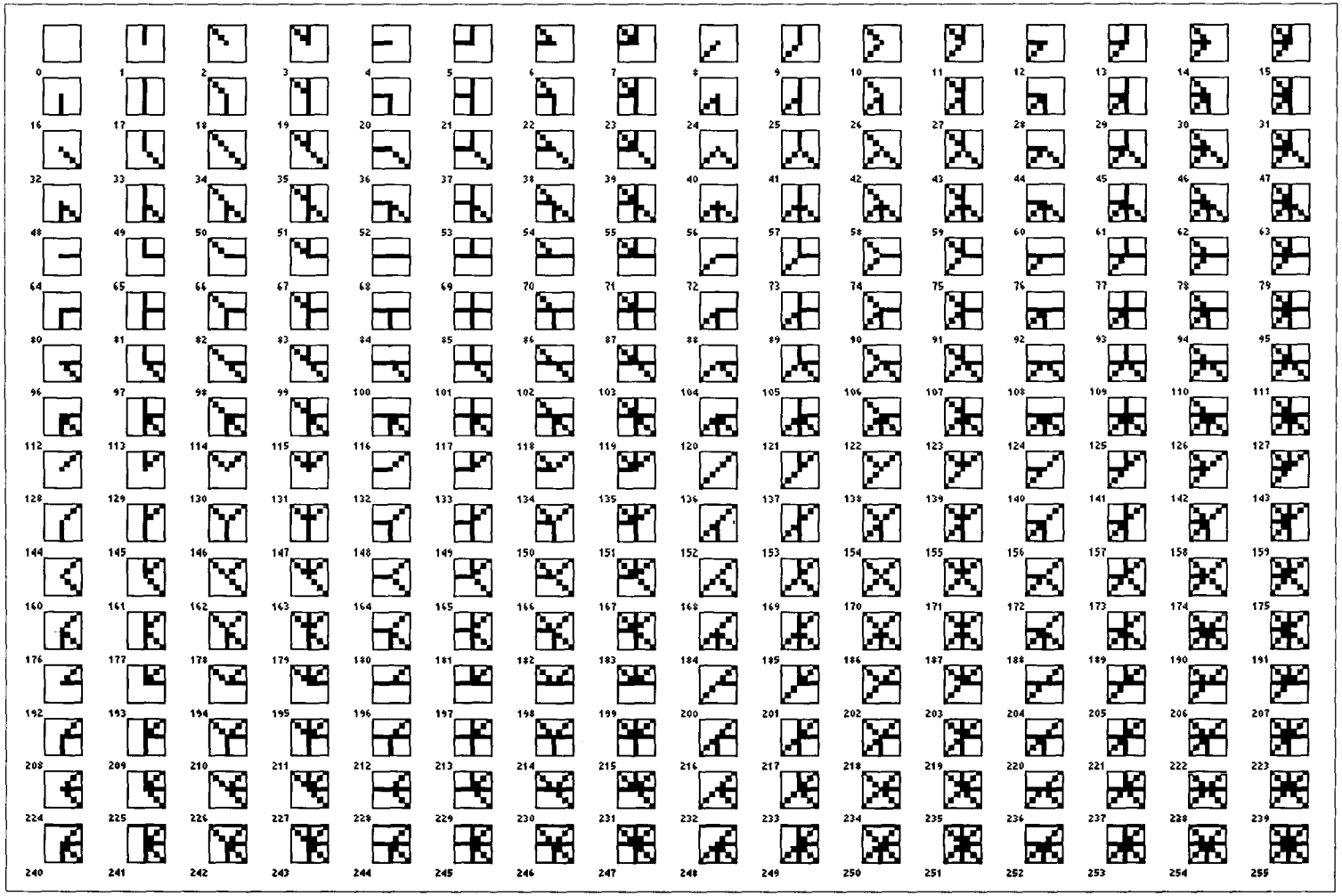


Figure 6.11 The Fracture Alphabet

6.3.3 Fracture classifier ANN details

During training and after pre-processing, the 7 x 7 pixel bitmap fracture alphabet exemplars are presented at the inputs of the network.

A multilayered perceptron architecture was used to learn the training data. This perceptron had 7 x 7 input nodes (to match the input character grid), 8 artificial neurons in a single 'hidden' layer and 8 artificial neurons at its output layer (to match the target 8 bit binary code). The sigmoidal artificial neuron activation function was used throughout the network. The adopted mesostructure of the ANN was determined heuristically, subject to requirements of minimisation of the number of hidden layers and minimisation of the number of artificial neurons in hidden layers in order to keep training and recall times of the ANN as low as possible.

The generalised delta rule with backpropagation of error was used to train the network. All network weights were initialised with random values from the interval [-0.3,0.3], momentum and learning rate parameters were 0.6 and 0.1 respectively, throughout all the artifical neurons of the network.

Six training sessions were undertaken, differing only in the specific random values used to initialise the network weights. After 700 training cycles, all of these networks had a root mean squared training error of 0.05 or 5% of the dynamic range of the target values. The sets of weights of each the six networks were compared connection by connection. A key observation made was that weights in the same network positions differed appreciably. This is supported by the plot shown in Figure 6.12 below which shows the distribution of coefficient of variation of network weights, with each weight position within the networks forming a datum.

As can be seen from the plot, the mean coefficient of variation is around 10 which means that the magnitude of the standard deviation is ten times the average of six weights in the same network position but different networks. This shows, categorically, that the specific value of a weight within an ANN is meaningless alone, and only has value when considered with the other weights contained within its network. Consequently it is incorrect to endow specific interpretations to individual (trained) weights contained within a (multilayered perceptron) ANN. This is not the case for internal or external activations.

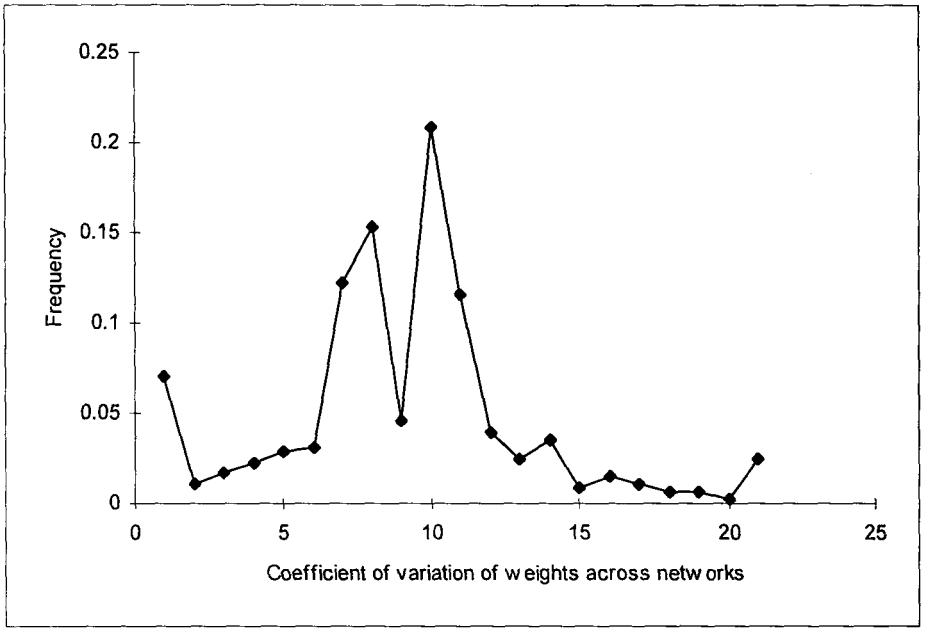


Figure 6.12: Distribution of the coefficient of variation of weights within six networks trained to reproduce the fracture alphabet. The average value of around 10 indicates that the magnitude of the standard deviation of the weight values is 10 times the mean of the values for network weights in the same 'location' within the ANN.

A seventh training session was undertaken, but this time the network was trained for longer: 912 cycles. At this stage, the root-mean-square error over the whole fracture alphabet was 0.009 (0.9%) and the maximum error was 0.099 (9.9%) at one element of one output vector. This trained network then formed the subject of further analysis which is discussed in the following.

6.3.4 Fracture classifier post-processing

During normal operational use after training, the ANN is presented with image patches from real digital imagery and outputs eight real numbers varying between 0 and 1 (after preliminary post-processing as mentioned previously). Two types of further post processing have been adopted to render the output 'crisp'.

(i) Each of the real outputs of the ANN are subjected to a high level and low level thresholding operation:

$$O_k := 1 \quad \text{if } O_i > \Theta_1$$
$$O_k := 0 \quad \text{if } O_i < \Theta_2$$

where Θ_1 and Θ_2 are the high level and low level thresholds respectively and $\Theta_1 = 1 - \Theta_2$. If any one of the eight network outputs have a real value that is intermediate to these thresholds, it is taken that no fracture is present in the input image content, and thus a null fracture classification results.

(ii) The sum of squared differences between the output vector and each of the ideal, binary output vectors are calculated. The ideal vector that results in

the minimum sum of squared differences is taken as the final output of the ANN.

The second type of post-processing will always succeed in that it will always be possible to find a closest matching ideal output binary vector (i.e. a null classification is not possible). It is thus possible for type (ii) post-processing to 'crisply' misclassify an input image patch. If this classification is correct, as will be seen subsequently, it is highly likely that type (i) post-processing will recover the same binary vector. If incorrect, the vectors produced by the respective post processing operations will differ. Thus to avoid misclassification, both type (i) and type (ii) post-processing operations are undertaken: in normal operation, a fracture recognition success occurs when the answer provided by both types of post-processing agree.

6.3.5 Image noise: a clarification

Using signal processing terms, the signal within rock mass digital imagery is the network of fractures apparent within it that is the goal for recovery. The remaining image content should thus be considered 'noise'. As demonstrated with the digital images and histograms presented earlier in this Chapter, the magnitude of the signal is often overwhelmed by the magnitude of the 'noise'. Consequently, recovering the signal represents a particularly challenging task. If an ANN methodology is to succeed, we must first be certain that it will perform robustly in the presence of noise.

A more conventional source of noise is also contained within rock mass imagery. This is the noise introduced in association with sensors and devices used to obtain the imagery. The magnitude of this noise compared to the magnitude of 'noise' using its meaning from the previous paragraph is taken to be insignificant providing that, subjectively, a fracture network is visible within high contrast grey scale imagery. Thus, in the following, when the term noise is used, it is the inherent variability of the rock mass image content that is being referred to.

The rough surfaces, mineral crystal grain composition, irregular topography and shadows are only a few of the reasons why the spatial distributions of pixel values within rock mass digital imagery are highly variable. One possible approach to recovering the signal is to a produce a functional model for the remaining image content and use it to filter the imagery by subtracting the model's predictions. In this instance, however, production of a luminescence model to account from roughness, mineral composition, topography, and shadows in rock mass imagery is at least as difficult as the current objective. An alternative approach is to design a filter that can 'pick out' the signal from noisy imagery. This task is made considerably easier if the signal content is assumed to be restricted to a finite set (in this case, the fracture

alphabet). Robust behaviour in the presence of noise is thus the key feature of ANNs that is being exploited within the work presented in this Section.

6.3.6 The effect of additive noise on fracture alphabet characters

The fracture characters in the fracture alphabet that were used to train the Fracture Classifier neural network are completely free of radiometric (grey level) noise. Thus the adequacy of the Fracture Classifier neural network cannot be assessed on the basis of the post-training root-mean-square error (0.009) or maximum error (0.099) alone.

Additive random noise was added to the fracture alphabet exemplars in increments. After each increment, the performance of the Fracture Classifier was assessed in terms of the number of noisy input fracture characters correctly classified. Here, it is assumed that random additive noise provides a reasonable model for the systematic noise associated with surface roughness, mineral composition, etc. As pre-processing of the input values without additive noise will always map all inputs into the range [-0.5,0.5], the uniform random noise was added to the fracture characters according to the following relations.

$$\text{if } I_j = 0 \text{ then } I_j := I_j + \left(\text{URV}\left(\text{P}/100\right)\right) - 0.5$$

$$\text{if } I_j = 1 \text{ then } I_j := I_j - \left(\text{URV}\left(\text{P}/100\right)\right) - 0.5$$

where P is the amount of noise to add, expressed as a percentage, and URV is a uniformly distributed random variable with values confined to the range [0,1). These relations ensure that the (pre-processed) input values always lie within the range [-0.5,0.5) which is taken to be a sufficiently good approximation to the required range [-0.5,0.5].

The effect of these functions on the pre-processed input values for increasing additive noise is shown in histogram format in Figure 6.13 below. As increased noise is added, so the number of occupied bins within these histograms increase: for example, when the additive noise is 30%, the uniformly distributed random variable has a range [0,0.3] which means that bins between (0.5-0.3 =) 0.2 to 0.5 and (-0.5+0.3=) -0.2 to -0.5 become filled approximately uniformly. The relative heights of LHS and RHS groups within the histograms reflects the original proportions of activated and non-activated grid cells across the fracture alphabet. When the additive noise becomes 50%, the LHS and RHS groups meet, and subsequently superimpose when the additive noise becomes yet greater.

The histograms in Figure 6.13 show that the additive noise model creates a stylised representation of the bi-modal histogram form expected when an image patch from real rock mass imagery contains a fracture (see Appendix C1). Subjecting characters of the fracture alphabet to additive noise as previously detailed thus

represents a fairly rigorous test of how the Fracture Classifier would be expected to perform in the presence of radiometric noise present in 'real' rock mass imagery.

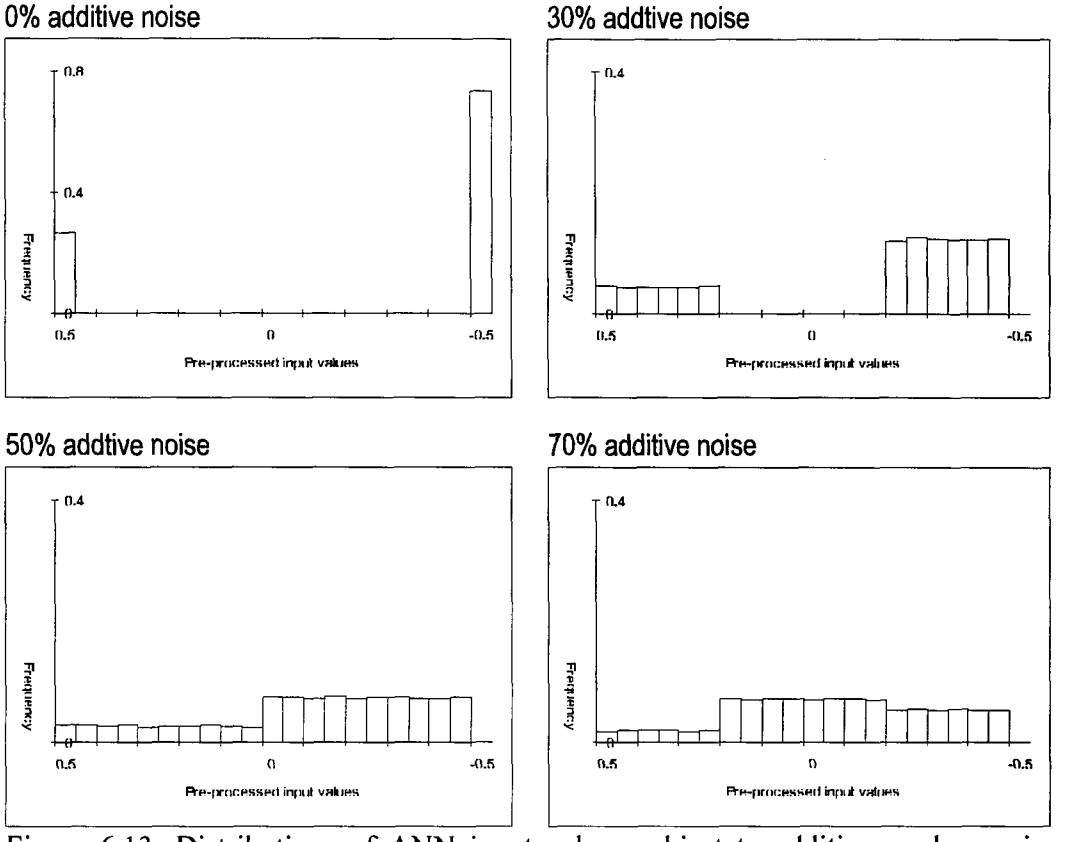


Figure 6.13: Distributions of ANN input values subject to additive random noise. Histograms taken over the complete fracture alphabet. Average frequencies taken over 5 simulations.

To gain further understanding of the effect of additive noise on characters of the fracture alphabet, consider Figure 6.14 below. This shows grey scale images of five characters of the fracture alphabet subjected to increasing additive noise. At 0% additive noise, the images represent the ideal fracture characters. At 30% additive noise, the fracture content is still clearly (subjectively) recognisable. At 50% recognition of the signal is still possible although difficult—even for a human! At 70% additive noise, a human would stand little chance of recognising the signal at all.

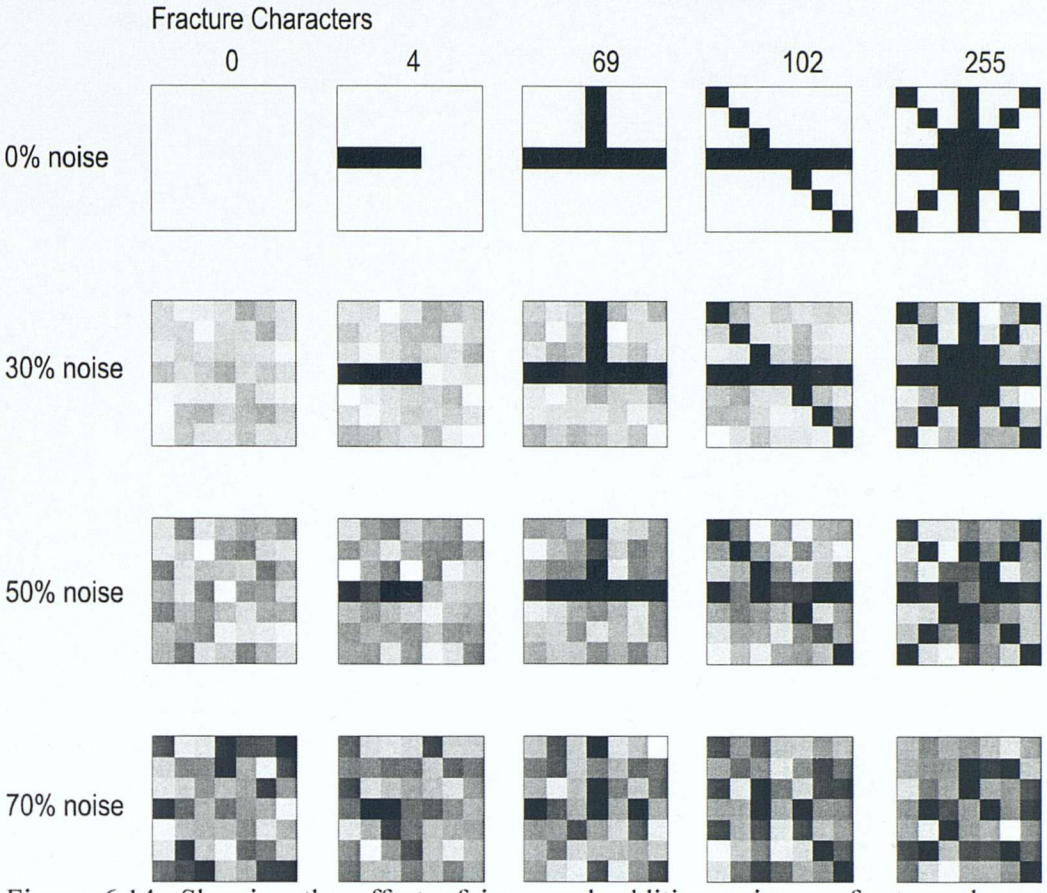


Figure 6.14: Showing the effect of increased additive noise on fracture character image content.

6.3.7 Performance of the Fracture Classifier

Having presented the details of the various tests undertaken, the results of performance testing the ANN Fracture Classifier module are now presented.

Figure 6.15 below shows a plot of fracture character recognition accuracy versus the low hard limit threshold Θ_2 (remembering that $\Theta_1 = 1 - \Theta_2$ always). Recognition accuracy is expressed as the percentage of fracture characters correctly identified when all of the characters of the fracture alphabet are presented to the neural network in turn. As using the additive noise model involves the use of a uniformly distributed random variable, the value for recognition accuracy is actually an average of five distinct presentations of the fracture alphabet for the same control parameters (% noise, and Θ_2). The error bars for each datum indicate the maximum and minimum recognition accuracy obtained during each set of 5 simulations.

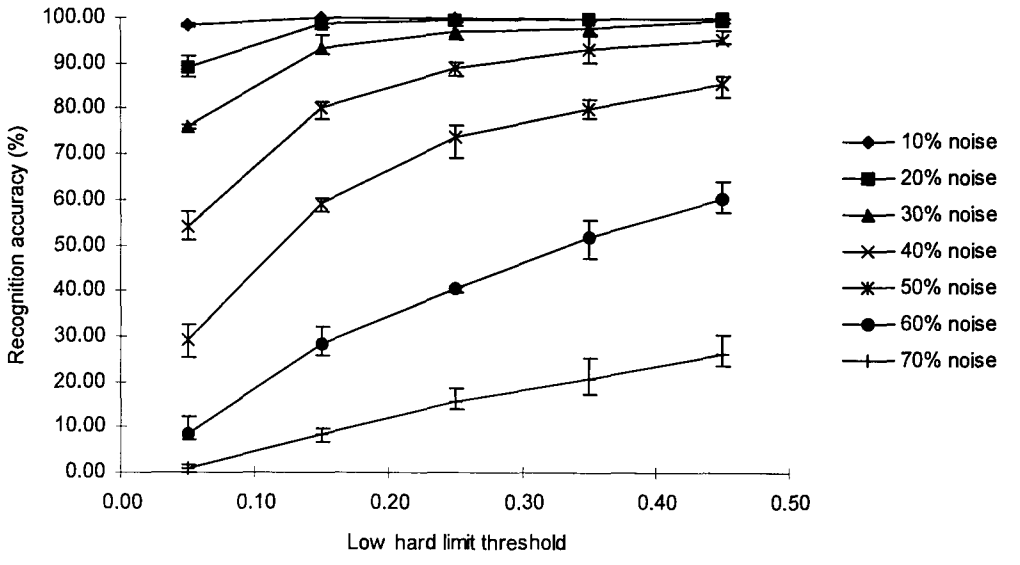


Figure 6.15 Fracture character recognition performance with type (i) post-processing, plotted against Θ_2 varying between 0.05 and 0.45 for increasing amounts of additive noise. Error bars show range of recognition accuracy across 5 sets of simulations. With additive noise less than 50% (see Figure 6.14) and Θ_2=0.45, recognition accuracy is maintained above 85%.

Observations from Figure 6.15 are as follows:

1. When the amount of additive noise is low (10%), recognition accuracy is near perfect and largely independent of Θ_2.
2. Recognition accuracy is higher when the additive noise is reduced.
3. Recognition accuracy is higher when Θ_2 becomes greater (a lowering of the 'rigour' of the threshold).
4. Variability in recognition accuracy becomes greater when the amount of additive noise is increased.
5. (Tentative) Variability in recognition accuracy becomes greater when Θ_2 becomes greater although more than 5 simulations for each datum are probably required to gain representative samples (the amount of processing time precluded their enlargement).
6. (Important) Recognition accuracy can be maintained at above 85% when additive noise is as great as 50% and when Θ_2=0.45. This observation is more clearly noticeable from Figure 6.16 shown below.

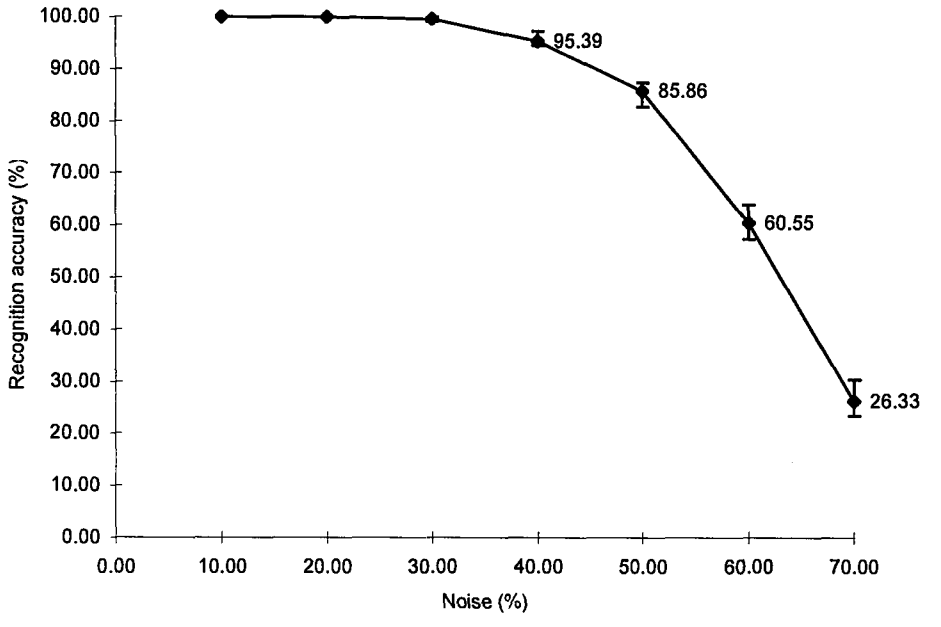


Figure 6.16: Fracture character recognition performance with type (i) post-processing and Θ_2=0.45, which confirms that recognition accuracy is maintained above 85% even when additive noise is 50%. Numbers on the diagram report recognition accuracy (%).

A further important observation on the recognition accuracy of the Fracture Classifier with type (i) post-processing can be made: the recognition accuracy does not degrade rapidly with increasing additive noise, starting from 10%. The additive noise must become greater than 50% for an appreciable reduction to occur. It is worth noting that the signal (fracture image content) was subjectively recognisable with difficulty at 50% additive noise in Figure 6.14. The subject (the reader) is also probably assisted by the 'answer' being on the 0% row of Figure 6.14.

The Fracture Classifier performance when type (ii) post-processing is adopted is shown in Figure 6.17 in comparison to the performance curve for type (i) post-processing. Assessment of the Fracture Classifier performance with type (ii) post processing was again undertaken for five distinct presentations of the fracture alphabet to the ANN and an average taken due to the stochastic nature of the simulations. Figure 6.17 shows that the form of the performance curve is similar to the comparable curve for type(i) post-processing, and that the recognition accuracy is improved a little.

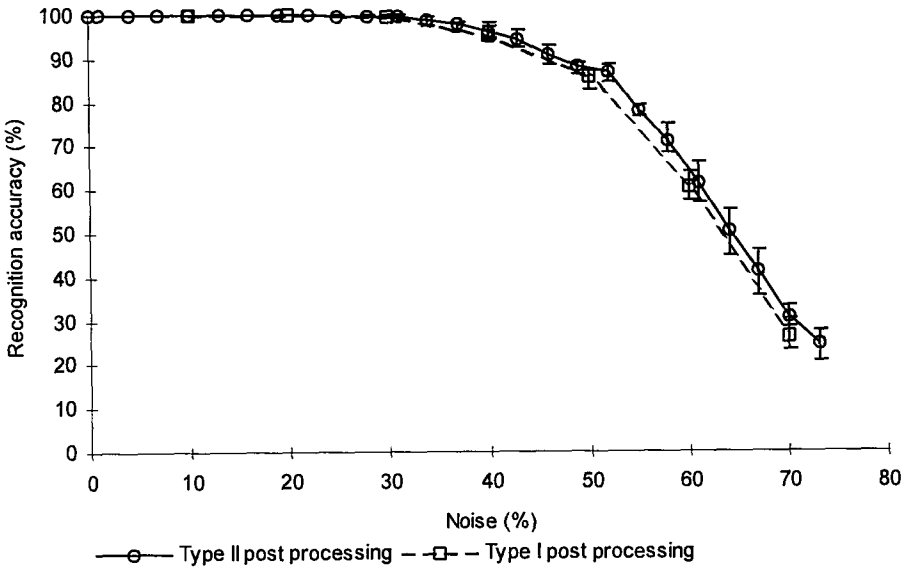


Figure 6.17: Comparison between Fracture Classifier performance for type(i) [Θ_2=0.45.] and type(ii) post-processing.

The following charts (Figure 6.18) compare the performance of type (i) and type (ii) post-processing fracture character by fracture character. The top chart shows that when the additive noise is low (< 30%) both methods are likely to produce the same (correct) results. Where the light grey and the dark grey meet, both methods had 100% recognition accuracy for a given fracture character.

When the additive noise increases, the likelihood that both methods recognise the same (correct) result reduces, their average recognition accuracy being expressed not as a percentage but as a variable in the interval [0,1]. Consequently, adopting both methods of post-processing in routine fracture recognition can provide a coarse measure of the reliability of the recognition process.

If the results from both methods agree, then the 'noise' of the image content can be estimated as being $\leq$30% and it can be said that a reliable recognition has been made. If the results do not agree, then the recognition process could be considered unreliable or the 'noise' of the image content can be said to be >30%.

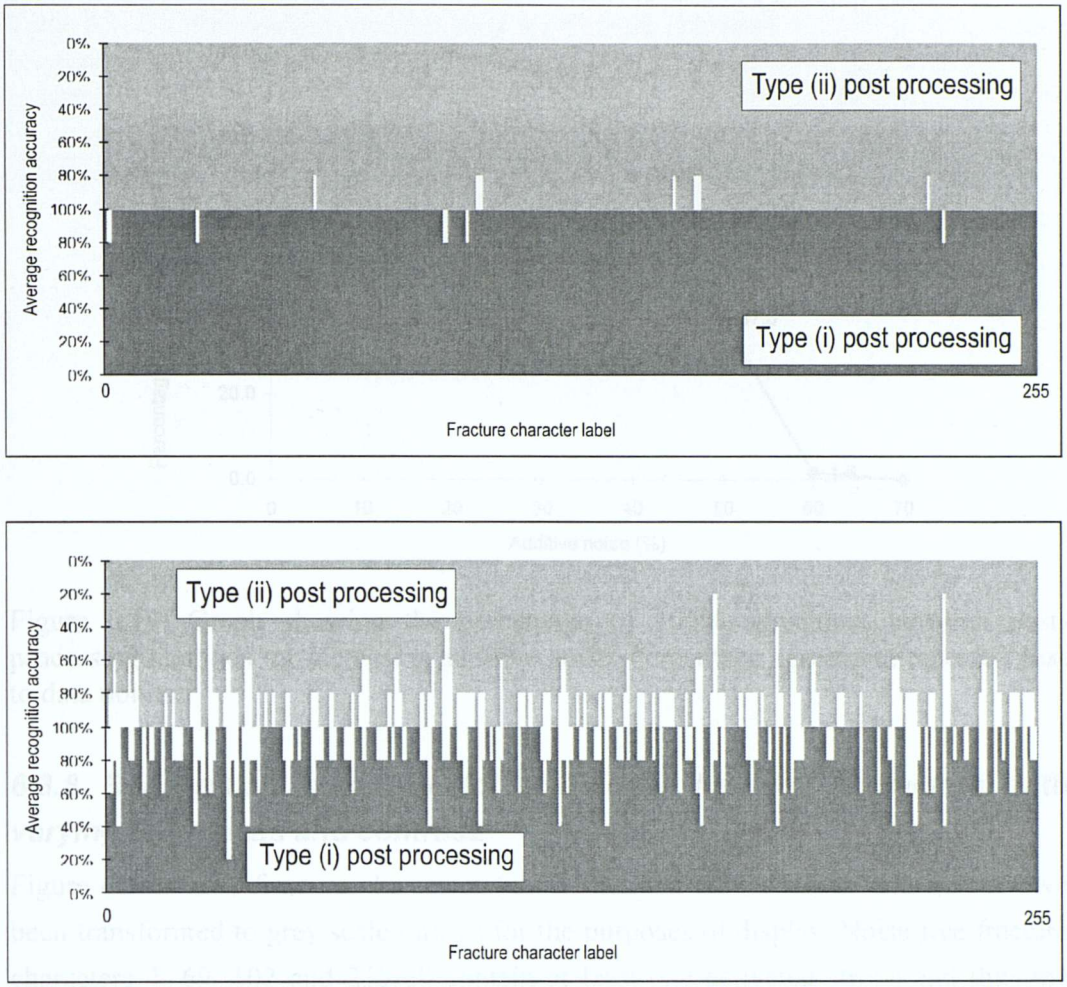


Figure 6.18: Average recognition accuracy for type (i) and type (ii) post-processing for (top) 30% additive noise with 95.7% agreement between the two methods and (bottom) 50% additive noise with 37.6% agreement. Where dark grey columns and light grey columns meet, 100% recognition accuracy is obtained by both type (i) and type (ii) post-processing methods.

The graph shown in Figure 6.19 extends this idea by plotting the percentage of 100% agreement between both post-processing methods as the quantity of additive noise is increased. This curve could form the basis for more rigorous recognition reliability estimates or, more generally, estimation of the 'noisy-ness' of rock mass image content: for example, if type (i) and type(ii) post processing methods result in agreement for 71.4% of the input patches, the quantitative assessment of the 'noisy-ness' of the imagery would be 40%.

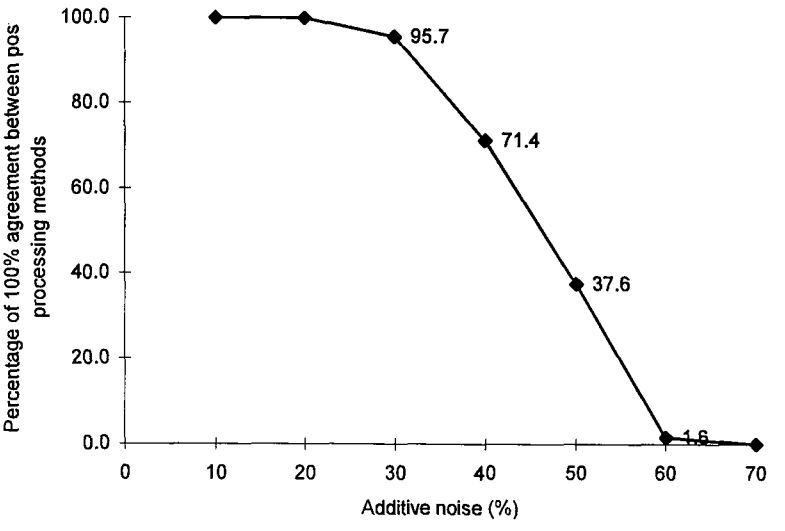


Figure 6.19: Graph showing the percentage of 100% agreement between post-processing methods for increasing additive noise. Percentage agreement reported next to data points.

6.3.8 Recognition accuracy for individual fracture characters with varying brightness and contrast.

Figure 6.14 shows fracture characters (noise free and with additive noise) that have been transformed to grey scale images for the purposes of display. Noise free fracture characters 4, 69, 102 and 255 all contain at least one activated stroke and thus the dynamic range of the grey scale image corresponding to each fracture character will be fully utilised, that is the black 'fracture pixels' will have grey values of 0 and the white 'background pixels' will have grey values of 255. Even when large amounts of additive noise are added it is still likely that the full dynamic range of the grey scale images will be nearly fully present. In other words, all the recognition accuracy assessments conducted thus far have been conducted on fracture characters corresponding to low brightness, high contrast grey scale images.

In operational fracture recognition practice, these conditions of brightness and contrast cannot be guaranteed to exist for input patches from a digital image of a real rock mass. In the case of the Fracture Discriminator, the sample fracture patches sensitise the trained ANN to the prevailing brightness and contrast features of the image. In the case of the Fracture Classifier, it is necessary to additionally explore the effects of varying brightness and contrast on fracture character recognition accuracy.

Tables 6.1 to 6.5 below show the fracture character labels recovered, one table each for the fracture characters 0, 4, 69, 102 and 255, varying contrast from 0 to 100% and brightness from 0 to 60% (Type (ii) post-processing only).

Table 6.1 Recognition performance for fracture character 0 varying contrast and brightness. Values indicate the fracture character label found.

Character **0**	Brightness (%) 0	10	20	30	40	50	60
Contrast (%)							
0	255	255	255	255	255	207	0
10	255	255	255	255	207	0	0
20	255	255	255	207	0	0	0
30	255	255	207	0	0	0	0
40	255	199	0	0	0	0	0
50	199	0	0	0	0	0	0
60	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0

Table 6.2 Recognition performance for fracture character 4 varying contrast and brightness. Values indicate the fracture character label found.

Character **4**	Brightness (%) 0	10	20	30	40	50	60
Contrast (%)							
0	255	255	255	255	255	207	0
10	255	255	255	255	79	0	0
20	255	255	255	79	4	4	0
30	255	255	79	4	4	4	4
40	255	71	4	4	4	4	4
50	71	4	4	4	4	4	4
60	4	4	4	4	4	4	4
70	4	4	4	4	4	4	4
80	4	4	4	4	4	4	4
90	4	4	4	4	4	4	4
100	4	4	4	4	4	4	4

Table 6.3 Recognition performance for fracture character 69 varying contrast and brightness. Values indicate the fracture character label found.

Character **69**	Brightness (%) 0	10	20	30	40	50	60
Contrast (%)							
0	255	255	255	255	255	207	0
10	255	255	255	255	207	64	0
20	255	255	255	79	69	64	0
30	255	255	79	69	69	0	0
40	255	77	69	69	69	0	0
50	109	69	69	69	68	0	0
60	69	69	69	69	68	0	0
70	69	69	69	69	68	0	0
80	69	69	69	69	68	0	0
90	69	69	69	69	68	0	0
100	69	69	69	69	68	0	0

Table 6.4 Recognition performance for fracture character 102 varying contrast and brightness. Values indicate the fracture character label found.

Character **102**	Brightness (%) 0	10	20	30	40	50	60
Contrast (%)							
0	255	255	255	255	255	207	0
10	255	255	255	255	127	66	0
20	255	255	255	127	102	64	0
30	255	127	127	102	102	64	0
40	127	127	127	102	102	64	0
50	127	127	102	102	102	64	0
60	127	126	102	102	102	64	0
70	126	102	102	102	102	64	0
80	102	102	102	102	102	64	0
90	102	102	102	102	102	64	0
100	102	102	102	102	102	64	0

Table 6.5 Recognition performance for fracture character 255 varying contrast and brightness. Values indicate the fracture character label found.

Character **255**	Brightness (%) 0	10	20	30	40	50	60
Contrast (%)							
0	255	255	255	255	255	207	0
10	255	255	255	255	255	207	0
20	255	255	255	255	255	207	0
30	255	255	255	255	255	207	0
40	255	255	255	255	255	239	0
50	255	255	255	255	255	239	0
60	255	255	255	255	255	239	0
70	255	255	255	255	255	239	0
80	255	255	255	255	255	239	0
90	255	255	255	255	255	239	0
100	255	255	255	255	255	239	0

Observations that can be made from Tables 6.1 to 6.5 are as follows:

1. In general, the optimal brightness for accurate fracture character recognition is around 20-30%.
2. At this brightness level, image contrast can vary from 50% upwards resulting in accurate fracture recognition.
3. Higher contrast input image patches result in accurate recognition performance, providing they are not so bright that they are over-saturated ('white out').
4. Fracture character recognition resulting in a label value of 255 is likely to mean: (i) the input patch has insufficient contrast, (ii) is too dark or (iii) contains fracture character 255(!).

5. Fracture character recognition resulting in a label value of 0 is likely to mean: (i) the input patch is too bright or (ii), does not contain fracture image content.

6.4 Sample image patch transformations

The previous Section has reported on a rigorous assessment of the performance characteristics of the Fracture Classifier with respect to: post-processing operations, additive noise, brightness and contrast. Overall the ANN Fracture Classifier seems to perform robustly in the presence of appreciable noise and operational values of contrast and brightness, and hard limiting threshold values for type (i) post processing have been found.

As this assessment programme has solely been based on manipulation or radiometric distortion of the idealised fracture alphabet characters, the Fracture Classifier module still awaits its 'acid test'—its performance in extracting fracture features from real rock mass imagery.

Before this is done, two further aspects must be discussed. Firstly, an input template size of 7 x 7 pixels may not always be suitable for image patches from real rock mass imagery that contain fracture image content. Consequently image patch rescaling to 7 x 7 may be required. Secondly, the observations listed in Section 6.3.8 support the use of additional contrast stretching as part of pre-processing. As it is not part of the main thesis presentation, an image patch rescaling procedure developed by the author that is less 'lossy' than sub-sampling methods and less computationally intensive for large patches than interpolation methods is presented in Appendix C5 together with a program listing. Details on the author's implementation of contrast stretching are presented in Appendix C6 together with a program listing.

6.5 Testing the Fracture Classifier with real rock mass imagery

6.5.1 Image scanning procedure

The Fracture Classifier was used to scan Figure 6.2 using sampling windows (patch sizes) of dimension varying from 7 x 7 to 32 x 32 pixels. The scanning window is centred on each pixel in turn, from top row to bottom row, from left to right (Figure 6.20). A non-evaluated border region results because the finite dimensions of the sampling window which means that edge pixels cannot be analysed as centre pixels.

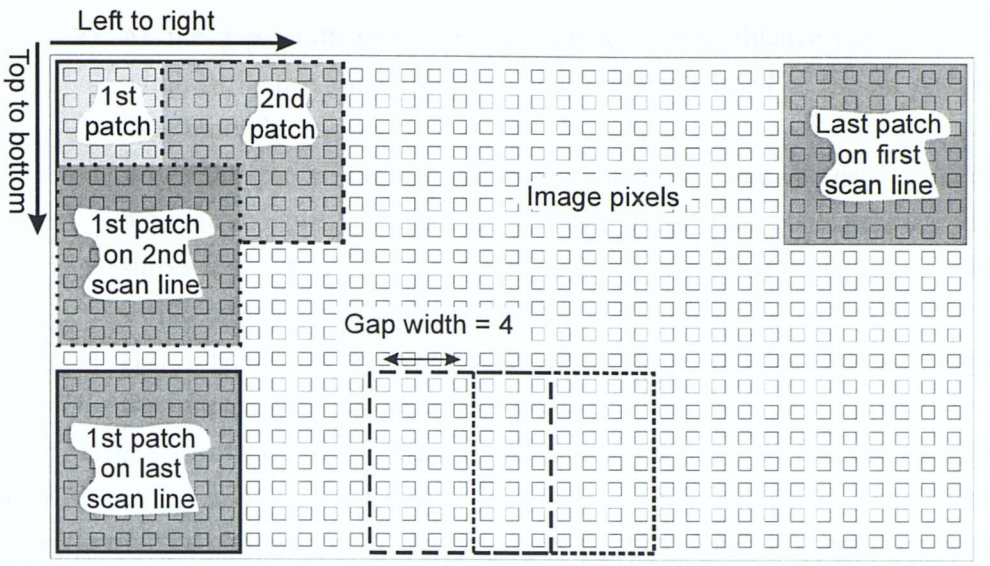


Figure 6.20: Image scanning procedure with fracture classifer. Illustration shows a patch size of 7 x 7 pixels and a gap width of 4 pixels.

Image rescaling with weighted average reduction (Appendix C5) was undertaken for sampling patch sizes greater than 7 x 7 pixels. Contrast stretching (Appendix C6) was applied to each sampled image patch. Fracture identification was assumed to occur when type (i) and type (ii) Fracture Classifier post processing agreed. The lower limit hard threshold was set at Θ_2=0.3. The Fracture Classifier output values were decoded to fracture character labels. If the label was 255, no fracture was taken to be present because the input image is assumed to be too dark – in accordance with the observations of Section 6.3.8. The fracture character labels were used to form a binary template of the same dimensions as the sampled image patch. The binary fracture character patch then replaced the grey scale information in the original image. Weighted average expansion was applied to the binary fracture character patches for sampling patch dimensions greater than 7 x 7.

The same scheme was run repeatedly with an increasing gap between patch centre pixels. For a given patch size, n, the process was repeated for gap widths of 2, 4, 8 and n. In the case of a gap width of n, there is no overlap between sequentially considered patches.

6.5.2 Results of a Fracture Classifier scan of Figure 6.2

The results of applying the Fracture Classifier to Figure 6.2 are presented in Figure 6.21. There are four important observations that can be made from Figure 6.21.

1) When the gap width is set at 1, the binary map produced by the Fracture Classifier resembles the fracture network evident in the input image.

2) As the gap width increases, the degree of resemblance reduces. At a gap width of 4, the binary maps of the Fracture Classifier appear to represent little more than noise.
3) When the gap width is equal to the patch size, that is, there is no overlap between sampled patches, the number of fracture characters identified is surprising low. Also, the fractures 'recognised' do not match those that would be expected to be recovered.
4) As the patch size increases, with gap width = 1, a coarser binary map of the fracture network results, as expected.

The performance of the Fracture Classifier on this real fracture imagery was viewed as disappointing, particularly when the results for patch size = gap width were considered. The performance of the Fracture Classifier under this condition appears to have defied the objectives it was 'designed' to achieve. Instead of a stylised, yet topologically correct, representation of the fracture network that ensures fracture intersections are properly represented, the fracture characters identified have disjoint strokes when compared with their neighbours and the binary map is effectively un-useable.

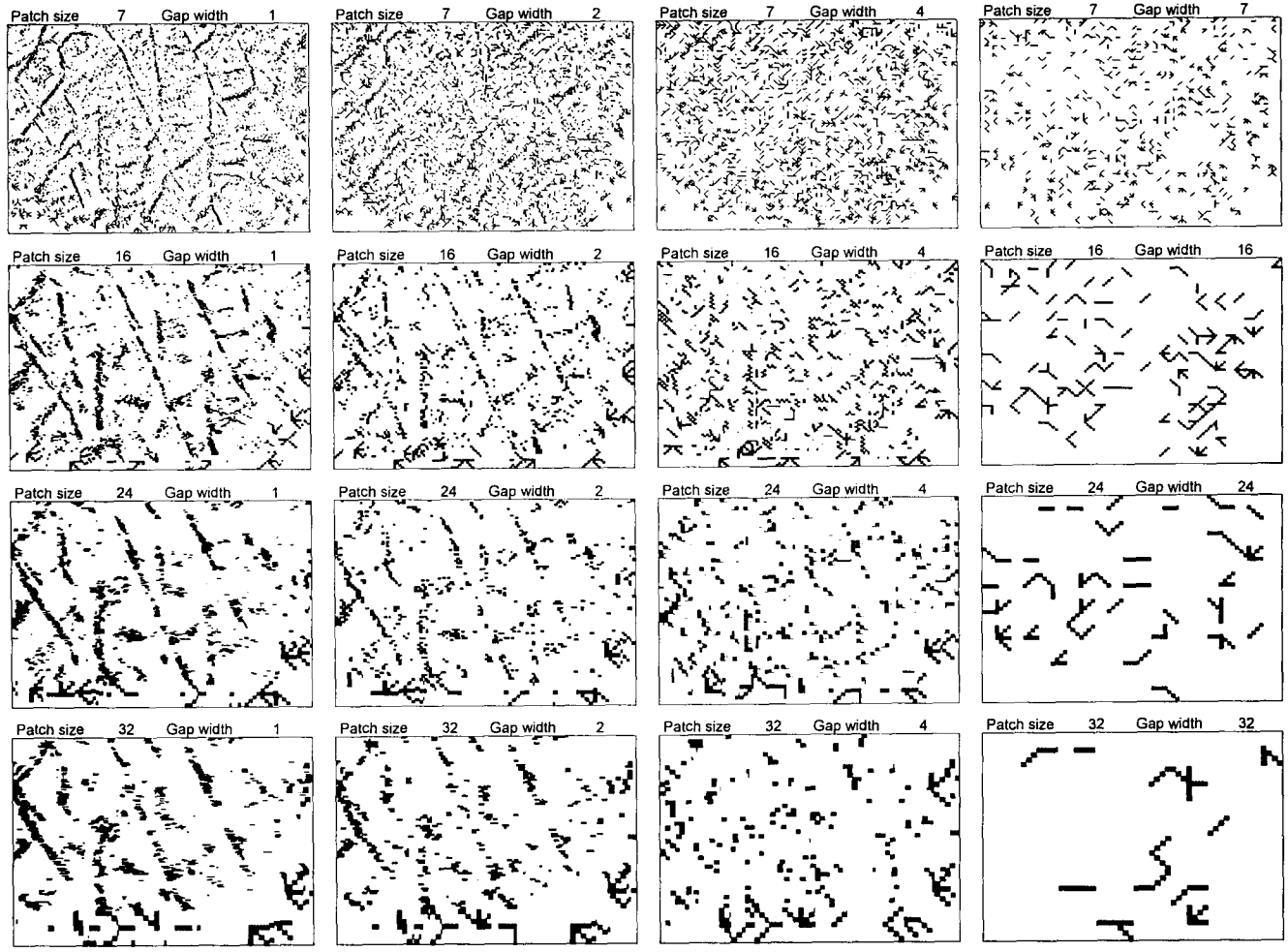


Figure 6.21: Binary fracture maps resulting from application of Fracture Classifier to Figure 6.2, for varying patch sizes and gap widths.

The best observable performance occurs with patch size = 7 and gap width = 1, as there does appear to be coherence within the binary map. However, with gap width = 1, the Fracture Classifier acts more like the Fracture Discriminator as information on the activated strokes of the character identified is effectively overwritten by the activated strokes of the next patch in the image scan. When the Fracture Classifier scans the image with gap width = 1, the result is that the activation state of the top left-hand-side pixel indicates whether or not *any* fracture was found when the patch was scanned. The Fracture Classifier is much more computationally expensive than the Fracture Discriminator because of the additional pre-processing and post-processing task associated with the former. Thus if the objective was to simply provide a measure of fracture presence, the Fracture Discriminator is the right tool.

Figure 6.22, which summarises 4 'offline' fracture recognition attempts on character 34 on radiometric noise free input patches, illustrates the one of two causes that were identified for to explain the poor Fracture Classifier performance. The second reason occurred when real image patches were being analysed. The contrast stretching process 'blackened' already dark, but clearly background, pixels which 'fooled' the classifier into returning activation values corresponding to the presence of a fracture stroke.

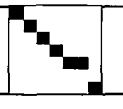
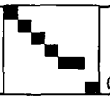
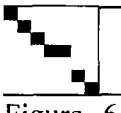
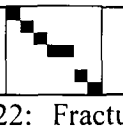
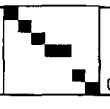
Input Patch	WIR Patch	Stretched Patch	ANN Outputs Stroke 8	7	6	5	4	3	2	1	Post Process Type I	Type II	Result
			0.004	0.001	0.983	0.019	0.021	0.000	0.977	0.002	34	34	34
			0.004	0.001	0.451	0.014	0.003	0.000	0.993	0.004	Failed threshold	2	Not Classified
			0.003	0.001	0.348	0.017	0.002	0.000	0.994	0.004	Failed threshold	2	Not Classified
			0.003	0.173	0.518	0.002	0.002	0.000	0.980	0.003	Failed threshold	34	Not Classified

Figure 6.22: Fracture recognition performance for radiometric noise free input patches, but with displaced stroke pixels. [Θ_2=0.30], illustrating one reason why the Fracture Classifier performance is poor (with the three lower examples).

In the first case presented in Figure 6.22, a 7 x 7 patch rendering of fracture character 34 is input to the process. The patch undergoes weighted image reduction (a null transformation in this case) and contrast stretching (also a null transformation in this case). The resulting pixel values are input to the ANN and the ANN output values are listed, against the fracture character stroke number. ANN activations for stroke 2 and stroke 6 are high, as expected. The classifier correctly identifies this as character 32 with both post-processing methods.

In the second case, a stroke disturbance is introduced into stroke 6 by de-activating the second pixel in the stroke and activating a pixel immediately above it (off stroke 6 and onto stroke 7). If the activation state for stroke 2 is considered, its value (0.993) correctly identifies stroke 2 as being present in the input image patch. The activation state for stroke 6 (0.451) means that the output pattern fails Type 1 post-processing because its value is above the low hard limit threshold of Θ_2=0.30. Type 2 post-processing identifies fracture character 2 as the closest character to the ANN activations.

A similar outcome results in the third case presented to the ANN where a different pixel in the same stroke is 'displaced'. In the fourth case presented to the ANN, again, the output ANN activations fail Type 1 post-processing, but Type 2 post-processing identifies fracture character 34 (correctly) as the closest to the ANN activation state.

The conclusion made based on this and other retrospective analyses that the fracture character stroke positions within the ANN training data set were too 'clean'. When the ANN was trained, despite being given 7 x 7 image patches at input, training had the effect of sensitising the ANN weights mostly to the values located in the character stroke positions on the grid (see Figure 6.9).

6.6 Effect of deliberately displacing 'stroke pixels' in the ANN training data on Fracture Classifier performance

As fractures in real digital imagery are unlikely to be 'neatly presented', in order to overcome the problems identified, the ANN training data set was expanded. It included, not only the standard fracture alphabet characters, but also additional training exemplars based on them but with displaced 'stroke pixels'. Stroke pixel displacement is practically implemented by inverting the binary values of stroke pixels in two adjacent locations on the fracture character grid. Due to its similarity with the genetic algorithms process of mutation (Section 4.3.3), the process was called fracture character mutation and is illustrated in Figure 6.23.

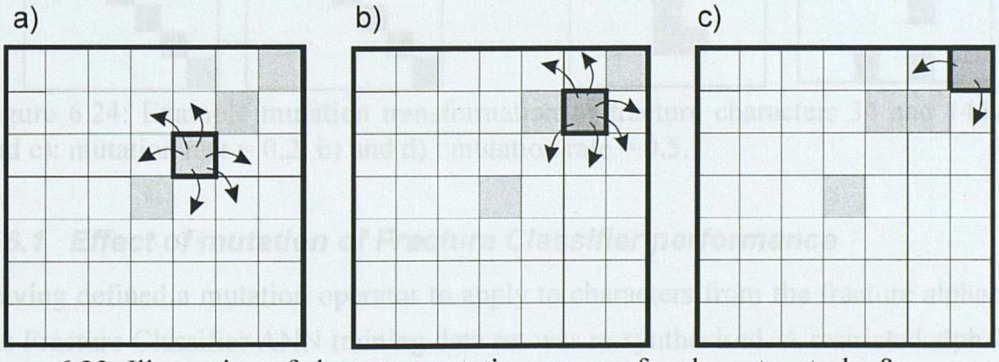


Figure 6.23: Illustration of character mutation process for character stroke 8.

Mutation here is effectively a local neighbour operator. Each activated site of each activated stroke of a fracture character is considered in turn. Figure 6.23a shows the first activated site of stroke 8 under consideration for mutation. Mutation takes the form of site de-activation, and simultaneous activation of an adjacent, currently non-activated site. Figure 6.23b shows that the first site of stroke 8 mutated with the site immediately above it. When the second site of the stroke is considered, this position becomes unavailable as a mutation 'partner', but site 2 can mutate to site 1's original location. Figure 6.23c shows that in fact site 2 did not mutate at all. This is a consequence of a computer-generated random number falling above a defined threshold mutation probability. Figure 6.23c also shows the available mutation options for the last site in the stroke. Examples of mutated characters 34 and 128 are presented in Figure 6.24, for values of the mutation rate = 0.2 and 0.5. A Visual Basic procedure listing for this fracture character mutation operator is given in Appendix C7.

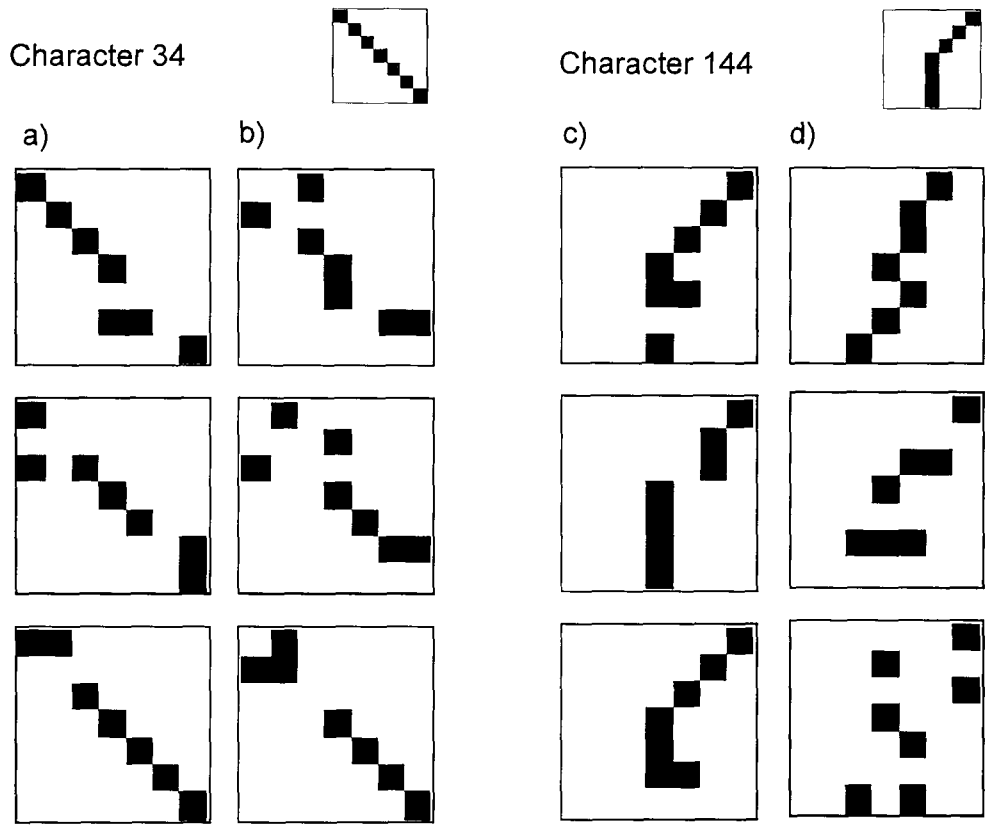


Figure 6.24: Example mutation transformation of fracture characters 34 and 144. a) and c): mutation rate = 0.2; b) and d) : mutation rate = 0.5.

6.6.1 Effect of mutation of Fracture Classifier performance

Having defined a mutation operator to apply to characters from the fracture alphabet, the Fracture Classifier ANN training data set was re-synthesised. A restricted alphabet was used. As characters with 5 or more activated strokes became total unrecognisable

under the mutation operator, these were omitted from the training data set. An alphabet of 163 characters resulted. The entire reduced alphabet was subjected to mutation 20 times with mutation rate = 0.4, the results being recorded to a training data file. The un-mutated, reduced, alphabet was written to this file once. Mutation rate = 0.4 leads to an average Hamming distance of approximately 2.44 between un-mutated and mutated characters. This means that on average 1.2 mutations occur per character (because one mutation leads to a Hamming distance of 2, as 2 bits are different)

ANN learning proceeded much more slowly than for the complete, un-mutated fracture alphabet. The learning rate for the back-propagation of error training algorithm was lowered to 0.00025 and the momentum parameter was increased to 0.95. RMS training error after 8000 training iterations was 0.002, with the maximum output error 0.999. The maximum output error was large, but on examination of the ANN output activations for each pattern, it was determined that this error applied for comparatively few training patterns. A listing of the Fracture Classifier's trained weights and thresholds is given in Appendix C8.

The performance of the Fracture Classifier improved in comparison to the version based on the non-mutated fracture alphabet. A few examples of its performance are presented in the following for illustrative purposes only. Firstly, performance with the examples of Figure 6.22 are presented again in Figure 6.25. Secondly, results of scanning the whole image with patch size 7 and 24 and gap widths of 1 and the patch size are presented in Figure 6.26, for comparison with Figure 6.21. Finally, 8 specially selected samples of Figure 6.2 and their fracture classifications are presented in Figures 6.27 and 6.28.

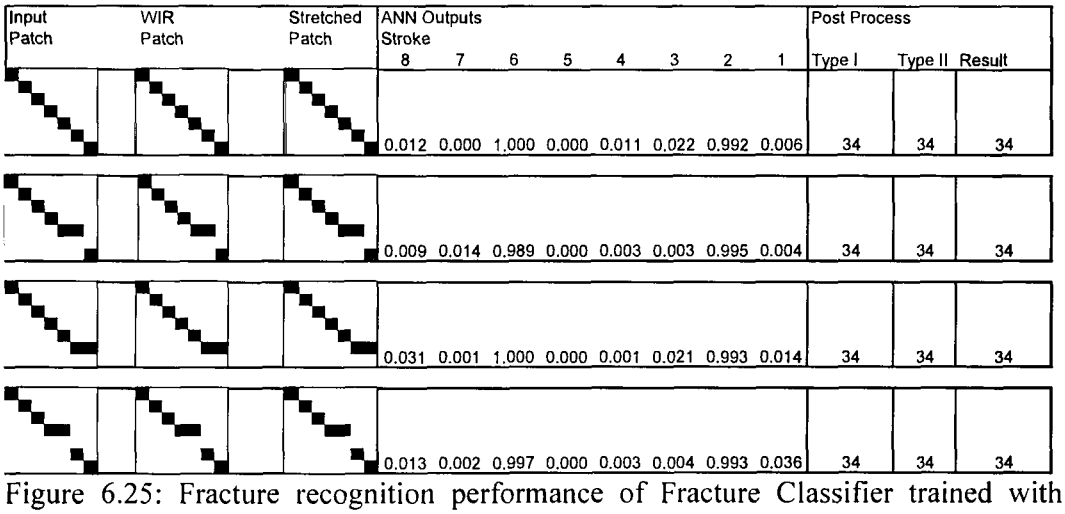


Figure 6.25: Fracture recognition performance of Fracture Classifier trained with mutated fracture character training patterns on the same test patterns as shown in Figure 6.22. [Θ_2=0.30.]. Improved performance compared with 6.22 should be noted.

Figure 6.25 shows that the effect of mutating the training examples had the desired effect of reducing the ANN's sensitivity to stroke positions in the presence of no radiometric noise. This confirms the earlier assertion that the Fracture Classifier needs to be robust against both radiometric and to 'spatial disturbance' of features. In essence, the Fracture Classifier has been trained on a more 'fuzzy' set than previously, the rationale being that consequently it should be more accurate in 'difficult' situations.

Figure 6.26 shows the results of applying the revised ANN filter to the whole image. With gap width set at 1 (overlapping samples), the results for patch sizes 7 and 24, both reveal a stronger signal than in the corresponding cases in Figure 6.21. The binary map produced with patch size = 24 in particular seems to have made classifications for all the main discontinuity features of Figure 6.2, with comparatively fewer recognitions between the main features. The results with zero overlap (gap-width = 7,24) are again disappointing, but it is evident that more features have been identified in comparison with Figure 6.21, for patch sizes of 7 and 24.

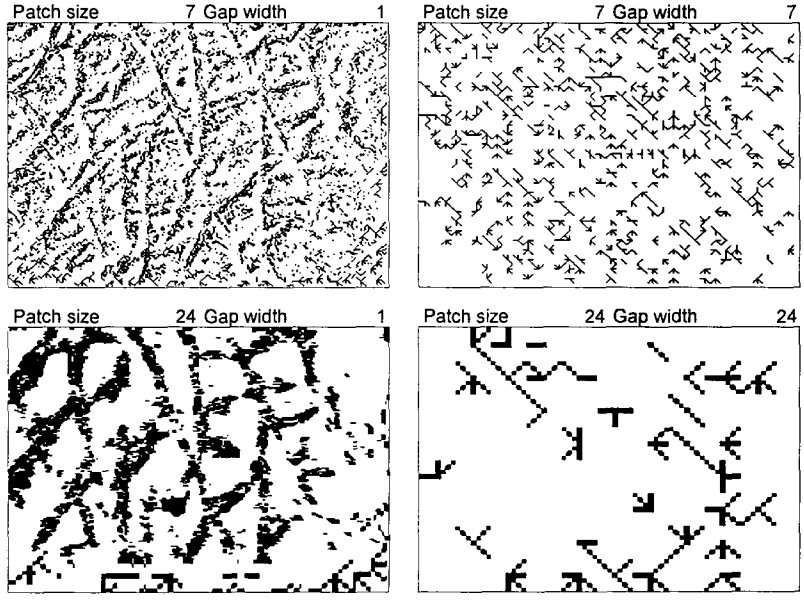


Figure 6.26: Results of application of Fracture Classifier trained with mutated fracture characters to Figure 6.2, for varying patch sizes and gap widths.

The 8 detailed examples of the fracture recognition process presented in Figures 6.27 and 6.28 provide insight into the reasons underlying the low level of performance of the revised classifier. The examples have been carefully selected from Figure 6.2 in order that cases, important from the rock mechanics point of view, can be highlighted. For each example, from the left to right, firstly the original image is presented. This is followed by the 7 x 7 weighted image reduced patch. Then the contrast stretched patch

is presented and finally the fracture character recognised is shown. If application of post-processing produces no classification, then Type 2 results are shown.

In Examples 1 and 2 both 'horizontal' and vertical fracture line segments respectively are correctly classified. This confirms the general view that the Fracture Classifier can recognise simple line segments reliably. Example 3 treats an example of a X intersection formed by two discontinuities. In this case, the Fracture Classifier actually failed to recognise the correct character because ANN output activations were not sufficiently high so that Type 1 post processing (applying thresholds to the activations) led to a result. The fracture imagery in the patch was correctly recognised under Type 2 post-processing. The suggestion is in this case that fracture classification failed due to the ANN training problems mentioned earlier, as the weighted image reduced and contrast stretched patches, clearly indicate that the good input to the ANN was available. It is useful to note that in the operational mode in scanning an image, this fracture recognition trial would result in a 'Not classified' response (because the post-processing stages disagree).

Example 4 shows a fracture line segment that runs diagonally across the input patch from the top left to the bottom right. The initial expectation may be that the Fracture Classifier would identify character 18 or perhaps character 17 of the fracture alphabet. The result of applying the Fracture Classifier to the input patch is a 'Not classified' response. The reason for this is that the ANN output activations failed Type I post-processing. Type 2 post processing identified character 57 in the patch.

This is not a counter-intuitive result, given the distribution of pixel values provided in the contrast-stretched patch. The values around the stroke 6 position are intense, and the input stroke broad such that it is conceivable that stroke 5 may also be activated. The original image has minor fracture content toward the lower left-hand-side. It is interesting that this information has clearly been preserved through the image reduction and contrast stretch process and that, as a result, stroke 4 is activated too. Consequently, a revised expectation may be that fracture characters 25 or 26 should be identified.

Whether the expectation is character 17 or 18, or 25 or 26, the result 'Not classified' is correct because Type 2 processing (which will always find a result) acting unchecked by Type 1 processing would have incorrectly classified the fracture content of the patch. This is why it is not appropriate to disable Type 1 post processing of the ANN output activations, because they provide a useful filter against false classification.

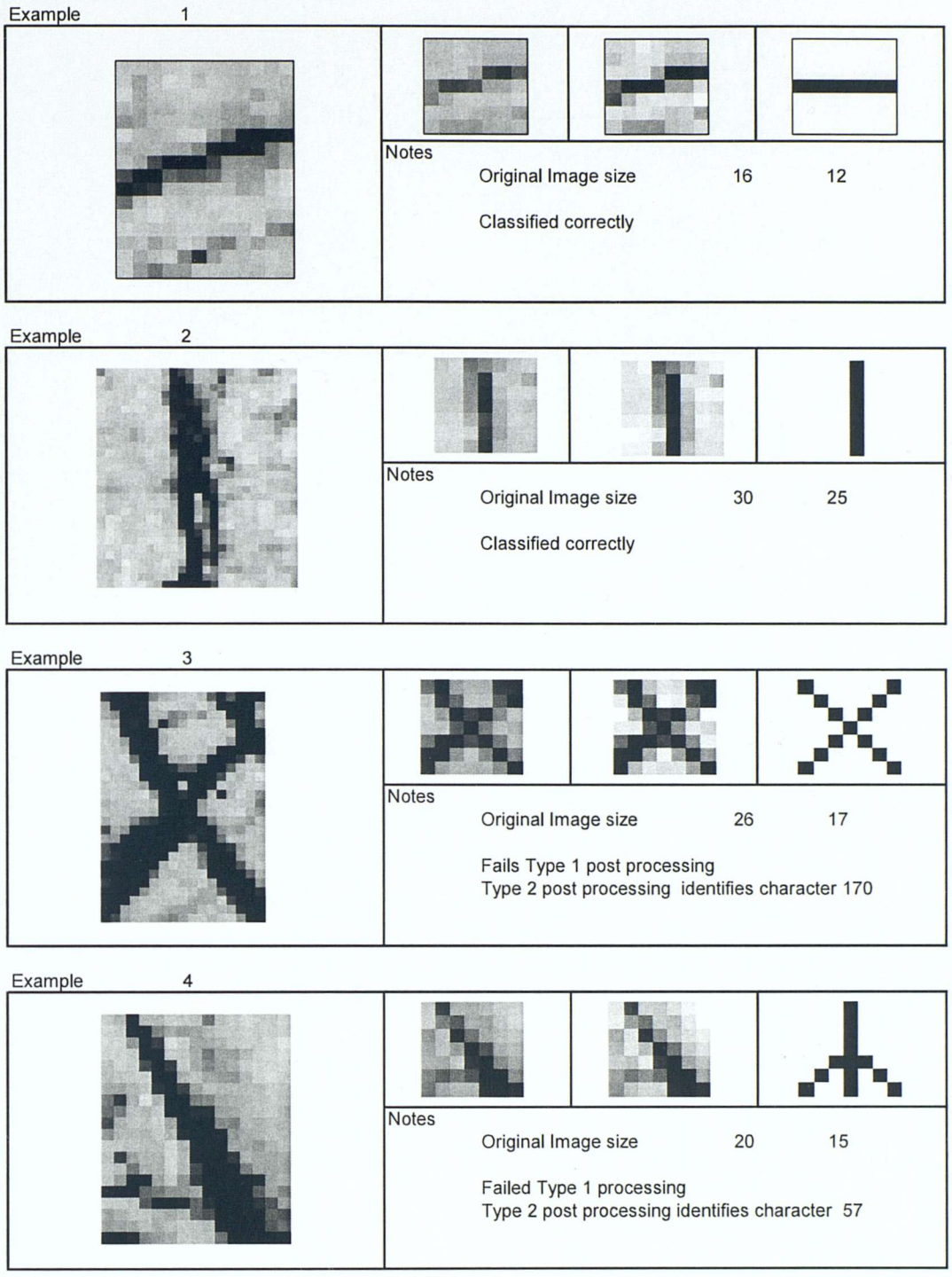


Figure 6.27: Examples of using the Fracture Classifier on selected samples from Figure 6.2. Example 1 : 'horizontal' line segment; Example 2 : 'vertical' line segment; Example 3 : X-intersection of two line segments; Example 4: 'diagonal' line segment.

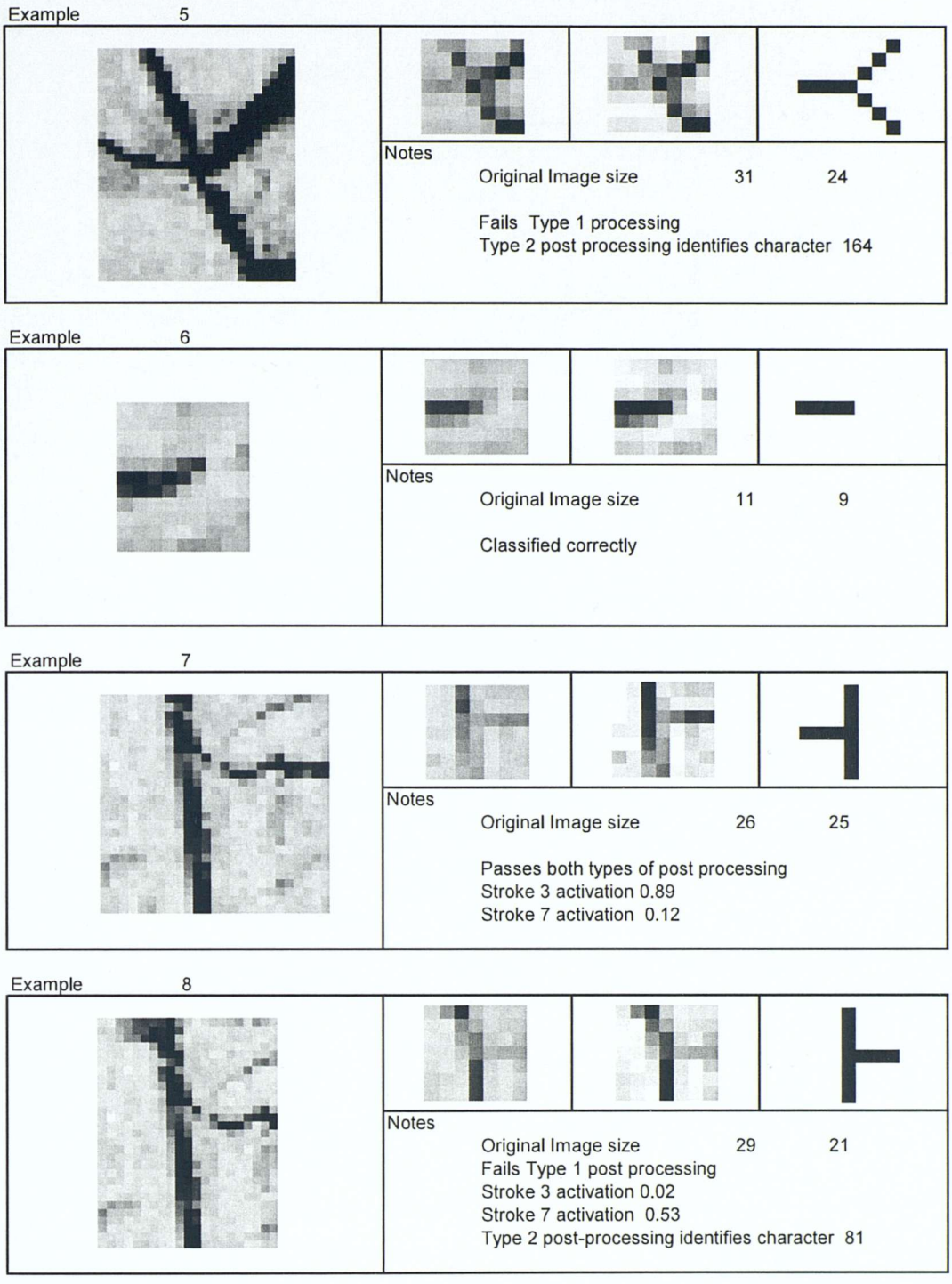


Figure 6.28: Further examples of using the Fracture Classifier on selected samples from Figure 6.2. Example 5 : 3 intersecting line segments; Example 6 : termination in intact rock; Example 7 : T-termination of a fracture against another fracture; Example 8: re-centred example 7.

A similar result is found in example 5. In this case, the Fracture Classifier finds 3 of the 4 strokes that would match expectations of a correctly classified result. The

result of the classification is again 'Not classified', a correct result given the ANN output activations checked using Type 1 post-processing.

The sample from Figure 6.2 used in example 6 shows a discontinuity evidently terminating within intact rock. The image content is correctly classified as being an example of fracture character 4. The post-processing checks do not always effectively filter false positive results. This is evident in example 7. Here, fracture character 21 has been positively identified whereas any reasonable expectation would be that character 81 should be recovered. One of the reasons found for the misclassification is that in the reduced and contrast-stretched imagery at least, the fracture image content is not centred in the patch. Re-centring the sample, as shown in example 8, results in a narrow failure to recover the correct character (Type 2 post-processing identifies the correct character). The performance of the Fracture Classifier in example 7 is a result of the presence of mutated fracture alphabet characters in the training data set, which allows the migration of radiometric content away from stroke locations. This is combined with the fact that, the despite the good root mean square error recovered in the ANN training process, the maximum output error was close to 1.

6.6.2 Options for fracture map assembly after post-processing

Application of the Fracture Classifier can be decomposed into 2 processes:

A) Initial scanning of the input image with the ANN and determination of fracture character numbers present (or not) in each patch scanned

B) Assembly of the results of the ANN scans to form the fracture map of the input image

The investigations reported in this section thus far have concentrated on the process A, that of determining the fracture character number. The best results of tests on real imagery occur when there is overlap between the individual patches scanned and each pixel of the image is considered in sequence (Figure 6.21, left column).

In this sub-section, the options for fracture map assembly are presented, and the optimal policy identified. The options are:

1) Single-pass direct replacement. This means direct replacement of image content with its identified fracture character equivalent, scanning the image left to right and top to bottom. This has been the option adopted thus far. With this option, binary fracture characters can be overwritten by direct replacements that follow in the assembly process, as fracture characters found to be centred on subsequent pixels are considered. The resulting fracture map is binary (i.e. comprises an array of 1s and 0s only).

2) Two-pass direct replacement. Same as option 1) but with a subsequent assembly pass over the whole image from right to left and bottom to top. Results from each pass are stored in intermediate buffers and then summed at the end of the second pass. This measure aims to compensate for the fact that fracture characters can be 'overwritten' with the single pass directly replacement process. The output image is no longer binary (i.e., comprising just 1s and 0s). The highest pixel values in the output fracture map are then 2 and the fracture map needs contrast stretching for visualisation.

3) Four-pass direct replacement. Same as option 1) but with three subsequent assembly passes over the whole image: i) left to right, bottom to top, ii) right to left, top to bottom, and; iii) right to left, bottom to top. This measure also aims to compensate for the fact that fracture characters can be 'overwritten' with the single pass process. The output image is no longer binary (i.e., comprising just 1s and 0s). The highest pixel values in the output fracture map are then 4 and the fracture map needs contrast stretching for visualisation.

4) Single-pass cumulative replacement. The fracture character numbers identified by the ANN filter are considered left to right and top to bottom. Where character strokes are active pixel values in the fracture map are incremented by one. Consideration of subsequent fracture characters can also lead to increments of the same fracture map pixels. The result is that stroke activations accumulate as overlapping fracture characters are considered. After the complete pass, pixels in the fracture map that are identified as active on multiple occasions will be the darkest when visualised.

5) Single-pass activation replacement. The input image is scanned left to right and top to bottom. Image patches are sampled and passed to the ANN filter which identifies the stroke activation values. Instead of applying Type 1 and Type 2 post processing to the ANN output, the stroke activation values are directly assigned to the fracture map pixel positions. The stroke activations from subsequent samples increment any already existing in the fracture map pixel positions. The resulting real valued array of cumulated stroke activation values is mapped to an integer valued array and subjected to contrast stretching for visualisation.

The results of applying these five options for fracture map assembly to a synthetic, high contrast, input image are presented in Figure 6.29.

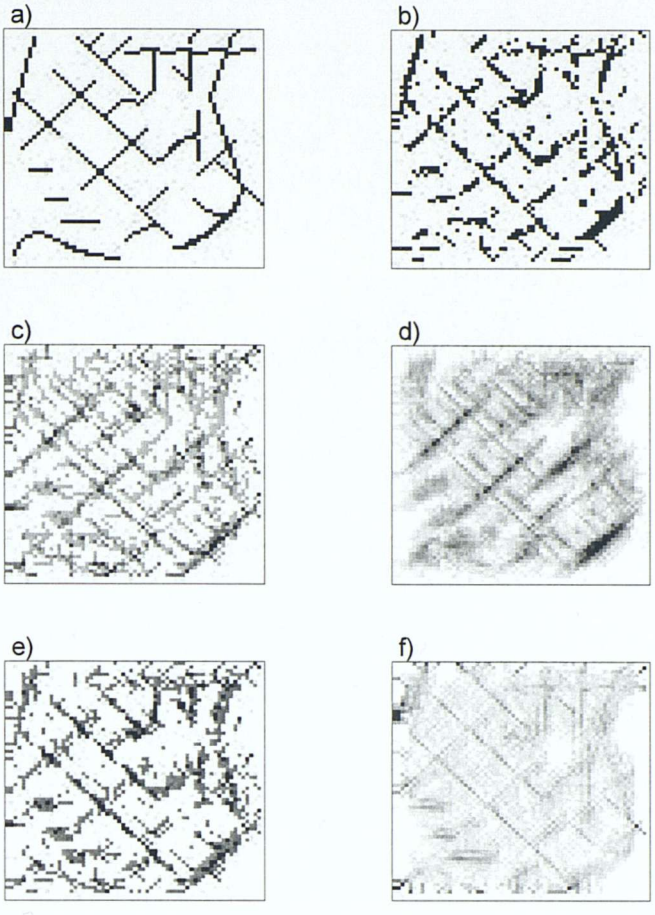


Figure 6.29: Results of application of Fracture Classifier utilising five options for fracture map assembly. a) Original image (Synthetic). b) Single-pass direct replacement (Option 1). c) Two-pass direct replacement (Option 2). d) Four-pass direct replacement (Option 3). e) Single-pass cumulative replacement (Option 4). f) Single-pass activation replacement (Option 5).

Through comparison of the fracture maps obtained for each of the five assembly options and the same input image, the simplest method, single-pass direct replacement, produces the best result. A synthetic image was used as input for this experiment to permit examination of the assembly options, free of concerns regarding the reliability of fracture character recognition from the ANN filter. Results Figure 6.29c and Figure 6.29e are credible but suggest that an appreciable computational effort will still be required for interpretation of these images to recover the Fracture Classifier's ideal 'result' Figure 6.29a. This is also true for results Figure 6.29d and Figure 6.29f, but interpretative effort is likely to be greater.

The result 6.29b provides an example of the quality of output that can be expected from the overall Fracture Classifier process alone when providing an ideal input image to the process. In general, result Figure 6.29b is good. Fractures have been identified where they have been present in the original image and many of the

defined geometric relations between individual fractures have been preserved. This provides strong support for the author's design concept, as detailed in Section 6.3.1 and as illustrated in Figure 6.8. However, detailed inspection of result Figure 6.29b reveals that some of the fracture termination and persistence characteristics have not been recovered perfectly indicating that, although the results presented thus far are promising, further work is required to convert the Fracture Classifier to a 'production' fracture recognition system.

6.6.3 Further work for the Fracture Classifier

The detailed examination of Fracture Classifier performance presented in this Section suggests that ANNs have a strong contribution to make in expectation guided approaches for computer aided recognition of discontinuities in digital imagery. The work conducted has not resulted in a production quality system but has demonstrated that there is merit in the approach and has also led to the identification of 'next steps'. The motivation for following these avenues is that many classification attempts result in robust performance and reasons for imperfect performance are understood. The avenues of further work worth exploring are:

1) Experimentation with a larger grid to define the fracture alphabet. Preliminary experiments have been conducted with a 9 x 9 grid template that indicates that diagonal strokes could be more easily discriminated than with a 7 x 7 grid.
2) Experimentation with 'circular' grids using an increased number of available strokes. This should increase the directional sensitivity but would also lead to higher computational load due to a requirement for image re-sampling via interpolation.
3) Development of a confidence metric associated with recovery of the fracture character number from Type 1 and Type 2 post-processing. One strong candidate is the Euclidean distance between the Type 2 character recovered and the stroke activation values. Such a metric could then be used to apply the single-pass direct replacement assembly process selectively, where confidence is highest.
4) Use of the Fracture Classifier with a supporting module such as the Fracture Discriminator. This would form an ANN with macrostructure. The data driven approach of the Fracture Discriminator focuses the expectation guided approach of the Fracture Classifier on those areas where there is high probability of fracture presence.

6.7 Conclusions

In this Chapter, two complementary methods for analysing the discontinuity geometry rock mass imagery have been presented.

The first, which has been called the Fracture Discriminator, represents a robust approach that should work with most rock mass imagery to provide a basis for the creation of a fracture trace map. This method works by sampling the image(s) to be processed to create a data base of examples on which an ANN is trained. After training, the ANN scans the complete image(s) to identify the locations within the imagery where there is a high probability of occurrence of a fracture. The resulting fracture probability maps can be passed to raster to vector conversion programs to produce discontinuity trace maps. Alternatively, the raster output of the Fracture Discriminator can be usefully operated upon directly. This was illustrated by the development of a novel procedure for the estimation of fracture intensity parameter $P_{2,2}$ using the Fracture Discriminator output.

The second method presented was called the Fracture Classifier. This procedure has the objective of producing a binary raster trace map of fractures in imagery that aims to preserve the geometric relations between the various fractures in the imagery, in particular fracture intersections and terminations. The approach does not rely on samples from the imagery, but it trains an ANN to learn the patterns in a finite synthetic alphabet of possible fracture geometries.

In testing the ANN, after training, it was found that fracture recognition was extremely robust in the presence of appreciable radiometric noise. Use of the classifiers on real imagery, revealed a reduction of recognition performance due to positioning irregularities in the character strokes of characters in the fracture alphabet compared to the actual 'strokes' of fractures in the input image. The author partially overcame this problem by introducing a neighbourhood mutation operator for fracture characters and training another ANN on the mutated data set. The performance of this ANN module on real rock mass imagery was an improvement on the non-mutated version. The Fracture Classifier uses a single-pass direct replacement (with overwriting) method of assembling the fracture trace map; this was demonstrated to be the best method of fracture map assembly in the previous section.

Applying the Fracture Classifier to complete images with the largest possible overlap between sampled patches produced binary maps similar in form to that from the Fracture Discriminator. This confirms that the Fracture Classifier can also discriminate between portions of imagery that contain fracture content. The Fracture Discriminator is data driven, that is, it uses a bottom up approach; whereas, the Fracture Classifier is expectation guided, that is, it uses a top down approach. In the context presented here, the data-driven approach worked best, but ultimately the using

the two methods in a complementary fashion should lead to an improved overall performance

The material on the Fracture Classifier highlighted the attention that has to given to pre-processing of digital imagery of fractured rock masses in order that subsequent image processing filters, ANN-based or otherwise, stand a reasonable chance of interpreting the features in the imagery.

The material on the Fracture Discriminator in this Chapter focussed on procedures followed to make an assessment of ANN performance when their capability to handle noisy or incomplete input data is exploited. The adoption of a visual application example greatly facilitated this process.

Undoubtedly, in the near future ANN methods such as those presented here or as reported Hadjigeorgiou *et al.* (2003), or Lemy and Hadjigeorgiou (2003) will result in the production of useful utilities for rock engineers. However, a capability to allow fracture trace maps to be created in an automated and robust fashion has been pursued vigorously over the past 15 years. The state-of-the-art is probably best represented by CSIRO's SIROvision research product that has only recently become commercially available software (see for example, Australian Mining, 2007). The photogrammetric and terrain analysis procedures used within this software still require significant user intervention to produce reliable fracture trace maps, but this situation would undoubtedly be improved through data fusion with output from other specialist algorithms such as the Fracture Discriminator detailed herein. However, in development of the Fracture Discriminator and the Fracture Classifier, the author has formed the view that it is doubtful 100% reliability of trace maps produced in a fully automated manner is achievable, but also that the best performance will ultimately be found from procedures that invoke algorithms that adapt to the nature of the image content, and automate the development of new algorithms when those already tried have failed.

Anyone who has used the best available optical character recognition software will understand that even with this easier task, the output of the automated routines needs to be carefully checked. Having said this, it would be unusual for the output of such programs to be considered totally unintelligible. It is the author's belief that ultimately, the same situation will arise in the automated abstraction of fracture features. It must be recognised that even the best available fracture trace mapping software will always be inadequate in terms of a 100% correct reconstruction of rock mass geometry. However, despite these deficiencies, it is unlikely that the output will be considered unintelligible or worthless, especially if the results are to be carried forward to discontinuity analysis methods with a statistical basis.

7 Contributions, Discussion and Conclusions

In this concluding Chapter, firstly the original contributions of the work presented in this thesis are listed to make them clear and readily identifiable. In the Discussions Section, firstly, rock engineering systems are re-visited so that the roles that parallel distributed processing methods have within them can be highlighted. Then conclusions relating to specific applications developed across the chapters are put into the Rock Engineering Systems context. Then statements regarding whether all the criteria defined at the outset of the thesis have been met are presented. The Chapter, and the thesis then closes with the Conclusions section which reiterates the key results and makes a few general observations.

7.1 Original Contributions

The original contributions made by this author and reported in this thesis are as follows.

1. The author proved that ANNs can learn and reproduce mechanical behaviour outside that the ANN experienced during the training process. (Chapter 4)
2. An original technique was developed and proven by the author for embedding constitutive behaviour of any material using an ANN into a finite difference numerical modelling code for use in rock mechanics. (Chapter 5)
3. The author identified that, as constitutive models in finite difference codes, ANN formulations that predict increments to stress components are superior to those which estimate the stress component values directly. (Chapter 5)
4. The author identified that the transducers used in triaxial testing of rock need to have better sensitivity than at present, if the observations are to be used to train an ANN that predicts stress component increments from strain component increments. Alternatively, the time between observations logged from the equipment needs to follow a distribution, rather than be fixed throughout a test. (Chapter 5)
5. The author developed a method that trains an ANN to produce a robust measure of the likelihood of fracture presence in a digital image and used this to develop a novel procedure for the estimation of the fracture intensity parameter $P_{2,2}$. (Chapter 6)
6. The author has proposed the concept of a 'Fracture Alphabet' as a set of templates for fracture termination conditions and has developed an ANN that can recognise Fracture Alphabet 'characters' in digital imagery of rock mass exposures. (Chapter 6)
7. The author developed a procedure for the downward re-scaling of digital imagery that better preserves the relative position of image content than

conventional resizing or resampling techniques used in image processing. (Chapter 6 and Appendix C5)

7.2 Discussion

7.2.1 The Rock Engineering System

Figure 1.2 of Chapter 1 presented a schematic rock engineering system where the engineering component of the system formed a feedback loop with a rock mass system. Figure 7.1 upgrades Figure 1.2 in two main ways. Firstly, the engineering box is retitled the 'concept and design' subsystem, and expanded so that the processes comprising the engineering task and their inter-relations can be identified. Secondly, the nature of the input and output channels forming the link with rock mass system is detailed.

The input mapping to engineering is titled 'Rock Mass Monitoring'. The rock mass monitoring box represents the passage of information gathered from, about, or within, the rock mass to the rock engineering team. The meaning of this channel can be abstract: for example, it could represent a field geologist's notebook containing mapping and lithology descriptions; or it could also represent the results of a factual report of a geotechnical site investigation. Alternatively, where there is extensive monitoring installed within a rock mass, this mapping could represent the flow of digital information down the wires to a logging system.

The channel from the concept and design sub-system that feeds back to the rock mass system is called 'Excavation / Installation'. The meaning of this channel can be abstract but it is meant to represent the implementation of physical change to the rock mass system, where the changes should arise as a result of engineering policy. The nature of the physical changes sent down this channel can be major, for example a full specification of excavation works, or a minor one. Examples of the latter are: a schedule of exploratory holes to be drilled, specification of geophysical investigations, a sampling schedule, and field scale *in situ* testing.

The box titled 'Engineering Objective Definition' represents the process of defining what needs to be achieved from the engineering works, in the 'language' with which judgements on the performance of the rock mass will be made. Defining the engineering objective is a process that must be completed early within a rock engineering project, and frequently, it is this module which initiates the various mechanisms that arise between the other rock engineering units.

The engineering objective can vary from one part of a rock engineering project to another and can be hierarchically defined. For example, within the periphery of a proposed underground excavation, the engineering objective is to cause the rock to become overstressed such that it breaks; at locations remote from the periphery, the

engineering objective is that the rock mass be relatively undisturbed. On the periphery, the engineering objective is to cause the rock to become more stressed than it was before so that it helps to support the opening, but the rock should not become so highly stressed that it fractures.

'Rock mass characterisation' is the module within the rock engineering system that represents the process of *interpretation* of information being gathered about the rock mass system, and the proposed engineering measures. The function of the 'Rock Mass Characterisation' module is to update the 'Rock Mass Model'. Rock mass characterisation can mean the process of observation taking; however, here the meaning is strict —it represents the interpretative task. Rock mass characterisation associates observations of the rock mass system with observations of how it is perturbed, either naturally or via engineering. By detection of differences in rock mass system behaviour, specified or predicted, with that measured or interpreted, deficiencies in the rock mass model or engineering specification variances can be identified, so that they can be eliminated.

Within Figure 7.1, the 'Rock Mass Model' is not considered a process but is considered an object. This is why it is shown with a slightly different kind of box, in comparison to, for example, the 'Rock Mass Characterisation' module. Conceptually, at its simplest, this can be considered a list of properties and characteristics. In the sense represented in Figure 7.1, the Rock Mass Model does not include rock mass modelling, as would be conducted via numerical modelling; it is conceived more as a prescription of character.

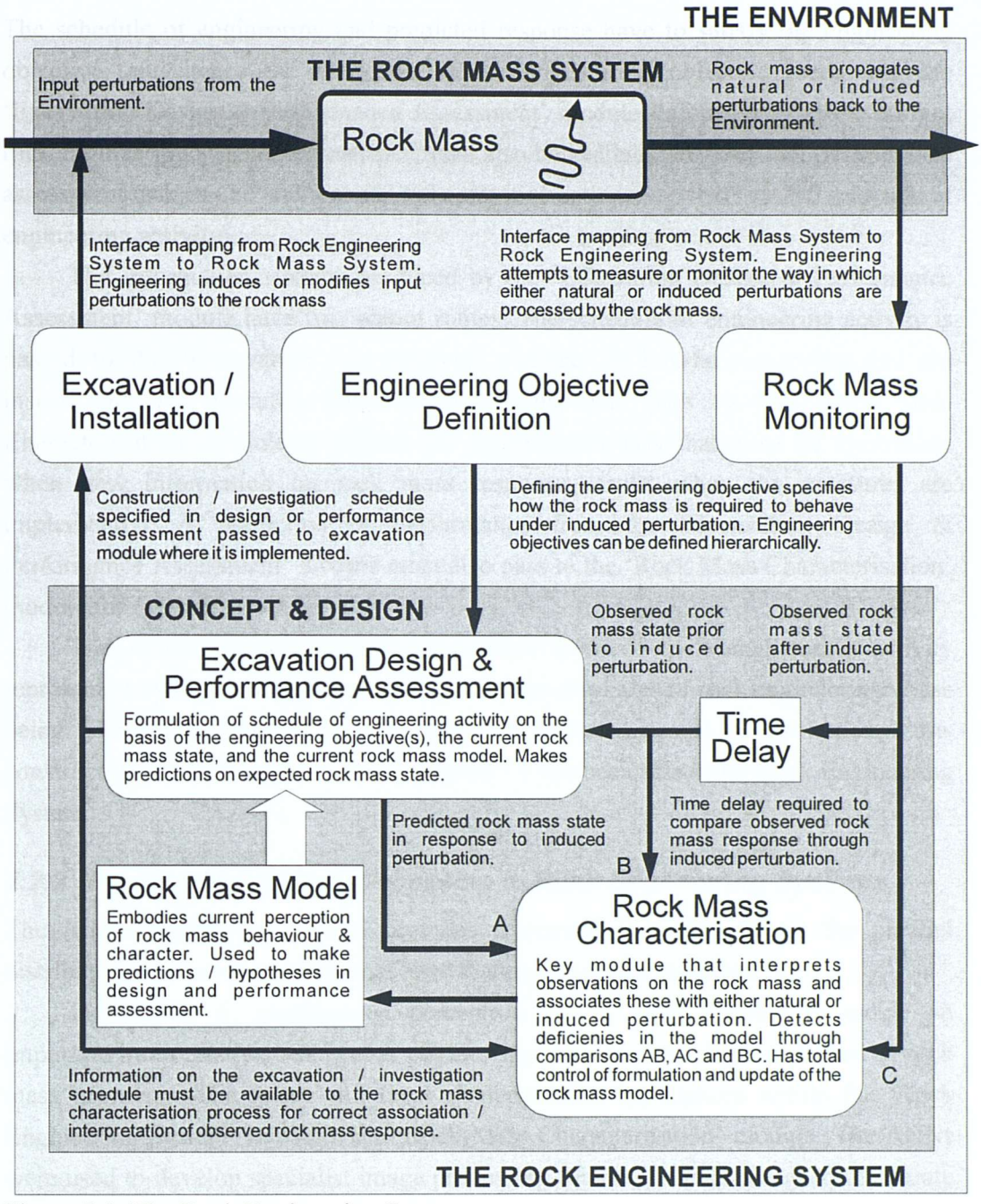


Figure 7.1: The Rock Engineering System

The 'Excavation Design & Performance Assessment' module of Figure 7.1 is where the rock mass model is used to conduct analyses, using any possible available method (static, dynamic, numerical, closed form, etc) to provide two new streams of information:

1) A prediction of likely response of the rock mass to perturbation, either naturally caused, or as a result of implementation of a schedule of engineering activity.
2) A schedule of engineering activity (that is, the engineering plans).

The schedule of engineering and predicted response have to satisfy the engineering objective, and hence the definition of the engineering objective feeds into the 'Excavation Design & Performance Assessment' module. Information on the current rock engineering system performance must also be available in order that performance assessment task can be undertaken, and corrective action specified (via the schedule of engineering activity).

The information streams produced by the 'Excavation Design & Performance Assessment' module have two output routes. The schedule of engineering activity is passed to the 'Excavation / Installation' engineering interface mapping and the instructions are executed. This schedule must also pass to the 'Rock Mass Characterisation' module to support the interpretative task that must be undertaken when new information on rock mass response results when the measures are implemented. A prediction of performance from the 'Excavation Design & Performance Assessment' module must also pass to the 'Rock Mass Characterisation' module for the same reason.

The model of process and information flow of rock engineering activity represented by Figure 7.1 is broadly applicable to all phases of rock engineering, these being: Conceptual studies, Site investigation, Detailed design, Construction, Post-construction monitoring. This is why Figure 7.1 has been titled The Rock Engineering System.

7.2.2 Parallel Distributed Processing in Rock Engineering Systems.

The Rock Engineering System provides a context in which to set the parallel distributed processing methods that have featured in earlier chapters.

In Chapter 6, multilayered perceptron ANNs were adopted to tackle an important interpretative task in rock engineering. This was the characterisation of rock mass geometry. The 'place' for these demonstration applications within the 'Rock Engineering System' is within the 'Rock Mass Characterisation' module. The ANNs were used to develop specialist image processing filters, specially designed to operate upon imagery of rock mass exposures. The output of the ANNs were fracture trace maps which could be carried forward to condition the synthesis of discrete fracture network rock mass models. In their application to the fracture identification task, only a moderate degree of success was attained. However, avenues for further development have been identified. Thus, while it cannot be said that these applications of parallel distributed processing methods definitely have a role to play in the 'Rock Mass Characterisation' part of the 'Rock Engineering System', it can be said that they have great potential.

In Chapter 5, a multilayered perceptron ANN was used to represent the total constitutive behaviour of a sandstone rock in a manner suitable for direct use within a finite difference numerical method. The ANN is trained on data from laboratory triaxial testing of the rock. In this application, the ANN effectively replaces the traditional interpretative task of establishing numerical values for rock properties that are used in constitutive models that are stylised approximations of true rock behaviour. The ANN encapsulates all test data on the rock's constitutive behaviour in a compact and modular form, which can be used in a numerical modelling code. In this instance, the process lies within the 'Rock Mass Characterisation' component of the 'Rock Engineering System'. The result (the trained ANN) lies within the 'Rock Mass Model' object of the 'Rock Engineering System'. The result is used in 'Excavation Design & Performance Assessment'.

The main function of Chapter 4 was to prepare the reader for Chapter 5 where training of an ANN with real rather than simulated triaxial test data was undertaken. However, Chapter 4 also showed that the ANN method explained could be used to learn *specified* behaviour from synthetic data. The author considers that this demonstrates an attractive potential application area of ANNs where they are used to summarise the results of a series of runs of a numerical model that is of necessarily small scale due to complex geometry or constitutive behaviour. The trained ANN is then used within the numerical code again, in recall mode only, to allow the modelling of the same situation at a much larger, engineering, scale. Again, it is evident that the 'place' for this application in the 'Rock Engineering System' is the channel between the 'Rock Mass Characterisation' module and the 'Rock Mass Model'.

Within this Section, the demonstrations of parallel distributed processing developed across the chapters of this report have been put into the context of the 'Rock Engineering System'. One of the key observations from the review of the ANN literature presented in Chapter 3 was that, already, there are few areas of the disciplines of rock mechanics or rock engineering that have not been 'touched' by ANNs. In Chapter 3 ANNs were shown to have made contributions in the following areas which are set against the corresponding processes in the 'Rock Engineering System'.

Area of Rock Engineering	Component of Rock Engineering System
Constitutive Modelling	Rock Mass Model
Numerical Modelling	Exc. Design & Perf. Ass.
Soil / Rock Characterisation	Rock Mass Characterisation
Geological Identification	Rock Mass Characterisation
Underground Excavations	Exc. Design & Perf. Ass.
Foundations / Slopes	Exc. Design & Perf. Ass.
Plant control & performance	Excavation / Installation
Blasting / Seismics	Exc. / Install. & Rock Mass Mntrg.
Soil Hydrogeology	Rock Mass Characterisation

Perceived benefits of using ANN methodologies in these areas included: new capabilities, automation, better/alternative uses of data, and better / alternative models or methods.

A significant portion of the design process is a search for 'adequate optimality' subject to many and varied constraints. Increasingly, the rock mass characterisation process also involves a search for optimality in a back analysis context. With these two observations in mind, it is perhaps unsurprising that the search and optimisation capabilities of GAs are now being exploited in roles that span individual components of the Rock Engineering System. Such applications integrate the benefits of both conventional and alternative methods such as ANNs. While parallel distributed processing methods are not as fundamental, or important, innovations as, for example, the emergence of numerical modelling, they have auxiliary roles that introduce enhanced capability that cannot be ignored.

7.2.3 Review of Objectives: Rock Constitutive Behaviour

In Section 1.4.1, the criteria for the development of any ANN method developed to support constitutive behaviour modelling of rock were as follows.

i) The artificial neural network should encapsulate the total rock behaviour and that this should be accomplishable in a manner accessible to practitioners.

ii) The artificial neural network should also be able to be embedded in the numerical modelling codes in a straightforward way.

iii) The data source to calibrate the artificial neural network should be compatible with already well established methods.

The conclusion is that all these criteria have been met, as demonstrated by the material set out in Chapters 4 and 5. Specifically,

i) The ANN has been shown to be able to encapsulate the total rock behaviour. The training methods have been explained in detail, and have been presented so that they can be repeated or checked (Chapters 4 & 5, Appendices A & B).

ii) Chapter 5 provided the details of how to embed the ANN within the *FLAC* finite difference numerical method as an example. This together with the automated procedures listed in Appendix B6 means that this task can be completed with comparatively little effort.

iii) Chapter 5 adopted standard triaxial test data as the data source.

7.2.4 Review of Objectives: Rock Mass Geometry Modelling

In Section 1.4.2, the criteria for any ANN based method designed to support the task of characterisation of rock mass geometry were as follows.

i) Site presence for data gathering needs to reduced to one or two hours rather than days.

ii) Personnel involved in the data gathering task need to work from a safe location

iii) The interpreted result should be of a quality at least equal to that obtained from conventional means and obtained in a time at most equal to that expended using conventional techniques.

The conclusion is that at least two of three of these criteria have been met in the methods outlined in Chapter 6. Specifically,

i) An advantage of any remote method of rock geometry characterisation is that the time required on site can be spent ensuring lighting arrangements are suitable, and that suitable locations for image capture are selected. These are far less time-consuming and less dangerous tasks than conducting scanline observations.

ii) See i) above.

iii) Vector fracture trace maps were presented in Chapter 6, but any assessment of their quality can only be subjective at this stage. Objective assessments of any newly developed technology could be made with a standard set of test imagery. Some of these could be synthetic, produced by realistic renderings of discrete fracture models.

The majority of objectives set out in Chapter 1 have been attained.

7.3 Conclusions

1) Parallel distributed processing methods have been found to be effective in many roles within rock engineering. The Chapters presented in this thesis have provided supporting examples. However, it has not been proved, with conventional rigour,

here or anywhere, that ANN methods are 'better'. ANNs are empirical approaches to problems that may work when more conventional approaches do not. Where conventional approaches may themselves be computationally intensive, ANNs may be a better alternative, if the ANN training process is found to be straightforward.

2) In the experience of this author, ANNs correlation mappings can be found for any sets of variables, properties or parameters that are, in fact, known to be somehow dependent on one another. Put another way, ANNs 'work' for most sensibly compiled data sets.
3) When such correlative studies are reported in the literature, it is crucial that full details of ANN activation functions, weights and network structure are provided. Any pre- or post- processing procedures should also be fully reported. Unless there is any distinctive methodological development in the application of the ANN, failure to report these things simply demonstrates that ANNs 'worked'. There is a clear need for a standard in reporting work involving ANNs that is compatible with other literature reporting empirical relations.
4) Multilayered perceptrons have been trained with triaxial compression test data and have been shown to work as user-defined constitutive models within *FLAC*.
5) A data driven ANN image processing filter was developed that successfully produced image maps of the probability of discontinuity occurrence within digital imagery of rock faces. Output of this filter can be sent to conventional raster to vector conversion utilities to produce fracture trace maps that are reasonably faithful to fractures present in the imagery when inspected manually. Alternatively, the output can be used to produce a measure of fracture intensity.
6) An expectation –guided ANN image processing filter was developed with the specific intention that it be able to preserve connectivity conditions of discontinuities present in digital imagery. This filter needs further work before it can be considered robust enough for routine use.

There should be a concerted and open effort throughout the rock mechanics and rock engineering scientific community to revisit key data sets upon which many of the most important empirical tools have been based. As conventional approaches to establishing empirical relations have been shown to be inferior to ANNs on many occasions, these should be released so that the advantages that parallel distributed processing methods have can be exploited.

ANNs have a reputation in some quarters simply as 'black box' approaches, that cannot be trusted. In the view of this author, this reputation may be somewhat deserved, the main reason being the unfortunate inadequacy in reporting standards in

the literature. However, they should not be simply dismissed as non-viable options because of this reputation. On occasion, an ANN may be the only option in the absence of an available model to fit a data set. Also, despite their 'blackness', ANNs can be engineered, but they are engineered through judicious selection and preparation of the data sets on which they are trained, not by direct manipulation of ANN weights. Techniques are also available to 'data mine' ANNs to produce relations and curves that can be interpreted conventionally. Supervised learning by genetic algorithms offers the same benefits of robust fitting as ANNs but in a totally transparent manner, when a model is available. The work presented in this thesis has also demonstrated that using an ANN can lead to novel capabilities over a broad range of problem areas important in rock mechanics and rock engineering.

8 References

Aldrich, C. and van Deventer J.S.J., 1993. New computational methods for the application of connectionist networks to mineral processing. *Proc. XXIV APCOM, 31 October - 3 November, Montreal, Canada, p 211-218.*

Aleksander, I., and Morton, H., 1990. An Introduction to Neural Computing. Chapman & Hall, London.

Almenara Chau, J.R., (1992). Investigation of the cutting process in sandstones with blunt PDC cutters. PhD Thesis, Imperial College, London.

Alvarez-Grima, M., Bruines, P.A., Verhof, P.N.W., 2000. Modelling tunnel boring machine performance by neuro-fuzzy methods. *Tunnelling and Underground Space Tech., Vol.15, No. 3, pp. 259-269.*

Amorosi, A, Rampello, S., and Millar, D.L., 1996. On the use of artificial neural networks as generic descriptors of geomaterial mechanical behaviour. *Proc ISRM Int. Symp. On Prediction and Performanvce in Rock Mechanics and Rock Engineering (EUROCK 96), Ch.178, pp.161-168, 2-5 Sept 1996, Turin, Italy.*

Anderson, J.A., and Rosenfeld, E., eds. (1988), Neurocomputing: Foundatons of Research, Cambridge, MA: The MIT Press

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, Vol. 84, pp.-413-451.* Reprinted in Anderson and Rosenfeld (1988).

An, P., Chung, C.F., and Rencz, A.N., 1995. Digitial lithology mapping from airborne geophysical and remote sensing data in the Melville Peninsula, Northern Canada, using a neural network approach. *Remote Sensing of the Environment, Vol.53, No.2, pp.76-84.*

Aoshima, S., Takeda, K., and Yabuta, T., 1993. Autotuning of feedback gains using a neural network for a small tunnelling robot. *JSME Int. J. Series C-Dynamics, Control, Robotics Design and Manufacturing. Vol.36, No.4, pp.435-441.*

Arnold, P.N., 1993. The Development of a Rock Engineering Systems Methodology. Ph.D. Thesis, Imperial College, University of London.

Attohokine, N.O., and Fekpe, E.S.K.,1996. Strength characteristics modeling of lateritic soils using adaptive neural networks. *Construction & Building Materials, Vol.10, No.8, pp.577-582.*

Australian Mining, 2007. CSIRO mapping the future. News article available online at: http://www.miningaustralia.com.au/articles/CSIRO-mapping-the-future_z54797.htm. Accessed 18:10, 21 Sep 2008.

Azimisadjadi, M.R., Sheedvash, S., and Trujillo, F.O., 1993. Recursive dynamic node creation in multilayer neural networks. *IEEE Trans. Neural Networks, Vol.4, No.2, pp.242-256.*

Chapter 11: References

Barron, L., Smith, M.L., Prosbrey, K., 1994. Neural network pattern recognition of blast fragment size distributions. *Particulate Sci. and Tech. Vol.12, No.3, pp.235-242.*

Barton, N.R., Lien, R. and Lunde, J. 1974. Engineering classification of rock masses for the design of tunnel support. *Rock Mech.Vol. 6, No. 4, pp. 189-239.*

Barton, N., Løset, F., Lien, R. and Lunde, J. 1980. Application of the Q-system in design decisions. In *Subsurface space,* (ed. M. Bergman) *Vol. 2, pp. 553-561. New York: Pergamon.*

Basheer, I.A., and Najjar, Y.M., 1995. A neural network for soil compaction. *Proc. 5th Int. Symp. Num. Models in Geomech. Ch.109, pp.435-440. 6-8 Sept 1995, Davos, Switzerland.*

Berry, P., and Reid, D., 1987. An introduction to soil mechanics. McGraw-Hill, London.

Bizjak, K.F., and Petkovšek, B., 2004. Displacement analysis of tunnel support in soft rock around a shallow highway tunnel at Golovec. *Engineering Geology Vol. 75, pp 89–106.*

Bonifazi, G., 1993. Mineral particle characterisation by use of neural network strategies. *Preprints Conf. Artificial Intelligence in the Minerals Sector, 20 April, Nottingham, UK, 12 p.*

Bottou, L., and Vapnik, V., 1992. Local learning algorithms. *Neural Computation, Vol.4, No.6, pp.888-900.*

Brady, B H G, and Brown, E T, 1980. Rock Mechanics for Underground Mining, Chapman and Hall.

Brown, E.T., 1981. Rock Characterisation, Testing and Monitoring, ISRM Suggested Methods. Pergamon Press.

Cabello, E., Sanchez, M.A., and Delgado, J., 2002. A New Approach to Identify Big Rocks with Applications to the Mining Industry. *Real-Time Imaging. Vol 8, pp 1-9.*

Cai, J., Zhao, J., Hudson, J.A., and Wu, X., 1996. Using neural networks in rock engineering systems for cavern performance auditing. *Proc ISRM Int. Symp. On Prediction and Performanvce in Rock Mechanics and Rock Engineering (EUROCK 96), Ch.178, pp.965-972, 2-5 Sept 1996, Turin, Italy.*

Cai, Y.D., 1995. The application of artificial neural network in determining the blasting classification of rocks. *Proc. 2nd Int. Conf. Eng. Blasting Technique, Ch.90, pp.24-27.* 7 - 10 Nov 1995, Kunming, People's Republic of China.

Caiti, A. and Parisini T., 1991. Mapping of ocean sediments by networks of parallel interpolating units. *Proc. IEEE Conf. Neural Networks for Ocean Engineering, 15-17 August, Washington, DC, p 231-238.*

Cal, Y., 1995. Soil Classification by neural-network. *Advances in Eng. Software, Vol.22, No.2, pp.95-97.*

Chapter 11: References

Calderbank, P.A., 1994. *A Study of Rock Deformability Behaviour using Artificial Neural Networks.* MSc Thesis. Imperial College, Royal School of Mines, London, UK.

Cardon, H., van Hoogstraten R. and Davies P., 1991. A neural network application in geology: Identification of genetic facies. *Proc. Int. Conf. Artificial Neural Networks, 24-28 June, Espoo, Finland, p 809-813.*

Carpenter, G.A., Grossberg, S. 1987, ART 2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics, Vol. 26, pp. 4919-4930.*

Carr, J.R., and Hibbard, M.J., 1991. Open-ended mineralogical textural rock classification. *Computers & Geosciences, Vol.17, No.10, pp.1409-1463.*

Castleman K.R., 1979 Digital Image Processing. Prentice Hall.

Chandler, J.H. and Moore, R., 1989. Analytical photogrammetry: a method for monitoring slope instability. *Quarterly Journal of Engineering Geology, Vol. 22, pp 97-110.*

Chang, H.C., Chen, H.C., Fang, J.H., 1997. Fuzzy associative memory neural network. *IEEE Trans. Geoscience & Remote Sensing, Vol.35, No.3, pp.773-780.*

Chang, H.C., Kopaska-Merkel, D.C., Chen, H.C., Durrans, S.R., 2000. Lithofacies identifcation using multiple adaptive resonance theory neural networks and group decision expert system. *Computers & Geosci., Vol 26, pp.591-601.*

Chang, H.C., Kopaska-Merkel, D.C., and Chen, H.C., 2002. Lithofacies identifcation using Kohonen self-organising maps. *Computers & Geosciences., Vol 28, pp.223-229.*

Chow, Y.K., Chan, W.T., Liu, L.F., and Lee, S.L., 1995. Prediction of pile capacity from stress-wave measurments - A neural network approach. *Int. J. Num. & Analytical Methods Geomech. Vol.19, No.2, pp.107-126.*

Chunduru, R.K., Sen, R.K., Stoffa, P.L., Nagendra, R., 1995. Nonlinear inversion of resistivity profiling data for some regular geometrical bodies. *Geophysical Prospecting, Vol.43, No.8, pp.979-1003.*

Cizb, R. and Rudajev, V., 2007. Linear and nonlinear attributes of ultrasonic time series recorded from experimentally loaded rock samples and total failure prediction. *Int. J. Rock Mechs. Min Sci., Vol. 44, pp. 457-467.*

Clarici, E., Millar, D.L., & D. Owen, 1994. On the practical use of artificial neural networks in minerals engineering. *Proc. 3rd Cong. of Italian-Brazillian Mining Engineering. Verona, Italy, 26th-27th September, pp 183-187.*

Clarici, E., Owen D. and Durucan S., 1993a. Modelling and representation of spatial data using artificial neural networks. *Proc. Innovative Mine Design for the 21st Century, 23 - 26 August, Kingston, Canada, p 199-204.*

Clarici, E., Owen, D., Durucan, S. and Ravenscroft, P., 1993b. Recoverable reserve estimation using a neural network. *Proc. XXIV APCOM, 31 October - 3 November, Montreal, Canada, p 145-152.*

Coca-Stefaniak, J.A., 1993. Identification of Slope Instability in the Panama Canal: A Neural Network Approach. MSc Thesis. Imperial College, Royal School of Mines, London, UK.

Cutmore, N.G., Liu, Y., and Middleton, A.G., 1997. Ore characterisation and sorting. *Minerals Eng., Vol.10, No.4, pp.421-426.*

Dawson, M.S., Fung A.K., Manry, M.T., 1997. A robust statistical-based estimator for soil moisture retrieval from radar measurements. *IEEE Trans. Geoscience & Remote Sensing, Vol.35, No.1, pp.57-67.*

Denby, B. and Burnett C.C.H., 1993. A neural network based tool for grade estimation. *Proc. XXIV APCOM, 31 October-3 November. Montreal, Canada, p 153-160.*

Deng, J.H., and Lee, C.F., 2001. Displacement back analysis for a steep slope at the Three Gorges Project site. *Int. J. Rock Mechs. Min Sci., Vol. 38, pp. 259-268.*

Dershowitz, W.S. and Einstein, H.H., 1988. Characterising Rock Joint Geometry with Joint System Models. *Rock Mechanics and Rock Engineering, Vol 21, pp 21-51.*

Dowd, P. A. and Sarac C., 1994. A neural network approach to geostatistical simulation. *J. Math. Geol. Vol. 26, No. 4, p 491-504.*

Duboisa, M.K, Bohlinga, G.C. and Chakrabarti, S. 2007. Comparison of four approaches to a rock facies classification problem. *Computers & Geosciences, Vol. 33, pp 599-617.*

Dyminsky, A.S., Parente, E., and Romanel, C., 1996. Residual soil modeling using neural networks. *Proc. 3rd Int. Conf. Computational Structures Technology, Ch.45, pp.103-106. 21-23 Aug 1996. Budapest, Hungary.*

Ellis, G.W., Yao, C., Zhao, R., and Penumadu, D., 1995. Stress-strain modelling of sands using artificial neural networks. *J. Geotech. Eng. Vol.121, No.5, pp.429-435.*

Feng, X., Lin, Y. and Wang, L. 1993. Neural networks for optimal design of underground opening support systems. *Proc. Conf. Innovative Mine Design for the 21st Century.* pp 375-383.

Feng, X. Zhang, Z. Xu, P., 1999. Adaptive and intelligent prediction of deformation time series of high rock excavation slope. *Trans of Nonferrous Metals Society of China (English Edition), Vol. 9, No. 4, pp.842-846* (Abstract only).

Feng, X.T., Zhang, Z., and Sheng, Q., 2000. Estimating mechanic rock mass parameters relating to the Three Gorges Project permanent shiplock using an intelligent displacement back analysis method. *Int. J. Rock Mechs.Min. Sci. Vol. 37, pp.-1039-1054.*

Fookes, J., 1985. Lecture notes on mining geostatistics. Unpublished. Presented at the Royal School of Mines, Imperial College, 1985.

Ghaboussi, J. Garrett, J.H., & X. Wu, 1991. Knowledge-based modelling of material behaviour with neural networks. *J. Eng. Mechs., Vol. 117, No. 1, pp 132 - 153.*

Ghaboussi, J., Sidarta, D.E., and Lade, P.V., 1994. Neural network based modelling in geomechanics. *Proc. 8th Int. Conf. Computer Methods and Advances in Geomechanics. Vol 1, Ch.115, pp.153-164. 22-28 May 1994, Morganstown, WV.*

Gokceoglu C., Yesilnacar, E., Sonmez, H. and Kayabasi, A, 2004. A neuro-fuzzy model for modulus of deformation of jointed rock masses. *Computers & Geotechnics, Vol. 31, pp 375-383.*

Goh, A.T.C., 1994. Seismic liquefaction potential assessed by neural networks. *J. Geotech. Eng. Vol.120, No.9, pp.1467-1480.*

Goh, A.T.C., 1995. Modelling soil correlations using neural networks. *J. Computing in Civ. Eng., Vol.9, No.4, pp.275-278.*

Goh, A.T.C., 1996a. Neural network modelling of CPT seismic liquefaction data. *J. Geotech. Eng. Vol.122, No.1, pp.70-73.*

Goh, A.T.C., 1996b. Pile driving records reanalyzed using neural networks. *J. Geotech. Eng. Vol.122, No.6, pp.492-495.*

Gong, P., 1996. Integrated analysis of spatial data from multiple sources using evidential reasoning and artificial neural network techniques for geological mapping. *Photogrammetric Eng. & Remote Sensing, Vol.62, No.5, pp.513-523.*

Gonzalez, R.C. and Wintz, P., 1983. Digital Image Processing (2nd edition). Addison Wesley.

Goricki, A., 2003. Classification of Rock Mass Behaviour based on a hierarchical Rock Mass Characterisation for the Design of Underground Structures. Doctoral Thesis, TU-Graz.

Greig, T. 1996. Back analysis for predictive design: Genetic algorithms with *FLAC* as an optimisation method to tackle the problem of limited data in geomechanics. MSc disertation, Dept. Earth Resource Engineering, Imperial College.

Gregory, R.L., 1972. Eye and Brain: The psychology of seeing. (2nd edition) Weidenfeld & Nicholson, London.

Gribb, M.M., and Gribb, G.W., 1994. Use of neural networks for hydraullic conductivity determination in unsaturated soil. *Proc. 2nd Int. Conf. Ground Water Ecology. Ch.40, pp.155-163.* 30 Mar 1994, Atlanta, GA.

Hadjigeorgiou, J. and Flament F., 1993. Neural network applications in mining. *Proc. XXIV APCOM, 31 October.- 3 November, Montreal, p 161-168.*

Hadjigeorgiou, J. Lessard, J.S., and Flament, F., 1995. Predicting joint behaviour using artificial neural networks, *CAMI-95.*

Hadjigeorgiou, J., Lemy, F., Cote, P., and Maldague, X., 2003. An evaluation of image analysis algorithms for constructing discontinuity trace maps. *Rock Mech Rock Eng Vol. 36, No. 2, pp.163–79.*

Hajiabdolmajid, V., Kaiser, P.K., and Martin, C.D., 2002. Modelling brittle failure of rock. *Int. J. Rock Mechs. Min. Sci. Vol. 39, pp 731-741.*

Harrison, J.P., 1995. Novel methods for discontinuity analysis in rock engineering. *PhD Thesis, Imperial College, University of London.*

Hashhash, Y.M.A., Jung, S., Ghaboussi, J., 2004. Numerical implementation of a neural network based material model in finite element analysis. *Int. J. Numer. Meth. Engng* Vol. 59. pp 989–1005.

Hebb, D.O., 1949. The Organisation of Behaviour. Wiley & Sons, New York.

Hirano, M., Moriyama, T., and K. Kawahara (1994). Prediction of occurrence of debris flow by use of the neural networks. *Proc. Int. Workshop on Floods and Inundations Related to Large Earth Movements,* pp C1.1-C1.10.

Hoek, E. 2003, Practical Rock Engineering.
http://www.rocscience.com/hoek/PracticalRockEngineering.asp

Hoek E., and Brown, E.T., 1980. Underground Excavations in Rock. *Institution of Mining and Metallurgy, London*

Homand-Etienne, F., Hoxhaa, D., & Shaob, J. F., 1998. A continuum damage constitutive law for brittle rocks. *Computers and Gotechnics, Vol. 22, No. 2, pp. 135-l 51.*

Hong, Y.S., and Rosen, M.R., 2001. Intelligent characterisation and diagnosis of the groundwater quality in an urban fractured-rock aquifer using an artificial neural network. *Urban Water, Vol. 3, pp-193-204.*

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci.,* Vol. 79, pp. 2554-2558. Reprinted in Anderson and Rosenfeld (1988).

Hopfield, J.J., and Tank, D.W., 1985. Neural computation of decision in optimisation problems. *Biol. Cybernetics, Vol 52. pp. 1-12.*

Huang, Y. and Wanstedt, S., 1996. Application of neural network model for ore boundary delineation based on geophysical logging data. *Proc IEEE Int. Conf. Neural Networks (ICNN), Ch. 398, pp.2148-2153. 2-6 June 1996, Washington DC, USA.*

Huang, Y., and Wanstedt, S., 1998. The introduction of neural network system and its applications in rock engineering. *Engineering Geology. Vol. 49, pp 253-260.*

Huang, Z.H., and Williamson, M.A., 1996. Artificial neural network modeling as an aid to source rock characterization. *Marine & Petrol. Geol., Vol.13, No.2, pp.277-290.*

Hudson, J.A., 1992. Rock Engineering Systems, Theory and Practice. *Ellis Horwood, London.*

Hudson J.A. and Hudson, J.L., 1993. Establishing potential behavioural modes of rock engineering systems by computer simulation of interaction matrix energy flux. *Int. J. Rock Mechs. Min. Sci. & Gemech. Abst. Vol. 30, No.4, pp.457-468.*

Hudson, J.A. and Harrison, J.P., 1997. Engineering Rock Mechanics, An introduction to the principles. Pergamon, Oxford.

Hudson, J.A., Sheng, J., and Arnold, P.N., 1992. Rock engineering risk assessment through critical mechanism and parameter evaluation. *Proc. 6th Australia-New Zealand Conference on Geomechanics, Christchurch, New Zealand.*

Ikegawa, Y., 1992. Three-dimensional geometrical analysis of rock mass structure. PhD Thesis, Imperial College, London.

International Society of Rock Mechanics, 1979. Suggested methods for determining the uniaxial compressive strength and deformability of rock materials. (Commission on Standardisation of Laboratory and Field Tests), *Int J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol 16, pp135-140.*

International Society of Rock Mechanics, 1983. Suggested methods for determining the strength of rock materials in triaxial compression: Revised version, (Commission on Standardisation of Laboratory and Field Tests), *Int J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol 20, pp238-290.*

ISRM Commission on Testing Methods, 1981. Rock Characterization Testing & Monitoring, ISRM Suggested Methods (E.T. Brown, ed.), Pergamon Press, Oxford.

ISRM Commission on Testing Methods, 1999. Draft ISRM Suggested Method for the complete stress-strain curve for intact rock in uniaxial compression. *Int J. Rock Mechs. Min. Sci., Vol. 36, pp279-289.*

ISRM Commission on Testing Methods, 2003. Suggested Methods for Rock Stress Estimation. *Int J. Rock Mechs. Min. Sci., Vol. 40, No. 7-8, pp955-1276.*

Itasca Consulting Group, 1993. FLAC- Fast Lagrangian Analysis of Continua Users Guide Version 3.22, (Itasca Consulting Group).

Itasca Consulting Corp., 1995. FLAC Fast Lagrangian Analysis of Continua, Version 3.3. Itasca Consulting Corp.

Jaeger, J.C and Cook, N.G.W., 1979. Fundamentals of Rock Mechanics, 3rd edition. Chapman & Hall, London.

Jamialahmadi, M., and Javadpour, F.G., 2000. Relationship of permeability, porosity and depth using an artificial neural network. *J. Pet. Sci. & Eng. Vol. 26, pp. 235-239.*

Jing, L., and Hudson, J.A., 2002. Numerical methods in rock mechanics. *Int. J. Rock. Mech, & Min. Sci., Vol. 39, pp 409–427.*

Jing, L., 2003. A reviewof techniques, advances and outstanding issues in numerical modelling for rock mechanics and rock engineering. *Int. J. Rock. Mech, & Min. Sci., Vol. 40, No. 3, pp 283–353.*

Javadi, A.A., Tan, T.P., and Zhang, M., 2003. Neural network for constitutive modelling in finite element analysis. *Computer Assisted Mechanics and Engineering Sciences, Vol. 10, pp 523-529.*

Kahramana, S., Altunb, H., Tezekicib, B.S., and Fenerc, M., 2006. Sawability prediction of carbonate rocks from shear strength parameters using artificial neural networks. *Int. J. Rock Mech. Min. Sci., Vol. 43, pp 157-164.*

Kamewada, S., Gi, H.S., Taniguchi, S., and Yoneda, H., 1990. Application of borehole image processing system to survey of tunnel. *Proc. Int. Symp. on Rock Joints, Loen, Norway, June 1990.*

Katz, S.A., Vernik, L. and Chilingar, G.V., 1999. Prediction of porosity and lithology in siliclastic sedimentary rock using cascade neural assemblies. *J. Pet. Sci. & Eng. Vol. 22, pp. 141-150.*

Khandelwal, M., and Singh, T.N., 2006. Prediction of blast induced ground vibrations and frequency in opencast mine: A neural network approach. *J. Sound and Vibration Vol. 289, pp 711–725.*

Kim, C.Y., Bae, G.J., Hong, S.W., Park, C.H., Moon, H.K., and Shin, H.S., 2001. Neural network based prediction of ground surface settlements due to tunnelling. *Computers & Geotechnics, Vol., 28, pp. 517-547.*

King, R.L., Hicks, M.A., Signer, S.P., 1993. Using unsupervised learning for feature detection in a coal-mine roof. *Eng. Applications Artificial Intelligence, Vol.6, No.6, pp.565-573.*

Klose, C.D., 2006. Self-organizing maps for geoscientific data analysis: geological interpretation of multidimensional geophysical data. *Compuers &. Geosciences, Vol. 10, No.3, pp 265–277.*

Klose C.D., Loew S., Giese R., and Borm G. 2007. Spatial predictions of geological rock mass properties based on in-situ interpretations of multi-dimensional seismic data. *Engineering Geology, Vol. 93, pp 99-116.*

Kohonen, T., 1977. Associative memory: A system theoretic approach. Berlin: Springer-Verlag.

Kohonen, T., 1982. Self-organised formation of topologically correct maps. *Biological Cybernetics, Vol 43. pp. 59-69.*

Kohonen, T., 1987. Adaptive, associative, and self-organising functions in neural computing. *Applied Optics, Vol 26, No. 23, pp 4910-4918.*

Kohonen, T., 1990. The self-organising map. *Proc. IEEE, Vol. 78, No. 9, pp 1464-1480.*

Kohonen, T., Kangas, J., and Laaksonen, J., 1992a. SOM_PAK, The Self Organising Map Package, Version 1.2. SOM Programming Team, Helsinki University of Technology, Laboratory of Computer and Information Science. (Available on Internet via ftp: cochlea.hut.fi, "anonymous", /pub/som_pak).

Kohonen, T., Kangas, J., Laaksonen, J and Torkkola, K., 1992b. LVQ_PAK, The Learning Vector Quantization Program Package, Version 2.1. *LVQ Programming Team, Helsinki University of Technology, Finland* (Available on Internet via ftp : cochlea.hut.fi, "anonymous", /pub/lvq_pak).

Kohonen, T., Hynninen, J., Kangas, J. and Laaksonen, J., 1995a. SOM_PAK, The Self Organising Map Program Package, Version 3.1. *SOM Programming Team, Helsinki University of Technology, Finland* (Available on Internet via ftp : cochlea.hut.fi, "anonymous", /pub/lvq_pak).

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J and Torkkola, K., 1995b. LVQ_PAK, The Learning Vector Quantization Program Package, Version 3.1. *LVQ Programming Team, Helsinki University of Technology, Finland* (Available on Internet via ftp : cochlea.hut.fi, "anonymous", /pub/lvq_pak).

Konagai, K., and Johansson, J., 2001. Two dimensional lagrangian finite difference method for modelling large soil deformations. *J. Struct. Mech. Earthquake Eng. Vol.18, No. 2 pp.105s-110s.*

Kwok, T.Y., and Yeung, D.Y., 1997. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Trans. Neural Networks, Vol.8, No.3, pp.630-645.*

Lee, C.A., and Sterling, R.L., 1992a. Identifying probable failure modes for underground openings using a neural network. *Int. J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol. 29, No. 1, pp 49-67.*

Lee, C.A. and Sterling, R.L., 1992b. A neural network-expert system hybrid approach for tunnel design. *Proc 33rd US Symp. Rock Mechanics, p 501-510.*

Lemy, F., and Hadjigeorgiou, 2003. Discontinuity trace map construction using photographs of rock exposures. *Int. J. Rock Mechs. Min. Sci. Vol, 40, pp.903-917.*

Lerner, B. Guterman, H., Aladjem, M., and Dinstein, I., 2000. On the Initialisation of Sammon's Nonlinear Mapping. *Pattern Analysis & Applications Vol.3 No.1*

Leu, S.S., Chen, C.N., Chang., S.L., 2001. Data mining for tunnel support stability: neural network approach. *Automation in Construction, Vol. 10, pp. 429-441.*

Lever, P.J.A., Wang, F.Y., Chen, D.D., and Shi, X.B., 1995. Autonomous robotic mining excavation using fuzzy-logic and neural networks. *J. Intelligent & Fuzzy Systems, Vol.3, No.1, pp.31-42.*

Levine, E.R., Kimes, D.S., and Sigillito, V.G., 1996. Classifying soil structure using neural networks. *Ecological Modelling, Vol.92, No.1, pp.101-108.*

Lippman, R.P., 1987. An introduction to computing using neural nets. *IEEE ASSP Magazine, Vol. 4, pp. 4-22.*

Logar J., and Turk G., 1997. Neural network as a constitutive model of soil. *Zeitschrift fur Angewandte Mathematik und Mechanik, Vol.77, No.S1, pp.S195-S196* (Abstract only).

Lu, H.J., Setiono, R., and Liu, H., 1996. Effective data mining using neural networks. *IEEE Trans. Knowledge & Data Eng., Vol.8, No.6, pp.957-961.*

MacDonald, J.A. and Carr, J.R., 1989. Application of geostatistics to image analysis. *Proceedings IGARSS'89 (Vancouver, Canada July 1989).*

McClulloch, W.W., and Pitts,W., 1943. A logical calculus of the ideas imminent in nervous activity. *Bull. Math. Biophys., Vol. 5., pp. 115-133.*

Maren, A.J., Harston, C.T., and Pap, R.M., 1990. Handbook of Neural Computing Applications. Academic Press, Inc., San Diego, California.

Marmoa, R., Amodiob,S., Tagliaferrid, R., Ferrerib, V., and Longof, G. 2005. Textural identification ofcarbonate rocks by image processing and neural network: Methodology proposal and examples. *Computers and Geosciences, Vol 31, pp 649-659.*

Marti, J., and Cundall, P., 1982. Mixed discretisation procedure for accurate modelling of plastic collapse. *Int J. Numerical and Analytical Methods in Geomech. Vol 6., pp.129-139.*

Martin, C.D. and Christiansson, R., 1990. Overcoring in highly stressed granite— the influence of microcracking. *Int. J. Rock Mechs. Min. Sci.& Geomech. Abs., Vol, 28, No. 1, pp.53-70.*

Martin, C.D. and Chandler, N.A., 1994. The progressive fracture of the Lac du Bonnet granite. *Int. J. Rock Mechs. Min. Sci. Vol, 31, No. 6, pp.643-659.*

Mase, H., Sakamoto, M and Sakai, T, 1995. Neural network for stability analysis of rubble-mound breakwaters. *J. Waterway, Port, Coastal & Ocean Eng. Vol.121, No.6, pp.294-299.*

Meulenkamp, F., and Alvarez-Grima, M., 1999. Application of neural networks for the prediction of the unconfined compressive strength (UCS) from Equotip hardness. *Int. J. Rock Mechs. Min. Sci., Vol. 36, pp. 29-39.*

Millar, D.L. 1992. CARD: Computer-Aided Recognition of Discontinuities A Progress Report submitted to the University of London (Imperial College of Science, Technology and Medicine) in fulfilment of the requirements for transfer of registration from M.Phil. to Ph.D.

Millar, D.L. & Clarici, E., 1994. Investigation of back-propagation artificial neural networks in modelling of the stress strain behaviour of sandstone rock. *Proc. IEEE Int. Conf. Neural Networks. June 26- July 2, Orlando, Florida, USA. Vol 7.*

Millar, D.L., and Hudson, J.A., 1994. Performance monitoring of rock engineering systems using neural networks. *Trans. Inst. Min. & Metall. Sect. A-Mining Industry, Vol.103, pp.A 3-A 16.*

Millar, D.L., Jiao, Y., Arnold, P.N., and Hudson, J.A., 1994. Rock Engineering Systems Approach to Risk Assessment. *Proc. Conf. Risk Assessment in the Extractive Industry, 23-24 March 1994, Exeter, UK, 19p.*

Millar, D.L., and Calderbank, P.A., 1995. On the investigation of multilayer feedforward neural network model of rock deformability behaviour. *Proc 8th Int. Cong. Rock Mechs., Ch.189, pp.933-938. 25 - 30 Sep 1995, Tokyo, Japan.*

Millar, D.L., Clarici, E., Calderbank, P.A., and Marsden, J.R., 1995. On the practical use of a neural network strategy for the modeling of the deformability behaviour of Croslands Hill sandstone rock. *Proc. Conf. Applcation of Computers and Operations Research in the Minerals Industries (APCOM XXV), Vol.95, Ch.83, No.4, pp.457-465.* 9 - 14 Jul 1995, *Brisbane, Australia.*

Mimuro, T., Michiie, T., Sasaki, T., Kurioka, H., and Ino, M., 1991. Analysis of optical characteristics of rock image data by borehole TV. *Proc. 7th Int. Cong. on Rock Mechanics. Aachen, 1991. Balkema/Brookfield.*

Minsky, M. & Papert, S., 1969. Perceptrons: An Introduction to Computational Geometry, MIT Press, Cambridge, MA.

Mohaghegh, S., Arefi, R., Ameri, S., Aminiand, K., and Nutter, R., 1996. Petroleum reservoir characterization with the aid of artificial neural networks. *J. Pet. Sci. & Eng., Vol.16, No.4, pp.263-274.*

Moore, J.F.A., 1974. Mapping major joints in the Lower Oxford Clay using terrestrial photogrammetry. *Quarterly Journal of Engineering Geology, Vol 7, pp 57-67.*

Motooka, S., Kitmai, M., Ogawa, K, and Takano, N., 1995. Ocean experiments to discriminate seabed materials by echo pulse and neural network at Uchiura Bay. *Japanese J. of Applied Physics, Part 1, Vol.34, No.5B, pp.2930-2935.*

Muller, B. and Reinhardt, J., 1990. The Physics of Neural Networks. Springer-Verlag, Berlin.

Musil, M., and Plesinger, A., 1996. Discrimination between local earthquakes and quarry blasts by multilayer perceptrons and Kohonen maps. *Bulletin Seismological Soc. America. Vol.86, No.4, pp.1077-1090.*

Neelakanta, P.S., and De Groff, D.F., 1994. Neural Network Modelling : Statistical Mechanics and Cybernetic Perspectives. CRC Press, Boca Raton.

Nie, X. and Zhang, Q. 1994. Prediction of rock mechanical behaviour by artificial neural network, a comparison with traditional method. *Proc. Conf. Integral Approach to Applied Rock Mechanics. Santiago, Chile, pp 279-287.*

Nielsen, N.J., 1965. *Learning Machines: Foundations of Trainable Pattern Classifying Systems.* McGraw Hill, New York.

Ninomiya, Y., 1995., Quantitative estimation of SiO_2 content in igneous rocks using thermal infrared-spectra with a neural network approach. *IEEE Trans. Geoscience & Remote Sensing. Vol.33, No.3, pp.684-691.*

Ohnishi, Y., and Soliman, M., 1995. Seepage under concrete dam founded on rock formation using artificial neural networks. *Proc. Int. Workshop on Rock Foundation. Ch.70, pp.355-360. 30 Sept 1995, Tokyo, Japan.*

Pachepsky, Y.A, Timlin, D., and Varallyay, G., 1996. Artificial neural networks to estimate soil-water retention from easily measurable data. *J. Soil Science Soc. America, Vol.60, No.3, pp.727-733.*

Pao. Y.H., 1989. Adaptive Pattern Recognition and Neural Networks, Addison Wesley.

Parking, R.M., and Calkin, D.W., 1994. Roadstone aggregate - An intelligent route to computer integrated manufacturing. *Robotics & Computer-Integrated Manufacturing, Vol.11, No.4, pp.345-350.*

Perez, R. del Villar R. and Flament F., 1993. Level detection in a flotation column using an artificial neural network. *Proc. XXIV APCOM, 31 October - 3 November, Montreal, Canada p 174-181.*

Pezeshk, S., Camp, C.V., and Karprapu, S., 1996. Geophysical log interpretation using neural network. *J. Computing Civ. Eng. Vol.10, No.2, pp.136-142.*

Priest, S.D., 1985. Hemispherical Projection Methods in Rock Mechanics. George Allen and Unwin.

Priest, S.D., 1992. Discontinuity Analysis for Rock Engineering. Kluwer Academic, Dordrecht.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., 1992. Numerical Recipes in C, The Art of Scientific Computing (2nd Edition). Cambridge University Press.

Reid, T.R., 1998. A methodology for the detection of discontinuity traces in digital images of rock mass exposures. Ph.D. thesis, Imperial College, London, UK.

Reid T.R., and Harrison, J. P., 2000. A semi-automated methodology for discontinuity trace detection in digital images of rock mass exposures, *Int. J. Rock Mechs. Min Sci, Vol.37, No. 7, pp.1073-1089*

Rosenblatt, F., 1959. Two theorems of statistical separbility in the Perceptron. *Proc 10th Symp. Mechanisation of Thought Processes., held at the National Physical Laboratory, UK, Nov. 1958. Vol. 1, pp. 421-456.*

Rosenblatt, F., 1962. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan, New York.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., 1986. Learning internal representations by error propagation. In, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition.* Rumelhart & McClelland, (eds.). Vol.1 pp 318-362, MIT Press, Cambridge, MA.

Samecka-Cymerman, A., Stankiewicz, A., Kolon, K. and Kempers, A.J., 2007. Self-organizing feature map (neural networks) as a tool in classification of the relations between chemical composition of aquatic bryophytes and types of streambeds in the Tatra national park in Poland. *Chemosphere Vol. 67, pp 954–960.*

Sammon, Jr., J.W., 1969. A nonlinear mapping for data structure analysis. *IEEE Trans. Computers, Vol C-18, No. 5, pp 401-409.*

Schaap, M.G., and Bouten, W., 1996. Modelling water-retention curves of Sanely soils using neural networks. *Water Resources Research. Vol.32, No.10, pp.3033-3040.*

SAIC, 1992. ANSIM Users Manual. Science Applications International Corporation.

Shin, H.S. and Pande, G.N., 2000. On self-learning finite element codes based on monitored response of structures. *Computers & Geotechnics, Vol 27, pp 161-178.*

Singh, V.K., Singh, D., and Singh, T.N., 2001. Prediction of strength properties of some schistose rocks from petrographic properties using artificial neural networks. *Int. J. Rock Mechs. Min. Sci., Vol.38, pp. 269-284.*

Sirat, M., and Talbot, M.J., 2001. Application of artificial neural networks to fracture analysis at the Aspo HRL, Sweden: fracture sets classification. *Int. J. Rock Mechs Min. Sci., Vol. 38, pp 621-639.*

Softcover International Ltd, 2001. Scan2CAD V6 User Manual – Regular and Pro versions, (2nd Edition). Available from http://www.rastertovector.com

Sonmeza,H., Gokceoglua, C., Nefeslioglub, H.A., Kayabasic, A., 2006. Estimation of rock modulus: For intact rocks with an artificial neural network and for rock masses with a new empirical equation. *Int J. Rock Mechs. Min. Sci., Vol. 43, pp 224-235.*

Smith, G.N., 1978, Elements of soil mechanics for Civil and Mining Engineers (4th edition). Granada, London.

Swanson, S R and Brown, W S, 1972. An observation of loading path independence of fracture in rock, *Int J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol 8, pp277-281.*

Swiniarski, R., Hidalgo, H., and Gomeztrevino, E., 1993. Neural network applied in the geophysical inversion problem. *Ground Sensing, Vol.1941, Ch.29, pp.151-158.*

Szwedzicki, T., 2003. Rock mass behaviour prior to failure. *Int. J. Rock Mechs Min. Sci., Vol. 40, pp 573-584.*

Takahashi, K., and Hosaka, H., 1996. Neural-network-based controller used for directional control of tunneling machine. *JSME Int. J. Series C Dynamics Control Robotics Design & Manfacturing, Vol.39, No.4, pp.781-789.*

Takahashi, K., and Yamada, T., 1997. A radial-basis-function-network-based controller derived from a neural-network-based controller and its application to controlling mechanical systems. *JSME Int. J. Series C Dynamics Control Robotics Design & Manfacturing, Vol.40, No.I, pp.42-51.*

Tanimoto. C., Murai, S., Kiyama, H. and Michihiro, K., 1991. Hydraulic behaviour of rock joints through the infrared photo image analysis. *Proc. 7th Int. Cong. on Rock Mechanics, Aachen, 1991. A.A. Balkema/Brookfield.*

Tamari, S., Wosten, J.H.M., and RuizSuarez, J.C., 1996. Testing an artificial neural network for predicting soil hydraulic conductivity. *J. Soil Science Soc. America. Vol.60, No.6, pp.1732-1741.*

Teh, C.I., Wong, K.S., Goh A.T.C., and Jaritngam, S., 1997. Prediction of pile capacity using neural networks. *J. Computing Civ. Eng., Vol.11, No.2, pp.129-138.*

Tohmaz, A.S., and Hassan, A.E., 1995. Application of artificial neural networks to skidder traction performance. *J. Terramechanics, Vol.32, No.3, pp.105-114.*

Topping, B.H.V., Khan, A.I., and Bahreininejad, A. 1997. Parallel training of neural networks for finite element mesh decomposition. *Computers & Structures, Vol.63, No.4, pp.693-707.*

Tsoi, A.C., 1989. Multilayer perceptron trained using radial basis functions. *Electronic Lett. Vol.25, No.19, pp.1296-1297.*

Vermeer, P.A. and de Borst, R., 1984. Non-associated plasticity for soils, concrete and rock. *Heron, Vol.29, No.3, pp. 1-64*

Wagner, B.J., 1995. Sampling design methods for groundwater modeling under uncertainty. *Water Resources Res. Vol31. No.10, pp 2581-2591.*

Wang, H.B., Xu W.Y., and Xu, R.C., 2005. Slope stability evaluation using Back Propagation Neural Networks. *Engineering Geology, Vol. 80, pp302– 315.*

Waversik, W R and Fairhurst, C, 1970. A study of brittle rock fracture in laboratory compression experiments, *Int J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol 7, pp561–575.*

Whittaker, R.N., Singh, R.N. and Sun, G. 1992. Rock Fracture Mechanics, Principles, Design and Applications, Elsevier, Amsterdam.

Willams, T.P., and Gucunski, N., 1995. Neural network for backcalculation of modulii from SASW test. *J.Computing Civ. Eng., Vol.9, No.1, pp.1-8*

Wong, P.M., and Gedeon, T.D., 1995. A new method to detect and remove the outliers in noisy data using neural networks - Error sign testing. *Systems Research & Information Science Vol.7, No.1, pp.55-65.*

Wong, P.M., Gedeon, T.D., and Taggart, I.J., 1995a. An improved technique in porosity prediction - A neural network approach. *IEEE Trans. Geoscience & Remote Sensing, Vol.33,*

No.4, pp.971-980.

Wong, P.M., Taggart, I.J., and Gedeon, T.D., 1995b. Use of a neural network to predict porosity and permeability of a petroleum reservoir. *Artificial Intelligence Applications, Vol.9, No.2, pp.27-37.*

Wu, X and Ghaboussi, J, 1992. Neural network modelling of composite material with the emphasis on reinforced concrete, *Proc. Conf.Am. Soc. Civ. Eng.*, Dallas, Texas.

Wu, X.P., and Zhou, Y.X., 1993. Reserve estimation using neural network techniques. *Computers and Geosciences, Vol.19, No.4, pp.567-575.*

Xu, L., Krzyzak, A., and Oja, E., 1993. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Trans. Neural Networks, Vol.4, No.4, pp.636-649.*

Yang, H.S., and Kim, N.S., 1996. Determination of rock properties by accelerated neural network. *Proc 2nd North American Rock Mechs. Symp. (NARMS 96), Ch.259, pp.1567-1572. 19-21 June 1996, Montreal, Canada.*

Yang, C.P., 2004. Estimating cement take and grout efficiency on foundation improvement for Li-Yu-Tan dam. *Engineering Geology. Vol. 75, pp I –14.*

Yi, H., and Lindqvist, P.A., 1995. Prediction of rock quality parameters by using neural network models. *Proc. 4th Int. Symp. Mine Planning & Equipment Selection. Ch.180, pp.933-937. 31 Oct - 3 Nov 1995, Calgary, Canada.*

Zhang, Q. Song, J.R., and Nie, X.Y., 1991. Application of neural network models to rock mechanics and rock engineering. *Int. J. Rock Mechs. Min. Sci. & Geomech. Abstr., Vol. 28, No. 6, pp 535-540.*

Zhang, Y.X., Wang, T.Q., Wang, Y.J., Duan, X.H., Zhang, P.Y., Kong, L.N., Sheng, T.S., 1995. An artificial neural network for forecasting the amount of Chinese colloery roadway surrounding rock deformation. *Proc. Conf. Application of Computers and Operations Research in the Minerals Industries (APCOM XXV), Vol.95, Ch.83, No.4, pp.123-124.* 9 - 14 Jul 1995, *Brisbane, Australia.*

Zhou, Y.X., and Wu, X.P., 1994. Use of neural networks in the analysis and interpretation of site investigation data. *Computers and Geotechnics, Vol.16, No.2, pp.105-122.*

A VBA code used to simulate triaxial and polyaxial ('plane strain') compression tests

The code listing in this section is mainly presented to provide a compact, documented record of how the synthesised data sets discussed in Chapter 5 were produced. In order to use the code, it should be keyed into a single VBA module within the Microsoft Excel environment. On a single worksheet called "TX", the following named ranges (all comprising single cells) need to be defined:

"E",	Young's modulus
"v",	Poisson's ratio
"ConfPress",	Confining pressure
"MaxAxialStrain",	Maximum axial strain to be used in the similated
"PlaneStrain",	Whether or not a plane strain simulation is conducted (Boolean)
"SampleRate",	Parameter that controls the number of samples in the data set.
"RandomParameter",	Simulates the degree of control of the triaxial test (see text)
"RandomSeed"	Random seed

These named ranges (and the corresponding values) should be located within the first 9 rows of the "TX" worksheet.

```
Option Explicit
'
' Triaxial test simulator
'
'  Module scope arrays for stress / strain components and their increments
'
   Dim sxx() As Double
   Dim syy() As Double
   Dim szz() As Double

   Dim exx() As Double
   Dim eyy() As Double
   Dim ezz() As Double

   Dim dsxx() As Double
   Dim dsyy() As Double
   Dim dszz() As Double

   Dim dexx() As Double
   Dim deyy() As Double
   Dim dezz() As Double
'
'  Module scope variables for elastic constants
'
   Dim E As Double
   Dim v As Double
   Dim G As Double
```

```
    Dim k As Double
    Dim Lam1 As Double
    Dim Lam2 As Double
'
'   Other variables
'
    Dim i As Integer
    Dim Length As Integer

Sub SimulateTXtests()
'
'   Driver for SimulateTX procedure
'   Sets the random number generator seed
'
    Dim WS As Worksheet
    Dim NamedRange As Range
    Dim RandomSeed As Single

    Set WS = Worksheets("TX")
    Set NamedRange = WS.Range("RandomSeed")
    RandomSeed = NamedRange.Value

    Rnd -1
    Randomize RandomSeed

    i = 0
    Length = 0
    SimulateTX
End Sub


Private Sub SimulateTX()
'
'   Procedure that implements simulations described in Chapter 5 of
'   Ph.D. Thesis titled "Parallel Distributed Processing in Rock Engineering"
'   by D.L. Millar
'
'   Mechanical variables
'
    Dim ConfPress As Double        'Target confining pressure for simulated TX test (Pa)
    Dim MaxAxialStress As Double   'Target peak stress for simulated TX test (Pa)
    Dim SampleRate As Double       'Parameter that increases the simulated sampling rate (Hz)
    Dim RandomParameter As Double  'Parameter that defines the variance in strain rate
    Dim PlaneStrain As Boolean     'Boolean switch to define a plane strain 'TX' compression
    Dim erate As Double            'Strain rate (units are: /second)
'
'   Variables to interact with the MS Excel worksheet
'
    Dim WS As Worksheet
    Dim NamedRange As Range
'
'   Other variables
'
    Dim Rand As Double

    Set WS = Worksheets("TX")
'
'   Read in values for E and v from named ranges in MS Excel worksheet TX
'
```

```
    Set NamedRange = WS.Range("E")
    E = NamedRange.Value
    Set NamedRange = WS.Range("v")
    v = NamedRange.Value
'
'   Read in confining pressure, maximum axial stress, sample rate and random parameter
'
    Set NamedRange = WS.Range("ConfPress")
    ConfPress = NamedRange.Value
    Set NamedRange = WS.Range("MaxAxialStress")
    MaxAxialStress = NamedRange.Value
    Set NamedRange = WS.Range("SampleRate")
    SampleRate = NamedRange.Value
    Set NamedRange = WS.Range("RandomParameter")
    RandomParameter = NamedRange.Value
    Set NamedRange = WS.Range("PlaneStrain")
    PlaneStrain = NamedRange.Value
'
'   Compute values for the remaining elastic constants
'
    G = E / (2 * (1 + v))
    k = E / (3 * (1 - 2 * v))
    Lam1 = k + (4 / 3) * G
    Lam2 = k - (2 / 3) * G
'
'   Allocate memory for the state arrays
'
    If Length - i < 50 Then
       Length = Length + 50
       AllocateArrays Length
    End If
'
'   Start with a stress and strain free state
'
    i = 1     'i should be 1 here
    sxx(i) = 0
    syy(i) = 0
    szz(i) = 0
    exx(i) = 0
    eyy(i) = 0
    ezz(i) = 0
'=============================================================================
'   Load hydrostatically to confining pressure
'
    Do
'
'      Ensure there's enough memory allocated for the state arrays
'
       If Length - i < 50 Then
          Length = Length + 50
          AllocateArrays Length
       End If

       i = i + 1
'
'      Compute the strain rate for the hydrostatic loading stage
'      In terms of stress rate:
'      Factor 0.75e6 (Pa/s) is mid way between the permissable range
'      Factor 0.5e6 (Pa/s) is lower end of permissible range
```

```
'    Factor 1.0e6 (Pa/s) is upper end of permissible range
'
     If RandomParameter = 0 Then
        erate = 750000# / (3 * k) ' units are: per second
     Else
        Rand = 10000 / RandomParameter
        erate = WorksheetFunction.BetaInv(Rnd(), Rand, Rand, 500000# / (3 * k), 1000000# / (3 * k))
     End If
'
'    Adjust for the sampling rate
'
     erate = erate / SampleRate

     dexx(i) = erate
     deyy(i) = erate
     dezz(i) = erate
'
'    Compute incremental stress tensor
'
     stress_from_strain i
'
'    Update the state of the simulated sample
'
     update_strain_state i
     update_stress_state i

  Loop While sxx(i) < ConfPress
'
' Correct last value such that it's equal to the confining pressure
'
  erate = (ConfPress - sxx(i - 1)) / (3 * k) ' units are: per second
  dexx(i) = erate
  deyy(i) = erate
  dezz(i) = erate

  stress_from_strain i
  update_strain_state i
  update_stress_state i

'*************************************************************************

  If Not PlaneStrain Then

'=========================================================================
' TX LOADING: Simulate the deviatoric loading at constant confining pressure
' Until specified maximum axial stress is attained.
' Structure of code is similar to HS loading section.
'
  Do
     If Length - i < 50 Then
        Length = Length + 50
        AllocateArrays Length
     End If

     i = i + 1

     If RandomParameter = 0 Then
        erate = 750000# / E
     Else
```

```
        Rand = 10000 / RandomParameter
        erate = WorksheetFunction.BetaInv(Rnd(), Rand, Rand, 500000# / E, 1000000# / E)
      End If
      erate = erate / SampleRate

      dezz(i) = erate
      dexx(i) = (-Lam2 / (Lam1 + Lam2)) * dezz(i)
      deyy(i) = dexx(i)

      stress_from_strain i
      update_strain_state i
      update_stress_state i

    Loop While szz(i) < MaxAxialStress

    erate = (MaxAxialStress - szz(i - 1)) / E ' units are: per second
    dezz(i) = erate
    dexx(i) = (-Lam2 / (Lam1 + Lam2)) * dezz(i)
    deyy(i) = dexx(i)

    stress_from_strain i
    update_strain_state i
    update_stress_state i
'=============================================================================
  ' TX UNLOADING
    Do
      If Length - i < 50 Then
        Length = Length + 50
        AllocateArrays Length
      End If

      i = i + I

      If RandomParameter = 0 Then
        erate = -750000# / E
      Else
        Rand = 10000 / RandomParameter
        erate = -WorksheetFunction.BetaInv(Rnd(), Rand, Rand, 500000# / E, 1000000# / E)
      End If
      erate = erate / SampleRate

      dezz(i) = erate
      dexx(i) = (-Lam2 / (Lam1 + Lam2)) * dezz(i)
      deyy(i) = dexx(i)

      stress_from_strain i
      update_strain_state i
      update_stress_state i

    Loop While szz(i) > ConfPress

    erate = (ConfPress - szz(i - 1)) / E ' units are: per second
    dezz(i) = erate
    dexx(i) = (-Lam2 / (Lam1 + Lam2)) * dezz(i)
    deyy(i) = dexx(i)

    stress_from_strain i
    update_strain_state i
    update_stress_state i
```

```
'*************************************************************************

   Else   ' It's a plane strain case

'=========================================================================
'  PLANE STRAIN : Simulate the deviatoric loading at constant confining pressure
'  Until specified maximum axial stress is attained
'
   Do
      If Length - i < 50 Then
         Length = Length + 50
         AllocateArrays Length
      End If

      i = i + 1

      If RandomParameter = 0 Then
         erate = (750000# * (1 + v) * (1 - v)) / E
      Else
         Rand = 10000 / RandomParameter
         erate = WorksheetFunction.BetaInv(Rnd(), Rand, Rand, (500000# * (1 + v) * (1 - v)) / E,
(1000000# * (1 + v) * (1 - v)) / E)
      End If
      erate = erate / SampleRate

      dezz(i) = erate
      dexx(i) = (-v / (1 - v)) * dezz(i)
      deyy(i) = 0

      stress_from_strain i
      update_strain_state i
      update_stress_state i

   Loop While szz(i) < MaxAxialStress
   erate = (MaxAxialStress - szz(i - 1)) * (1 + v) * (1 - v) / E ' units are: per second

   dezz(i) = erate
   dexx(i) = (-v / (1 - v)) * dezz(i)
   deyy(i) = 0

   stress_from_strain i
   update_strain_state i
   update_stress_state i
'=========================================================================
'  PLANE STRAIN Unload deviatorically
'
   Do
      If Length - i < 50 Then
         Length = Length + 50
         AllocateArrays Length
      End If

      i = i + 1

      If RandomParameter = 0 Then
         erate = -(750000# * (1 + v) * (1 - v)) / E ' units are: per second
      Else
         Rand = 10000 / RandomParameter
```

```
        erate = -WorksheetFunction.BetaInv(Rnd(), Rand, Rand, (500000# * (1 + v) * (1 - v)) / E,
(1000000# * (1 + v) * (I - v)) / E)
    End If
    erate = erate / SampleRate

    dezz(i) = erate
    dexx(i) = (-v / (1 - v)) * dezz(i)
    deyy(i) = 0

    stress_from_strain i
    update_strain_state i
    update_stress_state i

  Loop While szz(i) > ConfPress
  erate = ((ConfPress - szz(i - 1)) * (1 + v) * (1 - v)) / E ' units are: per second

  dezz(i) = erate
  dexx(i) = (-v / (1 - v)) * dezz(i)
  deyy(i) = 0

  stress_from_strain i
  update_strain_state i
  update_stress_state i

'*****************************************************************

  End If

'=================================================================
'  Unload hydrostatically
'
  Do
    If Length - i < 50 Then
      Length = Length + 50
      AllocateArrays Length
    End If

    i = i + I

    If RandomParameter = 0 Then
      erate = -750000# / (3 * k)
    Else
      Rand = 10000 / RandomParameter
      erate = -WorksheetFunction.BetaInv(Rnd(), Rand, Rand, 500000# / (3 * k), 1000000# / (3 * k))
    End If
    erate = erate / SampleRate

    dexx(i) = erate
    deyy(i) = erate
    dezz(i) = erate

    stress_from_strain i
    update_strain_state i
    update_stress_state i

  Loop While sxx(i) > 0

  erate = (0 - sxx(i - 1)) / (3 * k) ' units are: per second
  dexx(i) = erate
```

```
    deyy(i) = erate
    dezz(i) = erate

    stress_from_strain i
    update_strain_state i
    update_stress_state i
'
'   Trim the state arrays
'
    Length = i
    AllocateArrays Length
'
'   Wipe any state array values from the worksheet and output the state arrays
'
    Set NamedRange = WS.Range("statearrays")
    NamedRange.ClearContents

    For i = 1 To Length
        WS.Cells(17 + i, 3 + 1) = dexx(i)
        WS.Cells(17 + i, 3 + 2) = deyy(i)
        WS.Cells(17 + i, 3 + 3) = dezz(i)
        WS.Cells(17 + i, 3 + 4) = dsxx(i)
        WS.Cells(17 + i, 3 + 5) = dsyy(i)
        WS.Cells(17 + i, 3 + 6) = dszz(i)
        WS.Cells(17 + i, 3 + 7) = exx(i)
        WS.Cells(17 + i, 3 + 8) = eyy(i)
        WS.Cells(17 + i, 3 + 9) = ezz(i)
        WS.Cells(17 + i, 3 + 10) = sxx(i)
        WS.Cells(17 + i, 3 + 11) = syy(i)
        WS.Cells(17 + i, 3 + 12) = szz(i)
    Next
'
' Deallocate arrays
'
    AllocateArrays 1
End Sub

Private Sub AllocateArrays(Length As Integer)
'
'   This routine allocates memory to store stress / strain state and increments
'
    ReDim Preserve sxx(1 To Length) As Double
    ReDim Preserve syy(1 To Length) As Double
    ReDim Preserve szz(1 To Length) As Double

    ReDim Preserve exx(1 To Length) As Double
    ReDim Preserve eyy(1 To Length) As Double
    ReDim Preserve ezz(1 To Length) As Double

    ReDim Preserve dsxx(1 To Length) As Double
    ReDim Preserve dsyy(1 To Length) As Double
    ReDim Preserve dszz(1 To Length) As Double

    ReDim Preserve dexx(1 To Length) As Double
    ReDim Preserve deyy(1 To Length) As Double
    ReDim Preserve dezz(1 To Length) As Double
End Sub

Private Sub stress_from_strain(i As Integer)
```

```
'
'  This routine works out the stress component increments
'
   dsxx(i) = Lam1 * dexx(i) + Lam2 * deyy(i) + Lam2 * dezz(i)
   dsyy(i) = Lam2 * dexx(i) + Lam1 * deyy(i) + Lam2 * dezz(i)
   dszz(i) = Lam2 * dexx(i) + Lam2 * deyy(i) + Lam1 * dezz(i)
End Sub

Private Sub update_strain_state(i As Integer)
'
'  This routine updates the simulated specimen's state of strain
'
   exx(i) = exx(i - 1) + dexx(i)
   eyy(i) = eyy(i - 1) + deyy(i)
   ezz(i) = ezz(i - 1) + dezz(i)
End Sub

Private Sub update_stress_state(i As Integer)
'
'  This routine updates the simulated specimen's state of stress
'
   sxx(i) = sxx(i - 1) + dsxx(i)
   syy(i) = syy(i - 1) + dsyy(i)
   szz(i) = szz(i - 1) + dszz(i)
End Sub
```

B1 Detailed worked example of FLAC calculation sequence

In this Section, a numerical worked example of the calculation sequence is presented, for an explicit finite difference scheme written by the author that precisely shadows the calculation sequence of *FLAC* for a single zone. The intended readership of this Appendix is *FLAC* experts interested in the specifics of how the ANN UDM was coupled to the code. Due to the novelty of the approach, the ANN UDM was initially developed and tested outside the *FLAC* environment, in order it could be debugged effectively, by examining variable and parameter values at each stage of the calculation cycle. As this level of detail is not available within the *FLAC* environment, a single zone finite difference scheme, following the *FLAC* calculation cycle precisely, was developed in MS Excel.

A compression test conducted on a linearly elastic rock specimen with E= 40 GPa and ν=0.25 is simulated. The rate of 'axial' strain is specified at 2.650 E-06 m/s, to yield an axial stress increment of 0.75 MPa / s. The size of the specimen simulated approximately conforms to the ISRM standard (ISRM, 1979) for uniaxial compression tests (Figure B1.1), although it is severely distorted in comparison to the right cylinder that is specified. A decision was taken to deliberately distort the shape of the body simulated from that of a right cylinder in order to eliminate symmetry within the model that could have led to apparently correct answers through cancellation of errors.

The files required to do this worked example within *FLAC* are given in Appendix B2 in order that the reader can verify the results obtained. The *FLAC* plot of the grid and conditions applied produced within this file are shown in Figure B1.1 below.

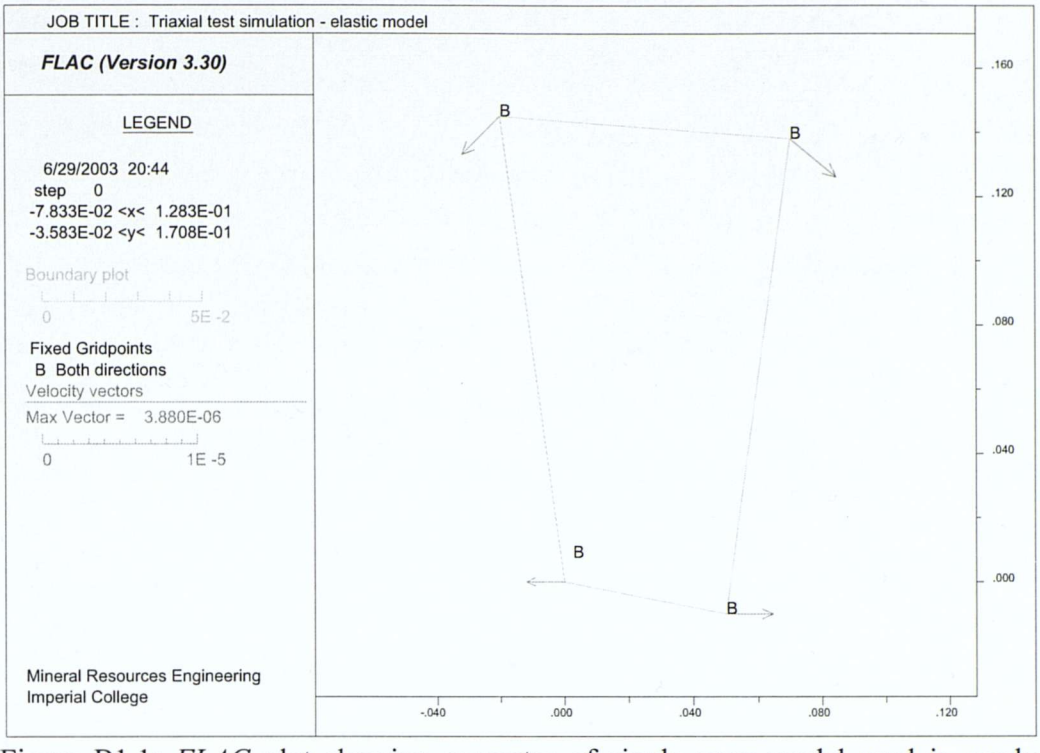


Figure B1.1: *FLAC* plot showing geometry of single zone model used in worked example, vertices where velocities are fixed (B), and applied velocity vectors.

B1.1 Co-ordinate system and sign convention

A Cartesian co-ordinate system (Figure B1.2) is adopted. The positive z-axis points out of the page. On a surface where the outward normal points in the positive co-ordinate direction, positive stress components (normal or shear) act in the positive co-ordinate directions. Tensile normal stress components are positive, contractile normal strain is negative. Shear strain components follow the convention for shear stress components. Angular measures are positive anticlockwise, referenced to the positive x-direction. The co-ordinate system adopted is identical to the one used within *FLAC*.

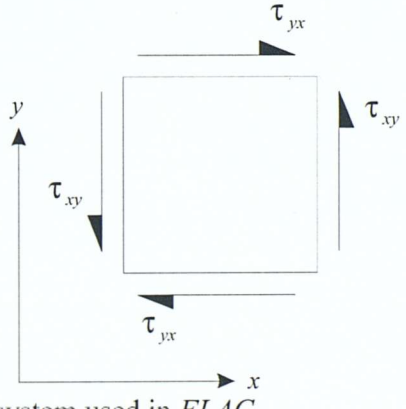


Figure B1.2: Co-ordinate system used in *FLAC*

B1.2 Finite difference grid

A single zone finite difference grid is assumed. The co-ordinates of the vertices of the quadrilateral zone are as shown in Figure B1.3. In the formulation, each quadrilateral zone is represented by two sets of two overlaid triangles in exactly the same way as in *FLAC*. These triangles, shown in Figure B1.3, are labelled triangles *a*, *b*, *c* and *d*. For a rectangular quadrilateral zone where vertex 1 of the rectangle is closest to the origin of the global co-ordinate system, the logical order of labelling of the triangles is as follows:

a	Top LHS triangle
b	Bottom RHS triangle
c	Top RHS triangle
d	Bottom LHS triangle

Various commands within *FLAC* allow the finite difference grid to be deformed to an arbitrary shape; rectangular or square zones need not be rectangular or square, and the deformation mapping can allow for rotation of the quadrilateral zone. In these cases, the order of the triangles may not appear to follow that indicated above.

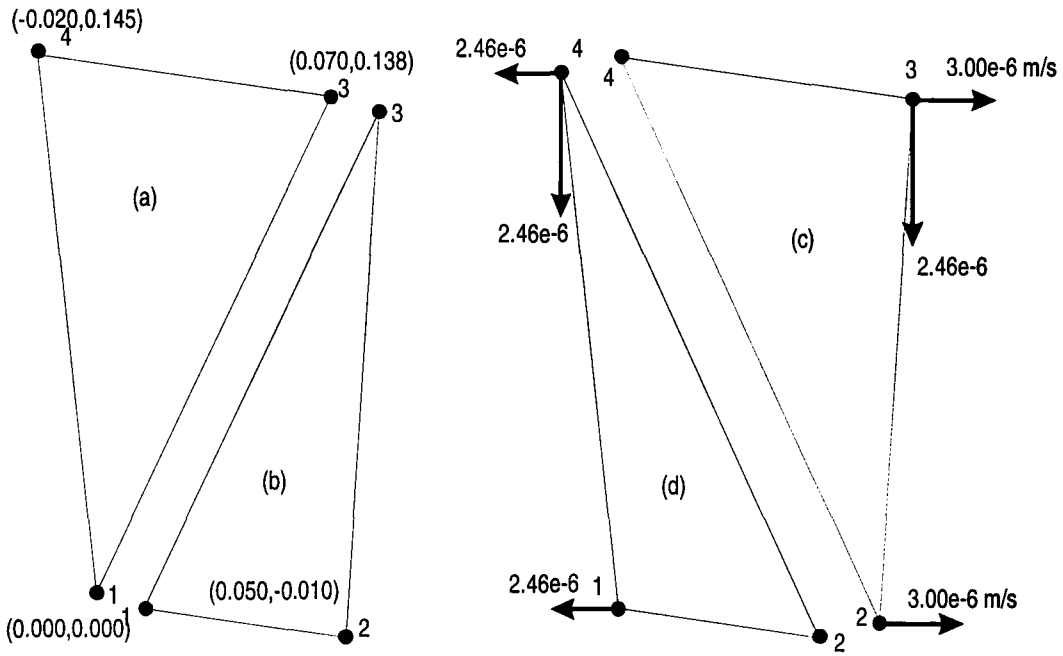


Figure B1.3: Two pairs of triangles that are used in the calculation of finite difference zone velocity gradients, strain increments, stress increments and nodal forces. Vertex co-ordinates (metres) shown on triangles (a) and (b). Vertex velocity magnitudes (m/s) shown on triangles (c) and (d) (not to scale).

B1.3 Input Data

The method used requires initial specification of vertex co-ordinates, vertex velocities and the magnitude of any initial internal or externally applied forces. A constitutive

model needs to be specified and numerical values for the constitutive parameters need to be defined. Boundary conditions also need to be specified and these can take the form of stress boundaries (where the force components acting on the boundary grid points are assumed constant) or velocity boundaries (where the velocity components acting on boundary grid points are assumed constant). In the worked example, the velocities indicated in Figure B1.3 are assumed constant.

B1.4 Calculation of triangle side length

The length of a side of one of the triangles between vertex *i* and vertex *j* of the triangle is found in a straightforward manner using Pythagoras' theorem, given the vertex co-ordinates, (x_i, y_i) and (x_j, y_j).

$$\Delta s^{(i,j)} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{B1.1}$$

The numerical lengths of the sides of the triangles are shown in Table B1.1. The parenthesised superscript identifies the two vertices involved.

Table B1.1: Triangle side lengths and outward unit normals for the worked example.

Side	Length $\Delta s^{(a,b)}$ (m)	Outward normals	
		n_x	n_y
1-2	0.0510	-0.1961	-0.9806
2-3	0.1493	0.9910	-0.1339
3-4	0.0903	0.0775	0.9970
4-1	0.1464	-0.9906	-0.1366
2-4	0.1701	0.9114	0.4116
3-1	-0.8918	-0.8918	0.4524

B1.5 Calculation of triangle side outward unit normals

The outward unit normal vector $\mathbf{n}^{(a,b)}$ of a side of one of the triangles between vertex *a* and vertex *b* can be calculated as follows:

$$\mathbf{n}^{(a,b)} = \begin{pmatrix} n_x^{(a,b)} \\ n_y^{(a,b)} \end{pmatrix} = \frac{1}{\Delta s^{(a,b)}} \begin{pmatrix} y_b - y_a \\ x_a - x_b \end{pmatrix} \tag{B1.2}$$

This assumes that in travelling from vertex *a* to vertex *b* the 'outside' is to the right of the line segment. It also defines an anticlockwise order for the triangle vertices. The values of the outward unit normal vector components for the worked example are also shown in Table B1.1.

In computation of velocity gradients, the products $n_x^{(a,b)} \cdot \Delta s^{(a,b)}$ and $n_y^{(a,b)} \cdot \Delta s^{(a,b)}$ are required. From equations (B1.1) and (B1.2) it can be readily seen that:

$$n_x^{(a,b)} \cdot \Delta s^{(a,b)} = (y_b - y_a)$$
$$n_y^{(a,b)} \cdot \Delta s^{(a,b)} = (x_a - x_b) \qquad \text{(B1.3)}$$

B1.6 Calculation of triangle area

The area of each triangle can be computed from knowledge of the co-ordinates of the triangle vertices:

$$A = \frac{(x_b - x_a)(y_c - y_a) - (x_c - x_a)(y_b - y_a)}{2} \qquad \text{(B1.4)}$$

For the worked example, the triangle areas are given in Table B1.2 below. The longest side lengths are the maximum lengths of sides within a given triangle. The role of both the longest side lengths and the triangle grid point masses, as well as how to compute the latter quantity, will be discussed subsequently in the Section dealing with numerical solution stability.

Table B1.2: Triangle areas, longest side lengths and gridpoint masses for the worked example.

Triangle	Area (m^2)	Longest side length Δs_{max} (m)	Triangle gridpoint mass (kg)
A	0.006455	0.154738	5.93912E+10
B	0.003800	0.154738	1.00887E+11
C	0.006730	0.170074	6.88145E+10
D	0.003525	0.170074	1.31382E+11

B1.7 Calculation of the velocity gradients

Gauss' divergence theorem equates the surface integral of a quantity with the area integral of the gradient of the same quantity. More formally:

$$\int_s \mathbf{n}.f \, ds = \int_A \frac{\partial f}{\partial x_i} dA \qquad \text{(B1.5)}$$

where $\int_s$ is the integral around the boundary of a closed surface and $\int_A$ is the integral over the surface area, A. f can be a scalar, vector or tensor quantity, ds is an incremental arc length of the surface, s and when $i = 1$, $x_i := x$ and when $i = 2$, $x_i := y$. If the average value of the gradient of f is defined as

$$\left\langle \frac{\partial f}{\partial x_i} \right\rangle = \frac{1}{A} \int_A \frac{\partial f}{\partial x_i} dA \qquad \text{(B1.6)}$$

then the finite difference form of (B1.6) is:

$$\left\langle \frac{\partial f}{\partial x_i} \right\rangle = \frac{1}{A} \sum_s \langle f \rangle n_i \, \Delta s \tag{B1.7}$$

where the summation is over all sides of the triangle. When the vertex velocity of each side is substituted for f in equation (B1.7) then the velocity gradient is approximately given by:

$$\frac{\partial u_j}{\partial x_i} \cong \frac{1}{2A} \sum_s \left(u_j^{(a)} + u_j^{(b)} \right) n_i \, \Delta s \tag{B1.8}$$

In equation (B1.8), the parenthesised superscripts of the velocity component terms refer to two adjacent vertices of a triangle. By way of illustration, for the case where $i = 1$ and $j = 1$, (B1.8) becomes:

$$\frac{\partial u_x}{\partial x} = \frac{1}{2A} \left[\left(u_x^{(a)} + u_x^{(b)} \right) n_x^{(a,b)} \Delta s^{(a,b)} + \left(u_x^{(b)} + u_x^{(c)} \right) n_x^{(b,c)} \Delta s^{(b,c)} + \left(u_x^{(c)} + u_x^{(a)} \right) n_x^{(c,a)} \Delta s^{(c,a)} \right] \tag{B1.9}$$

The full set of velocity gradient equations in the scheme are then:

$$\frac{\partial u_x}{\partial x} = \frac{1}{2A} \left[\left(u_x^{(a)} + u_x^{(b)} \right) \left(y_b - y_a \right) + \left(u_x^{(b)} + u_x^{(c)} \right) \left(y_c - y_b \right) + \left(u_x^{(c)} + u_x^{(a)} \right) \left(y_a - y_c \right) \right] \tag{B1.10}$$

$$\frac{\partial u_y}{\partial x} = \frac{1}{2A} \left[\left(u_y^{(a)} + u_y^{(b)} \right) \left(y_b - y_a \right) + \left(u_y^{(b)} + u_y^{(c)} \right) \left(y_c - y_b \right) + \left(u_y^{(c)} + u_y^{(a)} \right) \left(y_a - y_c \right) \right] \tag{B1.11}$$

$$\frac{\partial u_x}{\partial y} = \frac{1}{2A} \left[\left(u_x^{(a)} + u_x^{(b)} \right) \left(x_b - x_a \right) + \left(u_x^{(b)} + u_x^{(c)} \right) \left(x_c - x_b \right) + \left(u_x^{(c)} + u_x^{(a)} \right) \left(x_a - x_c \right) \right] \tag{B1.12}$$

$$\frac{\partial u_y}{\partial y} = \frac{1}{2A} \left[\left(u_y^{(a)} + u_y^{(b)} \right) \left(x_b - x_a \right) + \left(u_y^{(b)} + u_y^{(c)} \right) \left(x_c - x_b \right) + \left(u_y^{(c)} + u_y^{(a)} \right) \left(x_a - x_c \right) \right] \tag{B1.13}$$

which are suitable for coding. The four gradients are computed for each of the four triangles considered in the quadrilateral finite difference zone. Table B1.3 shows the results of applying (B1.10) to (B1.13) to the worked example.

Table B1.3 : Triangle displacement gradients for the worked example

Triangle	$\frac{\partial u_x}{\partial x}$	$\frac{\partial u_y}{\partial x}$	$\frac{\partial u_x}{\partial y}$	$\frac{\partial u_x}{\partial y}$
A	6.13245e-5	-1.33385e-6	8.45856e-6	-1.71495e-5
B	1.06326e-4	-3.23684e-6	-1.43684e-5	-1.61842e-5
C	6.00357e-5	-1.27935e-6	-1.811293e-6	-1.64487e-5
D	1.12298e-4	-1.74468e-5	0.00000e0	6.00000e-6

B1.8 Calculation of the strain rates

The strain rate components are defined

$$\dot{e}_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right] \tag{B1.14}$$

and thus can be computed using the velocity gradients found from the previous Section. Explicitly:

$$\dot{e}_{xx} = \frac{\partial u_x}{\partial x} \tag{B1.15}$$

$$\dot{e}_{yy} = \frac{\partial u_y}{\partial y} \tag{B1.16}$$

and

$$\dot{e}_{xy} = \dot{e}_{yx} = \frac{1}{2}\left[\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right]. \tag{B1.17}$$

Equation (B1.17) provides a useful internal computational check that the velocity gradients have been computed correctly. Equations (B1.15) to (B1.17) are applied to each triangle in turn, using the velocity gradients computed in the previous Section. Table B1.4 shows the strain rates computed for the worked example.

Table B1.4 : Strain rate components for each of the triangles in the worked example before the mixed discretisation procedure is applied.

Triangle	$\dot{e}_{xx}$	$\dot{e}_{yy}$	$\dot{e}_{zz}$	$\dot{e}_{xy}$	$\dot{e}_{vol}$
A	6.13245e-5	-1.71495e-05	0.00000e0	3.5236e-6	4.41751e-5
B	1.06326e-4	-1.61842e-5	0.00000e0	-8.80263e-6	9.01421e-5
C	6.00357e-5	-1.64487e-5	0.00000e0	-4.69614e-6	4.35869e-5
D	1.12298e-4	-1.74468e-5	0.00000e0	6.00000e-6	9.48511e-5

B1.9 Mixed discretisation adjustment for strain rates

Historically, some finite difference codes using quadrilateral elements or zones have suffered problems. Marti and Cundall (1982) state that the overlaid triangles scheme eliminates the problem of 'hourglass deformations', through the use of a 'mixed discretisation' scheme of gridding. In two-dimensions, four sets of strain rate components (one for each triangle in a zone) are used in application of their method, each set representative of the state of strain within the same zone. A second historical problem with finite difference codes called 'mesh locking' can be overcome by

treating the isotropic and deviatoric components of strain rates and stress separately. Isotropic stresses and strain rate components are taken as constant over the whole quadrilateral zone (all four triangular elements), the deviatoric components remain unchanged and are specific to each triangular element. Such a scheme is called 'mixed discretisation', a term first coined by Marti and Cundall (1982), and has now become used more widely, see for example, Konagai and Johansson, (2001).

Brief consideration of the figures in Table B1.4 show that the volumetric strain rate,

$$\dot{e}_{vol} = \dot{e}_{xx} + \dot{e}_{yy} \tag{B1.18}$$

(the isotropic component) is not the same for each of the triangles. The mixed discretisation adjustment takes these strain rate components and re-assigns them such that the isotropic component is the same for every triangle. The deviatoric strain rate,

$$\dot{e}_{d} = \dot{e}_{xx} - \dot{e}_{yy} \tag{B1.19}$$

remains the same for each triangle. The triangle pairs are treated separately. The new isotropic strain rate is an area weighted average of the initial isotropic components of each the respective triangles. For triangles A and B (denoted with parenthesised superscripts):

$$\langle \dot{e}_{vol} \rangle = \frac{A^{(A)}\left(\dot{e}_{xx}^{(A)} + \dot{e}_{yy}^{(A)}\right) + A^{(B)}\left(\dot{e}_{xx}^{(B)} + \dot{e}_{yy}^{(B)}\right)}{A^{(A)} + A^{(B)}}. \tag{B1.20}$$

Then:

$$\dot{e}_{xx}^{(A)} := \frac{\langle \dot{e}_{vol} \rangle + e_{d}^{(A)}}{2} \tag{B1.21}$$

$$\dot{e}_{yy}^{(A)} := \frac{\langle \dot{e}_{vol} \rangle - e_{d}^{(A)}}{2} \tag{B1.22}$$

and,

$$\dot{e}_{xx}^{(B)} := \frac{\langle \dot{e}_{vol} \rangle + e_{d}^{(B)}}{2} \tag{B1.23}$$

$$\dot{e}_{yy}^{(B)} := \frac{\langle \dot{e}_{vol} \rangle - e_{d}^{(B)}}{2}. \tag{B1.24}$$

For triangles C and D replace superscript *(A)* with *(C)* and replace superscript *(B)* with *(D)*. After the mixed discretisation process is applied to the strain rate components, the values carried forward into the constitutive model are shown in Table B1.5 for the worked example.

Table B1.5: Strain rate components for each of the triangles in the worked example after the mixed discretisation procedure is applied.

Triangle	$\dot{e}_{xx}$	$\dot{e}_{yy}$	$\dot{e}_{zz}$	$\dot{e}_{xy}$	$\dot{e}_{vol}$
A	6.98411e-5	-8.63293e-06	0.00000e0	3.5236e-6	6.12082e-5
B	9.18594e-5	-3.06512e-5	0.00000e0	-8.80263e-6	6.12082e-5
C	6.88463e-5	-7.63810e-6	0.00000e0	-4.69614e-6	6.12082e-5
D	9.54764e-5	-3.42682e-5	0.00000e0	6.00000e-6	6.12082e-5

B1.10 Calculation of new stress tensor components

The next step in the calculation is to compute the new stress tensor components using the initial stress tensor, adjusted strain rates and the constitutive law defined for the material under investigation. The body is assumed stress free initially.

Calculation of new stress tensor components is conducted four times per zone per time step, that is, once for each triangle in the quadrilateral finite difference zone. Firstly, the stress tensor is updated to account for contributions due to finite rotations of the quadrilateral zone.

$$\sigma_{ij} := \sigma_{ij} + \Delta\sigma_{ij}^{(rot)} \tag{B1.25}$$

where:

$$\Delta\sigma_{ij}^{(rot)} = \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj} \tag{B1.26}$$

and

$$\omega_{ij} = \begin{bmatrix} 0 & \frac{1}{2}\left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x}\right) & 0 \end{bmatrix}. \tag{B1.27}$$

In the case of the worked example, the body under study is assumed to be initially stress free, thus all components of σ_{ij} and thus $\Delta\sigma_{ij}^{(rot)}$ will be zero. In a second step, the stress tensor is updated to account for contributions due to straining of the material via the constitutive model.

$$\sigma_{ij} := \sigma_{ij} + \Delta\sigma_{ij}^{(cm)} \tag{B1.28}$$

The constitutive model has an incremental formulation, that is, increments in stress tensor components, $\Delta\sigma_{ij}^{(cm)}$, primarily depend on increments in strain tensor components, Δe_{ij}, although optionally the model can be state, (σ_{ij}, e_{ij}), and history dependent. κ is an optional history parameter.

$$\Delta\sigma_{ij}^{(cm)} = M\left(\Delta e_{ij}, \sigma_{ij}, e_{ij}, \kappa\right) \tag{B1.29}$$

Strain tensor component increments are found through the product of the strain rate components and the time step used in the simulation, that is:

$$\dot{e}_{ij}\Delta t = \Delta e_{ij} . \tag{B1.30}$$

For a unit time step, $\Delta t = 1$ and (B1.30) becomes:

$$\dot{e}_{ij} = \Delta e_{ij} . \tag{B1.31}$$

The assumption of a unit time step has important ramifications for the damping scheme applied within the numerical model that will be discussed subsequently. As, at present, the discussion is confined to isotropic linear elasticity the following equations represent the constitutive model and can be used to determine the stress component increments,

$$\begin{aligned}
\Delta\sigma_{xx} &= \lambda_1\dot{e}_{xx} + \lambda_2\dot{e}_{yy} + \lambda_2\dot{e}_{zz} \\
\Delta\sigma_{yy} &= \lambda_2\dot{e}_{xx} + \lambda_1\dot{e}_{yy} + \lambda_2\dot{e}_{zz} \\
\Delta\sigma_{zz} &= \lambda_2\dot{e}_{xx} + \lambda_2\dot{e}_{yy} + \lambda_1\dot{e}_{zz} \\
\Delta\sigma_{xy} &= 2G\dot{e}_{xy}
\end{aligned} \tag{B1.32}$$

where λ_1 and λ_2 are Lamé's constants and G is the shear modulus of the material. For the worked example, Table B1.6 below shows the values of the stress component increments (these also being the values of the new stress components, for this first time step).

Table B1.6 : New stress components for each of the triangles found using rotation adjustment and constitutive model contributions for the worked example.

Triangle	σ_{xx}	σ_{yy}	σ_{zz}	σ_{xy}	σ_0
A	3.21629e6	7.05118e5	9.81371e5	1.13995e5	1.96070e6
B	3.92087e6	5.34073e2	9.81371e5	-2.81684e5	1.96070e6
C	3.18445e6	7.36952e5	9.81371e5	-1.50276e5	1.96070e6
D	4.03662e6	-1.15212e5	9.81371e5	1.92000e5	1.96070e6

B1.11 Mixed discretisation for stress components

Table B1.6 also lists the mean normal stress in the *xy* plane:

$$\sigma_0 = \frac{\sigma_{xx} + \sigma_{yy}}{2}, \tag{B1.33}$$

which can be seen to be equal for each triangle. This is because in the specific case of this worked example, the contribution to the stress tensor due to rotation of the zone is

nil and the strain rate components used to compute the straining contributions have already be subjected to mixed discretisation. More generally, with non-zero contributions due to rotation, σ_0 will not be equal across all of the triangles in the quadrilateral and the mixed discretisation process needs to be invoked again, but this time on the stress tensor components. The overall procedure is the same however minor distinctions arise due to differences in the nature of volumetric strain and hydrostatic stress. Again, using parenthesised superscripts to denote the triangle concerned, the average, mean normal stress is the *xy* plane is found from:

$$\langle \sigma_0 \rangle = \frac{\frac{A^{(A)}}{2}\left(\sigma_{xx}^{(A)} + \sigma_{yy}^{(A)}\right) + \frac{A^{(B)}}{2}\left(\sigma_{xx}^{(B)} + \sigma_{yy}^{(B)}\right)}{A^{(A)} + A^{(B)}}. \tag{B1.34}$$

Then:

$$\sigma_{xx}^{(A)} := \langle \sigma_0 \rangle + \frac{\sigma_d^{(A)}}{2} \tag{B1.35}$$

$$\sigma_{yy}^{(A)} := \langle \sigma_0 \rangle - \frac{\sigma_d^{(A)}}{2} \tag{B1.36}$$

$$\sigma_{xx}^{(B)} := \langle \sigma_0 \rangle + \frac{\sigma_d^{(B)}}{2} \tag{B1.37}$$

$$\sigma_{yy}^{(B)} := \langle \sigma_0 \rangle + \frac{\sigma_d^{(B)}}{2} \tag{B1.38}$$

where:

$$\sigma_d^{(A)} = \sigma_{xx}^{(A)} - \sigma_{yy}^{(A)} \tag{B1.39}$$

for triangle *A* and similarly for triangle *B*. Values for the stress components after the mixed discretisation procedure are not given here for the worked example as they are identical to the values before the mixed discretisation procedure, as explained earlier.

B1.12 Computation of nodal forces

With the stress components applying within each triangle found, the equivalent system of forces acting at the vertices of the triangle can be determined. For a triangle with vertices labelled *a*, *b* and *c*, the equivalent force components acting at vertex *a* are found using:

$$F_i = \frac{1}{2}\sigma_{ij}\left(n_j^{(c,a)}\Delta s^{(c,a)} + n_j^{(a,b)}\Delta s^{(a,b)}\right) \tag{B1.40}$$

At vertex *a*, equation (B1.40) expands to:

$$F_x = \frac{1}{2}\sigma_{xx}\left(n_x^{(c,a)}\Delta s^{(c,a)} + n_x^{(a,b)}\Delta s^{(a,b)}\right) + \frac{1}{2}\sigma_{xy}\left(n_y^{(c,a)}\Delta s^{(c,a)} + n_y^{(a,b)}\Delta s^{(a,b)}\right) \quad \text{(B1.41)}$$

$$F_y = \frac{1}{2}\sigma_{yx}\left(n_x^{(c,a)}\Delta s^{(c,a)} + n_x^{(a,b)}\Delta s^{(a,b)}\right) + \frac{1}{2}\sigma_{yy}\left(n_y^{(c,a)}\Delta s^{(c,a)} + n_y^{(a,b)}\Delta s^{(a,b)}\right) \quad \text{(B1.42)}$$

Equations (B1.41) and (B1.42) are evaluated at each vertex of a triangle and the whole process is repeated for each of the four triangles. For each of the overlaying pairs of triangles, the force components acting at a given vertex are summed up. The results of this process for the worked example are shown in Table B1.7 below. The force components at vertices arising from each set of overlaid pair of triangles are then averaged.

Table B1.7: Sums of force components acting at vertices of each overlaid pair of triangles for the worked example (units are Newtons).

Triangles A and B			Triangles C and D		
Vertex	F_x	F_y	Vertex	F_x	F_y
A	-3.09348e5	-1.12793e4	A	-3.19558e5	-1.08476e4
B	2.80399e5	-1.94549e4	B	2.90192e5	-1.98690e4
C	2.46883e5	1.39208e4	C	2.41535e5	1.41469e4
D	-2.17934e5	1.68134e4	D	-2.12169e5	1.65697e4

B1.13 Computation of new grid point velocities and positions

The new grid point velocities are found by applying Newton's second law of motion to each vertex of the quadrilateral. The equation, including artificial damping to expedite the solution to static problems, is:

$$u_i^{(t+\Delta t/2)} = u_i^{(t-\Delta t/2)} + \left(U_i^{(t)} - D_i^{(t)}\right)\frac{\Delta t}{m_n} \quad \text{(B1.43)}$$

where $U_i^{(t)}$ is the sum of vertex forces arising from all quadrilaterals surrounding the vertex, vertex forces arising from applied loads and body forces due to gravity; m_n is the so-called grid point mass; Δt is the time increment. Where velocity boundary conditions are in force, equation (B1.43) is not invoked as the velocity is unchanging. $D_i^{(t)}$ is a damping force given by:

$$D_i^{(t)} = \alpha\left|U_i^{(t)}\right|\mathrm{sgn}\left(u_i^{(t-\Delta t/2)}\right) \quad \text{(B1.44)}$$

where $\alpha = 0.8$ and

$$\mathrm{sgn}(x)=\begin{cases}1; x>0\\ -1; x<0\\ 0; x=0\end{cases} \tag{B1.45}$$

New co-ordinates for grid points can be determined when the new velocity has been determined:

$$x_i^{(t+\Delta t)} = x_i^{(t)} + u_i^{(t+\Delta t/2)}\Delta t \tag{B1.46}$$

For the conditions specified in the worked example, velocities at each of the quadrilateral vertices are prescribed. No changes to the velocities result.

B1.14 Ensuring solution stability

Solution stability is ensured by adjusting the grid point mass used Newton's second law of motion, given that $\Delta t = 1$. The key constraint for solution stability is that speed of propagation of computed information within the model cannot be greater than the speed at which physical information such as force or velocity can propagate within the medium the model represents. The p-wave velocity applying to the medium is representative of this threshold speed.

$$C_p = \sqrt{\frac{K+4G/3}{\rho}}. \tag{B1.47}$$

The speed of propagation of computed information within the model must be greater than this. It depends on the spatial and temporal discretisation, that is:

$$C = \frac{\Delta x}{\Delta t} \tag{B1.48}$$

Suppose that the allowable propagation distance is (such that equality holds in (B1.48)):

$$\Delta x = \frac{A}{\Delta s_{\max}} \tag{B1.49}$$

where $\Delta s_{\max}$ is the length of the longest side of a triangle of area A (see Table B1.2), then:

$$\Delta t = \frac{A}{C_p \Delta s_{\max}}. \tag{B1.50}$$

For a unit time step, $\Delta t = 1$,

$$\sqrt{\frac{K+4G/3}{\rho}} = \frac{A}{\Delta s_{\max}} \tag{B1.51}$$

or,

$$\rho A = \frac{(K + 4G/3)}{A} \Delta s_{\max}^2 . \qquad \text{(B1.52)}$$

One third of the triangle mass, ρA, is taken to act at any one of the triangle's vertices. Summing the contributions of all triangles connected to a given node on the finite difference grid results in the following expression for the grid point mass that ensures numerical stability under the assumption that $\Delta t = 1$.

$$m_n = \sum \frac{(K + 4G/3)}{2 \cdot 3A} \Delta s_{\max}^2 \qquad \text{(B1.53).}$$

The factor 2 in the denominator of (B1.53) accounts for the fact that each quadrilateral has two overlaid sets of triangles.

For the worked example problem, the grid point masses found according to (B1.53) and used in Newton's second law of motion (B1.43) are shown in Table B1.8 below.

Table B1.8: Vertex grid point masses for the worked example

Grid point	Connected to triangle	m_n	$1/m_n$
A	A,B,D	1.4583e11	6.85730e-12
B	B,C,D	1.50542e11	6.64268e-12
C	A,B,C	1.14516e11	8.73010e-12
D	A,C,D	1.29794e11	7.70453e-12

B2 *FLAC* commands data file for single zone verification example

```
;
;          Single zone triaxial test simulation
;
title
Triaxial test simulation - elastic model
;
set large
set gravity 0 0
grid 1 1
;
gen 0.000,0.000 -0.020,0.145 0.070,0.138 0.050,-0.010

model elastic
prop den 2500 bulk 26.66667e9 shear 16e9

def initialise
          myizones = izones
          myjzones = jzones
          myigp = igp
          myjgp = jgp
          halfizones = myizones / 2
          halfigp = igp/2
          halfjgp = jgp/2
          halfizonesplus1 = halfizones + 1
end
initialise

call elastic.fis
opt m_elastic

model elastic i = 1,halfizones
prop den 2500 bulk 26.7e9 shear 16e9 i=1,halfizones

model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k 26.7e9 m_g 16e9    i= halfizonesplus1 ,myizones

ini xv -2.46e-6 i=1
ini xv 3.0e-6 i=myigp
ini yv 0 j=1
ini yv -2.46e-6 j=myjgp

; The following commands fix the velocity of the selected gridpoints
fix x i=1
fix x i=myigp
fix y j=1
fix y j=myjgp

;step 1
```

Appendix B2: FLAC commands data file for single zone verification example

```
; Utilities to interrogate model
def getforces
        $xf1=xforce(1,1)
        $xf2=xforce(2,1)
        $xf3=xforce(2,2)
        $xf4=xforce(1,2)

        $yf1=yforce(1,1)
        $yf2=yforce(2,1)
        $yf3=yforce(2,2)
        $yf4=yforce(1,2)
end

def getvelocities
        $xv1=xvel(1,1)
        $xv2=xvel(2,1)
        $xv3=xvel(2,2)
        $xv4=xvel(1,2)

        $yv1=yvel(1,1)
        $yv2=yvel(2,1)
        $yv3=yvel(2,2)
        $yv4=yvel(1,2)
end
```

B3 Spreadsheet calculations for single zone verification example

Materials presented in this section support the discussion of Appendix B1. Spreadsheet calculations are presented for reference for the reader should he/she wish to reproduce the calculation sequence. The precision of figures presented herein is greater than would be required for formal reporting in order that the reader can use the reported figures to check their own implementation of the code – if required. The results should agree exactly (to the last decimal place) with those presented herein.

B3.1 Input data defining zone

Coordinates of zone (t)		
Vertex x	y	
1	0	0
2	0.05	-0.01
3	0.07	0.138
4	-0.02	0.145

Velocities at vertices (t-dt/2)		
Vertex Ux	Uy	
1	-2.46E-06	0
2	3.00E-06	0
3	3.00E-06	-2.46E-06
4	-2.46E-06	-2.46E-06

Forces at vertices (t)		
Vertex Fx	Fy	
1	0	0
2	0	0
3	0	0
4	0	0

B3.2 Triangle Geometry

Triangle (a)

Position		
Vertex	x	y
1	0	0
3	0.07	0.138
4	-0.02	0.145
Velocity		
	ux	uy
1	-0.00000246	0
3	0.000003	-0.00000246
4	-0.00000246	-0.00000246
Side outward normals		
	nx	ny
1-3	0.891827226	-0.452376157
3-4	0.077543572	0.996988952
4-1	-0.99062109	-0.13663739
Side lengths		
1-3	0.154738486	
3-4	0.090271808	
4-1	0.14637281	
Longe:	0.154738486	
Area	0.006455	m^2
Mass	16.1374989	kg/m
GPM	59391178081	

Triangle (b)

Position		
Vertex	x	y
1	0	0
2	0.05	-0.01
3	0.07	0.138
Velocity		
	ux	uy
1	-0.00000246	0
2	0.000003	0
3	0.000003	-0.00000246
Side outward normals		
	nx	ny
1-2	-0.196116135	-0.980580688
2-3	0.990992486	-0.133917898
3-1	-0.891827226	0.452376157
Side lengths		
1-2	0.050990194	
2-3	0.149345234	
3-1	0.154738486	
Longe:	0.154738486	
Area	0.003800	
Mass	9.50000016	kg/m
GPM	1.00887E+11	

Triangle (c)

Position		
Vertex	x	y
2	0.05	-0.01
3	0.07	0.138
4	-0.02	0.145
Velocity		
	ux	uy
2	0.000003	0
3	0.000003	-0.00000246
4	-0.00000246	-0.00000246
Side outward normals		
	nx	ny
2-3	0.990992486	-0.133917898
3-4	0.077543572	0.996988952
4-2	-0.911370635	-0.411586732
Side lengths		
2-3	0.149345234	
3-4	0.090271808	
4-2	0.170073509	
Longe:	0.170073509	
Area	0.006730	
Mass	16.82499889	kg/m
GPM	68814471908	

Triangle (d)

Position		
Vertex	x	y
1	0	0
2	0.05	-0.01
4	-0.02	0.145
Velocity		
	ux	uy
1	-0.00000246	0
2	0.000003	0
4	-0.00000246	-0.00000246
Side outward normals		
	nx	ny
1-2	-0.196116135	-0.980580688
2-4	0.911370635	0.411586732
4-1	-0.99062109	-0.13663739
Side lengths		
1-2	0.050990194	
2-4	0.170073509	
4-1	0.14637281	
Longe:	0.170073509	
Area	0.003525	
Mass	8.812499582	kg/m
GPM	1.31382E+11	

B3.3 Strain Increments

Triangle (a)

Velocity gradients

dux/dx	6.13246E-05
duy/dx	-1.33385E-06
dux/dy	8.45856E-06
duy/dy	-1.71495E-05

Strain rates

		Checks
exx	6.13246E-05	4.4175E-05
eyy	-1.71495E-05	7.8474E-05
ezz	0	
exy	3.56236E-06	

Mixed discretisation procedure

em	6.12082E-05
ed	7.84741E-05
exx	6.9841E-05
eyy	-8.6329E-06
ezz	0.0000E+00
exy	3.5624E-06

Triangle (b)

Velocity gradients

dux/dx	0.000106326
duy/dx	-3.23684E-06
dux/dy	-1.43684E-05
duy/dy	-1.61842E-05

		Checks
exx	1.06326E-04	9.0142E-05
eyy	-1.61842E-05	1.2251E-04
ezz	0	
exy	-8.80263E-06	

ed	1.22511E-04
exx	9.1859E-05
eyy	-3.0651E-05
ezz	0.0000E+00
exy	-8.8026E-06

Triangle (c)

Velocity gradients

dux/dx	6.00357E-05
duy/dx	-1.27935E-06
dux/dy	-8.11293E-06
duy/dy	-1.64487E-05

		Checks
exx	6.00357E-05	4.3587E-05
eyy	-1.64487E-05	7.6484E-05
ezz	0	
exy	-4.69614E-06	

em	6.12082E-05
ed	7.64844E-05
exx	6.8846E-05
eyy	-7.6381E-06
ezz	0.0000E+00
exy	-4.6961E-06

Triangle (d)

Velocity gradients

dux/dx	0.000112298
duy/dx	-3.48936E-06
dux/dy	1.54894E-05
duy/dy	-1.74468E-05

		Checks
exx	1.12298E-04	9.4851E-05
eyy	-1.74468E-05	1.2974E-04
ezz	0	
exy	6.00000E-06	

ed	1.29745E-04
exx	9.5476E-05
eyy	-3.4268E-05
ezz	0.0000E+00
exy	6.0000E-06

B3.4 Stress Increments

Triangle (a)

dsxx	3.216E+06	
dsyy	7.051E+05	
dszz	9.814E+05	
dsxy	1.140E+05	
New Stress components		
		Checks
sxx	3216287.365	1960702.461
syy	705117.5574	2511169.808
szz	981371.3672	
sxy	113995.3602	
Mixed discretisation procedure		
Isotropic stress		
so	1960702.461	
so	1960702.461	
sd	2511169.808	
		Checks
sxx	3216287.365	1960702.461
syy	705117.5574	2511169.808
szz	981371.3672	
sxy	113995.3602	
Stress tensor		
	3216287	113995
	113995	705118

Triangle (b)

dsxx	3.921E+06	
dsyy	5.341E+02	
dszz	9.814E+05	
dsxy	-2.817E+05	
New Stress components		
		Checks
sxx	3920870.849	1960702.461
syy	534.0732152	3920336.776
szz	981371.3672	
sxy	-281684.2058	
so	1960702.461	
sd	3920336.776	
		Checks
sxx	3920870.849	1960702.461
syy	534.0732152	3920336.776
szz	981371.3672	
sxy	-281684.2058	
Stress tensor		
	3920871	-281684
	-281684	534

Triangle (c)

dsxx	3.184E+06	
dsyy	7.370E+05	
dszz	9.814E+05	
dsxy	-1.503E+05	
New Stress components		
		Checks
sxx	3184452.958	1960702.506
syy	736952.0537	2447500.904
szz	981371.3894	
sxy	-150276.3843	
so	1960702.506	
so	1960702.506	
sd	2447500.904	
		Checks
sxx	3184452.958	1960702.506
syy	736952.0537	2447500.904
szz	981371.3894	
sxy	-150276.3843	
Stress tensor		
	3184453	-150276
	-150276	736952

Triangle (d)

dsxx	4.037E+06	
dsyy	-1.152E+05	
dszz	9.814E+05	
dsxy	1.920E+05	
New Stress components		
		Checks
sxx	4036617.498	1960702.506
syy	-115212.4864	4151829.984
szz	981371.3894	
sxy	192000.0091	
so	1960702.506	
sd	4151829.984	
		Checks
sxx	4036617.498	1960702.506
syy	-115212.4864	4151829.984
szz	981371.3894	
sxy	192000.0091	
Stress tensor		
	4036617	192000
	192000	-115212

B3.5 Nodal forces

Triangle (a)

Vertex	Fx	Fy
1	-16386.79391	-32129.27382
3	234320.7759	15315.83664
4	-217933.9814	16813.43386

Triangle (b)

Vertex	Fx	Fy
1	-292961.2739	20849.97115
2	280399.0456	-19454.90347
3	12562.26907	-1395.07065

Triangle (c)

Vertex	Fx	Fy
2	-4383.143748	-32636.87513
3	241535.4363	14146.90025
4	-237152.2882	18489.97166

Triangle (d)

Vertex	Fx	Fy
1	-319557.8408	-10847.56305
2	294574.7734	12767.87601
4	24983.10702	-1920.311266

Quadrilateral 1

Nodal Forces - Triangle set 1

Vertex	Fx	Fy
1	-309348.0625	-11279.30273
2	280399.0313	-19454.9043
3	246883.0469	13920.7666
4	-217933.9844	16813.43359

Quadrilateral 2

Nodal Forces - Triangle set 2

Vertex	Fx	Fy
1	-319557.8438	-10847.56348
2	290191.625	-19868.99805
3	241535.4375	14146.90039
4	-212169.1719	16569.66016

Nodal Forces supplied by the material

Vertex	Fx	Fy
1	-3.145E+05	-1.106E+04
2	2.853E+05	-1.966E+04
3	2.442E+05	1.403E+04
4	-2.151E+05	1.669E+04

Nodal forces acting on the nodes

Vertex	Fx	Fy
1	3.1445E+05	1.1063E+04
2	-2.8530E+05	1.9662E+04
3	-2.4421E+05	-1.4034E+04
4	2.1505E+05	-1.6692E+04

B3.6 New velocities and corresponding displacements

External Forces acting on Nodes		
Vertex	Rx	Ry
1	0.000E+00	-1.106E+04
2	0.000E+00	-1.966E+04
3	0.000E+00	0.000E+00
4	0.000E+00	0.000E+00

Dampi alpha		0
Vertex	Fd(x)	Fd(y)
1	0.000E+00	0.000E+00
2	0.000E+00	0.000E+00
3	0.000E+00	0.000E+00
4	0.000E+00	0.000E+00

Accelerating forces on each node		
Vertex	Fx'	Fy'
1	3.145E+05	0.000E+00
2	-2.853E+05	0.000E+00
3	-2.442E+05	-1.403E+04
4	2.151E+05	-1.669E+04

Quadrilateral Gridpoint Mass		
Vertex	fgpm	1/fgpm
1	1.4583E+11	6.857E-12
2	1.50542E+11	6.643E-12
3	1.14546E+11	8.730E-12
4	1.29794E+11	7.705E-12

Old Velocities on Each Node		
Vertex	Ux	Uy
1	-0.00000246	0
2	0.000003	0
3	0.000003	-0.00000246
4	-0.00000246	-0.00000246

New velocities		
Vertex	Ux	Uy
1	-3.0370E-07	0.0000E+00
2	1.1049E-06	0.0000E+00
3	8.6803E-07	-2.5825E-06
4	-8.0313E-07	-2.5886E-06

Sign of Old velocities		
Vertex	sgn(Ux)	sgn(Uy)
1	-1	0
2	1	0
3	1	-1
4	-1	-1

Displacements		
1	-3.0370E-07	0.0000E+00
2	1.1049E-06	0.0000E+00
3	8.6803E-07	-2.5825E-06
4	-8.0313E-07	-2.5886E-06

B4 *FLAC* data files for verification Test Types 1,2 and 3

B4.1 Test Type 1, Case 1

```
;
; UDM test simulations
;
title
UDM Test Type 1 - elastic UDM, Case 1 : confined compression

set large
set gravity 0 0
grid 1 1
;
; Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic
prop den 2500 bulk 26e9 shear 16e9

def getKandG
  YoungsMod = 40e9
  Poissons = 0.25

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
end
initialise


model elas i = 1,halfizones
prop den 2500 bulk BulkMod shear ShearMod i=1,halfizones

call elastic.fis

model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones

ini xv 0 i=1
ini xv 0 i=myigp
ini yv 0 j=1
ini yv -2e-6 j=myjgp

; The following commands fix the velocity of the selected gridpoints
fix x i=1
```

```
fix x i=myigp
fix y j=1
fix y j=myjgp

call monitor.dat

his nstep 1
his unbal
his ex
his ey
his exy
his sxx i=1 j=1
his syy i=1 j=1
his szz i=1 j=1
his sxy i=1 j=1

step 200

;set plot po 0 0 0.55 0.55 0
;set output t1-pc-m.ps
;plot pen model grid fix vel
;set output t1-pc-r1.ps
;plot pen hist 5 6 7 8 vs 3
```

B4.2 Test Type 1, Case 2

```
;
; UDM test simulations
;
title
UDM Test Type 1 - elastic UDM, Case 2: shear along model top

set large
set gravity 0 0
grid 1 1
;
; Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic
prop den 2500 bulk 26e9 shear 16e9

def getKandG
  YoungsMod = 40e9
  Poissons = 0.25

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfigp = igp/2
  halfjgp = jgp/2
```

```
  halfizonesplus1 = halfizones + 1
end
initialise


model elas i = 1,halfizones
prop den 2500 bulk BulkMod shear ShearMod i=1,halfizones

call elastic.fis

model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones

ini xv 0
ini yv 0
ini xv 1e-6 j=myjgp

; The following commands fix the velocity of the selected gridpoints
fix x i=1
fix x i=myigp
fix y j=1
fix y j=myjgp

call monitor.dat

his nstep 1
his unbal
his ex
his ey
his exy
his sxx i=1 j=1
his syy i=1 j=1
his szz i=1 j=1
his sxy i=1 j=1

step 200

;set plot po 0 0 0.55 0.55 0
;set output t1-ps-m.ps
;plot pen model grid fix vel
;set output t1-ps-r1.ps
;plot pen hist 8 vs 4
```

B4.3 Test Type 1, Case 3

```
;
;  UDM test simulations
;
title
UDM Test Type 1 - elastic UDM, Case 3 : Case 1 & 2 combined

set large
set gravity 0 0
grid 1 1
;
;  Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic
```

```
prop den 2500 bulk 26e9 shear 16e9

def getKandG
  YoungsMod = 40e9
  Poissons = 0.25

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
end
initialise


model elas i = 1,halfizones
prop den 2500 bulk BulkMod shear ShearMod i=1,halfizones

call elastic.fis

model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones

ini xv 0 j=1
ini xv 1e-6 j=myigp
ini yv 0 j=1
ini yv -2e-6 j=myjgp

; The following commands fix the velocity of the selected gridpoints
fix x i=1
fix x i=myigp
fix y j=1
fix y j=myjgp

call monitor.dat

his nstep 1
his unbal
his ex
his ey
his exy
his sxx i=1 j=1
his syy i=1 j=1
his szz i=1 j=1
his sxy i=1 j=1

step 200

set plot po 0 0 0.55 0.55 0
set output t1-cl-m.ps
```

```
plot pen model grid fix vel
set output t1-cl-r1.ps
plot pen hist 5 6 7 8 vs 3
set output t1-cl-r2.ps
plot pen hist 8 vs 4
```

B4.4 Test Type 2 (Main routine)

```
;
; UDM test simulations
;
title
UDM Test Type 2 - elastic UDM

set large
set gravity 0 0
grid 10 10
;
; Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic

def getKandG
  YoungsMod = 40e9
  Poissons = 0.25

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

prop den 2500 bulk BulkMod shear ShearMod

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfjzones = myjzones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
  halfjzonesplus1 = halfjzones + 1
end
initialise

model elas i = 1,halfizones
prop den 2500 bulk BulkMod shear ShearMod i=1,halfizones

call elastic.fis
model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones

ini xv 0 j=1
ini xv 0 j=myigp
ini yv 0 j=1
ini yv -1e-8 j=myjgp
```

```
ini sxx = -10e6
ini syy = -10e6
ini szz = -10e6

; The following commands fix the velocity of the selected gridpoints
fix y j=1
fix y j=myjgp
;
apply sxx -10e6 i=1 j=1,myjgp
apply sxx -10e6 i=myigp j=1,myjgp
;
;
call monitor.dat

def mdsxx
  $mdsxx =sxx(3,6) - sxx(8,6)
  if $mdsxx > maxmdsxx then
  maxmdsxx = $mdsxx
  end_if
  mdsxx=$mdsxx
end

def mdsyy
  $mdsyy = syy(3,6) - syy(8,6)
  if abs($mdsyy) > maxmdsyy then
  maxmdsyy = $mdsyy
  end_if
  mdsyy=$mdsyy
end

def mdszz
  $mdszz = szz(3,6) - szz(8,6)
  if abs($mdszz) > maxmdszz then
  maxmdszz = $mdszz
  end_if
  mdszz=$mdszz
end

def mdsxy
  $mdsxy = sxy(3,6) + sxy(8,6)
  if abs($mdsxy) > maxmdsxy then
  maxmdsxy = $mdsxy
  end_if
  mdsxy=$mdsxy
end


call servo.dat
set high_unbal=50e3
set low_unbal=25e3

his nstep 1
his unbal
his ey
his sigmay
his yv i=halfigp j = myjgp
his sxx i=3 j=6
his sxx i=8 j=6
his syy i=3 j=6
```

```
his syy i=8 j=6
his szz i=3 j=6
his szz i=8 j=6
his sxy i=3 j=6
his sxy i=8 j=6
his mdsxx
his mdsyy
his mdszz
his mdsxy

step 500

print maxmdsxx maxmdsyy maxmdszz maxmdsxy

set plot po 0 0 0.55 0.55 0
;
; Produce a plot of the model
;
;set output t2m.ps
;plot pen model grid fix app
;
; Produce a plot of the normal stress component histories
;
;set output t2rsign.ps
;plot pen hist 5 6 7 8 9 10
;
; Produce a plot of the shear stress component histories
;
;set output t2rsigs.ps
;plot pen hist 11 12
;
; Produce a plot of differences between stress component histories
;
;set output t2rsdf.ps
;plot pen hist 13 14 15 16

print maxmdsxx maxmdsyy maxmdszz maxmdsxy
```

B4.5 Test Type 2 (Auxiliary file 'Servo.dat')

```
; Fish funcion to adjust velocity such that
; unbalanced force is below high_unbal
; (and above low_unbal) to ensure small
; inertial effects.
def servo
  while_stepping
  if unbal>high_unbal then
    loop i (1,igp)
      yvel(i,jgp)=yvel(i,jgp)*0.975
      yvel(i,1)=yvel(i,1)*0.975
    end_loop
  end_if
  if unbal<low_unbal then
    loop i (1,igp)
      yvel(i,jgp)=yvel(i,jgp)*1.025
      yvel(i,1)=yvel(i,1)*1.025
    end_loop
  end_if
end
```

B4.6 Test Type 2 (Auxiliary file 'Monitor.dat')

```
set echo off
;
; Quantities monitored during stepping
;
;
; Strain in the y direction monitored on the RHS of the model
;
def ey
  ey=(ydisp(igp,jgp)-ydisp(igp,1))/(y(igp,jgp)-y(igp,1))
end
;
; Strain in x direction monitored at the top of the model.
;
def ex
  ex=(xdisp(igp,jgp)-xdisp(1,jgp))/(x(igp,jgp)-x(1,jgp))
end
;
; Shear strain, monitored on the RHS of the model
;
def exy
  eyx1=(xdisp(igp,jgp)-xdisp(igp,1))
  eyx1=eyx1/(y(igp,jgp)+ydisp(igp,jgp)-y(igp,1)-ydisp(igp,1))
  exy = 0.5*atan(eyx1)
end
;
; This works out the stress in the y direction on top of the model
;
def sigmay
  sum=0.0
  loop i (1,igp)
    sum=sum-yforce(i,jgp)
  end_loop
  sigmay=sum/(x(igp,jgp)-x(1,jgp))
end
;
; This works out the stress in the x direction on RHS of the model
;
def sigmax
  sum=0.0
  loop j (1,jgp)
    sum=sum-xforce(igp,j)
  end_loop
  sigmax=sum/(y(igp,jgp)-y(igp,1))
end
;
; This works out the shear stress on the RHS of the model
;
def sigmaxy
  sum=0.0
  loop j (1,jgp)
    sum=sum-yforce(igp,j)
  end_loop
  sigmaxy=sum/(y(igp,jgp)-y(igp,1))
end
;
;
set echo on
```

B4.7 Test Type 2 (Constitutive model file 'Elastic.fis')

```
;FISH version of standard elastic model
; based on FLAC user manual page 7-57.
set echo off

def m_elastic
 constitutive_model
 f_prop  m_g m_k
 f_prop  m_e1 m_e2 m_g2

 Case_of  mode
; ----------------------
; Initialisation section
; ----------------------
   Case 1
    m_e1   = m_k + 4.0 * m_g / 3.0
    m_e2   = m_k - 2.0 * m_g / 3.0
    m_g2   = 2.0 * m_g
; ---------------
; Running section
; ---------------
   Case 2
  zs11=zs11 + zde11*m_e1 + (zde22 + zde33) * m_e2
  zs22=zs22 + zde22*m_e1 + (zde33 + zde11) * m_e2
  zs33=zs33 + zde33*m_e1 + (zde11 + zde22) * m_e2
  zs12=zs12 + zde12 * m_g2
; ----------------------
; Return maximum modulus
; ----------------------
   Case 3
    cm_max = m_k + 4.0 * m_g / 3.0
    sm_max = m_g
   Case 4
; --------------------
; Add thermal stresses
; --------------------
    ztsa = ztea * m_k
    ztsb = zteb * m_k
    ztsc = ztec * m_k
    ztsd = zted * m_k
 End_case
end

opt m_elastic
set echo on
```

B4.8 Test Type 3 (Initialisation routine)

```
;
; UDM test simulations
;
title
UDM Test Type 3 - elastic UDM

set large
set gravity 0 0
grid 10 10
;
```

Appendix B4: *FLAC* data files for verification Test Types 1,2 and 3

```
; Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic

def getKandG
  YoungsMod = 40e9
  Poissons = 0.25

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

prop den 2500 bulk BulkMod shear ShearMod

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfjzones = myjzones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
  halfjzonesplus1 = halfjzones + 1
end
initialise

def distort
  loop i (2,izones)
    loop j (2, jzones)
          x(i,j)=x(i,j) + mag * (urand -0.5)
      y(i,j)=y(i,j) + mag * (urand -0.5)
    end_loop
  end_loop
  x(3,6)=2*0.005
  y(3,6)=5*0.014
  x(4,6)=3*0.005
  y(4,6)=5*0.014
  x(3,7)=2*0.005
  y(3,7)=6*0.014
  x(4,7)=3*0.005
  y(4,7)=6*0.014

  x(8,6)=7*0.005
  y(8,6)=5*0.014
  x(9,6)=8*0.005
  y(9,6)=5*0.014
  x(8,7)=7*0.005
  y(8,7)=6*0.014
  x(9,7)=8*0.005
  y(9,7)=6*0.014
end
set mag = 0.0025
distort

save t3init01.sav
```

B4.9 Test Type 3 (Running routine)

```
;
; UDM test simulations
;

restore t3init01.sav

title
UDM Test Type 3 - elastic UDM

set large
set gravity 0 0
initialise

model elas i = 1,halfizones
prop den 2500 bulk BulkMod shear ShearMod i=1,halfizones

call elastic.fis
model m_elastic i = halfizonesplus1 ,myizones
prop den 2500 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones

ini xv 0 j=1
ini xv 0 j=myigp
ini yv 0 j=1
ini yv -1e-8 j=myjgp
ini sxx = -10e6
ini syy = -10e6
ini szz = -10e6

; The following commands fix the velocity of the selected gridpoints
fix y j=1
fix y j=myjgp
;
apply sxx -10e6 i=1 j=1,myjgp
apply sxx -10e6 i=myigp j=1,myjgp
;
;
call monitor.dat

def mdsxx
  $mdsxx =sxx(3,6) - sxx(8,6)
  if $mdsxx > maxmdsxx then
  maxmdsxx = $mdsxx
  end_if
  mdsxx=$mdsxx
end

def mdsyy
  $mdsyy = syy(3,6) - syy(8,6)
  if abs($mdsyy) > maxmdsyy then
  maxmdsyy = $mdsyy
  end_if
  mdsyy=$mdsyy
end

def mdszz
  $mdszz = szz(3,6) - szz(8,6)
  if abs($mdszz) > maxmdszz then
```

```
  maxmdszz = $mdszz
  end_if
  mdszz=$mdszz
end

def mdsxy
  $mdsxy = sxy(3,6) + sxy(8,6)
  if abs($mdsxy) > maxmdsxy then
  maxmdsxy = $mdsxy
  end_if
  mdsxy=$mdsxy
end


call servo.dat
set high_unbal=50e3
set low_unbal=25e3

his nstep 1
his unbal
his ey
his sigmay
his yv i=halfigp j = myjgp
his sxx i=3 j=6
his sxx i=8 j=6
his syy i=3 j=6
his syy i=8 j=6
his szz i=3 j=6
his szz i=8 j=6
his sxy i=3 j=6
his sxy i=8 j=6
his mdsxx
his mdsyy
his mdszz
his mdsxy


set plot po 0 0 0.55 0.55 0
;
; Produce a plot of the model
;
set output t3m.ps
plot pen model grid fix app

step 500

;
; Produce a plot of the normal stress component histories
;
set output t3rsign.ps
plot pen hist 5 6 7 8 9 10
;
; Produce a plot of the shear stress component histories
;
set output t3rsigs.ps
plot pen hist 11 12
;
; Produce a plot of differences between stress component histories
;
```

```
set output t3rsdf.ps
plot pen hist 13 14 15 16

print maxmdsxx maxmdsyy maxmdszz maxmdsxy
```

B5 *FLAC* results from ANN UDM verification tests

B5.1 Test Type 1

A single zone model of linear elastic material with E=40 GPa and ν=0.25 was produced. All grid points in the model had their velocities fixed. 200 iterations of the main finite difference calculation cycle were carried out while monitoring the strain in the x and y directions and all of the zone stress components.

Three distinct strain paths were used to test the single zone model. These are summarised in Figures B5.1 to B5.3 showing the grid point velocities for each case. Case 1 corresponds to a perfectly confined compression. Case 2 corresponds to shear along the top surface of the model. Case 3 is a combination of Cases 1 and 2. The results of these verification tests are presented in Figures B5.4 to B5.7.

For verification Test Type 1, around 200 iterations of the single zone model were found to be required, to produce a suitable series of performance curves. UDM effectiveness was assessed through inspection of the gradients of the stress-strain curves produced.

In the case of Figure B5.4, the gradients of the curves are 48 GPa for σ_{yy} or 16 GPa for σ_{xx} and σ_{zz} (which are values consistent with the expected values of Lamé's constants with E=40 and ν=0.25) or zero for σ_{xy}. In the case of Figure B5.5, the gradient of the plot is $2G$ which, again, is expected behaviour. The gradients of the curves for Figures B5.6 and B5.7 for the case of the combined strain paths of Cases 1 and 2, are consistent with their component cases.

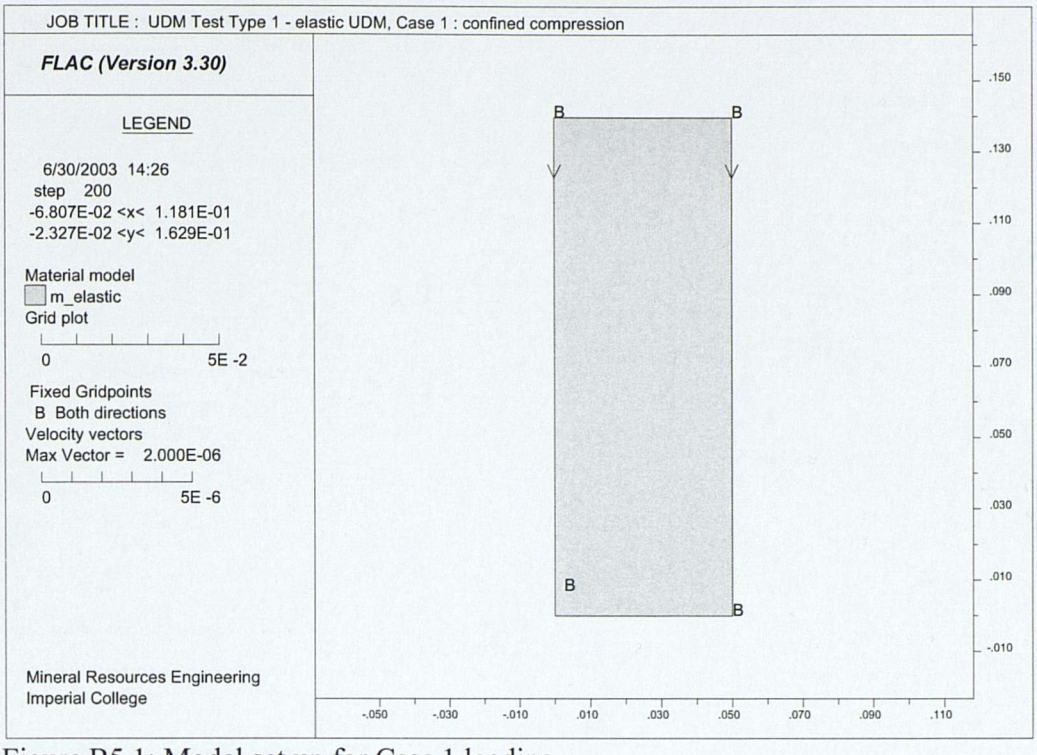


Figure B5.1: Model set up for Case 1 loading.

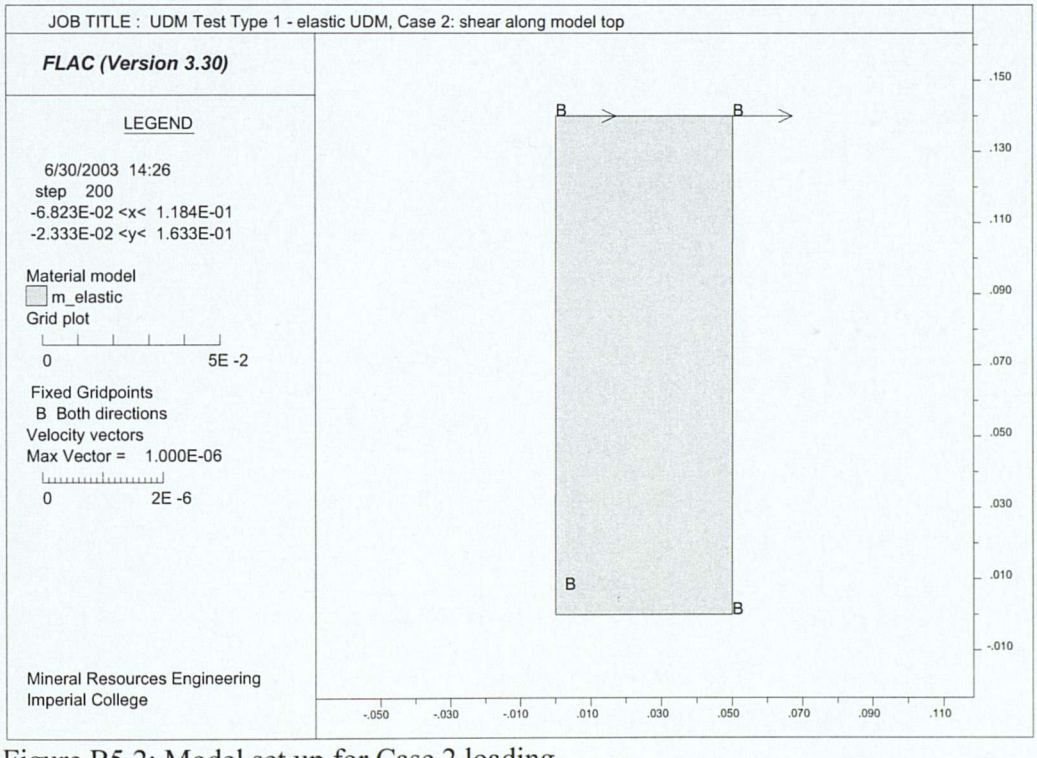


Figure B5.2: Model set up for Case 2 loading.

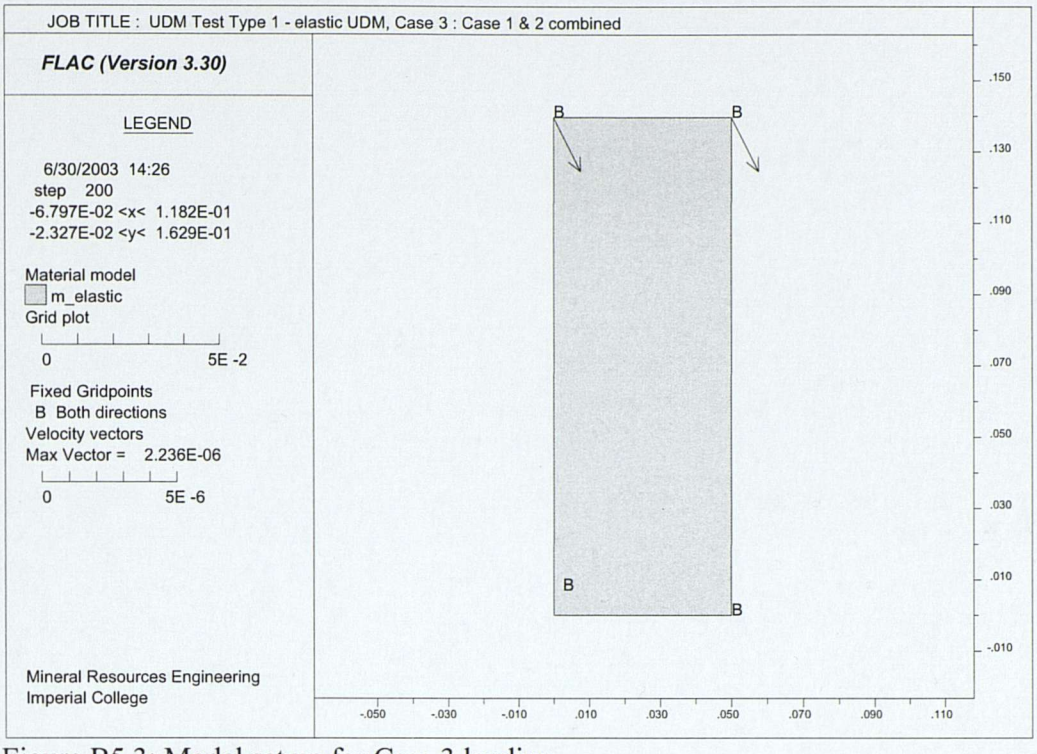


Figure B5.3: Model set up for Case 3 loading.

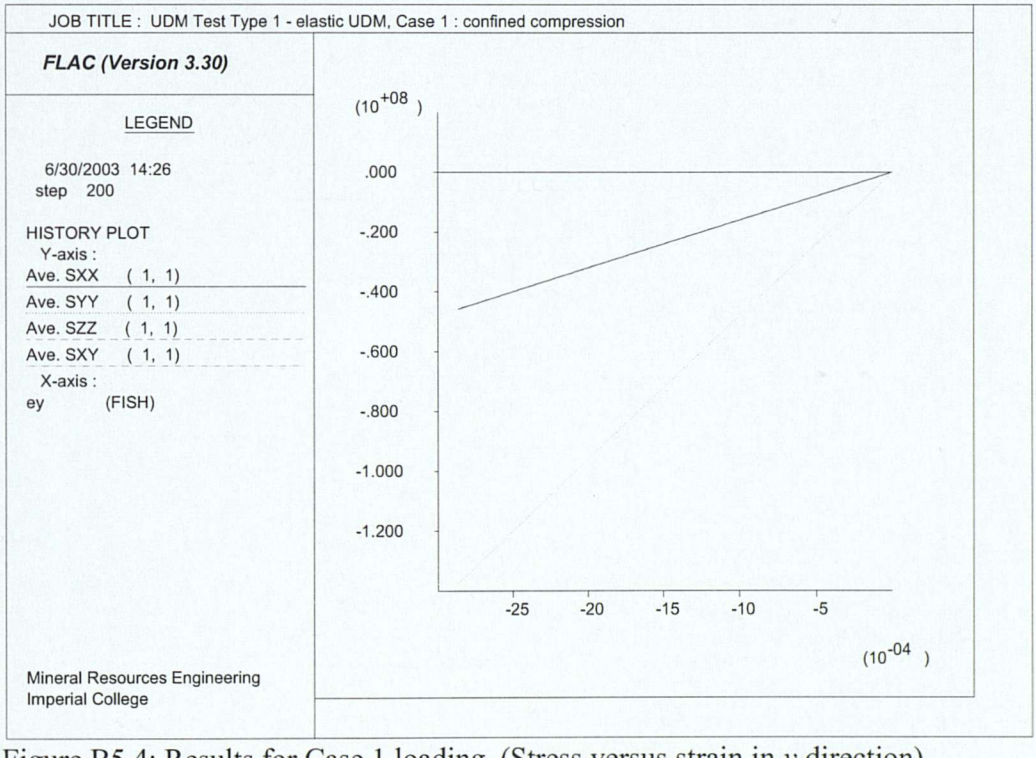


Figure B5.4: Results for Case 1 loading. (Stress versus strain in *y* direction)

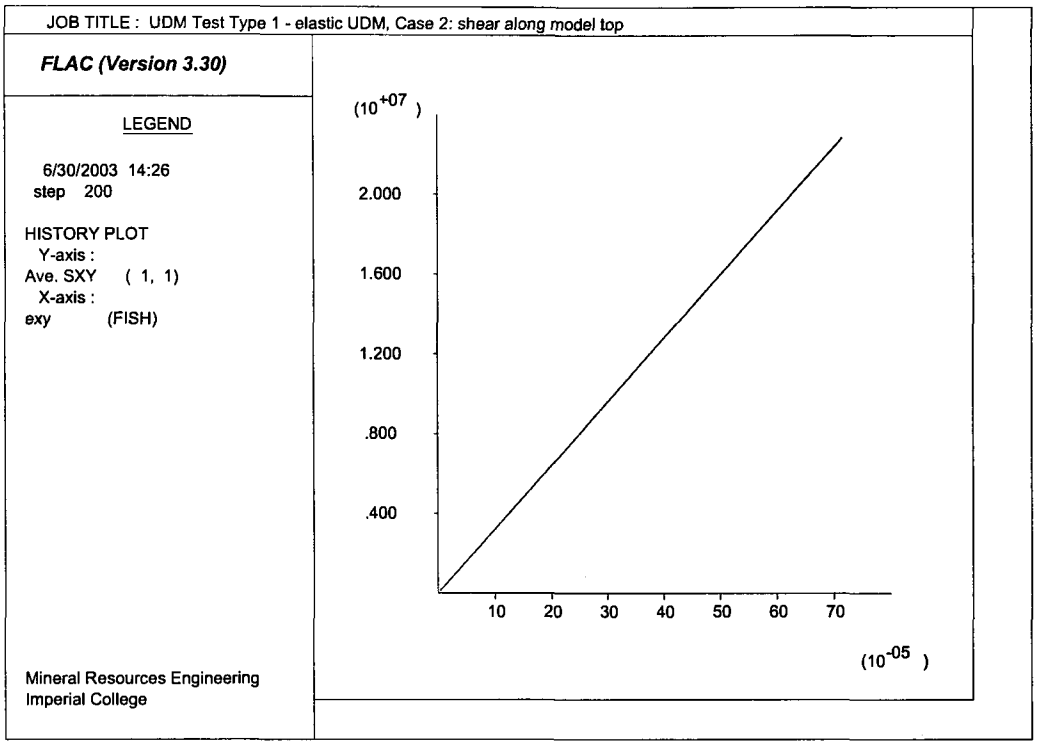


Figure B5.5: Results for Case 2 loading (Shear stress versus [tensorial] shear strain)

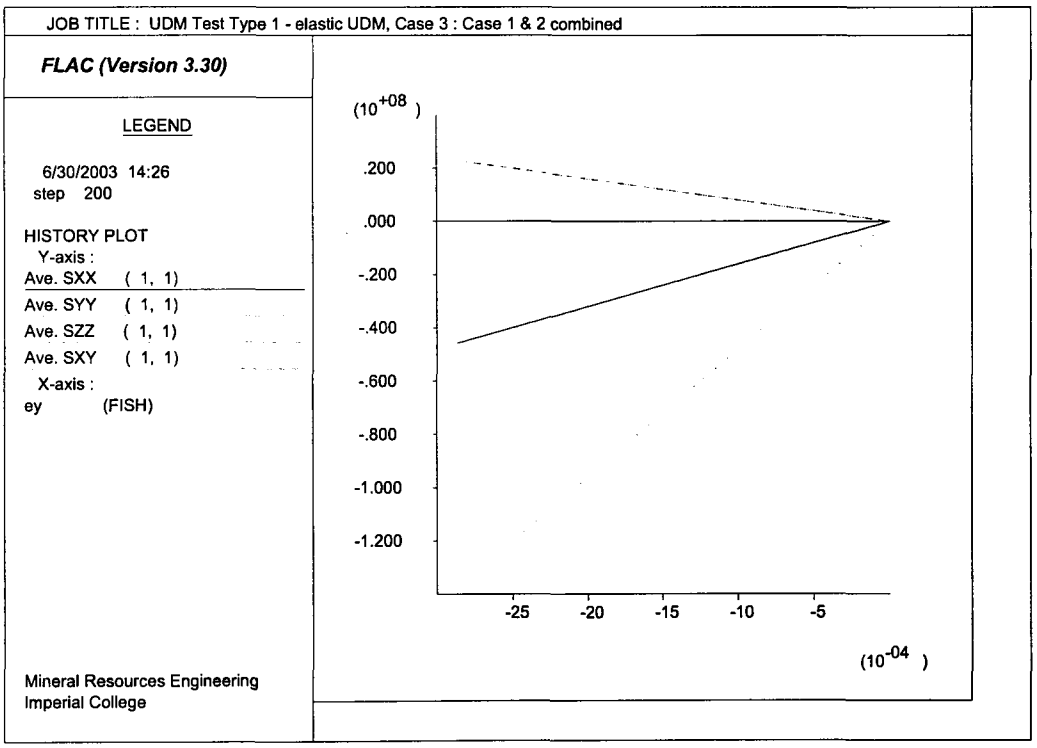


Figure B5.6: Results for Case 3 loading (Stress versus strain in y direction).

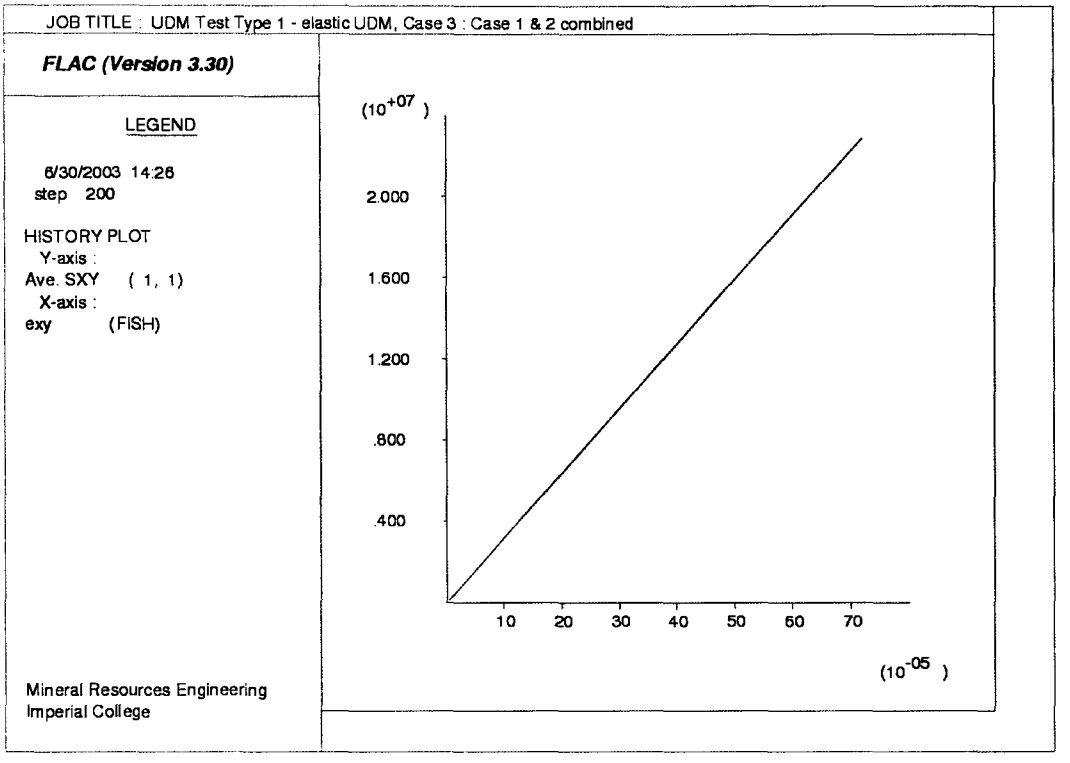


Figure B5.7 : Results for Case 3 loading (Shear stress versus [tensorial] shear strain).

B5.2 Test Type 2

The grid, material zones and applied boundary conditions used in test type 2 are shown in Figure B5.8. The LHS material zone comprises the *FLAC* in-built isotropic linear elastic constitutive model. The RHS material zones comprise the UDM isotropic linear elastic model used in the first verification test.

The problem modelled in Test Type 2 conforms to a triaxial compression test conducted with a confining pressure of 10 MPa. The model is monitored while cycling by recording the stress components at two locations, equidistant from the line of symmetry between the two materials.

For verification Test Type 2, around 500 time steps of the model are required to ascertain that the same macroscopic qualitative behaviour is recovered. The performance of the UDM is measured through the degree of state divergence between symmetrically positioned monitoring 'points'. Divergence between the values of normal and shear stress component magnitudes is obtained from around 250 time steps and is chaotic, that is, unpredictable in character, other than from around 500 time steps onwards, the amount of divergence remains between limits of -1.5×10^6 to $+1.5 \times 10^6$ Pa.

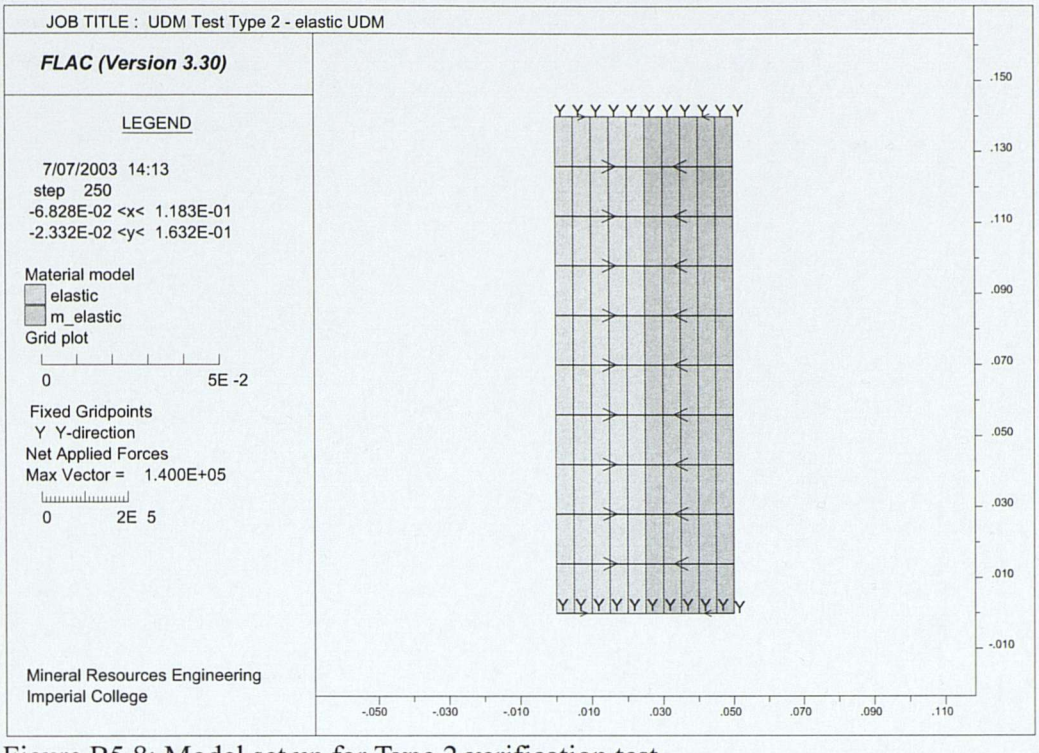


Figure B5.8: Model set up for Type 2 verification test.

These are zones located 3 and 8 zones along from the lower LHS of the model respectively and 6 zones from the base. The velocity of the top surface of the model was initially set at -1×10^{-8} m/s. A numerical servo-control procedure was activated during stepping in order to minimise inertial effects during the simulation. This increased (by a factor 1.025) or reduced (by a factor 0.975) the boundary y-velocity of vertices at the top of the model to ensure that the maximum unbalanced (accelerating) force within the model lay between 25 and 50 kN. Figure B5.9 shows a time step plot of σ_x, σ_y and σ_z for the two monitored locations after 500 time steps.

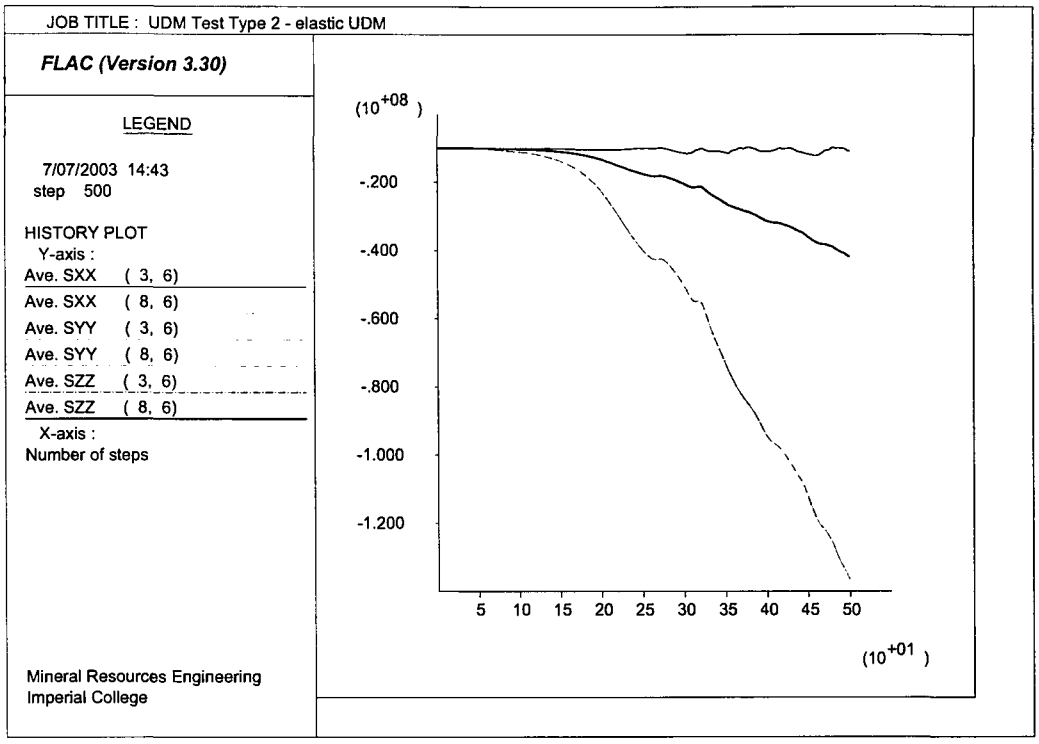


Figure B5.9: FLAC time step history plot of normal stress components at the two monitored locations in the model in Test Type 2.

Figure B5.9 plots six curves, and yet only three appear to be present. This is because, the values of the stress components at the two symmetrically located monitoring locations are close, across all time steps. This is the expected result if the elastic UDM is to be considered an exact replacement for the in-built elastic constitutive model. Figure B5.10 shows a time step plot of σ_{xy} at the two monitoring locations up to 500 time steps. In this case, the shear stress components at the two monitoring locations appear to have the same magnitude across all time steps but opposite sense. This is the expected result if the elastic UDM is to be considered an exact replacement of the in-built elastic constitutive model. The difference in sense is due to the symmetry of the global model and the sign convention applied for positive shear stress components.

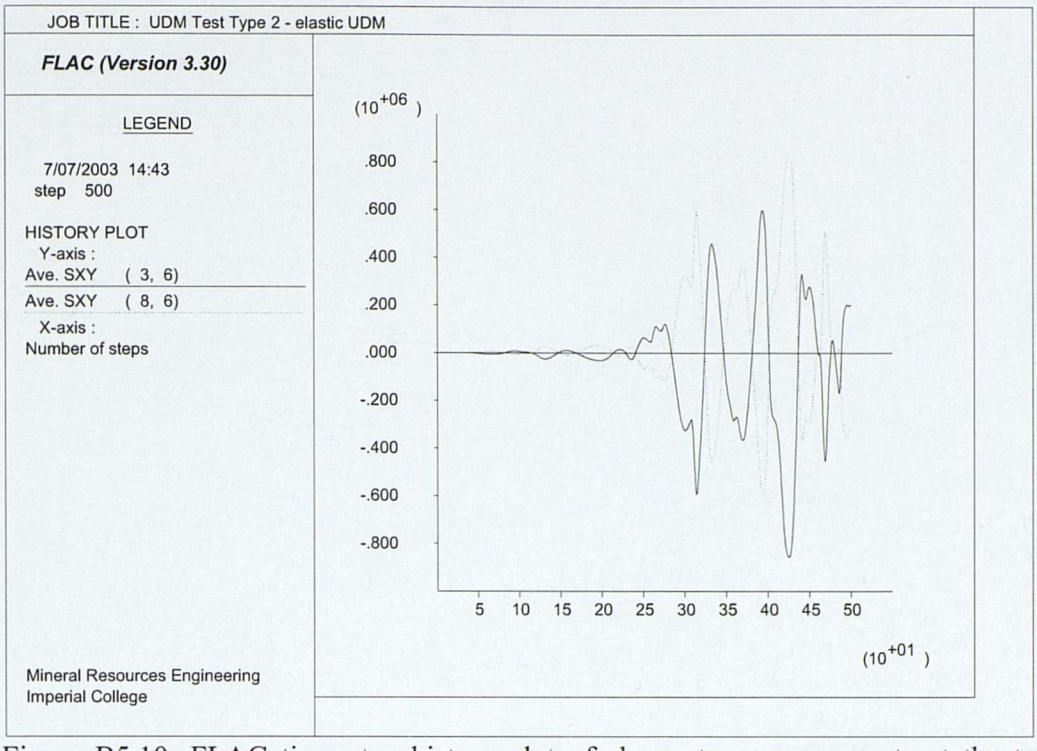


Figure B5.10: FLAC time step history plot of shear stress components at the two monitored locations in the model in Test Type 2.

If the differences in corresponding stress component values are examined in more detail, there are variations between them. Figure B5.11 shows a time step plot of the differences between the values, rather than time step plots of the values themselves. It can see seen from Figure B5.11 that the magnitudes of the maximum differences are insignificant up until around 250 time steps. Thereafter, the magnitudes of the differences increase and fluctuate appreciably up to 500 time steps, which may lead to the conclusion that there is an error in implementation. At 500 time steps the magnitudes of the maximum differences in the stress components are presented in Table B5.1 together with values of identical measures taken when the in-built *FLAC* elastic constitutive model is used throughout. A time history of the differences in stress component magnitudes for this latter case is presented in Figure B5.12, showing the same outcome as Figure B5.11 and hence that this is a phenomenon of *FLAC* generally, rather than any problem with the UDM.

Table B5.1: Maximum differences in stress component magnitudes after 500 time steps for the Test Type 2 UDM verification model

	Difference in σ_{xx}	Difference in σ_{yy}	Difference in σ_{zz}	Difference in σ_{xy}
Elastic UDM	1.5024×10^5	1.0480×10^4	1.8344×10^4	1.2137×10^5
FLAC elastic model throughout	2.0564×10^5	1.5459×10^5	8.4328×10^4	2.0597×10^5

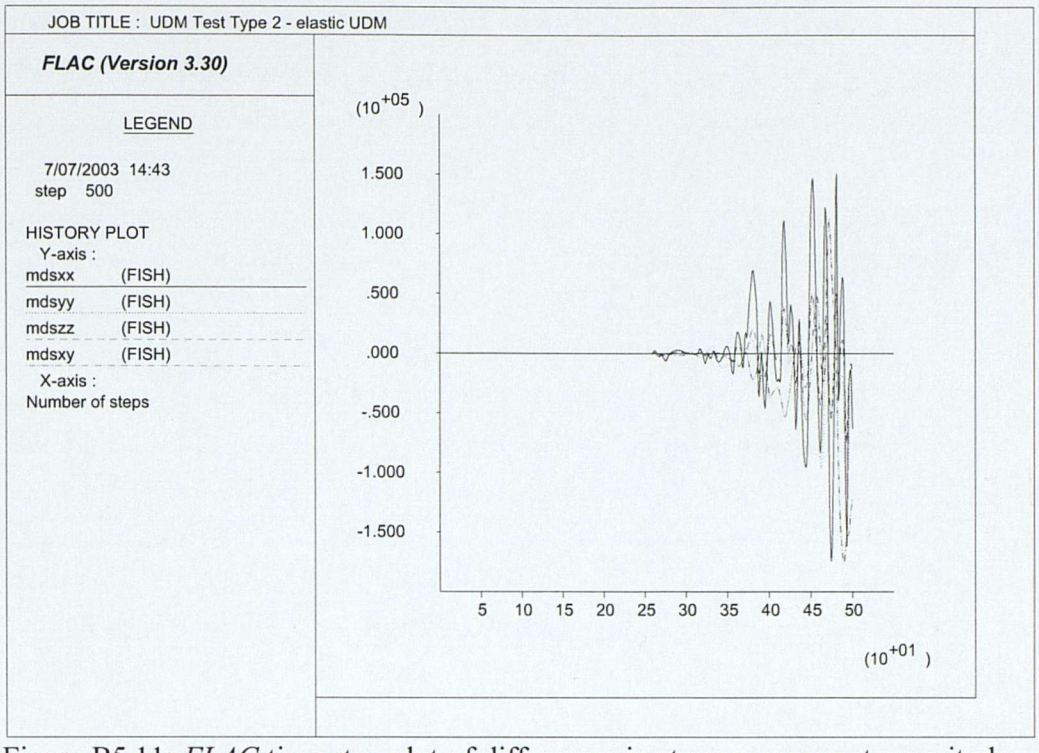


Figure B5.11: *FLAC* time step plot of differences in stress component magnitudes up to 500 time steps for the Test Type 2 UDM verification model with elastic UDM used on RHS of the model.

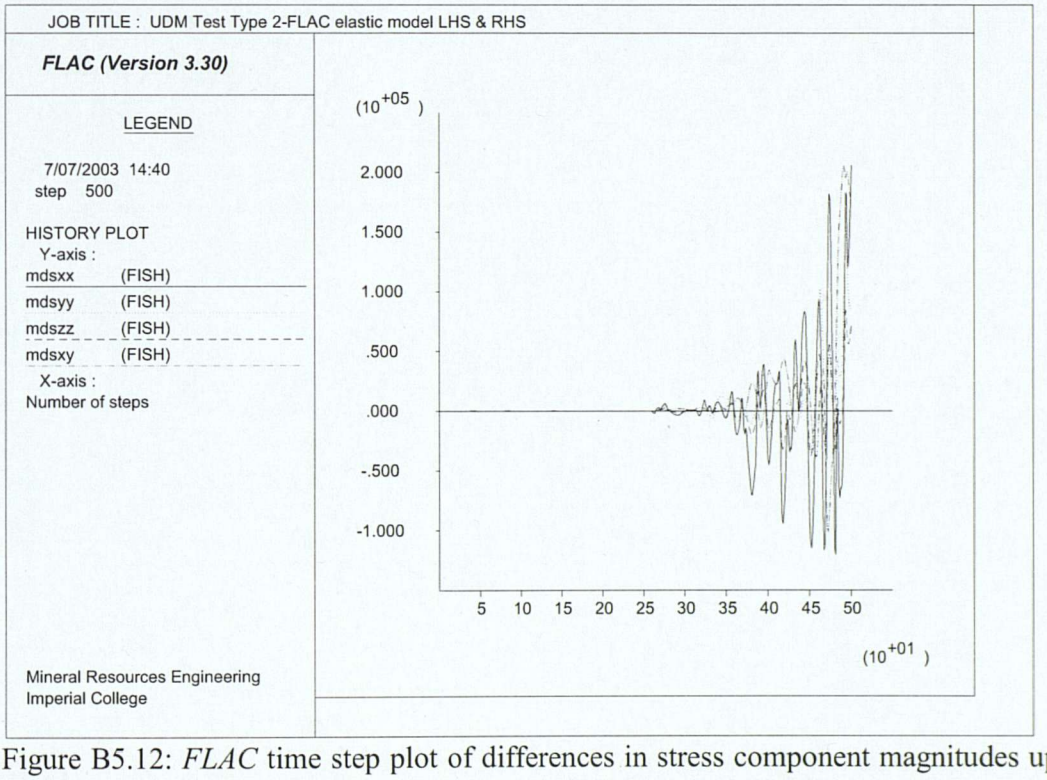


Figure B5.12: *FLAC* time step plot of differences in stress component magnitudes up to 500 time steps for the Test Type 2 UDM verification model with in-built *FLAC* elastic constitutive model used throughout the model.

It is clear from Figure B5.12 that even when the in-built elastic *FLAC* constitutive model is used throughout the finite difference grid, divergence in the state of symmetrically located 'observation points' is to be expected after a large number of iterations. It should be noted that after 500 simulated time steps while the differences in stress state at each observation point have an order of magnitude of 10^5, the magnitudes of the corresponding stress components themselves have an order of magnitude of 10^8. The precision of the stress component magnitudes at this time is thus of order 1/1000.

It is thought that in the current case, the divergence in state arises due to the numeric precision at which the computations are conducted within *FLAC*. The incremental formulation for the constitutive model means that the state of stress after a large number of time steps is the summation of the incremental contributions. After a great many arithmetic addition operations of floating point numbers, it is inevitable that rounding operations will lead to numerical variations that cause small differences in the conditions applying for the next time step. *FLAC* is known to be a computer code that is sensitive to small changes in initial conditions (Itasca, 1993). The variation in initial conditions, arising due to rounding operations is amplified as the number of time steps increases. As Figure B5.13 shows for σ_{yy} after 3500 time steps, state differences then settle to a steady, albeit, probably chaotic level of between 1.5 x 10^6 to -1.5 x 10^{-6}.

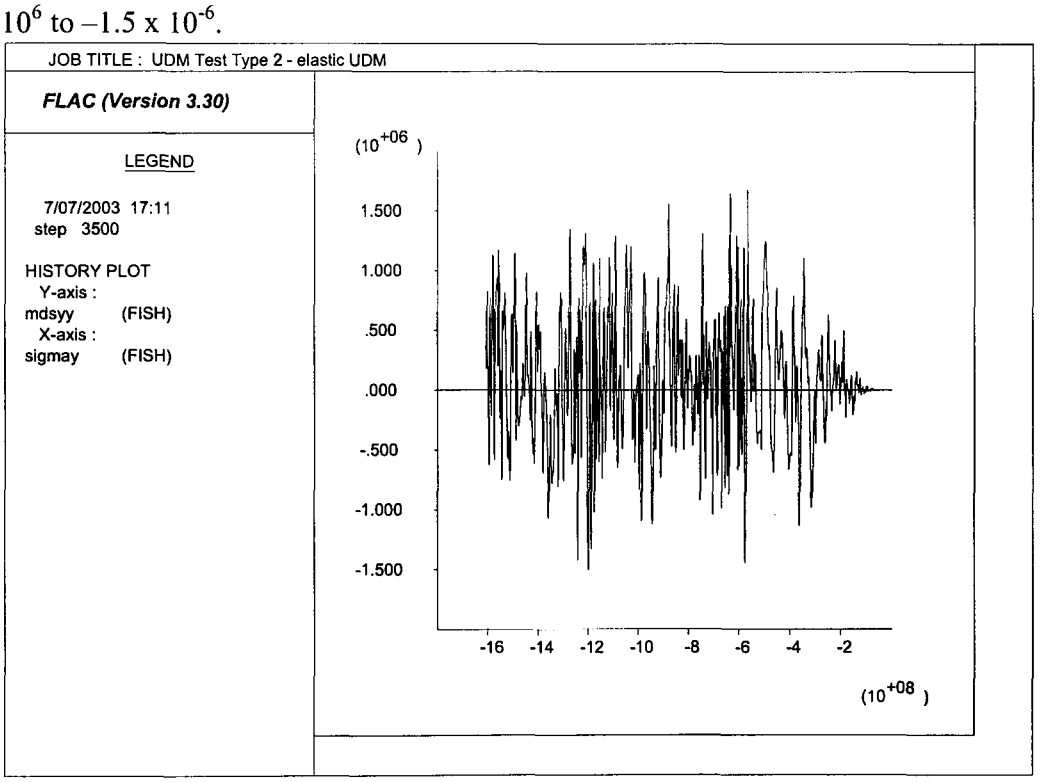


Figure B5.13: Plot of the difference in σ_{yy} between the symmetrically positioned monitoring location vs σ_{yy} over 3500 time steps.

B5.3 Test Type 3

The grid, material zones and applied boundary conditions used in test type 3 are shown in Figure B5.14. The LHS material zone comprises the *FLAC* in-built isotropic linear elastic constitutive model. The RHS material zones comprise the UDM isotropic linear elastic model used in the first and second verification tests. The finite difference grid has been distorted for this verification test.

For verification Test Type 3, around 500 time steps of the model are required to ascertain that the same macroscopic qualitative behaviour is recovered. The performance of the UDM is measured in the same way as Test Type 2. Divergence between the values of normal and shear stress component magnitudes is obtained from around 125 time steps and is also chaotic in character. Eventually the degree of divergence remains between limits of -2.0×10^6 to $+2.0 \times 10^6$ Pa. Symmetry is broken between the RHS and LHS shear stress component magnitudes much earlier (in terms of time steps) than the normal stress component magnitudes. At this stage, it is thought that this is a result of the mixed discretisation procedure which forms part of the main calculation loop of the *FLAC* code.

Otherwise, the conditions of the test are identical to that for verification test type 2.

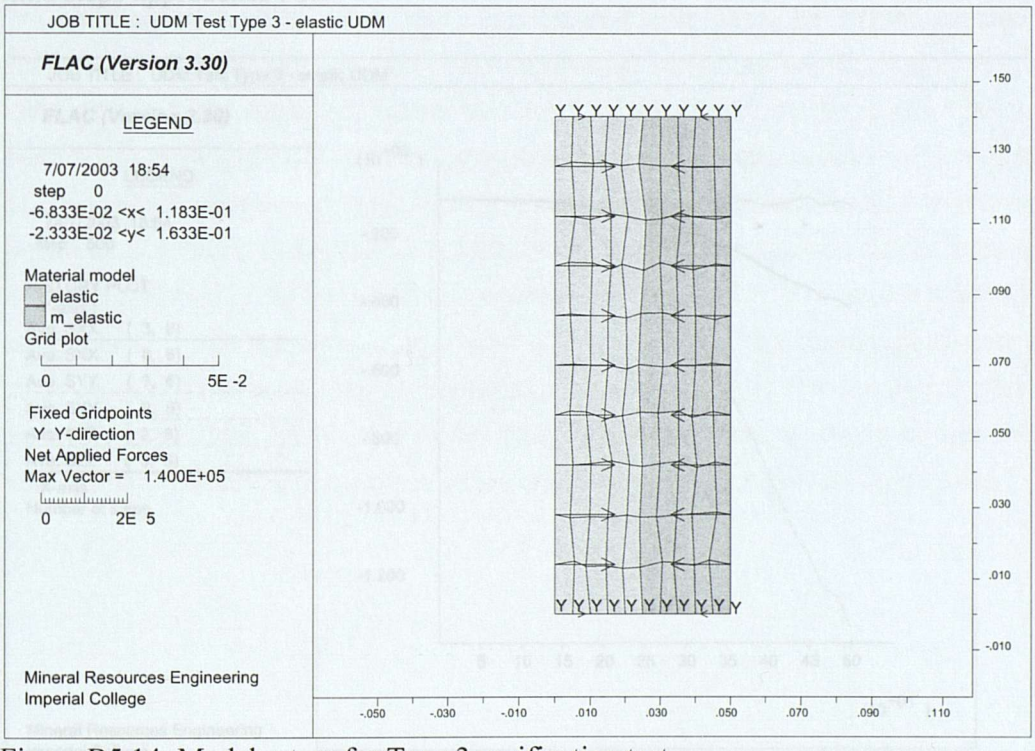


Figure B5.14: Model set up for Type 3 verification test.

The distortion process involved the use of a random number generator to offset the co-ordinates of each grid point in the grid using the following equations:

$$x := x + 0.0025 * \text{Rand} \qquad \text{(B5.1)}$$

$$y := y + 0.0025 * \text{Rand} \tag{B5.2}$$

where the function Rand produces a random number drawn from a uniform distribution between –0.5 and +0.5. Three distinct realisations of the grid were produced using different random seeds for the random number generator. This was done to provide some measure of the precision to be expected when conducting tests on developed UDMs under these ideal conditions. After the grid was deformed, the grid nodes of the finite different zones used to monitor were reset to their original values. This was done in order that symmetric monitoring locations could be re-established. Logically, the monitoring locations are in the same positions as for test type 2, 3 and 8 zones to the right from the LHS and 6 cells from the base of the model.

The results of test type 3 are shown in Figure B5.15 and B5.16. Figure B5.15 shows that there is excellent agreement between the values for the stress component magnitudes at the two monitoring locations within the model. Again, six curves have been plotted, and yet on initial inspection, only three appear to have been plotted. Detailed inspection reveals that there are small variations between the respective curves. In comparison with test type 2, it is thus evident that with the current more rigorous test, slightly inferior performance is obtained, especially as the number of time steps approaches 500.

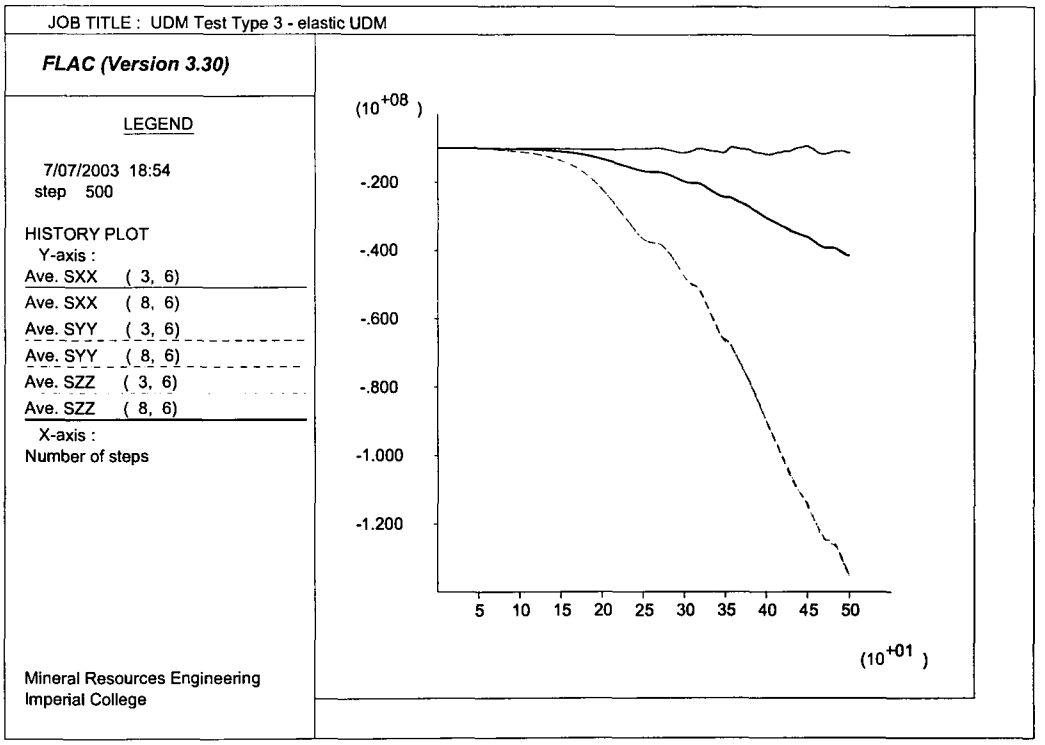


Figure B5.15: FLAC time step history plot of normal stress components at the two monitored locations in the model in Test Type 3.

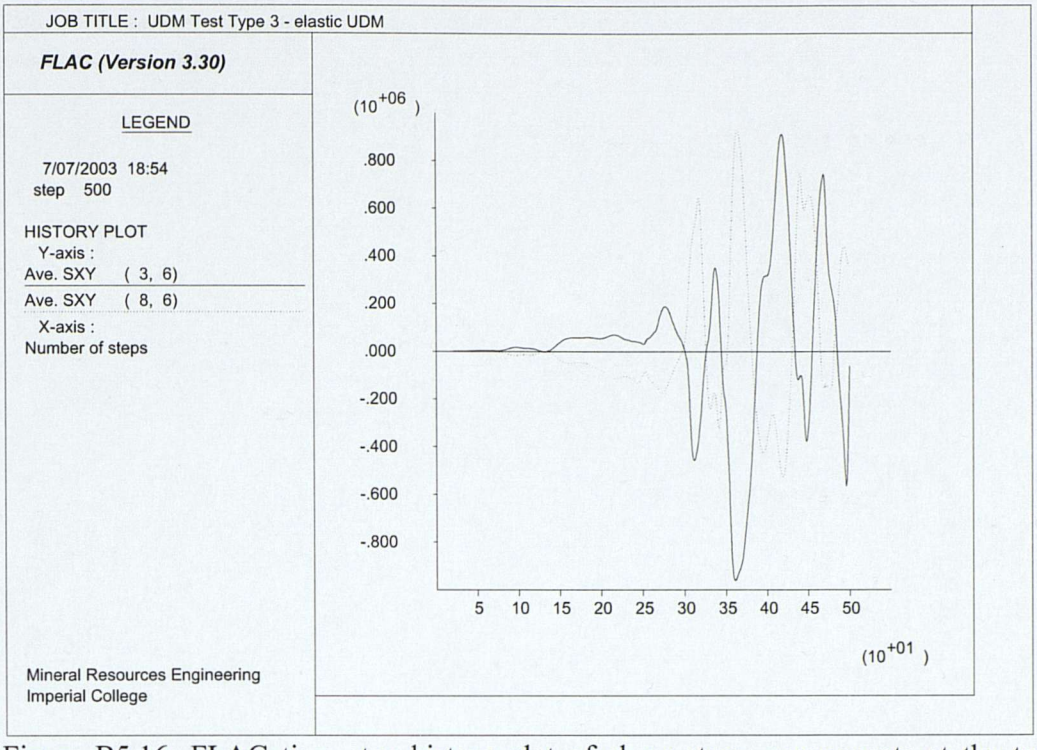


Figure B5.16: FLAC time step history plot of shear stress components at the two monitored locations in the model in Test Type 3.

If Figure B5.15 from test type 3 is compared with Figure B5.9 from test type 2, it is evident that the same normal stress component behaviour is produced in the two tests, which is an important additional check. If Figure B5.16 from test type 3 is compared with Figure B5.10 from test type 2, then a much greater level of divergence of shear stress component magnitudes for the two symmetrically located monitoring positions is evident. In Figure B5.16, the symmetry between the two curves is broken after approximately 125 time steps, whereas in Figure B5.10, symmetry breaking occurs later in the simulation, at around 475 time steps. The history plot of stress component differences shown in Figure B5.17 below, suggests that the magnitudes of these are larger than in the case of test type 2, however they appear to be limited to approximately 2 orders of magnitude (lower) of the stress component magnitudes.

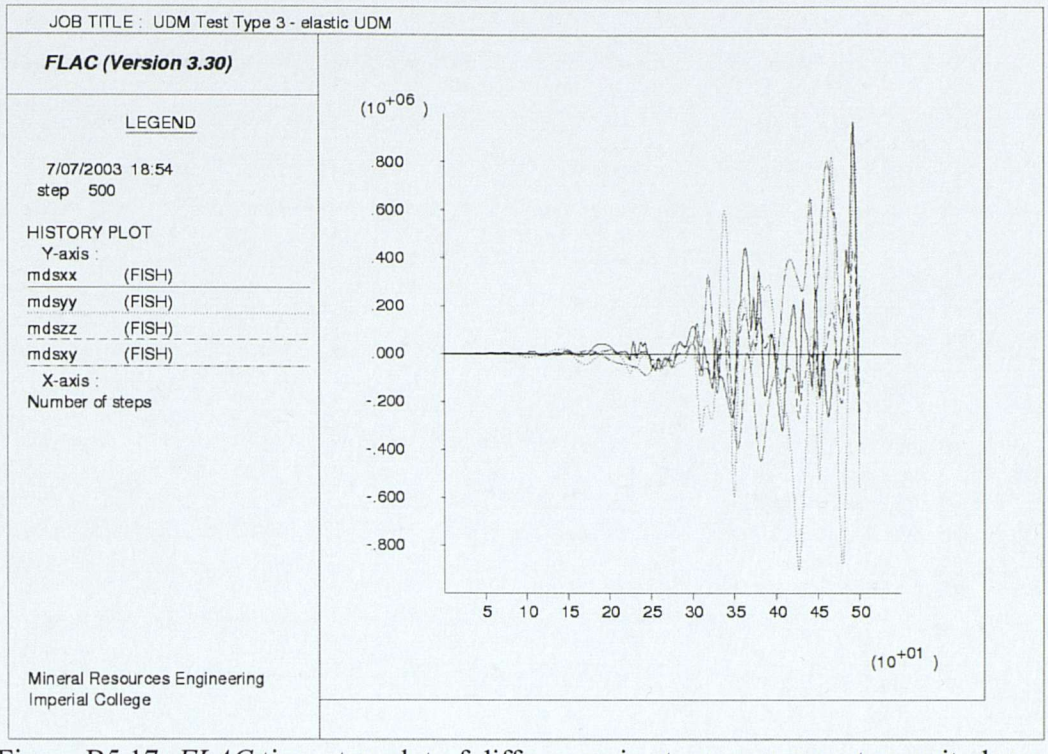


Figure B5.17: *FLAC* time step plot of differences in stress component magnitudes up to 500 time steps for the Test Type 3 UDM verification model with elastic UDM used on RHS of the model.

Table B5.2 reports the maximum stress component magnitude differences after 500 time steps, for the realisation for which results are presented, as well as two further realisations and the case where the in-built isotropic linear elastic model is used throughout. The figures in the table confirm that even for these near perfect conditions for comparison, significant variance in detailed performance of the UDM compared with the in-built *FLAC* constitutive model should be expected.

Table B5.2: Maximum differences in stress component magnitudes after 500 time steps for the Test Type 3 UDM verification model.

x 10^5	Difference in σ_{xx}	Difference in σ_{yy}	Difference in σ_{zz}	Difference in σ_{xy}
Elastic UDM Realisation 1	9.765	8.461	4.353	8.036
Elastic UDM Realisation 2	5.954	8.120	-1.177	6.918
Elastic UDM Realisation 3	7.210	1.198	4.254	7.254
FLAC elastic model throughout	7.548	7.677	3.525	7.354

B6 Procedures to facilitate use of ANN UDMs in *FLAC*

The four sections of this Appendix contain all the key numerical code to successfully embed an ANN UDM into *FLAC*. ANSIM (SAIC, 1992) was the primary ANN code used to conduct the ANN training in this work. ANSIM produces binary .net files for trained ANNs. The procedure presented in B6.1 imports the ANSIM .net files into Microsoft Excel. The procedure presented in B6.2 exports the same data to a *FLAC* FISH data file and is used to initialise the ANN within the UDM. The listing presented in B6.3 shows the output of the process in B6.2, and a FLAC .dat file showing the ANN UDM is coded is presented in B6.4. In the implementation used in this work, the listing in B6.4 remains the same for any constitutive behaviour modelled; the constitutive behaviour is altered for one material to another by creating an alternative initialisation file (B6.3) using the utility (B6.2). The code presented here streamlines the process of investigating ANN UDMs using *FLAC*. To see how the ANN UDM is used in a full *FLAC* simulation, refer back to the *FLAC* verification test listings of Appendix C4.

B6.1 Visual Basic for Applications procedure to import ANSIM .net ANN files into Microsoft Excel.

```
Sub OpenNetwork()
'
' This routine uses the ANSIM utility program called 'dumpnet' to convert a binary
' artificial neural network file into a text file. The procedure then reads in the weights
' thresholds and ANN structure and places on a worksheet in named ranges.
'
   Dim fileToOpen As Variant
   Dim Original_Directory As String
   Dim current_directory As String
   Dim file_direcotry As String
   Dim TString As String
   Dim ProcID As Integer
   Dim command_string As Variant
   Dim FileNum As Integer
   Dim numlayers As Integer
   Dim NumThresholds As Integer
   Dim NumWeights As Integer
   Dim i As Integer
   Dim j As Integer
   Dim k As Integer
   Dim l As Integer
   Dim ll As Integer
   Dim Sheetname As String
   Dim NetSheet As String

   Dim Structure() As Integer
   Dim Weights() As Double
   Dim Thresholds() As Double
'
' Find the binary .net file
```

Appendix B6: Procedures to facilitate use of ANN UDMs in *FLAC*.

```
'
   Sheetname = ActiveSheet.Name
   Original_Directory = CurDir()

   fileToOpen = Application.GetOpenFilename("ANSIM ANN Files (*.net), *.net")
   current_directory = CurDir("C")

   If fileToOpen <> False Then
      fileToOpen = GetFilenameFromPath(CStr(fileToOpen))
      NetSheet = fileToOpen
   End If
'
'  Convert it to ASCII (.ann) using dumpnet.exe utility (spawn to a command shell).
'
   command_string = "c:\ansim\dumpnet " & fileToOpen & ".net >" & fileToOpen & ".ann"
   If command_string <> False Then
      ProcID = Shell(command_string)
      Application.Wait (Now + TimeValue("00:00:01"))
   End If
'
'  Set up worksheet to write the information
'
   Application.ScreenUpdating = False
   Sheets.Add
   ActiveWindow.DisplayGridlines = False
   ActiveSheet.Name = NetSheet
   fileToOpen = current_directory & "\" & fileToOpen & ".ann"
   ActiveSheet.Cells(1, 1) = fileToOpen
   ActiveSheet.Cells(2, 1) = "Weights"
   ActiveSheet.Cells(2, 2) = "Thresholds"
   ActiveSheet.Cells(2, 3) = "Structure"
'
'  Open the .ann file and read structure data
'
   FileNum = FreeFile()
   Open fileToOpen For Input Access Read Shared As FileNum

   Do While Not EOF(FileNum) ' Loop until end of file.
      Line Input #FileNum, TString ' Read line into variable.
      If StrComp(Left(TString, 18), "Number of layers =") = 0 Then
         numlayers = Format(Mid(TString, 19), "#####")
         Exit Do
      End If
   Loop

   ReDim Structure(numlayers) As Integer
   Do While Not EOF(FileNum) ' Loop until end of file.
      Line Input #FileNum, TString ' Read line into variable.
      If StrComp(Left(TString, 17), "Layer dimensions:") = 0 Then
         For i = 1 To numlayers Step 1
            Structure(i) = 1
            Line Input #FileNum, TString ' Read line into variable.
            l = Len(TString)
            For j = 1 To 1 Step -1
               If StrComp(Mid(TString, j, 1), "(") = 0 Then
                  For k = j To 1 Step 1
                     If StrComp(Mid(TString, k, 1), ",") = 0 Then Exit For
                  Next
                  Structure(i) = Structure(i) * Format(Mid(TString, j + 1, k - 1 - j), "####")
```

```
                For ll = k To l Step 1
                    If StrComp(Mid(TString, ll, 1), ")") = 0 Then Exit For
                Next
                Structure(i) = Structure(i) * Format(Mid(TString, k + 1, ll - k - 1), "####")
                Exit For
            End If
        Next
    Next
    Exit Do
  End If
Loop
NumThresholds = 0
NumWeights = 0
For i = 2 To numlayers
  NumThresholds = NumThresholds + Structure(i)
  NumWeights = NumWeights + Structure(i - 1) * Structure(i)
Next
'
' Now read thresholds
'
ReDim Thresholds(NumThresholds) As Double
ReDim Weights(NumWeights) As Double

Do While Not EOF(FileNum) ' Loop until end of file.
  Line Input #FileNum, TString ' Read line into variable.
  If StrComp(Left(TString, 36), "Biases for hidden and output layers:") = 0 Then
    k = 0
    ll = l
    For i = 2 To numlayers Step 1
      Line Input #FileNum, TString ' Read line into variable.
      k = k + Structure(i)
      For j = ll To k Step 1
        Input #FileNum, Thresholds(j)
      Next
      ll = ll + k
    Next
    Exit Do
  End If
Loop
'
' Now read the weights
'
Do While Not EOF(FileNum) ' Loop until end of file.
  Line Input #FileNum, TString ' Read line into variable.
  If StrComp(Left(TString, 60), "Weight matrices: (to unit I from unit J, J changing fastest)") = 0
Then
    k = 0
    ll = l
    For i = 2 To numlayers Step 1
      Line Input #FileNum, TString ' Read line into variable.
      k = k + Structure(i - 1) * Structure(i)
      For j = ll To k Step 1
        Input #FileNum, Weights(j)
      Next
      ll = ll + k
    Next
    Exit Do
  End If
Loop
```

```
    Close FileNum
    Kill fileToOpen
'
'   Write the weights, thresholds and structure to designated areas on worksheet
'
    k = 3
    For i = 1 To NumWeights
        ActiveSheet.Cells(k, 1) = Weights(i)
        k = k + 1
    Next
    k = 3
    For i = 1 To NumThresholds
        ActiveSheet.Cells(k, 2) = Thresholds(i)
        k = k + 1
    Next
    k = 3
    For i = 1 To numlayers
        ActiveSheet.Cells(k, 3) = Structure(i)
        k = k + I
    Next
'
'   Name the areas for future reference
'
    TString = "='" & NetSheet & "'" & "!$A$3:$A$" & Format(NumWeights + 2, "##########")
    ActiveWorkbook.Names.Add Name:="Weights", RefersToR1C1:=Application.Evaluate(TString)

    TString = "='" & NetSheet & "'" & "!$B$3:$B$" & Format(NumThresholds + 2, "##########")
    ActiveWorkbook.Names.Add Name:="Thresholds", RefersToR1C1:=Application.Evaluate(TString)

    TString = "='" & NetSheet & "'" & "!$C$3:$C$" & Format(numlayers + 2, "##########")
    ActiveWorkbook.Names.Add Name:="Structure", RefersToR1C1:=Application.Evaluate(TString)

    ChDir Original_Directory
    Worksheets(Sheetname).Activate
    Application.ScreenUpdating = True
End Sub

Private Function GetFilenameFromPath(Path As String) As String
' Utility proceedure to strip out filename from a complete (text) path to the file.
'
    Dim i As Integer
    Dim j As Integer
    Dim l As Integer

    l = Len(Path)
    For i = l To 1 Step -1
        If StrComp(Mid(Path, i - 1, 1), "\") = 0 Then Exit For
    Next

    For j = l To 1 Step -1
        If StrComp(Mid(Path, j - 1, 1), ".") = 0 Then Exit For
    Next

    GetFilenameFromPath = Mid(Path, i, j - i - 1)
End Function
```

B6.2 Visual Basic for Applications procedure to export ANN data to a *FLAC* ANN UDM initialisation file.

```
Sub WriteFLACInitialisationFile()
'  This proceedure identifies weight, threshold and ANN structure data on a MS Excel
'  worksheet and then writes it to a FLAC initialisation file.
'
   Dim OStream As Integer
   Dim Original_Directory As String
   Dim FileSaveName As Variant
   Dim WriteString As String
   Dim Weights As Range
   Dim Thresholds As Range
   Dim Structure As Range

   Dim wts() As Single
   Dim thr() As Single
   Dim str() As Single
   Dim vths As Single

   Dim numlayers As Integer
   Dim nunits As Integer
   Dim i, j, k, l, m As Integer
'
'  Find the named ranges (see OpenNetwork procedure)
'
   Set Weights = Range(ActiveWorkbook.Names("Weights"))
   Set Thresholds = Range(ActiveWorkbook.Names("Thresholds"))
   Set Structure = Range(ActiveWorkbook.Names("Structure"))

   numlayers = Structure.Rows.Count
   For i = 1 To numlayers
      nunits = nunits + Structure(i, 1)
   Next
'
'  Decide where to save the FLAC initialisation file and open for writing
'
   Original_Directory = CurDir()
   FileSaveName = Application.GetSaveAsFilename(fileFilter:="FLAC ANN Initialisation File (*.dat),
*.dat")
   If FileSaveName <> False Then
      OStream = FreeFile()
      Open FileSaveName For Output Access Write Shared As OStream
   End If
'
'  Write header information
'
   WriteString = ";THIS FILE INITIALISES THE BASIC NETWORK STRUCTURE AND
SPECIFIES"
   Print #OStream, WriteString
   WriteString = ";THE WEIGHTS AND THRESHOLDS THAT HAVE BEEN PRODUCED IN
TRAINING"
   Print #OStream, WriteString
   WriteString = ";THE NEURAL NETWORK EXTERNALLY. IN ADDITION, PLACES FOR THE
INPUT MAPPING"
   Print #OStream, WriteString
   WriteString = ";PARAMETERS ARE SPECIFIED AS WELL AS THE OUTPUT REMAPPING
PARAMETERS."
   Print #OStream, WriteString
```

```
    WriteString = ";File produced with automatic FLAC ANN initialisation file write routine"
    Print #OStream, WriteString
    WriteString = ";Written by D.L. Millar"
    Print #OStream, WriteString

    Print #OStream, ";"
'
'   Write meta data into file for ease of reference
'
    WriteString = "Def Initialise_Network"
    Print #OStream, WriteString
    WriteString = "; Initial ANN structural details"
    Print #OStream, WriteString
    WriteString = "   inputs = " & CStr(Structure(1, 1))
    Print #OStream, WriteString
    WriteString = "   outputs = " & CStr(Structure(numlayers, 1))
    Print #OStream, WriteString
    WriteString = "   nlayers = " & CStr(numlayers)
    Print #OStream, WriteString
    WriteString = "   nunits = " & CStr(nunits)
    Print #OStream, WriteString
    WriteString = "; Number of the table containing thresholds"
    Print #OStream, WriteString
    WriteString = "   thresh=nunits+outputs+1"
    Print #OStream, WriteString
    WriteString = "; Number of the table containing activations"
    Print #OStream, WriteString
    WriteString = "   act=nunits+outputs+2"
    Print #OStream, WriteString
    WriteString = "; Number of the table containing the network structure"
    Print #OStream, WriteString
    WriteString = "   struct=nunits+outputs+3"
    Print #OStream, WriteString
    WriteString = ";"
    Print #OStream, WriteString
    WriteString = ";"
    Print #OStream, WriteString
    WriteString = ";"
    Print #OStream, WriteString
    WriteString = "; The first table defined contains the network structure."
    Print #OStream, WriteString
    WriteString = "; Notes:"
    Print #OStream, WriteString
    WriteString = "; The x values of the (x,y) pairs are the sequence number in the table."
    Print #OStream, WriteString
    WriteString = "; This is intended as an aide-memoire."
    Print #OStream, WriteString
    WriteString = "; This table is initialised first to make information available on the network structure"
    Print #OStream, WriteString
    WriteString = "; available but as the table numbers are used in indexing to access weights,
thresholds"
    Print #OStream, WriteString
    WriteString = "; and activations, the table number is assigned a high value."
    Print #OStream, WriteString
    WriteString = ";"
    Print #OStream, WriteString
    WriteString = "   COMMAND"
    Print #OStream, WriteString
'
```

```
' Write the structure of the ANN into a FLAC table
'
   WriteString = "     TABLE struct "
   For i = 1 To numlayers
      WriteString = WriteString & CStr(i) & " " & CStr(Structure(i, 1)) & " "
   Next
   Print #OStream, WriteString
'
' Write the normalisation / denormalisation coefficients into FLAC tables
'
   WriteString = ";"
   Print #OStream, WriteString
   WriteString = "; Table 1 contains the remapping parameters for the input quantities"
   Print #OStream, WriteString
   WriteString = "; to ANN space for the first input neuron."
   Print #OStream, WriteString
   WriteString = "; Notes:"
   Print #OStream, WriteString
   WriteString = "; This is a linear remapping between the maximum and minimum values for sigma1"
   Print #OStream, WriteString
   WriteString = "; within the training dataset and the interval -0.5 to 0.5."
   Print #OStream, WriteString
   WriteString = "; The x component of the (x,y) pairs gives the gradient of the remapping function"
   Print #OStream, WriteString
   WriteString = "; The y component of the (x,y) pairs gives the intercept of the remapping function"
   Print #OStream, WriteString
   WriteString = "; The tables that follow the first also contain the remapping parameters up until"
   Print #OStream, WriteString
   WriteString = "; tablenumber = number of neurons in the input layer."
   Print #OStream, WriteString
   WriteString = ";"
   For i = 1 To Structure(1, 1)
      WriteString = "     TABLE " & " " & CStr(i) & " " & "Norm Norm"
      Print #OStream, WriteString
   Next
   WriteString = ";"
'
' Write the ANN weights into FLAC tables
'
   WriteString = "; Tables " & CStr(i) & " to " & CStr(nunits) & " contain the network weights."
   Print #OStream, WriteString
   WriteString = "; Notes"
   Print #OStream, WriteString
   WriteString = "; The table number corresponds to the neuron number and is found through"
   Print #OStream, WriteString
   WriteString = "; lookup of the number of neurons in the input layer stored in table 1"
   Print #OStream, WriteString
   WriteString = "; The first " & CStr(Structure(1, 1)) & " neurons being inputs have no preceedent
weights thus"
   Print #OStream, WriteString
   WriteString = "; the next table number corresponds to the number of the first neuron"
   Print #OStream, WriteString
   WriteString = "; in the second layer of the neural network."
   Print #OStream, WriteString
   WriteString = "; The x values of the (x,y) pairs are the sequence number in the table."
   Print #OStream, WriteString
   WriteString = "; This is intended as an aide-memoire."
   Print #OStream, WriteString
   l = Structure(1, 1)
```

Appendix B6: Procedures to facilitate use of ANN UDMs in *FLAC*.

```
    For i = 2 To numlayers
      For j = 1 To Structure(i, 1)
        l = l + 1
        WriteString = "     TABLE " & CStr(l)
        For k = 1 To Structure(i - 1, 1)
          m = m + 1
          WriteString = WriteString & " " & CStr(k) & " " & CStr(Weights(m, 1))
        Next
        Print #OStream, WriteString
      Next
    Next
'
'   Write the denormalisation parameters into a FLAC table
'
    WriteString = "; The next " & CStr(Structure(numlayers, 1)) & " tables contain the parameters"
    Print #OStream, WriteString
    WriteString = "; to map from ANN space to stress space."
    Print #OStream, WriteString
    For i = 1 To Structure(numlayers, 1)
      l = l + 1
      WriteString = "     TABLE " & CStr(l) & " DeNorm DeNorm"
      Print #OStream, WriteString
    Next
'
'   Write the threshold quantities into a FLAC table
'
    WriteString = "; The next table contains the neuron thresholds (biases)."
    Print #OStream, WriteString
    WriteString = "; Notes:"
    Print #OStream, WriteString
    WriteString = "; The first " & CStr(Structure(1, 1)) & "neurons (being inputs) have no bias."
    Print #OStream, WriteString
    WriteString = "; The x values of the (x,y) pairs are the sequence number in the table."
    Print #OStream, WriteString
    WriteString = "; This is intended as an aide-memoire."
    Print #OStream, WriteString
    WriteString = ";"
    Print #OStream, WriteString

    k = 0
    For i = 1 To numlayers
      WriteString = "     TABLE thresh "
      For j = 1 To Structure(i, 1)
        k = k + 1
        If i = 1 Then
          vths = 0
        Else
          vths = Thresholds(k - Structure(1, 1), 1)
        End If
        WriteString = WriteString & " " & CStr(k) & " " & CStr(vths)
      Next
      Print #OStream, WriteString
    Next

    WriteString = "     TABLE thresh  1 0 2 0 3 0 4 0 5 0"
    WriteString = "     TABLE thresh 6 0.63194 7 -5.44017 8 -0.08008 9 -5.84121 10 -5.72145"
    WriteString = "     TABLE thresh 11 1.92727 12 -2.39840 13 -2.23618"
    WriteString = ";"
    Print #OStream, WriteString
```

```
'
' Write the neuron activation values to a FLAC table (all zero initially)
'
  WriteString = "; The next table contains the neuron activations."
  Print #OStream, WriteString
  WriteString = "; Notes:"
  Print #OStream, WriteString
  WriteString = "; The first " & CStr(Structure(1, 1)) & " pairs correspond to input values."
  Print #OStream, WriteString
  WriteString = "; The last " & CStr(Structure(numlayers, 1)) & " pairs correspond to input values."
  Print #OStream, WriteString
  WriteString = "; All values initialised at zero to begin."
  Print #OStream, WriteString
  WriteString = ";"
  Print #OStream, WriteString

  k = 0
  For i = 1 To numlayers
    WriteString = "     TABLE act "
    For j = 1 To Structure(i, 1)
      k = k + 1
      WriteString = WriteString & " " & CStr(k) & " " & CStr(0)
    Next
    Print #OStream, WriteString
  Next


  WriteString = "     TABLE act 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0 11 0 12 0 13 0"
  WriteString = ";"
  Print #OStream, WriteString
  WriteString = "   END_COMMAND"
  Print #OStream, WriteString
  WriteString = "END"
  Print #OStream, WriteString
'
' That's it - close the file
'
  Close OStream
End Sub
```

B6.3 *FLAC (FISH)* linear elastic ANN-UDM initialisation data file (ANN-ILE.dat) used in Section 6.2.3.

```
;THIS FILE INITIALISES THE BASIC NETWORK STRUCTURE AND SPECIFIES
;THE WEIGHTS AND THRESHOLDS THAT HAVE BEEN PRODUCED IN TRAINING
;THE NEURAL NETWORK EXTERNALLY. IN ADDITION, PLACES FOR THE INPUT
MAPPING
;PARAMETERS ARE SPECIFIED AS WELL AS THE OUTPUT REMAPPING PARAMETERS.
;File produced with automatic FLAC ANN initialisation file write routine
;Written by D.I.. Millar
;
Def Initialise_Network
; Initial ANN structural details
  inputs = 3
  outputs = 3
  nlayers = 3
  nunits = 12
; Number of the table containing thresholds
  thresh=nunits+outputs+1
```

Appendix B6: Procedures to facilitate use of ANN UDMs in *FLAC*.

```
; Number of the table containing activations
  act=nunits+outputs+2
; Number of the table containing the network structure
  struct=nunits+outputs+3
;
;
;
; The first table defined contains the network structure.
; Notes:
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
; This table is initialised first to make information available on the network structure
; available but as the table numbers are used in indexing to access weights, thresholds
; and activations, the table number is assigned a high value.
;
  COMMAND
    TABLE struct 1 3 2 6 3 3
;
; Table 1 contains the remapping parameters for the input quantities
; to ANN space for the first input neuron.
; Notes:
; This is a linear remapping between the maximum and minimum values for sigma1
; within the training dataset and the interval -0.5 to 0.5.
; The x component of the (x,y) pairs gives the gradient of the remapping function
; The y component of the (x,y) pairs gives the intercept of the remapping function
; The tables that follow the first also contain the remapping parameters up until
; tablenumber = number of neurons in the input layer. (Normalisation)
    TABLE 1 450 0
    TABLE 2 450 0
    TABLE 3 450 0
; Tables 4 to 12 contain the network weights.
; Notes
; The table number corresponds to the neuron number and is found through
; lookup of the number of neurons in the input layer stored in table 1
; The first 3 neurons being inputs have no preceedent weights thus
; the next table number corresponds to the number of the first neuron
; in the second layer of the neural network.
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
    TABLE 4 1 -11.4607 2 -12.43377 3 -14.72538
    TABLE 5 1 -0.79215 2 -0.36142 3 3.74302
    TABLE 6 1 1.61049 2 15.38415 3 7.23676
    TABLE 7 1 6.09981 2 -11.10187 3 4.00536
    TABLE 8 1 -1.75093 2 3.49071 3 -15.65733
    TABLE 9 1 -19.39757 2 -4.63068 3 2.34955
    TABLE 10 1 -11.50593 2 -0.79421 3 1.64819 4 6.08491 5 -1.76353 6 -19.41473
    TABLE 11 1 -12.40741 2 -0.36626 3 15.38124 4 -11.12053 5 3.51378 6 -4.60196
    TABLE 12 1 -14.73818 2 3.7435 3 7.25253 4 3.98477 5 -15.6629 6 2.36838
; The next 3 tables contain the parameters
; to map from ANN space to stress space (Denormalisation).
    TABLE 13 3.333333e6 0
    TABLE 14 3.333333e6 0
    TABLE 15 3.333333e6 0
; The next table contains the neuron thresholds (biases).
; Notes:
; The first 3neurons (being inputs) have no bias.
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
;
```

```
    TABLE thresh  1 0 2 0 3 0
    TABLE thresh  4 -0.00032 5 0.00739 6 0.00011 7 0.00026 8 -0.00088 9 0.00023
    TABLE thresh  10 0.00084 11 0.00104 12 -0.01212
;
; The next table contains the neuron activations.
; Notes:
; The first 3 pairs correspond to input values.
; The last 3 pairs correspond to input values.
; All values initialised at zero to begin.
;
    TABLE act  1 0 2 0 3 0
    TABLE act  4 0 5 0 6 0 7 0 8 0 9 0
    TABLE act  10 0 11 0 12 0
;
  END_COMMAND
END
Initialise_Network
```

B6.4 *FLAC (FISH)* linear elastic ANN-UDM file (ANN-ILE.fls) used in Section 6.2.3.

```
; FISH version of isotropic elastic ANN model with
; forward strain tensor transformation and
; backward stress increment tensor transformation
;set echo off

def ANNILE
 constitutive_model
 f_prop  m_g m_k
 f_prop  m_e1 m_e2 m_sh2
 float $e11i $e22i $e12i $e33i $edif $e0 $erad $e1 $e2 $e3
 float $zs11 $zs22 $zs12 $zs33
 float $ei $eii $pedif $cs2 $si2 $dc2 $dss
 int  $icase

 Case_of  mode
; ----------------------
; Initialisation section
; ----------------------
  Case 1
   m_e1   = m_k + 4.0 * m_g / 3.0
   m_e2   = m_k - 2.0 * m_g / 3.0
   m_sh2  = 2.0 * m_g
  Case 2
; ---------------
; Running section
; ---------------
; --- get strain components in global cartesian axes ----
;
    $e11i = zde11
    $e22i = zde22
    $e33i = zde33
    $e12i = zde12
;
; --- find principal strain components
;
    $edif = $e11i - $e22i
    $e0   = 0.5 * ($e11i + $e22i)
    $erad = 0.5 * sqrt ($edif*$edif + 4.0 * $e12i*$e12i)
```

```
    $ei   = $e0 - $erad
    $eii  = $e0 + $erad
    $pedif = $ei - $eii
;
; --- determine case ---
;
    section
     if $e33i > $eii then
; --- e33 is major p.str. ---
      $icase = 3
      $e1   = $ei
      $e2   = $eii
      $e3   = $e33i
      exit section
     end_if
     if $e33i < $ei then
; --- e33 is minor p.str. ---
      $icase = 2
      $e1   = $e33i
      $e2   = $ei
      $e3   = $eii
      exit section
     end_if
; --- e33 is intermediate ---
     $icase = 1
     $e1   = $ei
     $e2   = $e33i
     $e3   = $eii
    end_section
;
; --- Compute principal stress increments from principal strain increments
;
; --- Using ANN code
;
;--- Load strain increments into ANN input layer
;--- Strain increments trained in the ANN adopted the geomechanics convention,
;--- whereas in FLAC tensile strains are positive.
    ytable(act,1)=-$e1
    ytable(act,2)=-$e2
    ytable(act,3)=-$e3

;--- Insert the ANN Feedforward Code Here. Uses principal stresses and strains ---

    last_in_prevlayer=0
    loop layer (2, nlayers)
;
; Remap the input values to interval -0.5 to 0.5
;
      if layer = 2 then
        loop neuron (1,ytable(struct,layer-1))
          ytable(act,neuron)=xtable(neuron,1)*ytable(act,neuron)
       ytable(act,neuron)=ytable(act,neuron)+ytable(neuron,1)
        end_loop
      end_if
;
; Main multilayer perceptron feedforward calculation
;
      last_in_prevlayer=last_in_prevlayer+ytable(struct,layer-1)
      first_in_layer=last_in_prevlayer+1
```

```
        last_in_layer=first_in_layer+ytable(struct,layer)-1

        loop neuron (first_in_layer,last_in_layer)
          sum=0.0
          weight=0
          loop input(first_in_layer-ytable(struct,layer-1),first_in_layer-1)
            weight=weight+1
            sum=sum+ytable(neuron,weight)*ytable(act,input)
          end_loop
          sum=sum+ytable(thresh,neuron)
          ytable(act,neuron)=(1/(1+exp(-sum)))-0.5
        end_loop
      end_loop
;
; Remap the outputs of the feedforward calculation into stress space
;
      count=0
      loop i (nunits-ytable(struct,nlayers)+1,nunits)
        count=count+1
        ytable(act,i)=ytable(act,i)*xtable(nunits+count,1)
        ytable(act,i)=ytable(act,i)+ytable(nunits+count,1)
      end_loop
;
; End of ANN code
;
; Output the new principal stresses from the ANN
      $zds1 = -ytable(act,nunits-2)
      $zds2 = -ytable(act,nunits-1)
      $zds3 = -ytable(act,nunits)
;
;  LINEAR ELASTIC CONSTITUTIVE LAW : FOR REFERENCE ONLY
;
;     $zds1 = $e1 * m_e1 + $e2 * m_e2 + $e3 * m_e2
;     $zds2 = $e1 * m_e2 + $e2 * m_e1 + $e3 * m_e2
;     $zds3 = $e1 * m_e2 + $e2 * m_e2 + $e3 * m_e1
;
; --- Transform principal stress increments back to global axes
;
      section
; --- direction cosines ---
        if $pedif = 0.0 then
          $cs2  = 1.0
          $si2  = 0.0
        else
          $cs2  = $edif     / $pedif
          $si2  = 2.0 * $e12i / $pedif
        end_if
; --- resolve back to global axes ---
        case_of  $icase
          case 1
            $dc2  = ($zds1 - $zds3) * $cs2
            $dss  =  $zds1 + $zds3
            $zs11  = 0.5 * ($dss + $dc2)
            $zs22  = 0.5 * ($dss - $dc2)
            $zs12  = 0.5 * ($zds1  - $zds3) * $si2
            $zs33  = $zds2
          case 2
            $dc2  = ($zds2 - $zds3) * $cs2
            $dss  =  $zds2 + $zds3
```

```
        $zs11  = 0.5 * ($dss + $dc2)
        $zs22  = 0.5 * ($dss - $dc2)
        $zs12  = 0.5 * ($zds2  - $zds3) * $si2
        $zs33  = $zds1
       case 3
        $dc2  = ($zds1 - $zds2) *$cs2
        $dss  =  $zds1 + $zds2
        $zs11  = 0.5 * ($dss + $dc2)
        $zs22  = 0.5 * ($dss - $dc2)
        $zs12  = 0.5 * ($zds1  - $zds2) * $si2
        $zs33  = $zds3
      end_case
    end_section
;
; --- Add the transformed stress increments to old stress tensor
;
  zs11 = zs11 + $zs11
  zs22 = zs22 + $zs22
  zs33 = zs33 + $zs33
  zs12 = zs12 + $zs12

  Case 3
; ---------------------
; Return maximum modulus
; ---------------------
    cm_max = m_k + 4.0 * m_g / 3.0
    sm_max = m_g
  Case 4
; ---------------------
; Add thermal stresses
; ---------------------
    ztsa = ztea * m_k
    ztsb = zteb * m_k
    ztsc = ztec * m_k
    ztsd = zted * m_k
 End_case
end
```

B7 *FLAC/FISH* code listings for ANN UDM modelling Crosslands Sandstone

B7.1 Verification Test Type 1, Case 3 (Section 5.4.3)

Cases 1 and 2 are simplifications of Case 3, as indicated in Appendix B4. Thus only the listing for Case 3 is presented here.

```
;
; UDM test simulations
;
title
UDM Test Type 1 - ANN UDM, Case 3 : cases 1 & 2 combined
set large
set gravity 0 0
grid 1 1
;
; Set up for a right cylinder under TX compression
;
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic
prop den 2500 bulk 26e9 shear 16e9

def getKandG
; Note parameters for Crosslands Sandstone
  YoungsMod = 30e9
  Poissons = 0.15

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
end
initialise


model elas i = 1,halfizones
prop den 2450 bulk BulkMod shear ShearMod i=1,halfizones

; Upload the ANN weights, thresholds etc into FLAC tables
call net1-60K.dat
; Upload the ANN UDM
call ANNUDMS\NEWNET1.fis

model NET1 i = halfizonesplus1 ,myizones
; The elastic properties are dummy for GPM computation only.
prop den 2450 m_k BulkMod m_g ShearMod i= halfizonesplus1 ,myizones
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
ini xv 0 j=1
ini xv 1e-6 j=myjgp
ini yv 0 j=1
ini yv -1e-5 j=myjgp

; The following commands fix the velocity of the selected gridpoints
fix x i=1
fix x i=myigp
fix y j=1
fix y j=myjgp

call monitor.dat

his nstep 1
his unbal
his ey
his ex
his exy
; Strain histories
;his $md_e11 i=1 j=1
;his $md_e22 i=1 j=1
;his $md_e33 i=1 j=1
;his $md_e12 i=1 j=1
; Stress histories
his sxx i=1 j=1
his syy i=1 j=1
his szz i=1 j=1
his sxy i=1 j=1

step 100

set plot po 0 0 0.55 0.55 0
set output t1-cl-r1.ps
plot pen hist 5 6 7 8 vs 2
set output t1-cl-r2.ps
plot pen hist 8 vs 4
```

B7.2 Verification Test Type 2 (Section 5.4.4)

```
;
; UDM test simulations
;
title
UDM Test Type 2 - Crosslands ANN UDM
set large

def getKandG
; Note parameters for Crosslands Sandstone
  YoungsMod = 30e9
  Poissons = 0.15

  ShearMod = YoungsMod / (2.0 * (1+Poissons) )
  BulkMod = YoungsMod / (3.0 * (1.0 - 2.0 * Poissons) )
end
getKandG
;
; Set up for a right cylinder under TX compression
;
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
grid 10 10
gen 0.000,0.000 0.000,0.140 0.050,0.140 0.050,0.000
model elastic
prop den 2450 bulk BulkMod shear ShearMod

def initialise
  myizones = izones
  myjzones = jzones
  myigp = igp
  myjgp = jgp
  halfizones = myizones / 2
  halfjzones = myjzones / 2
  halfigp = igp/2
  halfjgp = jgp/2
  halfizonesplus1 = halfizones + 1
  halfjzonesplus1 = halfjzones + 1
  Ax = 1
     Ay = 1
  Bx = 1
     By = 1
end
initialise

; Upload the ANN weights, thresholds etc into FLAC tables
call net1-60K.dat
; Upload the ANN UDM
call ANNUDMS\NEWNET1.fis
model NET1
;model elastic
prop den 2450 m_k BulkMod m_g ShearMod
;prop den 2450 bulk BulkMod shear ShearMod
;
; Initial conditions
;
; Velocities
;
ini xv 0 j=1
ini xv 0 j=myigp
ini yv 0 j=1
ini yv -1e-8 j=myjgp
;
; Stress
;
ini sxx = -35e6
ini syy = -35e6
ini szz = -35e6
;
; Boundary conditions
;
; Velocities
;
fix y j=1
fix y j=myjgp
;
; Pressures
;
apply sxx -35e6 i=1 j=1,myjgp
apply sxx -35e6 i=myigp j=1,myjgp
;
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
; Functions defined to assist model monitoring
;
call MONITOR.DAT
;
; Numerical servo control procedure
;
call SERVO.DAT
set high_unbal=5.0e4
set low_unbal=2.5e4
;
; Histories
;
his nstep 1
his unbal
his ey
his sigmay
his yv i=halfigp j = myjgp
his sxx i=3 j=6
his sxx i=8 j=6
his syy i=3 j=6
his syy i=8 j=6
his szz i=3 j=6
his szz i=8 j=6
his sxy i=3 j=6
his sxy i=8 j=6

step 8000

set plot po 0 0 0.55 0.55 0
;
; Produce a plot of the model
;
set output t2m.ps
plot pen model grid fix app disp
;
; Produce a plot of the normal stress component histories
;
set output t2rsign.ps
plot pen hist 5 7 9 11 vs 2
```

B7.3 ANN UDM initialisation file Net1-60k.dat (Trained ANN weights for NET1 ANN UDM format)

```
;THIS FILE INITIALISES THE BASIC NETWORK STRUCTURE AND SPECIFIES
;THE WEIGHTS AND THRESHOLDS THAT HAVE BEEN PRODUCED IN TRAINING
;THE NEURAL NETWORK EXTERNALLY. IN ADDITION, PLACES FOR THE INPUT MAPPING
;PARAMETERS ARE SPECIFIED AS WELL AS THE OUTPUT REMAPPING PARAMETERS.
;File produced with automatic FLAC ANN initialisation file write routine
;Written by D.L. Millar
;
    Def Initialise_Network
    ; Initial ANN structural details
      inputs = 7
      outputs = 3
      nlayers = 3
      nunits = 17
    ; Number of the table containing thresholds
      thresh=nunits+outputs+1
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
; Number of the table containing activations
  act=nunits+outputs+2
; Number of the table containing the network structure
  struct=nunits+outputs+3
;
;
;
; The first table defined contains the network structure.
; Notes:
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
; This table is initialised first to make information available on the network structure
; available but as the table numbers are used in indexing to access weights, thresholds
; and activations, the table number is assigned a high value.
;
  COMMAND
    TABLE struct 1 7 2 7 3 3
;
; Table 1 contains the remapping parameters for the input quantities
; to ANN space for the first input neuron.
; Notes:
; This is a linear remapping between the maximum and minimum values for sigma1
; within the training dataset and the interval -0.5 to 0.5.
; The x component of the (x,y) pairs gives the gradient of the remapping function
; The y component of the (x,y) pairs gives the intercept of the remapping function
; The tables that follow the first also contain the remapping parameters up until
; tablenumber = number of neurons in the input layer.
    TABLE  1 9.000000e+00 0.000000e+00
    TABLE  2 9.000000e-10 0.000000e+00
    TABLE  3 9.000000e-10 0.000000e+00
    TABLE  4 9.000000e-10 0.000000e+00
    TABLE  5 4.500000e+03 0.000000e+00
    TABLE  6 4.500000e+03 0.000000e+00
    TABLE  7 4.500000e+03 0.000000e+00
; Tables 8 to 17 contain the network weights.
; Notes
; The table number corresponds to the neuron number and is found through
; lookup of the number of neurons in the input layer stored in table 1
; The first 7 neurons being inputs have no preceedent weights thus
; the next table number corresponds to the number of the first neuron
; in the second layer of the neural network.
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
;
;
    TABLE 8 1 4.662140e+00
    TABLE 8 2 2.098560e+00
    TABLE 8 3 -2.025400e-01
    TABLE 8 4 -5.825200e-01
    TABLE 8 5 1.142240e+00
    TABLE 8 6 9.782300e-01
    TABLE 8 7 8.189600e-01
;
;
    TABLE 9 1 3.325660e+00
    TABLE 9 2 1.946690e+00
    TABLE 9 3 1.459340e+00
    TABLE 9 4 -3.883390e+00
    TABLE 9 5 6.816650e+00
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
        TABLE 9 6 6.748540e+00
        TABLE 9 7 -4.199690e+00
    ;
    ;
        TABLE 10 1 3.820310e+00
        TABLE 10 2 2.243920e+00
        TABLE 10 3 -3.770620e+00
        TABLE 10 4 1.407070e+00
        TABLE 10 5 6.578210e+00
        TABLE 10 6 -4.173450e+00
        TABLE 10 7 6.807180e+00
    ;
    ;
        TABLE 11 1 2.572080e+00
        TABLE 11 2 -3.118790e+00
        TABLE 11 3 2.525410e+00
        TABLE 11 4 2.617000e-02
        TABLE 11 5 -7.168500e-01
        TABLE 11 6 2.327800e+00
        TABLE 11 7 1.058740e+00
    ;
    ;
        TABLE 12 1 -3.218970e+00
        TABLE 12 2 4.113340e+00
        TABLE 12 3 -1.931420e+00
        TABLE 12 4 -1.190760e+00
        TABLE 12 5 4.391240e+00
        TABLE 12 6 -6.925800e+00
        TABLE 12 7 -6.391550e+00
    ;
    ;
        TABLE 13 1 -2.448410e+00
        TABLE 13 2 -2.338000e-01
        TABLE 13 3 2.033040e+00
        TABLE 13 4 -1.776940e+00
        TABLE 13 5 -1.275290e+00
        TABLE 13 6 6.292600e-01
        TABLE 13 7 -1.430820e+00
    ;
    ;
        TABLE 14 1 1.005740e+00
        TABLE 14 2 1.317290e+00
        TABLE 14 3 9.952800e-01
        TABLE 14 4 -2.874290e+00
        TABLE 14 5 1.769720e+00
        TABLE 14 6 9.513500e-01
        TABLE 14 7 -1.158000e+00
    ;
    ;
        TABLE 15 1 -3.558870e+00
        TABLE 15 2 4.463700e+00
        TABLE 15 3 4.221370e+00
        TABLE 15 4 2.005500e+00
        TABLE 15 5 1.774900e+00
        TABLE 15 6 1.219780e+00
        TABLE 15 7 -2.294440e+00
    ;
    ;
        TABLE 16 1 -1.487020e+00
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
    TABLE 16 2 4.103860e+00
    TABLE 16 3 -1.601400e+00
    TABLE 16 4 -3.636250e+00
    TABLE 16 5 -4.484160e+00
    TABLE 16 6 -1.413990e+00
    TABLE 16 7 -7.872200e-01
;
;
    TABLE 17 1 -1.388430e+00
    TABLE 17 2 -1.789820e+00
    TABLE 17 3 4.313880e+00
    TABLE 17 4 -1.415000e+00
    TABLE 17 5 -3.976880e+00
    TABLE 17 6 2.745780e+00
    TABLE 17 7 2.326310e+00
; The next 3 tables contain the parameters
; to map from ANN space to stress space.
    TABLE 18 2.222222e+06 0.000000e+00
    TABLE 19 2.222222e+06 0.000000e+00
    TABLE 20 2.222222e+06 0.000000e+00
; The next table contains the neuron thresholds (biases).
; Notes:
; The first 7 neurons (being inputs) have no bias.
; The x values of the (x,y) pairs are the sequence number in the table.
; This is intended as an aide-memoire.
;
    TABLE thresh 1 0.000000e+00
    TABLE thresh 2 0.000000e+00
    TABLE thresh 3 0.000000e+00
    TABLE thresh 4 0.000000e+00
    TABLE thresh 5 0.000000e+00
    TABLE thresh 6 0.000000e+00
    TABLE thresh 7 0.000000e+00
    TABLE thresh 8 -1.238330e+00
    TABLE thresh 9 -1.787350e+00
    TABLE thresh 10 -1.788660e+00
    TABLE thresh 11 -7.997000e-01
    TABLE thresh 12 1.646980e+00
    TABLE thresh 13 8.934700e-01
    TABLE thresh 14 -7.230800e-01
    TABLE thresh 15 1.250540e+00
    TABLE thresh 16 1.464660e+00
    TABLE thresh 17 1.426120e+00
;
; The next table contains the neuron activations.
; Notes:
; The first 7 pairs correspond to input values.
; The last 3 pairs correspond to input values.
; All values initialised at zero to begin.
;
    TABLE act 1 0
    TABLE act 2 0
    TABLE act 3 0
    TABLE act 4 0
    TABLE act 5 0
    TABLE act 6 0
    TABLE act 7 0
    TABLE act 8 0
    TABLE act 9 0
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
      TABLE act 10 0
      TABLE act 11 0
      TABLE act 12 0
      TABLE act 13 0
      TABLE act 14 0
      TABLE act 15 0
      TABLE act 16 0
      TABLE act 17 0
;
  END_COMMAND
END
Initialise_Network
```

B7.4 NET1 format ANN UDM for Croslands Sandstone (Section 5.4)

Note that this listing is almost identical to B6.4 however volumetric strain is included here as an additional history parameter. Provided here for clarity, not compactness.

```
set echo off
; FISH ANNUDM
; Using volumetric strain as a history parameter.
; Using stress tensor components as state parameters.
; Using strain increment components as updating variables.
; ANN produces new principal stress component increments.
; Principal stress components increments are transformed to global axes.
; Transformed increments are added to old global stress tensor.

def NET1
  constitutive_model 81
  f_prop  m_g m_k
  f_prop  m_e1 m_e2 m_sh2
;
; Declare properties to store the volumetric strain
;
  f_prop  $md_vol
;
; --- Declare variables needed for strain increment transformation
; --- to principal axes
;
  float $dei $deii $pdedif $dedif $de0 $derad
  float $de1 $de2 $de3
  float $cs2 $si2 $dc2 $dss
  float $fdc2 $fdss $sxy
;
; --- Declare variables needed for state parameters
;
  float $si $sii $psdiff $sdiff $s0 $srad
  float $s1 $s2 $s3
;
; --- Declare variables for output
;
  float $zds1 $zds2 $zds3
  float $zs11 $zs22 $zs12 $zs33
;
; --- Declare variables to accumulate volumetric strain
;
  float $m_vol
;
; --- Declare other variables
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
;
 int  $icase

 Case_of mode
; ---------------------
; Initialisation section
; ---------------------
  Case 1
;
; --- Initialise variables needed in gpm calculation
;
    m_e1   = m_k + 4.0 * m_g / 3.0
    m_e2   = m_k - 2.0 * m_g / 3.0
    m_sh2  = 2.0 * m_g
; ---------------
; Running section
; ---------------
  Case 2
;
; --- GET PRINCIPAL STRAIN INCREMENT COMPONENTS
;
    $dedif  = zde11 - zde22
    $de0    = 0.5 * (zde11 + zde22)
    $derad  = 0.5 * sqrt ($dedif*$dedif + 4.0 * zde12*zde12)
    $dei    = $de0 - $derad
    $deii   = $de0 + $derad
    $pdedif = $dei - $deii
;
; --- determine case ---
;
    section
     if zde33 > $deii then
; --- e33 is major p.str. ---
       $icase = 3
       $de1   = $dei
       $de2   = $deii
       $de3   = zde33
       exit section
     end_if
     if zde33 < $dei then
; --- e33 is minor p.str. ---
       $icase = 2
       $de1   = zde33
       $de2   = $dei
       $de3   = $deii
       exit section
     end_if
; --- e33 is intermediate ---
     $icase = 1
     $de1   = $dei
     $de2   = zde33
     $de3   = $deii
    end_section

; --- direction cosines ---
     if $pdedif = 0.0 then
       $cs2  = 1.0
       $si2  = 0.0
     else
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
        $cs2  = $dedif     / $pdedif
        $si2  = 2.0 * zde12 / $pdedif
      end_if
;
; --- GET PRINCIPAL STRESS COMPONENTS
;
      $fdc2 = (zs11 - zs22) *$cs2
      $fdss = zs11 + zs22
      $s1 = 0.5 * ($fdss + $fdc2) + zs12 * $si2
      $s2 = 0.5 * ($fdss - $fdc2) - zs12 * $si2
      $sxy = zs12 * cs2 + 0.5 * (zs11 - zs22) * $si2
      $s3 = zs33
;
; --- GET PRINCIPAL STRESS COMPONENTS - OLD
;
;     $sdif = zs11 - zs22
;     $s0   = 0.5 * (zs11 + zs22)
;     $srad = 0.5 * sqrt ($sdif*$sdif + 4.0 * zs12*zs12)
;     $si   = $s0 - $srad
;     $sii  = $s0 + $srad
;     $psdiff = $si - $sii
;
; --- determine case ---
;
;     section
;       if zs33 > $sii then
; --- zs33 is major p.str. ---
;         $s1   = $si
;         $s2   = $sii
;         $s3   = zs33
;         exit section
;       end_if
;       if zs33 < $si then
; --- zs33 is minor p.str. ---
;         $s1   = zs33
;         $s2   = $si
;         $s3   = $sii
;         exit section
;       end_if
; --- zs33 is intermediate ---
;       $s1   = $si
;       $s2   = zs33
;       $s3   = $sii
;     end_section

;
; ---------------------------------------------------------------
;
; --- Compute principal stress increments from principal strain increments
; --- Using ANN code
;
;--- Load volumetric strain, principal stress state and strain increments into ANN input layer
;--- Stresses and strain learnt in the ANN adopted the geomechanics convention,
;--- whereas in FLAC tensile stress and strain are positive.
;
    ytable(act,1)=-$md_vol
    ytable(act,2)=-$s1
    ytable(act,3)=-$s2
    ytable(act,4)=-$s3
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
        ytable(act,5)=-$de1
        ytable(act,6)=-$de2
        ytable(act,7)=-$de3

        last_in_prevlayer=0
        loop layer (2, nlayers)
    ;
    ; --- Normalise input quantities
    ;
          if layer = 2 then
            loop neuron (1,ytable(struct,layer-1))
              ytable(act,neuron)=xtable(neuron,1)*ytable(act,neuron)
           ytable(act,neuron)=ytable(act,neuron)+ytable(neuron,1)
            end_loop
          end_if
    ;
    ; --- Start of main MLP feedforward calculation
    ;
          last_in_prevlayer=last_in_prevlayer+ytable(struct,layer-1)
          first_in_layer=last_in_prevlayer+1
          last_in_layer=first_in_layer+ytable(struct,layer)-1

          loop neuron (first_in_layer,last_in_layer)
            sum=0.0
            weight=0
            loop input(first_in_layer-ytable(struct,layer-1),first_in_layer-1)
              weight=weight+1
              sum=sum+ytable(neuron,weight)*ytable(act,input)
            end_loop
            sum=sum+ytable(thresh,neuron)
            ytable(act,neuron)=(1/(1+exp(-sum)))-0.5
          end_loop
        end_loop
    ;
    ; --- End of ANN feed forward code
    ;
    ;
    ; --- Denormalise outputs
    ;
        count=0
        loop i (nunits-ytable(struct,nlayers)+1,nunits)
          count=count+1
          ytable(act,i)=ytable(act,i)*xtable(nunits+count,1)
          ytable(act,i)=ytable(act,i)+ytable(nunits+count,1)
        end_loop
    ;
    ; --- Assign the new principal stress increments from the ANN output
    ; --- remembering sign convention difference.
    ;
        $zds1 = -ytable(act,nunits-2)
        $zds2 = -ytable(act,nunits-1)
        $zds3 = -ytable(act,nunits)
    ;
    ; --- End of ANN code
    ;
    ; ------------------------------------------------------------------
    ;
    ; --- Transform principal stress increments back to global axes
    ;
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
; --- resolve back to global axes ---
    case_of $icase
      case 1
       $dc2 = ($zds1 - $zds3) * $cs2
       $dss = $zds1 + $zds3
       $zs11 = 0.5 * ($dss + $dc2)
       $zs22 = 0.5 * ($dss - $dc2)
       $zs12 = 0.5 * ($zds1 - $zds3) * $si2
       $zs33 = $zds2
      case 2
       $dc2 = ($zds2 - $zds3) * $cs2
       $dss = $zds2 + $zds3
       $zs11 = 0.5 * ($dss + $dc2)
       $zs22 = 0.5 * ($dss - $dc2)
       $zs12 = 0.5 * ($zds2 - $zds3) * $si2
       $zs33 = $zds1
      case 3
       $dc2 = ($zds1 - $zds2) *$cs2
       $dss = $zds1 + $zds2
       $zs11 = 0.5 * ($dss + $dc2)
       $zs22 = 0.5 * ($dss - $dc2)
       $zs12 = 0.5 * ($zds1 - $zds2) * $si2
       $zs33 = $zds3
    end_case

;
; --- Add the transformed stress increments to old stress tensor
;
  zs11 = zs11 + $zs11
  zs22 = zs22 + $zs22
  zs33 = zs33 + $zs33
  zs12 = zs12 + $zs12
;
; --- Accumulate volumetric strain
;
  $m_vol=$m_vol + zde11 + zde22 + zde33

  if zsub > 0.0 then
  $md_vol = $md_vol + $m_vol / zsub
    $m_vol = 0.0
  end_if
; ----------------------
; Return maximum modulus
; ----------------------
  Case 3
;
; --- Needed for gpm calculation
;
    cm_max = m_k + 4.0 * m_g / 3.0
    sm_max = m_g
; ----------------------
; Add thermal stresses
; ----------------------
  Case 4
;
; --- Not used, but required
;
    ztsa = ztea * m_k
```

Appendix B7: *FLAC/FISH* code listings for ANN UDM modelling Croslands Sandstone

```
        ztsb = zteb * m_k
        ztsc = ztec * m_k
        ztsd = zted * m_k
    End_case
end
opt NET1

set echo on
```

C1 Image processing fundamentals

This description is restricted to 'black and white' or grey-scale image processing, to remove unnecessary complications associated with independent treatment of, say, red, green and blue channels of full colour images. The main sources for the material presented in this Section are the introductory texts on digital image processing by Castleman, 1979, and Gonzalez and Wintz, 1983.

C1.1 Digital Images

A digital image is composed of an array of picture elements called pixels, each of which is associated with (generally) an integer value which represents the intensity of light which originally fell on the sensor used to capture the image.

The majority of black and white image processing, images use a single byte (8 bits) to represent the integer value of each pixel that makes up the image. Thus the 'colour' of each pixel can be any one of a fixed range of 2^8 (256) different shades of grey with zero corresponding to the 'colour' black and 255 corresponding to the 'colour' white. Figure C1.1 illustrates this idea.

106 089 111 002 016 110 053 099 123 047 113 114
031 042 013 000 049 114 027 080 016 102 002 094
091 255 125 121 255 076 116 255 255 255 025 056
106 255 104 085 255 035 088 042 255 086 012 112
119 255 100 078 255 012 104 095 255 005 119 124
076 255 092 112 255 107 029 105 255 102 050 074
053 255 094 005 255 051 096 117 255 116 022 110
124 255 050 046 255 061 050 126 255 047 077 031
069 255 255 255 255 029 083 019 255 054 061 123
104 255 072 095 255 124 057 037 255 050 022 027
048 255 059 084 255 055 068 054 255 070 045 011
053 255 078 029 255 068 025 042 255 090 004 065
095 255 017 078 255 073 025 043 255 088 097 057
003 255 014 103 255 053 116 005 255 092 092 029
056 255 050 019 255 112 123 027 255 075 077 021
049 255 116 085 255 048 027 094 255 103 079 088
113 255 020 117 255 092 035 255 255 255 025 080
074 067 072 093 043 072 118 026 034 073 006 104
097 014 067 070 124 123 066 021 067 003 101 123
090 027 099 088 099 077 066 006 018 034 040 122
060 057 106 107 062 126 094 066 083 107 053 090
000 025 002 110 040 082 003 069 002 105 038 061

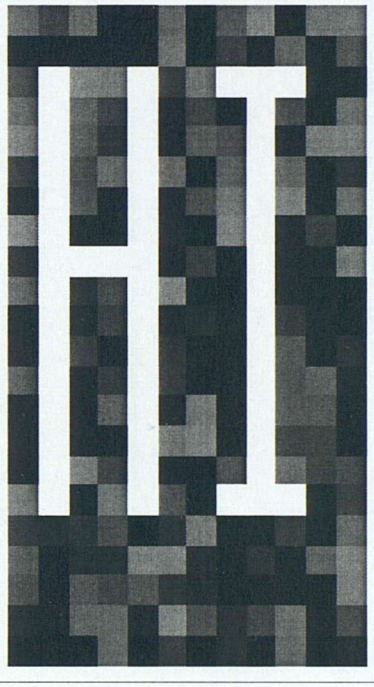


Figure C1.1: Showing an array of byte pixels and corresponding image which has been enlarged through pixel replication.

C1.2 Types of image processing operations

Generally, image processing modules fall into three categories:

(i) Intensity based transformations; those that transform the pixel values of an original image to pixel values in a new image, keeping spatial relations the same.

(ii) Geometric transformations; those that change the spatial relations of pixels within an original image while maintaining the pixel values.

(iii) Combinations of (i) and (ii).

Within this Appendix we will be mainly concerned with type (i) transformations, that is those that involve manipulation of the pixel values alone.

C1.3 Grey level histograms

Histograms of images are frequently used to examine the distribution of pixel values without reference to their positions within an image. Figure C1.2 shows the rock mass image that is the primary data source for Chapter 6. (This image has been used by other workers, notably Reid (1998), who have developed alternative methods for semi-automatic fracture recognition from digital imagery: use of the same data source permits ready comparison of methods.)

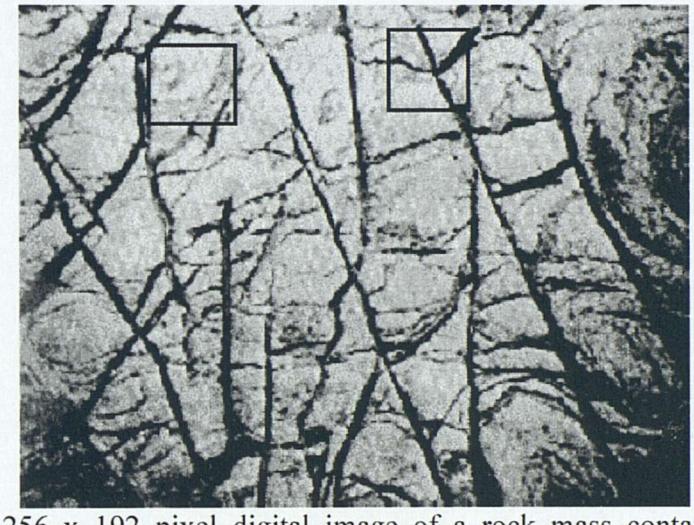


Figure C1.2: 256 x 192 pixel digital image of a rock mass containing fractures. Scanned using a flatbed scanner from a photograph shown in Whittaker, Singh and Sun (1992). Rectangles show the areas analysed in Figures C1.4 and C1.5.

Figure C1.2 is a 'clean' image of fractures within a rock mass. In comparison to the image shown in Figure 7.1 of the main text, the fractures are easily (visually) identified because there is great contrast between fractures (dark) and intact bodies of rock (light). Figure C1.3 shows the histogram of grey values in this image.

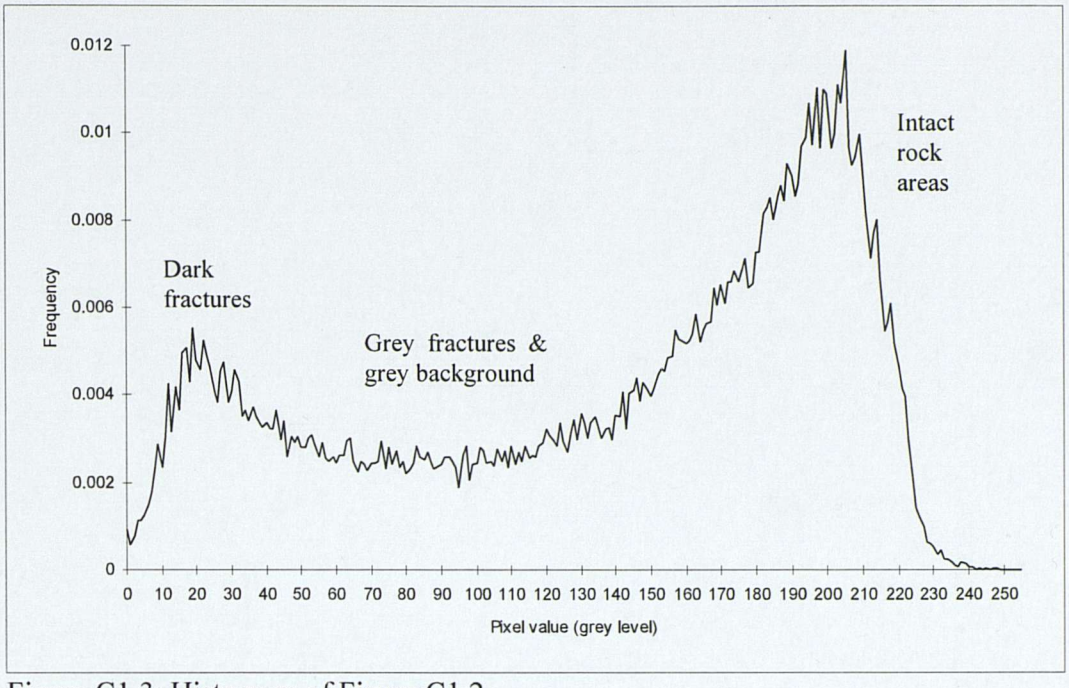


Figure C1.3: Histogram of Figure C1.2

The form of the histogram shows two peaks and is thus termed 'bi-modal'. The left hand peak corresponds to dark pixels and the right hand peak corresponds to light pixels. As the area under the histogram curve for the LHS peak is less than that of the RHS peak it can be said that there are on the whole more lighter pixels than dark. Many of the dark pixels within the LHS peak will correspond to locations within the image (Figure C1.2) where there are fractures; however, as is apparent from Figure C1.2, fairly large dark regions exist around the lower corners of the image and pixels in these areas also contribute to the height of the LHS peak in the histogram.

When a smaller scale sample (32 x 32 pixels) is taken from the original image (Figure C1.2), as is shown below in Figure C1.4, and its histogram is examined, it can be seen that a bi-modal histogram form persists. This is true generally for other image patches that contain significant fracture image content. In this instance it is more certain that the LHS peak corresponds to fracture pixels.

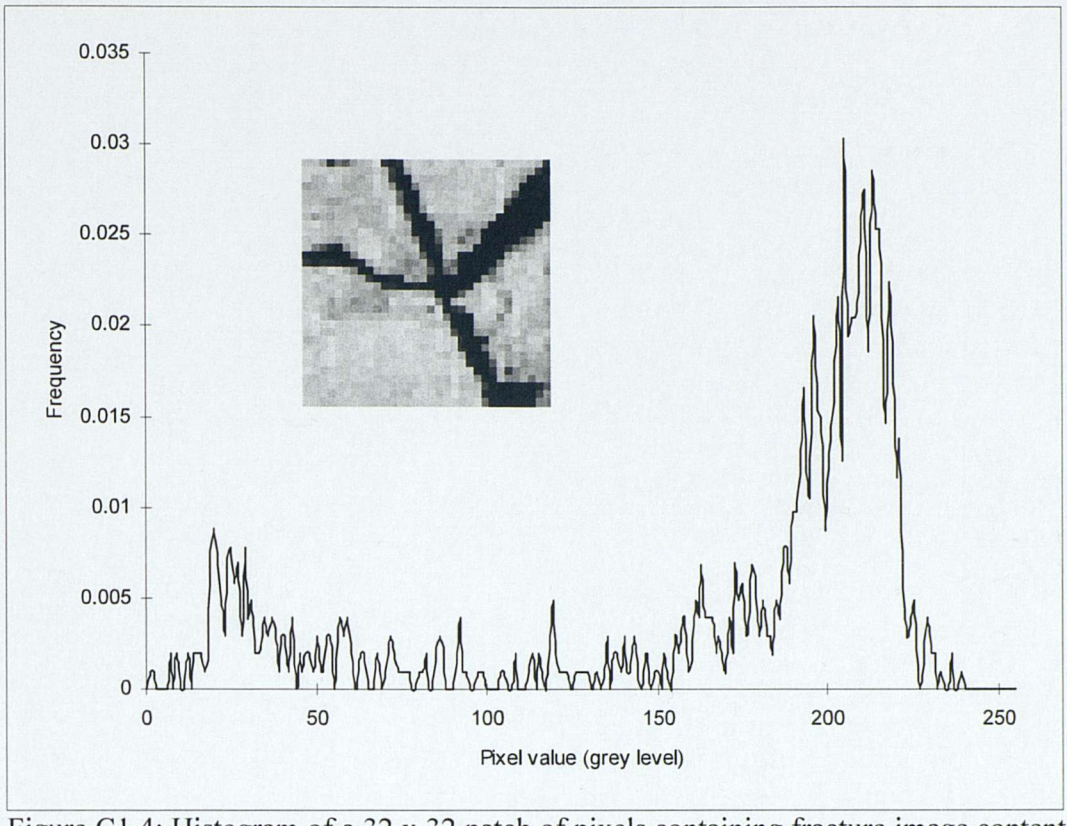


Figure C1.4: Histogram of a 32 x 32 patch of pixels containing fracture image content (shown as inset) taken from the digital image shown in Figure C1.2. The sample patch has been taken from just to the right of the middle of the top of the original image and resized through pixel replication for clarity.

For comparative purposes, a second patch of the same dimensions that has only minor fracture image content has been sampled from the original image. This is shown with its histogram in Figure C1.5 below. It can be seen that there is no significant LHS peak within the histogram, which corresponds well with the observation that no large fracture is visible within the patch. In general a uni-modal histogram corresponds to the absence of fracture image content. However the minor feature in the bottom right hand quadrant of the patch may be important in the geomechanical sense. If so, an automated fracture recognition module that was based on histogram characterisation alone would fail to identify the feature in this case.

The histogram operation ignores the spatial distribution of values. It considers the distribution of byte values alone. Consequently, one could envisage conducting a geostatistical analysis of image patch values and undertake image content recognition on the basis of semi-variogram classification. While this is feasible, this approach has been explored elsewhere (MacDonald and Carr, 1989) and is not the approach adopted here.

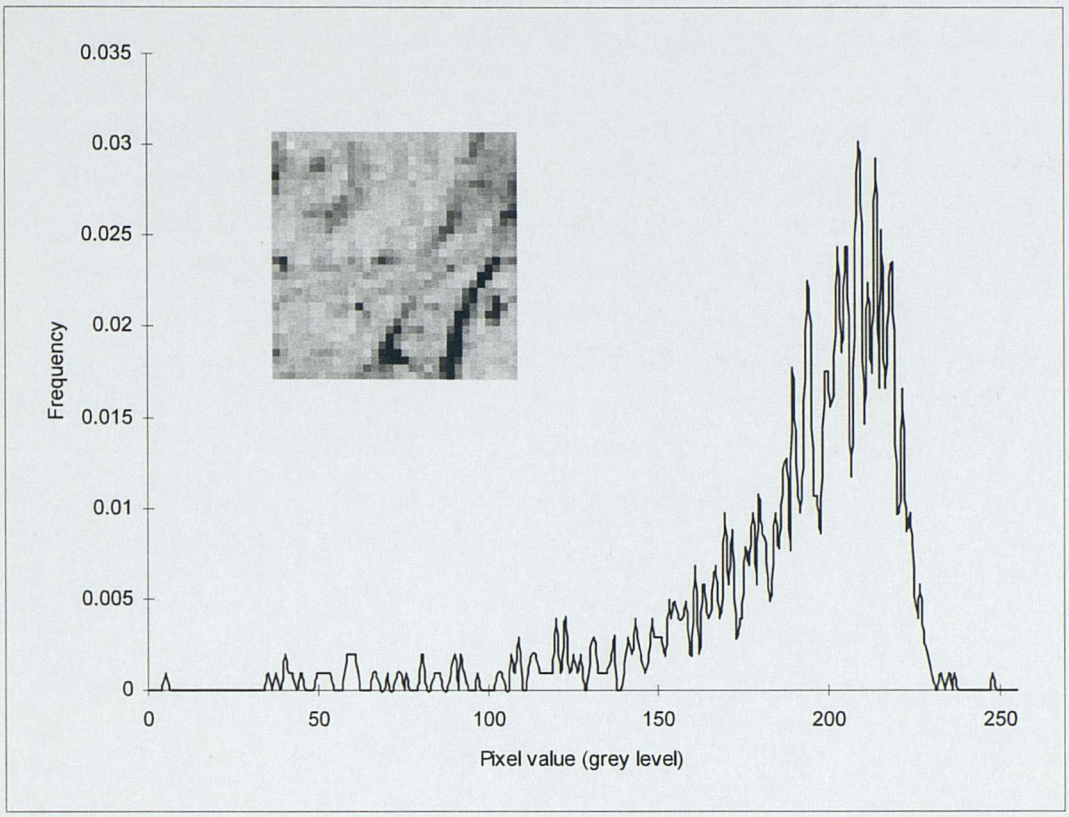


Figure C1.5: Histogram of a 32 x 32 patch of pixels that shows minor fracture image content (shown as inset) taken from the digital image shown in Figure C1.2. The sample patch has been taken from the middle of the top left quadrant of the original image and resized through pixel replication for clarity.

Through supervised learning, an ANN is trained to interpret the spatial and intensity based information contained within image patches to discriminate the image content directly. This model of processing is much closer to the way in which humans interpret what they see.

C1.4 Brightness and Contrast

Human eyes are capable of resolving at best around 25 to 30 different levels of grey (Gregory, 1967). Thus, subjective (i.e., human) perception of image quality does not depend so much on the number of different grey levels used within the image, but depends more on the quantities *brightness* and *contrast*. Figure C1.6 shows images of high and low brightness, and high and low contrast with their histograms. Very bright pictures have their histograms positioned to the right hand side of the grey level axis; low brightness images have histograms located towards the left hand side. Histograms of low contrast images occupy only a narrow range of grey levels of the available range; high contrast images occupy a wide range of the available grey levels.

For the purposes of this work, the brightness of an image is defined specifically to be the value of the lowest pixel grey level present within an image, expressed as a percentage of the available dynamic range. For an 8-bits per pixel image, a brightness value of 15% means that the lowest pixel value is 38; a brightness value of 75% means that the lowest pixel value is 192. In addition, the contrast of an image is defined to be the difference between the highest pixel value and the lowest, again expressed as a percentage of the available range. For an 8-bits per pixel image, a contrast value of 15% means that the difference between the highest pixel value and the lowest in the image is 38; a contrast value of 75% means that the same difference is 192.

With these terms defined, it can be seen from Figure C1.6 that, high contrast, low brightness images are perceived to be of the best quality.

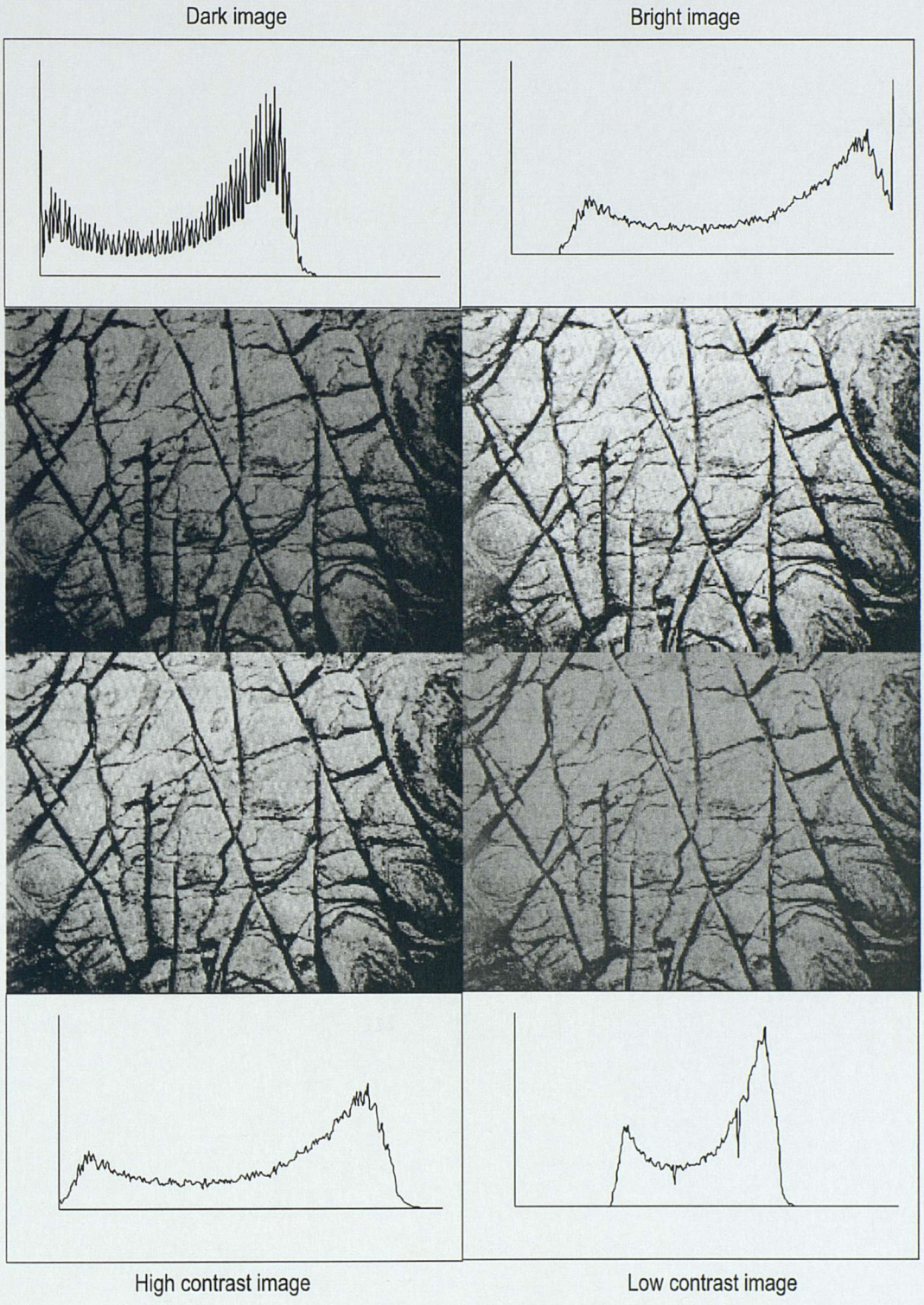


Figure C1.6: Showing examples of: (top left) dark imagery, brightness = 0%, contrast = 70.7%; (top right) bright imagery, brightness = 12.9%, contrast = 86.7%; (bottom left) high contrast, brightness = 0%, contrast = 96.9%; and, (bottom right) low contrast, brightness = 25%, contrast = 48.8%. Bottom right is brighter than top right, but has much lower contrast.

C2 Fracture Discriminator Weights and Thresholds

ANNs are effectively multivariate regression techniques. If results obtained using ANNs are to be utilised by workers other than those who originated the work, it is vital that the structure, ANN type and values of weights and thresholds are presented, just as the fitting parameters and quality of fit must be reported for a multivariate regression exercise. This is the reason why the following data are presented in this Appendix; they are required if anyone else other than the author wishes to implement the specific fracture discriminator discussed in the main text of Chapter 6.

Back Propagation network.

Number of layers = 3
layers numbered from 1 (input) to 3 (output)

Layer dimensions:
layer 1: (16, 16)
layer 2: (4, 4)
layer 3: (1, 1)

Cycles trained = 100

Weights initialized from -0.300 to 0.300

Noise = 0.00000, decay = 0.00000

Tolerance = 0.01000

Learning rates for layers:
layer 2: 0.10000
layer 3: 0.10000

Momentum terms for layers:
layer 2: 0.90000
layer 3: 0.90000

Biases for hidden and output layers:
layer 2 biases:
0.40689 -1.01591 -0.68846 0.67782 1.14935 -0.20743 0.39988 0.82801
-0.40488 -0.21164 0.02809 0.46964 0.16006 -0.92190 -0.42178 -0.94249
layer 3 biases:
-1.67427

Weight matrices: (to unit I from unit J, J changing fastest)
layer 1 to 2 weights:
-0.49964 0.03877 -0.27976 0.15994 0.10685 0.27605 0.30902 0.44260
0.07727 0.04252 -0.15807 0.46724 0.23026 -0.24485 -0.38076 -0.36659
-0.22995 -0.21901 -0.24800 -0.33569 0.20183 -0.15553 -0.03058 -0.18796
-0.44106 -0.15075 0.01147 0.08537 0.05405 -0.27795 -0.49277 0.00292
0.16394 -0.12688 -0.28671 -0.11073 -0.14148 -0.05204 -0.01101 -0.06995
0.03400 0.14199 0.38427 0.44317 -0.14231 -0.32454 0.05594 -0.18261
0.29344 -0.19470 0.21437 -0.15175 -0.13707 -0.07588 0.12820 0.07319
-0.40134 -0.11786 0.11164 -0.19135 -0.19607 -0.33134 0.17341 -0.31100
-0.02191 0.33244 -0.23381 -0.18843 -0.19620 -0.10626 0.43687 0.15187
-0.24527 0.05806 -0.00303 0.01010 -0.18943 0.18457 0.01081 -0.18983
0.13696 0.05115 0.29615 0.30090 0.15725 0.05054 -0.02544 0.05157
-0.06505 0.38836 0.16166 0.30697 -0.04845 -0.29284 -0.30662 0.16258
-0.17254 -0.19775 0.44406 0.36338 0.12959 0.48273 0.09986 0.74686
0.43312 0.22732 0.11537 -0.28155 0.25426 0.09192 -0.43729 0.30647
0.01567 -0.03910 -0.20404 0.00368 0.42966 0.44877 0.32360 0.52907
0.65278 0.50834 0.42576 0.09432 0.07761 0.06154 0.04592 0.14876
-0.34715 -0.09549 0.22727 -0.29077 0.09195 0.32161 0.35104 0.61997
1.11027 0.92961 0.37751 0.05165 -0.37818 -0.53328 0.16249 -0.29535

Appendix C2: Fracture Discriminator Weights & Thresholds

-0.11003 -0.27704 0.39728 0.00945 0.27918 0.12051 0.07880 0.79546
1.02484 0.83167 0.20695 0.25435 -0.52595 0.18744 0.17696 -0.21093
-0.12604 -0.30803 0.22307 0.04312 0.09700 -0.17908 0.29412 -0.12119
-0.03618 0.36864 -0.01075 0.27507 -0.15475 0.06351 -0.05797 0.28053
0.09038 0.07950 -0.04284 -0.04039 -0.12990 -0.09310 -0.54547 -0.39255
0.37141 0.15201 -0.14091 -0.21481 -0.02876 -0.27788 -0.10315 -0.02178
0.11549 0.21203 -0.03728 -0.42610 0.09476 -0.26354 0.00939 -0.15698
0.07313 -0.44107 -0.13959 -0.20731 0.05234 -0.22409 -0.01646 0.32281
0.13572 0.44337 -0.11112 -0.12455 -0.11320 0.55586 0.35408 0.55558
-0.36389 -0.15022 -0.34753 -0.35901 0.09635 -0.32209 0.41340 -0.14873
-0.12078 0.26417 0.13996 -0.03927 0.35682 0.33317 0.20446 -0.14884
-0.02432 -0.24443 0.08727 -0.11140 0.03311 -0.06650 -0.19978 0.16579
-0.12962 0.24235 0.11654 0.15356 -0.12035 -0.22453 -0.09727 -0.04462
-0.16912 -0.59648 -0.25001 -0.45153 -0.17918 -0.23697 0.04840 0.19788
0.32935 -0.21277 0.07458 0.19188 0.11572 -0.22800 -0.18875 -0.48181
0.13062 0.30562 -0.14398 -0.14532 -0.17596 -0.01605 -0.03698 0.28898
0.16640 0.25355 -0.20861 -0.04976 -0.19176 -0.20603 -0.08552 0.06116
0.09678 -0.04492 -0.03557 -0.15081 0.00574 0.27649 0.03129 -0.04311
-0.00189 -0.10395 0.18715 0.40028 0.44924 0.26155 -0.14431 0.20857
-0.12655 -0.29067 0.10757 -0.09906 0.21893 0.34647 -0.31996 0.20860
0.08026 -0.05652 -0.26294 0.31032 0.16256 0.13109 0.18616 -0.09111
0.30300 -0.22725 0.12361 -0.13845 0.12522 0.21130 -0.05716 -0.05302
0.02339 0.03754 0.20820 0.05641 -0.02753 0.30390 -0.36220 0.06410
0.41611 -0.18116 0.30907 0.01009 0.11181 0.25063 0.11510 -0.16606
0.29383 0.09262 -0.16622 -0.11299 -0.20746 0.15423 0.27785 -0.03226
-0.22980 -0.22910 -0.32493 0.11409 -0.00953 0.39577 0.27571 -0.01482
0.32541 0.25615 -0.38690 -0.33478 0.06571 0.10402 -0.20635 -0.48870
-0.31834 -0.04275 -0.10902 0.02014 0.04186 0.20164 0.37900 0.10935
0.19914 0.19859 0.28006 0.24495 -0.23677 -0.07030 -0.23948 -0.33677
-0.44423 -0.05290 -0.53643 0.02870 0.18264 0.00146 0.16492 -0.25415
-0.02027 -0.18110 0.22217 -0.13145 -0.13135 0.12821 0.11066 -0.24235
-0.33969 -0.26398 -0.15644 0.26184 0.24737 0.64776 -0.07809 0.00455
0.01487 0.41485 -0.35203 -0.33831 -0.07356 -0.01165 -0.30302 -0.44439
-0.24982 -0.72382 -0.22611 -0.12590 0.58886 -0.07216 -0.30960 -0.24697
-0.05299 0.35154 -0.06895 0.01997 0.32877 0.27222 0.02241 0.24200
0.00930 -0.15912 -0.11784 -0.23799 0.09462 0.26028 -0.08632 -0.21408
0.10066 -0.06757 0.18339 0.47018 0.01129 0.21073 -0.03560 0.43917
-0.07292 0.21324 0.01570 -0.23962 -0.09156 -0.03262 -0.32783 0.01620
-0.22121 0.04913 0.33172 -0.05439 0.15801 0.23489 0.27293 0.24330
-0.07782 0.33978 -0.10998 0.23772 0.14774 0.20168 -0.19473 0.24256
0.07534 0.00349 -0.22421 0.10542 0.25370 -0.40282 -0.38827 -0.41870
0.35863 0.23763 -0.05892 0.44059 0.00733 0.10690 -0.26807 -0.16258
-0.18410 -0.04672 0.08034 -0.34991 0.04117 -0.24514 -0.43368 0.24424
-0.18536 0.52487 0.27176 0.44368 -0.15795 0.09337 -0.12237 -0.04662
-0.28793 -0.11982 -0.21081 0.33914 -0.31934 -0.13008 -0.07793 -0.06930
-0.05682 0.44386 0.45585 0.40911 0.04921 0.40297 0.07118 0.00097
0.39257 -0.01066 0.31648 0.10270 -0.26521 -0.08896 -0.21925 -0.18626
0.33239 0.06228 0.03384 -0.09447 -0.17266 0.11557 0.09692 -0.22069
0.17613 -0.07864 -0.22292 0.01074 0.12209 -0.18326 -0.30464 -0.29365
0.18385 -0.00958 0.07408 0.13886 0.01420 0.34135 0.13691 0.26617
-0.04736 -0.16877 0.26783 0.34237 0.14389 0.43698 -0.17962 0.19400
0.15251 -0.04528 0.02463 0.05467 -0.11077 -0.05983 -0.20536 -0.05157
0.19815 0.14017 0.08998 0.48869 0.22971 0.18180 0.18735 -0.21717
0.10411 0.04934 -0.00076 0.10995 0.25702 0.16792 0.10204 -0.19155
-0.31399 -0.16659 0.19376 0.31553 0.11598 0.17163 -0.37994 0.23434
-0.01817 -0.15430 0.21058 -0.02402 0.23789 -0.19843 -0.18097 -0.31878
-0.29912 -0.13777 -0.18192 0.00484 -0.25936 -0.12346 -0.35999 -0.04119
0.14832 -0.31288 0.05308 0.08246 0.23083 0.17066 -0.19711 -0.05165
-0.13343 0.26734 -0.19283 0.08575 -0.03510 -0.08989 -0.07473 -0.21805
-0.28849 -0.26797 -0.33113 0.22717 -0.16259 -0.22597 0.03854 0.09540
-0.04912 -0.01631 0.02026 -0.38132 -0.16584 -0.28535 -0.27001 -0.19350
-0.40251 -0.46667 -0.12569 0.22765 0.09476 0.36612 -0.19620 -0.05311
0.29698 -0.03236 -0.10780 -0.08891 -0.34609 -0.08873 -0.33638 -0.84306
-0.85059 -0.45882 0.01450 0.18434 0.20161 0.46158 0.11057 0.30967
0.23374 0.16194 -0.00180 -0.32123 0.08922 -0.05956 -0.11943 -0.35851
-0.50042 -0.37865 -0.02619 -0.09601 -0.11310 -0.05972 -0.17402 -0.17682
0.22742 0.01917 0.22271 -0.19189 0.15796 0.07612 0.16581 -0.26386
-0.11659 0.12785 0.12677 0.28643 -0.16406 0.15990 0.09651 0.17217
0.26844 0.28919 0.12865 -0.04437 0.08389 -0.20041 0.02827 0.23676
0.03076 -0.13236 0.11173 -0.03142 -0.20129 0.18756 0.12912 -0.23717
-0.06953 -0.01143 0.03483 0.28954 -0.26273 0.20552 0.41126 -0.22213
-0.00338 0.31464 -0.07544 -0.11267 -0.09272 0.13284 0.22673 0.02999
-0.08244 0.05952 -0.17150 0.11778 -0.18869 -0.30787 0.10073 -0.25728

Appendix C2: Fracture Discriminator Weights & Thresholds

-0.23091 0.31362 -0.13347 -0.08210 -0.18090 -0.06401 0.00423 -0.03952
-0.07391 -0.11300 -0.37736 -0.04998 -0.01719 -0.23123 -0.01564 0.32468
0.06975 0.25592 -0.18219 0.17800 -0.10456 0.09337 0.39061 -0.11575
-0.28778 0.00136 0.05560 0.22413 0.21320 0.25584 -0.01028 0.32172
0.23640 0.48929 0.38382 0.29008 0.43808 0.47684 0.28146 0.01069
-0.18076 -0.25915 -0.15842 0.16970 0.00394 -0.04718 -0.07984 0.20097
0.09102 -0.31975 -0.25437 0.04483 0.13392 0.00543 -0.37292 -0.25989
-0.28425 -0.19924 -0.32726 0.19522 0.01774 -0.34140 0.33579 0.12519
-0.13605 -0.32484 -0.19656 -0.07292 0.06666 -0.04086 -0.45911 0.13174
-0.13259 0.04003 0.01280 0.16588 -0.21800 -0.29667 -0.15726 0.18903
-0.20033 -0.02741 0.08740 -0.16681 0.16345 0.09649 -0.15142 -0.03323
0.13979 0.25504 0.17066 -0.15235 -0.19486 0.04649 0.13191 -0.05806
-0.13815 -0.16477 -0.07783 0.15822 -0.25936 0.00099 0.04038 0.25360
0.19369 0.15873 -0.02607 -0.10008 0.00225 -0.00291 0.41011 -0.19519
-0.08761 0.35893 -0.07327 0.24395 -0.11482 -0.05547 -0.09710 0.28751
0.25261 -0.16809 -0.11778 -0.09609 0.16329 0.29691 0.15231 -0.18022
0.05896 -0.18800 0.08859 0.03436 -0.15523 -0.39604 0.25036 -0.15849
-0.07728 -0.27369 -0.06295 0.24360 0.03560 -0.03461 0.24421 0.20277
0.11399 -0.00180 0.18412 -0.23025 -0.04250 -0.03319 0.04431 0.02321
-0.42186 -0.18100 -0.12137 -0.16383 0.29310 0.36636 0.50295 0.36709
0.35794 0.33216 0.44884 0.16064 0.23786 -0.09145 -0.14921 -0.15022
0.16668 0.15168 -0.04768 -0.28407 0.25896 0.27752 -0.06586 0.41126
0.71308 0.11793 -0.02141 -0.11464 -0.45694 -0.22247 -0.26507 -0.04757
0.03515 0.00649 0.17714 0.29087 0.16689 -0.09929 0.31932 0.27506
0.75503 0.66564 -0.08033 -0.19898 -0.22547 -0.26529 0.11542 -0.17101
0.20143 -0.29631 -0.09890 -0.16704 -0.25614 -0.18532 -0.21561 0.21561
0.26076 0.34083 -0.27957 -0.04404 -0.07178 0.23877 0.02863 0.07677
0.07799 -0.23365 -0.02550 -0.39455 0.23001 0.28464 0.07057 -0.34781
0.13896 0.02658 0.14526 0.15247 -0.20945 0.21386 0.22297 -0.18672
-0.02896 -0.23749 -0.03875 0.05925 0.17403 -0.35230 0.05207 -0.01313
0.31248 0.03611 0.05010 -0.26544 -0.11518 -0.06073 -0.32724 0.07888
-0.29552 0.27531 0.17447 -0.20002 -0.20261 -0.15731 0.26767 -0.16931
-0.17261 -0.02859 0.09191 -0.23871 0.16677 0.04132 -0.06391 -0.05346
0.23160 -0.20704 -0.12325 -0.16042 -0.08657 0.39272 -0.01935 -0.18018
0.12637 -0.47306 0.09407 -0.35583 0.18093 -0.29664 -0.30830 -0.07307
-0.14427 0.23614 0.26070 0.15698 0.20351 0.02297 -0.39843 -0.31271
-0.34629 -0.39643 -0.21314 -0.32943 0.02273 -0.14380 -0.00270 0.25172
-0.20235 0.10583 0.20540 0.25738 0.32488 0.51819 0.21654 0.18578
-0.18598 -0.24706 0.30433 0.35885 0.19780 -0.52102 -0.17346 0.11532
0.16506 0.01922 -0.27191 -0.00593 -0.18041 -0.00160 0.46245 0.39549
-0.39804 0.13072 -0.05369 -0.28713 -0.20725 -0.39986 -0.50404 0.17947
0.18612 -0.17836 0.14125 0.05589 -0.48275 -0.48026 0.17239 -0.09474
-0.12722 0.17544 0.29028 0.03139 -0.19391 0.20415 0.32792 0.11751
-0.11536 0.14765 -0.02865 -0.48378 -0.72014 -0.32970 0.03172 -0.06877
-0.39889 0.20567 -0.34224 0.34297 -0.38787 0.17824 0.32719 -0.04952
-0.01419 0.08384 -0.13588 -0.05174 -0.37679 -0.53326 0.42502 -0.20274
-0.41001 0.08922 -0.09141 0.02796 0.05550 -0.02348 0.07514 0.06845
0.17586 0.16302 -0.19024 0.12529 -0.24973 -0.23197 0.24216 0.09600
0.17929 0.12357 0.25173 -0.29384 -0.17312 -0.51978 0.04586 0.39616
-0.23548 0.09383 0.03687 0.41564 0.25256 0.12422 0.03595 0.62811
0.19997 0.34624 -0.03387 -0.40104 -0.11257 -0.01349 -0.30485 0.03541
-0.50672 -0.28778 -0.03933 0.27696 0.06598 0.58838 0.32516 0.41716
0.07572 0.71652 0.66648 -0.04550 0.02741 -0.22359 -0.08167 0.12760
-0.08664 -0.34032 -0.10915 -0.26124 0.31737 0.38047 0.58676 0.77370
0.76675 0.27906 0.13973 -0.09522 -0.49040 -0.79783 0.14411 -0.32074
0.17143 0.09296 0.15610 -0.05288 0.00662 0.05633 0.42071 0.36669
0.57045 0.92292 0.32816 -0.22477 -0.55725 -0.21621 0.31721 0.03744
-0.36331 0.15514 -0.25227 0.01275 -0.19629 -0.23529 -0.16786 -0.13350
0.15537 0.42677 -0.24788 -0.28770 -0.14876 -0.05670 0.08698 0.09236
-0.12175 0.16251 -0.01187 -0.36563 -0.05480 0.00497 -0.05109 -0.17466
0.38349 0.11645 0.24150 0.10551 0.14836 -0.12672 0.34639 -0.18150
0.22904 0.21443 -0.34167 -0.07215 0.05723 -0.36108 -0.49538 -0.37878
0.10123 -0.06575 -0.35071 -0.45824 -0.19258 0.14981 0.11447 0.16033
0.12008 0.05429 0.16774 0.17809 0.21739 0.01498 0.29120 -0.06740
-0.35969 -0.48098 -0.17902 -0.12424 -0.17630 -0.18892 0.01680 -0.25038
0.19626 -0.12042 -0.04835 0.13945 0.14714 0.42172 -0.01153 0.10974
-0.28256 -0.59770 -0.31829 -0.09972 -0.18029 -0.13080 -0.33241 0.24831
0.16507 0.29242 0.35812 0.19307 -0.19808 -0.25285 -0.13260 0.03258
0.01687 -0.75889 -0.06047 0.00040 -0.38363 -0.30495 0.08502 0.05812
-0.16069 0.27340 0.25612 0.12374 0.03165 -0.14166 -0.15912 -0.16594
0.01734 -0.04693 -0.09620 -0.15976 -0.00652 0.07783 0.24866 0.13973
0.28526 -0.16483 -0.00629 -0.18837 0.22589 -0.03466 -0.08018 0.02473
-0.00950 -0.12467 -0.07826 -0.21244 -0.12606 -0.17806 0.26620 0.16403

Appendix C2: Fracture Discriminator Weights & Thresholds

0.21877	0.11133	0.13011	-0.15768	0.12056	-0.29676	0.11630	0.23556
-0.11762	-0.26742	0.03451	-0.25288	0.18294	0.09104	0.07558	0.08552
0.10554	0.20985	0.12709	0.09393	0.09982	0.18509	-0.18058	0.00918
0.16578	-0.26653	-0.00775	-0.26021	0.01174	-0.10366	-0.22018	-0.14201
0.40278	0.07456	0.11500	-0.20844	-0.04560	0.27874	0.30004	-0.05317
-0.14226	-0.04345	-0.25241	0.31972	-0.15020	-0.21687	-0.13867	-0.08625
0.24951	0.18277	-0.10238	0.01810	-0.10809	-0.05038	0.18918	-0.06936
0.15008	-0.13666	0.03576	0.19919	-0.04667	-0.24039	-0.08332	0.18390
0.25206	0.12508	0.37060	-0.07019	0.39309	0.50261	0.29658	0.30023
0.24995	0.24102	-0.07819	0.38714	-0.07100	-0.18998	-0.13721	-0.08888
-0.03632	0.19358	-0.07386	0.37388	0.09624	0.39794	0.09463	0.00663
0.33210	0.16619	0.12206	0.23029	0.38619	-0.13379	0.27271	-0.03734
0.27025	0.11314	-0.12645	-0.06548	0.02528	0.18347	0.10834	0.18171
0.09453	0.02782	0.32041	0.34207	0.14061	0.16680	-0.26957	-0.02085
0.05755	0.25429	-0.08888	0.11708	0.21790	0.34515	0.42141	0.46455
0.36383	0.14725	0.33555	-0.11203	0.10366	0.16614	-0.00020	0.33042
-0.26020	-0.04450	-0.24805	0.16177	-0.06683	-0.09344	-0.05018	-0.05828
0.32331	0.25881	-0.17641	0.06342	-0.19535	0.37102	-0.15351	0.11918
0.27485	0.08197	0.25886	-0.02255	0.01422	0.04113	0.13486	0.25058
0.33238	-0.13492	-0.28564	0.08846	-0.25016	0.14531	0.03607	0.36273
0.06588	-0.12506	0.01228	-0.04155	0.19295	0.24496	-0.25825	-0.23206
-0.02513	-0.16293	-0.15556	0.12730	0.18641	-0.07751	-0.12432	0.11039
0.00489	0.35641	0.03844	-0.01883	0.01290	0.00846	0.10112	0.25388
-0.08912	-0.11051	0.23488	-0.07347	-0.10210	0.05112	0.20850	-0.22695
-0.17486	-0.10832	-0.18817	-0.08737	0.01539	0.10173	-0.07843	-0.28746
0.04563	-0.27310	-0.14897	0.13546	-0.05678	0.25619	-0.18980	-0.22634
0.05105	-0.11413	-0.28406	0.02753	0.14653	-0.25952	0.26332	-0.11010
0.20585	-0.08125	-0.20399	-0.14771	-0.05865	-0.18112	-0.08003	0.09333
0.11644	-0.04939	-0.07676	0.15057	0.18335	-0.12039	-0.04730	0.24970
0.14709	0.04302	-0.05820	0.09506	-0.19984	-0.17898	-0.30921	-0.11210
0.25698	-0.20514	-0.11561	-0.24250	-0.11525	-0.22919	-0.03547	-0.12144
-0.26946	-0.11191	-0.24820	0.01545	0.27547	0.03917	-0.15427	0.21553
-0.09194	-0.00081	0.00995	0.06125	-0.28198	0.17270	-0.13563	-0.06952
0.12363	0.22279	0.14560	0.26811	-0.37485	-0.17682	-0.06961	0.27152
0.28726	0.03178	0.18184	-0.28215	0.07053	-0.15464	-0.19477	0.28824
-0.33846	-0.10241	-0.08445	0.12757	-0.04120	-0.05586	-0.17405	0.27462
-0.08072	-0.21295	0.24081	-0.05064	0.07812	-0.26414	-0.08604	-0.01013
0.02850	0.17002	0.14735	0.22260	-0.18415	-0.01267	0.18065	-0.16915
0.35955	0.13048	-0.03928	-0.09886	-0.07745	-0.07534	-0.03636	-0.04699
0.30043	0.19833	0.23361	-0.09712	0.04645	0.10186	0.22836	0.32605
0.16485	-0.06189	-0.02143	-0.03139	0.47313	-0.03627	0.46462	0.21995
0.49093	0.16775	0.01310	0.13748	0.25142	-0.04823	-0.11750	0.17464
-0.18146	-0.02401	0.05850	0.33499	0.35573	0.45937	0.32629	0.53621
0.37656	0.23465	0.22334	-0.19292	0.27266	0.13297	-0.09479	-0.03633
-0.20198	-0.08985	-0.22607	0.25877	-0.06080	0.12543	0.06297	0.72895
0.55480	0.57019	0.04495	0.09066	0.03680	-0.22542	0.08088	0.11145
-0.19850	-0.15421	0.34939	0.20364	-0.09094	0.30320	-0.07864	0.53787
0.19288	0.25815	-0.09795	0.08932	-0.01782	0.01487	-0.06472	-0.19308
-0.16114	-0.29875	-0.02744	-0.21033	-0.11894	0.24474	0.09372	-0.02829
0.42429	0.38000	-0.14985	-0.10021	0.14617	0.09757	-0.00935	0.02778
-0.01359	0.18629	-0.01467	-0.32722	0.23035	0.12406	-0.31114	-0.00877
0.41229	0.11276	0.00630	-0.01313	0.23499	0.05396	0.01535	0.28291
0.08223	-0.16124	0.00841	-0.28392	-0.02701	0.03539	0.15545	-0.07717
0.30710	0.03165	-0.10866	-0.19970	-0.05339	0.06319	-0.05911	0.12122
0.06218	0.09198	0.25833	-0.17169	-0.22698	-0.07651	-0.09330	-0.04138
-0.16591	-0.15018	0.08247	-0.18112	-0.30352	0.00925	0.30368	0.17455
-0.14133	0.31858	0.00292	-0.13432	0.20120	-0.15329	0.38613	-0.03419
-0.09124	-0.42437	-0.19259	0.10075	0.04040	-0.29955	-0.39712	0.10819
0.28760	0.15271	-0.09789	-0.08643	0.30467	-0.24963	-0.11854	-0.06525
-0.19036	-0.32424	-0.01970	-0.22581	-0.26406	-0.12328	0.22697	-0.02105
-0.42045	0.03836	-0.18571	-0.00956	0.02630	0.58337	0.21948	0.32721
-0.37886	-0.29295	-0.25128	-0.01557	0.16070	-0.16909	-0.33670	-0.11749
0.07184	-0.28453	-0.02047	-0.43855	0.03774	-0.27149	0.42042	0.10858
0.05701	-0.15887	0.03276	0.07506	-0.09903	-0.23091	-0.13213	-0.36580
0.32214	-0.17092	-0.23188	-0.09352	-0.28127	-0.44636	-0.03540	-0.19429
-0.45885	0.13660	0.03905	0.03738	-0.42598	-0.06453	-0.02851	-0.11775
-0.15089	0.00484	0.11946	-0.42910	-0.52267	-0.76238	-0.08193	0.37960
-0.45832	-0.15518	-0.39063	-0.00501	-0.31887	0.21527	-0.07082	-0.20342
0.28945	0.17002	-0.15233	-0.26960	-0.25660	-0.43505	0.33112	-0.32850
-0.20968	0.30204	-0.50450	0.10062	-0.30130	-0.12176	0.06060	0.11027
-0.01596	-0.25963	0.14892	0.20705	-0.03091	0.10590	0.25946	-0.07562
-0.36952	0.20340	0.23088	-0.17482	-0.11271	-0.11272	0.02194	-0.07774
-0.27696	-0.11285	-0.00553	0.22206	0.02461	0.21508	0.07765	0.65001

Appendix C2: Fracture Discriminator Weights & Thresholds

0.40509 0.49189 0.33064 0.03842 -0.09655 -0.27100 -0.59162 -0.12030
-0.15706 0.09105 -0.13975 -0.09522 0.33590 0.48436 0.42083 0.51551
0.83139 0.55968 0.85860 -0.21741 -0.10230 -0.16678 -0.42928 0.35223
-0.21859 -0.02930 -0.36436 -0.07434 0.15548 -0.01362 0.45642 0.73269
0.77715 0.85497 0.58581 0.17030 -0.33491 -0.61396 0.10635 -0.08332
-0.02738 -0.14160 0.15953 -0.15799 0.26969 -0.15792 0.17841 0.74547
1.03962 1.18354 0.34688 -0.36104 -0.46701 -0.14709 0.10638 -0.03615
-0.05939 -0.23333 -0.17701 -0.05402 -0.12413 0.11557 -0.18028 -0.21815
0.45154 -0.11644 -0.09630 -0.31987 -0.14415 0.05774 -0.02862 0.12153
-0.39070 -0.31323 -0.06335 -0.06471 0.20681 0.08159 -0.40563 -0.41827
0.45768 -0.03180 -0.13717 0.04167 -0.07953 0.03640 0.32058 0.04793
-0.30242 -0.18099 -0.12263 -0.09874 0.18342 -0.45461 -0.21375 -0.18950
0.31871 -0.25563 -0.16421 -0.49896 -0.34027 -0.28568 -0.15062 0.04589
-0.30154 0.14659 -0.20371 0.21603 0.14035 -0.08198 0.02732 0.31213
0.03480 -0.26744 -0.06112 -0.66174 -0.15459 -0.06521 0.17987 0.20585
-0.23190 0.34060 0.41143 0.17140 0.47999 0.30897 -0.01937 0.01925
-0.12643 -0.62460 -0.40606 -0.38283 0.07565 -0.20723 -0.16001 0.15592
0.35746 0.14580 0.19090 -0.44320 0.16115 0.04607 0.01060 -0.05897
-0.47988 -0.93444 -0.55249 -0.43363 -0.20235 -0.27233 -0.26134 -0.16516
0.33497 -0.03647 -0.11942 0.27536 0.12369 -0.21464 -0.30550 -0.08819
0.13409 0.31927 0.13059 0.02055 -0.12773 0.14850 0.40561 0.07903
-0.10161 0.03791 0.10006 0.19281 0.03669 0.16212 -0.25720 -0.29874
-0.17684 -0.17526 -0.01638 -0.21766 0.07437 0.36431 -0.02404 0.20579
-0.04354 0.25811 -0.23471 -0.05561 0.15721 0.23159 0.12707 -0.26944
0.01457 -0.01702 0.20114 -0.35309 0.04323 -0.21619 -0.06674 -0.12540
0.10353 0.18119 0.09353 0.21547 -0.04032 -0.00452 0.24553 -0.13786
0.20099 0.11497 0.26428 0.13050 0.29242 0.00126 -0.08652 0.13636
-0.01631 -0.13868 0.17224 -0.12875 0.02159 -0.18506 0.04088 0.12579
0.05304 -0.31317 -0.20168 -0.19598 0.13232 0.17472 -0.07249 -0.19617
-0.07916 0.06062 -0.07577 -0.24614 0.02042 0.15446 -0.21140 -0.30650
0.20102 -0.09187 -0.05368 -0.13615 0.23751 0.27165 0.19716 -0.11427
-0.19410 0.02724 0.09244 -0.12309 -0.23319 -0.36345 -0.19716 -0.50256
-0.02643 -0.37447 0.14409 0.14031 -0.09307 -0.14159 0.02686 0.02529
0.33380 0.27648 -0.15110 0.01771 -0.15726 -0.04153 -0.10408 -0.26807
-0.61611 -0.51265 -0.49350 0.09013 -0.18230 -0.10910 -0.07425 0.14330
0.24157 -0.16004 -0.14529 -0.17526 0.18579 -0.34444 -0.02017 -0.33502
-0.75240 -0.36257 -0.39811 -0.19145 -0.02756 0.38209 0.23209 -0.02470
-0.23498 0.21665 0.07229 -0.04482 -0.07937 -0.18629 -0.43707 -0.66615
-0.33415 -0.81185 -0.25340 -0.15468 -0.11261 0.05386 -0.19521 0.10197
0.13568 0.10848 0.22953 0.07397 0.07776 -0.01951 -0.35652 -0.22269
-0.44116 0.00845 0.23607 0.02643 -0.12958 -0.11266 -0.31859 -0.08256
-0.12613 0.04002 -0.04835 0.07360 0.05784 0.02444 0.21513 0.09684
-0.37263 0.27631 0.11022 -0.27567 -0.14087 -0.03236 -0.30987 0.25307
-0.01383 0.09615 0.07237 0.21821 -0.04781 0.22665 -0.09362 0.30060
0.13176 0.03281 -0.10891 -0.11941 0.28878 0.29909 0.04596 -0.28046
-0.28891 -0.23448 -0.14003 -0.05678 0.08524 0.03905 -0.14300 0.08188
0.09177 0.35399 -0.11237 -0.00006 -0.11286 -0.02049 -0.19957 -0.17556
0.14823 0.07930 0.08616 -0.14229 -0.18967 0.12053 -0.39493 0.24236
-0.07174 0.18266 -0.10870 0.26624 -0.11322 -0.10187 0.33722 -0.12363
0.14229 0.17518 0.00145 -0.15809 -0.05355 0.13093 0.27556 0.38218
0.39605 0.27179 -0.12186 -0.16634 0.20939 0.38440 0.07231 0.03528
0.08781 -0.15969 0.15175 -0.10888 0.14576 -0.38863 0.10595 -0.38764
-0.05818 -0.11146 -0.12268 -0.40136 -0.16329 -0.11606 0.23230 0.01272
-0.30493 0.25228 -0.08866 0.03387 -0.03033 0.11730 -0.06596 -0.03266
0.07054 -0.21845 -0.08243 0.12547 0.22259 0.13580 0.27057 0.19348
0.18545 0.14382 -0.20659 0.10652 0.08427 0.23419 0.15155 0.09212
0.13182 0.02935 -0.11200 0.15387 0.32301 0.02849 -0.19405 0.03999
0.08935 0.09001 0.23965 0.14734 -0.01876 0.08990 -0.23531 0.14740
-0.13071 -0.22591 0.17422 0.06141 0.06550 0.06983 0.24848 0.29202
0.02760 0.00621 0.06564 -0.23315 0.29221 0.29038 -0.04195 -0.17225
-0.23612 0.17078 0.24276 -0.26182 -0.16411 0.22834 -0.04994 0.11405
-0.24871 -0.10096 -0.26606 -0.26257 -0.07657 -0.30074 -0.16619 -0.01069
0.15529 -0.15470 -0.08073 -0.24434 0.27611 0.00415 -0.08723 0.22031
0.12329 0.05382 0.18055 0.01001 -0.16585 -0.08557 -0.18204 -0.03071
0.03269 0.02364 0.12983 0.04621 0.05289 -0.15520 0.03304 -0.05438
0.34415 0.26213 -0.25832 -0.24924 -0.24017 -0.11389 -0.17828 0.17401
0.09169 -0.08931 -0.13620 -0.04225 0.30460 0.06136 -0.18579 0.16721
-0.02947 0.15409 0.31496 -0.12711 -0.27581 -0.05618 -0.27902 -0.25180
-0.30186 0.10826 -0.33627 0.27292 0.32370 0.33323 0.18301 0.19298
0.05010 -0.22893 -0.30269 -0.00293 0.07026 -0.21856 -0.16627 -0.00202
-0.31211 -0.34125 -0.00154 0.28068 0.39399 0.31387 0.17671 0.02702
0.27228 0.16503 -0.11392 0.30617 -0.03491 0.28620 0.04511 -0.01704
-0.26511 0.07920 -0.25462 -0.08129 0.02551 -0.18827 -0.32005 -0.25395

Appendix C2: Fracture Discriminator Weights & Thresholds

0.05149 -0.07914 0.23681 0.10105 -0.08679 -0.33031 -0.14513 -0.09303
-0.06885 -0.29780 0.14212 -0.25985 0.27056 0.18148 -0.20811 -0.06260
-0.01496 0.13914 0.25468 0.31629 -0.01318 0.12379 -0.10651 0.06864
0.02019 0.20900 0.30347 0.07494 -0.22564 -0.07888 0.24078 0.16245
0.12772 0.18552 0.16607 -0.22895 -0.25250 -0.35683 0.13769 -0.14513
-0.16418 0.13107 -0.17837 -0.04106 0.17333 0.07055 0.22147 -0.20852
0.05353 -0.23813 -0.13517 -0.33027 -0.17405 0.03893 0.05365 0.20429
0.10643 0.01802 0.01464 0.15656 -0.26996 -0.10398 -0.12633 -0.24351
0.03095 -0.32776 0.22157 -0.10218 0.24756 -0.00002 -0.24621 0.05706
-0.00057 -0.08308 0.00659 -0.16921 -0.03650 -0.09637 -0.29083 -0.18453
0.05270 0.21362 -0.11542 0.03716 0.23227 0.12351 -0.27483 0.16759
0.06563 -0.27026 -0.24713 0.00631 0.07258 -0.03817 -0.27475 0.14170
0.28893 -0.04890 -0.26233 -0.11535 0.18881 -0.24749 0.23838 0.04038
0.10199 -0.20171 0.07581 -0.27655 -0.11561 0.05742 0.10457 0.11181
0.23627 0.05186 0.10137 0.03888 0.26834 -0.11792 -0.28506 0.09539
-0.18455 0.17403 0.23113 -0.12970 0.24476 0.01861 0.07977 0.19516
-0.22181 0.19049 0.01861 -0.11379 -0.05309 0.08396 -0.24026 0.26749
-0.28360 -0.16018 0.17620 0.04875 -0.07772 -0.25866 0.29120 0.14722
-0.05824 0.22329 -0.21596 -0.12154 -0.11137 0.04617 -0.25021 -0.20329
0.04193 -0.03000 0.00276 0.24581 -0.24477 -0.01769 -0.16664 0.17899
-0.02452 -0.00096 -0.09617 0.02531 0.08455 -0.01151 0.02113 0.10328
0.17433 0.03648 -0.02942 -0.13508 0.34466 -0.17502 -0.03993 0.02310
-0.13609 0.25306 0.33518 0.24607 0.10488 0.26810 0.02389 0.14130
-0.05490 0.09532 -0.01997 0.04329 -0.20698 0.04360 -0.07834 -0.04747
-0.26002 0.05385 0.26217 0.25037 0.19228 0.19876 0.28405 0.47341
0.15853 0.41020 0.25564 0.20578 0.24559 0.30062 0.08746 -0.01310
-0.05954 0.19689 -0.14410 -0.09139 -0.17417 0.04545 0.05702 0.16279
-0.06626 0.14353 0.14900 -0.13246 0.24858 -0.17475 0.14291 0.21106
-0.04447 -0.16357 0.22862 0.18991 -0.04578 -0.15924 -0.14461 -0.01208
0.17724 0.10667 0.17952 0.09636 0.34104 0.02464 -0.10640 0.08594
0.23490 0.03014 0.08694 -0.13298 0.13998 0.06634 0.27574 -0.03309
0.12129 0.09891 0.18677 0.35494 -0.14299 -0.04618 -0.19928 -0.23385
-0.06239 -0.18980 -0.15958 0.11259 -0.23322 0.31437 0.20481 0.26500
-0.04452 0.01811 0.06417 -0.02634 0.21281 -0.14713 -0.08264 0.09730
-0.08850 0.15922 0.01246 0.20462 0.05331 -0.22810 0.03890 0.26169
-0.01363 0.08436 -0.18334 -0.15683 0.16533 0.32246 -0.00039 -0.10572
-0.16459 0.25014 0.03815 0.13421 -0.26139 -0.07387 -0.18944 0.29654
0.30315 -0.00542 0.19461 -0.27709 0.01559 0.22153 0.14353 -0.01886
0.09189 0.18011 0.14229 -0.12084 0.22950 0.07712 -0.04332 0.07268
0.20806 0.00214 -0.23332 -0.22231 -0.10213 0.06873 0.25340 -0.07597
-0.00432 0.10389 -0.29764 -0.18908 0.01485 0.22697 0.17676 -0.18136
0.15995 0.04229 -0.16384 -0.17938 -0.16378 -0.00844 0.14444 0.17659
-0.46542 0.13815 0.17568 -0.27744 -0.28063 0.37130 0.37706 0.04930
-0.14964 0.09132 -0.25432 0.42002 -0.01910 -0.02310 -0.07212 0.07262
-0.20059 -0.05166 0.11218 -0.07041 0.04336 -0.14617 0.05015 0.19010
-0.11988 0.05599 -0.17605 0.05127 0.20444 -0.31074 -0.08350 0.17008
0.09435 0.16226 0.00145 -0.04920 -0.00065 -0.05017 -0.05818 0.02514
-0.44074 0.04351 -0.01163 0.29814 -0.28256 -0.04458 0.11231 -0.06470
0.31269 0.12741 -0.03461 -0.36992 -0.29116 -0.50406 0.17715 -0.11905
-0.30609 -0.01027 -0.32902 -0.22209 -0.19220 -0.22088 -0.04588 -0.06895
0.45274 0.23106 -0.12835 -0.11639 -0.07664 -0.15099 0.11222 -0.29952
-0.40969 -0.04362 -0.35169 -0.17156 -0.17178 -0.33260 0.03737 0.22358
0.02696 0.08361 -0.15033 0.35162 -0.10098 0.12012 0.45696 0.37029
0.21380 0.14473 0.14012 -0.10684 -0.17468 -0.01539 -0.34234 0.12790
-0.42754 -0.27386 0.04084 0.36909 0.08853 0.45540 0.50341 0.38666
0.31641 0.19953 0.36176 -0.15887 -0.26240 0.15440 -0.38240 0.28825
-0.37086 -0.07132 0.19624 0.12918 0.03174 0.10649 0.21939 0.38923
0.16928 0.76689 0.80250 -0.10191 0.23461 -0.31714 0.13214 -0.08495
0.07380 0.26905 -0.27914 -0.35455 -0.13916 0.28065 0.45153 0.79272
0.56972 0.71666 0.49549 0.10914 -0.35637 -0.32788 0.07853 -0.37692
-0.04085 0.09528 -0.11639 0.22086 0.11201 0.25913 0.26590 0.32549
0.59002 0.91470 0.26251 0.13852 -0.37563 0.02680 0.11971 -0.06451
-0.09343 -0.02229 0.20869 0.18911 -0.04442 0.15569 0.27919 0.01712
0.07280 0.34817 -0.25217 -0.24478 -0.14182 0.01202 -0.01204 0.21022
0.08304 0.01352 -0.11538 -0.24997 0.17978 -0.16546 -0.03701 0.15192
0.28743 -0.31174 -0.13651 0.11222 0.22786 -0.20082 0.00070 -0.26694
-0.25330 -0.13146 -0.10818 0.10432 0.09660 0.03419 -0.29371 -0.02158
0.01686 -0.07972 -0.43134 0.01893 0.24815 -0.05484 -0.11525 -0.25369
0.00837 0.24718 0.04036 0.06641 0.04761 0.06661 0.18305 -0.06259
-0.39341 -0.36287 -0.23007 -0.51421 -0.04190 0.03806 0.19782 0.03871
0.22143 -0.22007 0.29585 0.15398 0.48882 0.00627 -0.08945 0.06909
0.15464 -0.21506 -0.24838 -0.02947 -0.23626 -0.01305 -0.40414 0.13355
-0.02630 -0.01003 0.22432 -0.16648 -0.13271 -0.08973 -0.37031 -0.14546

Appendix C2: Fracture Discriminator Weights & Thresholds

-0.40301 -0.34865 -0.02465 0.04047 -0.27026 -0.30017 0.17810 0.32571
-0.26187 0.09356 0.19681 0.11151 -0.16342 -0.25535 -0.33335 -0.06029
0.22966 -0.25426 -0.02652 0.18028 0.11505 -0.23636 -0.10063 -0.04290
0.20907 -0.16871 0.09329 -0.22606 -0.09315 0.01218 -0.21829 0.07584
-0.21469 -0.01053 0.02544 -0.25325 0.19945 0.27522 0.00527 -0.09791
0.15016 -0.09117 -0.25174 0.10612 0.11363 0.20892 -0.10459 0.25641
-0.09101 0.14516 -0.15485 -0.19850 -0.12292 -0.04120 -0.08835 -0.16112
-0.01123 -0.17813 0.19741 -0.00629 -0.22305 0.26376 -0.07997 -0.13781
-0.14410 0.15075 0.26169 0.11107 0.14520 0.27566 -0.07750 0.22602
0.00890 -0.23684 0.18417 -0.18302 -0.04820 0.15323 0.08546 0.03887
0.30525 -0.34803 0.27022 0.02432 -0.02855 -0.05624 -0.09947 0.03007
0.02643 -0.08822 -0.35009 -0.30648 0.01161 -0.21183 -0.29107 -0.33070
-0.09212 -0.23550 -0.31882 0.20520 -0.14295 0.16232 0.02418 -0.32707
-0.15697 0.17525 0.13610 -0.32840 0.02913 0.04617 -0.36456 -0.36970
0.14110 0.16981 -0.14791 -0.04292 0.11633 0.13258 -0.21069 0.08238
0.02272 -0.04507 -0.18075 0.13479 -0.31463 -0.09776 0.02514 0.06189
0.07517 -0.26865 -0.09515 0.02416 -0.12457 0.05625 0.27109 0.09552
0.24246 0.28046 0.07979 -0.30086 0.00334 0.11420 -0.21012 -0.03577
-0.17196 -0.19562 -0.39625 -0.03346 0.03816 0.03147 0.22840 -0.09114
-0.13928 0.15062 -0.09677 -0.13766 -0.02028 0.24076 0.15988 -0.30143
-0.31297 -0.29298 0.13775 -0.27085 0.22448 -0.17925 0.18524 0.00407
-0.14384 0.27841 -0.05139 0.12719 -0.31980 -0.08588 0.07137 -0.15607
0.04298 -0.25299 -0.24974 -0.07189 -0.22068 0.20227 0.24731 0.13649
-0.19597 0.04291 0.24806 -0.05589 0.18588 -0.28151 -0.13279 -0.14736
0.05698 -0.09826 0.28078 -0.01459 0.06406 0.00664 0.07068 0.00478
-0.12787 0.19944 0.28100 -0.26114 0.16403 -0.19314 -0.00897 -0.20281
0.06375 0.13057 0.10892 0.21218 0.22664 0.07495 0.19892 0.24571
0.23954 -0.33588 -0.31397 0.20086 0.19132 0.07019 0.10871 0.00842
0.04464 -0.11202 0.10391 0.32688 -0.13699 0.19363 0.24328 -0.07993
-0.02733 0.08703 0.14086 0.17004 0.19329 -0.31520 -0.25174 -0.20028
-0.22448 -0.18855 0.30199 -0.02903 0.15056 0.03766 -0.19354 -0.23066
0.10336 0.19537 -0.00403 0.03992 -0.14073 0.17727 -0.04667 -0.12989
0.20668 0.17671 -0.17834 0.18564 -0.07699 0.07866 -0.06618 -0.21176
0.38245 -0.16790 0.40487 0.20835 0.04049 -0.57123 -0.62866 -0.19680
-0.10979 0.12388 -0.14134 -0.02171 0.13517 0.45080 0.36833 0.00199
-0.26530 0.31811 0.23479 0.01718 0.25556 0.26814 -0.31201 0.12335
0.25606 -0.23572 0.04859 -0.23325 -0.05791 0.44300 0.54910 0.11628
-0.38790 0.15220 -0.05852 -0.05927 0.14899 -0.01247 0.08123 -0.02132
0.00553 0.16158 -0.07537 -0.39145 -0.01596 0.23817 -0.26491 0.24463
0.27527 0.33132 -0.30528 0.23768 0.67828 0.26452 -0.33921 0.08692
0.50872 0.01919 0.09509 -0.16089 0.30116 0.29117 -0.18157 0.15990
-0.16656 0.01299 0.10426 0.03748 0.30977 0.30262 -0.40645 -0.07240
0.10224 -0.28437 -0.00546 0.24637 0.12686 -0.08435 -0.11584 0.09482
0.17550 0.07050 0.27582 0.07240 0.18000 -0.25446 0.21561 0.01366
0.03213 -0.40602 -0.34796 0.20060 0.38716 0.51808 0.40683 -0.15394
0.51812 0.25203 -0.46920 -0.14266 -0.45077 0.02753 -0.44120 -0.64579
-0.27589 -0.03333 -0.28780 0.43042 0.01997 -0.09467 0.27900 -0.03209
0.12999 0.42827 -0.13144 -0.09457 -0.23329 -0.25254 -0.23041 -0.13954
-0.19565 -0.46272 -0.45444 0.19171 0.18558 0.07635 0.07951 0.09887
0.27942 0.01564 -0.12063 0.42114 -0.12501 -0.09358 -0.34506 -0.93022
-0.79404 -0.77144 -0.27018 -0.13839 0.65586 0.83246 0.18417 0.01918
0.07813 0.09642 -0.18382 0.15978 -0.18802 -0.15948 -0.32772 -0.27948
-1.01707 -0.93121 -0.13793 0.09109 0.25347 0.43943 -0.09027 0.21740
0.32846 0.33494 -0.14198 0.00536 0.04683 -0.11042 0.11136 0.17068
-0.05486 -0.23393 -0.10232 0.11811 0.34413 0.09325 -0.24132 -0.37926
0.00381 0.07569 0.17236 0.21000 -0.07761 0.04355 0.31595 0.13361
-0.23697 0.40771 0.20936 0.06061 0.32340 0.23834 -0.13247 0.21471
0.13907 -0.14174 -0.00684 0.43858 -0.04619 -0.00807 0.17830 0.01350
-0.24001 0.69351 -0.05243 0.54159 0.12696 0.30264 -0.16720 0.04277
-0.02650 -0.14347 -0.16594 0.25144 -0.27624 0.08765 -0.42852 -0.05229
0.19324 0.21175 -0.12594 0.29235 0.12973 0.23448 0.03798 0.01824
0.09218 -0.11851 0.03692 -0.20391 -0.14837 -0.07651 -0.30643 -0.11766
0.03574 0.55480 0.12254 0.27362 -0.15063 0.12077 0.54477 -0.42035
-0.11455 0.05345 -0.27944 -0.03408 0.07018 0.06280 0.16408 0.39981
0.39216 0.59806 0.27668 0.24843 0.49172 0.39320 0.11078 -0.23491
-0.08275 0.20773 0.10777 -0.18780 0.02522 -0.19821 -0.35476 -0.17377
-0.08347 0.22923 0.14371 -0.02812 0.11318 0.11624 0.35014 -0.25570
0.25295 0.24108 -0.08322 0.31597 -0.23934 0.24609 -0.41299 -0.25028
-0.08484 -0.31083 -0.13275 -0.12025 -0.05548 0.47027 0.42124 0.33245
-0.08966 -0.07648 -0.04292 0.08433 0.19293 -0.09690 -0.03263 0.21490
0.21316 -0.11703 -0.19132 -0.36389 0.22063 0.32224 -0.33169 0.18412
-0.10976 -0.07392 0.23709 0.53379 0.51042 0.43623 0.15286 -0.06751
-0.00873 -0.09044 0.05761 -0.11154 -0.10967 -0.02298 -0.34297 -0.10410

Appendix C2: Fracture Discriminator Weights & Thresholds

0.05806 0.02059 -0.15750 -0.04467 0.39842 0.19353 0.08013 -0.29724
0.30964 -0.41329 0.14278 -0.02568 0.38544 0.22812 0.30532 0.12511
-0.19449 0.20345 -0.24436 -0.02602 -0.31554 0.10852 0.00887 -0.00565
0.28101 0.04896 -0.01022 -0.07218 0.20082 0.21815 -0.02109 -0.30505
-0.07422 0.11549 0.08260 0.08792 -0.21716 -0.21173 0.11019 -0.53964
-0.44487 -0.06609 -0.31348 -0.21810 0.09804 -0.08980 0.39175 0.08752
0.10270 0.10285 0.00768 0.19379 0.18577 0.05242 -0.43412 -0.42677
-0.45731 -0.41212 -0.05384 0.14960 -0.17773 0.02144 0.14393 0.15750
-0.01835 0.20156 0.14122 0.02415 -0.04582 -0.07527 0.07796 -0.30982
-0.52158 -0.34826 0.03492 0.17445 0.28154 0.53701 0.13637 0.31438
0.05298 -0.04445 -0.22106 -0.24736 -0.28216 -0.25332 -0.04238 -0.59608
-0.16751 -0.40853 -0.26331 -0.14843 -0.03858 0.37096 -0.17014 0.10771
0.08257 0.24689 -0.05892 -0.09304 0.24028 0.03725 0.13603 -0.13803
-0.25193 0.04731 -0.04849 -0.27705 0.01883 -0.12772 -0.34068 0.17879
0.01004 0.25777 0.10329 0.34613 -0.17172 -0.14269 0.04115 0.29723
-0.41526 0.03997 -0.33355 -0.12076 0.17219 0.09756 -0.06042 -0.22384
-0.09526 -0.18025 -0.07763 0.26409 0.10491 -0.05280 -0.05652 0.17198
-0.14758 0.12053 0.04819 0.26038 -0.13363 0.26070 0.30980 -0.09519
-0.16996 0.09751 -0.24041 -0.22927 -0.17099 0.11319 -0.26959 -0.09919
-0.10973 -0.10934 0.10197 0.40767 -0.08144 -0.12557 -0.30213 -0.15704
-0.02307 0.09654 -0.30701 0.19524 -0.16282 -0.34329 -0.37265 -0.10515
0.30667 -0.00564 0.21054 0.06387 0.00621 0.30725 0.18146 0.12243
0.03543 -0.02138 -0.32825 -0.15472 0.03758 -0.09490 0.11692 -0.21735
0.41566 0.39022 0.29869 0.22344 0.34004 0.45213 -0.01147 -0.00454
0.16047 -0.09542 0.21353 -0.06922 -0.36548 -0.06469 -0.30454 -0.08611
0.02696 0.07349 -0.35426 -0.03196 0.01452 0.38642 -0.03225 -0.16859
0.04280 -0.19826 -0.06698 0.08737 -0.04622 -0.02185 -0.24139 -0.31111
-0.23583 -0.31563 0.26333 -0.19416 -0.20671 0.42234 0.46200 0.03028
-0.38728 -0.21206 -0.00390 0.17546 0.49336 0.22346 -0.28067 0.09509
-0.16927 -0.30748 -0.00281 -0.03729 -0.00100 0.27916 -0.30243 0.17673
0.18184 -0.13803 0.17178 0.44643 0.38682 0.22806 -0.37955 -0.44844
0.26207 -0.32122 0.05275 -0.02644 0.08171 -0.02494 0.10162 0.02695
-0.08825 0.21585 -0.02418 0.30456 0.47418 0.22210 0.10833 -0.17109
0.48706 -0.14055 0.19386 -0.22913 0.25642 -0.17942 -0.12029 -0.07016
0.29343 0.13213 0.10770 -0.30187 -0.11656 0.18473 -0.17220 -0.12589
-0.09684 0.15447 0.14682 -0.01684 0.13559 0.34263 -0.21767 -0.34434
-0.02927 0.32060 -0.15379 -0.10374 0.08971 -0.10558 0.12130 -0.14576
-0.36052 -0.37949 -0.05202 0.16012 0.41119 -0.16527 0.49164 -0.22930
0.03695 0.39168 0.11284 0.24940 -0.18983 -0.37610 -0.34089 -0.22289
0.09824 -0.40900 -0.05250 0.18900 0.21916 0.39185 0.03011 0.04613
-0.21750 0.25076 -0.02307 -0.14271 -0.12171 -0.14456 -0.30629 -0.17824
-0.45954 -0.56907 -0.03463 -0.12819 0.39998 0.43344 0.02587 0.09517
-0.01781 -0.05433 -0.24571 0.11750 -0.34700 -0.08607 -0.21015 -0.56913
-0.66200 -0.35984 0.01886 -0.15056 0.53035 0.08341 0.04158 0.23380
0.06668 0.19985 -0.21745 -0.08384 -0.01210 -0.11434 -0.18098 0.08430
0.10827 -0.31080 0.22785 0.07538 -0.12183 -0.04356 0.04460 -0.26495
0.18080 0.07579 -0.16143 0.20615 0.17422 0.11655 0.52241 0.04021
-0.07447 0.09083 -0.03388 0.03198 0.01979 0.13251 -0.21526 -0.02843
0.03558 -0.18359 -0.04949 0.17419 0.02241 -0.13573 0.35114 0.24543
-0.29521 0.18281 0.31525 0.37167 -0.31908 -0.07796 0.24721 -0.23963
-0.05950 -0.11016 0.23768 -0.05305 0.01959 0.09005 -0.11124 -0.27984
0.17605 -0.08650 -0.02523 0.01527 0.31646 0.22358 -0.28176 -0.03244
-0.24629 -0.13321 0.00025 -0.31473 -0.29350 -0.15035 -0.31259 -0.07429
0.28714 0.49580 -0.21704 0.47829 0.24395 0.24118 0.02462 -0.38838
-0.01809 -0.39074 -0.32048 0.26137 -0.27917 0.09638 0.23342 -0.00272
0.34850 0.52915 0.02385 0.39257 0.31955 0.15208 -0.17905 -0.42764

layer 2 to 3 weights:

-2.53905 2.46397 1.67703 -1.63851 -3.40294 0.43549 -0.74591 -2.97635
1.34088 0.87050 -0.09671 -2.04232 0.29963 3.09067 1.92497 2.58592

C3 Classification of edge detection image processing filters

C4 Visual Basic code to reproduce the Fracture Alphabet

```
Function GenerateFractureAlphabet(Censor As Boolean, StringFormat As Boolean) As Variant
'
'  Populates array VectPatch() with either 8 digit binary character strings or 7x7 image patches
'  representing the Fracture Alphabet
'
'     Censor = True only admits fracture characters with 4 active strokes or less
'     Censor = False returns the whole fracture alphabet
'
'     StringFormat=True returns a 1D binary array
'     StringFormat=False returns a 2D image patch

    Dim Frac As String
    Dim FracNum() As Integer
    Dim VectPatch() As Variant
    Dim ExemplarCount As Integer
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    ReDim VectPatch(1 To 256) As Variant

    For i = 1 To 256
      ' Generate the binary string
      Frac = Integer2String(CLng(i - 1), 8)
      k = 0
      ' Censor if appropriate
      If Censor Then
        For j = 1 To 8
          If Mid(Frac, j, 1) = 1 Then k = k + 1
        Next
        If k < 5 Then
          ExemplarCount = ExemplarCount + 1
          VectPatch(ExemplarCount) = Gen_Frac(Frac, StringFormat)
        End If
      Else
        ExemplarCount = ExemplarCount + 1
        VectPatch(ExemplarCount) = Gen_Frac(Frac, StringFormat)
      End If
    Next

' Trim the VectPatch array and return to calling procedure
    ReDim Preserve VectPatch(1 To ExemplarCount) As Variant
    GenerateFractureAlphabet = VectPatch
End Function


Function Gen_Frac(Binary As String, Flag As Boolean) As Variant
' If Flag is TRUE, Gen_Frac returns a long binary string (for ANN training)
' If Flag is FALSE, Gen_Frac returns a binary patch (for image reconstruction)
' Binary is the 8 bit fracture character code.
    Dim Character(1 To 7, 1 To 7) As Integer
    Dim StringVer As String
    Dim i As Integer
    Dim j As Integer
```

```
  For j = 1 To 7 Step 1
    For i = 1 To 7 Step 1
      Character(i, j) = 0
    Next
  Next
  StringVer = ""
  Character(4, 4) = 1

  For i = 1 To 8 Step 1
    If Val(Mid(Binary, i, 1)) = 1 Then
      Select Case i
'Segment 8
        Case 1
          Character(1, 7) = 1
          Character(2, 6) = 1
          Character(3, 5) = 1
'Segment 7
        Case 2
          Character(4, 5) = 1
          Character(4, 6) = 1
          Character(4, 7) = 1
'Segment 6
        Case 3
          Character(5, 5) = 1
          Character(6, 6) = 1
          Character(7, 7) = 1
'Segment 5
        Case 4
          Character(5, 4) = 1
          Character(6, 4) = 1
          Character(7, 4) = 1
'Segment 4
        Case 5
          Character(5, 3) = 1
          Character(6, 2) = 1
          Character(7, 1) = 1
'Segment 3
        Case 6
          Character(4, 1) = 1
          Character(4, 2) = 1
          Character(4, 3) = 1
'Segment 2
        Case 7
          Character(1, 1) = 1
          Character(2, 2) = 1
          Character(3, 3) = 1
'Segment 1
        Case 8
          Character(1, 4) = 1
          Character(2, 4) = 1
          Character(3, 4) = 1
      End Select
    End If
  Next

  If Flag Then
    For i = 1 To 7 Step 1
      For j = 1 To 7 Step 1
        StringVer = StringVer & Str(Character(i, j))
```

```
        Next
      Next
      Gen_Frac = StringVer
    Else
      Gen_Frac = Character
    End If

End Function


Function Integer2String(i As Long, Bits As Integer) As String
' Returns a binary string "Binary" of length up to 31 bits from a denary integer less than 2147483647
' This means the maximum precision of any parameter is roughly 5e-10
' I is the Denary integer
' n is the number of bits with which the denary number is to be encoded.
' The maximum number of bits that can be used is 31 (a consequence of using the Long data type)
    Dim Index As Integer
    Dim Max As Long
    Dim j As Long
    Dim Binary As String

    Max = (2 ^ Bits) - 1
    If i <= Max Then
      For Index = Bits - 1 To 0 Step -1
        j = i Mod 2 ^ Index
        If j <> i Then
          Binary = Binary & "1"
          i = j
        Else
          Binary = Binary & "0"
        End If
      Next
    Else
      Binary = "Not enough bits to characterise integer"
    End If
    Integer2String = Binary
End Function
```

C5 Weighted Image Reduction algorithm

C5.1 Rescaling sample image patches

For the Fracture Classifier, the dimensions of the input template are fixed at 7 x 7 pixels. When image patch dimensions are greater than 7 x 7, a method must be available to reduce the size of the input patch to 7 x 7. From the standard methods available within most image processing packages, two candidate methods are available: image *resampling* and image *resizing*. Using a pixel neighbourhood as a basis, image resampling solves for a set of coefficients for a function that is used to fit the variation of pixel values throughout an image patch. Pixel values at new coordinate positions are calculated from these functions.

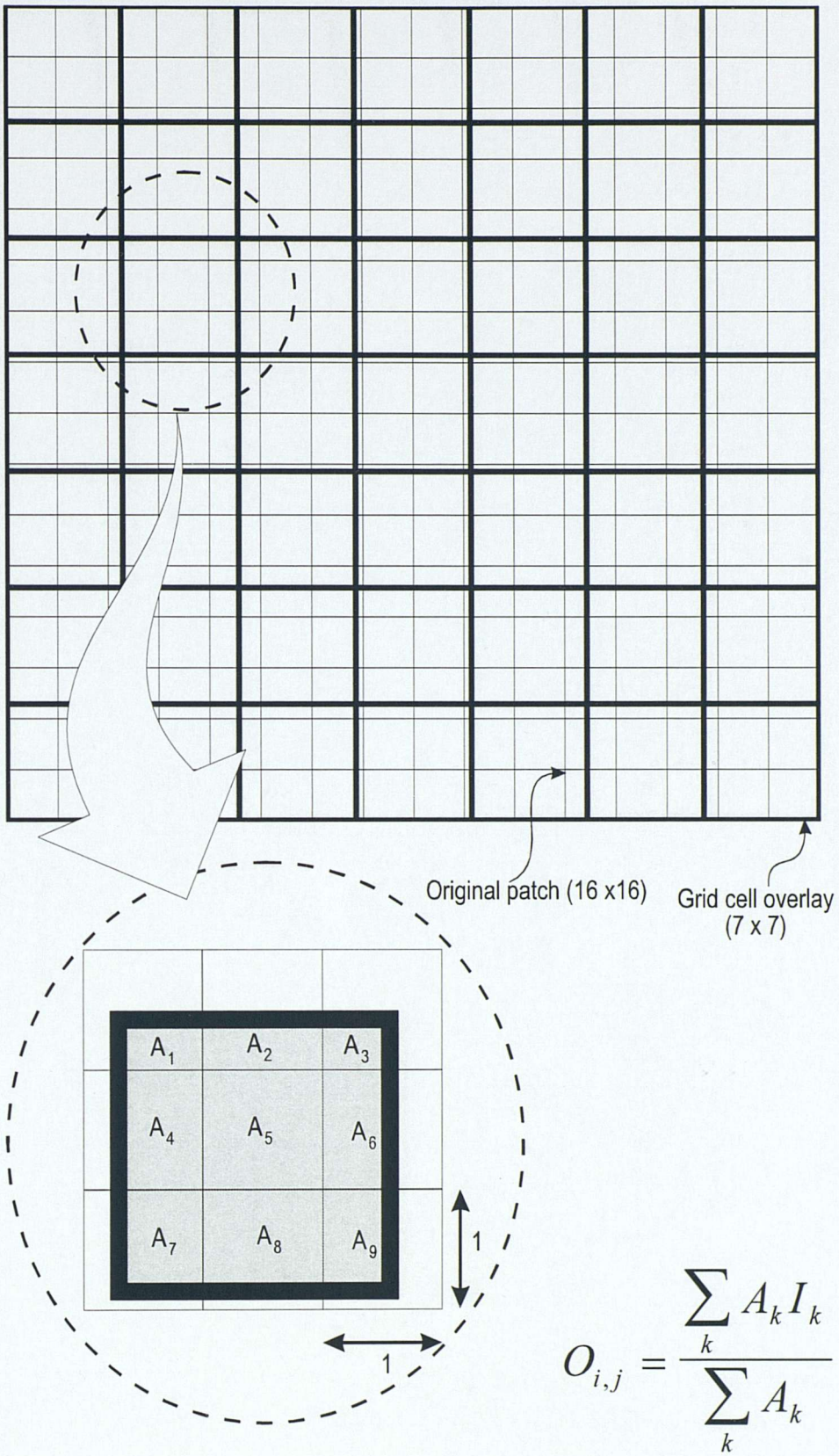
Image resizing operates by a straightforward removal or replication of existing pixels within an image, depending on whether the input image is to be reduced or enlarged respectively.

Both methods have their drawbacks. Image resampling requires solution of a set of simultaneous equations and thus is relatively computationally expensive. This method is advantageous should, in addition to an adjustment of scale, an input image patch requires rectification from a geometrically distorted state. Image resampling is an ideal method to adopt as it accommodates a reduction in the number of image rows and columns as a degenerate case of an affine transformation.

In reducing the size of an input image patch, image resizing deletes pixels. There is no guarantee that with this method, pixels that represent the locations of fractures within a digital image would be preserved. With enlargement using image resizing by pixel replication, clearly no such loss of information content occurs. Most of the images presented hitherto have been subjected to enlargement using image resizing for presentation purposes.

As a result of the drawbacks of both of these methods, as well as the fact that an affine transformation will rarely, if ever, be required within this application, an additional method has been developed by the author that represents a compromise between the standard techniques of image reduction. The method adopted, Visual Basic source code for which is presented in Appendix C5.2, places a grid over the input patch, and replaces the pixels contained within each overlaid grid cell with their average. The contribution of pixels in the input patch that underlie the lines of the overlaid grid (border pixels) is accounted for by weighting their values by their area that lies within the grid cell. Consequently, this method has been termed *weighted average image reduction*. If the input image size modulo the reduced image size is zero then the method degenerates to a case of pixel replacement by a simple average.

Appendix C5: Weighted Image Reduction algorithm



$$O_{i,j} = \frac{\sum_k A_k I_k}{\sum_k A_k}$$

Figure C5.1: Illustration of the weighted image reduction algorithm

The basis of this algorithm is illustrated in Figure C5.1. When an image patch of great dimensions is reduced by weighted average reduction, the effect of the border pixels on the final average is small. However, in the current work, it is expected that the size of the input image patches will be a fairly low multiple of the 7 x 7 reduced patch size. Accordingly, border pixels will maintain their 'importance' within the average.

Figure C5.2 compares the three methods for the specific case of the 32 x 32 pixel input image shown in its top left hand corner. Figure C5.2b shows the effect of image resampling to a 7 x 7 patch; Figure C5.2d shows the effect of image resizing to a 7 x 7 patch and Figure C5.2c shows the results of a weighted average reduction to a 7 x 7 patch. Note that, for presentation purposes, each of the images have undergone a resizing transformation to 128 x 128 pixels (a lossless transformation in terms of information content) and have had their quality reduced by the halftoning operation used in printing.

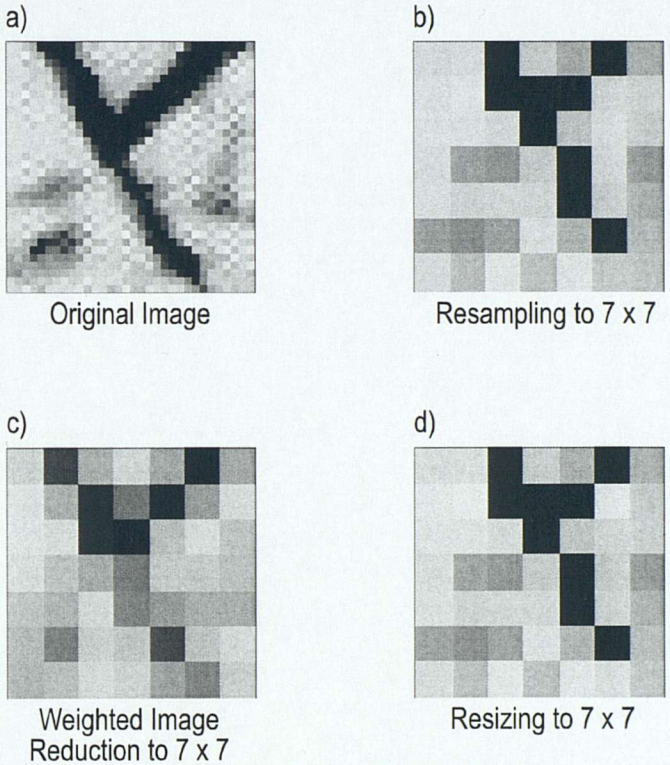


Figure C5.2: Comparison of image patch rescaling methods

It can be seen from Figure C5.2 that, for the case shown, resampling and resizing methods of image patch rescaling yield similar results. These methods also seem to have preserved the grey scale image content of the original patch; however, the spatial distribution of pixel values has suffered distortion or displacement. These detrimental effects are less noticeable with weighted average reduction and the spatial distribution

of pixel values has been better preserved. As is expected through the use of averages, weighted average reduction has resulted in a loss of contrast relative to the original image and the rescaling results using the other two methods. Low contrast has been identified as one of the reasons for poor fracture character classification and consequently, after weighted average reduction, a contrast correction is necessary.

C5.2 Visual Basic for Applications Listing of the Weighted Image Reduction algorithm

```
Function Weighted_Image_Reduce(Source As Variant, BigRows As Integer, BigCols As Integer, _
                               NewRows As Integer, NewCols As Integer, IntFlag As Boolean) As Variant
'
' Takes a 2D image array or patch from an image (Source) of dimensions BigRows X BigCols and applies the
' (area) weighted image size reduction procedure to produce an output image (or patch) which is NewRows x
' NewCols in size. By specifying NewRows or NewCols to be larger than BigRows or BigCols respectively,
' weighted image enlargement is acheived.
'
  Dim Target() As Single
  Dim XStartcell As Double
  Dim YStartcell As Double
  Dim XEndcell As Double
  Dim YEndcell As Double
  Dim Xcell As Double
  Dim Ycell As Double
  Dim i As Integer
  Dim j As Integer
  Dim k As Integer
  Dim l As Integer
  Dim m As Integer
  Dim N As Integer
  Dim A As Integer
  Dim B As Integer
  Dim C As Integer
  Dim D As Integer
  Dim Weight As Single
  Dim XExactFlag As Boolean
  Dim YExactFlag As Boolean
  Dim Area As Double


  XExactFlag = YExactFlag = False
  If BigRows Mod NewRows = 0 Then XExactFlag = True
  If BigCols Mod NewCols = 0 Then YExactFlag = True

  If BigRows = NewRows And BigCols = NewCols Then
'   Nothing to do!
    Weighted_Image_Reduce = BigRows
    Exit Function
  End If

  ReDim Target(1 To NewRows, 1 To NewCols) As Single
'
' Xcell is the width of the pixel in the new image (in terms of pixels in the old image)
' Ycell is the height of the pixel in the new image (in terms of pixels in the old image)
'
  Xcell = BigRows / NewRows
  Ycell = BigCols / NewCols

  For j = 1 To NewCols
    For i = 1 To NewRows
```

```
'
'       i.e.  For a pixel in the new image
'
        XStartcell = (i - 1) * Xcell
        YStartcell = (j - 1) * Ycell
        XEndcell = XStartcell + Xcell
        YEndcell = YStartcell + Ycell
        Target(i, j) = 0
'
'       Get the new pixel limits on the old image
'
'       A is the first pixel in the row to count from
'       B is the last pixel in the row to count to
'       C is the first pixel in the column to count from
'       D is the first pixel in the column to count to
'
        If XStartcell - Int(XStartcell) < 0.0000001 Then
          If XStartcell = 0 Then
            A = XStartcell + 1
          ElseIf XExactFlag Then
            A = XStartcell + 1
          Else
            A = Int(XStartcell)
          End If
        Else
          A = Int(XStartcell) + I
        End If
        If XEndcell - Int(XEndcell) < 0.0000001 Then
          B = Int(XEndcell)
        Else
          B = Int(XEndcell) + 1
        End If
        If YStartcell - Int(YStartcell) < 0.0000001 Then
          If YStartcell = 0 Then
            C = Int(YStartcell) + 1
          ElseIf YExactFlag Then
            C = Int(YStartcell) + 1
          Else
            C = Int(YStartcell)
          End If
        Else
          C = Int(YStartcell) + 1
        End If
        If YEndcell - Int(YEndcell) < 0.0000001 Then
          D = Int(YEndcell)
        Else
          D = Int(YEndcell) + I
        End If
'
'       Add up the weighted grey values from the old image
'
        For N = C To D
          For m = A To B
            k = m - 1
            l = N - I

            Weight = 1
            If l < YStartcell Then
              If k < XStartcell Then
                Weight = ((k + 1) - XStartcell) * ((l + 1) - YStartcell)
              ElseIf k + 1 > XEndcell Then
                Weight = ((XEndcell - k) * ((l + 1) - YStartcell))
              Else
                Weight = ((l + 1) - YStartcell)
              End If
            End If
```

```
            If l + 1 > YEndcell Then
               If k < XStartcell Then
                  Weight = ((k + 1) - XStartcell) * (YEndcell - l)
               ElseIf k + 1 > XEndcell Then
                  Weight = (XEndcell - k) * (YEndcell - l)
               Else
                  Weight = (YEndcell - l)
               End If
            End If

            If k < XStartcell And Weight = 1 Then
               Weight = ((k + 1) - XStartcell)
            End If

            If k + 1 > XEndcell And Weight = 1 Then
               Weight = (XEndcell - k)
            End If
'
'           Add contribution from current pixel on old image
'
            Target(i, j) = Target(i, j) + Source(m, N) * Weight
            Area = Area + Weight
         Next
      Next
'
'     Work out the area weighted average pixel intensity in the new image
'
'     Disable rounding function when trying to find stroke probabilities
'
      If IntFlag Then
         Target(i, j) = Application.Round(Target(i, j) / Area, 0)
      Else
         Target(i, j) = Target(i, j) / Area
      End If
      Area = 0
   Next
 Next
 Weighted_Image_Reduce = Target
End Function
```

C6 Contrast stretching algorithm

C6.1 Description of process

Contrast stretching is a standard process (i.e. one not developed by the author, in contrast to the weighted average reduction procedure of the previous section) that operates on the grey values of pixels within an image while maintaining their relative locations. The method determines the current image contrast of an image or image patch, and then determines an offset and scaling factor that will map the original contrast to the complete dynamic range: the lowest pixel value in the original image is mapped to the value 0 and the highest pixel value is mapped to the value 255. The author's implementation of constrast stretching is presented as a Visual Basic for Applications module, shown in Appendix C6.2. The results of contrast stretching for the original image and its weighted average reduction to 7 x 7 is shown in Figure C6.1 below.

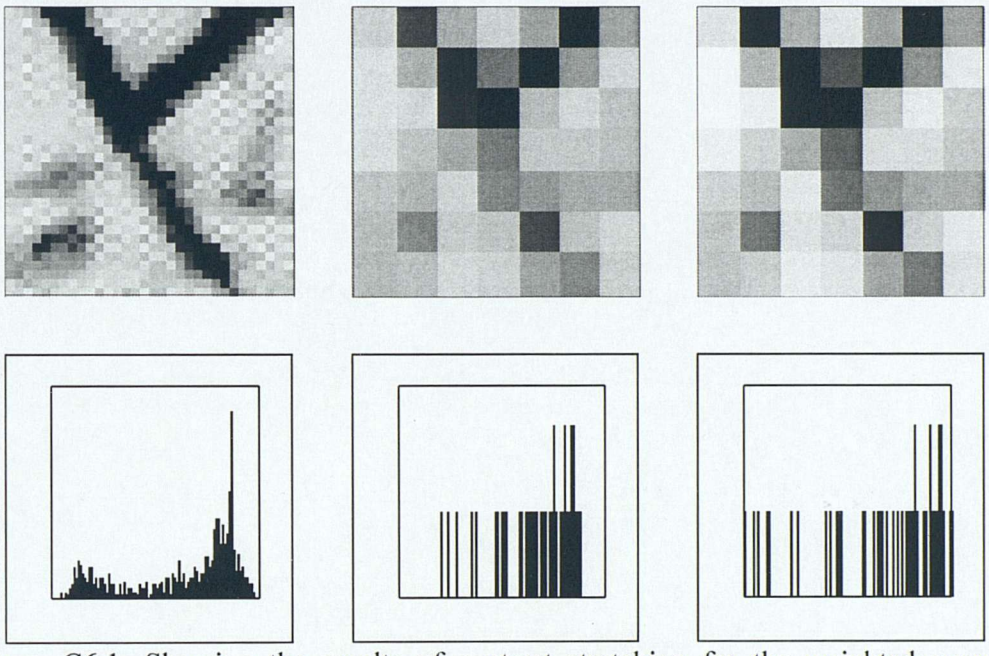


Figure C6.1: Showing the results of contrast stretching for the weighted average reduction of the image patch discussed in Appendix D5.

It can be seen from the histogram of the original image patch shown in Figure C6.1 that it has high contrast. The histogram of the weighted average reduced patch shows that this operation resulted in a truncation of the occupation of grey levels. Contrast stretching of the weighted average reduced patch expands the range of grey levels occupied within its histogram. The number of grey levels occupied within the histogram does not change.

C6.2 Visual Basic for Applications listing of the program

```
Function GreyStretch(Source As Variant) As Variant
'
' This function is a driver for a contrast stretch of an image patch.
' The sums are done by function Grey_Stretch_Path listed below.
' It's a custom MS Excel VBA function which should work for any 2D integer arrays with little modification.
' To use this function:  1) Select a blank area on a worksheet of the target number of rows and columns.
'                        2) Type in the function, providing a range argument covering the image to be resized.
'                        3) Enter the function with <Ctrl> + <Shift> + <Enter> (because it's an array function).
'
' This function can also be called within VBA by passing a 2D array for 'source'
'
  Dim OldX As Integer
  Dim OldY As Integer
  Dim NewX As Integer
  Dim NewY As Integer
  Dim NewImage As Variant

  If IsObject(Source) Then
    ' It must be a selection
    NewX = Selection.Rows.Count
    NewY = Selection.Columns.Count
    OldX = Source.Rows.Count
    OldY = Source.Columns.Count

    If NewX = 1 And NewY = 1 Then
      MsgBox "Error: must have a range with more than 1 cell selected to use this function"
      End
    End If
    If NewX <> OldX Or NewY <> OldY Then
      MsgBox "Warning: Target selection area has different dimensions from input range"
    End If
  Else
    OldX = UBound(Source, 2)
    OldY = UBound(Source, 1)
    NewX = UBound(Source, 2)
    NewY = UBound(Source, 1)
  End If

  NewImage = Grey_Stretch_Patch(Source, OldX, OldY, NewX, NewY)
  GreyStretch = NewImage
End Function
```

Appendix C6: Contrast stretching algorithm

```
Private Function Grey_Stretch_Patch(Patch As Variant, OldX As Integer, OldY As Integer, NewX As Integer,
NewY As Integer) As Variant
  Dim gmax As Integer
  Dim gmin As Integer
  Dim gmid As Double
  Dim xdim As Integer
  Dim ydim As Integer
  Dim NewPatch() As Single
  Dim sf As Double
  Dim i As Integer
  Dim j As Integer

  gmax = Application.Max(Patch)
  gmin = Application.Min(Patch)
  gmid = (gmax + gmin) / 2

  ReDim NewPatch(1 To OldX, 1 To OldY) As Single

  For j = 1 To OldY Step 1
    For i = 1 To OldX Step 1
      NewPatch(i, j) = Patch(i, j) + (127.5 - gmid)
    Next
  Next

  gmax = gmax + (127.5 - gmid)
  gmin = gmin + (127.5 - gmid)

  If gmax <> gmin Then
    sf = 255# / CDbl(gmax - gmin)
  End If

  For j = 1 To OldY Step 1
    For i = 1 To OldX Step 1
      NewPatch(i, j) = CInt((NewPatch(i, j) - 127.5) * sf + 127.5)
    Next
  Next

  Grey_Stretch_Patch = NewPatch

End Function
```

C7 Program listing of the Fracture Character mutation algorithm

```
Function Mutate_Fracture(OldFrac As Variant, MutRate As Single) As Variant
'
' Mutates a fracture character provided as a 2D array in OldFrac
' MutRate is a mutation probability: MutRate = 1, mutation is certain; Mutate = 0, mutation never occurs
'
  Dim i As Integer
  Dim j As Integer
  Dim RowOff As Integer
  Dim ColOff As Integer
  Dim Mutated As Boolean
  Dim Count As Integer
  Dim MyRnd As Single
  Dim Frac(1 To 7, 1 To 7) As Integer
  Dim Length As Integer

  For j = 1 To 7
    For i = 1 To 7
      Frac(j, i) = OldFrac(j, i)
    Next
  Next

  For j = 1 To 7
    For i = 1 To 7
      If i = 4 And j = 4 Then
      Else
      ' Mutate the active bits in the Fracture Patch
        If OldFrac(j, i) = 1 And Rnd < MutRate Then
          Do
            GetOffsets i, j, RowOff, ColOff
            If Frac(j + RowOff, i + ColOff) = 0 Then
              Frac(j, i) = 0
              Frac(j + RowOff, i + ColOff) = 1
              Mutated = True
              Count = 0
            ElseIf Count = 8 Then
              Mutated = True
              Count = 0
            Else
              Mutated = False
              Count = Count + 1
            End If
          Loop While Not Mutated
        End If
      End If
    Next
  Next
  Mutate_Fracture = Frac
End Function
```

Appendix C7: Program listing of the Fracture Character mutation algorithm

```
Private Sub GetOffsets(i As Integer, j As Integer, RowOff As Integer, ColOff As Integer)
  Dim MyRnd As Single

  MyRnd = Rnd
  If j <> 1 And j <> 7 And i <> 1 And i <> 7 Then
        ' It's in the middle somewhere
    If MyRnd < 0.125 Then
      RowOff = -1
      ColOff = 0
    ElseIf MyRnd < 0.25 Then
      RowOff = -1
      ColOff = -1
    ElseIf MyRnd < 0.375 Then
      RowOff = 0
      ColOff = -1
    ElseIf MyRnd < 0.5 Then
      RowOff = 1
      ColOff = -1
    ElseIf MyRnd < 0.625 Then
      RowOff = 1
      ColOff = 0
    ElseIf MyRnd < 0.75 Then
      RowOff = 1
      ColOff = 1
    ElseIf MyRnd < 0.875 Then
      RowOff = 0
      ColOff = 1
    Else
      RowOff = -1
      ColOff = 1
    End If
  ElseIf j = 1 And i = 1 Then
    ' Top LHS
    If MyRnd < 0.333333333 Then
      RowOff = 1
      ColOff = 0
    ElseIf MyRnd < 0.66666666 Then
      RowOff = 1
      ColOff = 1
    Else
      RowOff = 0
      ColOff = 1
    End If
  ElseIf j = 7 And i = 1 Then
    ' Bottom LHS
    If MyRnd < 0.333333333 Then
      RowOff = 0
      ColOff = 1
    ElseIf MyRnd < 0.66666666 Then
      RowOff = -1
      ColOff = 1
    Else
      RowOff = -1
      ColOff = 0
    End If
  ElseIf j = 7 And i = 7 Then
    ' Bottom RHS
    If MyRnd < 0.333333333 Then
      RowOff = -1
      ColOff = 0
    ElseIf MyRnd < 0.66666666 Then
      RowOff = -1
      ColOff = -1
    Else
      RowOff = 0
      ColOff = -1
```

Appendix C7: Program listing of the Fracture Character mutation algorithm

```
    End If
  ElseIf j = 1 And i = 7 Then
  ' Top RHS
    If MyRnd < 0.333333333 Then
      RowOff = 0
      ColOff = -1
    ElseIf MyRnd < 0.66666666 Then
      RowOff = 1
      ColOff = -1
    Else
      RowOff = 1
      ColOff = 0
    End If
  ElseIf j = 1 Then
  ' Top Row
    If MyRnd < 0.2 Then
      RowOff = 0
      ColOff = -1
    ElseIf MyRnd < 0.4 Then
      RowOff = 1
      ColOff = -1
    ElseIf MyRnd < 0.6 Then
      RowOff = 1
      ColOff = 0
    ElseIf MyRnd < 0.8 Then
      RowOff = 1
      ColOff = 1
    Else
      RowOff = 0
      ColOff = 1
    End If
  ElseIf i = 1 Then
    ' Left side
    If MyRnd < 0.2 Then
      RowOff = 1
      ColOff = 0
    ElseIf MyRnd < 0.4 Then
      RowOff = 1
      ColOff = 1
    ElseIf MyRnd < 0.6 Then
      RowOff = 0
      ColOff = 1
    ElseIf MyRnd < 0.8 Then
      RowOff = -1
      ColOff = 1
    Else
      RowOff = -1
      ColOff = 0
    End If
  ElseIf j = 7 Then
  ' Bottom side
    If MyRnd < 0.2 Then
      RowOff = 0
      ColOff = 1
    ElseIf MyRnd < 0.4 Then
      RowOff = -1
      ColOff = 1
    ElseIf MyRnd < 0.6 Then
      RowOff = -1
      ColOff = 0
    ElseIf MyRnd < 0.8 Then
      RowOff = -1
      ColOff = -1
    Else
      RowOff = 0
      ColOff = -1
```

Appendix C7: Program listing of the Fracture Character mutation algorithm

```
    End If
  ElseIf i = 7 Then
  ' Right side
    If MyRnd < 0.2 Then
      RowOff = -1
      ColOff = 0
    ElseIf MyRnd < 0.4 Then
      RowOff = -1
      ColOff = -1
    ElseIf MyRnd < 0.6 Then
      RowOff = 0
      ColOff = -1
    ElseIf MyRnd < 0.8 Then
      RowOff = 1
      ColOff = -1
    Else
      RowOff = 1
      ColOff = 0
    End If
  End If
End Sub
```

C8 Fracture Classifier Weights and Thresholds

ANNs are effectively multivariate regression techniques. If results obtained using ANNs are to be utilised by workers other than those who originated the work, it is vital that the structure, ANN type and values of weights and thresholds are presented, just as the fitting parameters and quality of fit must be reported for a multivariate regression exercise. This is the reason why the following data are presented in this Appendix; they are required if anyone else other than the author wishes to implement the specific fracture classifier discussed in the main text of Chapter 6.

Back Propagation network.

Number of layers = 3
layers numbered from 1 (input) to 3 (output)

Layer dimensions:
layer 1: (49, 1)
layer 2: (8, 1)
layer 3: (8, 1)

Cycles trained = 8000

Weights initialized from -0.300 to 0.300

Noise = 0.00000, decay = 0.00000

Tolerance = 0.00000

Learning rates for layers:
layer 2: 0.00025
layer 3: 0.00025

Momentum terms for layers:
layer 2: 0.95000
layer 3: 0.95000

Biases for hidden and output layers:
layer 2 biases:
2.36818 -1.73773 -0.81182 -0.38700 0.27892 -0.54066 -0.91907 0.47309
layer 3 biases:
-0.83882 -0.10490 -1.17652 0.11643 -1.04524 -1.63180 -0.34231 -0.38921

Weight matrices: (to unit I from unit J, J changing fastest)
layer 1 to 2 weights:
-0.02539 0.01131 -0.01092 0.65804 -0.08820 0.71871 1.34629 0.30457
0.00820 0.00993 0.50498 0.03161 0.30022 1.27424 0.29365 0.46259
0.28946 -1.14397 -0.23254 -0.21887 0.12890 0.82701 0.26937 0.26009
-0.35463 -0.14308 0.09542 0.23768 0.55510 0.26128 0.11727 -0.28610
1.01962 0.50412 0.23334 -0.30370 0.13071 -0.32822 -0.64052 -0.21685
2.00109 0.87473 0.57679 0.77585 -0.96780 -1.95034 -0.92339 -0.02012
0.95434 0.03304 0.07559 -0.02447 -0.13078 -0.02701 0.69849 0.63066
0.00115 0.07879 -0.01905 0.00445 -0.15202 0.48564 1.37122 0.49831

-0.13822 -0.17934 -0.63901 -0.30109 -0.22630 0.59865 0.69788 -0.02496
-0.72858 0.10732 -0.75381 -0.04237 0.34630 0.45168 0.01485 -0.54353
-1.14160 -1.28432 -0.44692 -0.13791 0.27434 0.40833 -1.13018 -0.96642
-1.02019 -1.19721 -0.59975 0.17169 -0.59455 -0.67807 -1.31517 -1.30481
-0.67443 -0.96493 -3.84956 -2.56901 -0.17680 0.55106 -0.29649 0.06803
0.39506 -3.04494 -2.34488 -0.28808 -0.25656 0.16308 0.21792 0.52452
-0.05076 -0.22902 -0.59787 0.47789 -0.12673 0.45288 0.59895 0.60063
0.46732 0.05848 -0.28232 0.29198 0.29607 0.67606 0.27716 0.71535
-0.10035 0.18631 0.07683 0.57675 0.22614 0.36573 0.53556 0.25398
0.03764 -0.21308 0.11703 -0.05044 0.05150 0.46518 0.07051 0.30399
0.62944 0.53639 -0.20913 -1.01983 -0.39655 0.71152 0.17199 -0.68212
-2.45864 -2.67705 -0.28390 0.33917 0.04474 0.39575 -0.38048 -1.26826
-1.79913 0.63939 0.62316 0.45656 -0.08071 -0.72999 -0.05576 -0.42373
0.19011 0.02057 -0.28733 -2.13332 -0.36338 -0.15873 0.26782 0.37889
-0.08916 -0.36907 -0.97273 -0.86979 0.51035 0.08032 1.21591 -0.01382
-0.82606 -0.89070 -0.92740 0.11326 1.25884 0.44265 -0.62228 -0.80836
-0.75053 0.36926 1.36280 1.30873 0.99881 0.70545 -0.10010 -0.06617
0.22490 0.83726 0.62824 0.62026 -0.41163 -0.14660 -0.15186 -0.23766
-0.19871 0.34042 -1.23979 -0.96134 -1.14979 -0.78095 -0.30370 -0.46780
-0.52666 -1.55151 -1.51276 -1.26169 0.28702 -0.44174 -0.03433 -0.09845
-0.98190 -0.96951 -0.83802 -0.91766 -0.10708 0.16820 -0.13170 0.76611
-0.19742 -0.60701 0.29641 0.19476 0.85725 1.16822 0.62642 0.35057
0.30594 0.16561 0.69509 1.32009 1.48756 0.02624 0.09292 0.15086
0.17870 -0.34888 0.97659 0.81362 -0.03029 -0.07777 0.00270 -0.27401
0.48055 -0.30221 0.35492 0.87418 0.38828 0.05128 0.26587 0.27640
0.19788 -0.19518 1.39328 0.27253 0.74275 -1.11597 0.31978 -0.01195
-0.39332 0.19541 -0.60718 -1.17708 -0.18467 -0.10903 0.50766 0.07534
-1.69838 -2.00300 -0.63619 0.15685 0.22586 0.10110 1.17010 -3.09773
-2.46419 -0.36462 -0.05909 -0.82310 -0.14225 -0.07304 -0.68849 -0.49629
0.95444 2.55447 1.54270 -1.18260 -1.53972 -1.10907 0.13111 1.00493
1.30798 1.25948 0.69306 -1.61886 -0.44158 -0.26223 0.45287 1.08227
0.75981 -0.15920 -0.56600 -1.06469 -0.44780 -0.68775 0.01906 0.30544
-0.35368 -0.25909 -0.24125 -0.43905 -0.52534 -0.46796 -0.35979 0.30360
-0.04123 0.76792 -0.19653 -0.00404 -0.43740 -0.39666 0.16053 0.13137
0.50833 0.19142 -0.48448 -0.22009 -0.08436 -0.39346 -0.16234 -0.48010
-0.52816 -0.00219 0.84592 -0.05350 1.73309 1.37735 0.05395 0.16827
-0.02082 -0.14569 0.10372 -0.33772 0.83432 -0.17882 -0.15692 0.33963
0.08030 -0.75749 -1.25241 -0.95277 -0.01781 0.40521 0.35984 -0.29095
-1.28321 -1.82840 -2.86854 0.43169 -0.07135 -0.13914 -0.28255 -0.55482
-1.14261 -1.24166 0.76303 -0.13625 0.09054 0.31411 0.31598 0.28940
0.66022 0.83389 0.64019 0.12128 0.82330 0.29842 0.34190 1.08808

layer 2 to 3 weights:

6.04927 7.69358 1.66995 -14.87998 2.53464 1.75653 1.47742 0.88289
0.27961 -2.33659 0.97341 -1.11665 -2.08098 -1.45738 1.00156 -14.63479
13.14395 -11.42241 0.55909 3.81731 6.78231 1.60540 -2.65679 1.45319
-11.22694 -13.79169 1.07918 -5.89678 -2.39025 -0.23242 -3.07346 2.18016
1.49043 2.91302 1.12028 0.85340 -1.82253 -14.53103 -1.14958 1.87466
5.82419 -1.00293 2.17408 3.09346 -15.71145 3.15186 -5.18212 2.58720
0.27883 -0.25045 -13.53951 -0.84994 -0.88816 -0.46329 0.10185 -1.89102
1.33683 -3.12740 0.50855 0.42788 -5.76779 0.41062 15.45011 1.14573