USING NEURAL NETWORKS TO POSITION LIVE LOADS ON BRIDGE PIERS

By

MARK ERIK WILLIAMS

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2000

ACKNOWLEDGMENTS

I express sincere appreciation to my advisor, Dr. Marc Hoit, for his guidance, encouragement, motivation, and support during this research. Secondly, I thank my co-chair, Dr. Kurtis Gurley, and the remaining committee members -- Dr. Gary Consolazio, Dr. Fernando Fagundo, and Dr. Michael Rabens -- for their supportive effort and additional insight into my work.

I am equally grateful to my wife Michelle for her patience and loving support. I am particularly indebted to her for the encouragement that guided me through the completion of this dissertation.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS iii

LIST OF TABLES viii

LIST OF FIGURES x

ABSTRACT xiv

CHAPTERS

1 INTRODUCTION 1

1.1 Background 1
1.2 Present Research 3
1.2.1 Modeling of Highway Bridges 3
1.2.2 Modeling of Live Loads 4
1.2.3 Neural Network Predictions of Live Load Positions 6
1.3 Literature Review 7
1.3.1 Modeling of Highway Bridges 7
1.3.2 Modeling of Live Loads 9
1.3.3 Neural Network Applications to Structural Analysis 12
1.4 Organization of Dissertation 14

2 MODELING OF HIGHWAY BRIDGES 16

2.1 Introduction 16
2.2 Modeling of Structural Components 16
2.2.1 Modeling of Deck Slab 17
2.2.2 Modeling of Girders 19
2.2.3 Modeling of Diaphragms 20
2.2.4 Modeling of Pier Supports 21
2.3 Assembled Bridge Model 21
2.4 Modeling Vehicle Live Loads 23
2.4.1 Design Truck 23
2.4.2 Tandem Truck 25
2.4.3 Lane Load 25
2.5 Two-dimensional Application of Live Loads 26
2.5.1 Single Truck Loading 27

2.5.2 Dual Truck Loading 27
2.5.3 Tandem Loading 28
2.6 Three-dimensional Live Load Models 28
2.6.1 Wheel Load Distribution 29
2.6.2 Lane Load Distribution 30
2.6.3 Live Load Reduction 31
2.7 Live Load Generation 32
2.7.1 Design Lane Configuration 33
2.7.2 Transverse Live Load Placement 34
2.7.3 Live Load Combinations 35

3 MODELING OF BRIDGE PIERS 37

3.1 Introduction 37
3.2 Modeling of Bridge Pier Components 37
3.2.1 Modeling of Piles and Drilled Shafts 38
3.2.2 Modeling of Soil 41
3.2.3 Modeling of Pile Cap 43
3.2.4 Modeling of Pier Columns 45
3.2.5 Modeling of Pier Cap 47
3.2.6 Modeling of Bearing Pads 48
3.3 Complete Pier Model 48
3.4 Maximum Force Effects 49
3.4.1 Failure Ratio 50
3.4.2 SRSS Criteria 53
3.4.3 Maximum Force Effects in Pier Components 54
3.5 Load Live Transfer to Piers 54
3.5.1 Load Transfer to Bearing Pads 55
3.5.2 Case Studies in Live Load Generation 56

4 NEURAL NETWORKS 58

4.1 Introduction 58
4.2 Network Architecture and Operation 59
4.3 Problem Representation 63
4.4 Network Learning Models 64
4.5 Supervised Training Techniques 67
4.6 Optimal Solution Techniques 70
4.7 Backpropagation Training 74
4.7.1 Single Example and Batch Training 77
4.7.2 Adaptive Learning 78
4.7.3 Momentum 79
4.7.4 Second Order Techniques 82
4.8 Refined Network Topology 83
4.8.1 Number of Hidden Layers 83
4.8.2 Number of Hidden Neurons 85

4.9 Network Pruning 85

5 POSITIONING LIVE LOADS ON SINGLE COLUMN PIERS 88

5.1 Introduction 88
5.2 Overview of Network Creation and Operation 89
5.3 Encoding Structural Behavior 90
5.4 Dimensionless Network Parameters 91
5.4.1 Input Parameters 91
5.4.2 Output Parameters 96
5.5 Maximum Force Combinations 97
5.5.1 Failure Ratio 98
5.5.2 Square Root of Sum of Squares (SRSS) 99
5.6 Critical Load Positions and Load Symmetry 99
5.7 Networks for Predicting Critical Load Positions 100
5.7.1 Maximum Pile Force Combination 101
5.7.2 Maximum Pier Column Force Combination 102
5.7.3 Maximum Pier Cap Shear Force 103
5.7.4 Maximum Pier Cap Bending Moment 103
5.8 Network Training Set 104
5.9 Implementation of Networks 105
5.10 Network Results and Validation 106
5.10.1 Network Results Using NetSim 107
5.10.2 Network Results Using MatLab 111
5.11 Final Live Load Positioning 113
5.12 Comparison of Predicted and Actual Design Loads 114

6 POSITIONING LIVE LOADS ON MULTIPLE COLUMN PIERS 119

6.1 Introduction 119
6.2 Encoding Structural Behavior 120
6.3 Dimensionless Network Parameters 120
6.3.1 Input Parameters 121
6.3.2 Output Parameters 122
6.4 Maximum Force Combinations 122
6.5 Critical Load Positions and Load Symmetry 123
6.6 Networks for Predicting Critical Load Positions 124
6.6.1 Maximum Pile Force Combination 125
6.6.2 Maximum Pier Column Force Combination 126
6.6.3 Maximum Pier Cap Shear Force 127
6.6.4 Maximum Pier Cap Bending Moment 128
6.7 Network Training Set 128
6.8 Implementation of Networks 129
6.9 Network Results and Validation 130
6.9.1 Network Results Using NetSim 130
6.9.2 Network Results Using MatLab 134

6.10 Final Live Load Positioning 136
6.11 Comparison of Predicted and Actual Design Loads 137

7 NETWORK TOPOLOGY OPTIMIZATION 143

7.1 Introduction 143
7.2 Network Pruning Techniques 144
7.2.1 Exhaustive Search 144
7.2.2 Penalty Methods 145
7.2.3 Weight Elimination 146
7.3 A Simple Implementation of Network Pruning 149
7.4 Modifications to the NetSim Program 154
7.5 Pruning Single Column Pier Networks 155
7.6 Pruning Multiple Column Pier Networks 159

8 CONCLUSION 162

8.1 Live Load Modeling and Generation 162
8.2 Live Load Application to Bridge Piers 163
8.3 Neural Network Prediction of Load Positions 164
8.4 Network Topology Optimization 165
8.5 Suggestions for Future Study 165

APPENDICES

A SINGLE COLUMN PIER NETWORK TRAINING PARAMETERS 167

B MULTIPLE COLUMN PIER NETWORK TRAINING PARAMETERS 172

REFERENCES 182

BIOGRAPHICAL SKETCH 187

LIST OF TABLES

Table page

2.1 Multiple Presence Factors. 31

5.1 Dimensional variables (Single Column Piers)....94

5.2 Network Error Statistics for Single Column Piers....110

5.3 Pile Force Combination Training Results (Single Column Pier) 111

5.4 Column Force Combination Training Results (Single Column Pier) 112

5.5 Pier Cap Shear Training Results (Single Column Pier)....112

5.6 Pier Cap Moment Training Results (Single Column Pier) 112

6.1 Network Error Statistics for Multiple Column Piers 133

6.2 Pile Force Combination Training Results (Multiple Column Pier)....134

6.3 Column Force Combination Training Results (Multiple Column Pier)....134

6.4 Pier Cap Shear Training Results (Multiple Column Pier) 135

6.5 Pier Cap Moment Training Results (Multiple Column Pier)....135

7.1 Pruning Results (Single Column Piers)....158

7.2 Pruning Results (Multiple Column Piers) 161

A.1 Single Column Piers Input Parameters....167

A.2 Single Column Piers Output Parameters - Pile Force Combination....168

A.3 Single Column Piers Output Parameters - Column Force Combination 169

A.4 Single Column Piers Output Parameters - Pier Cap Shear Force....170

A.5 Single Column Piers Output Parameters - Pier Cap Bending Moment 171

B.1 Multiple Column Piers Input Parameters.....172

B.2 Multiple Column Piers Output Parameters - Pile Force Combination.....174

B.3 Multiple Column Piers Output Parameters - Column Force Combination.....176

B.4 Multiple Column Piers Output Parameters - Pier Cap Shear Force 178

B.5 Multiple Column Piers Output Parameters - Pier Cap Moment.....180

LIST OF FIGURES

Figure page

2.1 Bilinear and Quadratic Elements 17

2.2 Zero-Energy Mode in Lagrangian Quadratic Element 18

2.3 Standard Beam Element 19

2.4 Girder Model with Rigid End Offsets 20

2.5 Cross-section of Slab and Girder Assembly 22

2.6 Complete Bridge Model with Four Girders 23

2.7 Design Truck Specifications 24

2.8 Tandem Truck Specifications 25

2.9 HL-93 Live Load Model 26

2.10 Single Truck Loading 27

2.11 Dual Truck Loading 28

2.12 Tandem Truck Loading 28

2.13 Equivalent Nodal Loads 30

2.14 Equivalent Lane Load 31

2.15 Transverse Live Load Positions 35

3.1 Angle and Displacement Relations for Discrete Element 40

3.2 Typical P-Y and T-Z Soil Curves 42

3.3 Pile-Soil Model 43

3.4 Shell and Pile Element Connection 45
3.5 Connector Elements at Base of Column 46
3.6 Column to Pier Cap Connection 47
3.7 Typical Finite Element Model of Pier 49
3.8 Interaction Diagram and Failure Surface 51
3.9 Illustration of Failure Ratio 52
3.10 Significant Forces on Piles, Pier Columns, and Pier Cap 53
3.11 Live Load Transfer to Piers 55
3.12 Maximum Force Effects for Case Study 57
4.1 Typical Feed Forward Neural Network 60
4.2 Nonlinear Computing Neuron 62
4.3 Sigmoid Activation Functions 63
4.4 Errors in Training and Validation 69
4.5 Hypothetical Training Error Surface 72
4.6 Variation of Learning Rate 79
4.7 Oscillating Solutions and Momentum 81
4.8 Escaping Shallow Local Minima 81
4.9 Mapping Network Decision Boundaries (After Bishop) 84
5.1 Overview of Network Training and Operation 90
5.2 Single Column Pier Configuration Variables 92
5.3 Normalized Placement of Live Loads Across the Bridge 96
5.4 Network Configuration for Single Column Piers 101
5.5 Maximum Force Combination in a Single Pier Column 102

5.6 Maximum Pier Cap Shear in a Single Column Pier 103
5.7 Maximum Pier Cap Moment in a Single Column Pier 104
5.8 Pile Force Combination Network Error (Single Column Pier) 108
5.9 Column Force Combination Network Error (Single Column Pier) 108
5.10 Pier Cap Shear Force Network Error (Single Column Pier) 109
5.11 Pier Cap Bending Moment Network Error (Single Column Pier) 109
5.12 Final Load Positioning (Single Column Pier) 113
5.13 Single Column Pier Configuration used for Load Comparison 114
5.14 Pile Force Combination Force Comparison (Single Column Pier) 115
5.15 Column Force Combination Force Comparison (Single Column Pier) 116
5.16 Pier Cap Shear Force Comparison (Single Column Pier) 117
5.17 Pier Cap Bending Moment Comparison (Single Column Pier) 118
6.1 Multiple Column Pier Configuration Variables 121
6.2 Network Configuration for Multiple Column Pier 125
6.3 Maximum Force Combination in a Multiple Column Pier 127
6.4 Maximum Pier Cap Shear in a Multiple Column Pier 127
6.5 Maximum Pier Cap Moment in a Multiple Column Pier 128
6.6 Pile Force Combination Network Error (Multiple Column Pier) 131
6.7 Column Force Network Error (Multiple Column Pier) 131
6.8 Pier Cap Shear Force Network Error (Multiple Column Pier) 132
6.9 Pier Cap Moment Force Network Error (Multiple Column Pier) 132
6.10 Final Load Positioning (Multiple Column Pier) 136
6.11 Multiple Column Pier Configuration used for Load Comparison 137

6.12 Pile Force Combination Force Comparison (Multiple Column Pier)..... 139

6.13 Column Force Combination Force Comparison (Multiple Column Pier)..... 140

6.14 Pier Cap Shear Force Comparison (Multiple Column Pier) 141

6.15 Pier Cap Bending Moment Comparison (Multiple Column Pier)..... 141

7.1 Weight Elimination -- Optimal Brain Damage and Optimal Brain Surgeon..... 149

7.2 Network Error for a Single Weight..... 151

7.3 Training Along the Error Surface..... 152

7.4 Sensitivity Diagrams Before Pruning (Single Column Piers)..... 157

7.5 Sensitivity Diagrams Before Pruning (Multiple Column Piers) 160

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

USING NEURAL NETWORKS TO POSITION
LIVE LOADS ON BRIDGE PIERS

By

Mark Erik Williams

May 2000

Chairman: Marc I. Hoit
Major Department: Civil Engineering

The structural analysis of bridges and their supporting pier foundations is an intriguing subject that is frequently debated among many structural engineers. Unlike most building designs, modern highway bridge designs must consider the variability of loads on the bridge and the uncertainty of their application. In particular, the application of vehicular live loads is not straightforward. At any given time, vehicles can traverse the bridge at unknown speeds and paths producing different force effects. Fortunately, the correct application of vehicle loads to the bridge superstructure has been documented by the American Association of State, Highway, and Transportation Officials (AASHTO) as well as by other research institutions. However, the subsequent application of these live loads to the supporting bridge piers is still not well understood and is only briefly addressed by the AASHTO-LRFD Design Specifications.

A common situation arises when determining the maximum force effects on the superstructure and pier foundation. The application of vehicular live loads to the superstructure to achieve the maximum force effects in the superstructure does not necessarily produce the maximum force effects in the pier foundation. In other words, an entirely different live load application may produce the maximum force effects in the pier foundation.

An exhaustive study of live load position combinations across the bridge deck can produce thousands of possible design load cases. The most critical live load positions can then be determined by studying the results of all the live load combinations. In an attempt to circumvent this tedious process, eight neural networks have been trained to predict the live load positions across the bridge width to achieve the maximum force effects in the pier foundation. The networks use geometric input parameters that describe the structure and produce truck and lane load positions in each design lane for output. Results are very encouraging though further training and network enhancements are still possible.

CHAPTER 1
INTRODUCTION

1.1 Background

The design and analysis of modern highway bridges remains a complex undertaking. Highway bridges, like many other structures, are subjected to uncertain and variable loads throughout their life span. At any given time, the bridge must sustain a combination of applied loads without allowing any significant structural damage to occur. In this regard, highway bridges are particularly susceptible to uncertain loading effects due to the presence of various combinations of live loads on the bridge at one time. Because of the complexity in loading and behavior of most bridges, some computer software must be utilized during the design process.

With the advent of faster computers in the past decade, bridge designers are now performing analyses of more complex bridge structures using various loading schemes. While there has been tremendous growth in the capabilities of structural analysis software, much of the effort has focussed on expanding general-purpose finite element programs. As a result, most bridge designers must utilize a general-purpose program to model a specific complex bridge structure or rely on very simplified bridge modeling programs. Both methods can produce an inaccurate analysis of the bridge. The use of general-purpose finite element programs can produce incorrect results for the inexperienced bridge modeler. On the other hand, the use of a very simplified program fails to include the complete structural behavior of the bridge. Either way, the determination of the correct

structural behavior is unnecessarily hampered by the lack of efficient bridge analysis software.

The demand for effective bridge analysis software is certainly evident. It is estimated that at least half of the 500,000 bridges in the United States are in need of some repair (Better Roads 1994). With the design of new replacement bridges or the retrofitting of current bridges, the demand for some automation in bridge analysis programs is increasing rapidly. Foremost, bridge designers can clearly benefit from an automatic generation of the complete finite element model for a bridge. The model could be generated based on geometric parameters provided by the designer for example. Secondly, in addition to some automated procedure to model the bridge superstructure and support piers, methods are needed to automate the live loading processes. The live loading of bridges can be quite cumbersome, requiring many different load case applications of wheel and lane loads to determine the maximum force effects.

A comprehensive analysis of vehicle live loading on bridges must consider the loading of multiple lanes across the width of the bridge. This task is not only computationally demanding but also requires much effort on the part of the bridge designer, since multiple structural components must be designed to sustain live loading. Any methods or guidelines that automate the live loading process are certainly advantageous. Although the live loads are applied to the bridge superstructure, any automated procedures can be extended to apply the live loads to the supporting pier structure. This automation of the live loading process would allow designers to allocate more time understanding the structural behavior, thereby wasting less time conducting time consuming analyses.

1.2 Present Research

The research presented in this dissertation focuses on three primary aspects of highway bridge analysis. They are:

1. Modeling the structural components that comprise the bridge superstructure and support pier foundation in order to analyze the behavior of the bridge system under vehicular live loading.

2. Developing a live load generation routine to determine which combinations of live loads produce the maximum force effects in the different bridge pier components.

3. Developing separate neural networks to predict the position of the live loads in each traffic lane across the bridge width to produce the maximum force effects in the bridge pier components.

Each of these objectives addresses an area that is currently deficient from an analytical viewpoint. The objectives seek to expand the current understanding of bridge behavior by providing a comprehensive insight into the live loading of modern highway bridges. A detailed description of each of these objectives follows.

1.2.1 Modeling of Highway Bridges

The finite element method (FEM) has become the most widespread technique for analyzing the behavior of modern highway bridges. Using the FEM, all of the structural components can be modeled individually and assembled to form a complete model of the bridge system. In this investigation, the complete model includes the bridge superstructure as well as the supporting pier foundations. Depending on the complexity of this complete bridge model, the results from the finite element analysis (FEA) can very closely match the true behavior of the bridge under externally applied loads.

Many different types of finite element models have been constructed to analyze highway bridges. The large number of different bridge models is due in part to the uniqueness of the bridges. Since bridges are individually designed to achieve a specific functionality, it is rare to find two bridges with the same geometric and material properties. As a result, most highway bridge designers create a unique finite element model for each bridge. In this regard, the work presented in this dissertation does not propose a new procedure for the modeling of highway bridges. Rather, this work uses a refined highway bridge superstructure model based on the previous work of other researchers to determine the subsequent effects on the supporting bridge piers. That is, in this dissertation, the finite element models of the bridge superstructure are used as a primary tool for analyzing the behavior of bridge piers.

The primary focus of this dissertation is the investigation of vehicle live loads on bridge piers. In the procedures described herein, finite element models of bridge piers are created to study the behavior of the piers under live load application. This live load application to the bridge piers is a two step process. The live loads are first applied to the finite element model of the bridge superstructure. These externally applied loads are then transferred to the bridge piers via the bearing pad reactions for the girders in the bridge. At this point the analysis can proceed to determine the behavior of the interior pier supports under the imposed live loading.

1.2.2 Modeling of Live Loads

Many different models exist for the modeling of vehicle live loads on highway bridges. Most bridge analyses utilize some form of the American Association of State, Highway, and Transportation Officials (AASHTO) live load specifications to apply

vehicle loads to the bridge. These vehicle models were developed primarily to determine the forces and stresses in the bridge superstructure for design purposes. As a result, most of the research on vehicle live loading focuses on the correct placement of the live loads on the bridge to achieve the desired maximum force effect in a particular component of the superstructure. There has been little work to determine the correct placement of the live loads on the bridge to determine the maximum force effects in the supporting pier foundation.

For most bridge designs, the live loads are positioned on the bridge to produce the maximum desired force effect in the bridge superstructure. For example, to determine the maximum moment in a girder for a simply supported span, the live load is placed near the center of the span. Equally important, to determine the maximum shear force in a girder in the bridge, the live load is positioned close to the end of the girder, near the support. Quite often the procedure to determine the maximum force effects in the support pier is greatly simplified. For many interior bridge pier designs, the maximum shear force in the girder at the support is applied to the supporting pier foundation to determine the forces and stresses in the pier. The work in this dissertation will show that the live load positions that cause the maximum force effects in the bridge superstructure do not necessarily cause the maximum force effects in the interior pier foundation. The vehicles must be repositioned along the bridge in order to determine the maximum forces in the piers.

This dissertation will specifically address the analysis of bridge piers under vehicle live loading. A methodology will be developed to generate all possible combinations of live loads that may cause the maximum force effects in the interior pier supports. Only after generating all possible combinations of live loads can some

procedure be developed for choosing the worst load cases for the bridge piers. This selective procedure will utilize neural networks to predict the position of the live loads in each of the traffic lanes to achieve the maximum force effects in the bridge piers.

1.2.3 Neural Network Predictions of Live Load Positions

Neural networks are numerical models based on human cognition. A neural network attempts to mimic the function of the human brain by forming relationships between various input stimuli and the subsequent response of the brain. While the biological applications of neural networks are useful, it has been the extended applications of neural networks to other fields that have gathered notoriety. In practical applications, neural networks are most often used to determine the complex relationship between input and output parameters for a given problem. Since the fundamental operation of neural networks is not dependent on the type of any physical quantities, the neural networks can be applied to many different types of physical systems. Therefore, whether the problem is pattern recognition, financial forecasting, or any other predictive application, the functionality of the network remains the same.

In this dissertation, neural networks will be utilized for the application of live loads on highway bridges. In particular, neural networks will be developed to predict the position of the live loads in each traffic lane that produce the worst force effects in the interior pier support. To arrive at this prediction of live load positions, network input parameters must be specified. In this particular application of neural networks, the input parameters describe the geometry of the bridge superstructure and interior pier support. A successful application of the neural networks would produce a relationship between the geometric input parameters and the output prediction of live load positions. Since the

problem involves the interaction of many design variables, it is unlikely that the solution is obvious from visual inspection. This shortfall is overcome by the functionality of neural networks, which are superior to statistical regression techniques for recognizing patterns between input and output parameters.

Since there are different maximum force effects for each structural component of the pier, different neural networks will be developed to predict the critical load positions for each force effect. For this investigation, a total of eight separate neural networks will be developed that correspond to the four identified maximum force effects in the structural components for both single column and multiple column piers. These four maximum force effects are identified as the maximum force combination in the piles and pier columns as well as the maximum shear and bending moment in the pier cap. These four maximum force effects will most likely control the design of bridge pier for vehicle live loading.

1.3 Literature Review

The research presented herein focuses on the modeling and computation analysis of highway bridge structures. The following sections discuss in detail the contributions of previous researchers to the understanding of highway bridge behavior.

1.3.1 Modeling of Highway Bridges

Modern highway bridges are composed of many unique structural components. In order to analyze the bridge using the finite element method, each component must be modeled to represent the correct structural behavior. Given the uniqueness of the problem and the number of structural components in the bridge, the complete finite element model is typically very complex. In past investigations, limitations in computer facilities prevented

the complex analysis of bridge structures. With the advent of faster computers though, researchers are undertaking more complex analyses of modern highway bridges. Some of the significant contributions to the modeling of the bridge superstructure and supporting pier foundation are discussed below.

Over the past decade the abilities of general purpose finite element analysis software has increased drastically. Unfortunately the increase in functionality is coupled with an increase in modeling complexity. Commercial finite element programs such as ADINA (1997), ANSYS (1996), and ABAQUS (1998) provide a complex set of element models but do not offer any guidance for creating specific structural models, such as highway bridges. As a result, the user must devote a great deal of time to creating a model, leaving very little time to study the behavior of the modeled structure. To this extent, problem-specific finite element software packages can provide a particular advantage over general purpose programs.

Several finite element programs have been developed to specifically address the analysis of highway bridges. The implementation and capabilities of different highway bridge analysis program vary significantly. The *CONSPAN LRFD* program (Leap Software 1998) provides a comprehensive analysis of simple and continuous prestressed bridge beams. This program does not formulate a complete bridge model though. A similar program, *QConBridge* (1996), analyzes a bridge using a two-dimensional implementation. A more refined analysis of bridges is provided by the *STRAP Bridge* program (1998). STRAP Bridge is an extension of the general purpose STRAP program. STRAP Bridge program provides a complete three-dimensional analysis of the bridge, including influence surfaces to model the effect of three-dimensional live loading. The

STRAP Bridge program, however, does not model all the unique structural components in a typical modern highway bridge. Perhaps the most comprehensive bridge analysis program is *BRUFEM* (Hays et al. 1995). The BRUFEM program models each structural component of the bridge superstructure to create a very realistic model of the bridge. The BRUFEM program considers the unique behavior of each structural component and implements a method to model the behavior accordingly.

There are relatively few software packages that address the design and analysis of bridge piers. The simplest of the software packages, *Analysis of Multiple Column Piers* (1976), analyzes piers as a frame using a moment distribution technique. This program is limited to bridge piers with shallow foundations and does not provide a means of analyzing a pile foundation. The *RCPier* software package (Leap Software 1994) provides more features for bridge pier analysis. The RCPier program implements many design code checks as well as a limited live load generator. Unfortunately the current version does not support the analysis of pile foundations, which are the most common type of pier foundation. The most versatile software package for the analysis and design of bridge piers is the *Florida Pier* program (Hoit et al. 1998). The Florida Pier program analyzes bridge piers including the pier structure, nonlinear piles, and nonlinear soil interaction. Florida Pier is the only program of its kind that can analyze the pile-soil interaction of pier foundation. The Florida Pier program will be implemented in this research due to its superiority in both the analysis and design of bridge piers.

1.3.2 Modeling of Live Loads

The fundamental understanding of vehicle characteristics is essential to the analysis of highway bridges. While many different types of vehicles travel across highway bridges,

commercial trucks are of primary importance. These trucks transmit the most significant live loads to the bridge structure with the greatest uncertainty. At any given time, there is some variability in the vehicle speed, weight, and path as the vehicle traverses the bridge. Also, the simultaneous application of multiple trucks in different traffic lanes can further complicate the understanding of the vehicle behavior. Recognizing the uncertainty of vehicle loading on bridges, much research has been devoted to produce accurate vehicle models for the analysis of highway bridges.

Given the variability of trucks on highway bridges, it was determined that a standard design truck load needed to be developed. Although a truck loading methodology was developed in the early 1900's, the design truck loads did not reach their current AASHTO form until 1944 (Tonias 1995). Later, in 1975, the United States federal government increased the allowable gross weight, thereby exceeding the design loads specified by AASHTO. In response, some states adopted new truck load models to accommodate the increased loading. Until 1994, AASHTO continued to specify design truck loads without modification for the increased loads. When the AASHTO-LRFD design code was introduced in 1994 though, one major loading modification was made. In the past AASHTO codes, the bridge design was determined by either by a set of concentrated truck axle loads *or* a lane load with a single centered concentrated load. The AASHTO-LRFD design code (1994) now mandates a combination of both concentrated truck axle loads *and* lane load.

While most bridge designs are governed by the combination of truck axle load and lane load, as specified in the AASHTO-LRFD design code (1994), the design must also consider a design tandem loading. In 1956, the Federal Highway Administration (FHWA)

adopted the design tandem loading, then known as *alternative military loading*. This new loading was intended to model heavy military vehicles with very close wheel spacing. Bridges designed on the U.S. Interstate highway system are required to consider the design tandem loading and the design truck loading. The loading configuration that causes the greatest stresses is then selected.

Additional vehicle load refinements have been proposed by various government agencies. The Florida Department of Transportation (FDOT) has adopted a modified wheel spacing for the width of the vehicle. This modified wheel spacing more accurately reflects the characteristics of Florida highway trucks and has been implemented in the BRUFEM program (Hays et al. 1995). This modified truck has also been applied to the present investigation. To specifically address overweight vehicles, the California Department of Transportation (CALTRANS 1993) developed a live loading configuration known as the *permit design loads*. While permit loads are not directly considered in this investigation, their impact can be significant for some bridge designs. Finally, Alaska has adopted *extralegal live loads* for design of bridges in the logging areas to support unusually heavy logging trucks. With all the variant truck load models adopted by various state government agencies, it is clear that the live loading can be very specific to geographic regions.

While much research has addressed different live load models for design of the bridge superstructure very few research efforts focus on the design of the bridge piers. The AASHTO-LRFD design code (1994) states that the piers must be designed to transmit the live loads from the bridge superstructure to the supporting ground. However, the methodology for applying the live loads to the piers in the AASHTO-LRFD code refers

back to the application of live loads for the bridge superstructure design. Essentially, the placement of live loads on bridge piers is left to engineering judgement based on the guidelines of the AASHTO-LRFD design code.

Due to the lack of information or emphasis on live loading for bridge piers very few software packages address the issue. Of the limited programs available, the *RCPier* program most effectively addresses the live loading of bridge piers. The RCPier program moves a set of axle loads (positioned above the bridge pier) across the width of the bridge to obtain the design live loads. Currently, the RCPier program is limited to live load generation for *only* one vehicle. This limitation of one vehicle load is very significant since multiple trucks frequently traverse a bridge.

1.3.3 Neural Network Applications to Structural Analysis

The applications of neural networks have grown significantly in the past decade. The neural networks that were once applied to solve obscure problems in the computer science field have now spread to many disciplines including civil engineering. In particular, the structural engineering field is now reaping the benefits of neural network applications. Some of the structural applications for neural networks are outlined below.

The first paper to influence the development of neural networks as a structural analysis tool was produced by VanLuchene and Sun (1990). This paper provides three very practical implementations for neural networks. Using basic principles, neural networks were developed to predict the moment in a simply supported beam, the strength of a reinforced concrete section, and location and magnitude of the maximum moment in a simply supported rectangular plate. Although these structural applications were simple, the implications on the structural field were very important.

Rogers (1994) utilized neural networks to obtain optimal solutions for design problems. A structural design problem typically requires numerous iterations before a final design is adopted. This work compares the time to compute numerous design iterations with the time to create, train, and run a neural network to produce the optimal result. The results indicate that while the initial network creation and training are more time intensive, much less time required for the network to determine new optimal designs when compared to the iterative design process. In other words, once the network has been trained, the optimal designs can be predicted very quickly. This is very important since a change in at least one design parameter could require a complete iterative redesign, while the neural network could produce the new optimal design in one quick iteration.

A procedure for computing truck attributes with neural networks was proposed by Gagarin et al. (1994). This paper describes the application of neural networks to the problem of determining truck attributes (such as velocity, axle spacing, and axle loads) purely from strain-response readings taken from the structure over which the truck is traveling. This paper points to the importance of reducing the neural network structure down to simple components to solve several smaller tasks. In this work, the implementation of separate neural networks provided a very fast, accurate, and convenient means of determining the truck attributes.

A comprehensive investigation of using neural networks to predict the displacement of the bridge superstructure under live loading was conducted by Consolazio (1995). This work uses a collection of neural networks to predict the displacement of a two-span bridge. The displaced shape of the bridge was then used to seed an iterative equation

solver. Among other important findings, this work suggests that many training sets may be needed to obtain an accurate result from the neural network.

Mukherjee and Deshpande (1995) discussed modeling the initial design process using neural networks. This paper proposes a neural network for the initial design of simple span reinforced concrete beams. While this design application is important, the paper is more notable for its analysis of the neural network architecture. In addition to discussing the rules and guidelines for network construction, the paper addresses the concept of *fault tolerance* in great detail. This analysis of fault tolerance indicates the effect of damaging network connections on the output of the network. This contribution is a significant compliment to the network pruning concepts discussed later.

Various other papers have contributed to the structural engineering field without introducing specific structural analysis applications. A comprehensive introduction to the use of neural networks in civil engineering is provided in a two-part paper by Flood and Kartam (1994). In addition, Yeh et al. (1993) investigated a procedure for diagnosing prestressed concrete piles using neural networks. This paper also provides an in depth analysis of network training parameters and network architecture. Finally, Hajela and Berke (1991) examined the role of neural computing strategies in structural analysis and design.

1.4 Organization of Dissertation

This dissertation is organized into eight chapters. Included in the introduction are the general description of the live load application problem, the objective, and scope of the study. An extensive literature survey follows with a review of previous works that relate

to the investigation herein. Chapter 2 contains the details for modeling the structural components of the bridge superstructure as well as the application of the live loads. Chapter 3 contains the details for the modeling of the bridge pier and the live load transfer mechanism for the bridge piers. An overview of neural networks is then presented in Chapter 4 in preparation for the specific neural network applications in the chapters that follow. Chapter 5 addresses load positioning on single column piers and Chapter 6 addresses multiple column piers. The architecture of the networks created in Chapters 5 and 6 are then optimized using a network pruning algorithm presented in Chapter 7. Finally, Chapter 8 summaries the results of the investigations and suggests some issue for possible future investigation.

CHAPTER 2
MODELING OF HIGHWAY BRIDGES

2.1 Introduction

This chapter will describe the modeling of highway bridges using the finite element method (FEM). Each structural component of the bridge is modeled using different finite element techniques. After identifying the essential components that determine the structural behavior of the bridge, the different element types are assembled to form a complete structural model of the bridge. After assembling the complete finite element model, various live load models are discussed to simulate highway vehicle loading.

2.2 Modeling of Structural Components

A correct finite element model of a highway bridge must consider all the essential components that contribute to the structural response. These structural components include the deck slab, girders, diaphragms, and support conditions. The features of each structural component and its contribution to the overall model are discussed below. This discussion does not seek to limit the number of essential components, however. Other components can be added to the overall finite element model if their influence on the structural response can be justified.

2.2.1 Modeling of Deck Slab

The deck slab of a bridge is most commonly modeled using plate or shell elements. Plate elements are typically used for non-composite modeling between the deck slab and the girders. The plate element allows out-of-plane translation and in-plane bending. Shell elements are used for composite modeling between slab and girder. The shell element combines the flexural response of a plate element with the in-plane translation of a membrane element.

The plate bending element used in this research is based on a Reissner-Mindlin formulation (Reissner 1945, Mindlin 1951). In this formulation, shear deformations are included and therefore the line originally normal to the midsurface in general does not remain normal to the midsurface after loading. Since the theory accounts for transverse shear deformations, the method is especially suited for analyzing thick slabs from highway bridge decks.

The behavior of the plate or shell elements depends on the number of nodes in the element. The number of nodes usually ranges from four to nine as shown in Figure 2.1. A rectangular plate or shell element with four nodes is termed a *bilinear* element, whereas an element with nine nodes is termed a "Lagrangian" *quadratic* element.

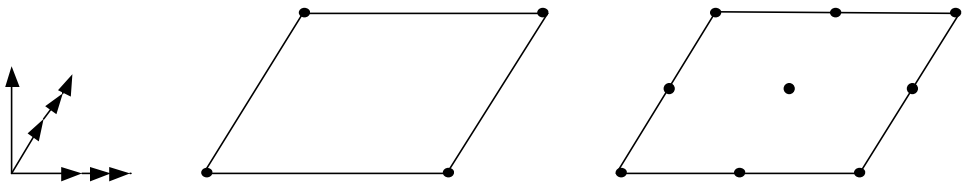


Figure 2.1 Bilinear and Quadratic Elements

The Lagrangian quadratic element is a higher order element than the bilinear element, and in general produces more accurate results. However, Lagrangian quadratic elements can suffer from zero-energy modes far more frequently than bilinear elements when used in highway bridge analysis (Consolazio 1995). The term “zero-energy mode” refers not to a rigid body motion, but to a case where displacements produce zero strain energy. This phenomena or *instability* arises from a deficiency in the element formulation process, such as the use of a low-order Gauss quadrature rule (Cook 1989). If for example, the strains in the elements are zero at the quadrature sampling points, even though strains at other points may be non-zero, an instability will occur. This type of element is said to be *rank deficient* due to the instability and can create a singular element stiffness matrix. A possible zero-energy mode (Consolazio 1995) is shown in Figure 2.2.

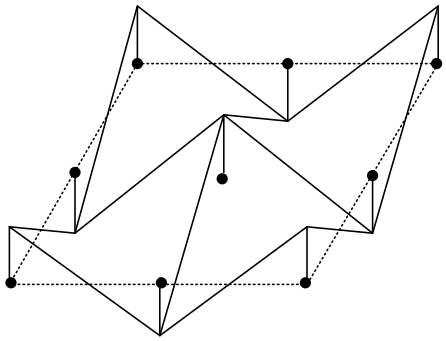


Figure 2.2 Zero-Energy Mode in Lagrangian Quadratic Element

The advantage gained in accuracy using the Lagrangian quadratic elements is diminished by the complications described above. For the purposes of this research, the shell elements are required to distribute the live load to the girders to thereby determine the reaction forces on the bearing pads. For simplicity and speed of computation, the author has decided to employ four node bilinear shell elements in the analysis of the bridge

superstructure. A nine node quadratic shell element will be introduced later in the modeling of the pier foundation. This element uses a reduced integration scheme to improve the behavior of the element.

2.2.2 Modeling of Girders

The girders of a bridge are modeled with three-dimensional Hermitian beam elements. The beam element is assumed to be of a bar of uniform cross-section capable of resisting the axial forces, bending moments, and torsional moments shown in Figure 2.3. The girder properties are based on the either standard AASHTO type prestressed concrete girders or steel plate girders, both with an assumed linear elastic behavior. For this analysis, the girders are of uniform cross-section and are of the same type across the width of the bridge. In addition, a uniform girder spacing is assumed, which is typical of highway bridge designs.

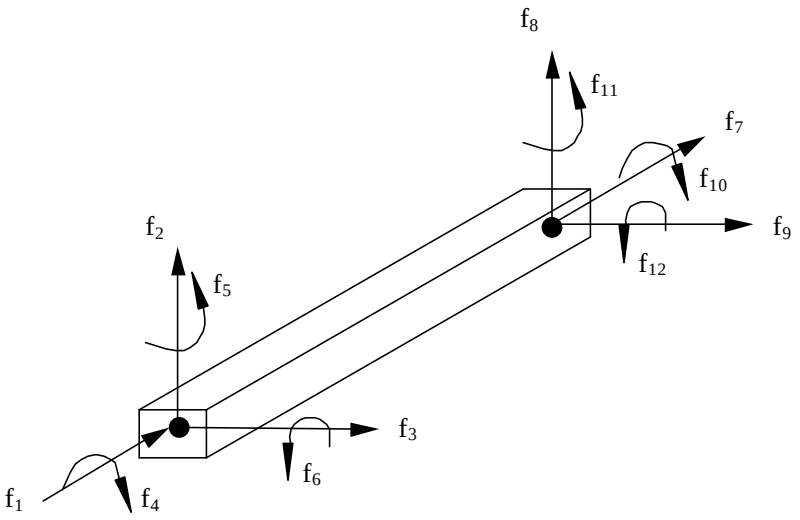


Figure 2.3 Standard Beam Element

In order to more accurately model the behavior of the girders, beam elements are placed at the center of gravity of the girder cross-section. Rigid end offsets are used to connect the eccentric beam elements of the girder to the midsurface nodes of the shell elements in the slab as shown in Figure 2.4. This rigid link construction produces the correct constraint relation for the displacements of the slab and beam (Chen 1996).

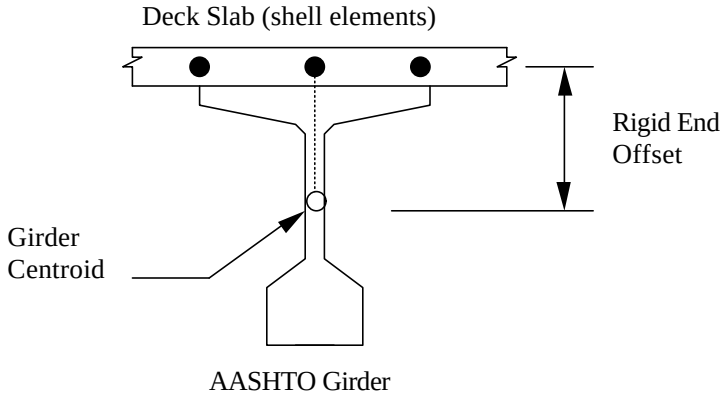


Figure 2.4 Girder Model with Rigid End Offsets

2.2.3 Modeling of Diaphragms

Diaphragms are used to provide lateral support for the girders and to distribute the vertical load on the bridge deck to the girders. The diaphragm is considered to be a non-load bearing secondary member in the bridge structure. According to the AASHTO LRFD code, diaphragms shall be provided at abutments, piers, and hinged joints. Intermediate diaphragms may be used to provide torsional resistance and to support the deck at points of discontinuity or at angle points of girders. In this investigation, diaphragms were provided at the two abutments and over the interior support using rigid offsets from the deck slab to one half the distance to the center of gravity of the girders.

2.2.4 Modeling of Pier Supports

Linear elastic spring elements are used to model the support condition at the end of the bridge spans. Vertical translational springs are attached to the end of each girder. A spring stiffness of 1000 kip/in was used for the end supports and 3000 kip/in for the interior support, which is consistent with elastic stiffness supplied by the supporting foundations (Hays et al. 1994). These spring elements allow for quick determination of the support reactions and provide an accurate modeling of the bearing pad stiffness. The reaction forces in the springs will be applied to the pier structure. The method of load transfer from the bridge to the pier foundation is outlined in Chapter 3.

2.3 Assembled Bridge Model

The different element components described above can be assembled to form a complete model of the bridge superstructure. To assemble and analyze this complete bridge model, the author has made significant modifications to an existing finite element program called *Livegen* (Lertpaitoonpan 1997). The Livegen program was developed to study live load generation on bridges in accordance with the AASHTO-LRFD design specifications. The program creates a complete bridge model consisting of a two-span bridge with interior support conditions to model the pier support. The two spans can be different though the width of the bridge must remain constant across both spans.

The different element components are assembled to accommodate the geometric configuration of the bridge. Two shell elements are assembled over the top flange of each girder as well as two shell elements between the top flanges of the girders as shown in Figure 2.5. One shell element is used to model the overhanging portion of the deck slab.

As stated before, the center of gravity of the girders are connected to the midsurface of the slab elements by rigid end offsets. There is no current limit on the number of girders.

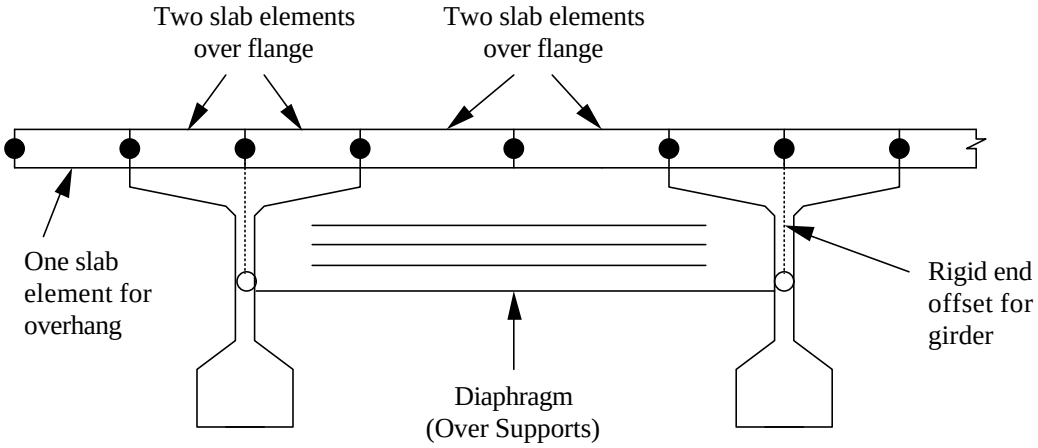


Figure 2.5 Cross-section of Slab and Girder Assembly

Some modeling guidelines are necessary to assemble the longitudinal portion of the bridge. The AASHTO LRFD design specification states that the use of nine longitudinal nodes per span for beam elements is sufficient in lieu of a more complex analysis. In addition, the design specifications limit the shell element geometry to an aspect ratio of five. To accommodate these recommendations by the AASHTO LRFD code, the Livegen program has been modified to generate ten longitudinal nodes per span, regardless of the span length. It is the author's experience that this modeling configuration is sufficient to produce accurate results over a variety of span lengths. A typical finite element model of the complete bridge superstructure is shown in Figure 2.6. Although four girders are assumed for this illustration, any number of girders can be used when generating the finite element model.

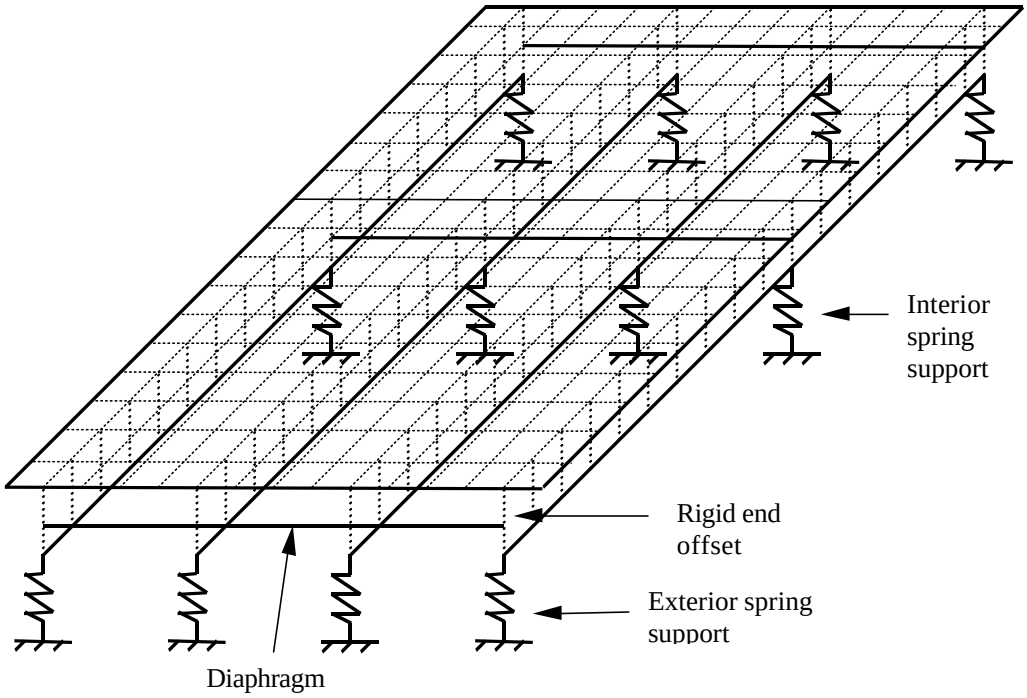


Figure 2.6 Complete Bridge Model with Four Girders

2.4 Modeling Vehicle Live Loads

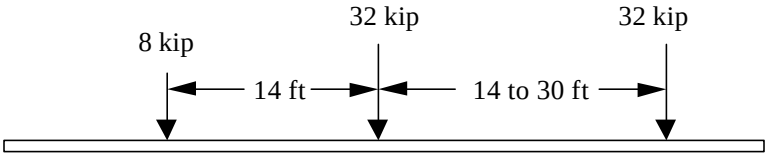
The loading of vehicles on highway bridges has been studied extensively. After much research into the field, AASHTO has established standard vehicle models for the design of highway bridges. These models are intended to simulate the actual vehicle loads from commercial truck traffic. Although AASHTO recognizes 25 different types of vehicle loads the most commonly used loading models are discussed below.

2.4.1 Design Truck

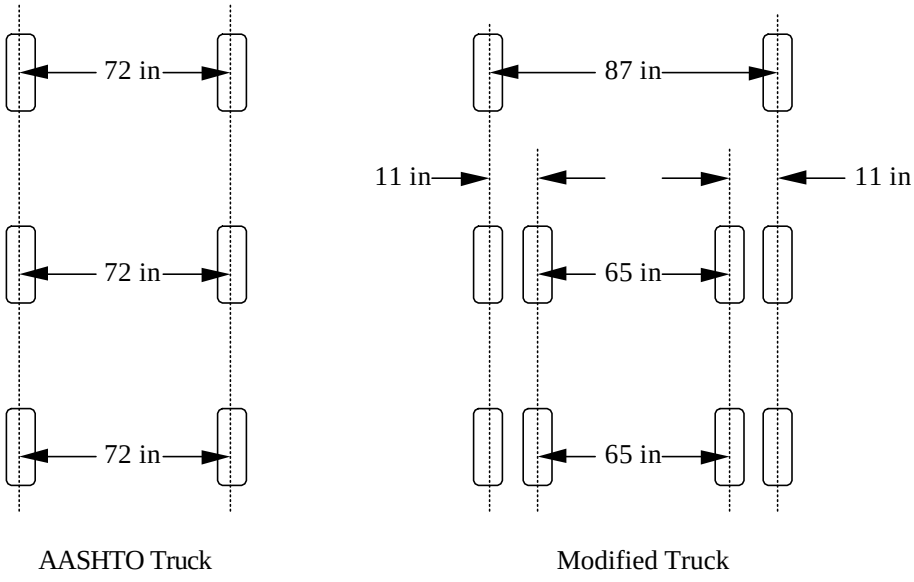
The standard design truck simulates an actual truck loading though a series of concentrated axle loads. The three axle loads are spaced at a distance consistent with most highway trucks. Except as specified in Article 3.6.1.4.1 in the AASHTO LRFD code, the

first two axles are spaced at 14 feet and the rear axle spacing can vary from 14 feet to 30 feet to produce a maximum force effect in the bridge. In addition, the design truck is has a center-to-center wheel spacing of six feet in lieu of more refined vehicle characteristics. The prescribed vehicle width is used for a transverse loading analysis.

A modified vehicle is adopted in this research based on more refined vehicle measurements (Hays et al. 1994). The modified truck model provides for a more realistic wheel spacing for the front axle and rear axles. The spacing between the axles, however, follows the AASHTO LRFD spacing recommendations. The characteristics of standard truck and the modified truck model are shown in Figure 2.7.



(a) Two-dimensional Axle Loads

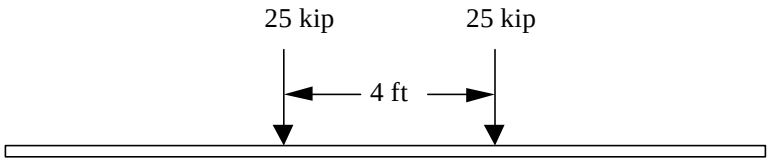


(b) Three-dimensional Wheel Loads

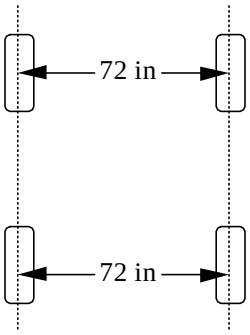
Figure 2.7 Design Truck Specifications

2.4.2 Tandem Truck

The tandem truck model simulates special vehicles with small axle spacing such as military trucks. This loading configuration will typically govern the design of very short span bridges. According to AASHTO LRFD specifications, the tandem load is composed of two 25 kip axle loads spaced four feet apart. Like the standard design truck, the center-to-center wheel spacing is six feet. The characteristics of the tandem truck are shown in Figure 2.8.



(a) Two-dimensional Axle Loads



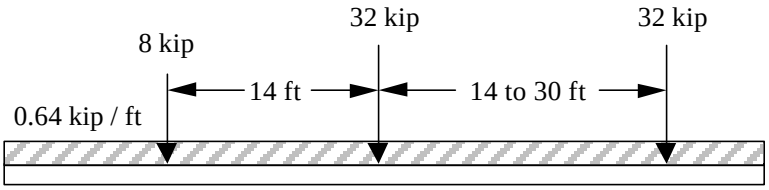
(b) Three-dimensional Wheel Loads

Figure 2.8 Tandem Truck Specifications

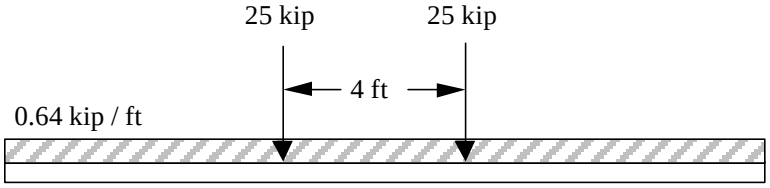
2.4.3 Lane Load

The AASHTO LRFD code requires the inclusion of an uninterrupted lane load along the entire span of the bridge. The lane load in intended to simulate a static pressure from stationary vehicles on the bridge. The lane load must be included with either the standard truck or the tandem truck. In addition, the lane load must be applied to produce

the maximum desired force effect. This combination of truck and lane load is termed “HL-93” in the AASHTO LRFD code. The two possible HL-93 model configurations are shown in Figure 2.9.



(a) Design Truck + Lane Load



(b) Tandem Truck + Lane Load

Figure 2.9 HL-93 Live Load Model

2.5 Two-dimensional Application of Live Loads

When designing the bridge girders or the slab, the design vehicle must be positioned along the span to produce the maximum desired force effect. The truck, either design or tandem, must be positioned to produce the worst force effect. Likewise, when determining the maximum reaction at the interior support for the pier design, the worst case truck model must also be used. For most pier designs, the choice of vehicle model is not obvious. The tandem truck can produce a greater force effect in shorter span bridges. The design truck will control the interior pier design for longer span bridges. Since it is unclear which vehicle model will control the design, the Livegen program calculates the interior reaction forces from both the design truck and tandem truck to determine which

configuration produces the maximum force effect on the interior support. Since the goal of this research is to determine the maximum vertical reaction forces at the interior support, several different loading schemes must be considered. An uninterrupted lane load occupying both spans is superimposed with each truck configuration discussed below.

2.5.1 Single Truck Loading

The single truck loading represents the simplest loading configuration the can be placed over the interior support. The axles of the truck are position to produce the maximum reaction in the interior support as shown in Figure 2.10. For the case of uneven span lengths, the front axle is positioned on the shorter span and the rear axle on the longer span to achieve the maximum reaction force. Regardless of the span lengths, the minimum allowable spacing of 14 feet between the rear axles will produce the maximum support reaction. The design truck load is then combined with a lane load in both spans to calculate the maximum interior pier reaction.

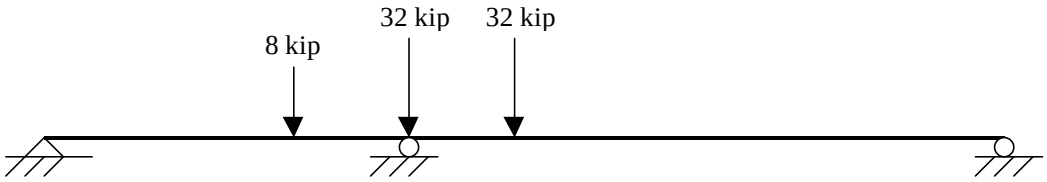


Figure 2.10 Single Truck Loading

2.5.2 Dual Truck Loading

A more complicated loading specified by the AASHTO LRFD code consists of two design trucks spaced 50 feet apart. This configuration simulates two vehicles traversing the bridge in a single lane. The axles of the trucks are positioned to produce the maximum reaction in the interior support as shown in Figure 2.11. When determining the interior

support reaction, 90% of the force effect from both vehicles is applied along with 90% of an uninterrupted lane load on both spans.

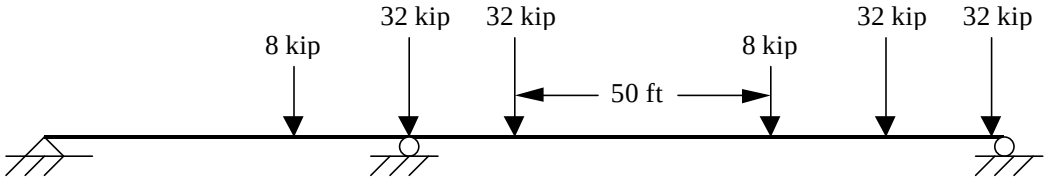


Figure 2.11 Dual Truck Loading

2.5.3 Tandem Loading

The final loading configuration positions a tandem truck over the interior support. To obtain the maximum interior support reaction, one of the two 25 kip axle loads is positioned directly above the support as shown in Figure 2.12. The tandem truck load is then combined with a lane load in both spans to calculate the maximum interior pier reaction. Unlike the design truck provisions, the AASHTO LRFD code does not specify a variable axle spacing for the tandem truck and lane load combination.

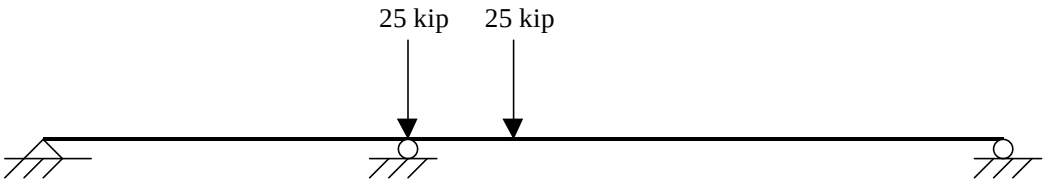


Figure 2.12 Tandem Truck Loading

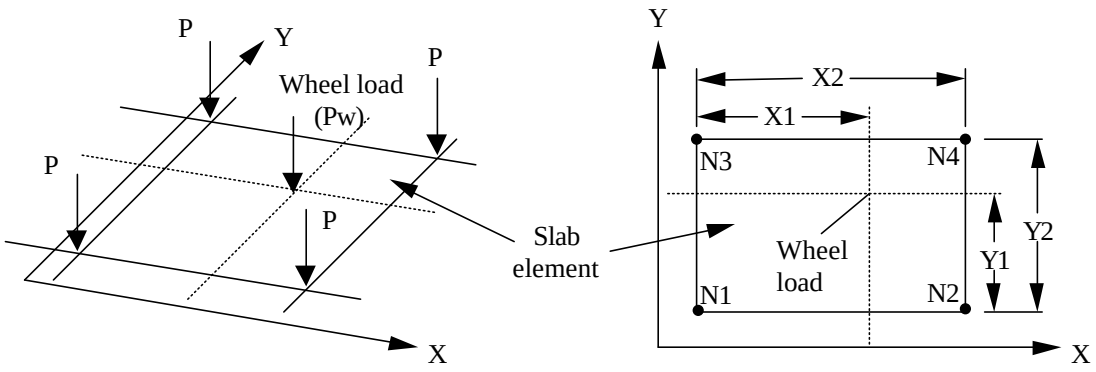
2.6 Three-dimensional Live Load Models

The two-dimensional live load models have several shortcomings that affect the validity of the load analysis. First, the two-dimensional model does not directly account for the distribution of lateral load across the width of the bridge. Instead, a live load distribution factor is applied to the load to approximate the lateral distribution in

accordance with the AASHTO LRFD design code. Second, the two-dimensional model does not account for an uneven loading across the width of the bridge. Even the simplest example where a single vehicle occupies an exterior lane of the bridge produces an unsymmetrical loading configuration. These types of unsymmetrical loading will produce different girder reactions at the interior support pier, which can not be accounted for in the two-dimensional model. Due to the inadequacy of the two-dimensional live load model, the Livegen program implements a complete three-dimensional live load model. The details of the three-dimensional live loads are outlined below.

2.6.1 Wheel Load Distribution

It is unlikely that the applied wheel loads will be positioned exactly at the nodes of the shell elements of the slab deck. For this reason the Livegen program converts the applied wheel loads to nodal loads through linear interpolation between the closest nodes. These nodal loads are statically equivalent to the applied wheel loads. The equivalent nodal loads are shown in Figure 2.13 along with static distribution factors. These static distribution factors determine the contribution of each wheel load to the adjacent nodes. If the wheel is positioned off of the bridge, then the contribution of that wheel is not included in the analysis. Likewise, if the wheel load is positioned directly above one of the nodes in the slab, then the magnitude of the wheel load is applied only to that node.



Node Number	Statically Equivalent Nodal Loads
1	P1 = Pw (1-α)(1-β)
2	P2 = Pw (α)(1-β)
3	P3 = Pw (1-α)(β)
4	P4 = Pw (α)(β)
Static Distribution Factors α = X1 / X2 β = Y1/ Y2	

Figure 2.13 Equivalent Nodal Loads

2.6.2 Lane Load Distribution

The lane load simulates the presence of many closely spaced vehicles on the bridge. The model is defined by placing a static pressure equivalent to the two-dimensional lane load specified by AASHTO LRFD code. The lane load pressure occupies a width of 10 feet within the prescribed 12 foot design lane width. The distribution of the pressure to the nodes closest to the pressure application in the shell elements is slightly more difficult than the distribution of wheel loads. The pressure is first multiplied with the length of the shell elements in the longitudinal direction (along the span) to get a load per element. This equivalent load is then converted to concentrated loads (W1 to W4 in Figure 2.14) by first lumping half of the load five feet from the center of application in each direction along the transverse axis. Finally, each equivalent

concentrated load is then treated like a wheel load and then interpolated to the closest nodes. The lane load transformation process in illustrated in Figure 2-14.

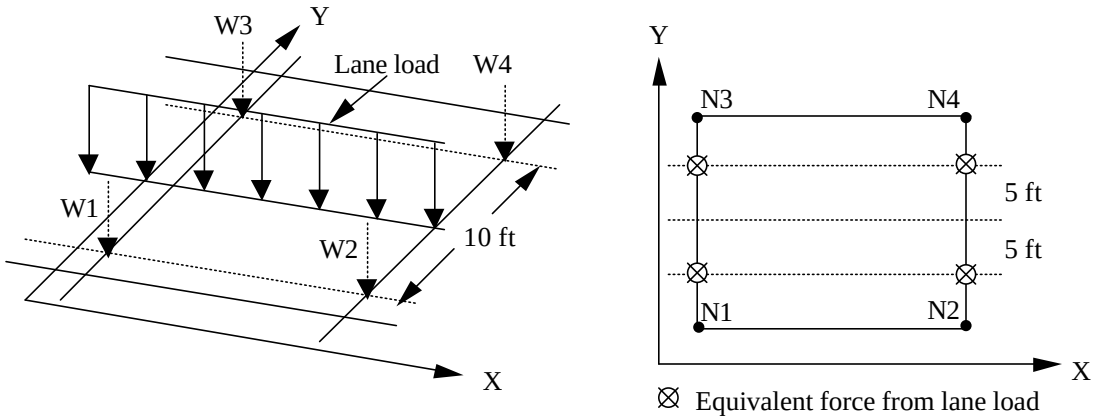


Figure 2.14 Equivalent Lane Load

2.6.3 Live Load Reduction

The extreme live load force effect shall be determined from a combination of vehicles on the bridge. As the number of loaded design lanes increases, it becomes more unlikely that all lanes will be loaded simultaneously. With this consideration, the AASHTO LRFD code allows for a reduction in live load through the use of a load reduction multiplier. This multiplier, termed the *multiple presence factor*, is applied to all loaded lanes. The values for the multiple presence factors are shown in Table 2.1

Table 2.1 Multiple Presence Factors.

Number of Loaded Lanes	Multiple Presence Factor
1	1.20
2	1.00
3	0.85
>3	0.65

2.7 Live Load Generation

The three-dimensional live load model adds the ability to study the effect of loading multiple design lanes. One of the stated advantages of the three-dimensional model was the capability of determining a more accurate distribution of load at the interior pier support. Since the primary goal of this effort is to determine the maximum reactions at this support, it is essential to study all possible loading configurations in the three dimensions. By increasing the dimensionality of the problem though, the number of possible loading configurations drastically increases. While some intuition can be employed to reduce the complexity of the problem, a large number of load position combinations still exists.

Following the conclusions of the two-dimensional analysis, it is apparent one of the vehicle axles must be placed directly above the interior support to achieve the maximum reaction force. For the design truck, the remaining two axles should be spaced at the minimum specified distance of 14 feet from the center axle. The front axle of the design truck should always be placed on the shorter of the two spans. For the tandem truck, the remaining axle should be spaced four feet from the axle positioned over the interior support. The remaining tandem truck axle should be placed on the longer of the two spans. In either case, if the two spans are equal, the remaining axles can be placed in either span. With the longitudinal placement of the wheel loads now known, the configuration of the design lane must then be considered before proceeding with the live load generation procedure.

2.7.1 Design Lane Configuration

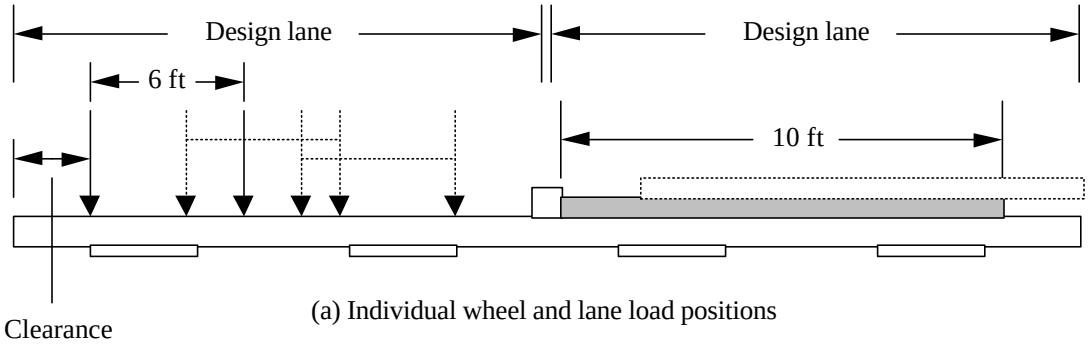
The AASHTO LRFD design code has established guidelines for determining the placement of the design lanes across the bridge. The number of design lanes is determined not by the final number of traffic lanes in use on the bridge, but rather by the number of possible design lanes that will fit across the width of the bridge. This insistence on using the maximum number of possible design lanes serves a major purpose. Since bridges are costly structures, a design considering the maximum number of possible design lanes will accommodate any future changes in the traffic lane configuration. According the AASHTO LRFD design code, the number of design lanes should be determined by taking the integer part of the ratio of clear roadway width divided by 12.0 (feet). The clear roadway with is defined as the distance between the left and right curbs and/or barriers. In cases where the design lanes are less than 12 feet wide, the number of design lanes shall be equal to the number of traffic lanes. For this case, the width of the design lane is taken as the width of the traffic lane. This situation, although possible for narrow bridges, was not considered in this investigation.

Once the number of design lanes has been determined, the placement of the wheel loads and lane loads can be addressed. According to the AASHTO LRFD design code, the design truck or tandem shall be positioned transversely such that the center of any wheel load is not closer than two feet from the edge of the design lane. The placement of the lane load is not restricted as long as it occupies a width of ten feet within the 12 foot design lane. It is important to emphasize that although the wheel loads and lane load simultaneously occupy the design lane they should be independently positioned within the design lane to achieve the maximum desired force effect.

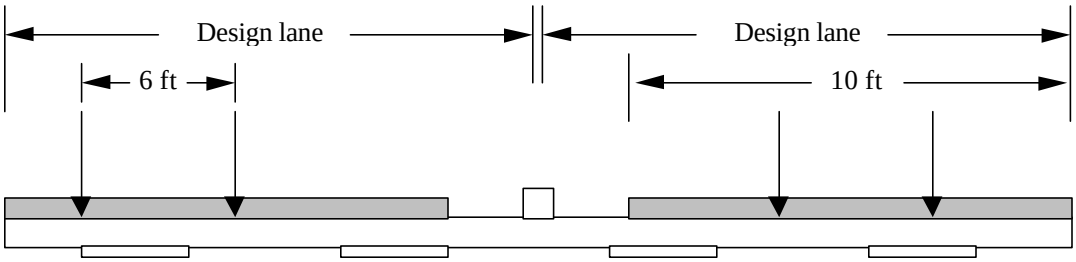
2.7.2 Transverse Live Load Placement

At any given time, the number of vehicles traversing the bridge is not known. The vehicles are free to travel any path within the specified traffic lanes. In order to determine all the possible interior pier reaction forces, all combinations of transverse vehicle loading positions must be considered. This investigation can be automated by placing vehicles in the leftmost lane positions and shifting the vehicle position to the right by a given increment. Based on an investigation by the author, the optimum vehicle shifting increment was determined to be two feet. Smaller shifting increments were very computationally demanding with no significant increase in positioning accuracy. In contrast, larger shifting increments were too coarse and failed to identify the some of the critical load positions.

Since all combinations must be investigated, each truck and lane position in each lane must first be moved individually as shown for a two lane bridge in Figure 2.15 (a). After any individual vehicle placement, all possible combinations of load positions for multiply loaded lanes must be considered as shown in Figure 2.15 (b). It is important to note that as the number of design lanes increases, the number of combinations of load positions increases rapidly. For the typical two lane bridges studied in this investigation, the number of load combinations was usually less than 50. For three lane bridges, the number of load combinations was typically less than 400. As the investigation moved to four lane bridges, the number of load combinations quickly increased to more than 2000 for a bridge using a vehicle shifting increment of two feet. Although some of the load combinations are mirror images, it is difficult to identify which combinations to eliminate to increase the efficiency of the live load generation.



(a) Individual wheel and lane load positions



(b) Combinations of wheel and lane

Figure 2.15 Transverse Live Load Positions

2.7.3 Live Load Combinations

The three-dimensional live load generation can be very time and resource consuming. When considering the simultaneous loading of multiple design lanes of the bridges and the variability of each vehicle position in each lane, it is not uncommon to have thousands of load combinations. While most of these combinations will not produce the maximum force effects in the interior pier support, it is difficult to establish the critical load cases. For this reason, intelligent methods are necessary to identify the worst load cases. In Chapters 5 and 6, intelligent techniques will be utilized to construct artificial neural networks to predict which live load combinations will produce the desired extreme force effects in the interior pier support. These neural networks will circumvent the

tedious live load generation process by predicting the four worst load combinations rather than generating the thousands of possible load combinations.

CHAPTER 3
MODELING OF BRIDGE PIERS

3.1 Introduction

This chapter will describe the modeling of bridge pier foundations using finite element techniques. Ordinarily, the pier foundations transfer all externally applied loads on the bridge superstructure to the ground. The piers can either support the end of a bridge span as an *abutment*, or two adjacent spans as an *interior pier* support. While both types of pier foundations are integral to the bridge behavior, only the interior bridge piers are addressed in this work. Since the interior piers support two adjacent spans, the interior piers must receive and transfers the greatest portion of the applied live loads to the supporting soil.

3.2 Modeling of Bridge Pier Components

Since the pier foundations are partially embedded in soil, the behavior can be very complex due the nonlinear material properties exhibited by the soil. There have been many attempts to model the correct behavior of pier foundations embedded in soil. Some general purpose finite element programs require the construction of a complex and computationally demanding three-dimensional finite element model. Such models rely on constitutive equations for elasticity and/or plasticity even when there are few known parameters to describe the behavior of the soil. In contrast, other programs provide highly simplified

elastic soil spring models to approximate the soil behavior. These simplified programs do not account for the nonlinear behavior of the soil and to this extent are limited in their analytical capabilities. Fortunately, there is a suitable compromise between computational accuracy and efficiency provided by the *Florida Pier* program (Hoit et al. 1998). The Florida Pier program addresses the unique behavior of the pier foundation by employing special finite element techniques to model soil behavior. The element formulations were developed specifically to model the nonlinear behavior of the soil as well as the material and geometric nonlinearities in the structural members. The characteristics and formulation of each of the elements used in the modeling of the pier foundations with Florida Pier are explained below.

3.2.1 Modeling of Piles and Drilled Shafts

Structural components that are embedded in the ground must be capable of conforming to the nonlinear behavior of the soil under any type of loading condition. Soil, unlike many structural materials, can experience nonlinear behavior under routine service loading. For this reason, the piles and drilled shafts (now referred to as piles solely for simplicity) must be capable of deforming to satisfy the static equilibrium of the pile-soil system.

In this investigation, each pile was modeled using 16 two-node, three-dimensional beam elements. The elements can exhibit linear material behavior or nonlinear material and geometric behavior using the discrete-element formulation described below. Both elements are capable of modeling bending in both planes, torsion, and axial deformations. The first element in each pile connects the pile to the pile cap and represents the freestanding length of the pile (above the ground). In order to obtain an accurate solution

for the nonlinear behavior of the freestanding portion of the pile, the element is divided into five elements and then statically condensed back to the two-node beam formulation. The remaining 15 beam elements beneath the ground surface are equally spaced along the length of the pile. If the bottom of the pile cap is embedded in the ground, then all the pile elements are formulated with equal lengths. In addition, piles can be battered (sloped) in the x- and y- directions.

The Florida Pier program provides two formulations for analyzing the nonlinear soil-pile behavior. The first formulation uses a standard three-dimensional beam element with nonlinear soil springs connected to each node to model the soil-pile behavior. Under this formulation, the beam remains in the linear elastic region while the soil is subjected to nonlinear deformations. The second, more realistic formulation utilizes a discrete- element formulation (Mitchell 1973) along with nonlinear soil springs to model the pile-soil interaction. Unlike the standard beam element formulation, the discrete-element formulation models the nonlinear material and geometric behavior of the piles. A brief overview of the discrete-element formulation and its pertinence to the present research is provided below.

The discrete-element model is used in the analysis of nonlinear pile behavior. The nonlinear behavior is modeled using the existing stress-strain curve data for the material that is integrated over the cross-section of the piles. The nonlinear geometric behavior is modeled using P-Δ moments (moments of the axial force times the displacement of one end of the element to another) on the discrete element (Hoit et al. 1996). In addition, P-y moments (moments of axial force times internal displacements within the members due to bending) is also modeled.

The discrete-element model essentially consists of very simple rigid link sections connected by nonlinear springs (Hoit et al. 1996). The element formulation is illustrated by angle and displacement relationships shown in Figure 3.1. The center block of the element can twist and extend to provide torsional and axial effects. The center block is connected by a universal joint to a rigid end block. The universal joints allow bending at the quarter pointers of the element. This configuration permits discrete angle changes Ψ_1, Ψ_2, Ψ_3, and Ψ_4, which correspond to the four moments M_1, M_2, M_3, and M_4. The discrete-element permits a discrete axial shortening δ, which corresponds to the axial thrust T. Finally, the element permits a twisting angle Ψ_5, which corresponds to the torsional moment M_5 in the center of the bar.

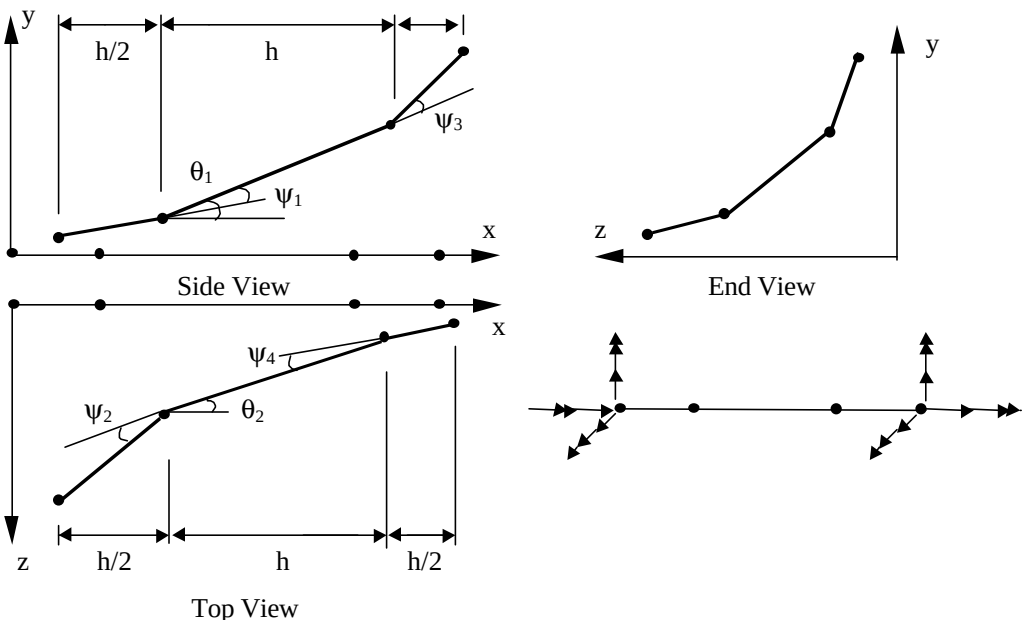


Figure 3.1 Angle and Displacement Relations for Discrete Element

In the discrete-element formulation, the discrete angle changes are used to determine the curvatures in the displaced condition. These curvatures are then used to

determine the linear strain variation in the cross-section. The strains at each point in the cross-section are then matched with a corresponding value from the material stress-strain curve. Finally, the stresses in the cross-section are integrated to check element equilibrium with the soil-pile system. If static equilibrium is not achieved at each pile node, further iterations are performed using the out-of-balance force until an equilibrium convergence tolerance is reached. Florida Pier uses the value of the largest out-of-balance force as the measure of convergence of the analysis (Hoit et al. 1996). Since the nonlinear solution proceeds with secant stiffness iteration, rather than tangent stiffness, several iterations are typically required to reach equilibrium.

3.2.2 Modeling of Soil

The soil must ultimately provide a bearing mechanism to support the pier foundation. This bearing support is typically provided through the tip bearing at the end of the piles and skin friction along the surface of the piles. When the externally applied loads from the bridge superstructure are transferred to the piles in the pier foundation, the soil deforms in response to the loading. Unlike many structural components though, the soil material behavior is highly nonlinear. This nonlinear behavior leads to a decrease in the soil stiffness with increasing deformations. For most soils, the soil stiffness continues to decrease as the load is applied. As a result, an iterative solution process is needed to achieve equilibrium of the pile-soil system. Currently, Florida Pier uses a secant stiffness iteration technique, so several iterations may be necessary to achieve static equilibrium in the piles.

The soil modeling used by Florida Pier includes axial and lateral effects of pile displacements by attaching nonlinear springs to the pile nodes. The soil springs are assigned a nonlinear behavior through the use of load-displacement curves for different types of soil. The axial behavior of the soil is modeled using T-Z curves. The lateral soil behavior is modeled using various P-Y curves (Hoit et al. 1996). The load displacement curves are established from load tests on single piles. The general shape of the curves is shown in Figure 3.2. Equations for the curves based on the test results are readily available and implemented in Florida Pier. For multiple-pile behavior, P-Y multipliers can be applied to reduce the strength of the soil due to pile group action. This strength reduction is necessary for closely spaced piles that mobilize a block of soil during deformation. In addition to the axial and lateral soil models, torsional effects can be included using a hyperbolic or user-defined soil torsional model.

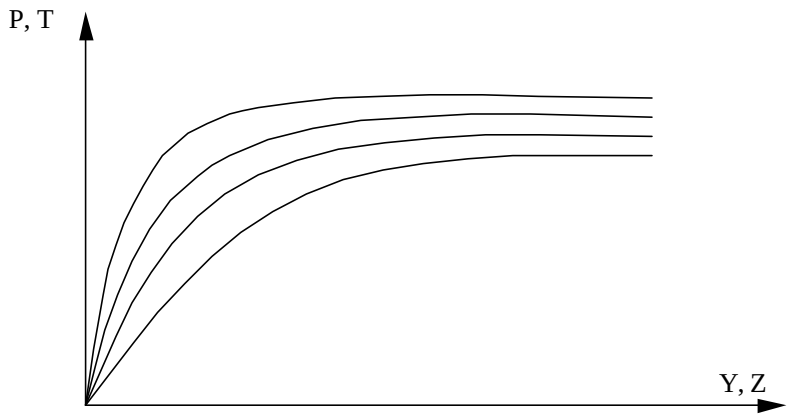


Figure 3.2 Typical P-Y and T-Z Soil Curves

As mentioned above, the nonlinear soil springs are attached to each node of the pile elements that are embedded in the ground. Soil springs are not applied to the first pile node since it is embedded in the pile cap. A typical lateral soil reaction and the

configuration for the pile-soil model is shown in Figure 3.3. In addition to the soil springs, an extra tip spring stiffness can be placed at the end of the pile to simulate pile tip bearing. For pier foundation models without soil, the tip of the pile can be restrained from motion in all directions to maintain global stability.

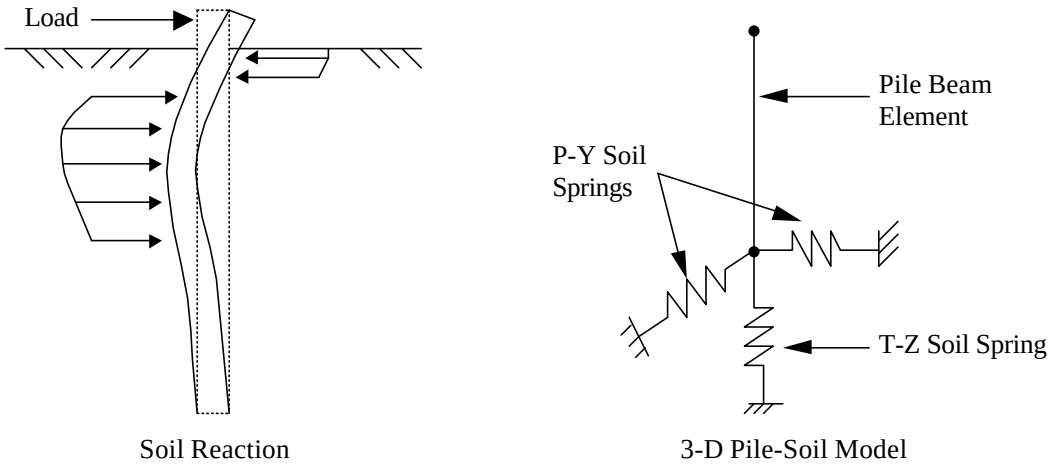


Figure 3.3 Pile-Soil Model

3.2.3 Modeling of Pile Cap

The primary purpose of the pile cap is to provide a rigid connection between the pile foundation and the pier columns. The pile caps are typically very thick, allowing for a sufficient embedment depth for the piles. Equally important, the thick cap provides an effective mechanism for resisting high shear forces and stresses in the transfer of load from the pier columns to the piles. Due to the thickness and massive size of the pile cap, the Florida Pier program assumes that the concrete cap elements will remain in the linear elastic region during loading. This assumption should not be viewed as a limitation of the program, but rather an efficient implementation of the shell element for modeling the pier cap behavior.

The element formulation for the pile cap is based on a Reissner-Mindlin plate bending formulation (Reissner 1945, Mindlin 1951). In this formulation, shear deformations are included and therefore the line originally normal to the midsurface in general does not remain normal to the midsurface after loading. Since the theory accounts for transverse shear deformations, the method is especially suited for analyzing the thick pile cap in the pier foundation. In addition to out-of-plane bending and rotation effects, the shell element used to model the pile cap provides for an in-plane membrane behavior.

The pile cap is modeled using nine-node shell elements. The pile elements are attached to any of the four corner nodes of the element. The remaining interior nodes at the midpoints and center of the elements are not attached to pile elements as shown in Figure 3.4. This configuration allows for more accurate determination of the stresses by providing sufficient flexibility between the adjacent piles. To account for the torsional rigidity of the shell elements, an equivalent beam analogy is adopted. This beam formulation applies an equivalent beam torsional stiffness to each of the corner nodes of the shell elements to transfer any torsional effects from the pile elements to the pile cap. In addition to the torsional effects, the pile connections with the pile cap can have either pinned or fixed-head connections to model the proper flexural behavior.

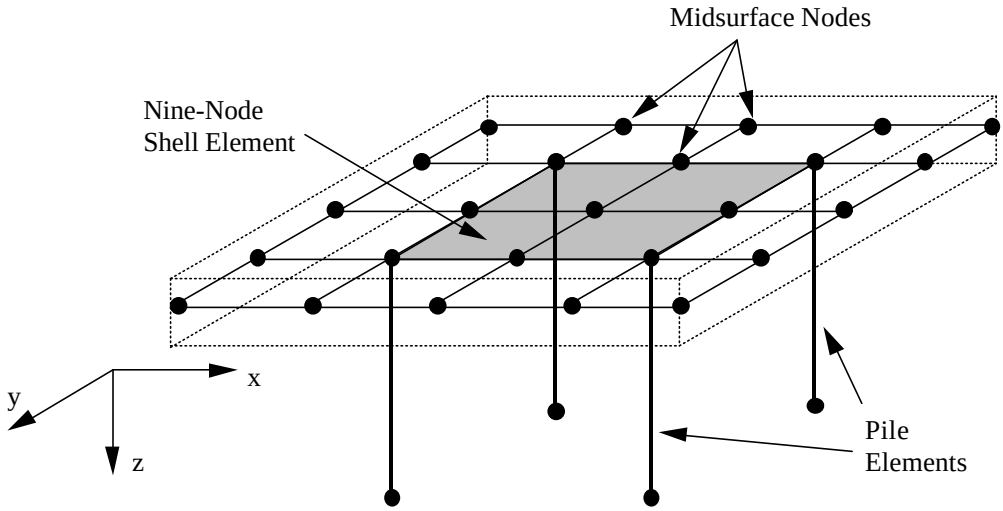


Figure 3.4 Shell and Pile Element Connection

As reported for slab elements in Chapter 2, the nine-node shell element is susceptible to numerical instabilities. If for example, the element employs a low-order Gauss integration scheme, zero-energy modes can develop due to zero strains at the sampling points. Also, locking of Mindlin plate elements can be caused by too many transverse shear constraints (Cook et al. 1989). To avoid these numerical errors, Florida Pier uses an eight-point reduced integration scheme that properly accounts for shear deformations and prevents shear locking.

3.2.4 Modeling of Pier Columns

The pier columns provide a linkage between the pile cap and the pier cap beams that support the bridge superstructure. The columns are typically subjected to high axial forces as well as bending moments due to live loading. The columns must transfer any

externally applied live load to the pile cap. This load is subsequently distributed to the pile foundation.

The columns are modeled using two-node, three-dimensional beam elements or discrete-elements. The standard beam elements are used for an assumed linear elastic pier behavior. The discrete-elements are employed to model both nonlinear material and geometric behavior of the structure. The discrete-element formulation is identical to the formulation previously discussed for nonlinear pile elements. As with the pile behavior, this nonlinear element permits P-Δ effects, which amplify the force effects in the pier structure. Regardless of the element selection, the element connection to the pile cap and pier cap remains the same. The beam or discrete-elements are attached to the midsurface nodes of the pile cap. Since the columns are typically large in diameter or width, special connector elements attach the base of the column to the four closest nodes as shown in Figure 3.5. These connector elements do not correspond to any structural components, but serve to distribute the column axial load over an area consistent with the size of the column. These connector elements also stiffen the cap at the column connect. The top of the column connects to the pier cap beams with a standard nodal connection.

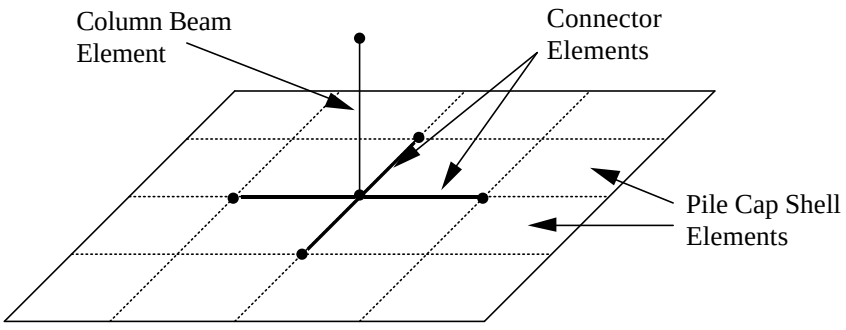


Figure 3.5 Connector Elements at Base of Column

3.2.5 Modeling of Pier Cap

The pier cap transfers the reactions from the bridge girders to the one or more pier columns in the foundation. The pier cap must be capable of resisting shear and bending effects due to the large magnitude of the concentrated loads from the girder reactions. The pier cap is most commonly modeled with beam elements to represent the beam behavior in flexure. To accommodate most pier designs, Florida Pier permits tapered beam sections in the cantilever sections as well as variable cross-sections for different portions of the pier cap. The properties for nonprismatic members are determined by linear interpolation and are kept constant across each individual element.

In Florida Pier, the pier cap is modeled using two-node, three-dimensional beam elements or discrete-elements. Again, the standard beam elements are used for an assumed linear elastic pier behavior. The discrete-elements are employed to model both nonlinear material and geometric behavior of the structure. The beam or discrete-elements are attached to the pier column beam elements at the top of the columns using a standard nodal connection as illustrated in Figure 3.6.

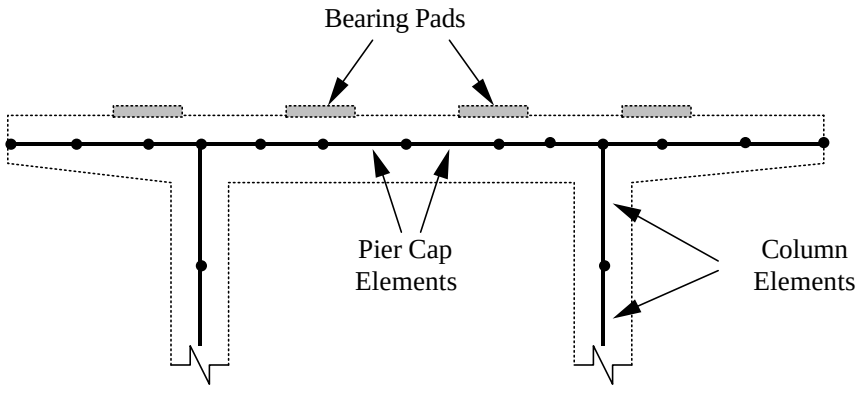


Figure 3.6 Column to Pier Cap Connection

3.2.6 Modeling of Bearing Pads

The concentrated reaction forces from the girders of the bridge superstructure are applied at the bearing pad locations. These bearing locations can be uniformly spaced or individually spaced depending on the spacing of the girders. Since Florida Pier generates the locations of the nodes in the pier cap based on the spacing of the girders, a node is placed in the exact location of each bearing pad. This placement of the nodes provides for the correct transfer of force from the girders to the pier foundation. The remaining nodes in the cap are evenly distributed between the next bearing location or pier column. Although there is some stiffness corresponding to the elastomeric bearing pads, Florida Pier does not add a value for the pad stiffness to the bearing pad nodes. Under this assumption, the bearing pads are not treated as structural components in the pier, but rather are considered to merely be a load transfer mechanism.

3.3 Complete Pier Model

All of the structural components described in the previous sections can be assembled to form a complete bridge pier model. This procedure is automated using Florida Pier preprocessing interface. Based on geometric parameters describing a pier, the program generates the nodes and elements connectivity for each of the components of the structure. The finite element model is assembled using a default number of elements for each pier component. This procedure does not, however, limit the model creation since the user still maintains the ability to specify any number of nodes in the pier to achieve more accuracy. The program also permits the use of extra flexure members and springs to

increase the stiffness of the pier model. A typical finite element model generated with Florida Pier is shown in Figure 3.7.

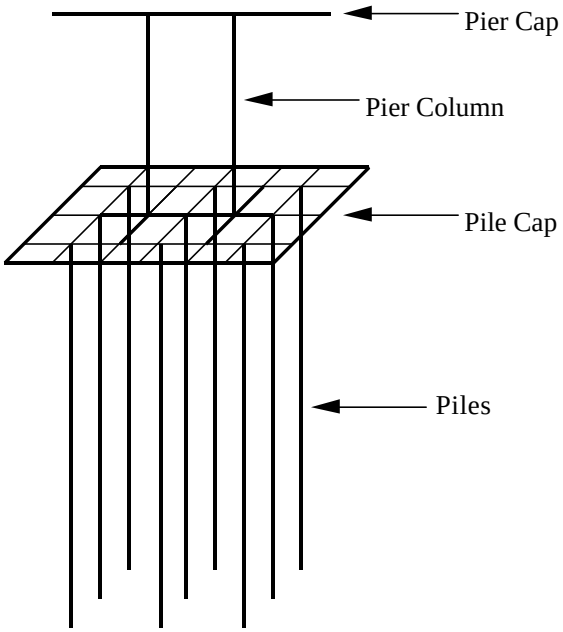


Figure 3.7 Typical Finite Element Model of Pier

3.4 Maximum Force Effects

When analyzing bridge piers, some criteria must be established to determine the maximum force effects in the different pier components. While most structural components are not expected to fail under service live loads, the location of the maximum force effects in the pier can be very useful for design. The Florida Pier program uses two main criteria for determining the force effects in the structural components. The two different methods are implemented depending on the whether the material properties are known for the cross-section. For pier components modeled using the discrete-element formulation or linear beam element with cross-sectional properties, a failure ratio can be determined. Otherwise, for components modeled with linear beam elements without full cross-section

properties, a square-root-of-the-sum-of-squares (SRSS) combination of section forces and moments can be used to establish the adequacy of the structural component. The linear beam formulation without cross-sectional properties is primarily used for quick computation during the preliminary design and analysis of the pier foundation.

3.4.1 Failure Ratio

The nominal section capacity must be calculated in order to determine whether a structural component can sustain a given loading. The relationship between applied axial force and bending moments is most often displayed in terms of an interaction diagram for the section. The simple two-dimensional interaction diagram in Figure 3.8 shows the maximum combination of factored axial load and a uniaxial bending moment that a section can sustain without failure. This load-moment relationship can be shown by a curve representing the failure envelope for all combinations of load and moment. In the two-dimensional case, any combination of axial load and uniaxial moment inside the failure line is considered safe, meaning that the structural component can sustain the applied loading. A more complex depiction of the biaxial load-moment interaction can be illustrated with the three-dimensional failure surface shown in Figure 3.8. In the three-dimensional case, any combination of axial load and biaxial bending moment inside the failure surface is considered safe. In either case, a combination of axial load and moment outside of the failure envelope requires a larger section capacity or the structural component will fail under the given loading.

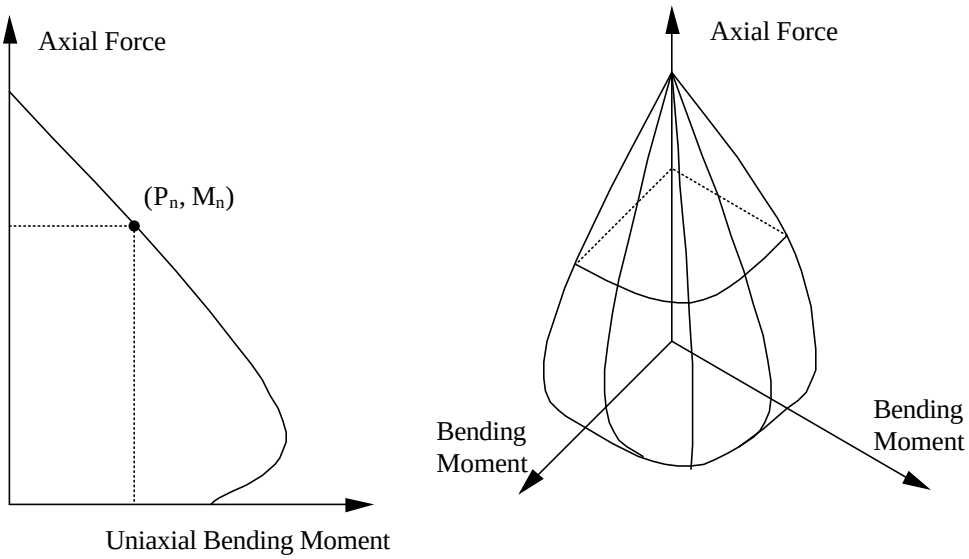


Figure 3.8 Interaction Diagram and Failure Surface

Florida Pier uses the geometric and material properties of the cross-section to compute the section capacity. The interaction surface can be computed for concrete, steel or composite concrete and steel sections. The interaction routine operates by computing numerous slices of the interaction surface for a nominal axial force (P_n) and nominal bending moments (M_{nx} and M_{ny}). From this procedure, a series of M_{nx} and M_{ny} interaction curves for various magnitudes P_n can be calculated. These interaction curves can be represented by Equation 3.1

$$\left(\frac{M_{nx}}{M_{ox}}\right)^{a} + \left(\frac{M_{ny}}{M_{oy}}\right)^{b} = 1 \qquad \textbf{(3.1)}$$

The moments M_{ox} and M_{oy} represent the nominal moment strength at axial load (P_n) for uniaxial bending about the x and y axes, respectively. The exponents, α and β, can be

determined by a least squares method so that Equation 3.1 fits the computed interaction curve.

The adequacy of the structural component is determined in terms of a failure ratio. The failure ratio is defined as the ratio of the length of the vector for the actual forces (in 3-D space – M_x, M_y, and P) divided by the length of the vector with the same direction as the actual results but with that vector touching the three-dimensional failure surface (Hoit et al. 1998). This concept of vector extension is illustrated in Figure 3.9. Using this methodology, a failure ratio greater than one corresponds to the failure of the structural component, while a failure ratio less than one corresponds to a safe load combination. For clarity, Figure 3.9 shows a safe load combination since the force combination vector does not extend all the way to the failure surface. The failure ratio has the effect of assuming that both of the moments and axial load will increase proportionally until failure (Hoit et al. 1998).

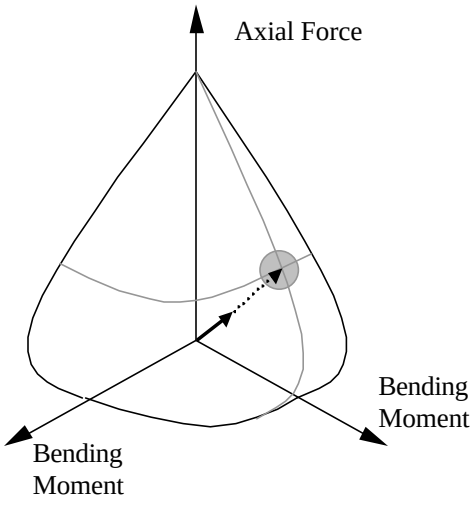


Figure 3.9 Illustration of Failure Ratio

3.4.2 SRSS Criteria

When the section properties are not known, a simplified method can be applied to determine the force effects in the structural component. The square-root-of-the-sum-of-the-squares (SRSS) method determines the force effect based on the most significant forces acting on the section from the application of vehicular live loads. For pile and pier column sections, the most significant forces acting on the section will be the axial force and the two principal bending moments. For the pier cap sections, the significant forces acting on the section are the principal bending moments and the vertical shear force. Unlike the piles and pier columns, the axial force in the pier cap can be neglected since the applied live loads induce very small axial deformations. For both cases shown in Figure 3.10, the significant forces are individually squared, then added. The square root is then taken of summation of squared force quantities to calculate the SRSS value for the section under the given loading. This SRSS value can be compared to other sections to determine the location of the maximum force effects in the different pier components.

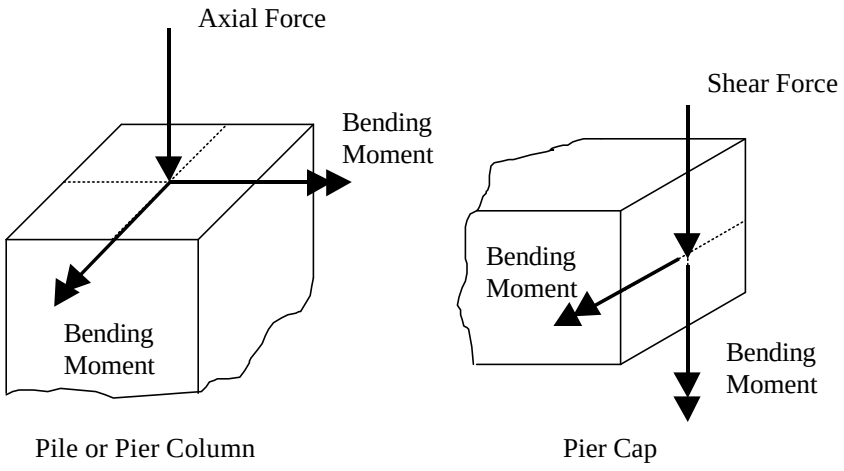


Figure 3.10 Significant Forces on Piles, Pier Columns, and Pier Cap

3.4.3 Maximum Force Effects in Pier Components

Some criteria for determining the maximum force effects in the pier must be established before proceeding with any live loads analysis. For the pile and pier columns, the maximum force effect will be determined by the maximum value of the failure ratio or SRSS. If the cross-sectional properties are specified in the Florida Pier model, a failure ratio will be used otherwise the SRSS value will be used. The pier cap is unique since the two most important force effects, shear and moment, will not occur at the same location. For this reason, two maximum force effects are considered for the pier cap, one for the maximum moment in the cap and the other for the maximum shear. In total, four maximum force effects are considered. Since it is unlikely that one live load application will produce all four maximum force effects, the four worst load cases corresponding to the four maximum force effects are considered in this investigation. It is possible, though for one load case to produce one of more of the four maximum force effects.

3.5 Load Live Transfer to Piers

In order to determine the maximum force effects in the different components of the bridge pier, all possible live load scenarios must be considered. In Chapter 2, a procedure was described for generating all possible combinations of three-dimensional live loads on the bridge superstructure. Currently, the finite element models for the bridge superstructure and pier structure are not coupled because they were developed independently with different software packages. Therefore the applied live loads on the superstructure must be transferred to the pier via the bearing pads before the pier analysis can proceed.

3.5.1 Load Transfer to Bearing Pads

To evaluate all possible load combinations, each load combination from the live load generation is considered as a separate load case in the Florida Pier program. For each load case, the loads transferred from the bridge superstructure will appear as concentrated loads on the pier bearing pads as illustrated for the single column pier in Figure 3.11. Each set of concentrated loads must be analyzed separately by the Florida Pier program. For large piers with four design lanes, this can result in several thousand load cases. Only after evaluating all load cases applied to the pier model can the maximum force effects in the different pier components be determined. The four load cases that produce the four maximum force effects are considered to be the worst live load cases for the pier.

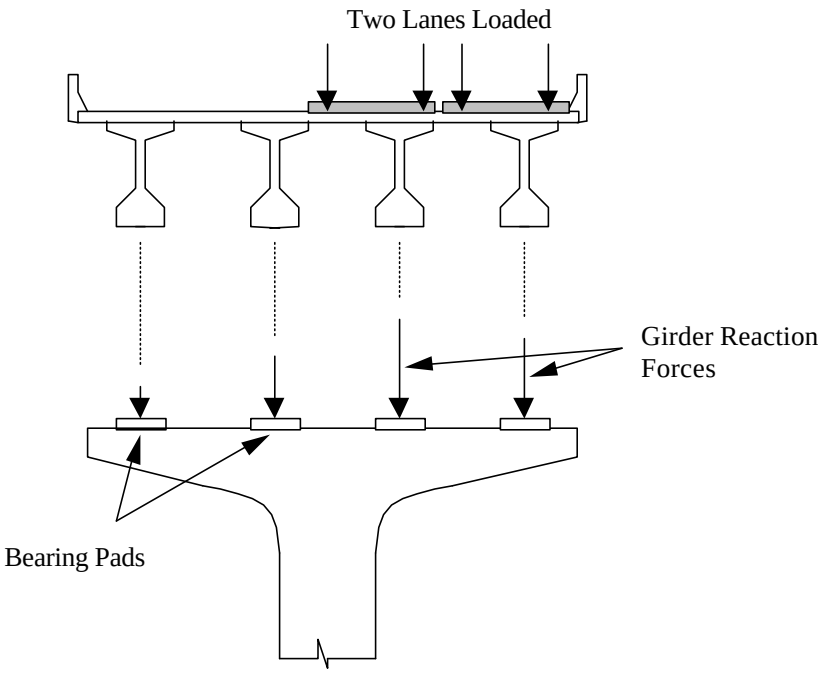


Figure 3.11 Live Load Transfer to Piers

3.5.2 Case Studies in Live Load Generation

A brief case study is now presented to justify the need for a three-dimensional live load analysis to determine the girder reactions on a simple interior pier. This case study compares the determination of girder reactions using a two-dimensional and three-dimensional live load model. For the two dimensional model, a lane load is applied as a line load across both spans along with concentrated truck axle loads over the interior support in accordance with methods outlined in Chapter 2. The reaction at the interior support determined from the two-dimensional analysis is then applied to each bearing pad on the pier in this study. For the three-dimensional model, the lane load was modeled using an equivalent lane pressure and the concentrated truck wheel loads were positioned laterally over the interior support. The individual girder reactions determined from the three-dimensional analysis are then applied to each corresponding bearing pad. It is important to understand that the three-dimensional model considers unique bearing pad reactions while two-dimensional model assumes a uniform reaction for all bearing pads.

The results of the two-dimensional and three-dimensional live load analysis are very interesting. The two-dimensional case considered only one possible load position. The three-dimensional case considered seven possible load positions. After analyzing all seven load cases on the simple pier, the force effects in the different pier components were compared to the force effects produced from the two-dimensional load case. The results in Figure 3.12 show that the three-dimensional load cases produced greater extreme force effects than the two-dimensional load case.

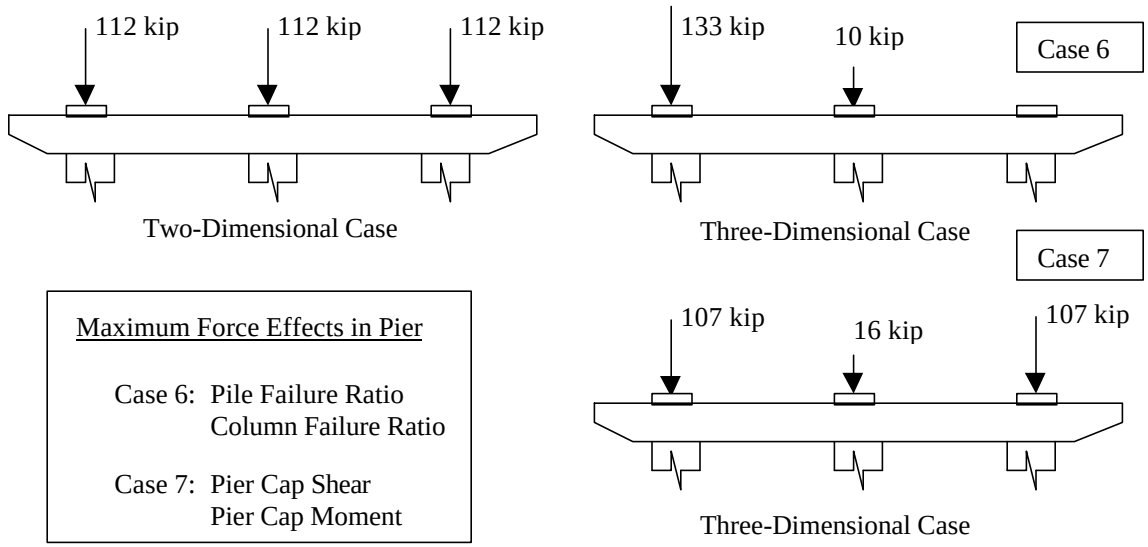


Figure 3.12 Maximum Force Effects for Case Study

The results of this preliminary case study clearly show that greater force effects can be achieved with a three-dimensional live load model. For this reason, the work that follows utilizes the three-dimensional model to determine the maximum force effects in the pier components. When considering a three-dimensional model though, the number of possible live load combinations greatly increases with the size of the bridge model. To overcome this impediment, a neural network implementation is developed in Chapter 5 to predict the correct position of the live loads on the bridge superstructure to produce the worst force effects. The neural network will predict the load positions to produce the four worst load cases to apply to the bridge pier, thereby eliminating any need for a complex live load generation study.

CHAPTER 4
NEURAL NETWORKS

4.1 Introduction

Neural networks are analytical tools originally created to model the cognitive function of the human brain. Their development has been deeply rooted in the biological sciences. For some time, neural networks were touted as a biological tool to understand the human learning and behavior. More recently though, the applications of neural networks have broadened to encompass issues in artificial intelligence and problem solving. This success of neural networks is due impart to their ability to be trained based on observed information. This key feature provides more flexibility in implementation over rule-based expert systems. During their initial developed, neural networks were known as *artificial neural networks* because of their ability to mimic the learning and memory of the human brain. These biological methods are now simply known as neural networks due to the widespread applications in both science and engineering.

Although there is a widespread usage of different types neural networks to solve problems in engineering, the implementations can be divided into several categories. Thus far, neural networks have been successfully implemented for applications involving pattern recognition, optimization, system identification, and function approximation. As the complexity of the network implementations grow though, various networks are likely to take on attributes of one of more of these categories to solve complicated problems.

In recent years, the knowledge and implementation of neural networks have increased drastically. This rapid growth is due inpart to faster and more reliable computing environments. The widespread acceptance of neural network techniques is also due to the persistent work of certain researchers. Since physicist John Hopfield (1982) published a paper on the collective computational capabilities of neural networks, research efforts have grown significantly. Even with the contribution of Hopfield, the flexibility of the network architecture was very limited. In 1986, however, Rumelhart, Hinton, and Williams removed the network complexity barrier with the introduction of the backpropagation training method. From this point on, neural networks have increased in scope and functionality. For more information concerning the development of neural networks, the reader is encouraged to read works by Hecht-Nielsen (1991) and Wasserman (1989).

4.2 Network Architecture and Operation

Neural networks attempt to simulate the functioning of the human brain by constructing a highly parallel computational structure. Much like their biological counterparts, neural networks are composed of many processing units known as *neurons* that are interconnected to form a parallel processing system. Most neural networks operate by processing some type of input signal to produce some form of output response. There is some distinction between the methods of information processing. If the signal travels from an input layer to the output layer in a single forward position, the network is called a *feed forward* network. Likewise, networks that work in reverse from output signals to input signals are termed *feed back* networks. Other networks that process signals forward and

backwards are known as *recurrent* networks. While these three processing schemes are the most common, other connection methods are certainly possible.

The topology of a neural network refers to the layout or architecture of the network. Most networks use a connective scheme whereby all of the processing neurons are linked by *connection weights*. These connection weights not only serve as a memory for the network but also act to amplify or inhibit the effects of the signal processing. The weighted connections are typically used to join layers of neurons as shown in Figure 4.1. The network layout begins with an input layer of processing units to represent the input signal. After the input layer at least one layer of hidden processing neurons are provided, followed by a final layer of output neurons to express the results of the processing. While the network operates as a parallel system, the individual elements of the network serve a unique purpose in the processing. Each neuron identifies a feature relating the input and output signals. Since the relationship between input signals and output response is quite often complex, many neurons may be necessary to determine the complicated mapping between input and output signals.

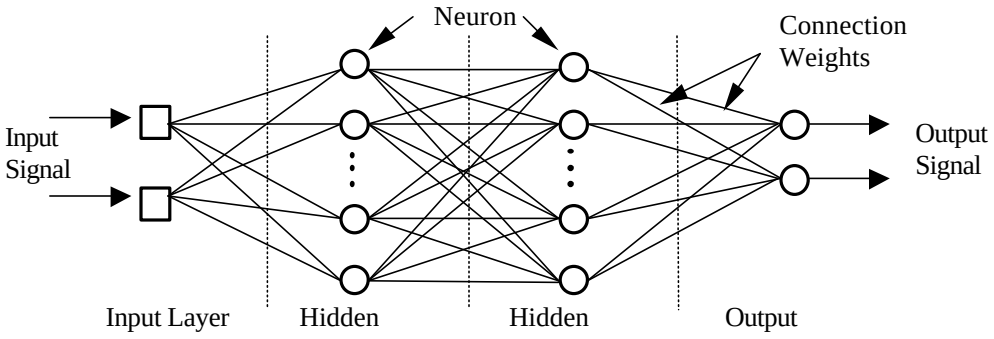


Figure 4.1 Typical Feed Forward Neural Network

The processing of information through the neural networks begins at the input neurons. These neurons sole function is to transmit a vector of input data, typically called an *input pattern*, to the first layer of *hidden neurons*. This vector of input data should consist of some representation of the problem to be solved. In most situations, there will be many input vectors to represent the scope of the problem, however, only one vector is processed at a time. During the input processing, each input pattern is individually sent along weighted connections that *fan-out* to link to all of the hidden processing neurons in first hidden layer. At the first hidden layer, the signals are transformed before the signals continue to travel through the network.

The method of signal processing is highly dependent on the function of the processing neurons. For most applications of neural networks, the processing is achieved using *nonlinear computing neurons* for all layers except the input layer. These nonlinear computing neurons allow the network to map complex relations between the input patterns and expected output responses. The neuron operates by first summing all of the input signals times their weighted connection values to achieve a weighted input sum. After adding an optional bias term, the weighted sum is then passed through a nonlinear transfer function to compute an output signal. The output signals from the current processing layer are then sent in parallel along weighted connections to the next layer of processing neurons. This signal transformation process is shown in Figure 4.2. In accordance with the underlying principles of neural networks, high signal strengths are encouraged to travel through the network, while low signal strengths are inhibited from proceeding through the network. This process mimics the pathways in the human brain that allow the most important signals to proceed through the network.

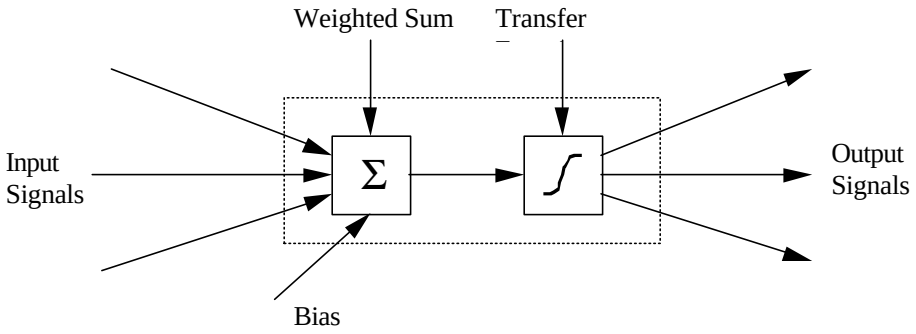


Figure 4.2 Nonlinear Computing Neuron

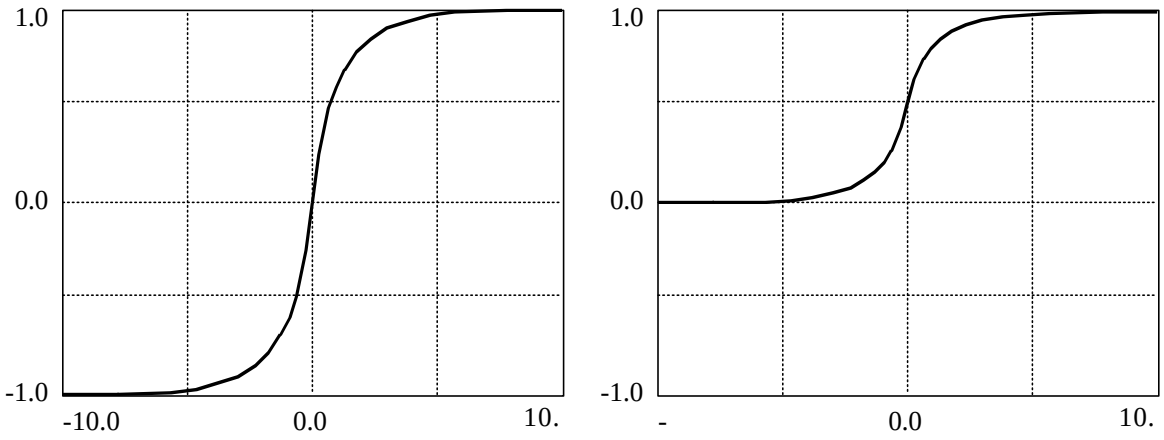
Neural networks can employ many different types of transfer functions to process the signal data. The simplest implementation uses a *Heaviside* or *threshold activation function* to transfer the data. This implementation is limited since it fails to form a smooth mapping between continuous variables. A more suitable mapping can be obtained using *sigmoid activation functions*. Unlike the threshold activation functions, these functions are capable of form smooth mappings between continuous variables. The two most popular sigmoid transfer functions have the form

$$g(a) = \frac{1}{1+e^{-a}} \qquad \textbf{(4.1)}$$

and

$$h(a) = \tanh(a) = \frac{1-e^{-a}}{1+e^{-a}} \qquad \textbf{(4.2)}$$

where a is the weighted sum at a neuron and the functions $g(a)$ and $h(a)$ serve to *squash* the signal values into a given range. Both the sigmoid activation functions are shown in Figure 4.3.



$$h(a) = \tanh(a) = \frac{1 - e^{-a}}{1 + e^{-a}} \qquad g(a) = \frac{1}{1 + e^{-a}}$$

Figure 4.3 Sigmoid Activation Functions

After the weighted sum of input signals is transformed with a nonlinear transfer function, the output of the neuron proceeds along weighted connections to the next layer of neurons. In this process the output signals generated from the neurons become the input signals for the next layer of hidden neurons. At the next layer of neurons, the same processing procedure is repeated with a nonlinear transfer function. The signals continue to pass through all of the hidden layers until the output layer is reached. The output layer usually utilizes a nonlinear transfer function as well, though it is not a necessity. For some neural networks, the output layer performs a summation of signal data but not the nonlinear transfer.

4.3 Problem Representation

The fundamental goal of employing a neural network is to produce some type of predicted solution to a problem. Most network configurations achieve a prediction after the successful completion of two tasks. First, the network must be taught how to solve the

particular problem. This step is usually accomplished by providing a series of example problems known as *training* vectors where the correct output response is known in advance. The network use stage then tells the network to solve the problem given a new set of input parameters.

In order for a neural network to be a viable solution to a problem, the network must present some advantage over alternative solution paths. In most cases, neural networks are advantageous in terms of time and computational savings. For complex problems, the training phase requires a significant amount of time to learn the relationship between the input and output data. However, the network use phase is very quick. Therefore, for one-of-a-kind problems, a neural network does not provide a viable solution due to the initial time outlay for training. For repetitive problems though, a neural network can provide a significant advantage since training occurs only once regardless of the number of problem variations.

4.4 Network Learning Models

Neural networks must in some manner learn the relationship between the input signals and expected output response. In terms of network operation, the network must posses some type of memory of previous problems in order to predict the response to a new problem. This memory is not arbitrarily assigned, but rather must be determined through a learning process. The method of learning varies greatly depending on the implementation of the neural network. No learning model is ideal and most have some limitations. For this reason, network training algorithms continue occupy more research hours than any other aspect of neural networks (Wasserman 1993). On a fundamental level

though, all learning models must present a group of closely related training problems to the network to commence the learning process. In doing so, this process follows the models of human cognition where individuals are exposed to a number of problems to increase their understanding and intuition about the subject matter. After the presentation of a group of training problems though, the methods of reinforcing the learning process vary significantly

Regardless of the learning model, the neural network must be modified in some way to encode the relationship between the input and output parameters. The most common network training implementations modify the connection weights between all the neurons during the training process. In this manner, the networks store the relation between input and output implicitly, without regard to any explicit set of learning rules. In the case of feed forward networks, the relationship is expressed by the weighted connections and the nonlinear transfer function at each neuron.

One of the oldest training algorithms was proposed by Hebb (1961). The method, commonly termed *Hebbian* learning, modifies the connection weight between two neurons depending on the activation of the two neurons. If both the source and destination neurons were active during a particular pattern presentation, the magnitude of the connection would be increased. In this manner, often-used paths in the network were strengthened while less-used paths were weakened. This method of learning is *unsupervised*, meaning that no verification of the learning process was provided. In Hebbian learning, the correct solutions to the training problems are not necessary since the method operates based on input parameters only. Hebbian learning provides an essential step in the evolution of network training algorithms. Since its development, however, more effective training algorithms have been created.

The most commonly implemented learning model for training neural networks uses *supervised learning* techniques. In supervised learning, the network is trained on a group of training patterns. For each input vector presented to the network, there is a corresponding output vector. This output vector mimics the function of a *teacher* to provide some validation of the learning process. During the training process, the teacher makes a comparison between the network output vector and the correct target vector. Any variation in the two vectors indicates the error in the network prediction to the given problem. The most widespread implementation of supervised training employs *error backpropagation* techniques (Rumelhart et al. 1996). In error backpropagation training, the difference between the network output and the correct output is used to adjust the connection weights. The method sends the error in output values backwards through the network to adjust the weights in a systematic manner. Critics of error backpropoagation argue that the method is biologically implausible, nevertheless the method has successfully been applied to many complex problems.

The final class of training algorithms rely solely on *unsupervised learning* techniques. In unsupervised learning, the training patterns do not include solutions or target vectors so the network is left to determine these automatically in the training process. Typically, unsupervised networks do not seek a direct problem solution, but rather use various clustering algorithms to classify the given problems. In its simplest form, training proceeds by adjusting weights in a way that forces each output neuron to respond to as many training patterns as possible without overlapping with the response of other output neurons (Flood and Kartam 1994a). When training is completed, each of the output neurons will represent a cluster of training patterns with similar attributes. Applications of

unsupervised learning have been limited due to their inability to predict direct solutions to problems. The methods are most appropriate for specific problems in pattern recognition and optimization and are not considered for implementation in this dissertation.

4.5 Supervised Training Techniques

Supervised training techniques are by far the most widely implemented learning models for training neural networks. To initiate the training, the method requires the pairing of each input vector with a target vector. Given these training vectors, the basic object of supervised training is to adjust the connection weights so that the output of the network closely matches the desired output. To achieve this goal, some measure of error in network prediction must be established. This measurement of error is most often expressed as

$$E = \frac{\frac{1}{2}\sum_{l=1}^{N_{sets}}\left\{\sum_{k=1}^{N_{out}}(T_k - O_k)^2\right\}}{N_{sets}N_{out}} \qquad \textbf{(4.3)}$$

where N_{sets} is the number of training sets, N_{out} is the number of network output neurons, T_k is the target output value for the k^{th} neuron, and O_k is the network output vector for the k^{th} neuron. The squared summation of errors over all output neurons and training patterns provides a positive, averaged assessment of the error during network training. This error function is based on the mean square error value used to evaluate the goodness of fit in statistical problems. For problems where the network prediction closely matches the target solution, the error function will be very small. In contrast, poor network representations of problems will produce high error values. The goal in supervised

training is to reduce the error to a minimum value. This minimum error value can be determined using different optimization techniques to adjust the values of the connection weights.

The objective of supervised training is to produce an approximate solution surface that relates the input and output parameters. This does not imply, however, that the solution surface should exactly fit all of the training data. If this were true, the network would simply memorize the training data so that any input pattern from the training set that was presented to the network would produce the exact corresponding output vector. This implementation could produce very small training errors, but would be quite limited since the network would only duplicate the results of the examples in the training process. In actuality, neural networks must be able to find a generalized solution that is applicable to all possible problems of interest. That is, a practical implementation of a neural network must produce an approximate solution to new and unfamiliar problems. To test the effectiveness of the network, these new problems should not be included in the training set.

The success of network prediction and functionality is often measured with some *validation* criteria. Network validation compares the predicted result of a network to the expected values. This validation phase usually relies on a trained network to predict an output response. The difference between the network prediction and the expected values can be expressed as

$$V = \frac{\frac{1}{2}\sum_{l=1}^{Nvalid}\left\{\sum_{k=1}^{N_{out}}(T_k - O_k)^2\right\}}{N_{valid}N_{out}} \qquad \textbf{(4.4)}$$

where N_{valid} is the number of validation problems used to verify the success of the network. The formulation is very similar to the determination of error during the network training process.

The error in training and validation provide one of the best indications of the success of the network implementation. Usually, the validation error is computed after completing network training, however the error can be computed at each cycle. In this case, the validation is performed on the current trained state of the network. The idealized plot of both errors during the training process in Figure 4.4 shows the learning trends of the network.

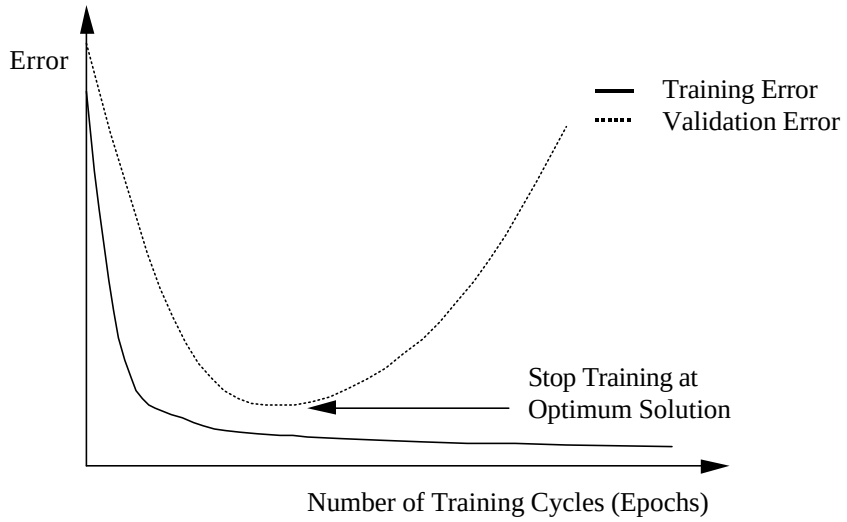


Figure 4.4 Errors in Training and Validation

An interesting observation can be made when comparing the error in training and validation during each training cycle or *epoch*. In a successful network implementation, the training error will decrease until a certain small error tolerance is reached. Most often at this point, there is very little change in error as the number of training cycles increases. The error with respect to the validation data should also decrease. For a proper

implementation though, the validation error should increase after reaching a minimum value. This increase in validation error ensures that the network has sufficient complexity to solve the problem. Otherwise, if the network is too small, there will not be a sufficient number of degrees of freedom to accurately model the problem. For networks with sufficient complexity, the increase in error corresponds to the memorization of the training data. This condition is termed *overtraining* and causes the network to lose its generalization capabilities (Hecht-Nielsen 1990). The optimum network solution is obtained by stopping the training process when the validation error is at a minimum. While the training error may not have reached a minimum, the generalization capability is optimal at the point of minimum validation error. Several training attempts may be necessary to identify the optimum stopping point for the network training.

4.6 Optimal Solution Techniques

There are a variety of solution techniques to determine the optimal network configuration. On a fundamental level, the different methods seek to adjust the values of the connection weights in order to minimize the network training error. This amounts to solving an unconstrained optimization problem. The time to obtain an optimal solution to the problem is most often dependent on the solution technique. Complex optimization routines may require additional computations per training cycle, but generally require fewer cycles to reach an optimal solution. Likewise, basic optimization routines require fewer computations per training cycle, but may require many cycles to reach an optimal solution. The key to obtaining a fast and reliable network solution is to select a method

that is both computationally efficient and simple to implement. Most techniques can be categorized based on either gradient descent or stochastic solution techniques.

Gradient descent techniques utilize the shape of the error surface to determine the optimal solution. The simplest method of gradient descent is called the method of *steepest descent*. In this method, the direction of steepest descent on the error surface (the gradient) is used as a search direction during each training iteration. Training begins by assigned random values to the connection weights to create an initial starting point for the optimization process. Without any intervention, the training process will proceed until a minimum training error value is achieved. Unfortunately, it is difficult to determine whether this value represents a local or global minimum. To check for global optimality, new starting points must be selected until there is some certainty that the global minimum value has been achieved. This process can be quite cumbersome and demonstrates the limitations of the method of steepest descent.

Before proceeding any further, it is important to visualize the gradient descent process. This optimization technique can be demonstrated with a simple example. A hypothetical error surface for a neuron with two weighted connections is shown in Figure 4.5. This error surface is plotted as a function of the two connection weights. The plot shows that there is an optimal combination of both weights that produces the minimum network training error. Depending on the initial starting point, the method of steepest descent will proceed to this optimal point in a variable number of steps. If the initial point is located where the gradient points directly to the minimum, the method will proceed efficiently to the minimum. However, as is most often the case, the initial point and

corresponding gradient are not as well aligned with the minimum. In this case, the indirect solution may require many iterations to finally reach the minimum error value.

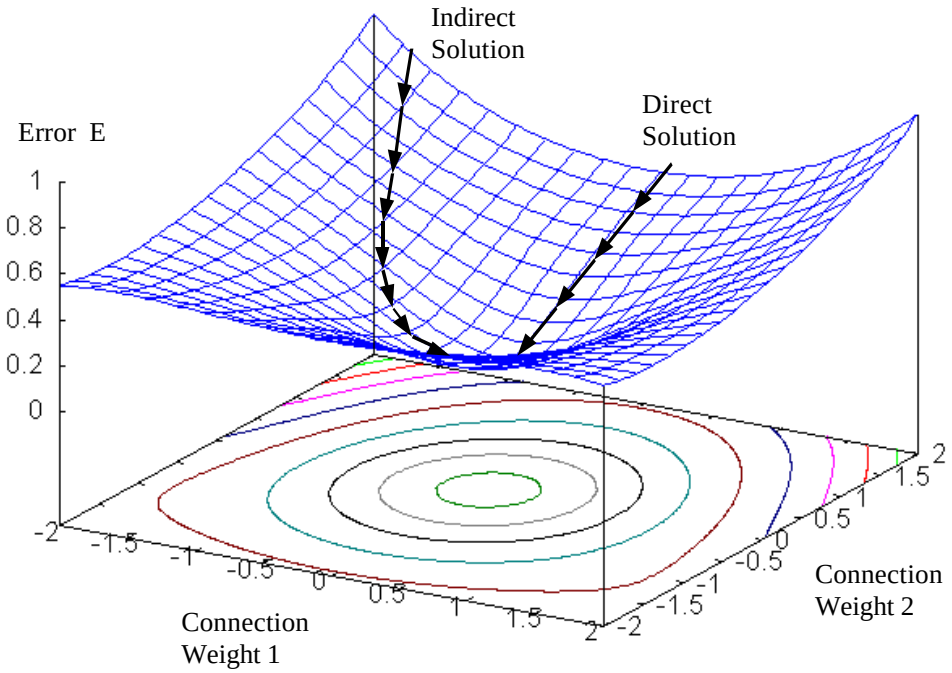


Figure 4.5 Hypothetical Training Error Surface

Many variations of the gradient descent method have been implemented to enhance the gradient descent process. The simplest methods seek to accelerate the solution process. The method of steepest descent makes no adjustments to the iteration step size. As a result, the choice of a small step size can lead to a very time consuming optimization process and a large step size can potentially miss points of minimum error. Methods such as adaptive learning and momentum (discussed in detail later) seek to accelerate the solution process and convergence to a local or global minimum. More complex methods employ additional knowledge about the shape of the error surface to accelerate the

convergence. Methods that utilize the slope and curvature of the error surface provide a more reasonable search directions to achieve a minimum solution. Among the most popular second order methods are the conjugate gradient descent and various descents based on Newton's Method.

The most significant limitation of the gradient based descents methods is the inability to determine the difference between a local and global minimum solution. Solution methods based on stochastic techniques seek to rectify this problem. *Stochastic* search techniques do not rely on the shape of the error surface, but rather employ some heuristic search technique to seek out the global minimum. These heuristic schemes usually favor evaluation points on the error surface with decreasing error values but allow a small probability of increasing error values. This allowance for increasing error values permits the search technique to escape points of local minimum.

There have been several practical implementations of stochastic search techniques for neural networks. The simplest technique takes a *random walk* across the error surface until a global minimum is found. This method can be very time consuming since virtually all of the error surface must be explored before confirming the existence of a global minimum. The second method, known as *simulated annealing*, bases the search process on the cooling of metals. This method utilizes a pseudo temperature parameter to control whether the search is allowed to permit in the direction of an increasing error value. This method is practical, though it requires the careful selection of parameters to avoid time consuming searches for the global minimum. The most recent method uses *genetic algorithms* to determine the global minimum. Genetic algorithms simulate Darwin's survival of the fittest premise, where different sampling points through many generations to

determine the global minimum. All three stochastic methods provide an alternative to the gradient based search techniques, however, they are almost always require more computational time to reach the optimal solution. However, the stochastic techniques virtually guarantee the discovery of the global minimum during the search process.

4.7 Backpropagation Training

Among the supervised learning techniques, the error backpropagation training method continues to be the most success technique for training multiple layer neural networks. It is estimated that over 85 percent of the neural network applications employ some form of an error backpropagation training (Wasserman 1993). The widespread implementation is primarily attributed to a theoretically sound technique for weight adjustment. The discovery of backpropagation training by Rumelhart, Hinton, and Williams in 1986 accelerated the growth of activity in the neural network field. Prior to this discovery, only single layer network solutions could be achieved. The supervised training technique used in error backpropagation enabled any number of connected layers to be trained, thereby overcoming the initial training barriers. There are now countless variations of error backpropagation due to its popularity, so the most basic form will be discussed first.

The training of multilayer networks is done in two steps: feed forward and back propagation. First the vector of input data is send through the network to produce an output vector response. Next this response is compared to a predetermined target vector to determine the error in network prediction. Any error is then sent backward through the network to adjust the connection weights in the different layers. This process proceeds

until the error reaches the input layer. The method then proceeds with a forward pass again using the new connection weights and computes a new reduced training error. Since this is an iterative process, many training cycles may be needed to reduce the training error to a minimum value.

In its fundamental form, error backpropagation training utilizes the method of steepest descent to locate a minimum point in the training error surface. When considering many training patterns, this method uses the sum of the errors over all of the output neurons and training sets to adjust the connection weights. Therefore, all training patterns must be presented to the network once before any weight adjustment can occur. After presenting all of the training patterns to the network, one *epoch* is said to have occurred. Since the method relies on an iterative process to adjust the connection weights, the network training is usually described in terms of the number of epochs.

Training usually begins by initializing all of the connection weights with small random values. This prevents a condition known as network *saturation*, where a collection of large weights activate nearly all of the neurons and impede the training process. Once the connection weights have been set, the first feed forward pass can begin to produce an initial output response. The output vector is then compared to the target vector to determine the network training error. It is extremely unlikely that the output vector will match the target vector on the first pass, so the error E is sent back through the network to adjust the connection weights.

The method of weight adjustment proceeds in one of two methods depending on the location of the connection weight. For connection weights between the last hidden layer and the output layer, the weight adjustment is straightforward. The weights are adjusted

using the difference between the output and target value at each output neuron along with the derivative of the transfer function at the neuron. The adjustment of connection weights in hidden layers is more complicated though since changing one of these weights changes the outputs of all neurons in the output layer. Using a process known as the *generalized delta rule* (Rumelhart et al. 1986), the error is sent backwards through the hidden layer in a recursive manner based on the chain rule from calculus. After all the weights in one layer are adjusted, the process is repeated, moving back toward the input layer by layer until all of the weights are adjusted. This iterative process continues until all weights have been adjusted to produce a minimum training error.

One further note of interest pertains to the shape of the error surfaces when using error backpropagation training. Keep in mind that the number of weighted connections represents the dimensionality of the problem. Since most network implementations use more than two weighted connections, the error surface can not be visualized but does nevertheless exist in multidimensional space. While some of the training behavior is still a mystery, three important observations have been made (Hecht-Nielsen 1990).

1. Many backpropagation error surfaces have extensively flat areas and troughs that have very little slope. In these areas it is necessary to move the weight value a considerable distance before a significant error reduction can occur.

2. Backpropagation surfaces have many global minima. This is due to the fact that there are many weight permutations that yield exactly the same overall network input/output function.

3. Local minima exist with error values greater than those at the global minima.

These three concepts are very important and lead the way to methods of enhancing error backpropagation methods to avoid these pitfalls.

The details of the mathematical foundations of different error backpropagation methods can be found in most books on neural networks. There are many variations of the methods to suit a particular problem solution. Most methods are directed towards improving the speed of convergence, a problem that plagues many backpropagation applications. With the simple introduction of various training techniques though, the time to reach convergence can be drastically reduced. The most practical enhancements to error backpropagation are discussed below.

4.7.1 Single Example and Batch Training

The rate of convergence can vary depending on the how the training sets are introduced to the network. For the error formula presented in Section 4.5, the error is summed over all of the output neurons and all training sets. The formulation requires that all training sets be presented to the network before any weight adjustment can occur. In a sense, this procedure has the effect of averaging the network training error over all of the training sets. However, since a typical network training involves many training sets, the training time per cycle can be significant under this method.

A second possibility for network training updates the connection weights after the presentation of each training pair or a batch of training pairs. When considering individual pairs, training can progress quickly since fewer computations are required to adjust the weights for a single example. This situation gives the impression that the network training is progressing quite rapidly, however the error in training is not representative of all of the training sets. The alternative is to process batches of training examples at one time. If each batch contains very similar examples, the training progresses rapidly since the subsequent examples require very little weight adjustment after the presentation of the first

example in the batch. Also, in the special case that two identical training problems in a given batch are presented, the connection weights will not need to be adjusted for the second training pair. In practical problems though, the training examples are presented at random so a batch mode may not be beneficial. The best representative training error will be found if the batch size is increased to the training set size.

4.7.2 Adaptive Learning

As previously stated, the error surface under backpropagation is usually relatively flat and widespread with intermittent steep canyons. Therefore, the choice of a step size for the optimization process is crucial. Since the standard backpropagation method does not adjust the step size during the optimization process, many iterations may be required to cross the relatively flat surface if a small step size is used. Equally important, the choice of a large step size may fail to locate a minimum by skipping over any drops in the error surface. Adaptive learning seeks to rectify both problems by modifying the step size during the optimization process.

The most prominent form of adaptive learning was proposed by Jacobs (1988). The original form of the method, known as the *delta-delta* rule, introduced separate learning rates for each weight in the network. This learning rate acts as a multiplier in front of the weight changes. Therefore, a small factor would slow down the weight adjustment process and thereby slow down the learning process. Likewise, a large factor would allow larger weight changes. Under the delta-delta rule, the learning rates are updated in a linear fashion according to the slope of the error surface during each training cycle. If the derivative of the error surface with respect the current weight possesses the same sign for consecutive iterations the learning rate will increase to speed up

convergence. If the derivative changes sign during consecutive iterations, the learning rate will be reduced to prevent oscillations in the solution process. Both scenarios are shown in Figure 4.6. As one further note, a modification can be made to the delta-delta rule to produce the *delta-bar-delta* rule. This new rule increments the learning rate linearly and decrements the learning rate exponentially. The exponential decrement penalizes the learning rate more severely than a linear decrement to better control oscillations around the minimum.

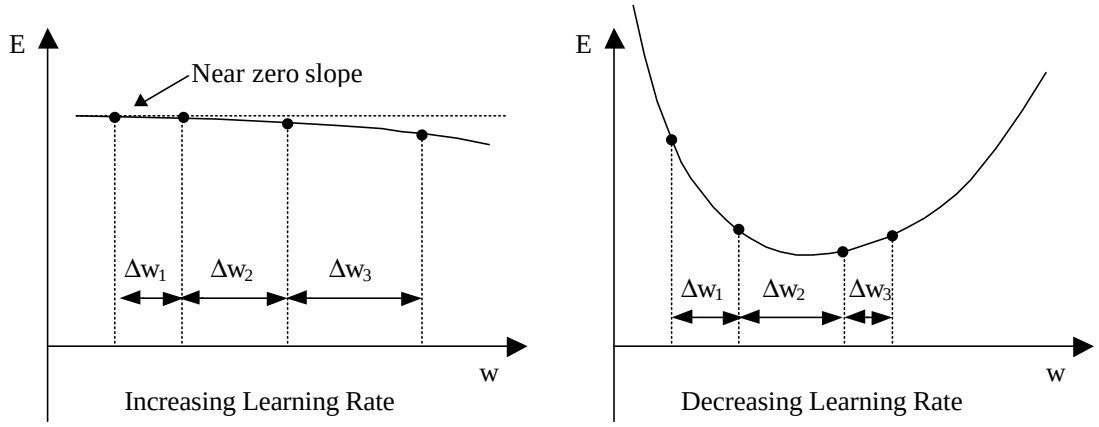


Figure 4.6 Variation of Learning Rate

4.7.3 Momentum

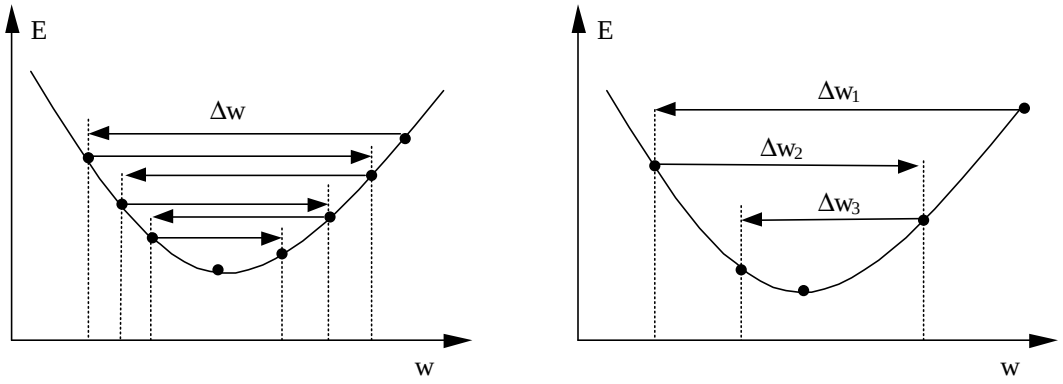
One simple technique to improve the rate of convergence during network training is to add a *momentum* term to the gradient descent formula. This momentum term has the effect of adding inertia to the motion through weight space and smoothes out the oscillations in local minima (Bishop 1995). This effect is accomplished by retaining the previous weight change during the training process. Depending on the sign of the previous weight change, the current weight change either increases or decreases with the supplemental momentum term. There are several variations of momentum however the

simplest form is described for implementation here. The change in connection weight for each training epoch is computed as

$$\Delta w_i = (1 - m)\Delta w_i + m\Delta w_{i-1} \quad \textbf{(4.5)}$$

where Δw_i is the weight change for the current epoch, Δw_{i-1} is the weight change for the previous epoch, and m is a momentum parameter. Unfortunately, this formulation requires the careful selection of a momentum parameter m, which must range from zero to one. If a value of $m = 0$ is used, then the previous weight change is ignored and the method proceeds as a simple gradient descent technique. Conversely, a value of $m = 1$ ignores the effect of the current weight change and uses the previous weight change instead. The ideal momentum parameter is usually determined through experimentation.

The most significant advantage of adding a momentum term to the backpropagation routine is that it prevents unnecessary oscillations around local minima. Unless the step size is exactly chosen, the standard backpropagation routine will oscillate around the minimum, slowly converging to the point of minimum error. Because of the requirement that the gradient of the error surface be zero at the minimum, convergence to this point can be very time consuming for steep slopes around the minimum. This phenomena is illustrated in Figure 4.7. With the addition of the momentum term however the solution proceeds rapidly to the point of minimum error.



Training without Momentum Training with Momentum

Figure 4.7 Oscillating Solutions and Momentum

The second advantage of a momentum term is to prevent entrapment in shallow local minima. For simple gradient descent solutions with a small step size, the method will stop at the first point of minimum error. Sometimes this point is a shallow local minimum next to a deeper global minimum. As stated before, the momentum term provides an inertia of motion to the gradient descent routine. In certain cases this inertia is enough to push the motion out of shallow local minima in an effort to reach a lower minimum as shown in Figure 4.8. This effect comes as a bonus with the inclusion of momentum to the training process.

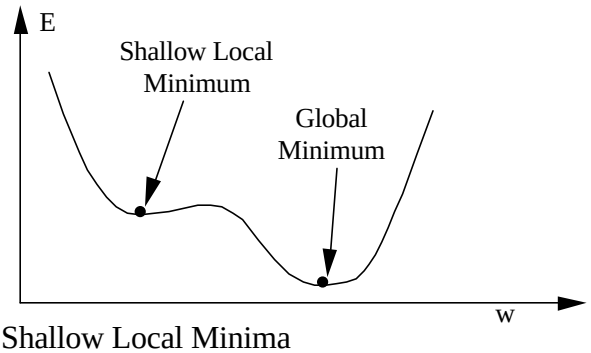


Figure 4.8 Escaping Shallow Local Minima

4.7.4 Second Order Techniques

Even with the addition of momentum and adaptive learning, backpropagation can not guarantee a direct path to the minimum solution. The backpropagation algorithms only use the first derivative (the slope) of the error surface to find the point of minimum error. By making use of the second derivatives (the curvatures) of the error surface, the convergence time can be reduced by up to a factor of 100 (Wasserman 1993). Although more computations maybe necessary to compute the second derivatives of the error surface for each training iteration, the end gain is a direct solution path to a minimum. Two of the most prominent second order methods, are discussed below for possible implementation.

A very common second order method for minimizing a function of many variables is the *conjugate gradient method*. Since neural networks consist of many variables (the connection weights), the method is particularly well suited for implementation into a network training algorithm. Unlike the steepest gradient descent method which results in many false steps through zig-zaging, the conjugate gradient method proceeds at directions that are conjugate to the direction of the previous step. This method allows a series of noninterfering steps to the minimum, where the minimization performed in one step is not partially undone in the next step (Wasserman 1993). The end result is a quick solution for the minimum error that does not zig-zag through many iterations.

A second very popular second order method for a minimizing function of many variables is the *Levenberg-Marquardt algorithm* (LM). This method was developed by Levenberg (1944) and Marquardt (1963) specifically for minimizing a sum-of-the-squares error. The method is also particularly suited for neural network applications since most applications use a sum-of-the-squares function to quantify the training error. On a

fundamental level, the method tries to keep a small step size so that the linear approximations to the solution remained valid. At the same time, the method utilizes the second derivatives of the error surface to find the optimum search direction. This approach turns out to be very effective and can reduce training time significantly.

4.8 Refined Network Topology

There is great debate about the suitable topology or layout of the network to optimize the overall functionality. Ultimately, the topology of the network is directly related to the expected application and complexity of the problem to be solved. Various researchers have established guidelines for network topology. Since part of the objective of this dissertation is to develop efficient neural networks, methods of refining topology are worth mentioning.

4.8.1 Number of Hidden Layers

Hidden layers are used to increase the complexity of the mapping between the input parameters and the output response. Networks without a hidden layer can only represent relationships that are linearly separable. For the most part, networks without a hidden layer are very limited. Networks with one hidden layer however can represent convex regions though the intersection of hyperplanes. For very complex data mappings, two hidden layers can be used to form relationships that are non-convex and disjoint. A simple example of all three cases (Bishop 1995) is shown in Figure 4.8.

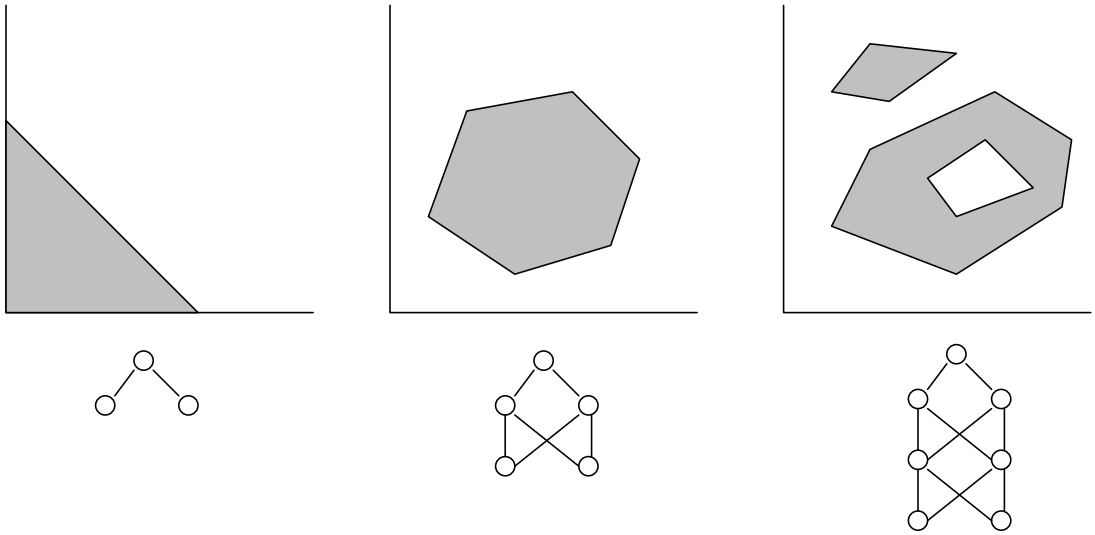


Figure 4.9 Mapping Network Decision Boundaries (After Bishop)

Much of the debate over the adequate number of hidden layers stems from the lack of a unified theorem of convergence. There are two limited theorems that propose solutions to the debate. The first theorem offered by Hecht-Nielsen (1990) provides a proof that one hidden layer is sufficient to model any surface of practical interest, provided that sigmoid activation functions are used. A second theorem by Lapedes and Farber (1988) shows that two hidden layers are sufficient to model a surface of any complexity. Researchers have identified some situations where a single hidden layer is not sufficient to map a problem (Mukherjee and Deshpande 1997). Likewise other researchers insist from experience that two hidden layers are not necessary (Rajasekaran and Ramasamy 1997). This work adopts the use of a single hidden layer to map the relationship between input and output parameters.

4.8.2 Number of Hidden Neurons

There is perhaps an equally great debate concerning the number of neurons in a hidden layer. According to Flood and Kartam (1994b), increasing the number of neurons provides a greater potential for developing a solution surface that fits closely to that implied by the training patterns. Unfortunately, increasing the number of neurons without increasing the number of training sets can produce an overfit solution. Such a solution would be nearly exact for the training set, but may deviate significantly from the trend of the surface at any verification points. According to Berke and Hajela (1991), the number of neurons in the hidden layer should be between the average and the sum of the nodes in the input and output layers. Other researches suggest no such limit for the number of hidden neurons. In this dissertation, a network pruning algorithm will be implemented to determine the optimum number of neurons in the hidden layer to produce the approximate solution to the problem.

4.9 Network Pruning

In order to guarantee a good generalization, the number of connection weights in the network should be less than the number of training sets. This idea suggests that the complexity of the network is limited by the number of training sets available to represent a particular problem. While this upper limit of complexity is true for most network implementations, there are few indications of a lower limit on the size of the network. That is, in some cases the size of the network can be reduced and still achieve good generalization while maintaining the same number of training sets. Since the number of computations increases with the complexity of the network, the goal is to obtain an optimal

network size. Before proceeding any further it is advised to first select a network architecture that provides suitable complexity for the problem to be solved. Once a particular architecture is established, the generalization capability can then be improved if the number of free parameters (connection weights) in the network is optimized (Hassoun 1995).

The general process of reducing the complexity of a network to optimize the functionality is termed *network pruning*. The term is derived from the act of removing or pruning the parts of a network to reduce the size. This leads to either the removal of various weighted connections or neurons in the network. The removal of weighted connections can be achieved by employing some method of weight decay, whereby the non-influential connections weights are reduced to zero through an iterative process. In theory, the removal of such weights will not affect the network output. The removal of neurons can be achieved by studying the activation of neurons during the training process. If a particular neuron is inactive during the training process it can be removed. This removal can damage the operation of the network though if the neuron is acting to impede certain signals from traveling through the network. This approach is far more risky for finding the optimal size of the network.

The most effective network pruning algorithms study the *sensitivity* of the error function (Karnin 1990). Such methods determine the effect on the training error with the inclusion or exclusion of various connection weights in the network. Under this system each connection weight is assigned a sensitivity value during the training process. Connection weights with high sensitivity values will greatly affect the training error. Likewise, connection weights with very low sensitivity values will not affect the training

error. Such connections with low sensitivities could ultimately be removed without affecting the network output. This method is particularly beneficial since the sensitivity information is gathered during training so that the network pruning can occur after training. Therefore, the network only needs to be trained once before an optimum network size is found. Further training can then be pursued to refine the network topology.

CHAPTER 5
POSITIONING LIVE LOADS ON SINGLE COLUMN PIERS

5.1 Introduction

Neural networks can be readily applied to solve problems in structural engineering. Since the functionality of the networks is not dependent on any physical quantities, the same network principles can be used to solve a variety of structural problems. Neural networks have already been employed to solve academic problems as well as complex structural systems. While simple examples demonstrate the potential of neural network techniques, the solution to complex structural problems are calling great attention. The analysis of complicated structures can be computationally expensive and very time consuming. Neural network implementations that reduce the solution time or circumvent the complex analytical process are certainly advantageous to structural designers.

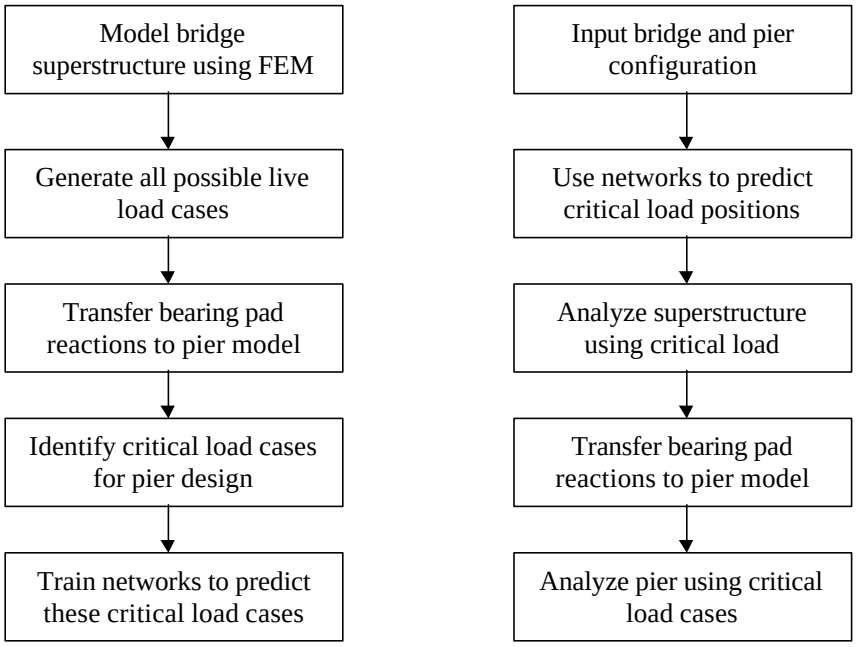
One of the main goals of this dissertation is to automate portions of the bridge design process. Given the complexity of modern highway bridges and the uncertain behavior of any externally applied loads, any automation to the design process is certainly welcomed. In this particular investigation, neural networks are implemented to predict the positioning of vehicle live loads on highway bridges to produce the worst force combinations in the supporting bridge piers. The networks are utilized to learn the behavior of the bridge system under certain loading patterns. The function of the neural network is in no way connected to the principles of structural analysis. Rather, the network

must formulate a solution based on the observations of patterns between the input and output parameters that describe the bridge system.

The formulation presented in this chapter addresses the positioning of the live loads on single column piers. The single column piers, commonly called “hammerhead piers,” are rather straightforward to analyze under live loading. The results can readily be verified by hand calculations and serve to validate the finite element analysis of the pier structure. Once satisfactory results are achieved the analysis can be broadened to consider the behavior of multiple column piers under live loading. Chapter 6 will discuss the more complicated positioning of live loads on multiple column piers.

5.2 Overview of Network Creation and Operation

Before proceeding with the analysis of single column piers it is beneficial to provide an overview of the methodology followed herein. The flowcharts in Figure 5.1 show the general steps that are followed to train and operate the neural networks. Figure 5.1(a) outlines how the procedures described in Chapters 2, 3, and 4 are implemented to train the neural networks. Figure 5.1(b) outlines how the networks will operate to predict the critical load positions for the piers after the training phase is completed. It should be noted that the network operation phase proceeds much more rapidly than the training phase since only a single pass through the network is required to determine the critical load positions. This methodology presented in Figure 5.1 will also be implemented for the multiple column pier analyses in Chapter 6.



(a) Network Training (b) Network Operation

Figure 5.1 Overview of Network Training and Operation

5.3 Encoding Structural Behavior

A general neural network does not incorporate any aspect of structural behavior in the network formulation. The network simply responds to a group of input parameters to produce an output response. The behavior of the pier system must therefore be encoded within the structure of the network in order to mimic the response of the pier system under live loading. This encoding can be achieved via the weighted connections between the processing neurons in the network. Through the training process outlined in detail in Chapter 4, the weighted network connection can be modified so that the output response closely matches a set of target network responses. In this application, these target results represent the positioning of the live loads on the pier system. Although the network has no direct knowledge of structural behavior, the network is fully capable of mapping the

structural response to some input source. Put simply, the network is capable of observing and recording the behavior of the structure without understanding any of the underlying physical principles for the behavior.

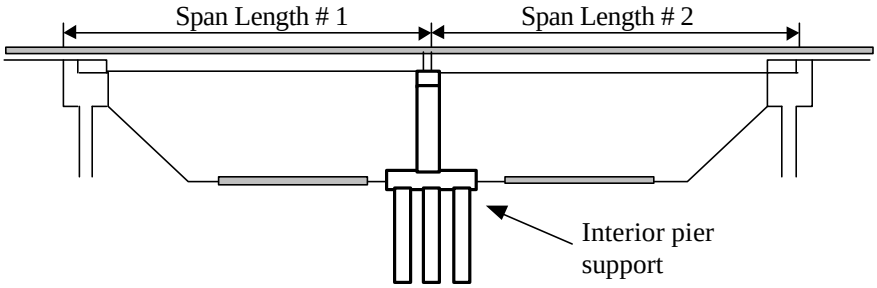
5.4 Dimensionless Network Parameters

For networks trained by error backpropagation, the encoding of structural behavior is achieved through the evaluation of input and output parameter sets during the network training process. The input and output parameters must be sufficient to describe the structural problem and encode the proper structural response under live loading. There are no restrictions on the magnitude of the parameters, however, previous research indicates that the use of dimensionless parameters can be advantageous for several reasons. Foremost, a large range of values for the input parameters can lead to an early saturation and subsequent stagnation of the network during training. Secondly, in terms of the optimization routines employed, a wide range of values can lead to numerical difficulties due to the formation of steep canyons in the solution surface. Finally, the use of dimensionless parameters can eliminate unrelated or redundant variables through the use of dimensional analysis.

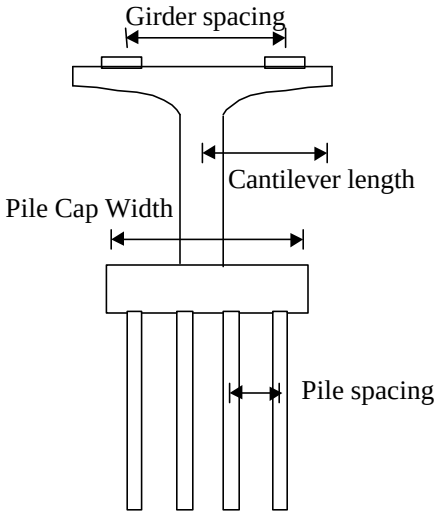
5.4.1 Input Parameters

The network input parameters initiate the operation of the network. With the emphasis herein on the correct positioning of the live loads on the bridge piers, the input parameters must foremost describe the pier configuration. Since the networks presented in this chapter are created for single column piers, information about the column geometry is

not needed since a single column geometry is already implied. Of primary importance for interior piers are the bridge span lengths on either side of the pier, the length of the cantilevered pier cap, the width of the clear roadway of the bridge, the spacing of the girders, and the pile spacing. These quantities, illustrated for the bridge and pier system in Figure 5.2, can be arranged into dimensionless ratios using the theory of dimensional analysis.



(a) Complete bridge system



(b) Pier configuration

Figure 5.2 Single Column Pier Configuration Variables

Dimensional analysis is commonly used to arrange variables into dimensionless groups. This can most effectively be done with the Buckingham Π Theorem (Buckingham

1915), which provides a generalized approach for forming these dimensionless groups. The Buckingham Π Theorem states that if there are *n* dimensional variables in a dimensionally homogenous equation, described by *m* fundamental dimensions, they may be grouped in *n* – *m* dimensionless groups, usually referred to as "Π terms". For example, if there are 5 dimensional variables and 3 fundamental directions there will be 5 – 3 = 2 dimensionless groups. This example can be shown mathematically in Equation 5.1.

$$D_1^0 D_2^0 D_3^0 = V_1^a V_2^b V_3^c V_x^d \quad \textbf{(5.1)}$$

In Equation 5.1 D_1, D_2, and D_3 are the three fundamental dimensions, V_1, V_2, and V_3 are the primary variables and V_x represents the remaining variables. In the Buckingham Π Theorem, the primary dimensional variables are equal in number to the fundamental dimensions and should be chosen such that each primary variable contains one or more of the fundamental dimensions. The remaining dimensional variables are substituted in for V_x one at a time when solving for each of the Π terms. By applying the remaining dimensional variables one at a time and writing each variable in terms of its fundamental dimensions, the exponents *a*, *b*, *c*, and *d* can be determined. In doing so, each Π term can then be expressed as a ratio of the dimensional variables.

The Buckingham Π Theorem can be applied to the present investigation. For this case, the fundamental dimensions are chosen to represent the dimensions of the pier structure and foundation. This work adopts three fundamental dimensions: the transverse pier structure length (L_S), the longitudinal length (L_L), and the transverse foundation length (L_F). For clarity, the bridge spans the longitudinal direction and the pier and foundation are oriented in the transverse direction. The dimensional variables represent the geometric

configuration of the bridge and pier system. For single column piers, the six variables are: girder spacing, pier cap cantilever length, span length #1, total span (span length #1 plus span length #2), pile spacing, and pile cap width. Since there are three fundamental dimensions, there will be three primary dimensional variables, chosen here as: girder spacing, span length #1, and pile spacing. The remaining three dimensional variables will be substituted into Equation 5.1 one at a time to determine the three Π terms. The dimensional variables are summarized in Table 5.1.

Table 5.1 Dimensional variables (Single Column Piers)

Dimensional variable	Symbol	Fundamental dimension
*Girder spacing	SG	L_S
*Span length #1	SL1	L_L
*Pile spacing	SP	L_F
Cantilever length	CL	L_S
Total span	TS	L_L
Pile cap width	CW	L_F

* Primary dimensional variables

The three Π terms are determined in Equations 5.2 through 5.6. To solve for the Π_1 term, the cantilever length is substituted into Equation 5.1 along with the primary dimensional variables, as shown in Equation 5.2.

$$L_S^0 L_L^0 L_F^0 = (SG)^a (SL1)^b (SP)^c (CL)^d \quad \textbf{(5.2)}$$

The dimensional variables can then be written in terms of their fundamental dimensions, as shown in Equation 5.3, in order to solve for the unknown exponents *a*, *b*, *c*, and *d*.

$$L_S^0 L_L^0 L_F^0 = (L_S)^a (L_L)^b (L_F)^c (L_S)^d \quad \textbf{(5.3)}$$

Equating the exponential terms containing the fundamental dimension L_S leads to the equation $0 = a + d$. The exponents b and d are not involved in this equation and can be shown to be zero. Assuming that the exponent a = -1, then d = 1 and the resulting Π_1 term can therefore be written as shown in Equation 5.4.

$$\Pi_1 = \frac{CL}{SG} = \frac{Cantilever\ length}{Girder\ spacing} \tag{5.4}$$

The Π_2 and Π_3 terms can be determined similarly and are shown in Equations 5.5 and 5.6.

$$\Pi_2 = \frac{SL1}{TS} = \frac{Span\ length\ \#1}{Total\ span} = Normalized\ span \tag{5.5}$$

$$\Pi_3 = \frac{SP}{CW} = \frac{Pile\ spacing}{Pile\ cap\ width} \tag{5.6}$$

These three Π terms along with the number of design lanes can now be used as dimensionless input parameters to the neural networks to identify the critical load positions for the single column piers.

Prior work by the author concluded that it was not practical to create a single network to correctly predict the load placement to produce all four maximum force effects. Rather, four separate networks are created to identify each of the maximum force effects. For simplicity, each of the four networks uses the same dimensionless input parameters to describe the bridge and pier configurations. The first parameter represents the number of design lanes for the bridge. The remaining three input parameters are determined based on the dimensional analysis presented above. These four input parameters should adequately

describe the geometry of the bridge and pier system and the effect of the soil-pile interaction in the foundation. The four input parameters are shown later with the complete neural network configuration in Section 5.7.

5.4.2 Output Parameters

The output parameters represent the final response of the network. For this investigation, the desired result was the correct placement of the live load across the clear roadway of the bridge to achieve the maximum forces in the pier. For simplicity, the placement of the center of gravity of the live loads was measured from the left curb of the bridge across the bridge width. The exact dimensional placement of the loads was first considered as a network output, however, a more effective result was achieved by normalizing the result. The output response was normalized by dividing the dimensioned load placement by the width of the clear roadway. This normalization procedure created a live load placement range from 0 to 1 as illustrated in Figure 5.3.

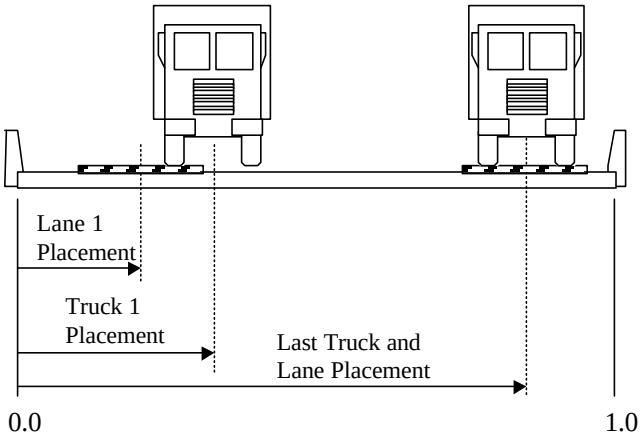


Figure 5.3 Normalized Placement of Live Loads Across the Bridge

The work presented herein considers bridges with a maximum of four design lanes. Since the design truck and lane load can be positioned independently within a lane in any of the four lanes, eight output neurons are required to represent the output response. These eight output neurons will be used for each of the four networks used to determine the load placement for the maximum force effects. Again, each of these output parameters represents a normalized placement of the load (truck or lane) across the width of the bridge. For unoccupied lanes, the parameters are set to zero.

5.5 Maximum Force Combinations

The objective of the live load placement on the pier is to achieve the maximum combination of internal forces in a particular element of interest in the pier model. In terms of the present analysis, the determination of the maximum forces in the piles, pier column, and pier cap are of primary importance for the subsequent design of the pier. Since it is not immediately obvious which live load placement will produce the maximum desired force effect, all combinations of live load must be considered in the analysis.

As stated before, this investigation considers the occurrence of four different maximum force effects in the pier. The first force effect considers the maximum combination of axial force and bending moments at a section in the piles. Likewise the second force effect considers the maximum combination of axial force and bending moments at a section in the pier column. The remaining two force effects consider the maximum shear force and bending moment in the pier cap. Although the shear and bending moment in the pier cap are related through structural mechanics, the maximum shear and moment effects are considered separately in the analysis. This separation is primarily due

to the fact that the point of maximum moment in the pier cap does not necessarily correspond to the point of maximum shear. A separate network is therefore needed to determine the live load placement to produce the maximum shear force in the pier cap.

The determination of the maximum force combination depends on the material behavior assumption in the analysis. Since this investigation considers a variety of piers, some analyses consider the nonlinear material behavior of the pier while others assume a linear material behavior. The Florida Pier program currently determines the maximum combination of forces at a section based on the amount of material information provided. For more complex linear and nonlinear analyses a failure ratio is calculated. For a more simplified linear analysis, a SRSS force combination is utilized.

5.5.1 Failure Ratio

For nonlinear analysis or linear analysis with cross-sectional properties, Florida Pier calculates the force combination in terms of a failure ratio. After considering all possible combinations of live load for the pier, the maximum failure ratios for the piles and pier column are calculated to determine the critical live load placement. The critical live load placements are then recorded for each pier in the training set. The load placements are later converted to output parameters through normalization and matched with the corresponding input parameters. The network can then be trained after determining all of the output parameters for each training set.

5.5.2 Square Root of Sum of Squares (SRSS)

For pure linear elastic analyses, only gross sectional properties are used. For this case, the combination of forces using the SRSS method outline in Chapter 3 is utilized to determine the critical load positions. In a manner similar to the failure ratio determination, the critical live load placements are recorded for each pier in the training set. The load placements are later converted to output parameters through the same normalization process and matched with the corresponding input parameters.

5.6 Critical Load Positions and Load Symmetry

The determination of the correct critical live load placement is essential to the success of the prediction capability of the neural network. The improper determination of the critical load position for a particular pier could lead to the failure of the network training process or more likely a misinterpretation of the structural behavior. Therefore, certain assumptions concerning the positioning of the loading were made to ensure that a consistent method was followed to determine the live load placement.

For this application of live load generation, many of the loading patterns were symmetric across the width of the bridge. That is, a load placed to the left edge of the bridge was also placed at the far right end of the bridge at some time during the live load generation. The same case was also true for multiple lanes loading. For many of these symmetric cases the extreme force effect sought from the finite element analysis of the pier system were very similar. The results were not identical though, due to slight numerical differences in the analysis, leading to a difference in some cases of less than 0.01%. Since the current implementation searches for the maximum force effect, one load pattern must be

chosen over the other even though the loading patterns are symmetric. For this reason, the critical load pattern was always taken as the leftmost symmetric loading pattern. To clarify, if a load positioning at the far right edge of the bridge produced the worst force effect, the load positioning at the left edge would be chosen due to the symmetry of the loads. If this condition were not enforced, the neural networks would have great difficulty in predicting the worst live load placement based on slight numerical differences. The neural networks function to provide an approximate solution to the problem and clearly fall short of providing a numerically precise solution.

5.7 Networks for Predicting Critical Load Positions

The objective of this investigation is to create a neural network to predict the load positions to produce the maximum force effects in the pier system. Due to the differences in the type of force effects, it was not possible to create a single network to identify all four maximum force effects. Given this inherent difficulty, four separate neural networks were created to incorporate the uniqueness of the force effects. For simplicity, the network configuration remained the same for all four networks. However, four different training sets were used with the network configuration to map the load positions to the four different maximum force effects. The network configuration used to position the loads on single column piers is shown in Figure 5.4.

Some assumptions were made about the initial network configuration. As explained before, the number of input parameters was set at four to sufficiently describe the problem. The number of output parameters was set at eight to accommodate a maximum of four lanes of traffic. One hidden layer of neurons was implemented given the simplicity of

the single column pier problem. Six processing neurons were used in the hidden layer in the initial network configuration. This number will later be reduced through a network pruning process to optimize the network topology. Both the hidden layer and output layer used sigmoid transfer functions to process the signals from the weighted connections. The sigmoid transfer functions squashed the data into the range [0,1] during the signal processing.

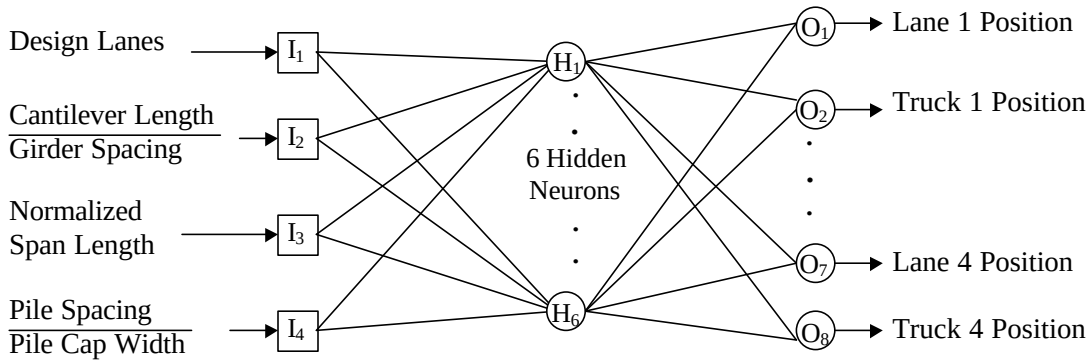


Figure 5.4 Network Configuration for Single Column Piers

The four networks were created to identify the loading to produce the four maximum force effects needed for the design of the pier. Each force effect is explained in detail below.

5.7.1 Maximum Pile Force Combination

The design of a pile system is based on both the axial and flexural bending behavior of the piles. For this case, the maximum force effect is identified as the maximum force combination in a single pile under axial load and bending moment. This combination of axial force and bending moment in turn depends on the spacing of the piles, the type of soil, and the characteristics of the pile. Since there can be great variability in soil

properties, pile configurations, and pile parameters, the results can vary significantly from pier to pier. Therefore, the neural network predictions are expected to be approximate, though still adequate for the design of pile system.

5.7.2 Maximum Pier Column Force Combination

The determination of the maximum force combination in the pier column for hammerhead piers is rather straightforward. Given the configuration of the pier structure, the live load positioning will produce the maximum combination of axial load and bending moment at the base of the column. It is not clear, however, how many lanes need to be loaded to produce the maximum force effect. Loading the exterior lane of the bridge will most likely produce the maximum bending moment in the pier column, but not the maximum axial force. Loading all of the lanes will produce the largest axial force, but no bending moment in the column if the load is symmetrically placed above the column. This condition is illustrated in Figure 5.5 for a trivial pier analysis case. For actual pier designs, this case is further complicated since there is a reduction in live load as the number of loaded lane increases.

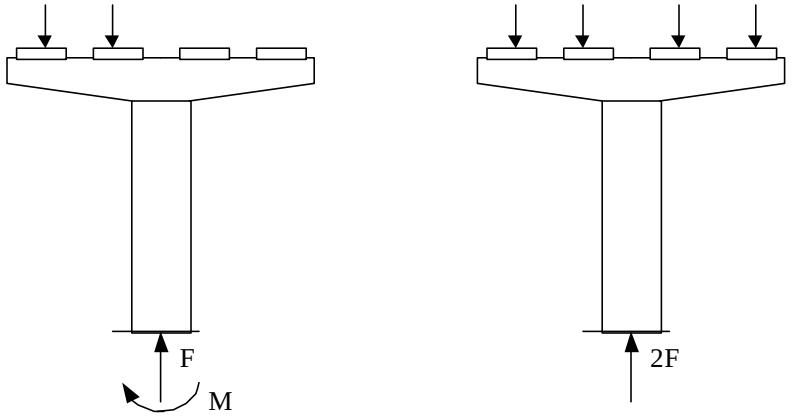


Figure 5.5 Maximum Force Combination in a Single Pier Column

5.7.3 Maximum Pier Cap Shear Force

The determination of the maximum shear force in the pier cap is particularly important for design of the pier cap section. Since the live loads are applied as concentrated forces via the bearing pads, the shear in the pier cap directly depends on the positioning and magnitude of the load. For this case, the spacing of the bearing pads and length of the cantilever section are very important. A typical case for maximum pier cap shear along with the corresponding shear diagram is shown in Figure 5.6.

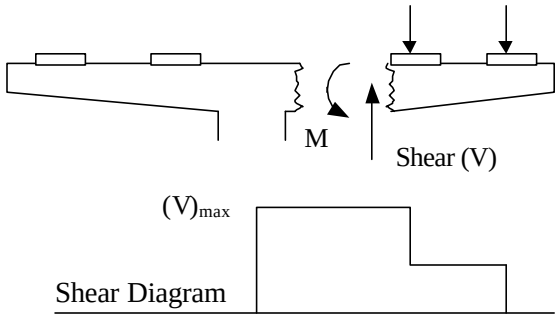


Figure 5.6 Maximum Pier Cap Shear in a Single Column Pier

5.7.4 Maximum Pier Cap Bending Moment

The determination of the maximum bending moment in the pier cap is equally important for design of the pier cap section. As was the case for the shear in the pier cap, the bending moment also directly depends on the positioning and magnitude of the load. For this case, the spacing of the bearing pads and length of the cantilever section are very important. For single column piers, the maximum moment will always occur at the base of the pier cap as shown in Figure 5.7. It is uncertain, however, how many traffic lanes should be loaded to produce the maximum moment. The lane loading directly depends on the width of the bridge and the AASHTO multiple presence factor.

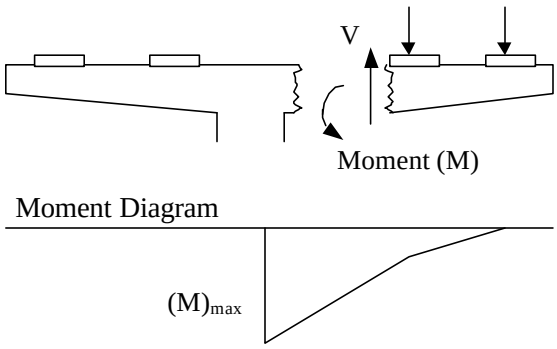


Figure 5.7 Maximum Pier Cap Moment in a Single Column Pier

5.8 Network Training Set

The error backpropagation techniques rely on a training set of input and output parameters to facilitate the training process. For this work, the training process was conducted using a database of existing pier structures. Since the training pairs were created from existing bridge piers, the size of the training set was limited by the number of piers that could be analyzed. That is, unlike some academic neural network problems, where training sets can be readily generated, this implementation relies on a limited database of training data. This database was compiled using data gathered from existing highway pier structures throughout the State of Florida in cooperation with the Florida Department of Transportation (FDOT). In time, additional training pairs can be added to the training set as more pier designs are completed.

The networks described in this chapter were trained using 28 training pairs with a validation set of 2 pairs. The training pairs represented a wide variety of pier designs to allow the networks to generalize a prediction to a new problem. The validation pairs were not used in the training process and represent new pier structures. It is of course possible for the network to encounter a new problem outside the training scope, thereby producing

questionable results. It is the author's observation that there is a certain standardization in highway bridge pier design so the creation of a pier outside of the training scope is not very likely. Such a unique pier would most likely warrant additional study due to its departure from standard designs. Of course, this unique type of pier could be added to the training set to increase the scope of the network functionality.

5.9 Implementation of Networks

The success of a neural network is greatly dependent on the process used to train the network. The implementation of feed forward neural networks in this work relies on error backpropagation training to adjust the weighted connections to match a specified target response. There are however, many variations and modifications to the standard backpropagation method, as originally described by Rumelhart, Hinton, and Williams (1986). Most modifications have been made to increase the speed of training and in some cases reduce the computer overhead during the solution process. This work implements a variety of solution techniques to ensure the accuracy and robustness of the network prediction.

Two neural network software packages were used for the creation, training, and validation of the networks. Each network was initially created and trained using the NetSim program (Consolazio 1995). The author chose to use this program due to the availability of the source code and the relative ease of use. The networks were originally trained using NetSim with a first-order steepest descent technique with both momentum and adaptive learning to reduce the training time. Unfortunately, the steepest descent usually does not produce the most direct solution to the problem. To overcome this limitation

within the NetSim program, the MatLab (Mathtools 1999) software package was then utilized to investigate other possible second-order solution techniques. The Neural Network Toolbox within MatLab features both gradient descent and a variety of second order optimization techniques to rapidly achieve an optimal solution. Training and validation statistics were generated with MatLab for comparison to the NetSim program. After comparing the initial network prediction results, the NetSim program was then modified by the author to implement a neural network pruning algorithm to optimize the topology of the networks. The network pruning procedure will be discussed in greater detail in Chapter 7.

5.10 Network Results and Validation

The output response of the neural networks must be verified before proceeding to any application of live load generation. For this implementation of feed forward networks, the network must complete the training phase before the presentation of any new problems. Recall that to complete the training process, the difference in the network output response and target response must fall below a certain tolerance. This training phase is critical since the network *must* learn the relationship among the training data before even attempting to predict the solution to an unforeseen problem. Once the network has been trained successfully, the validation phase can begin where the output response of the trained network is compared to a previously determined solution. In a manner similar to the training process, the validation error is defined in terms of the difference between the actual network response and the target response. A large validation error implies that the network is incapable of generalizing a solution to a new problem. In contrast, a small

error in validation provides some reassurance that the network can generalize well to produce a reasonable solution to a new problem.

5.10.1 Network Results Using NetSim

In this investigation, the training and validation phases were undertaken for each of the four networks created to determine the load positioning for the maximum force effects. The results that follow in Figures 5.8-5.11 are presented in terms of the network training and validation errors for single column piers. The results were generated using the NetSim program with error backpropagation training and sigmoid activation functions at each computational network layer. Momentum and adaptive learning were incorporated to expedite the training process. In each case, the variation of average root mean square (RMS) error for training and validation are plotted together against the number of training cycles (epochs). For this application, the average RMS error is defined as the sum of the RMS errors over all training sets divided by the number of training sets. The values vary from 1 (100% error) to 0 (0% error). The relatively small errors in training and validation for each of the four cases presented attest to the success of the networks in predicting the correct live load positioning. These results are somewhat expected given the simplicity of the single column pier behavior. The results do however serve as a valuable verification point before proceeding to the more complex multiple column pier networks.

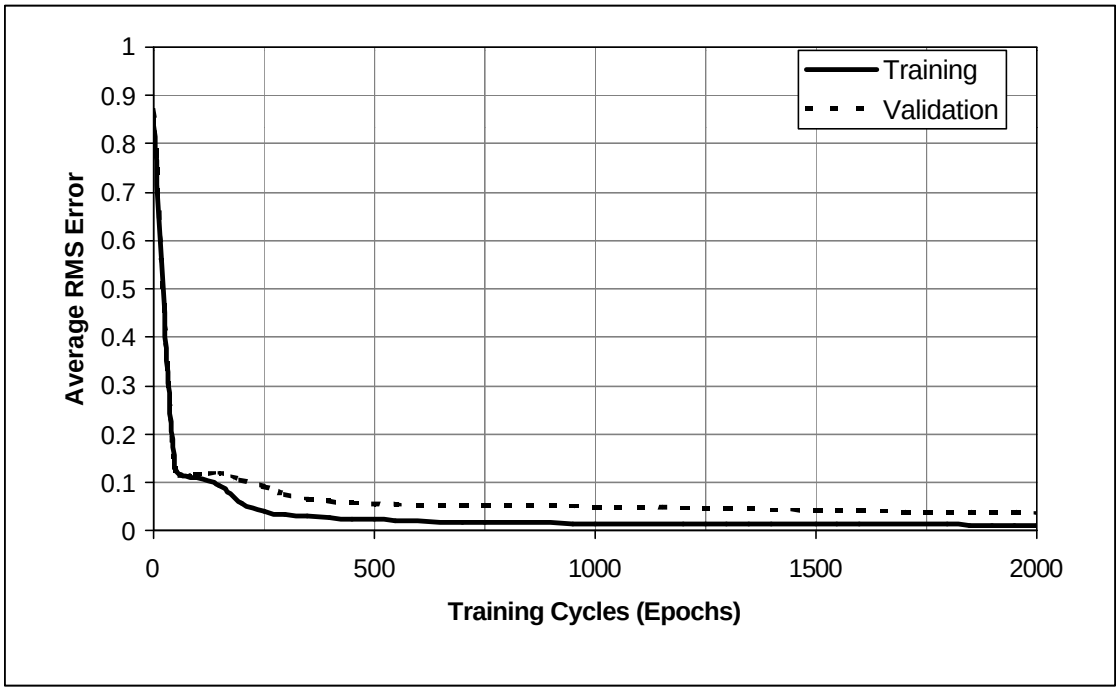


Figure 5.8 Pile Force Combination Network Error (Single Column Pier)

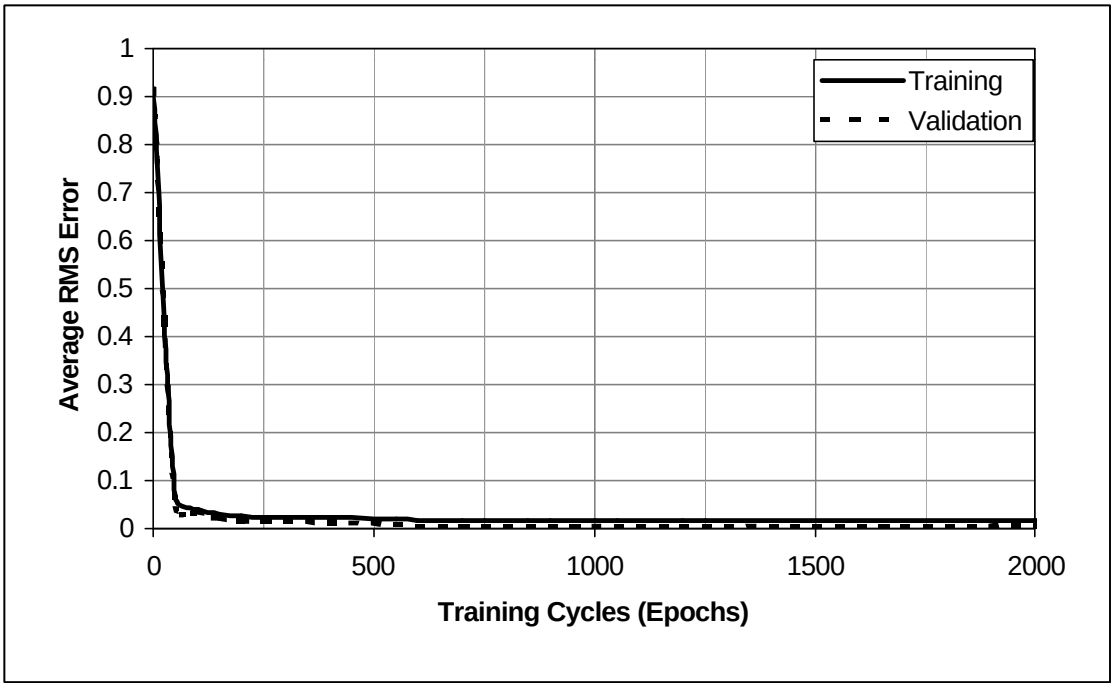


Figure 5.9 Column Force Combination Network Error (Single Column Pier)

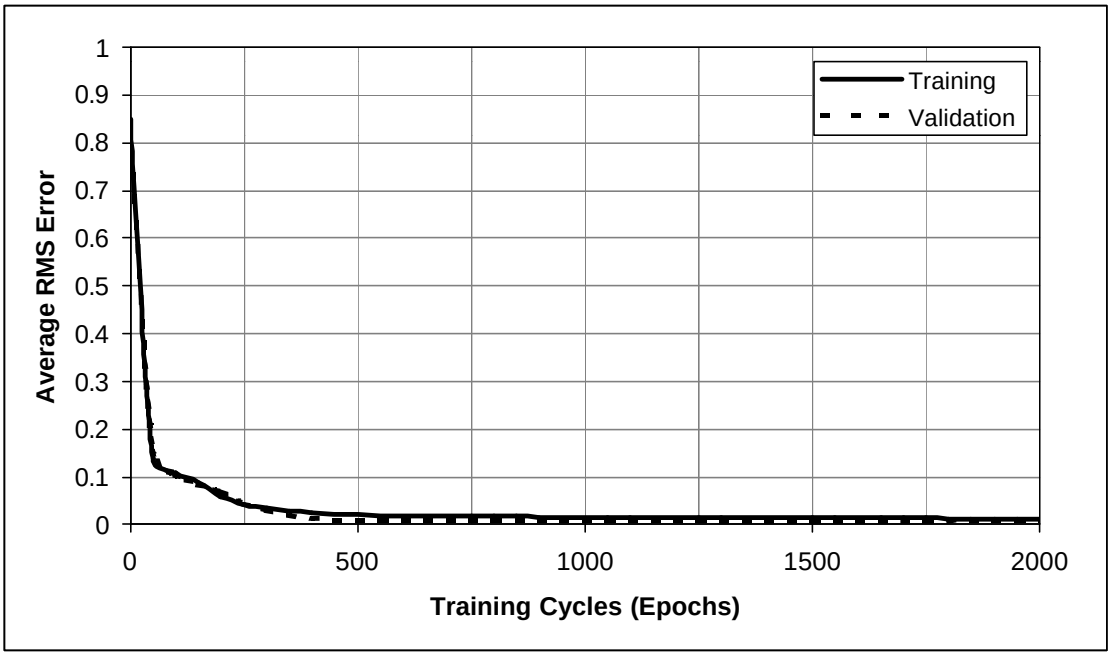


Figure 5.10 Pier Cap Shear Force Network Error (Single Column Pier)

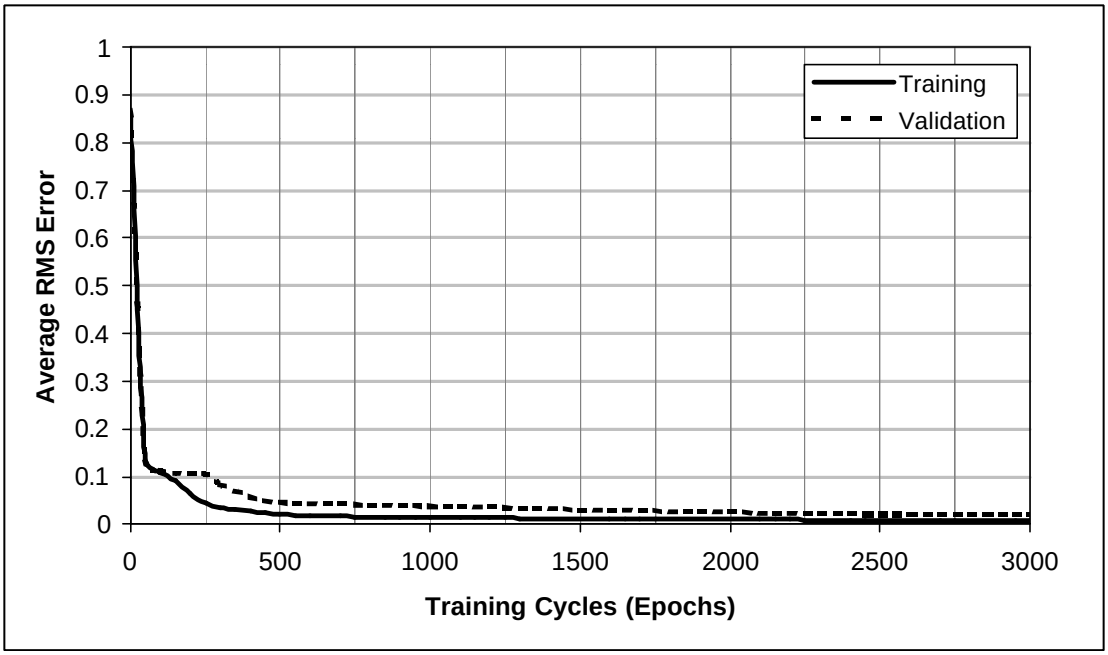


Figure 5.11 Pier Cap Bending Moment Network Error (Single Column Pier)

The plots of the variation of training error during the training process reveals the capability of the network to understand the relationship between the input and output parameters of the training set. For a well trained data set, the training error should decrease towards zero as the weighted connections are adjusted to their optimum values. This is clearly the case for each of the networks created to identify the maximum force effects. The plots in Figure 5.8 to Figure 5.11 show the rapid decrease in training error as the network encodes the relationship between input and output. Equally important, the validation error rapidly decreases as well, demonstrating that the network can predict a successful solution to the newly posed problems in the validation set. In most cases the validation error did begin to increase near the end of the training process, indicating that the network was beginning to overfit the training data. This increase in validation error was however very subtle.

The numerical errors for the training and validation phases are summarized in Table 5.2. For the four cases, the validation set produced a maximum average root mean square (RMS) error of 1.2%. This validation error is very reasonable given the variability of the soil conditions and pile configurations. The maximum validation error did not exceed 13.5% for the four networks. The maximum training error tolerance was 5%.

Table 5.2 Network Error Statistics for Single Column Piers

Network		Training		Validation	
	Number of Epochs	Average RMS Error	Maximum Error	Average RMS Error	Maximum Error
Pile Force	2080	0.01170	0.05000	0.03651	0.13540
Column Force	7990	0.00650	0.05000	0.00208	0.00475
Pier Cap Shear	2890	0.00892	0.05000	0.00631	0.01637
Pier Cap Moment	4650	0.00730	0.05000	0.01676	0.05570

5.10.2 Network Results Using MatLab

After achieving good results with the initial network training, different training algorithms were employed to check the validity of the training process. At this point, the training data was transferred to the MatLab software environment for additional testing. The first-order training results for standard backpropagation and backpropagation with momentum and adaptive learning were first repeated for comparison to NetSim. Then, several second-order backpropagation training algorithms were selected to further test the training process. For this study, three second-order algorithms were employed: conjugate gradients, Broyden-Fletcher-Goldfarb-Shanno (BFGS) and Levenberg-Marquardt (LM). The results of the different network training efforts are presented in Tables 5.3-5.6. Note that the MatLab training and validation results are given in terms of the RMS error, which is not the same as the average RMS error and maximum error from the NetSim results. The discrepancy exists because NetSim uses the maximum training error for a convergence tolerance and MatLab uses the RMS training error to test for convergence. There is currently no general consensus on the proper method of presenting network training and validation error so either method is acceptable.

Table 5.3 Pile Force Combination Training Results (Single Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	880	0.0498	0.0466
Momentum and Adaptive Learning	77	0.0466	0.0233
Conjugate Gradient	26	0.0420	0.0115
BFGS	22	0.0309	0.0292
Levenberg-Marquardt	6	0.0215	0.0278

Table 5.4 Column Force Combination Training Results (Single Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	458	0.0499	0.0287
Momentum and Adaptive Learning	51	0.0485	0.0209
Conjugate Gradient	10	0.0412	0.0415
BFGS	9	0.0251	0.0108
Levenberg-Marquardt	5	0.0155	0.0173

Table 5.5 Pier Cap Shear Training Results (Single Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	517	0.0499	0.0223
Momentum and Adaptive Learning	107	0.0257	0.0218
Conjugate Gradient	12	0.0236	0.0162
BFGS	9	0.0480	0.0203
Levenberg-Marquardt	4	0.0085	0.0114

Table 5.6 Pier Cap Moment Training Results (Single Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	151	0.0490	0.0694
Momentum and Adaptive Learning	82	0.0405	0.0407
Conjugate Gradient	14	0.0154	0.0141
BFGS	7	0.0426	0.0213
Levenberg-Marquardt	8	0.0385	0.0470

The results presented in Tables 5.3-5.6 verify the success of the additional network training algorithms. Each of the training methods converged quickly to the specified RMS error tolerance of 5%. The methods show some difference in the number of epochs needed to reach the converged result. The second-order training algorithm provided a converged result in the fewest number of epochs however the methods did require more computer overhead to operate. In contrast, the standard backpropagation algorithm, using simple gradient descent, required the most epochs to reach a converged result. It should be noted

that although the second-order algorithms do reach a converged solution in fewer epochs, the time needed to complete each epoch of training is significant. Therefore, it is possible for a simple gradient training to reach a converged result faster than a second-order techniques even though more training epochs are required.

5.11 Final Live Load Positioning

The small training and validation errors indicate the success of the networks in positioning the live loads across the width of the bridge. Before concluding this investigation though, it is important to show the predicted loads positions compared to the correct load positions from the validation set. The difference in load positioning can be best shown pictorially. Such an illustration removes the obscurity in the numerical value of the error and converts the difference into an observable conclusion. An illustration of the load positioning to produce the maximum pile force combination is shown in Figure 5.12 for the first problem in the validation set. The average RMS error in the network prediction was 4.3%.

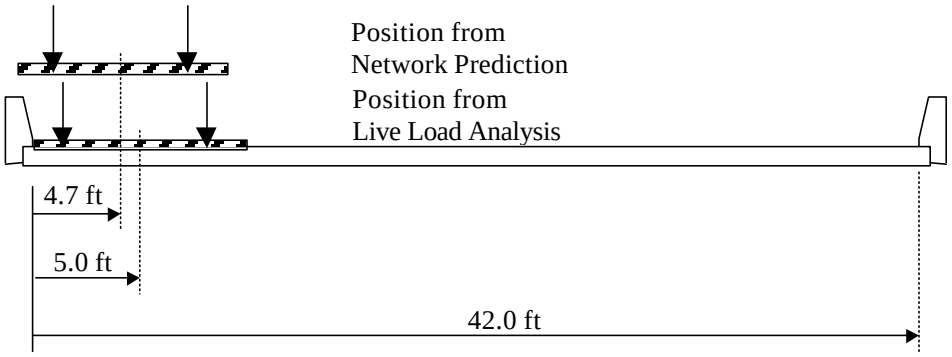


Figure 5.12 Final Load Positioning (Single Column Pier)

5.12 Comparison of Predicted and Actual Design Loads

Once the loads are positioned, the bridge structure can be analyzed to determine the girder reactions on the bearing pads. These bearing pad reactions can then be applied to the pier model for the pier analysis. For a final comparison, these reaction forces can be matched up to the actual forces calculated by the engineer for the pier design. This comparison is considered for one of the piers in the validation set shown in Figure 5.13.

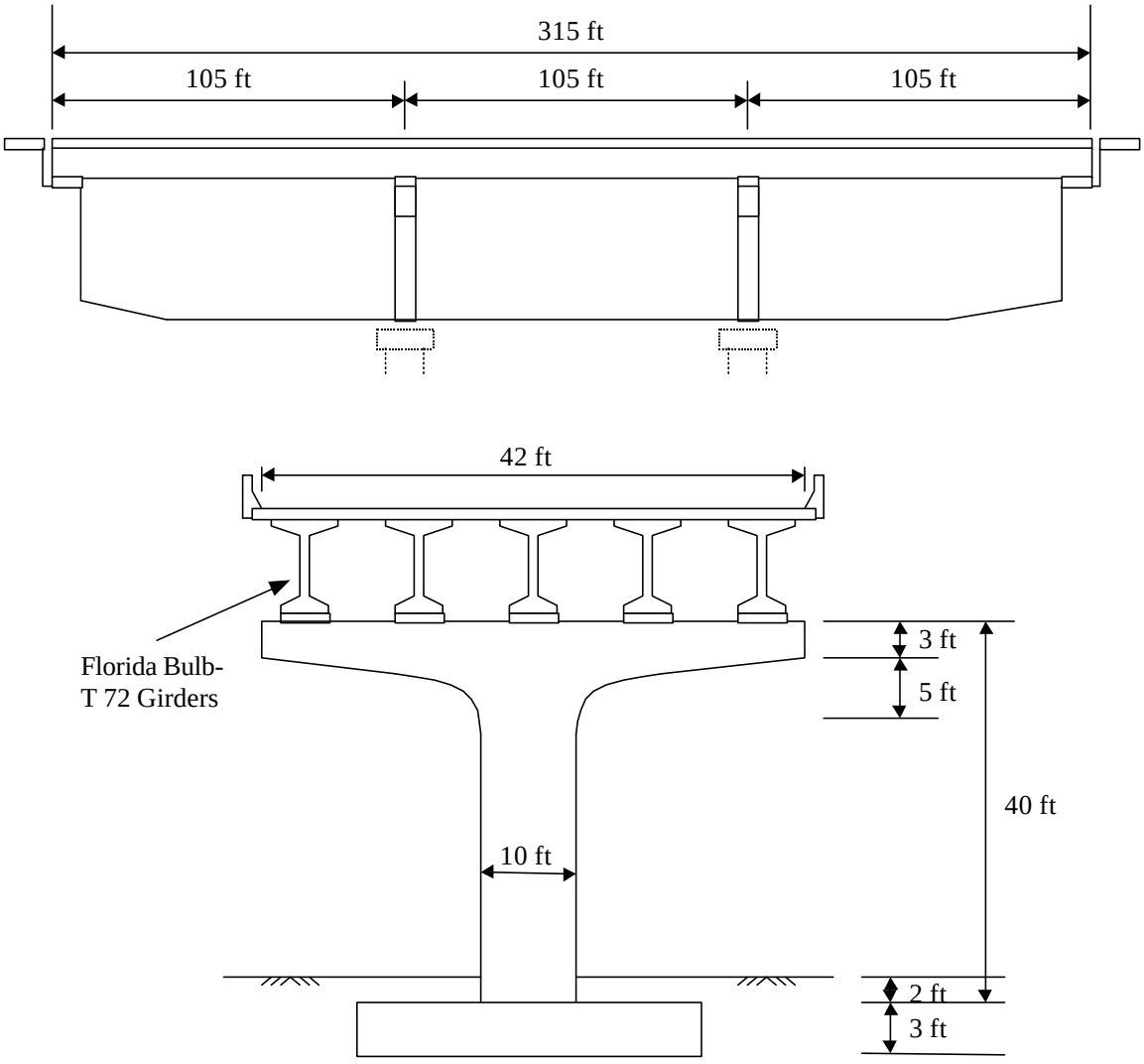


Figure 5.13 Single Column Pier Configuration used for Load Comparison

The girder reactions and resulting force effects from both the predicted load position and actual design calculations are compared in Figures 5.14-17. The results show that the live load reactions from the network positioning are slightly larger than the design calculations and therefore produced slightly larger force effects for all four cases. This difference could be viewed as a conservative approach to the live load generation.

The first load comparison shows the maximum pile force combination obtained from the live loading. For the chosen validation problem, the maximum force effect occurred equally in the two center piles, highlighted in Figure 5.14. The worst force effects are calculated using the failure ratio since the piles were allowed to exhibit nonlinear behavior. The maximum failure ratios for the two center piles are shown with a plot of the failure ratio along the length of the pile. The finite element model of the pile is also shown for completeness. The solid line indicates the failure ratio from the neural network predicted loads and the dotted line represents the failure ratio from the actual design loads. As shown, the resulting pile failure ratios are similar.

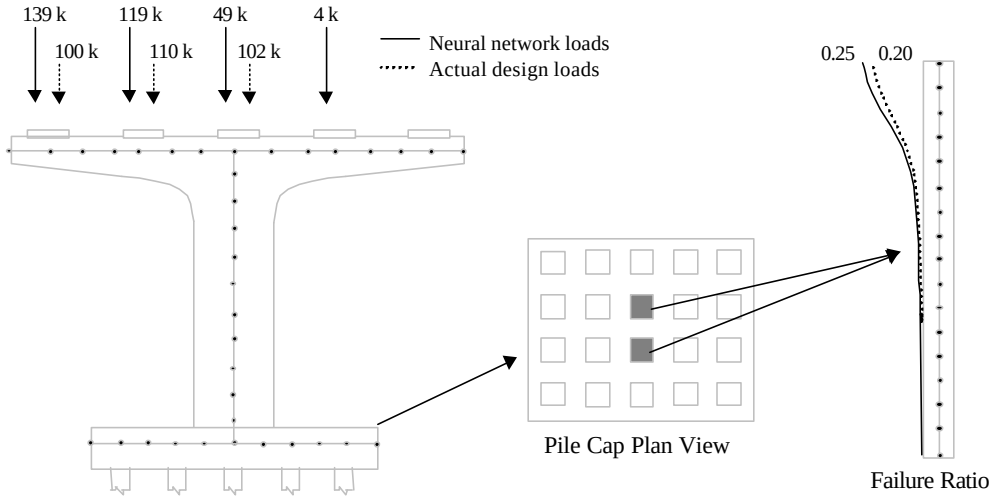


Figure 5.14 Pile Force Combination Force Comparison (Single Column Pier)

The second load comparison shows the maximum column force combination obtained from the live loading. For the single column validation problem, the maximum force effect had to occur in the only column. The worst force effects are calculated using the SRSS combination since the pier structure was assigned a linear material behavior. The maximum SRSS values for the column are constant for along the column and are shown in Figure 5.15. The pier outline and finite element model of the pier are shown for completeness. Again, the column force combination results from the neural network predicted loads and the actual design loads are similar.

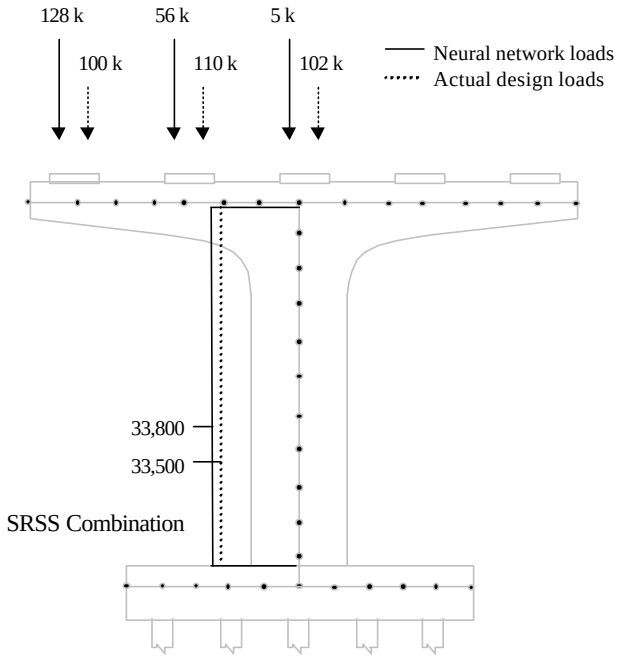


Figure 5.15 Column Force Combination Force Comparison (Single Column Pier)

The third load comparison shows the maximum shear force in the pier cap obtained from the live loading. The worst shear force effects were determined directly from the finite element analysis of the pier and are plotted along with the design shear forces in Figure 5.16. As shown, the variation in the results from the neural network predicted loads

and the actual design were within 10%. Equally important, the neural network predicted the loading of two lanes, which matches the design loading procedure.

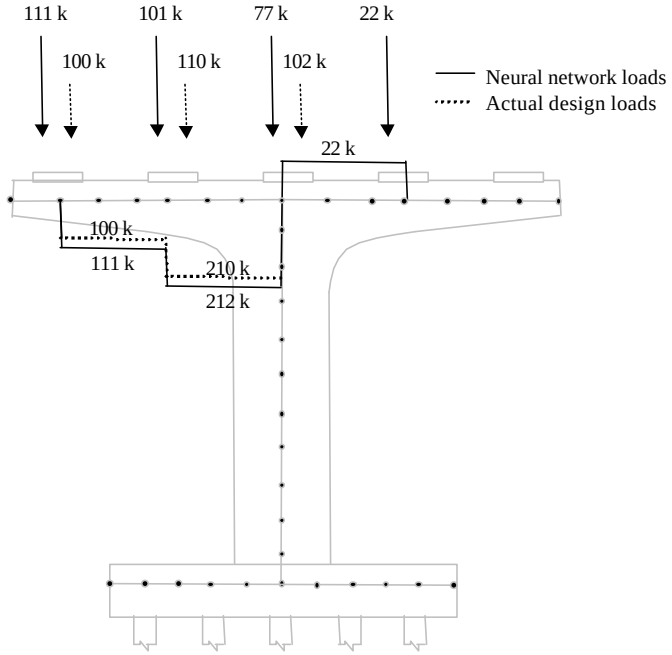


Figure 5.16 Pier Cap Shear Force Comparison (Single Column Pier)

The final load comparison shows the maximum bending moment in the pier cap obtained from the live loading. The worst bending moment force effects were determined directly from the finite element analysis of the pier and are plotted along with the design bending moments in Figure 5.17. As shown, the variation in the results from the neural network predicted loads and the actual design were noticeable along the pier cap, but the results were very similar at the cantilever base. The bending moments varied by 1% at the base of the cantilevered pier cap. Equally important, the neural network predicted the loading of one lane, which matches the design loading procedure.

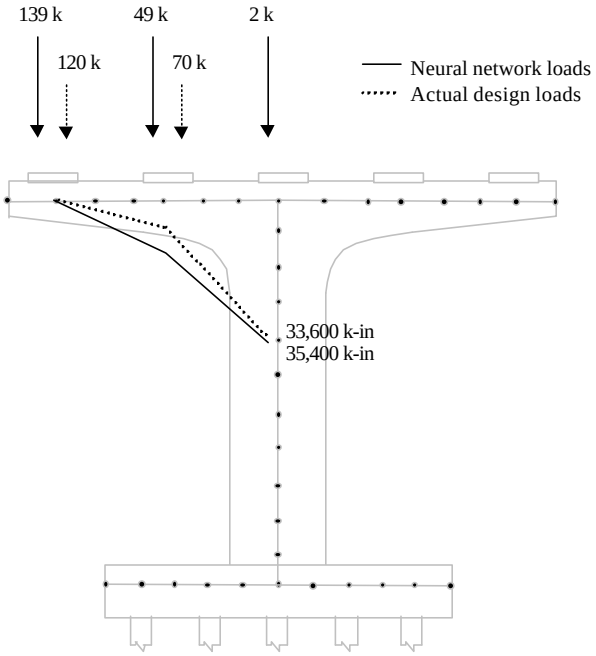


Figure 5.17 Pier Cap Bending Moment Comparison (Single Column Pier)

CHAPTER 6
POSITIONING LIVE LOADS ON MULTIPLE COLUMN PIERS

6.1 Introduction

The success of the network positioning of live loads on single column piers now leads to a more complex application of neural networks. For the single column pier problems, the network results were directly obtainable and easily verified by hand calculations. For multiple column piers though, the solution is not so straightforward. The multiple column pier configuration involves the interaction of many design parameters. This incorporation of more parameters is a direct result of the need to support and distribute the live loads from wider bridges. Coupled with the increase in design parameters is the additional autonomy in configuring of the pier structure with any number of pier columns and pile layouts.

The work presented in this chapter addresses the positioning of the live loads on multiple column piers. The procedure presented is a direct extension of the network formulations for single column piers with the inclusion of more input parameters to classify these more complicated problems. In a similar fashion to the single column problems, four neural networks are again created to automate the load positioning process. These four neural networks are implemented to predict the critical live load positioning to achieve the same maximum force effects in the multiple column piers.

6.2 Encoding Structural Behavior

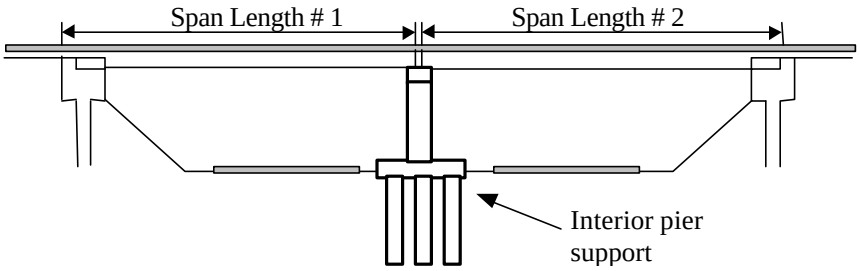
For networks trained by backpropagation, the encoding of structural behavior is achieved through the evaluation of input and output parameter sets. In this application, the input and output parameters must be sufficient to describe the multiple column pier configuration and encode the proper structural response under live loading. This encoding for multiple column piers is complicated in that many different pier configurations must be mapped to the correct structural behavior. It should be made clear at this point that any given pier can have two or more columns, supported by different pile configurations along with any number of bearing pads along the pier cap. This variability in pier configuration and corresponding structural response can be encoded by providing a sufficient amount of training problems to represent all feasible types of pier designs. Since it is unlikely that every type of pier can be included in the training set, the well-trained network must rely on its generalization capabilities to produce a reasonable response to the unforeseen problem.

6.3 Dimensionless Network Parameters

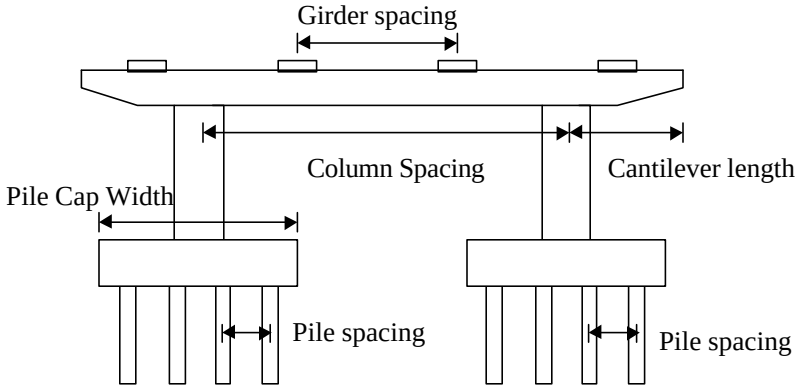
The success of the neural networks for single column piers was predicated on the identification of the necessary dimensionless parameters. The same predictive success can also be expected for the multiple column piers with the correct identification of the influential parameters that describe the pier system. Since the four maximum force effects are achieved through different structural behaviors, the group of dimensionless input parameters will be implemented in four different networks. The corresponding output parameters will again describe the positions of the live loads across the width of the bridge.

6.3.1 Input Parameters

Since the networks presented in this chapter are created for multiple column piers, information about the pier column configuration is necessary to classify the pier. Of additional importance for interior piers are the bridge span lengths on either side of the pier, the length of the cantilevered pier cap, the width of the bridge, the spacing of the girders, and the pile spacing. These quantities, illustrated for the bridge in Figure 6.1, can be arranged into dimensionless input parameters using the Buckingham Π Theorem.



(a) Complete bridge system



(b) Pier configuration

Figure 6.1 Multiple Column Pier Configuration Variables

In following the procedure for single column piers, this work considers the same four maximum force effects for the bridge piers. Again, four separate networks are created to identify each of the maximum force effects. For simplicity, each of the four networks

uses the same dimensionless input parameters to describe the bridge and pier configurations. In this multiple column pier investigation, the first input parameter represents the number of design lanes for the bridge. This parameter is already dimensionless and essential for the network to classify the live loading conditions. The remaining five input parameters are created for the networks based on the same dimensional analysis used for single column piers in Section 5.4.1. These input parameters should be sufficient to describe the geometry of the bridge and pier system and the effect of the soil-pile interaction. The input parameters are shown later with the complete network configuration in Section 6.6.

6.3.2 Output Parameters

Again, for this investigation, output parameters represented the correct placement of the live load across the width of the bridge to achieve the maximum forces in the pier. The design truck and lane load were again positioned independently within a lane in any of the four lanes, thereby requiring eight output neurons to represent the output response. In a manner similar to the single column pier networks, the output response for each neuron was normalized. This normalization was achieved by dividing the dimensioned placement of the center of the load by the total bridge width. For unoccupied lanes, the output response parameters are set to zero.

6.4 Maximum Force Combinations

The objective of the live load placement on the pier is to achieve the maximum force effects in a particular element of interest in the pier model. For this work, the maximum force effects are again defined as the maximum force combination in the piles

and pier columns along with the maximum shear and bending moment in the pier cap. For multiple column piers, the pier cap now includes the beam spanning between the pier columns as well as the cantilever ends.

The determination of the maximum force combination depends on the material behavior assumption in the analysis. For nonlinear analysis or linear analysis with cross-sectional properties, the failure ratio will be used to determine the critical load positions. For pure linear elastic analyses, the combination of forces using the SRSS method outline in Chapter 3 is utilized to determine the critical load positions. Since the Florida Pier program provides a database of standard pile types with nonlinear properties, most of the pier models used in the training set incorporated nonlinear pile behavior. In contrast, most of the pier structures are modeled using the linear elastic properties with the assumption at the pier structure will remain linear under live loading. Therefore, most of the piers in the training set contained a combination of linear and nonlinear components. Although the procedure for determining the extreme force effect is different for the linear and nonlinear components, the worst load position cases will still be identified using either method.

6.5 Critical Load Positions and Load Symmetry

In this investigation, neural networks are formulated to predict the critical load placement across the width of the bridge. For multiple column piers, this task is complicated given the configuration of the pier systems. The complexity is due to the fact that multiple column piers typically support bridges with three or more lanes of traffic. This additional lane capacity creates more possible load combinations across the width of the bridge. While this additional lane capacity was possible for large single column piers,

most hammerhead piers support two or three lanes of traffic. This inclusion of more traffic lanes makes the proper determination of the correct load position a critical issue. That is, any incorrect identification of the critical load positions could lead to the failure of the network training process or more likely a misinterpretation of the structural behavior. In this regard, certain assumptions concerning the positioning of the loading were made to ensure that a consistent method was followed to determine the live load placement.

The study of live load generation for the single column piers revealed that many of the loading patterns were symmetric across the width of the bridge. That is, a load placed to the left edge of the bridge was also placed at the far right end of the bridge at some time during the live load generation. This is now particularly true for the loading of multiple traffic lanes on the multiple column piers. Since the results from symmetric load patterns are virtually identical, the critical load pattern was always taken as the leftmost symmetric loading pattern. If this condition were not enforced, the neural networks would have great difficulty in predicting the worst live load placement based on slight numerical differences.

6.6 Networks for Predicting Critical Load Positions

The objective of this investigation is to create a neural network to predict the load positions to produce the maximum force effects in multiple column piers. Prior work on the single column pier networks concluded that it was not practical to create a single network to correctly predict the load placement to produce all maximum force effects. Instead, four separate networks were created to incorporate the uniqueness of the four force effects.

An initial network configuration was created to study the load positioning on multiple column piers. The initial network shown in Figure 6.2 used six input units to sufficiently describe the problem and eight output neurons to accommodate a maximum of four lanes of traffic. One hidden layer of ten neurons was implemented to encode the structural behavior. This number of hidden processing units will later be reduced through a network pruning process to optimize the network topology. Both the hidden layer and output layer used sigmoid transfer functions to process the signals from the weighted connections.

The four networks were created to identify the loading to produce the four maximum force effects needed for the design of the pier. Each force effect is explained in detail below.

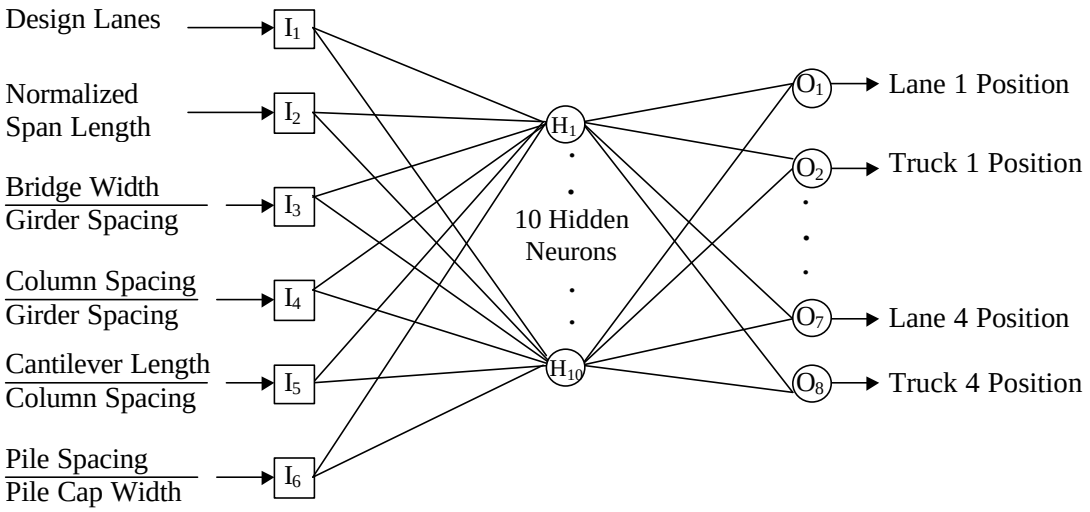


Figure 6.2 Network Configuration for Multiple Column Pier

6.6.1 Maximum Pile Force Combination

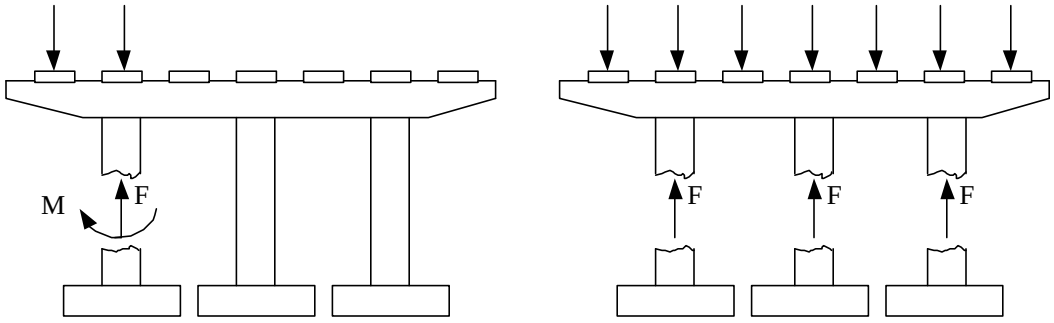
The design of a pile system for a multiple column pier requires the understanding of the load transfer through each of the pier columns. In most foundation designs, the pile and

pile cap configuration depends on the spacing of the pier columns. That is, for a relatively small column spacing, the pile cap can be designed as a continuous block of concrete than spans from column to column. However, for a large column spacing, most foundation designs provide a separate pile cap for each pier column. It is therefore quite likely that the forces in the pile will vary depending on the pile cap configuration. In either case, the maximum force effect is identified as the maximum load combination in a single pile under axial load and bending moment. This combination of axial force and bending moment is also likely to be influenced by the type of soil and the characteristics of the pile. Since there can be great variability in soil properties, pile configurations, and pile parameters, the results can vary significantly from pier to pier. With a sufficient number of training problems and the necessary input parameters though, the error in network prediction should be relatively small.

6.6.2 Maximum Pier Column Force Combination

The determination of the maximum force combination in a multiple column pier is not as straightforward as the hammerhead pier structures. It is not immediately obvious which column will be subjected to the maximum force combination particularly given the variable number of lanes loaded for a given load case. For actual pier designs, this case is further complicated since the AASHTO code allows a reduction in live load as the number of loaded lane increases. The network must therefore consider all of the pier columns in the critical load position prediction. Some intuition can be used in the decision process. For example, loading the exterior lane of the hypothetical bridge pier in Figure 6.3 (a) could most likely produce the maximum bending moment (M) in the pier column, but not the maximum axial force (F). Likewise, loading all of the design lanes as shown in Figure 6.3

(b) would most likely produce the largest axial force, but a negligible bending moment in the column if the load is symmetrically placed above the column.



(a) Exterior Lane Loaded (b) All Lanes Loaded

Figure 6.3 Maximum Force Combination in a Multiple Column Pier

6.6.3 Maximum Pier Cap Shear Force

The maximum shear force in the pier cap must be identified in order to design the pier cap section. Since the live loads are applied as concentrated forces via the bearing pads, the shear in the pier cap directly depends on the positioning and magnitude of the load. For multiple column piers, the maximum shear force can occur in the cantilever portion of the pier cap or in a section between the pier columns. Due to the multiple possible locations of the maximum shear force, special consideration must be given to the spacing of the pier columns and bearing pads as well as the length of the cantilever section. One possible case for the maximum pier cap shear force is shown in Figure 6.4.

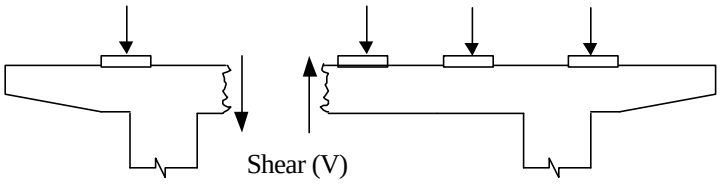


Figure 6.4 Maximum Pier Cap Shear in a Multiple Column Pier

6.6.4 Maximum Pier Cap Bending Moment

The determination of the maximum bending moment in the pier cap is important for flexural design of the pier cap section. Bending moment and shear are related and therefore the bending moment also directly depends on the positioning and magnitude of the load. Again, the spacing of the pier columns and bearing pads as well as the length of the cantilever section are very important. For multiple column piers, the maximum moment can occur in the cantilever portion of the pier cap or in a section between the columns. The latter case is shown for a maximum positive bending moment in the hypothetical pier shown in Figure 6.5. As a further complication, the maximum moment in the pier cap depends on the number of design lanes loaded along with the appropriate AASHTO multiple presence factor which modifies the live load. This uncertainty in loading must be encoded into the neural network in order to accurately predict the worst load positioning.

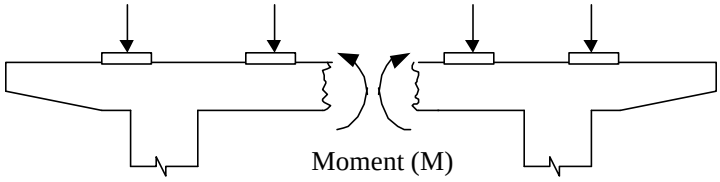


Figure 6.5 Maximum Pier Cap Moment in a Multiple Column Pier

6.7 Network Training Set

The training data using in this chapter was compiled using data gathered from existing highway pier structures throughout the State of Florida in cooperation with the Florida Department of Transportation (FDOT). The 47 training pairs used in this investigation represent a wide variety of pier designs to allow the networks to generalize a prediction to an unforeseen problem. Two validation pairs, not used in the training

process, were used to represent new pier problems for the network. The validation problems were selected to represent bridge systems that fall within the scope of the training. It is of course possible for the network to encounter a new problem outside the training scope, thereby producing questionable results. It is the author's observation that there is a certain standardization in highway bridge pier design so the creation of a pier outside of the training scope is not very likely. Any unusual piers that fall out of the training scope could be later added to the training set to increase the scope of the network functionality.

6.8 Implementation of Networks

The implementation of the multiple column pier neural networks follows the procedure established in Chapter 5 for single column piers. First, the NetSim program was implemented to create, train, and validate the initial network configurations. The training process followed first-order techniques to learn the relationship between the input parameters and the output load positions. Once the networks were functioning properly and satisfactory results were achieved, the networks were retrained using more advanced, second-order techniques. The retraining was conducted within the MatLab software environment using the Neural Network Toolbox. It is the author's opinion that this combination of solution techniques should ensure the accuracy and robustness of the network prediction.

As a final step, the network configurations were optimized. This investigation used a network pruning process to trim down the size of the network without significantly affecting the predictive capacity. The network pruning phase was again conducted with the NetSim program.

6.9 Network Results and Validation

The output response of the neural networks must be verified before proceeding to any application of live load generation. For this implementation of feed forward networks, the network must complete the training phase before the presentation of any new problems. Once the network has been trained successfully, the validation phase can begin where the output response of the trained network is compared to a previously determined solution. The validation phase provides a good indicator as to whether the network can generalize a solution to an unforeseen problem.

6.9.1 Network Results Using NetSim

In this investigation, the training and validation phases were undertaken for each of the four networks. The results that follow in Figures 6.6-6.9 are presented in terms of the network training and validation errors for multiple column piers. In each case, the variation of average root mean square (RMS) error for training and validation are plotted together against the number of training cycles. The error in training and validation were not as small as those for the single column pier case. This is expected since the given the complexity of the multiple column pier behavior.

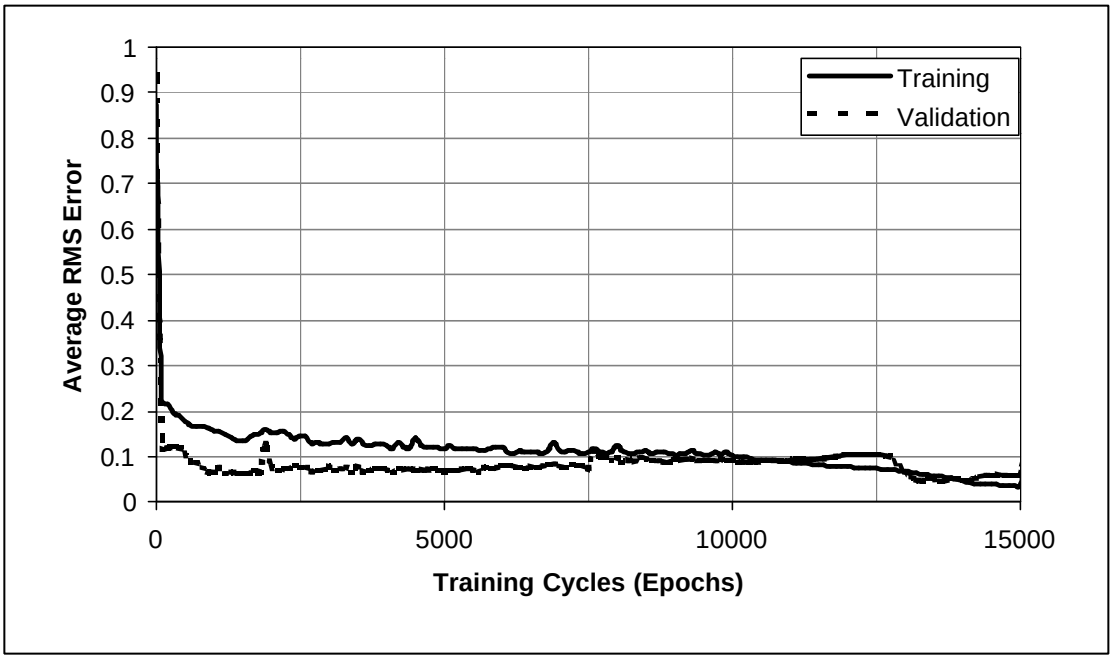


Figure 6.6 Pile Force Combination Network Error (Multiple Column Pier)

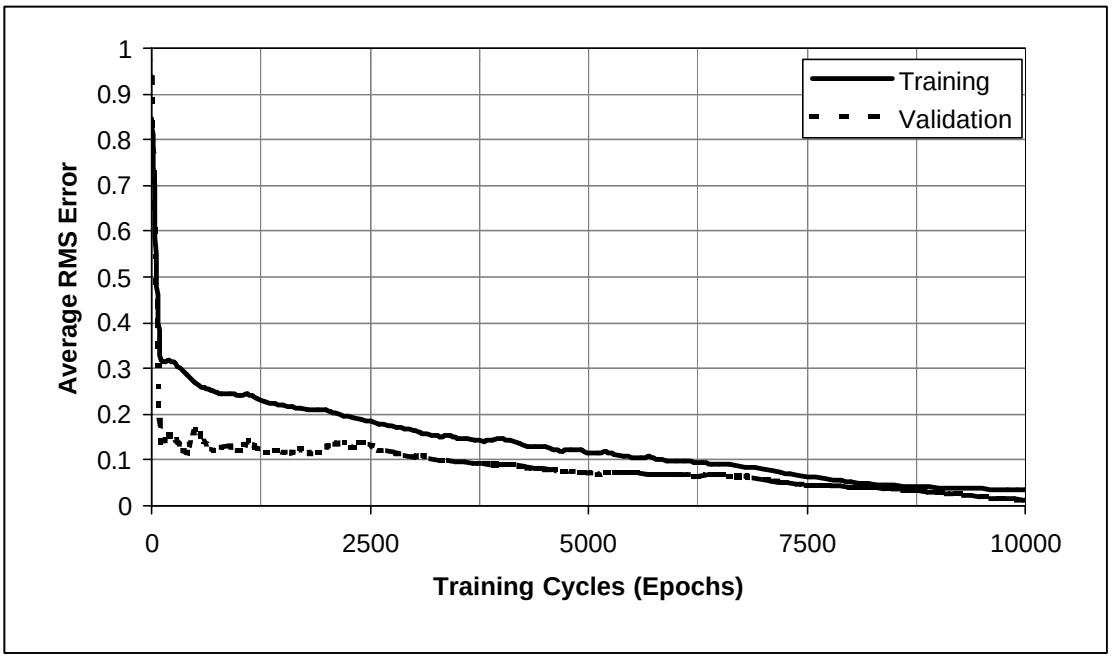


Figure 6.7 Column Force Network Error (Multiple Column Pier)

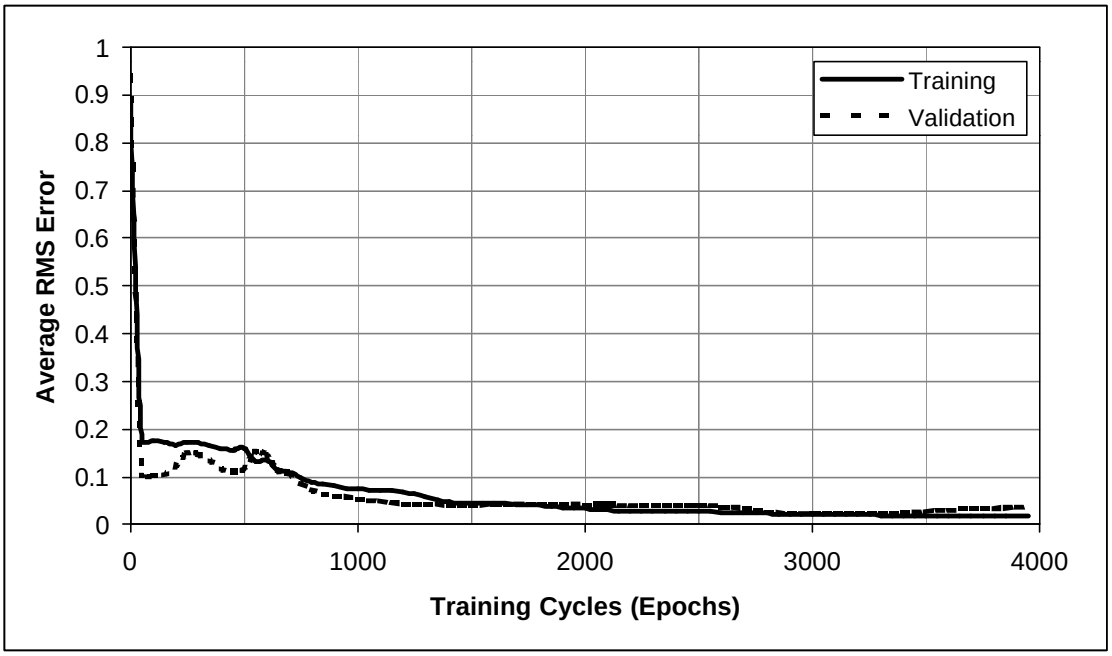


Figure 6.8 Pier Cap Shear Force Network Error (Multiple Column Pier)

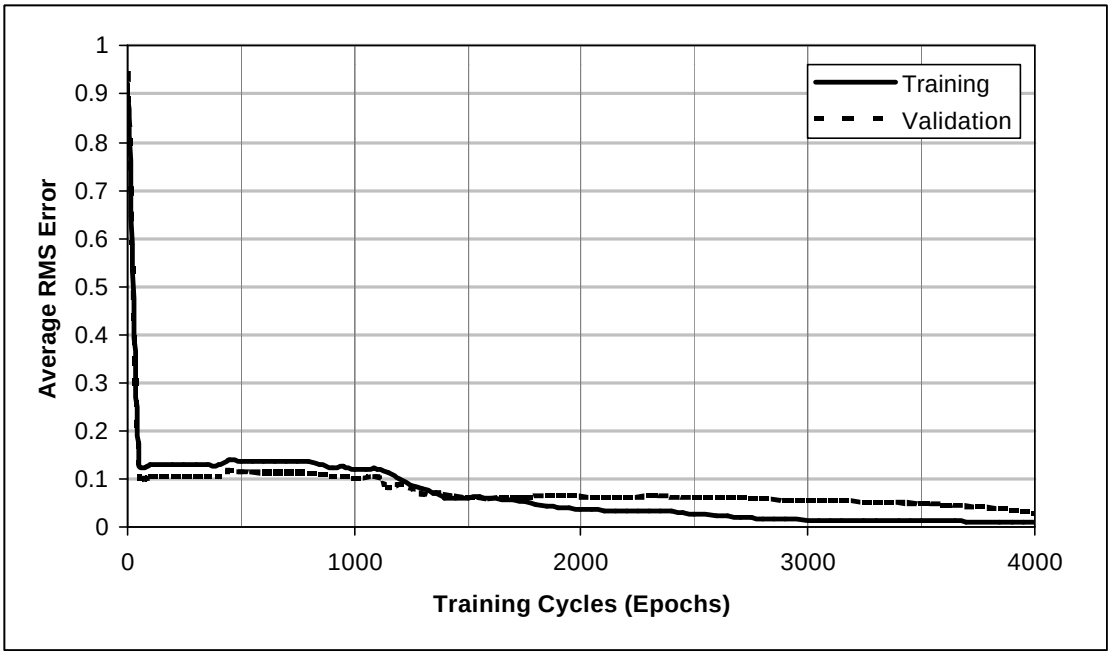


Figure 6.9 Pier Cap Moment Force Network Error (Multiple Column Pier)

The plots of the variation of training error during the training process reveals the capability of the network to understand the relationship between the input and output parameters of the training set. For a well trained data set, the training error should decrease to zero as the weighted connections are adjusted to their optimum values. This appears to be the trend for each of the networks created to identify the maximum force effects. The plots in Figure 6.6 to Figure 6.9 show the rapid decrease in training error as the network encodes the relationship between input and output. Equally important, the validation error rapidly decreases as well, demonstrating that the network can predict a successful solution to the newly posed problems in the validation set. Although slight increases in validation error were observable, the network training was allowed to continue until the error tolerance was reached.

The numerical errors for the training and validation phases are summarized in Table 6.1. The maximum training error was 60% for one training pair that greatly differed from the training scope. Despite this high error, the average RMS training error did not exceed 5%, which is certainly acceptable. For the four cases, the validation set produced a maximum average root mean square (RMS) error of 6%. This validation error is very reasonable given the variability of the soil conditions and pile configurations. The maximum validation error did not exceed 12% for the four networks.

Table 6.1 Network Error Statistics for Multiple Column Piers

		Training		Validation	
Network	Number of Epochs	Average RMS Error	Maximum Error	Average RMS Error	Maximum Error
Pile Force	15000	0.03647	0.33439	0.06146	0.12384
Column Force	13000	0.04751	0.59800	0.01517	0.04397
Pier Cap Shear	3950	0.01693	0.07008	0.03638	0.10275
Pier Cap Moment	4000	0.01166	0.15356	0.04491	0.09644

6.9.2 Network Results Using MatLab

Additional training and validation was undertaken in the MatLab environment to further verify the success of the network predictions. The training and validation phases were first repeated for standard backpropagation and backpropagation with momentum and adaptive learning for comparison to the results obtained from NetSim. Three second-order algorithms were then employed: conjugate gradients, Broyden-Fletcher-Goldfarb-Shanno (BFGS) and Levenberg-Marquardt (LM). The results of the different network training efforts are presented in Tables 6.2-6.5. The same 6x10x8 network topology was used for each training method employed with Matlab.

Table 6.2 Pile Force Combination Training Results (Multiple Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	801	0.0499	0.0363
Momentum and Adaptive Learning	69	0.0592	0.0192
Conjugate Gradient	14	0.0494	0.0229
BFGS	21	0.0495	0.0203
Levenberg-Marquardt	4	0.0412	0.0263

Table 6.3 Column Force Combination Training Results (Multiple Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	465	0.0873	0.0239
Momentum and Adaptive Learning	80	0.0925	0.0401
Conjugate Gradient	40	0.0855	0.0632
BFGS	34	0.0863	0.0364
Levenberg-Marquardt	8	0.0309	0.0357

Table 6.4 Pier Cap Shear Training Results (Multiple Column Pier)

Training Method	Number of Epochs For Training	RMS Error	
		Training	Validation
Standard Backpropagation	384	0.0469	0.0826
Momentum and Adaptive Learning	52	0.0478	0.0927
Conjugate Gradient	9	0.0395	0.0317
BFGS	32	0.0387	0.0370
Levenberg-Marquardt	7	0.0407	0.0822

Table 6.5 Pier Cap Moment Training Results (Multiple Column Pier)

Training Method	Number of Epochs For Training	Average RMS Error	
		Training	Validation
Standard Backpropagation	196	0.0498	0.0932
Momentum and Adaptive Learning	199	0.0320	0.0099
Conjugate Gradient	12	0.0448	0.0700
BFGS	12	0.0415	0.0383
Levenberg-Marquardt	9	0.0163	0.0098

The results presented in Tables 6.2-6.5 verify the overall success of the additional network training algorithms. In contrast to the single column pier results, some of the multiple column pier training techniques were not capable of reaching a 5% RMS error tolerance. For these cases, the training was stopped when the validation error began to increase after reaching a minimum value. After considering all methods, the RMS training and validation error did not exceed 9%. The methods did show some difference in the number of epochs needed to reach the converged result. The second-order training algorithms again reached a converged result in the fewer epochs than the first-order solution techniques. Of the second-order methods, the Levenberg-Marquardt algorithm reached a converged solution in the fewest number of epochs. It is important to reiterate that the second-order methods do require a lot of computer overhead to operate so training

large networks under these methods may not be feasible. For these cases, training may have to resort to simpler first-order solution techniques to reach a converged result.

6.10 Final Live Load Positioning

Before concluding this investigation, it is important to once again show the predicted loads positions compared to the correct load positions from the validation set. The final truck and lane positions can be plotted for each of the four network predictions. An illustration of the load positioning to produce the maximum column force combination is shown in Figure 6.10 for the first problem in the validation set. For this validation example, the average RMS error in the network prediction was 9%. Also, the lane and truck load positions within the leftmost design lane were nearly identical for the live load analysis and network prediction, so no distinction between the two load types is shown in Figure 6.10.

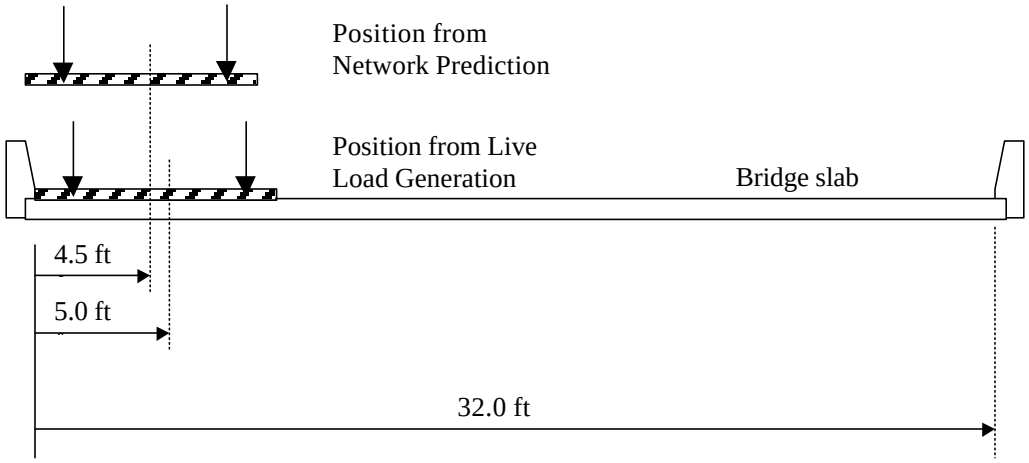


Figure 6.10 Final Load Positioning (Multiple Column Pier)

6.11 Comparison of Predicted and Actual Design Loads

Once the loads are positioned, the bridge structure can be analyzed to determine the girder reactions on the bearing pads. As demonstrated for single column piers, these bearing pad reactions can then be applied to the pier model for the subsequent pier analysis. For a final comparison, these reaction forces can be matched up to the actual loads calculated by the engineer for the pier design. A final comparison of girder reactions and the resulting force effects was conducted with the first pier model from the validation set. The geometry of the bridge superstructure and pier support are shown in Figure 6.11.

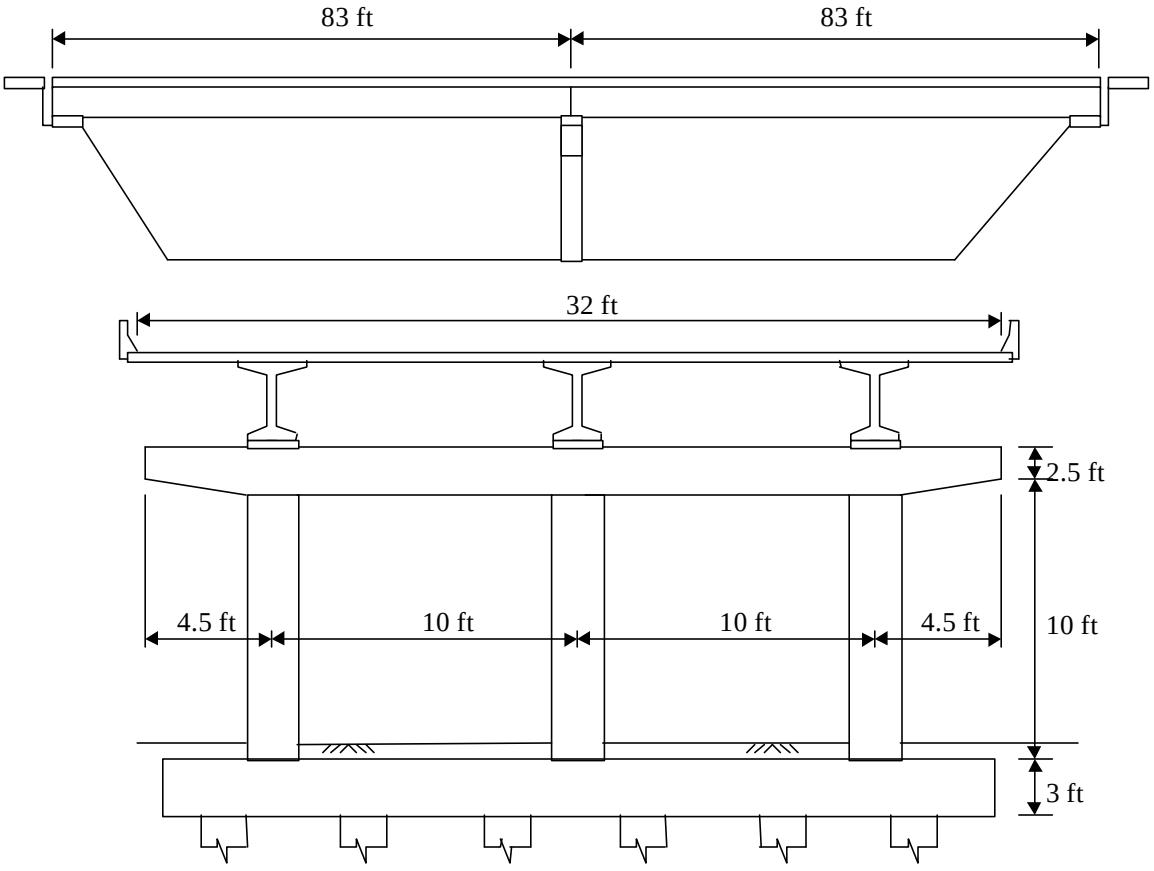


Figure 6.11 Multiple Column Pier Configuration used for Load Comparison

The girder reactions and resulting force effects from both the predicted load position and actual design calculations are compared in Figures 6.12-6.15. The girder reactions that produce the maximum force effects are shown next the design loads. For this pier, the design called for the application of the same girder reactions to every bearing pad under the assumption that loading three traffic lanes would govern the design. The resulting force effects from the neural network predicted loads are significantly different though from the loads from the design calculations. This discrepancy clearly demonstrates the uncertainty in live load placement on the piers and suggests that the live load placement should be studied carefully.

The first load comparison shows the maximum pile force combination obtained from the live loading. For the chosen validation problem, the maximum force effect occurred in the leftmost pile, highlighted in Figure 6.12. For this pier, the worst force effects are calculated using the SRSS combination since the piles were modeled for a linear material behavior. The maximum SRSS combinations for the leftmost piles are shown with a plot of the SRSS values along the length of the pile. The finite element model of the pile is also shown for completeness. The solid line indicates the SRSS values from the neural network predicted loads and the dotted line represents the SRSS values from the actual design loads. As shown, the resulting force effects are similar along the pile.

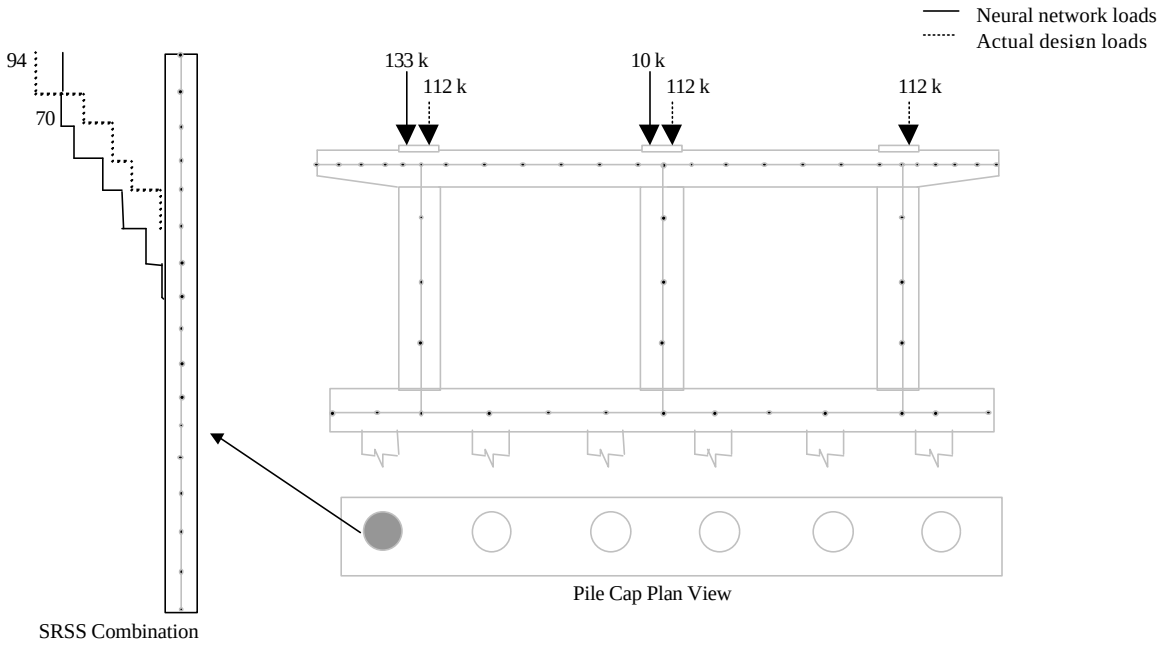


Figure 6.12 Pile Force Combination Force Comparison (Multiple Column Pier)

The second load comparison shows the maximum column force combination obtained from the live loading. For this case, the neural network predicted that the leftmost exterior lane should be loaded to produce the maximum force combination in the pier column. In contrast, the actual design loads were computed with the assumption of all three design lanes loaded. As shown in the Figure 6.13, the SRSS force combination results from the neural network predicted loads are greater than those corresponding to the actual design loads for the leftmost column. For this pier, the worst force effects are calculated using the SRSS combination since the pier structure was assigned a linear material behavior. The pier outline and finite element model of the pier are grayed and shown for completeness.

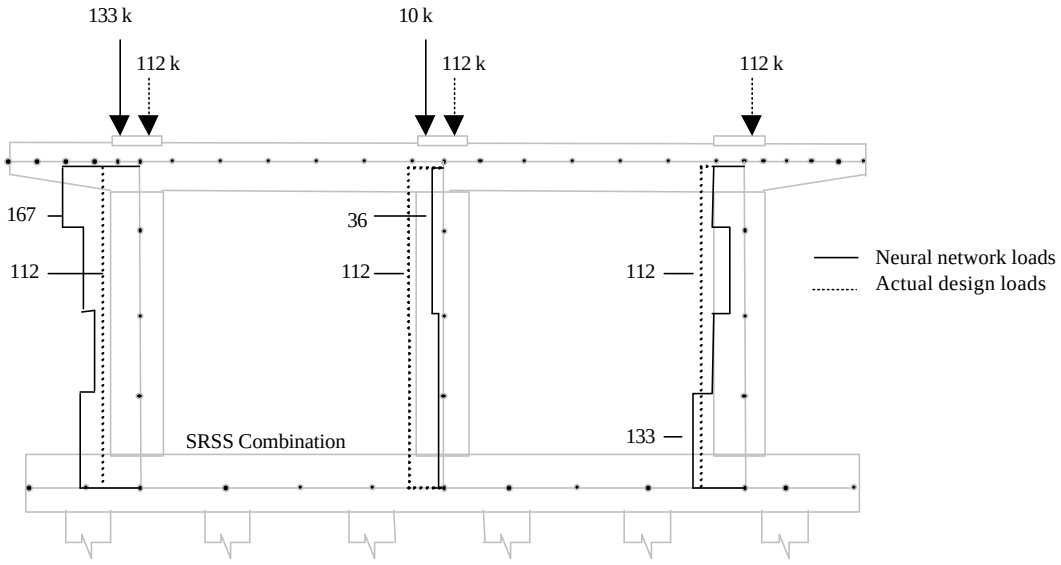


Figure 6.13 Column Force Combination Force Comparison (Multiple Column Pier)

The third load comparison shows the maximum shear force in the pier cap obtained from the live loading. The worst shear force effects were determined directly from the finite element analysis of the pier and are plotted along with the design shear forces in Figure 6.14. In this case the network predicted that the left and right exterior lanes should be loaded, while the actual design loaded all three design lanes. As shown, the variation in the results from the neural network predicted loads and the actual design are noticeable, though not significant for this pier. The pier cap shear force is negligible when all of the bearing pads are positioned directly over the pier columns.

The final load comparison shows the maximum bending moment in the pier cap obtained from the live loading. The worst bending moment force effects were determined directly from the finite element analysis of the pier and are plotted along with the design bending moments in Figure 6.15. As shown, the variation in the results from the neural network predicted loads and the actual design were noticeable along the pier cap, but again not significant for this bearing pad configuration.

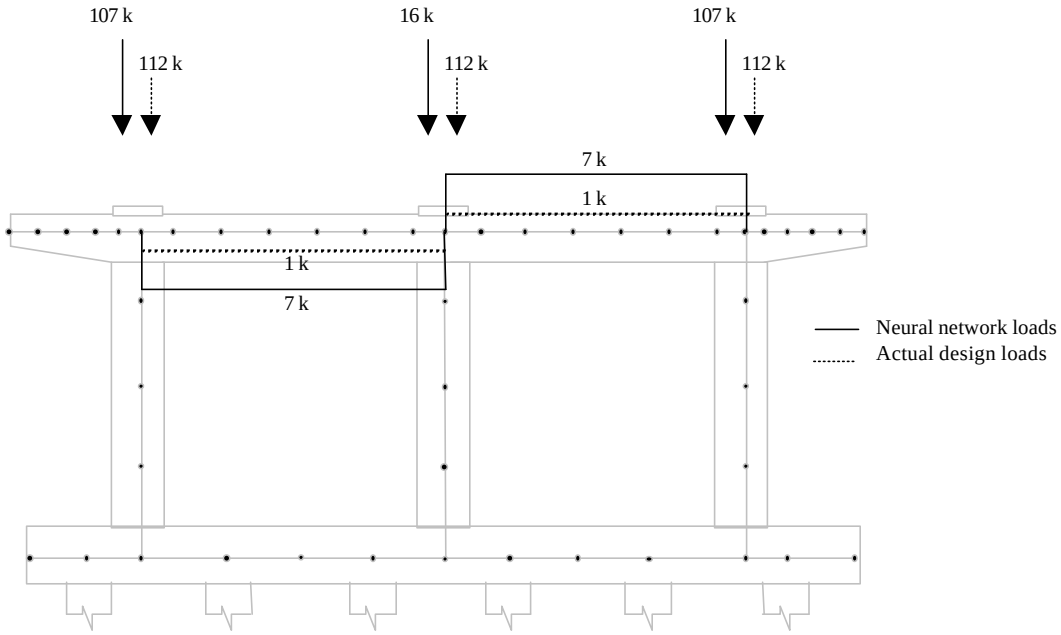


Figure 6.14 Pier Cap Shear Force Comparison (Multiple Column Pier)

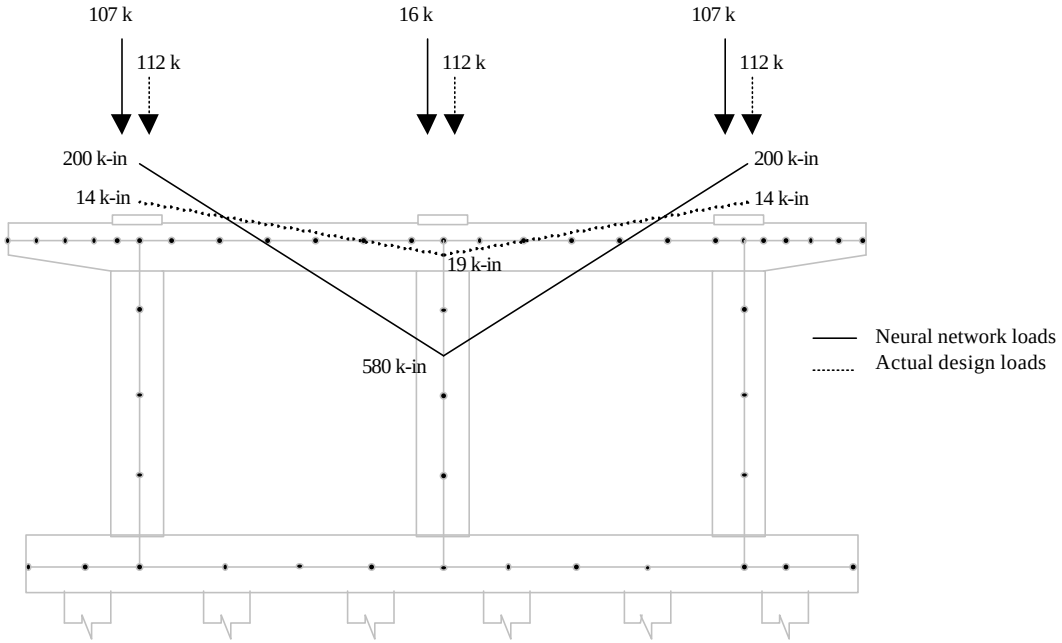


Figure 6.15 Pier Cap Bending Moment Comparison (Multiple Column Pier)

Now that the neural networks have been successfully trained and validated, the emphasis shifts to optimizing the network configurations. The next chapter introduces a

network pruning algorithm to optimize the size and subsequently increase the computational and fitting efficiency of the networks.

CHAPTER 7
NETWORK TOPOLOGY OPTIMIZATION

7.1 Introduction

The architecture or *topology* of a neural network has a significant influence on the performance of the network. As a general rule, as the size of the network increases so do the computational resources needed to solve the problem. Unfortunately, the size of the network is often dictated by the complexity of the problem. As the complexity increases, the network can be enlarged in several different ways. The network architecture can be changed by increasing the number of hidden layers as well as the number of neurons within each layer. This flexibility in network configuration can very easily result in a large number of possible network architectures. Given the many possible network configurations, methods have been devised to determine the ideal network configuration to solve the problem at hand.

The optimization of network architecture has most likely been a topic since the first inception of neural networks. Many methods have been created over the years to optimize the network configuration based on different approaches. Some methods begin with a very small network and gradually increase the size to accommodate the complexity of the problem. Other approaches prefer to start with a large network and slowly remove network components through a *pruning* process. Some of the pruning algorithms remove the weighted connection one at a time from an existing network configuration in order to

determine which connections can be removed without increasing the network output error. Another type of pruning algorithm removes neurons in order to reduce the size of the network. The network pruning methods are by far the most popular of the optimization techniques due to their ease of implementation.

7.2 Network Pruning Techniques

Network pruning techniques seek to reduce the size of an initial network configuration without significantly affecting the network output. This can be achieved through a number of different techniques that monitor the change in network output error. The methods range from a reliance on mathematical optimization theory to various heuristic techniques. The majority of the techniques can be classified into three main approaches: exhaustive search, penalty methods, and weight elimination. These three methods are discussed below for possible implementation in this work.

7.2.1 Exhaustive Search

The exhaustive search approach to optimizing the network topology relies on the work of many networks. As the name implies, the method calls for the exhaustive creation of all possible network configurations within a restricted class of network architectures. After examining all possible network configurations, the network that produced the smallest increase in output error is selected as the optimum network. This method can require a significant amount of computational effort. Even more important, the network training is discarded for all of the networks that were not selected in the optimization process. As a result, much of the computational work is lost in the process. Surprisingly,

although this method is very inefficient it is often adopted in practice (Bishop 1995). Since the method is highly inefficient though, the exhaustive search will not be implemented in this work

7.2.2 Penalty Methods

A second class of pruning methods directly modifies the magnitudes of the connection weights based on the error function used for network training. Most of the methods provide an additional penalty term to the error function. The penalty term encourages near zero weights during the training process, by penalizing the error function in proportion to the magnitude of the weights. This method results in a weight decay process where many small weight are favored over of a few larger ones (Bishop 1995). Unfortunately, most pruning algorithms seek to remove small weights because of their small influence on the network output. To this extent the method seems counterproductive since the network is forced to reduce the magnitudes of the connection weights, yet the small weights signal the pruning algorithm for weight removal. This problem can be overcome with a modified decay term that scales the weight terms (Weigend et al. 1990). The scaling term must be chosen though trial and error based on the desired effect of weight decay. This decay process can be time-consuming depending on the extent of the training process. A more direct approach can be obtained through a process of weight elimination where the non-influential weights are simply removed instead of relying on a decay process to slowly reduce their values.

7.2.3 Weight Elimination

The final class of network pruning methods focuses on the network response to the direct removal of connection weights. Although there are many different approaches to weight elimination, the basic process remains the same. The methods begin by training the initial large network configuration with one of the standard training algorithms. During the training process, the changes in the weighted connections are recorded. The changes in these network connections in turn provide a measure of the relative importance, or *saliency*, of the different weights (Bishop 1995). According to this premise, small changes in the network connections indicate that the connection does not contribute the network response. Likewise, significant weight changes indicate a possible strong connection to the network response. These assumptions are not universally true for all neural networks.

The weight elimination process to reduce the size of the network proves to be advantageous for several reasons. First, this elimination process removes many of the redundant connections in the network. According to Fu (1994), a fully connected network contains a large amount of redundant information encoded in the weights. Given this notion, it is possible to remove some weights without altering the network performance. Second, the weight elimination process increases the efficiency of the network by reducing the computational overhead. According to Karnin (1990) the cost of computation grows almost linearly with the number of connection weights. Hence a smaller network is more efficient in both the training and final network operation phases. Finally, since most network learning is based on a finite number of training patterns, a network that is too large will tend to memorize the training data. This phenomena, known as “tuning to the noise,” leads to poor generalization capabilities for the network (Karnin 1990). A reduction in

network size through weight elimination reduces the potential for overfitting the training data.

The simplest weight elimination process considers small weights to be of lesser importance than larger weights. It would seem that removing the small connection weights would not affect the network output response. While this idea seems intuitively correct, the remove of small weights can drastically affect the performance of the network (Bishop 1995). This finding is particularly noticeable if the input and output parameters are not normalized. Without normalization, the weights can vary significantly leaving much confusion as to what constitutes small and large weights. Given these shortfalls, other methods have been developed based on more sound principles.

A more reasonable approach to weight elimination considers the change in connection weights in terms of the error function used for network training. For this case, the saliency of the weight is defined in terms of the change in network error resulting from the removal of that weight. Using this idea, it is possible to set each weight to zero in turn and retrain the network each time to determine the effect on the output response. Mathematically speaking, the change in the network error function can be expressed as a Taylor expansion, retaining the first and second derivatives of the error function for simplicity. Assuming that the training process has reached a minimum value along the error function, the first derivative (the gradient) is zero. The change in the error function can then be written in terms of the Hessian matrix of second derivatives. Assuming that weight changes are uncorrelated, the Hessian matrix can be represented by a diagonal matrix. Based on this assumption, the saliencies of the weights can then be determined from the diagonals of the Hessian matrix. After completing training, the weights are sorted

by saliency and the low-saliency weights are deleted. Such a weight elimination procedure is often termed *optimal brain damage* (Le Cun et al. 1990). Unfortunately, neglecting the correlation between the changes in connection weights can lead to poor network performance after weight elimination.

An additional step can be taken to overcome the poor performance of networks that are pruned using the optimal brain damage technique. An improved weight elimination method knows as *optimal brain surgeon* (Hassibi and Stork 1993) considers the correlation between weight changes in the network. The method works by computing corrections to the remaining connection weights after the elimination of a particular weight in the network. This technique again relies on the Hessian matrix of second derivatives to determine the saliency of the weights. This time though, the Hessian matrix can be fully populated and need not be diagonal. Note that, if in fact the Hessian matrix is diagonal, then the optimal brain surgeon method reduces to the optimal brain damage method. After training commences, the weights are sorted by saliency and the low-saliency weights can be eliminated. This method reduces the need for network retraining since the weights are automatically adjusted to achieve the minimum error (Bishop 1995).

The methodology of the optimal brain damage and optimal brain surgeon techniques can be represented graphically. For a simple network with two connection weights, the contours of for a hypothetical error surface are shown in Figure 7.1 (after Bishop 1995). The minimum value of the error surface is located in the center of the error contours. If the optimal brain damage technique removes weight w_1, the error will increase from the minimum to point B. Likewise, if weight w_2 is removed instead, the error will increase from the minimum to point A. In contrast, the optimal brain surgeon technique considers

the correlation between the weight changes. The removal of weight w_1 would then cause the error to increase from the minimum to a point C. As illustrated, the weight w_2 is modified in response to the elimination of weight w_1.

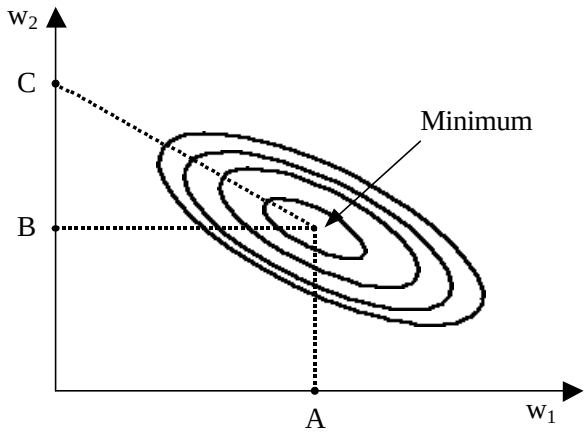


Figure 7.1 Weight Elimination -- Optimal Brain Damage and Optimal Brain Surgeon

7.3 A Simple Implementation of Network Pruning

The pruning techniques discussed to this point offer general solutions to the optimization of network topology. Depending on the complexity of the pruning technique though, the introduction of the methods into a neural network software package can be somewhat cumbersome. In particular, for the optimal brain damage and optimal brain surgeon techniques, the Hessian matrix of second derivatives of the error function must be computed even though it is not even used in the standard backpropagation training algorithm. The calculation of such matrices requires additional computer overhead and is not directly beneficial to the training process. An alternative to such unnecessary computations considers the approximate *sensitivity* of the error function to changes in the connections weights during training. This method, proposed by Karnin (1990), requires

minimum computational overhead and works in parallel with the training process. Unlike the previous approaches to the pruning problem, this method does not require the modification of the error function and does not interfere with the training process.

The pruning method proposed by Karnin (1990) estimates the sensitivity of the error function from changes in the connection weights. This can be achieved by keeping track of the incremental changes to the connection weights during the backpropagation training. After completing network training, the connection weights are ordered by decreasing sensitivity numbers. This sorting of sensitivities allows the network to be pruned in an efficient manner by removing the weights with low sensitivity at the bottom of the list. After pruning, the network can be trained once again to ensure an adequate representation of the data. A brief explanation of the method is outlined below.

The method begins by estimating the sensitivity of the error function based on the initial and final connection weights. The sensitivity S_{ij} can be written as

$$S = -\frac{E(w^f) - E(0)}{w^f - 0} w^f \qquad \textbf{(7.1)}$$

where $w^f = w_{ij}^f$ is the final connection weight after training, $E(w^f)$ is the error function expressed as a function of the connection weight, and $E(0)$ is the error when the weight is removed. Since the training process usually starts with some randomly assigned initial weight w^i, rather than $w = 0$, the sensitivity can be approximated based on the initial value of the error function $E(w^i)$

$$S \approx -\frac{E(w^f) - E(w^i)}{w^f - w^i} w^f \qquad \textbf{(7.2)}$$

where the initial and final weights w^i and w^f, respectively are know at the beginning and end of the training process. This approximation of sensitivity is obtained by measuring the average slope of the error function measured from w^i to w^f. An illustration of the method for a single weight change during training is provided by Karnin (1990) and repeated in Figure 7.2 for clarity.

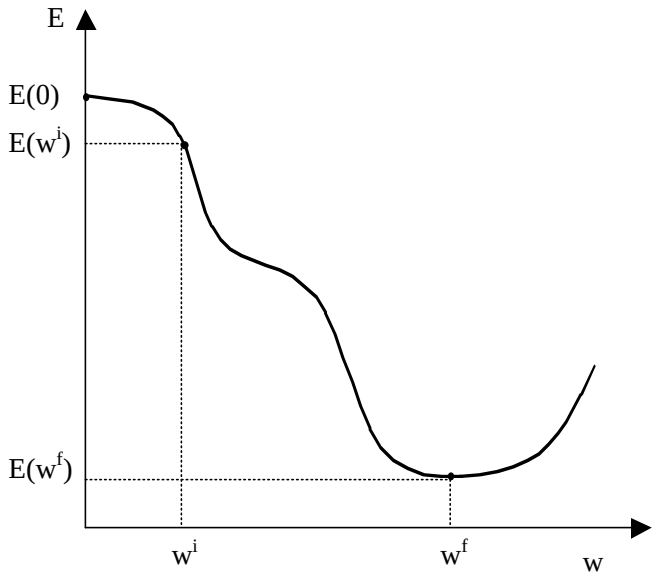


Figure 7.2 Network Error for a Single Weight

For backpropagation training, the changes in weights are correlated so the sensitivity can depend on some or all of the weight changes. However, this method approximates the sensitivity by considering one weight change at a time. Although this may seem limited at first, Karnin suggests that the method does provide a reasonable estimate of the true sensitivities. This approximate procedure can be demonstrated for a network with two connection weights. The training of the network is shown in Figure 7.3 with the error function contours plotted as a function of the two connection weights. In the figure, the point "I" represents the initial training point and the point "F" represents the minimum

value of the error function achieved through training. Also in the figure, the line from point "F" to point "A" represents the change in error by removing connection weight w_2. Likewise, the line from point "F" to point "B" represents the change in error by removing connection weight w_1.

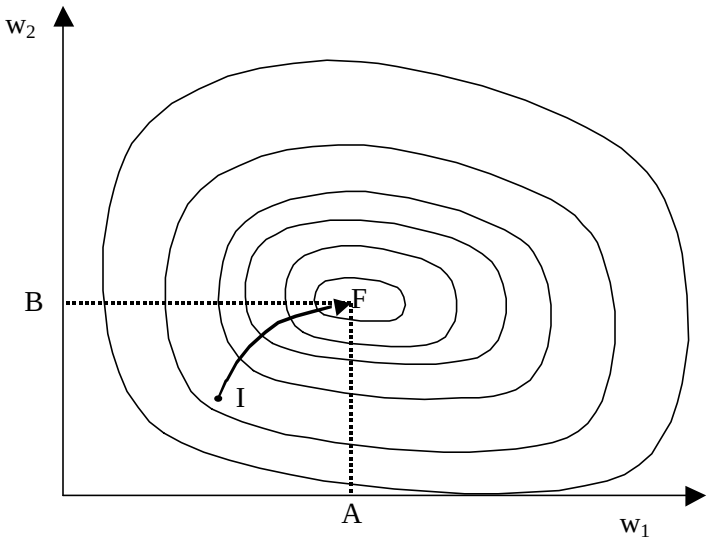


Figure 7.3 Training Along the Error Surface

The equation for the sensitivity depends on the change in error from the initial to the final point on the error surface and the initial and final weights. This method seeks to determine the total change in error from point "A" or point "B" to the final point "F" from the elimination of one of the weights. The total change in error is represented by the numerator in Equation 7.1. For the elimination of connection weight w_2, this quantity is given by the integral from point "A" to "F"

$$E(w^f) - E(0) = \int_A^F \frac{\partial E(w_1, w_2)}{\partial w_2} dw_2 \qquad \textbf{(7.3)}$$

However, since the training starts at some initial point "I", the total change in error from eliminating the connection weight w_2 must be approximated with the integral from point "I" to point "F"

$$E(w^f) - E(0) \approx \int_I^F \frac{\partial E(w_1, w_2)}{\partial w_2} dw_2 \qquad \textbf{(7.4)}$$

Since network training occurs in finite steps, the integral in Equation 7.4 can be replaced by a summation sign. Substituting the total change in weight into Equation 7.2, the sensitivity S_{ij} is given by

$$S_{ij} = -\sum_0^{N-1} \frac{\partial E}{\partial w}(n) \Delta w_{ij}(n) \left(\frac{w_{ij}^f}{w_{ij}^f - w_{ij}^i} \right) \qquad \textbf{(7.5)}$$

where N is the number of training epochs. For the special case of backpropagation training without momentum, Equation 7.5 reduces to

$$S_{ij} = \sum_0^{N-1} \frac{\left[\Delta w_{ij}(n)\right]^2}{\boldsymbol{h}} \left(\frac{w_{ij}^f}{w_{ij}^f - w_{ij}^i} \right) \qquad \textbf{(7.6)}$$

The sensitivity is computed for each connection weight at each epoch. The sensitivities for each weight are then summed over all of the training epochs. Upon completion of training, the corresponding connection weight sensitivities for each neuron in the hidden layer can then be grouped together to determine the overall sensitivity of each neuron to changes in its connection weights. The work presented herein proposes that the hidden neurons with the smallest connection weight sensitivities will have the smallest influence on the network error. Accordingly, these non-influential hidden neurons can be removed, thus completing

the pruning process. The network can of course be retrained after removing the selected connection weights.

7.4 Modifications to the NetSim Program

The pruning process requires the additional computation of the sensitivity of the connection weights in the network. To achieve this goal, the NetSim program was modified by the author to incorporate the above mentioned pruning algorithm into the training process. The modified version of NetSim calculates the sensitivity of each connection weight change to changes in the overall network output error. The sensitivities are calculated in parallel with the training process using the procedure presented in Section 7.3. The new procedure requires the addition of several arrays and matrices to track the initial connection weights, the sensitivities of the connection weights, the gradient of the error surface, and changes in the connection weights per epoch. These data structures are used with the implementation of Equation 7.5 and 7.6 in the training portion of the NetSim program. The final sensitivity results were written to a separate data file for further analysis after the completion of the training phase.

The pruning process implemented in NetSim can be applied to networks in with any given number of hidden layers. For the current application, both the single and multiple column pier networks utilize a single hidden layer for classification. Therefore the work only addresses the removal of neurons in the single hidden layer during the pruning process. The pruning results for both the single and multiple column piers networks are described in more detail in the following sections.

7.5 Pruning Single Column Pier Networks

The network pruning sensitivity algorithm discussed above is now implemented to reduce the size of the network configuration for single column piers. The original network configuration was determined based on trial and error and sizing heuristics. When establishing an initial network configuration, the author attempted to create a network that was too large rather than too small, given the complexity of the problem. This approach is advisable since a network that is too small is typically incapable of mapping the relationship between the input source and output response. The successful training of the single column pier networks in Chapter 5 implies that the networks are oversized to some extent. The pruning results that follow demonstrate that the initial network configurations are currently oversized and can be pruned to reduce the size of the network structure.

In order to prune the four single column pier networks, the training was repeated using a modified version of the NetSim program. The modified version saved the sensitivity of the connection weights to a data file for subsequent analysis. The sensitivity data in its raw form does not provide a clear indication of the contribution of each connection weight to the overall network output error. To provide additional clarity, this work presents the sensitivity of each connection weight in a graphical form. This graphical form is based on Hinton Diagrams that are used to represent the magnitude and sign of the connection weights with the size and color of boxes, respectively. This work presents a variant of the Hinton Diagram in which the size of the boxes depicts the magnitude of the sensitivity of each connection weight. Accordingly, connection weights with little effect on the network error are assigned small boxes. Likewise, connection weights that have a significant effect on the network output error are assigned a large box. Each increase in

box size represents a tenfold increase in weight sensitivity. Also, although it is not done in this work, the boxes could be assigned different colors to indicate the sign of the connection weight sensitivity. For identification purposes, these diagrams of weight sensitivities will be called *sensitivity diagrams*.

The sensitivity diagrams for the four single column pier networks are shown in Figure 7.4. These four diagrams show the variation of connection weight sensitivity between the input and hidden layer. To reiterate, the dotted boxes outline the hidden neurons that can possibly be removed without significantly affecting the network output error. As shown in Figure 7.4, it should be possible to remove two to three neurons from the networks without significantly affecting the network performance. It should be noted that the sensitivities obtained are approximate due to certain assumptions made during the derivation of the network sensitivity Equations 7.5 and 7.6. The examination of the sensitivity diagrams therefore provides some indication of which hidden neurons can be removed under some uncertainty.

Before removing any hidden neurons, the method of removal must be considered. The NetSim program requires the construction of a fully connected, feedforward network, where all hidden neurons participate in the data processing. It is currently not possible to prevent certain hidden neurons that are to be pruned from functioning in the network structure. Consequently, the network configuration must be reformed and retrained, using only the hidden neurons that are not to be pruned. In most cases, the neurons in the hidden layer are then reordered to fill in the gaps from the neurons that were just pruned to again create a fully connected network. Fortunately, each hidden neuron provides an independent classification feature for the network and the classification process does not depend on the

ordering of the neurons. This beneficial feature allows neural networks to perform calculations in parallel. As a result, if the neurons in the hidden layer are rearranged, the network should still be capable of obtaining the same outcome.

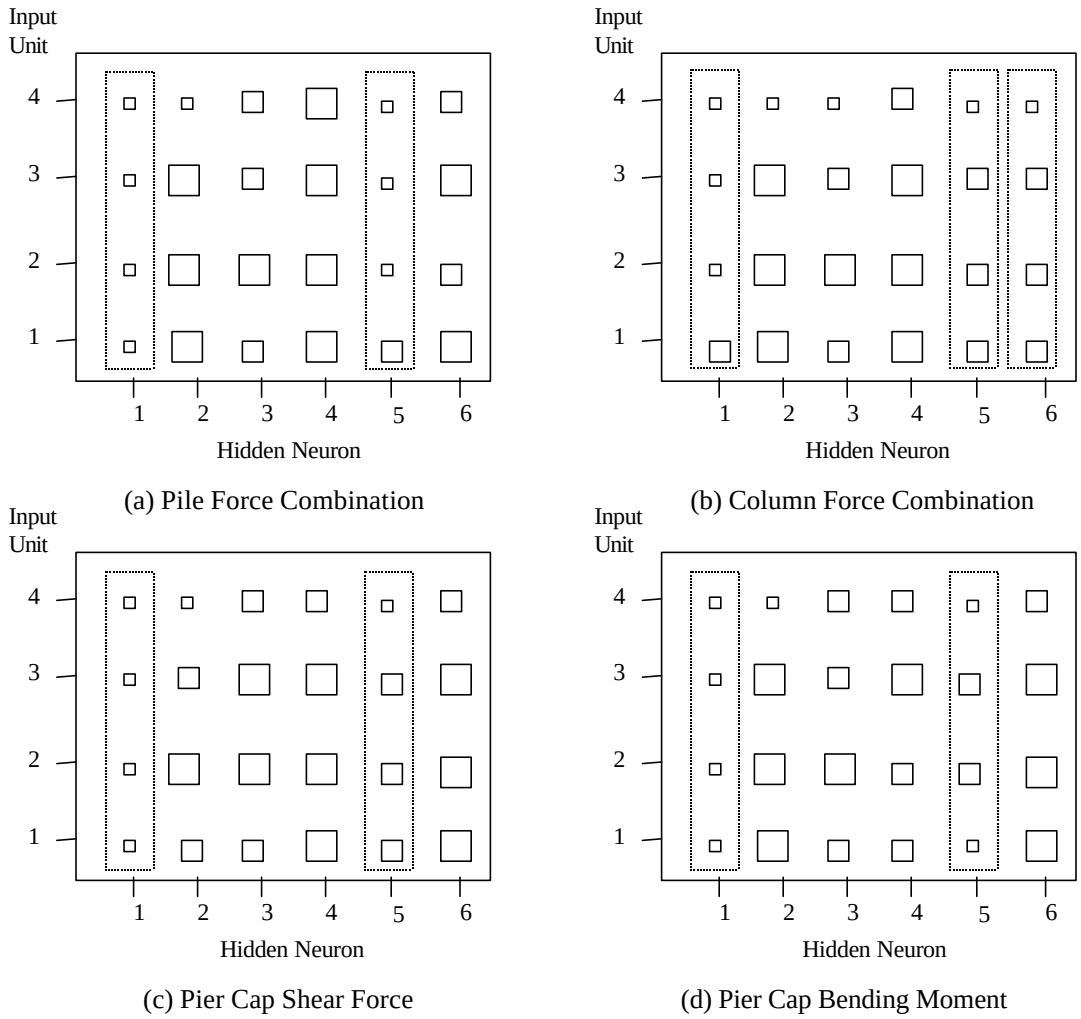


Figure 7.4 Sensitivity Diagrams Before Pruning (Single Column Piers)

To complete the pruning procedure, the hidden neurons identified by the dotted boxes in Figure 7.4 for each of the four networks were removed to reduce the size of the hidden layer. The resulting network configurations from the pruning process are presented in Table 7.1 along with the initial and final RMS validation error, before and after the

pruning, respectively. As show in Table 7.1, three of the four networks were retrained successfully with the removal of two hidden neurons. Efforts to remove additional neurons were unsuccessful during the training phase. The remaining column force combination network could not be retrained with three hidden neurons removed, but could be trained by removing only two neurons. This case demonstrates the approximate nature of the pruning algorithm which can not always exactly determine the number of neurons to prune.

Table 7.1 Pruning Results (Single Column Piers)

Network	Network Configuration		RMS Validation Error	
	Initial	Final	Initial	Final
Pile Force Combination	4x6x8	4x4x8	0.03651	0.10427
Column Force Combination	4x6x8	4x4x8	0.00208	0.00489
Pier Cap Shear Force	4x6x8	4x4x8	0.00631	0.02212
Pier Cap Bending Moment	4x6x8	4x4x8	0.01676	0.01833

The results from the pruned networks indicate that a satisfactory validation error can be obtained with the smaller network configuration. In each of the four networks, the RMS validation error increased, but not significantly. The largest increase in error was observed for the pile force combination network. This increase in error is somewhat expected given the variability of the soil conditions and possible nonlinear soil behavior for the piers in the training set. The remaining three networks, however, showed very little increase in validation error. Given these results, the pruned networks could certainly be used in place of the original network configuration. Not only is the smaller network configuration more computationally efficient, but the smaller number of network connections reduces the potential for overfitting the training data.

7.6 Pruning Multiple Column Pier Networks

The network pruning algorithm is now implemented to reduce the size of the network configuration for multiple column piers. As was the case for the single column pier networks, the original network configuration was determined based on trial and error. The network was first oversized so as to avoid the mapping and convergence problems common to undersized network configurations. Again, the successful training of the multiple column pier networks in Chapter 6 implies that the networks are indeed oversized. The pruning results demonstrate that the networks can be pruned to reduce the size of the network structure.

In order to prune the four multiple column pier networks, the training was repeated using the modified version of the NetSim program. The sensitivity results were obtained for each of the four multiple column networks. The results are shown in the sensitivity diagrams in Figure 7.5. Again, the dotted boxes represent neurons that can possibly be pruned due to low sensitivities.

The resulting network configurations from the pruning process are presented in Table 7.2. To complete the pruning procedure, the hidden neurons identified by the dotted boxes in Figure 7.5 for each of the four networks were removed to attempt to reduce the size of the hidden layer. As shown in Table 7.2, the pile force and column force combination networks were successfully retrained after removing two hidden neurons. Although the sensitivity diagrams in Figure 7.5 suggest that the pile force and column force combination networks could function with the removal of four and three neurons, respectively, the retraining could not achieve this goal. Again, this shortcoming reinforces the fact that the pruning procedure is approximate. It is the author's opinion that an error of

one or two neurons in a pruning procedure is certainly acceptable though. The pier cap shear and bending moment networks were successfully retrained after removing four hidden neurons. For these two cases, the sensitivity diagrams suggested that five neurons could be removed though retraining could not achieve this goal. For these two cases, the number of neurons to prune was overestimated by one neuron.

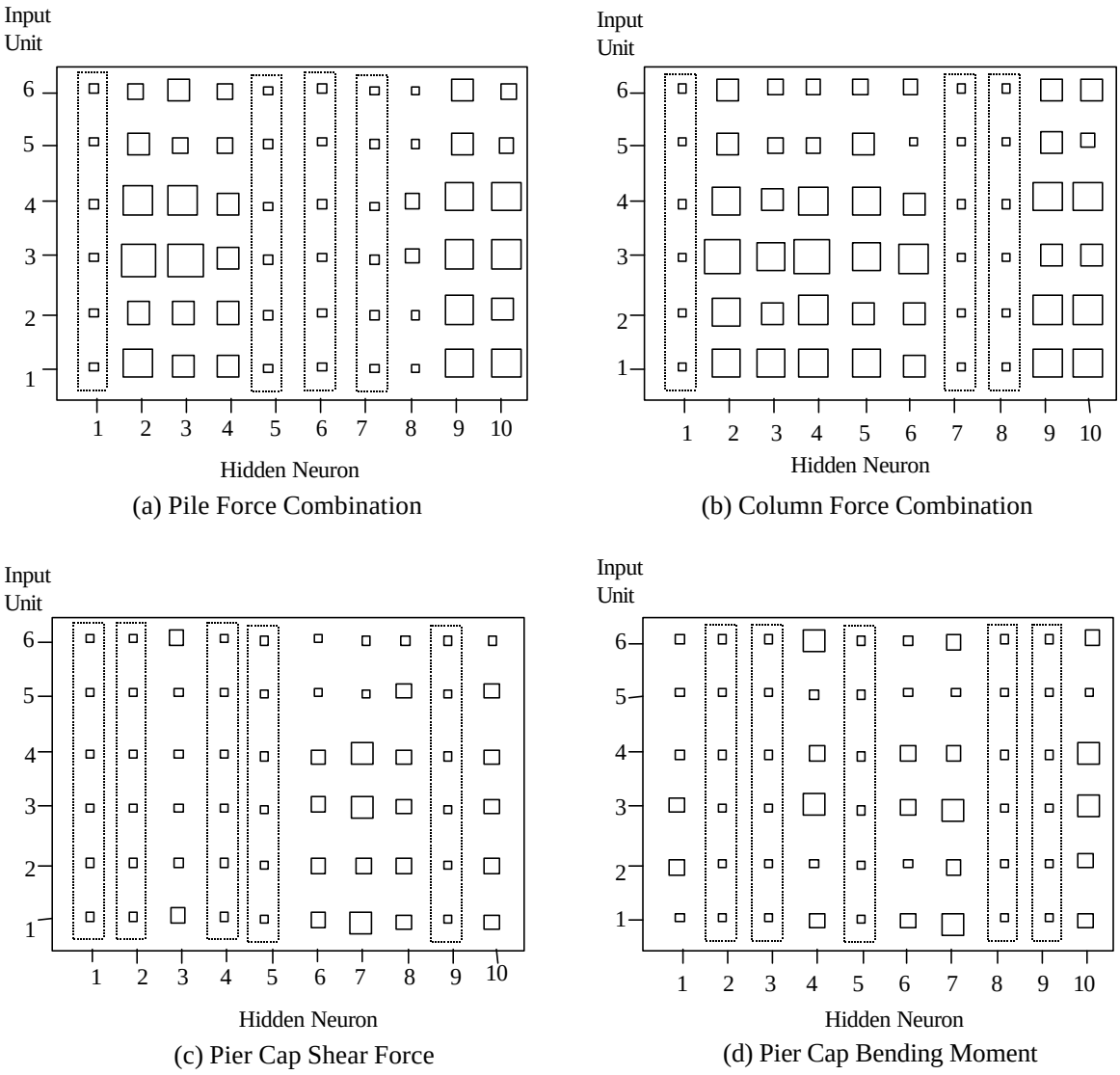


Figure 7.5 Sensitivity Diagrams Before Pruning (Multiple Column Piers)

Table 7.2 Pruning Results (Multiple Column Piers)

Network	Network Configuration		RMS Validation Error	
	Initial	Final	Initial	Final
Pile Force Combination	6x10x8	6x8x8	0.06146	0.06587
Column Force Combination	6x10x8	6x8x8	0.01517	0.11274
Pier Cap Shear Force	6x10x8	6x6x8	0.03638	0.06326
Pier Cap Bending Moment	6x10x8	6x6x8	0.04491	0.03918

The results from the pruned networks indicate that a satisfactory validation error can be obtained with the smaller network configuration. For three of the four networks, the RMS validation error increased, but not significantly. The largest increase in error was observed for the column force combination network. This increase could be due in part to an inability to reach a global minimum point on the error surface. It is possible that further training sessions with many different initial weight configurations could ultimately lead to a smaller validation error. The maximum pile force combination network and the pier cap shear force network showed very little increase in validation error. Surprisingly, the pier cap moment network produced a smaller validation error after pruning. This decrease in error could be attributed to a better fit of the training data with the smaller network configuration. Given these results, the pruned networks could be used in place of the original network configuration. Not only is the smaller network configuration more computationally efficient, but the smaller number of network connections reduces the potential for overfitting the training data.

CHAPTER 8
CONCLUSION

8.1 Live Load Modeling and Generation

A procedure has been developed to model vehicular live loads on modern highway bridges using finite element analysis (FEA). The approach considered the modeling of both design truck and design tandem vehicles along with a design lane load under the AASHTO-LRFD design guidelines. The loads were applied to a three-dimensional finite element model of a two-span bridge to more accurately model the contribution of the wheel loads and design lane at different points along the bridge. In doing so, this procedure eliminated the use of approximate AASHTO distribution factors to distribute the live load to each girder.

A live load generation routine was then implemented to study all possible combinations of load placement along the transverse direction of the bridge. This live load generation was undertaken in order to determine the most critical loading for the interior pier that supports two adjacent spans of the bridge. In accordance with the AASHTO-LRFD design specifications, the live load generation routine considered the presence of vehicle loads in multiple traffic lanes with the appropriate live load reduction factor. All possible live load cases were considered and a separate FEA was performed for each load case to determine the reactions at the bearing pads of the interior support. These

bearing pad reaction forces were then transferred to the Florida Pier program for the pier analysis.

8.2 Live Load Application to Bridge Piers

The live load generation was undertaken to determine all possible combinations of load that can be applied to the pier structure via the bearing pads. All combinations of load must be considered in order to determine which load cases produce the worst force effects in certain components of the bridge pier. This work has identified four force maximum effects that need to be determined for the design of the bridge pier. The maximum force effects are defined as the largest force combination in the piles and pier columns and the largest shear force and bending moment in the pier cap.

The live load generation across the width of the bridges produces several key observations concerning the maximum force effects in the bride piers. First, nearly all of the maximum force effects were produced by loading either one or two design lanes. In many cases, the difference in the magnitude of the force effects between one and two lanes loaded was often small indicating that either case could potentially govern the design. Second, it was often the case that a single lane loading in an exterior lane produced a larger force effect than all lanes loaded simultaneously. This occurrence contrasts many pier designs where all lanes are assumed to be loaded for the maximum force effects to occur. Finally, as anticipated, one particular loading did not produce the all four the maximum force effects. This final point reinforces the need for four separate neural networks to adequately position the loading on the bridge.

8.3 Neural Network Prediction of Load Positions

The use of neural networks has been explored extensively in this work in order to facilitate the pier design process. The networks were created to aid in the analysis of single and multiple column piers. Eight networks were created in all (four for single column piers and four for multiple column piers) in order to predict the live load placement across the width of the bridge to produce the maximum force effects in the pier. The networks were created, trained, and validated with two separate software packages to ensure the validity of the networks.

The creation of the networks first required the identification of the parameters that affect the behavior of the pier under live loading. Nondimensional input parameters were obtained that described the bridge and pier configuration. The output of the network consisted of the positions of the design truck (or tandem) and design lane in each of the four design lanes. The load positions were normalized using the clear roadway of the bridge. The relationship between the input parameters and the output positioning was encoded using a single hidden computation layer.

The training and validation phases were conducted with using data from existing highway bridges. The training pairs represented the broad range of input parameters that are possible for modern highway bridge designs. Although the size of the training set is limited to a database of currently existing bridges, the network prediction results are good. The success in network prediction indicates that the networks can predict the critical load positions for new problems with reasonable accuracy. It is still possible to achieve a poor prediction for input parameters outside the training scope though.

8.4 Network Topology Optimization

After training and validating all of the networks, the topology of each network was optimized to improve the computational efficiency of the networks. This work employed a network pruning process reduced the size of the network by analyzing the sensitivity of the network error to changes in each weighted connection. In this process the sensitivities of all the weighted connections feeding into each neuron were grouped together. The neurons then with the lowest collective weight sensitivities were then identified for possible removal from the network. The pruning results indicate that the neurons with low sensitivity connection weights can be removed from the network configuration without significantly impacting the network prediction error. After pruning, the network configuration requires fewer computations and it therefore more efficient. Equally important, the smaller network configurations are less likely to overfit the training data thereby improving the generalization capabilities of the networks.

8.5 Suggestions for Future Study

While this work has achieved the objectives established from the outset, additional aspects can still be studied for further improvement. Several areas of potential improvement are outlined below.

1. This work is based upon a limited set of training data since the author has chosen to model existing highway bridges. The problem set can certainly be expanded as new bridges are designed or as older bridges are reanalyzed. Since there are currently fewer training pairs than unknown connection weights in the networks, the network predictions are obtained from an overfitting of the current data. It is likely that the addition of more training pairs will reduce the potential for overfitting the training data.

2. More validation problems should be considered to verify the success of the network prediction. It is possible to choose a validation problem that lies outside the training scope and obtain a very poor network prediction. The use of additional validation problems could identify points where the training scope should be expanded to cover additional pier types.

3. Different normalization strategies could be investigated for representing the load positioning across the clear roadway of the bridge. This work uses the range from 0 to 1 for normalization in conjunction with a sigmoid activation function for the neural networks. A hyperbolic tangent activation function could be used instead with a more broad normalization from –1 to 1. For this case, 0 would represent the center of the bridge, with –1 and 1 representing the extreme left and right ends of the clear roadway. This wider normalization range might increase the accuracy of the network predictions.

4. Additional network pruning algorithms could be implemented to reduce the size of the networks. Network pruning is a relatively new topic in neural network literature that is evolving and improving with newer network applications.

APPENDIX A
SINGLE COLUMN PIER NETWORK TRAINING PARAMETERS

Table A.1 Single Column Piers Input Parameters

Pier	I_1	I_2	I_3	I_4
(Training)				
HP01	1	0.71	1.48	0.333
HP02	1	1.21	1.93	0.333
HP03	2	1.33	0.88	0.238
HP04	1	1.00	0.91	0.313
HP05	4	2.78	2.80	0.110
HP06	2	1.35	1.65	0.667
HP07	2	1.78	0.82	0.267
HP08	2	1.31	0.82	0.267
HP09	3	3.14	3.23	0.636
HP10	2	0.84	3.64	0.296
HP11	2	0.84	4.17	0.222
HP12	3	3.29	4.17	0.217
HP13	2	1.24	2.22	0.800
HP14	1	1.21	1.61	0.333
HP15	3	2.88	4.17	0.217
HP16	3	2.92	4.17	0.217
HP17	3	3.50	4.17	0.217
HP18	3	3.33	4.17	0.217
HP19	3	3.89	4.17	0.217
HP20	3	4.00	4.17	0.217
HP21	3	4.09	4.17	0.217
HP22	3	3.12	4.17	0.217
HP23	3	2.22	2.00	0.182
HP24	3	3.12	2.10	0.217
HP25	3	2.26	2.84	0.365
HP26	3	1.75	2.46	0.365
HP27	3	1.75	2.46	0.335
HP28	2	3.00	0.90	0.265
(Validation)				
FICE	3	2.10	2.15	0.205
Flyover	3	3.12	1.82	0.257

$$I_1 = Number\ of\ Design\ Lanes$$

$$I_2 = \frac{Cantilever\ Length}{Girder\ Spacing}$$

$$I_3 = Normalied\ Span$$

$$I_4 = \frac{Pile\ Spacing}{Pier\ Cap\ Width}$$

Table A.2 Single Column Piers Output Parameters - Pile Force Combination

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
HP01	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP02	0.294	0.294	0.000	0.000	0.000	0.000	0.000	0.000
HP03	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP04	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP05	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000
HP06	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP07	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP08	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP09	0.063	0.063	0.000	0.000	0.000	0.000	0.000	0.000
HP10	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP11	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP12	0.109	0.109	0.442	0.442	0.000	0.000	0.000	0.000
HP13	0.178	0.178	0.000	0.000	0.000	0.000	0.000	0.000
HP14	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
HP15	0.132	0.132	0.465	0.465	0.000	0.000	0.000	0.000
HP16	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP17	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP18	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP19	0.104	0.104	0.438	0.438	0.000	0.000	0.000	0.000
HP20	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP21	0.112	0.112	0.446	0.446	0.000	0.000	0.000	0.000
HP22	0.119	0.119	0.444	0.444	0.000	0.000	0.000	0.000
HP23	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP24	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP25	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
HP26	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP27	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP28	0.185	0.185	0.000	0.000	0.000	0.000	0.000	0.000
(Validation)								
FICE	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
Flyover	0.122	0.122	0.000	0.000	0.000	0.000	0.000	0.000

Table A.3 Single Column Piers Output Parameters - Column Force Combination

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
HP01	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP02	0.294	0.294	0.000	0.000	0.000	0.000	0.000	0.000
HP03	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP04	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP05	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000
HP06	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP07	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP08	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP09	0.063	0.063	0.399	0.399	0.000	0.000	0.000	0.000
HP10	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP11	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP12	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
HP13	0.178	0.178	0.000	0.000	0.000	0.000	0.000	0.000
HP14	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
HP15	0.132	0.132	0.000	0.000	0.000	0.000	0.000	0.000
HP16	0.139	0.139	0.000	0.000	0.000	0.000	0.000	0.000
HP17	0.139	0.139	0.000	0.000	0.000	0.000	0.000	0.000
HP18	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP19	0.104	0.104	0.000	0.000	0.000	0.000	0.000	0.000
HP20	0.122	0.122	0.000	0.000	0.000	0.000	0.000	0.000
HP21	0.112	0.112	0.000	0.000	0.000	0.000	0.000	0.000
HP22	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP23	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP24	0.122	0.122	0.000	0.000	0.000	0.000	0.000	0.000
HP25	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
HP26	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP27	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP28	0.185	0.185	0.000	0.000	0.000	0.000	0.000	0.000
(Validation)								
FICE	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
Flyover	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000

Table A.4 Single Column Piers Output Parameters - Pier Cap Shear Force

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
HP01	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP02	0.294	0.294	0.000	0.000	0.000	0.000	0.000	0.000
HP03	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP04	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP05	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000
HP06	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP07	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP08	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP09	0.063	0.063	0.399	0.399	0.000	0.000	0.000	0.000
HP10	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP11	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP12	0.109	0.109	0.442	0.442	0.000	0.000	0.000	0.000
HP13	0.178	0.178	0.000	0.000	0.000	0.000	0.000	0.000
HP14	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
HP15	0.132	0.132	0.465	0.465	0.000	0.000	0.000	0.000
HP16	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP17	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP18	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP19	0.104	0.104	0.438	0.438	0.000	0.000	0.000	0.000
HP20	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP21	0.112	0.112	0.446	0.446	0.000	0.000	0.000	0.000
HP22	0.119	0.119	0.444	0.444	0.000	0.000	0.000	0.000
HP23	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP24	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP25	0.111	0.111	0.444	0.444	0.000	0.000	0.000	0.000
HP26	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP27	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP28	0.185	0.185	0.000	0.000	0.000	0.000	0.000	0.000
(Validation)								
FICE	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
Flyover	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000

Table A.5 Single Column Piers Output Parameters - Pier Cap Bending Moment

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
HP01	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP02	0.294	0.294	0.000	0.000	0.000	0.000	0.000	0.000
HP03	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP04	0.333	0.333	0.000	0.000	0.000	0.000	0.000	0.000
HP05	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000
HP06	0.208	0.208	0.000	0.000	0.000	0.000	0.000	0.000
HP07	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP08	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP09	0.063	0.063	0.399	0.399	0.000	0.000	0.000	0.000
HP10	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP11	0.179	0.179	0.000	0.000	0.000	0.000	0.000	0.000
HP12	0.109	0.109	0.442	0.442	0.000	0.000	0.000	0.000
HP13	0.178	0.178	0.000	0.000	0.000	0.000	0.000	0.000
HP14	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
HP15	0.132	0.132	0.465	0.465	0.000	0.000	0.000	0.000
HP16	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP17	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
HP18	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP19	0.104	0.104	0.438	0.438	0.000	0.000	0.000	0.000
HP20	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP21	0.112	0.112	0.446	0.446	0.000	0.000	0.000	0.000
HP22	0.119	0.119	0.444	0.444	0.000	0.000	0.000	0.000
HP23	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
HP24	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000
HP25	0.111	0.111	0.444	0.444	0.000	0.000	0.000	0.000
HP26	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP27	0.119	0.119	0.452	0.452	0.000	0.000	0.000	0.000
HP28	0.185	0.185	0.000	0.000	0.000	0.000	0.000	0.000
(Validation)								
FICE	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
Flyover	0.122	0.122	0.455	0.455	0.000	0.000	0.000	0.000

APPENDIX B
MULTIPLE COLUMN PIER NETWORK TRAINING PARAMETERS

Table B.1 Multiple Column Piers Input Parameters

Pier	I_1	I_2	I_3	I_4	I_5	I_6
(Training)						
PR01	3	1.29	5.93	2.72	0.37	0.57
PR02	3	1.23	5.42	1.95	0.45	0.31
PR03	2	1.10	4.92	3.15	0.41	0.60
PR04	2	1.24	4.92	3.15	0.43	0.60
PR05	2	1.21	6.08	3.43	0.41	0.31
PR06	3	1.44	8.02	3.42	0.46	0.57
PR07	3	2.10	8.63	2.96	0.38	0.31
PR08	3	1.74	8.63	2.96	0.38	0.31
PR09	4	1.06	8.21	2.96	0.38	0.50
PR10	4	1.26	6.41	2.06	0.44	0.50
PR11	3	1.90	5.71	4.57	0.00	0.50
PR12	3	1.92	5.71	4.57	0.00	0.32
PR13	2	0.95	3.56	2.00	0.28	0.63
PR14	4	1.21	5.10	2.10	0.33	0.68
PR15	2	1.27	2.54	1.32	0.54	0.67
PR16	3	0.82	5.67	3.11	0.35	0.33
PR17	3	1.55	5.73	3.30	0.35	0.50
PR18	4	2.52	5.60	3.04	0.37	0.60
PR19	2	0.91	5.12	2.64	0.35	0.57
PR20	4	1.15	7.06	2.29	0.35	0.29
PR21	3	1.36	9.00	3.50	0.37	0.31
PR22	3	1.38	5.48	3.00	0.38	0.57
PR23	3	1.58	7.50	2.50	0.38	0.31
PR24	4	1.11	6.71	2.55	0.33	0.31
PR25	2	1.54	7.00	2.50	0.38	0.31
PR26	2	4.30	3.59	3.46	0.00	0.13
PR27	3	2.10	5.75	2.13	0.53	0.33
PR28	3	1.06	8.26	5.16	0.25	0.42
PR29	3	3.00	4.59	2.93	0.27	0.38
PR30	3	1.57	4.80	3.33	0.34	0.57
PR31	3	1.57	5.80	3.33	0.33	0.35
PR32	3	1.55	5.75	3.38	0.35	0.35
PR33	3	2.24	9.00	4.50	0.43	0.09
PR34	4	1.74	6.67	2.25	0.40	0.31

$$I_1 = Number\ of\ \ Design\ Lanes$$

$$I_2 = Normalized\ Span$$

$$I_3 = \frac{Bridge Width}{Girder\ Spacing}$$

$$I_4 = \frac{Column\ Spacing}{Girder\ Spacing}$$

$$I_5 = \frac{Cantilever\ Length}{Column\ Spacing}$$

$$I_6 = \frac{Pile\ Spacing}{Cap\ Width}$$

PR35	3	1.12	3.64	2.00	0.41	0.31
PR36	2	1.52	6.09	3.36	0.41	0.57
PR37	4	1.50	7.80	2.77	0.39	0.57
PR38	4	1.50	7.75	2.81	0.38	0.53
PR39	3	1.85	5.80	3.33	0.32	0.27
PR40	3	2.23	5.64	3.33	0.32	0.26
PR41	3	1.59	3.60	2.20	0.36	0.60
PR42	3	2.26	4.80	2.93	0.36	0.35
PR43	4	0.86	10.2	5.60	.038	0.38
PR44	4	0.80	10.2	5.00	0.44	0.33
PR45	3	0.95	4.62	2.72	0.42	0.33
PR46	3	1.62	4.73	3.33	0.36	0.37
PR47	2	4.30	3.59	3.46	0.00	0.13
(Validation)						
Blairst	3	1.18	3.00	1.00	0.46	0.17
SR20	2	0.35	3.33	1.00	0.42	0.35

Table B.2 Multiple Column Piers Output Parameters - Pile Force Combination

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
PR01	0.104	0.104	0.000	0.000	0.896	0.896	0.000	0.000
PR02	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR03	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR04	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR05	0.157	0.157	0.657	0.657	0.000	0.000	0.000	0.000
PR06	0.214	0.214	0.451	0.451	0.786	0.786	0.000	0.000
PR07	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR08	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR09	0.089	0.089	0.339	0.339	0.000	0.000	0.000	0.000
PR10	0.089	0.089	0.339	0.339	0.000	0.000	0.000	0.000
PR11	0.125	0.125	0.458	0.458	0.000	0.000	0.000	0.000
PR12	0.125	0.125	0.458	0.458	0.000	0.000	0.000	0.000
PR13	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR14	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000
PR15	0.192	0.192	0.000	0.000	0.000	0.000	0.000	0.000
PR16	0.098	0.098	0.000	0.000	0.902	0.902	0.000	0.000
PR17	0.117	0.117	0.000	0.000	0.884	0.884	0.000	0.000
PR18	0.071	0.071	0.000	0.000	0.000	0.000	0.929	0.929
PR19	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR20	0.083	0.083	0.000	0.000	0.000	0.000	0.000	0.000
PR21	0.222	0.222	0.444	0.444	0.778	0.778	0.000	0.000
PR22	0.119	0.119	0.000	0.000	0.879	0.879	0.000	0.000
PR23	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
PR24	0.000	0.000	0.404	0.404	0.596	0.596	0.000	0.000
PR25	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
PR26	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR27	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR28	0.206	0.206	0.461	0.461	0.000	0.000	0.000	0.000
PR29	0.227	0.227	0.440	0.440	0.000	0.000	0.000	0.000
PR30	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
PR31	0.218	0.218	0.448	0.448	0.000	0.000	0.000	0.000
PR32	0.218	0.218	0.448	0.448	0.000	0.000	0.000	0.000
PR33	0.093	0.093	0.426	0.426	0.000	0.000	0.000	0.000
PR34	0.083	0.083	0.000	0.000	0.000	0.000	0.000	0.000
PR35	0.148	0.148	0.000	0.000	0.000	0.000	0.000	0.000
PR36	0.149	0.149	0.000	0.000	0.000	0.000	0.000	0.000
PR37	0.086	0.086	0.000	0.000	0.000	0.000	0.915	0.915
PR38	0.081	0.081	0.000	0.000	0.000	0.000	0.919	0.919
PR39	0.115	0.115	0.000	0.000	0.000	0.000	0.000	0.000
PR40	0.096	0.096	0.492	0.492	0.000	0.000	0.000	0.000
PR41	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
PR42	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
PR43	0.098	0.098	0.366	0.366	0.000	0.000	0.000	0.000
PR44	0.098	0.098	0.348	0.348	0.000	0.000	0.000	0.000

PR45	0.111	0.111	0.444	0.444	0.000	0.000	0.000	0.000
PR46	0.205	0.205	0.462	0.462	0.000	0.000	0.000	0.000
PR47	0.206	0.206	0.706	0.706	0.000	0.000	0.000	0.000
(Validation)								
Blairst	0.167	0.167	0.500	0.500	0.000	0.000	0.000	0.000
SR20	0.250	0.250	0.000	0.000	0.000	0.000	0.000	0.000

Table B.3 Multiple Column Piers Output Parameters - Column Force Combination

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
PR01	0.104	0.104	0.000	0.000	0.896	0.896	0.000	0.000
PR02	0.098	0.098	0.000	0.000	0.903	0.903	0.000	0.000
PR03	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR04	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR05	0.343	0.343	0.657	0.657	0.000	0.000	0.000	0.000
PR06	0.214	0.214	0.451	0.451	0.000	0.000	0.000	0.000
PR07	0.085	0.085	0.000	0.000	0.900	0.900	0.000	0.000
PR08	0.085	0.085	0.000	0.000	0.900	0.900	0.000	0.000
PR09	0.089	0.089	0.339	0.339	0.000	0.000	0.911	0.911
PR10	0.089	0.089	0.339	0.339	0.000	0.000	0.911	0.911
PR11	0.208	0.208	0.458	0.458	0.000	0.000	0.000	0.000
PR12	0.208	0.208	0.458	0.458	0.000	0.000	0.000	0.000
PR13	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR14	0.000	0.000	0.402	0.402	0.598	0.598	0.000	0.000
PR15	0.192	0.192	0.000	0.000	0.000	0.000	0.000	0.000
PR16	0.098	0.098	0.000	0.000	0.902	0.902	0.000	0.000
PR17	0.117	0.117	0.000	0.000	0.884	0.884	0.000	0.000
PR18	0.071	0.071	0.000	0.000	0.000	0.000	0.929	0.929
PR19	0.156	0.156	0.844	0.844	0.000	0.000	0.000	0.000
PR20	0.083	0.083	0.000	0.000	0.000	0.000	0.917	0.917
PR21	0.222	0.222	0.444	0.444	0.000	0.000	0.000	0.000
PR22	0.119	0.119	0.000	0.000	0.879	0.879	0.000	0.000
PR23	0.111	0.111	0.000	0.000	0.889	0.889	0.000	0.000
PR24	0.000	0.000	0.404	0.404	0.596	0.596	0.000	0.000
PR25	0.119	0.119	0.000	0.000	0.881	0.881	0.000	0.000
PR26	0.109	0.109	0.000	0.000	0.891	0.891	0.000	0.000
PR27	0.109	0.109	0.359	0.359	0.000	0.000	0.000	0.000
PR28	0.000	0.000	0.539	0.539	0.000	0.000	0.000	0.000
PR29	0.227	0.227	0.440	0.440	0.000	0.000	0.000	0.000
PR30	0.194	0.194	0.472	0.472	0.000	0.000	0.000	0.000
PR31	0.115	0.115	0.000	0.000	0.885	0.885	0.000	0.000
PR32	0.115	0.115	0.000	0.000	0.885	0.885	0.000	0.000
PR33	0.093	0.093	0.000	0.000	0.000	0.000	0.000	0.000
PR34	0.083	0.083	0.000	0.000	0.000	0.000	0.917	0.917
PR35	0.148	0.148	0.000	0.000	0.000	0.000	0.000	0.000
PR36	0.149	0.149	0.000	0.000	0.000	0.000	0.000	0.000
PR37	0.086	0.086	0.000	0.000	0.000	0.000	0.915	0.915
PR38	0.081	0.081	0.000	0.000	0.000	0.000	0.919	0.919
PR39	0.115	0.115	0.000	0.000	0.000	0.000	0.000	0.000
PR40	0.096	0.096	0.000	0.000	0.841	0.841	0.000	0.000
PR41	0.139	0.139	0.472	0.472	0.000	0.000	0.000	0.000
PR42	0.194	0.194	0.472	0.472	0.000	0.000	0.000	0.000
PR43	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR44	0.098	0.098	0.000	0.000	0.000	0.000	0.902	0.902

PR45	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
PR46	0.205	0.205	0.462	0.462	0.000	0.000	0.000	0.000
PR47	0.294	0.294	0.706	0.706	0.000	0.000	0.000	0.000
(Validation)								
Blairst	0.167	0.167	0.500	0.500	0.000	0.000	0.000	0.000
SR20	0.250	0.250	0.000	0.000	0.000	0.000	0.000	0.000

Table B.4 Multiple Column Piers Output Parameters - Pier Cap Shear Force

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
PR01	0.104	0.104	0.000	0.000	0.000	0.000	0.000	0.000
PR02	0.236	0.236	0.430	0.430	0.000	0.000	0.000	0.000
PR03	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR04	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR05	0.343	0.343	0.657	0.657	0.000	0.000	0.000	0.000
PR06	0.214	0.214	0.451	0.451	0.000	0.000	0.000	0.000
PR07	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR08	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR09	0.089	0.089	0.000	0.000	0.000	0.000	0.000	0.000
PR10	0.089	0.089	0.000	0.000	0.000	0.000	0.000	0.000
PR11	0.125	0.125	0.458	0.458	0.000	0.000	0.000	0.000
PR12	0.125	0.125	0.458	0.458	0.000	0.000	0.000	0.000
PR13	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR14	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR15	0.192	0.192	0.000	0.000	0.000	0.000	0.000	0.000
PR16	0.235	0.235	0.431	0.431	0.000	0.000	0.000	0.000
PR17	0.117	0.117	0.000	0.000	0.000	0.000	0.000	0.000
PR18	0.179	0.179	0.321	0.321	0.000	0.000	0.000	0.000
PR19	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR20	0.167	0.167	0.333	0.333	0.000	0.000	0.000	0.000
PR21	0.222	0.222	0.444	0.444	0.000	0.000	0.000	0.000
PR22	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
PR23	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
PR24	0.154	0.154	0.346	0.346	0.000	0.000	0.000	0.000
PR25	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
PR26	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR27	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR28	0.206	0.206	0.461	0.461	0.000	0.000	0.000	0.000
PR29	0.227	0.227	0.440	0.440	0.000	0.000	0.000	0.000
PR30	0.194	0.194	0.472	0.472	0.000	0.000	0.000	0.000
PR31	0.218	0.218	0.448	0.448	0.000	0.000	0.000	0.000
PR32	0.218	0.218	0.448	0.448	0.000	0.000	0.000	0.000
PR33	0.093	0.093	0.000	0.000	0.000	0.000	0.000	0.000
PR34	0.167	0.167	0.333	0.333	0.000	0.000	0.000	0.000
PR35	0.148	0.148	0.000	0.000	0.000	0.000	0.000	0.000
PR36	0.351	0.351	0.649	0.649	0.000	0.000	0.000	0.000
PR37	0.165	0.165	0.335	0.335	0.000	0.000	0.000	0.000
PR38	0.169	0.169	0.331	0.331	0.000	0.000	0.000	0.000
PR39	0.218	0.218	0.448	0.448	0.000	0.000	0.000	0.000
PR40	0.238	0.238	0.429	0.429	0.000	0.000	0.000	0.000
PR41	0.139	0.139	0.000	0.000	0.000	0.000	0.000	0.000
PR42	0.194	0.194	0.472	0.472	0.000	0.000	0.000	0.000
PR43	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR44	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000

PR45	0.222	0.222	0.444	0.444	0.000	0.000	0.000	0.000
PR46	0.205	0.205	0.462	0.462	0.000	0.000	0.000	0.000
PR47	0.206	0.206	0.706	0.706	0.000	0.000	0.000	0.000
(Validation)								
Blairst	0.167	0.167	0.000	0.000	0.833	0.833	0.000	0.000
SR20	0.250	0.250	0.000	0.000	0.000	0.000	0.000	0.000

Table B.5 Multiple Column Piers Output Parameters - Pier Cap Moment

Pier	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
(Training)								
PR01	0.104	0.104	0.000	0.000	0.000	0.000	0.000	0.000
PR02	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR03	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR04	0.343	0.343	0.656	0.656	0.000	0.000	0.000	0.000
PR05	0.157	0.157	0.000	0.000	0.000	0.000	0.000	0.000
PR06	0.214	0.214	0.451	0.451	0.000	0.000	0.000	0.000
PR07	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR08	0.085	0.085	0.000	0.000	0.000	0.000	0.000	0.000
PR09	0.089	0.089	0.000	0.000	0.000	0.000	0.000	0.000
PR10	0.089	0.089	0.000	0.000	0.000	0.000	0.000	0.000
PR11	0.208	0.208	0.458	0.458	0.842	0.842	0.000	0.000
PR12	0.208	0.208	0.542	0.508	0.792	0.792	0.000	0.000
PR13	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR14	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR15	0.192	0.192	0.000	0.000	0.000	0.000	0.000	0.000
PR16	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR17	0.117	0.117	0.000	0.000	0.000	0.000	0.000	0.000
PR18	0.071	0.071	0.000	0.000	0.000	0.000	0.000	0.000
PR19	0.156	0.156	0.000	0.000	0.000	0.000	0.000	0.000
PR20	0.083	0.083	0.000	0.000	0.000	0.000	0.000	0.000
PR21	0.222	0.222	0.444	0.444	0.000	0.000	0.000	0.000
PR22	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
PR23	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
PR24	0.096	0.096	0.000	0.000	0.000	0.000	0.000	0.000
PR25	0.119	0.119	0.000	0.000	0.000	0.000	0.000	0.000
PR26	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR27	0.109	0.109	0.000	0.000	0.000	0.000	0.000	0.000
PR28	0.127	0.127	0.000	0.000	0.000	0.000	0.000	0.000
PR29	0.227	0.227	0.440	0.440	0.000	0.000	0.000	0.000
PR30	0.194	0.194	0.472	0.472	0.000	0.000	0.000	0.000
PR31	0.115	0.115	0.000	0.000	0.000	0.000	0.000	0.000
PR32	0.115	0.115	0.000	0.000	0.000	0.000	0.000	0.000
PR33	0.093	0.093	0.000	0.000	0.000	0.000	0.000	0.000
PR34	0.083	0.083	0.000	0.000	0.000	0.000	0.000	0.000
PR35	0.148	0.148	0.000	0.000	0.000	0.000	0.000	0.000
PR36	0.149	0.149	0.000	0.000	0.000	0.000	0.000	0.000
PR37	0.086	0.086	0.000	0.000	0.000	0.000	0.000	0.000
PR38	0.081	0.081	0.000	0.000	0.000	0.000	0.000	0.000
PR39	0.115	0.115	0.000	0.000	0.000	0.000	0.000	0.000
PR40	0.096	0.096	0.000	0.000	0.000	0.000	0.000	0.000
PR41	0.139	0.139	0.000	0.000	0.000	0.000	0.000	0.000
PR42	0.000	0.000	0.472	0.472	0.000	0.000	0.000	0.000
PR43	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000
PR44	0.098	0.098	0.000	0.000	0.000	0.000	0.000	0.000

PR45	0.111	0.111	0.000	0.000	0.000	0.000	0.000	0.000
PR46	0.205	0.205	0.538	0.538	0.795	0.795	0.000	0.000
PR47	0.206	0.206	0.706	0.706	0.000	0.000	0.000	0.000
(Validation)								
Blairst	0.167	0.167	0.000	0.000	0.833	0.833	0.000	0.000
SR20	0.250	0.250	0.000	0.000	0.000	0.000	0.000	0.000

REFERENCES

ABAQUS Theory Manual. (1998). Hibbitt, Karlsson, and Sorensen.

ADINA Theory and Modeling Guide. (1997). ADINA Research and Design.

American Association of State, Highway, and Transportation Officials (AASHTO). (1994). *AASHTO-LRFD Bridge Design Specifications*. Washington, D.C.: AASHTO.

ANSYS Structural Analysis Guide. (1996). ANSYS.

Artificial Neural Networks for Civil Engineers: Fundamentals and Applications. (1997). N. Kartem and I. Flood, Ed. New York: ASCE.

Basheer, I., and Najjar, Y. (1997). Discussion of Modeling Initial Design Process using Artificial Neural Networks. *Journal of Computing in Civil Engineering*, ASCE 2(2): 145.

Baum, E., and Haussler, D. (1989). What Size Nets Gives Valid Generalization? *Neural Computation* 1: 151-160.

Better Roads. (1994). 1994 Bridge Inventory. *Better Roads*, Wm. O. Dannhausen 64(11): 28.

Bigus, J. (1996). *Data Mining with Neural Networks*. New York: McGraw Hill.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Buckingham, E. (1915). Model Experiments and the Form of Empirical Equations. *Transactions ASME* 37: 263-296.

California Department of Transportation (CALTRANS). (1993). *Bridge Design Practice Manual*. Sacramento, CA: CALTRANS.

Carpenter, W., and Bartheremy, J. (1994). Common Misconceptions about Neural Networks as Approximators. *Journal of Computing in Civil Engineering*, ASCE 8(3): 345-358.

Chauvin, Y. (1990). A Backpropagation Algorithm with Optimal Use of Hidden Units. *Advances in Neural Information Processing* 1. D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann Publishers. pp. 519-526.

Chen, Y., and Aswad, A. (1996). Stretching Span Capability of Prestressed Concrete Bridges Under AASHTO LRFD. *Journal of Bridge Engineering,* ASCE 1(3): 112-120.

Consolazio, G. (1995). *Analysis of Highway Bridges using Computer Assisted Modeling, Neural Networks, and Data Compression Techniques*. Ph.D. Dissertation, University of Florida, Gainesville.

Cook, R., Malkus, D., and Plesha, M. (1989). *Concepts and Applications of Finite Element Analysis*. New York: John Wiley and Sons.

Flood, I. (1991). A Gaussian-Based Feedforward Network Architecture and Complimentary Training Algorithm. *Proceedings from the International Joint Conference on Neural Networks*. New York: IEEE: 171-176.

Flood, I., and Kartam, N. (1994a). Neural Networks in Civil Engineering I: Principles and Understanding. *Journal of Computing in Civil Engineering,* ASCE 8(2): 131-148.

Flood, I., and Kartam, N. (1994b). Neural Networks in Civil Engineering II: Systems and Applications. *Journal of Computing in Civil Engineering,* ASCE 8(2): 149-162.

Fu, LiMin. (1994). *Neural Networks in Computer Intelligence*. New York: McGraw-Hill.

Gagarin, N.,Flood, I., and Albrecht, P. (1994). Computing Truck Attributes with Artificial Neural Networks. *Journal of Computing in Civil Engineering,* ASCE 8(2): 179-200.

Hajela, P., and Berke, L. (1991). Neurobiological Computational Models in Structural Analysis and Design. *Computers and Structures* 41(4): 657-667.

Hanson, S., and Pratt, L. (1989). Comparing Biases for Minimal Network Construction with Backpropagation. *Advances in Neural Information Processing* 1. D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann Publishers. pp. 177-185.

Hassibi, B., and Stork, D. (1993). Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. *Advances in Neural Information Processing Systems* 5: pp. 164-171.

Hassoun, M. (1995). *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, New Jersey: Prentice Hall.

Hays, C., Hoit, M., Consolazio, G., and Kakhandiki, A. (1994) *Bridge Rating of Girder-Slab Bridges Using Automated Finite Element Technology*. Structure Research Report No. 94-1, Department of Civil Engineering, University of Florida, Gainesville.

Hebb, D. (1961). *Organization of Behavior*. New York: Science Editions.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Reading, MA: Addison-Wesley Publishing Company.

Hoit, M., McVay, M., and Hays, C., (1998). *Florida Pier Users Manual*. Department of Civil Engineering, University of Florida, Gainesville.

Hoit, M., McVay, M., Hays, C., and Andrade, P. (1996). Nonlinear Pile Foundation Analysis Using Florida Pier. *Journal of Bridge Engineering*, ASCE 1(4): 135-142.

Hopfield, J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences USA* 81: 6871-6874.

Jacobs, R. (1988). *Increased Rates of Convergence through Learning Rate Adaptation. Neural Networks* 1(4): 295-307.

Karnin, E. (1990). A Simple Procedure for Pruning Back-Propagation Trained Neural Networks. *IEEE Transactions on Neural Networks*. IEEE 1 (2): 239-242.

Kartalopoulos, S. (1996). *Understanding Neural Networks and Fuzzy Logic*. New York: IEEE Press.

Lapedes, A. and Farber, R. (1988). How Neural Networks Work. *Neural Information Processing Systems*. American Institute of Physics. pp. 442-456.

Le Cun, Y., Boser, B., Denker, J., and Solla, S. (1990). Optimal Brain Damage, *Advances in Neural Information Processing Systems* 2. D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann Publishers. pp.598-605.

Lertpaitoonpan, W. (1997). *Live Load Generation from Bridge Superstructures into Loads on the Bridge Substructures*. Master's Report, University of Florida, Gainesville.

Levenberg, K. (1944). A Method for the Solution of Certain Nonlinear Problems in Least Squares. *Quarterly Journal of Applied Mathematics II* 2: 164-168.

Marquardt, D. (1963). An algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society of Industrial and Applied Mathematics 11* 2: 431-441.

Mindlin, R. (1951). Influence of Rotarty Inertia and Shear on Flexural Motions of Isotropic Elastic Plates. *Journal of Applied Mechanics* 12: 31-38.

Mitchell, J. (1973). *A Nonlinear Analysis of Biaxially Loaded Beam-Columns Using a Discrete Element Method*. Ph.D. Dissertation, University of Texas at Austin.

Mozer, M., and Smolensky, P. (1989). Skeletonization: A Technique for Trimming the Fat from a Network Via Relevance Assessment. *Advances in Neural Information Processing* 1. D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann Publishers. pp. 107-115.

Mukherjee, A., and Deshpande, J. (1995). Modeling Initial Design Process using Artificial Neural Networks. *Journal of Computing in Civil Engineering,* ASCE 9(3): 194-200.

Mukherjee, A. and Deshpande, J. (1997). Closure to Modeling Initial Design Process using Artificial Neural Networks. *Journal of Computing in Civil Engineering,* ASCE pp. 145.

Neural Network Toolbox Version 3.0 User's Manual. (1999). Mathtools Corporation.

Rajasekaran, S. (1997). Discussion of Modeling Initial Design Process using Artificial Neural Networks. *Journal of Computing in Civil Engineering,* ASCE 2(2): 145.

Reissner, E. (1945). The Effect of Transverse Shear Deformations on the Bending of Elastic Plates. *Journal of Applied Mechanics* 67: A69-77.

Rogers, J. (1994). Simulating Structural Analysis with Neural Networks. *Journal of Computing in Civil Engineering,* ASCE 8(2): 252-265.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning Internal Representations by Error Propagation. *Parallel Distributed Processing*. Cambridge, MA: MIT Press.

Schwartz, D., Samalan, S., Solla, S., and Denker, J. (1990). Exhaustive Learning. *Neural Computation* 2 (3): 374-385.

Sietsma, J., and Dow, R. (1991). Neural Net Pruning—Why and How. *Proceeding of the International Joint Conference on Neural Networks* 1: 325-333.

STRAP Users Manual. (1998). ATIR Engineering Software Development.

Taylor, E. (1974). *Dimensional Analysis for Engineers*. Oxford: Oxford University Press.

Tonias, D. (1995). *Bridge Engineering*. New York: McGraw Hill.

Vanluchene, R., and Sun, R. (1990). Neural Networks in Structural Engineering, *Microcomputers in Civil Engineering* 5: 207-215.

Wasserman, P. (1989). *Neural Computing: Theory and Practice.* New York: Van Nostrand Reinhold.

Wasserman, P. (1993). *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold.

Weigend, A., Rumelhart, D., and Huberman, B. (1990). Backpropagation, Weight Elimination, and Time Series Prediction. *Proceedings of the 1990 Connectionist Models Summer School*, pp. 65-80.

Yeh, Y., Kuo, Y., and Hsu, D. (1993). Building KBES for Diagnosing PC Pile with Artificial Neural Networks. *Journal of Computing in Civil Engineering*, ASCE 7(1): 71-93.

BIOGRAPHICAL SKETCH

Mark Erik Williams was born in Pittsburgh, Pennsylvania, in 1974 and grew up mostly in Lakeland, Florida. After attending high school in Lakeland, he enrolled in the University of Central Florida in Orlando in 1992. Four years later, he received the degree of Bachelor of Science in civil engineering with honors. Shortly thereafter, he began working on a Master of Science degree in structures and foundations at the University of Central Florida. His thesis, which investigated the dynamic characteristics of highway bridges, was completed in May 1997.

In July of 1997, he and his wife Michelle were married in Orlando, Florida. They then moved to Gainesville, Florida, where Mark began work on a doctorate in structural engineering at the University of Florida, while Michelle continued in medical school. In 2000, he received the degree of Doctor of Philosophy from the University of Florida.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Marc I. Hoit, Chair
Professor of Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Kurtis Gurley, Cochair
Assistant Professor of Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Gary R. Consolazio
Assistant Professor of Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Fernando E. Fagundo
Professor of Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Michael Rabens
Assistant Professor of Architecture

This thesis was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 2000

_____

M. J. Ohanian
Dean, College of Engineering

_____

Winfred M. Phillips
Dean, Graduate School