Final Report

FHWA/IN/JTRP-2004/20

SIMPLIFIED LOAD DISTRIBUTION FACTOR FOR USE IN LRFD DESIGN

By

Elisa D. Sotelino
Associate Professor, Principal Investigator

Judy Liu
Assistant Professor, Principal Investigator

and

Wonseok Chung
Research Assistant

Kitjapat Phuvoravan
Research Assistant

School of Civil Engineering
Purdue University

Project No. C-36-56GGG
File No.: 7-4-58
SPR-2477

Prepared in Cooperation with the
Indiana Department of Transportation and the
U.S. Department of Transportation
Federal Highway Administration

Purdue University
West Lafayette, Indiana
September 2004

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/IN/JTRP-2004/20		

4. Title and Subtitle	5. Report Date
Simplified Load Distribution Factor for Use in LRFD Design	September 2004
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Elisa D. Sotelino, Judy Liu, Wonseok Chung, Kitjapat Phuvoravan	FHWA/IN/JTRP-2004/20

9. Performing Organization Name and Address	10. Work Unit No.
Joint Transportation Research Program 1284 Civil Engineering Building Purdue University West Lafayette, IN 47907-1284	
	11. Contract or Grant No. SPR-2477

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Indiana Department of Transportation State Office Building 100 North Senate Avenue Indianapolis, IN 46204	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared in cooperation with the Indiana Department of Transportation and Federal Highway Administration.

16. Abstract

The "S-over" equation for the load distribution factor (LDF) was first introduced in the 1930s in the AASHTO Standard. Finite element studies, however, have shown it to be unsafe in some cases and too conservative in others. AASHTO LRFD 1994 introduced a new LDF equation as a result of the NCHRP 12-26 project. This equation is based on parametric studies and finite element analyses (FEA). It is considered to be a good representation of bridge behavior. However, this equation involves a longitudinal stiffness parameter, which is not initially known in design. Thus, an iterative procedure is required to correctly determine the LDF value. This need for an iterative design procedure is perceived by practicing engineers as the major impediment to widespread acceptance of the AASHTO LRFD equation. In this study, a new simplified equation that is based on the AASHTO LRFD formula and does not require an iterative procedure is developed. A total of 43 steel girder bridges and 17 prestressed concrete girder bridges in the state of Indiana are selected and analyzed using a sophisticated finite element model. The new simplified equation produces LDF values that are always conservative when compared to those obtained from the finite element analyses and are generally greater than the LDF obtained using AASHTO LRFD specification. Therefore, the simplified equation provides a simple yet safe specification for LDF calculation.

This study also investigates the effects of secondary elements and bridge deck cracking on the LDF of bridges. The AASHTO LRFD LDF equation was developed based on elastic finite element analysis considering only primary members, i.e., the effects of secondary elements such as lateral bracing and parapets were not considered. Meanwhile, many bridges have been identified as having significant cracking in the concrete deck. Even though deck cracking is a well-known phenomenon, the significance of pre-existing cracks on the live load distribution has not yet been assessed in the literature. First, secondary elements such as diaphragms and parapet were modeled using the finite element method, and the calculated load distribution factors were compared with the code-specified values. Second, the effects of typical deck cracking and crack types that have a major effect on load distribution were identified through a number of nonlinear finite element analyses. It was found that the presence of secondary elements can result in a load distribution factor up to 40 % lower than the AASHTO LRFD value. Longitudinal cracking was found to increase the load distribution factor; the resulting load distribution factor can be up to 17 % higher than the LRFD value. Transverse cracking was found to not significantly influence the transverse distribution of moment. Finally, for one of the selected bridges, both concrete cracking and secondary elements are considered to invesitigate their combined effect on lateral load distribution. The increased LDF due to deck cracking is offset by the contributions from the secondary elements. The result is that the proposed simplified equation is conservative and is recommended for determination of LDF.

17. Key Words	18. Distribution Statement
bridges, load distribution factor, AASHTO, finite element method, secondary systems, cracking. .	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	203	

Form DOT F 1700.7 (8-69)

INDOT Research

TECHNICAL *Summary*

Technology Transfer and Project Implementation Information

TRB Subject Code: 25-1 Bridges *September 2004*
Publication No.: FHWA/IN/JTRP-2004/29, SPR-2477 *Final Report*

Simplified Load Distribution Factor for Use in LRFD Design

Introduction

The "S-over" equation for the load distribution factor (LDF) was first introduced in the 1930s in the AASHTO Standard specifications. Finite element studies, however, have shown it to be unsafe in some cases and too conservative in others. AASHTO LRFD specifications introduced a new LDF equation as a result of the NCHRP 12-26 project. This equation is based on elastic finite element analysis (FEA). It is considered to be a good representation of bridge behavior. However, this equation involves a longitudinal stiffness parameter, which is not initially known in design. Thus, an iterative procedure is required to correctly determine the LDF value. This need for an iterative design procedure is perceived by practicing engineers as the major impediment to widespread acceptance of the AASHTO LRFD equation.

Meanwhile, the FE model used in developing the LRFD LDF equation did not include some important features of bridges which may affect lateral load distribution. First, despite the presence of the secondary elements such as cross bracing, diaphragms, and parapets in bridges, these elements were not considered in the development of the AASHTO LRFD LDF equation. Second, previous research revealed a widespread presence of pre-existing cracks in concrete bridge decks. These cracks usually form even before the bridge is open to traffic. Even though deck cracking is a well-known phenomenon, the effect of deck cracking on the live load distribution has not yet been assessed.

The main objective of this study is to propose a new simplified equation that is based on the AASHTO LRFD formula and does not require an iterative procedure. The new simplified equation is intended to be at least as conservative as the LRFD equation. Additional objectives of the study are (1) to investigate the influence of secondary elements on the lateral load distribution of typical steel girder bridges; and (2) to examine the effects of deck cracking on the load distribution mechanism through nonlinear analyses.

Findings

A total of 43 steel girder bridges and 17 prestressed concrete girder bridges in the state of Indiana are selected and analyzed using a sophisticated finite element model. The LDF obtained from the FE analyses are compared with those obtained using AASHTO LRFD equation, AASHTO Standard equation, and the proposed Simplified equation. It has been found that the new simplified equation produces LDF values that are always conservative when compared to those obtained from the finite element analyses and are generally greater than the LDF obtained using AASHTO LRFD specification. Therefore, the simplified equation provides a simple yet safe specification for LDF calculation.

The effects of secondary elements and deck cracking on the LDF are investigated through case studies of several Indiana bridges. The presence of secondary elements can result in a load distribution factor up to 40 % lower than the AASHTO LRFD value. Longitudinal cracking has been found to increase the load distribution factor; the resulting load distribution factor can be up to 17 % higher than the LRFD value. Transverse cracking does not significantly influence the transverse distribution of moment. Finally, for one of the selected bridges, both concrete deck cracking and secondary elements are considered to investigate their combined effect on lateral load distribution. The increased LDF due to deck cracking is offset by the contributions from the secondary elements. The result is that the

proposed Simplified equation is conservative and is recommended for determination of LDF.

Implementation

The proposed, Simplified equation is expected to streamline the determination of LDF for bridge design without sacrificing safety. The simplified LDF equation eliminates the increased level of complexity introduced by the AASHTO LRFD equation, which has precluded its acceptance by the bridge engineering community, by removing iterative parameters. Thus, the simplified LDF minimizes undue burden on the bridge designer as well as reduce the likelihood for misinterpretation and error within the framework of the LRFD specifications. Initial, trial implementation of the Simplified LDF equation will be undertaken by the INDOT Design Division.

Contacts

For more information:

Prof. Elisa D. Sotelino
Principal Investigator
School of Civil Engineering
Purdue University
West Lafayette IN 47907
Phone: (765) 494-2228
Fax: (765) 496-1105
E-mail: sotelino@ecn.purdue.edu

Prof. Judy Liu
Principal Investigator
School of Civil Engineering
Purdue University
West Lafayette IN 47907
Phone: (765) 494-2254
Fax: (765) 496-1105
E-mail: jliu@ecn.purdue.edu

Indiana Department of Transportation
Division of Research
1205 Montgomery Street
P.O. Box 2279
West Lafayette, IN 47906
Phone: (765) 463-1521
Fax: (765) 497-1665

Purdue University
Joint Transportation Research Program
School of Civil Engineering
West Lafayette, IN 47907-1284
Phone: (765) 494-9310
Fax: (765) 496-7996
jtrp@ecn.purdue.edu
http://www.purdue.edu/jtrp

TABLE OF CONTENTS

LIST OF TABLES ..... **V**

LIST OF FIGURES ..... **VI**

CHAPTER 1. INTRODUCTION **1**

1.1 BACKGROUND & MOTIVATION 1

1.2 OBJECTIVES & SCOPE 3

1.3 METHODOLOGY 3

1.4 ORGANIZATION 4

CHAPTER 2. AASHTO GIRDER DISTRIBUTION FACTOR **6**

2.1 GENERAL 6

2.2 AASHTO-LRFD LDF DEVELOPMENT 6

2.3 LITERATURE REVIEW ON LOAD DISTRIBUTION FACTOR 10

CHAPTER 3. FINITE ELEMENT MODELING OF GIRDER BRIDGES **13**

3.1 GENERAL 13

3.2 PREVIOUS WORK 14

3.3 GRILLAGE ANALYSIS 20

3.4 SELECTED THREE-DIMENSIONAL FE MODEL 22

3.5 PRE-PROCESSING 26

3.6 POST-PROCESSING 34

3.6.1 ABAQUS Result Format *34*

3.6.2 Effective Width *35*

3.6.3 Interpolation of Results *36*

3.6.4 Moment in the Girder Section *37*

3.6.5 Moment from Beam Analysis *40*

3.6.6 LDF Calculation *40*

3.7 Comparison with Experimental Results 45
3.7.1 Elk River Bridge 45
3.7.2 Michigan Bridge 51
3.8 Summary 55

CHAPTER 4. MULTI-PARAMETER COMPARISON 56

4.1 General 56
4.2 Indiana Bridge Database 56
4.3 Applicable Range 57
4.4 Representative Bridges 58
4.5 Sensitivity of Bridge Parameters 59

CHAPTER 5. POSTULATION OF SIMPLIFIED EQUATION 82

5.1 Parameter Selection 82
5.2 Postulation of Simplified Formula 87
5.3 Comparison of LDFs 91

CHAPTER 6. PRESTRESSED CONCRETE BRIDGES 98

6.1 Introduction 98
6.2 Finite Element Modeling of PC Girder Bridges 98
6.3 Washington Field Test 105
6.4 LDF Comparisons of PC Girder Bridges 108

CHAPTER 7. EFFECTS OF SECONDARY ELEMENTS AND DECK CRACKING ON LOAD DISTRIBUTION FACTOR 113

7.1 Introduction 113
7.2 Research Methodology 114
7.3 Finite Element Modeling of Secondary Elements 117
7.4 Concrete Crack Model 125
7.5 Load Distribution Factor Comparisons 129
7.5.1 Effects of Secondary Elements 129
7.5.2 Effects of Bridge Deck Cracking 136

7.6 SUMMARY 143

CHAPTER 8. CONCLUSIONS AND RECOMMENDATIONS 144

8.1 SUMMARY 144

8.2 CONCLUSIONS 145

8.3 RECOMMENDATIONS FOR SIMPLIFIED SPECIFICATIONS 147

8.4 SIMPLIFIED LOAD DISTRIBUTION FACTOR SPECIFICATION 148

REFERENCES 149

APPENDIX A: PRE PROCESSOR CODE LIST 154

APPENDIX B: POSTPROCESSOR MANUAL 175

LIST OF TABLES

Table 3.1 Survey of Bridge Analysis Studies on Live Load Distribution 17
Table 3.2 Comparison of Distribution of Moments 47
Table 3.3 Michigan Test Bridge Information 52
Table 3.4 Girder Cross Section Dimensions for Michigan Test Bridge 52
Table 4.1 Representative Indiana Steel Girder Bridges 61
Table 4.2 Summary of Statistical Study 63
Table 4.3 Selected Mean Bridges 63
Table 5.1 Applicable Range for the New Simplified LDF Equation 88
Table 5.2 Wheel Load Distribution Formulas for Concrete Slab on Steel Girder Bridges and Skew Correction Factor (US Customary Units*) 89
Table 5.3 Wheel Load Distribution Formulas for Concrete Slab on Steel Girder Bridges and Skew Correction Factor (SI Units**) 90
Table 6.1 Selected Indiana Bridges 100
Table 6.2 Representative Indiana Prestressed Concrete Girder Bridges 109
Table 7.1 Selected Indiana Bridges (Secondary Elements) 115
Table 7.2 Selected Indiana Bridges (Pre-existing Cracks) 115
Table 7.3 List of Strain Gauges (US 52 Bridge) 119
Table 7.4 Load Cases for the US 52 Bridge Test 119

LIST OF FIGURES

Figure 3.1 Eccentric Beam Model. 18
Figure 3.2 Detailed Beam Model (Brockenbrough 1986). 18
Figure 3.3 Solid Deck Model (Tarhini and Frederick 1992). 19
Figure 3.4 Grillage Analysis (Hambly 1991) 21
Figure 3.5 Improved Eccentric Beam Model..... 25
Figure 3.6 Flowchart of the Procedure Used for the Determination of the LDF..... 29
Figure 3.7 AASHTO HS-20 Design Truck (AASHTO 2002)..... 30
Figure 3.8 View of Tire Contact Area of AASHTO HS20 truck (Dimensions are 4 by 10 in. and 8 by 20 in. for the Front and Rear Tires, Respectively)..... 31
Figure 3.9 Typical Bending DOFs and Node Numbering of Shell Elements..... 32
Figure 3.10 Discretization of Patch Load. 32
Figure 3.11 Equivalent Nodal Force Computation Algorithm. 33
Figure 3.12 Discretization Error of Patch Load..... 33
Figure 3.13 ABAQUS Notation of Beam Element..... 41
Figure 3.14 ABAQUS Notation of Shell Element..... 41
Figure 3.15 Interpolation of Finite Element Results for Shell Elements..... 42
Figure 3.16 Moments in the Girder Section for the Determination of the LDF. 43
Figure 3.17 The Beam Analysis Analogous to the Bridge Geometry. 44
Figure 3.18 Elk River Bridge..... 48
Figure 3.19 Cross Section of Elk River Bridge. 49
Figure 3.20 Longitudinal Dimensions and Loading Locations for Elk River Bridge. 49
Figure 3.21 Moment Envelope of Girder Section no. 2..... 50
Figure 3.22 Cross Sectional View of Michigan Test Bridge (Eom and Nowak 2001). ... 53
Figure 3.23 Layout of Strain Gauges of Michigan Test Bridge (Eom and Nowak 2001).53
Figure 3.24 Test Truck Configurations..... 54

Figure 3.25 Bottom Flange Strains at Mid-span (Michigan Test). 54
Figure 4.1 Indiana Bridge Database (Owner). 64
Figure 4.2 Indiana Bridge Database (Type of Service). 64
Figure 4.3 Indiana Bridge Database (Material). 65
Figure 4.4 Indiana Bridge Database (Type of Design). 65
Figure 4.5 Indiana Bridge Histogram (Number of Spans). 66
Figure 4.6 Indiana Bridge Histogram (Span Length). 66
Figure 4.7 Maximum Span Length. 67
Figure 4.8 Indiana Bridge Histogram (Transverse Width), ... 68
Figure 4.9 Indiana Bridge Histogram (Skew Angle). 68
Figure 4.10 Indiana Bridge Histogram (Traffic Lane) 69
Figure 4.11 Indiana Bridge Histogram (Year Built or Reconstructed). 69
Figure 4.12 Representative Bridge Scattergram (NBI Parameters, Width vs. Span) 70
Figure 4.13 Representative Bridge Scattergram (NBI Parameters, Number of Spans vs. Width) 70
Figure 4.14 Representative Bridge Scattergram (NBI Parameters, Skew vs. Number of Spans) 71
Figure 4.15 Representative Bridge Scattergram (NBI Parameters, Skew vs. Width). 71
Figure 4.16 Representative Bridge Histogram (Girder Spacing) 72
Figure 4.17 Representative Bridge Histogram (Number of Girders). 72
Figure 4.18 Representative Bridge Scattergram (Girder Spacing vs. Span) 73
Figure 4.19 Representative Bridge Scattergram (Girder Spacing vs. Stiffness) 73
Figure 4.20 Representative Bridge Scattergram (Stiffness vs. Span). 74
Figure 4.21 Representative Bridge Scattergram (Girder Depth vs. Span). 74
Figure 4.22 Comparison of Specifications (Girder Spacing) 75
Figure 4.23 Comparison of Specifications (Span Length) 76
Figure 4.24 Comparison of Specifications (Slab Thickness) 77
Figure 4.25 Comparison of Specifications (Longitudinal Stiffness Parameter). 78
Figure 4.26 Comparison of Specifications (Unitless Inertia). 79
Figure 4.27 Comparison of Specifications (Skew Angle). 80

Figure 4.28 Wheel Load Distribution of Indiana Representative Bridges..... 81
Figure 5.1 Scattergram of Longitudinal Stiffness Parameter and Span Length (from NCHRP 12-26 Database and Indiana Database). 84
Figure 5.2 Calibrated Longitudinal Stiffness Parameter..... 85
Figure 5.3 Exponential Trend Line. 86
Figure 5.4 LDF Comparisons Between Simplified Equation and LRFD Equation..... 93
Figure 5.5 LDF Comparisons Between Simplified Equation and Standard Equation..... 94
Figure 5.6 Positive Moment LDF Comparisons. 95
Figure 5.7 Negative Moment LDF Comparisons. 96
Figure 5.8 LDF Comparisons for Out of Range Bridges..... 97
Figure 6.1 Eccentric Beam Model Including Prestressing Tendon 101
Figure 6.2 PC Bridge Model A 102
Figure 6.3 PC Bridge Model B 102
Figure 6.4 PC Bridge Model C 103
Figure 6.5 Comparisons of Live Load Distribution Factor..... 104
Figure 6.6 Washington Bridge Girder (Barr et al. 2001)..... 106
Figure 6.7 Mid-span Moment Due to Truck Load Located on the Exterior Girder 107
Figure 6.8 Mid-span Moment Due to Truck Load Located on the First Interior Girder 107
Figure 6.9 Scattergram of Longitudinal Stiffness Parameter and Span Length 110
Figure 6.10 Positive Moment LDF Comparisons. 111
Figure 6.11 Negative Moment LDF Comparisons. 112
Figure 7.1 Standard Drawing of 33” Concrete Bridge Barrier 116
Figure 7.2 Finite Element Model 120
Figure 7.3 Layout of Strain Gauges at Mid-span (Nebraska Bridge) 121
Figure 7.4 Comparisons of FE results to Experimental Results (Nebraska Bridge) 122
Figure 7.5 Cross Section of Instrumented Span (US 52 Bridge)..... 123
Figure 7.6 Comparisons of FE Results to Experimental Results (US 52 Bridge) 124
Figure 7.7 State Determination for Cracked Concrete..... 128
Figure 7.8 Lateral Distribution of LDF (Secondary Elements) 131
Figure 7.9 Effect of Secondary Elements on LDF..... 132

Figure 7.10 Comparisons of FEM LDF to LRFD LDF 133
Figure 7.11 Comparisons of FEM LDF to Simplified LDF 134
Figure 7.12 Comparisons of FEM LDF to Standard LDF 135
Figure 7.13 Effect of Deck Cracking on LDF 138
Figure 7.14 Comparisons of FEM LDF vs. LRFD LDF 139
Figure 7.15 Comparisons of FEM LDF vs. Simplified LDF 140
Figure 7.16 Comparisons of FEM LDF vs. Standard LDF 141
Figure 7.17 Lateral Distribution of LDF for Bridge 16 142

CHAPTER 1. INTRODUCTION

1.1 Background & Motivation

In bridge design, the value of the maximum moment in the girders is necessary in the determination of the bridge section. The problem is three-dimensional and involves complex behavior of load transfer from concrete slab to steel girder. The AASHTO bridge specification suggests many methods to analyze bridges, i.e., finite element analysis, grillage analysis, and a load distribution factor (LDF) equation.

The LDF equation is introduced to facilitate in determination of maximum moment in the girders. Finite element analysis (FEA) is considered to be an accurate method, but it requires much effort in data preparation, bridge modeling and analysis, and interpretation of results. With the LDF equation, the maximum moment in the girders is obtained by multiplying the moment from a one-dimensional bridge analysis by the value obtained from the LDF equation.

The wheel load distribution factor from the "S-over" equation, the AASHTO standard equation (AASHTO 1996), for concrete slab on steel girder bridges with two or more design lanes loaded is

$$LDF = \frac{S}{5.5} \quad \text{(US customary unit)}$$
$$LDF = \frac{S}{1676} \quad \text{(SI unit)} \tag{1.1}$$

where S is girder spacing (ft, mm). The S-over equation, first introduced in 1930s, involves only one parameter. Although the S-over equation is simple to use, it is considered to be unsafe for some bridges and too conservative for others.

In 1994, a more accurate LDF equation was introduced in the AASHTO LRFD code (AASHTO 1998). This equation was based on FEA and statistics. The wheel load distribution factor equation from AASHTO LRFD for concrete slab on steel girder bridges with two or more design lanes loaded is

$$LDF = 0.15 + \left(\frac{S}{3}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{12Lt_s^3}\right)^{0.1} \qquad \text{(US customary unit)}$$
$$LDF = 0.15 + \left(\frac{S}{914}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{Lt_s^3}\right)^{0.1} \qquad \text{(SI unit)} \tag{1.2}$$

where S is girder spacing (ft, mm), L is span length (ft, mm), $K_g = n(I+Ae^2)$ is longitudinal stiffness (in^4, mm^4), t_s is slab thickness (in, mm), n is modular ratio between steel and concrete, I is girder stiffness (in^4, mm^4), A is girder area (in^2, mm^2), and e is eccentricity between centroids of girder and slab (in, mm).

The AASHTO LRFD equation is considered to represent well the actual behavior of bridges (Zokaie 2000 and Zokaie et al. 1991). However, since the equation requires parameters that are not known until girder selection, an iterative design procedure is necessary. These parameters are longitudinal stiffness, K_g, and slab thickness, t_s.

Compared to the AASHTO LRFD, the AASHTO Standard equation tends to give unconservative LDF when bridge span length and girder spacing are relatively small, and gives overly conservative LDF when bridge span length and girder spacing are relatively large. Although the AASHTO Standard equation is simple, it yields inaccurate LDF values. Conversely, the AASHTO LRFD equation produces accurate results, but it is considered to be cumbersome in practice.

The FE model used in developing the LRFD LDF equation did not include some important features of bridges which may affect lateral load distribution. First, despite the presence of the secondary elements such as cross bracing, diaphragms, and parapets in bridges, these elements were not considered in the development of the AASHTO LRFD LDF equation. Previous parametric studies (Eamon and Nowak 2004; Eamon and Nowak 2002; Mabsout et al. 1997) showed that consideration of secondary elements has a significant effect on lateral load distribution. Consequently, the AASHTO LRFD equation often provides overly conservative results. Second, previous research (Frosch et

al. 2003; French et al. 1999) revealed a widespread presence of pre-existing cracks in concrete bridge decks. These cracks are usually formed even before the bridge is open to traffic. Direction of cracking is typically transverse with respect to traffic direction but longitudinal cracking has also been observed. Even though deck cracking is a well-known phenomenon, the effect of deck cracking on the live load distribution has not yet been assessed.

1.2 Objectives & Scope

The main objective of this study is to propose a new simplified LDF equation for concrete slab on steel girder bridges that captures the load distribution behavior but does not require an iterative design procedure. In the proposed specification, the parameters in the AASHTO LRFD equation that introduce the need for iteration are eliminated. The new simplified equation is intended to be at least as conservative as the LRFD equation. The scope of the research is initially limited to concrete slab on steel I-girder bridges, and later extended to prestressed concrete girder bridges.

Additional objectives of the study are (1) to investigate the influence of secondary elements on the lateral load distribution of typical steel girder bridges; and (2) to examine the effects of deck cracking on the load distribution mechanism through nonlinear analyses.

1.3 Methodology

The approach adopted in this work includes the development of a reliable three-dimensional finite element model and the postulation and verification of the new simplified LDF equation. First the applicable range for each parameter is selected. Bridges that have parameters inside the applicable range are used in the postulation of the new simplified equation and in the verification phase. The new simplified LDF equation is formulated based on the AASHTO LRFD equation. The formulation involves the

elimination of the parameters that create the need for an iterative design procedure. Various finite element models for slab-on-girder bridges are studied. An appropriate model is selected and employed to verify the new simplified LDF equation, ensuring its safety. The finite element model is then extended to prestressed concrete girder bridges and is used to further validate the postulated equation for these bridges. To expedite the numerous analyses of both slab-on-girder and presteressed concrete bridges, pre- and post-processors are developed. This alleviates a number of repetitive procedures required in bridge analysis.

In order to examine the effects of secondary elements and deck cracking, a reliable three-dimensional finite element model including secondary elements and a concrete cracking constitutive model was developed. Then, eighteen Indiana bridges were selected and analyzed using the model. The load distribution factors obtained from the FE analyses were compared with those obtained using AASHTO LRFD equation, AASHTO Standard equation, and Simplified equation.

1.4 Organization

Chapter 2 presents the historical background of the AASHTO wheel load distribution factor. The derivation of the new AASHTO-LRFD formulas is then explained based on the NCHRP 12-26 project (Zokaie et al. 1991a, 1991b). The limitations of the NCHRP 12-26 project are presented. Prior studies related to the live load distribution of slab-on-girder bridges are also summarized.

Chapter 3 describes the different analytical modeling techniques for bridge analysis. First, plane grillage analysis, known as AASHTO Level II analysis, is introduced, and its limitations are summarized. FE modeling techniques of composite steel girder bridges are then discussed with a concentration on overall flexural behavior. The selection of the adopted finite elements and the numerical technique for modeling composite action is also discussed. Finally, the chosen FE model is verified by comparing the results of linear elastic analysis to experimental test results.

In Chapter 4 the applicable range is defined based on the full range of bridge structures in Indiana. The Indiana representative bridges are then selected. Finally, parametric studies of the AASHTO LRFD equation are performed.

A new Simplified wheel load distribution factor (LDF) equation, based on the current AASHTO-LRFD LDF formula, is postulated in Chapter 5. The accuracy and applicability of the Simplified equation are demonstrated through comparisons of LDF calculated by AAHSTO-Standard, AASHTO-LRFD, and AASHTO level three analysis, namely finite element (FE) analysis.

Chapter 6 discusses the finite element modeling techniques for prestressed concrete (PC) girder bridges. An appropriate model is selected. Indiana representative PC girder bridges are then analyzed. The simplified LDF equation is further evaluated for PC girder bridges.

Chapter 7 examines the effects of secondary elements and bridge deck cracking on the lateral load distribution of girder bridges. The results from case studies involving eighteen Indiana bridges are presented. Finally, a summary and conclusions are provided in Chapter 8.

CHAPTER 2. AASHTO GIRDER DISTRIBUTION FACTOR

2.1 General

AASHTO codes utilize the LDF to simplify the computation of load distribution. The maximum girder design moment can be calculated by the multiplication of the LDF with the maximum moment from one beam analysis. First, the historical background of the AASHTO wheel load distribution factor is introduced. The derivation of the new AASHTO-LRFD formulas based on the NCHRP (National Cooperative Highway Research Program) 12-26 project (Zokaie et al. 1991a, 1991b) is discussed. The limitations of the NCHRP 12-26 project are presented. Prior studies related to the live load distribution of slab-on-girder bridges are also summarized.

2.2 AASHTO-LRFD LDF Development

Empirical load distribution factors from Newmark's research (1938) have been used in AASHTO-Standard Bridge Specifications (1996) without major modification since the 1930's. However, many piecemeal changes in design codes have led to inconsistencies in the procedure. These inconsistencies include changes of design lane width and multiple presence factors.

In the AASHTO-Standard specification, the formulas were developed for the interior girders of simply supported bridges. A single parameter, girder spacing (S), was used for determining wheel load distribution factors in the form of S/D, where D is a constant based on the bridge type. These formulas were developed for straight and right-angled bridges. It has been reported that these formulas produce valid results for a certain

ranges of bridge geometry and lose accuracy rapidly as geometry parameters are varied. For this reason, these formulas have been criticized by bridge engineers. They result in values that are too conservative for long span bridges. In short span bridges with small girder spacing, however, they lead to unconservative results.

The NCHRP 12-26 project was initiated to address the controversy in the live load distribution formulas in the AASHTO-Standard Specification. Many bridge engineers claimed that other bridge parameters in addition to girder spacing, such as bridge geometric dimensions and material properties, have an effect on the lateral load distribution. A large number of modern bridges also require the wheel load distribution factor for skewed supports, continuous over interior supports, and exterior girders.

Three levels of analysis methods were considered in the NCHRP 12-26 project. The Level One analysis method used simplified formulas to predict the lateral load distribution. Level Two analysis methods involved graphical methods, influence surfaces, and plane grillage analyses. Level Three analysis methods are the most accurate and involve the detailed finite element modeling of bridge superstructures.

As part of the NCHRP 12-26 project, a database of nationwide bridges was constructed to determine the average bridge. Finite element (Level Three) or grillage analysis (Level Two) methods were then used to determine the simplified load distribution formulas (Level One) through a parametric study. The parameters sensitive to the lateral load distribution under AASHTO HS-20 design truck vehicles were identified based on the selected finite element modeling technique. Basic formulas were then developed including bridge parameters such as girder spacing, span length, girder inertia, and slab thickness. Multiple lane reduction factors were built into the basic equations.

The more accurate formulas developed in the NCHRP 12-26 project, as described earlier, have been adopted since the first edition of AASHTO-LRFD specification (1998). The wheel distribution factor (LDF) for a bending moment in steel girder bridges for interior beams is given by:

$$g_{\text{interior}} = 0.15 + \left(\frac{S}{3}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{12Lt_s^3}\right)^{0.1} \qquad \text{for two or more design lanes loaded}$$

where S is the girder spacing in feet, L is the span length in feet, t_s is the concrete slab thickness in inches, and K_g is the longitudinal stiffness parameter = $n(I + Ae_g^2)$. In this formula, n is the modular ratio between beam and deck, I is the moment of inertia of the girder in in^4, A is the area of girder, and e_g is the distance between the neutral axis of the girder and the slab in inches. For exterior girders, the LDF is given by:

$$g_{exterior} = e \cdot g_{\text{interior}} \qquad \text{for two or more design lanes loaded}$$

where

$e = 0.77 + \frac{d_e}{9.1}$ and d_e is the distance from the exterior beam to the exterior of the curb or traffic barrier in feet.

AASHTO-LRFD also includes several extensions to the basic LDF such as continuity and skew effect. According to the summary of NCHRP research by Zokaie (2000), the wheel load distribution factors in continuous bridges are slightly higher than simply supported bridges and that the average value of the adjacent spans are appropriate to use as a parameter. The skew effect was also studied, and it was found that in skew bridges, the moment is smaller and the shear at obtuse corners is larger when compared to right-angled bridges. The skew reduction factor of the wheel load distribution factor for moment is given by

$$f = 1 - c_1(\tan\theta)^{1.5}$$

where $c_1 = 0.25\left(\frac{K_g}{Lt_s^3}\right)^{0.25}\left(\frac{S}{L}\right)^{0.5}$ and θ is the skew angle in degrees. If the skew angle is less than 30 degrees, c_1 is taken as 0, and if the skew angle is greater than 60 degrees, the skew angle is taken as 60 degrees.

Figure 2.1 shows the comparison between LDFs from the AASHTO Standard and AASHTO LRFD equations. Note that the third term in the AASHTO LRFD equation, $K_g/12Lt_s^3$ for US customary units and K_g/Lt_s^3 for SI units, is assumed to be equal to unity as recommended for a first trial in design. For most bridges, this term ranges from 0.85 to 1.10. As seen in Figure 2.1, compared to the AASHTO LRFD, the AASHTO Standard

equation tends to give unconservative LDF when bridge span length and girder spacing are relatively small, and overly conservative LDF when bridge span length and girder spacing are relatively large.

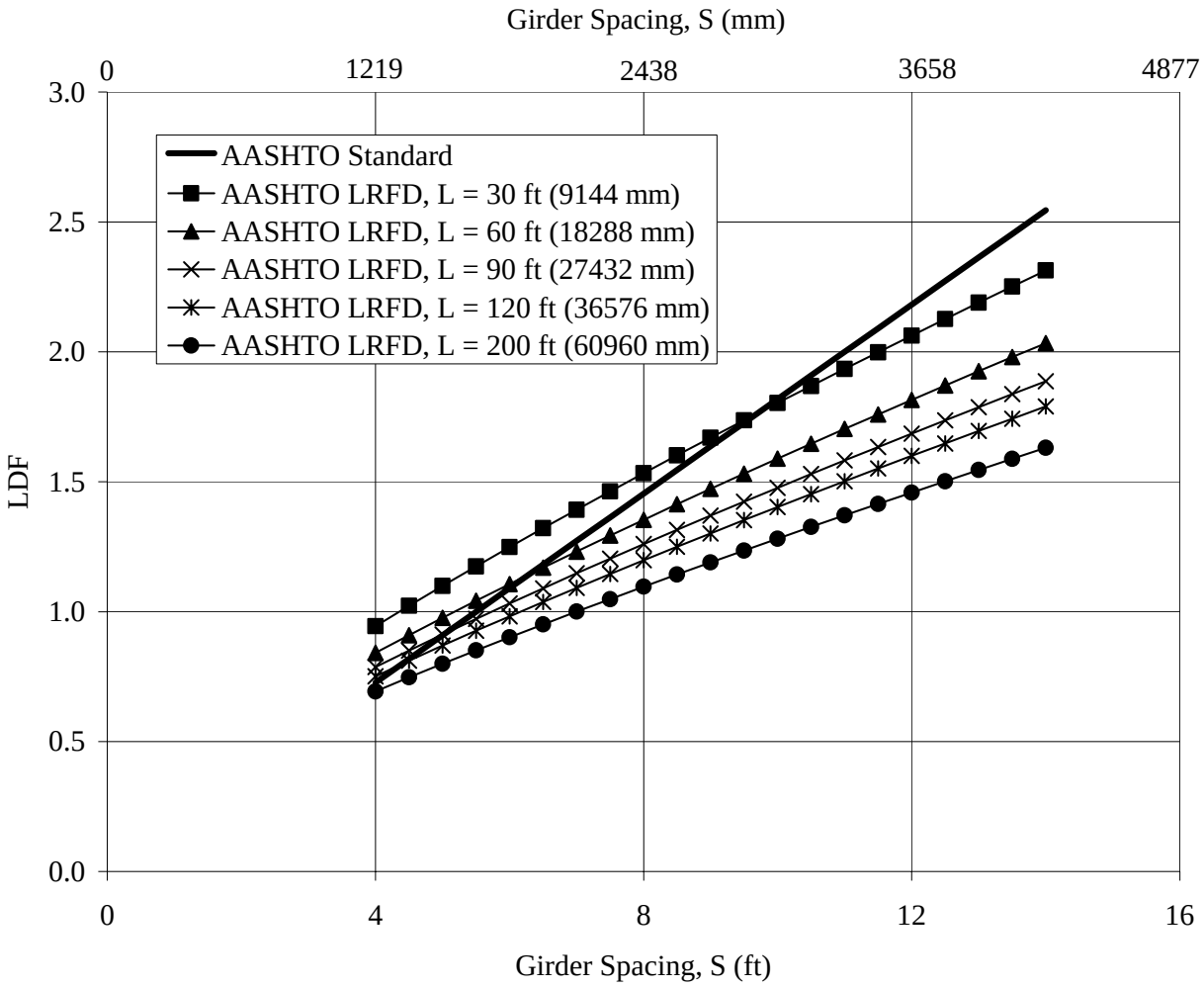


Figure 2.1 Comparison of LDFs from AASHTO Standard and AASHTO LRFD Equations. (Unitless Stiffness Term in AASHTO-LRFD Set to Unity)

2.3 Literature Review on Load Distribution Factor

The AASHTO-LRFD wheel load distribution factors were developed under certain assumptions. It was assumed that the girder spacing was evenly distributed and that all girder properties were the same. The design vehicle for developing the formulas was assumed to be the HS-20 design truck. It was also presumed that the thickness of slab was not varied. Recent research (Mabsout et al. 1997b, 1998; Tabsh and Tabatabai 2001; Chen 1995a, 1995b) on lateral distribution of live load has focused on the effects of parameters that are not considered in the current AASHTO-LRFD formulas. Experimental studies (Kennedy and Grace 1983; Kim and Nowak 1997; Eom and Nowak 2001; Barr et al. 2001) have been performed to validate the accuracy of the AASHTO-LRFD wheel load distribution formulas. Unless otherwise specified, the FE method was adopted to analyze bridge behavior. The literature survey on FE model is described in Chapter 3 of this report.

Tabsh and Tabatabai (2001), Goodrich and Puckett (2000) investigated the effects of truck configuration on the wheel load distribution and proposed modification factors for AASHTO-LRFD formulas to account for oversized truck loading. Oversized vehicles with a gauge larger than standard width were considered since formulas in AASHTO-LRFD code were developed for a specific truck type. Tabsh and Tabatabai (2001) found through a parametric study that the wheel load distribution factors under an oversized load were lower than those of AASHTO-LRFD. It was determined that the effect of gauge width on shear distribution was larger than on flexural distribution.

The wheel load distribution formulas in the current code are developed for uniform girder spacing. A series of studies by Chen (1995a, 1995b) proposed a method that predicts wheel load distribution factors on unevenly spaced bridge girders. While it is not desirable, a large number of bridges have unequally distributed girder spacing for various reasons. These are mainly due to modifications of existing bridges and geometric restrictions. It was found that the formulas in AASHTO-LRFD were not appropriate in these cases and that the distribution factors for these types of bridges were significantly affected by the effect of skew angle.

The effect of skewed support on the wheel load distribution factor was studied by a number of investigators: (Marx 1985; Bishara et al. 1993; Khaleel and Itani 1990; Barr et al. 2001). Khaleel and Itani (1990), and Barr et al. (2001) studied the load distribution for continuous prestressed concrete girder bridges, by varying the skew angle. Bishara et al. (1993) developed load distribution provisions for simply supported I-girder bridges. Verification studies were performed using experimental data. It was generally found that the presence of skewed supports decreased the distribution factor for interior girders and increased the distribution for exterior girders. At larger skew angles, a sudden decrease in the wheel load distribution was reported.

The influence of edge stiffening on bridge load distribution is investigated by Eamon and Nowak (2002) and Mabsout et al. (1997). The presence of parapet, barrier, and sidewalk gives the effect of stiffening and carries more load by reducing the load effects in the interior girders. Eamon and Nowak (2002) used solid elements to represent the parapet and barriers while Mabsout el al. (1997) used shell elements. Both models idealized the parapet to behave integrally with the deck slab. They found that the edge stiffening effect could reduce wheel load distribution factor and increase the load-carrying capacity.

A sensitivity study of continuous bridges on live load distribution was conducted in the NCHRP 12-26 project, but the correction for this effect was not included in AASHTO-LRFD. Instead of this correction factor, the continuity effect was embedded in the definition of span length. In the negative moment zone, the span length was the average of the adjacent spans. Mabsout et al. (1998) examined the influence of continuity on live load distribution for two span continuous steel I-girder bridges, and Barr et al. (2001) investigated three span continuous prestressed concrete girder bridges. Zokaie (2000) and Barr et al. (2001) found that the continuity of supports slightly increases wheel load distribution.

Nowak et al. (1999) performed field tests of five simply supported steel I-girder bridges. All bridges had short-span and small girder spacing. 11-axle trucks, which were twice as heavy as the HS-20 load, were placed to cause the maximum moment. Strain data were measured from at the bottom flanges of all girders at mid-span. The filtered

static strain data were obtained from crawling speed and regular speed tests at each girder at the same section along the length of the bridge. The LDF was calculated from the ratio of the static strain at the girder to the sum of static strains of all of the girders. They found that the field measurement results were sufficiently lower than those specified by both AASHTO specifications.

Eom and Nowak (2001) tested 17 steel-girder bridges with simply supported spans from 10 to 45 m. The methodology of the tests was the same as those used in Nowak et al. (1999). The results were compared with the code-specified values and FE results. They found that measured values were consistently lower than FE results and AASHTO code values.

CHAPTER 3. FINITE ELEMENT MODELING OF GIRDER BRIDGES

3.1 General

The response of a bridge under live load is important for both design and evaluation purposes. This is because this knowledge enables the engineer to find the strength and serviceability of a given superstructure. However, determining an accurate load distribution is difficult because of the complexity of bridge structures. In practice, under extensive simplifying assumptions, grillage analysis has primarily been used to determine overall bridge behavior. The grillage analysis method is inexpensive and easy to implement and comprehend, thus it has been favored over finite element (FE) analysis in the field of bridge engineering.

However, grillage analysis has serious limitations. Using this method, it is impossible to model important physical phenomena, such as the interaction between girders and deck slab, support location, and shear lag. This limitation comes from the fact that in grillage analysis structural members lie in one plane only. With the advances in the computer technology and modern finite element (FE) programs with user-friendly graphical interfaces, three-dimensional FE analysis method is replacing grillage analysis, even for more straightforward bridge analyses.

The objective of this chapter is to review previously proposed analytical models and to present a new three-dimensional FE model. All of these models involve detailed modeling of the bridge superstructure. As mentioned earlier, the exact load distribution factor often may not be found analytically or experimentally because of the complexity of bridge structures. However, since the actual behavior of a bridge structure is three-

dimensional in nature, a three-dimensional FE discretization scheme is the most appropriate analytical model to analyze this behavior.

It should be noted that some of the FE bridge models available in the literature introduce geometric errors and/or compatibility errors. This can potentially result in incorrect representation of flexural behavior. In this study, several FE modeling techniques were thoroughly investigated in order to avoid modeling errors by employing displacement transformation and a proper selection of finite elements.

The FE model chosen for this study was developed based on three-dimensional discretization. This model is capable of including physical behavior, such as composite action and the eccentricity effect between the slab deck and the girder. Using this model, it is also possible to capture shear lag, which is important in order to understand bridge deck behavior.

This chapter describes various analytic modeling techniques for bridge analysis. First, plane grillage analysis, known as AASHTO Level II analysis, is introduced, and its limitations are summarized. FE modeling techniques of composite steel girder bridges are then discussed, with particular attention to overall flexural behavior. The selection of the adopted finite elements and the numerical technique for modeling composite action are also discussed. The FE modeling schemes are verified by comparing the results of linear elastic analyses to experimental test results. Finally, the FE model chosen for this study is presented.

3.2 Previous Work

Three dimensional finite element (FE) analysis enables bridge engineers to determine the distribution of wheel loads more accurately than empirical or restricted code formulas. The literature review of finite element modeling for bridge superstructure includes a survey of 15 papers on load distribution studies published over the past 15 years. Table 3.1 is prepared using the results of the survey and is referred to throughout this section. The literature survey indicates that more than 85 % of current research

utilizes the FE method as an analysis tool over the grillage analysis or other simplified methods.

The three-dimensional FE modeling of bridges is generally divided into three categories: Eccentric beam model, Detailed beam model, and Solid deck model. The primary structural members for the distribution of live load are deck slab and girders. The modeling techniques of primary members are considered even though a number of studies also include secondary member modeling. Similarly to the research by Zokaie et al. (1991) current research on load distribution considers all materials as linear elastic.

A majority of studies (Barr et al. 2001; Chen 1999; Ebeido and Kennedy 1996; Shaway and Huang 2001; Zokaie 1991a, 1991b; Marx 1985) utilized the eccentric beam model to idealize the bridge superstructures as shown in Figure 3.1. In this paper, the concrete slab is modeled as quadrilateral shell elements that incorporate both membrane and bending actions. Steel girders are modeled using eccentrically connected two-node beam elements. The eccentricity of the girders is taken into account though the use of rigid links between the centroid of the concrete slab and the centroid of the steel girders.

Gupta and Ma (1977) and Balmer (1978) investigated the incompatibility between the beam element and the shell element in the eccentric beam model. The commonly used four-node thin shell element assumes that the shape functions are such that in axial and flexural modes of deformations are uncoupled. The axial mode is interpolated using linear shape functions, and the flexural mode is characterized by a Hermitian cubic for transverse displacement shape function. These researchers pointed out that the quadratic expression of the rotation in the linear constraint equation of axial displacement leads to inconsistencies. As a result, large errors in the deflections and stresses of stiffened plates occur.

Miller (1980) eliminated this incompatibility by adding internal degrees of freedom at the middle of element edges so that axial displacements were compatible with quadratic shapes. Marx (1985) and Sadek and Tawfik (2000) used nine-node Lagrangian elements based on Mindlin plate theory in the shell formulation and three-node Timoshenko beam elements with shear deformation. Since all shape functions are quadratic the compatibility of the axial displacement field between the shell and the beam

elements is ensured. Khaleel and Itani (1990), Chan and Chan (1999), and Prusty and Satsangi (2001) idealized the deck slab using an eight-node serendipity shell element, but the approach was basically similar to the one in Marx's study.

Brockenbrough (1986) and Tabsh and Tabataba (2001) modeled deck slab using four node shell elements that included membrane and bending effects. Each steel I-girder was divided into flange and web parts as shown in Figure 3.2. Each flange of the girder was idealized by Euler beam elements, and the web was modeled by the four-node shell elements. Bishara et al. (1993) adopted the same modeling technique to represent the girder, but they used three-node thin plate triangular elements. Fu and Lu (2003) idealized the flange of the steel girder with plate elements and the web by plane stress elements. The eccentricity between concrete deck and steel girder flange was modeled by a rigid link, but no details were given regarding the effect of the inconsistency in axial displacement fields between the beam and shell elements.

Tarhini and Frederick (1992), Mabsout et al. (1997), and Eom and Nowak (2001) used the three-dimensional solid elements with three degrees of freedom at each node, which have linear shape functions, to model deck slab as shown in Figure 3.3. The steel girder flanges and web were modeled by quadrilateral shell elements. By imposing no releases between the shell elements and beam elements, the composite behavior between concrete deck and steel girder was simulated. For non-composite action, Tarhini and Frederick (1992) placed three linear spring elements at the interface nodes in three orthogonal directions. The spring stiffness was then selected based on the amount of slip expected. No details were given about the interface compatibility between slab solid element and flange shell element in the full composite action case. The main drawback in the model of Tarhini and Frederick was the use of only one linear solid brick element (8 node) throughout the plate thickness direction. Many solid elements were required throughout thickness direction to accurately simulate the flexural behavior, since the solid element has a linear strain variation. Cook et al. (1989) also recommended that solid elements should not be used to model plates because of the computational cost associated with them.

Table 3.1 Survey of Bridge Analysis Studies on Live Load Distribution

Analytical method	Finite element analysis: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15 Grillage analysis: 10, 14
Finite Element Model	**Category** Eccentric beam model: 1, 4, 5, 7, 8, 9, 11, 14, 15 Detailed beam model: 2, 3, 8, 12 Solid deck model: 6, 8, 13 **Deck FE element** Solid (8 node): 6, 8, 13 Shell (3 node): 2 Shell (4 node): 1, 3, 4, 5, 8, 10, 11, 12, 14 Shell (8 node): 7, 15 Shell (9 node): 9 **Girder FE element** Beam (2 node): 1, 4, 5, 8, 11, 14 Beam (3 node): 7, 9 Flange (2 node beam) + Web (4 node shell): 2, 3, 8, 12 Flange, Web (4 node shell): 6, 8, 13 **Secondary member considered** (diaphragms, cross bracings, stiffeners, curb, parapets) Yes: 1, 2, 5, 6, 11, 12 No: 3, 4, 7, 8, 9, 13, 14 **Rigid link** Yes: 1, 2, 3, 4, 5, 7, 8, 9, 11, 12, 14, 15 No: 6, 8, 10 **Composite action** Full: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15 None: 8 **Proof of FE model** Experiment: 1, 2, 5, 6, 14 N/A: 3, 4, 7, 8, 9, 12, 13 **Program used** Commercial: 1, 8 (SAP), 2, 4 (ADINA), 3 (NASTRAN), 5, 6 (ABAQUS), 12 (ALGOR), 13 (ICES STRUDL II), 14 (GENDEK5A) N/A: 7, 9, 10, 11
Material	**Girder material** Steel: 2, 3, 4, 5, 6, 8, 9, 12, 13, 14 Concrete: 1, 4, 7, 9, 11, 14 **Constitutive model** Linear elastic

1. Barr et al. (2001)
2. Bishara et al. (1993)
3. Brockenbrough (1986)
4. Chen (1999)
5. Ebeido and Kennedy (1996)
6. Eom and Nowak (2001)
7. Khaleel and Itani (1990)
8. Mabsout et al. (1997)
9. Marx (1985)
10. Schwarz and Laman (2001)
11. Shahawy and Huang (2001)
12. Tabsh and Tabataba (2001)
13. Tarhini and Frederick (1992)
14. Zokaie et al. (1991)
15. Chan and Chan (1999)

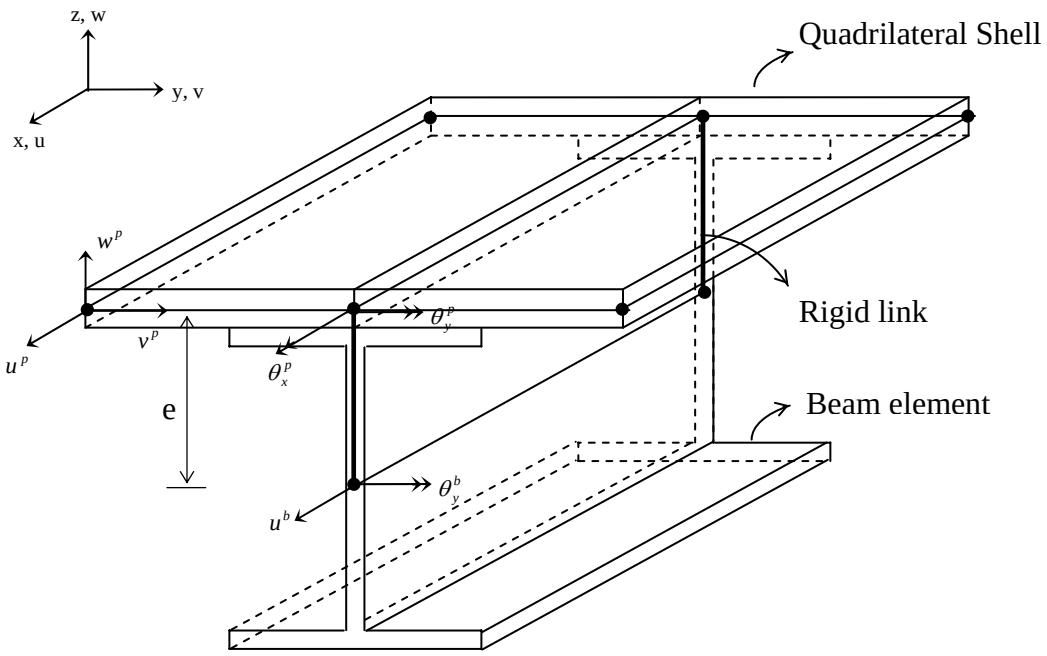


Figure 3.1 Eccentric Beam Model.

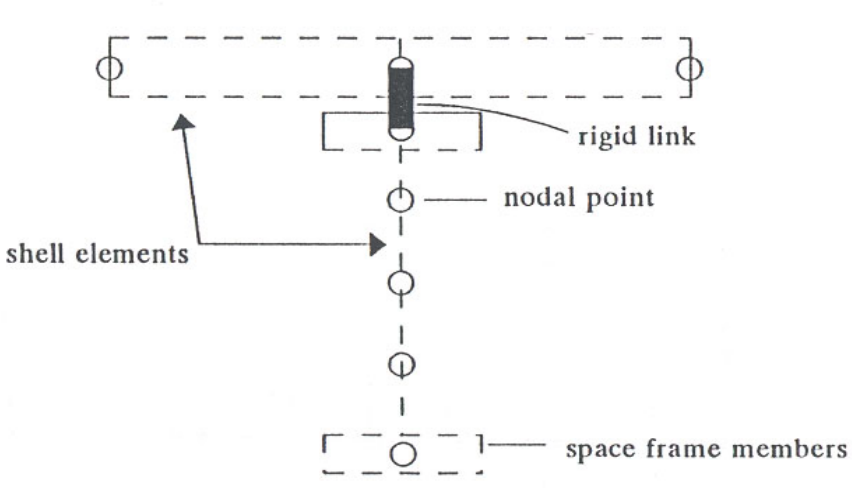


Figure 3.2 Detailed Beam Model (Brockenbrough 1986).

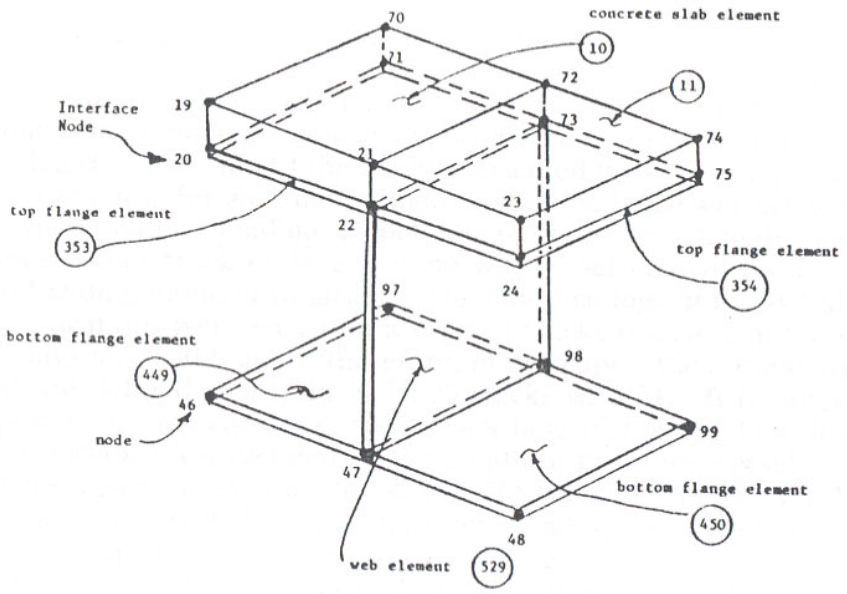


Figure 3.3 Solid Deck Model (Tarhini and Frederick 1992).

3.3 Grillage Analysis

The grillage analysis method involves the modeling of a bridge as a skeletal structure made up of a mesh of beams lying in one plane, as shown in Figure 3.4. Beam elements are used to model the behavior of the girders and the deck in the longitudinal direction, and other beam elements are used to simulate the behavior of the deck in the transverse direction. The properties of the longitudinal grillage members are determined from the properties of the girders and the portion of slab above them, about the neutral axis of the section. Similarly, the properties of the transverse grillage members are necessary to represent the transverse stiffness of the slab. In this way, grillage members represent the total stiffness of any portion of slab and girder. This plane grillage technique has been widely used and has been found to be robust for many structural shapes, loading conditions, and support arrangements (Keogh and O'Brien 1996). The major advantage of plane grillage analysis is that shear and moment values for girders are directly obtained. Thus, the integration of stresses is not needed.

However, there are significant disadvantages of using the two-dimensional grillage analysis model. For example, the grillage analysis method cannot account for shear lag. Thin slabs on a girder can be considered as wide flange beams. When this structure is subjected to flexural loading, normal stresses are generated in the section. The compressive stresses in the girder flange and the slab are not uniform in the transverse direction. As a result, longitudinal shear is generated. In other words, some portions of the slab between girders do not receive the same amount of axial stress as those near the center of the bridge. This phenomenon, known as shear lag, is dependent both on the geometric shape of the bridge deck and on the nature of the applied loading. Because of this, the neutral axis location in the bridge deck varies and moves towards the centroid of the wide flange section.

Another drawback of grillage analysis is that the moments in two longitudinal and two transverse grillage members meeting end-to-end are not necessarily the same. The discontinuity between moments is balanced as a discontinuity of torques in the beams in the opposite direction to keep the moment in equilibrium at the node. The moment

discontinuity may be exaggerated in the case of the edge of the grillage. The torque in the transverse beam, which has no other transverse beam to balance it, introduces discontinuity in the longitudinal beams.

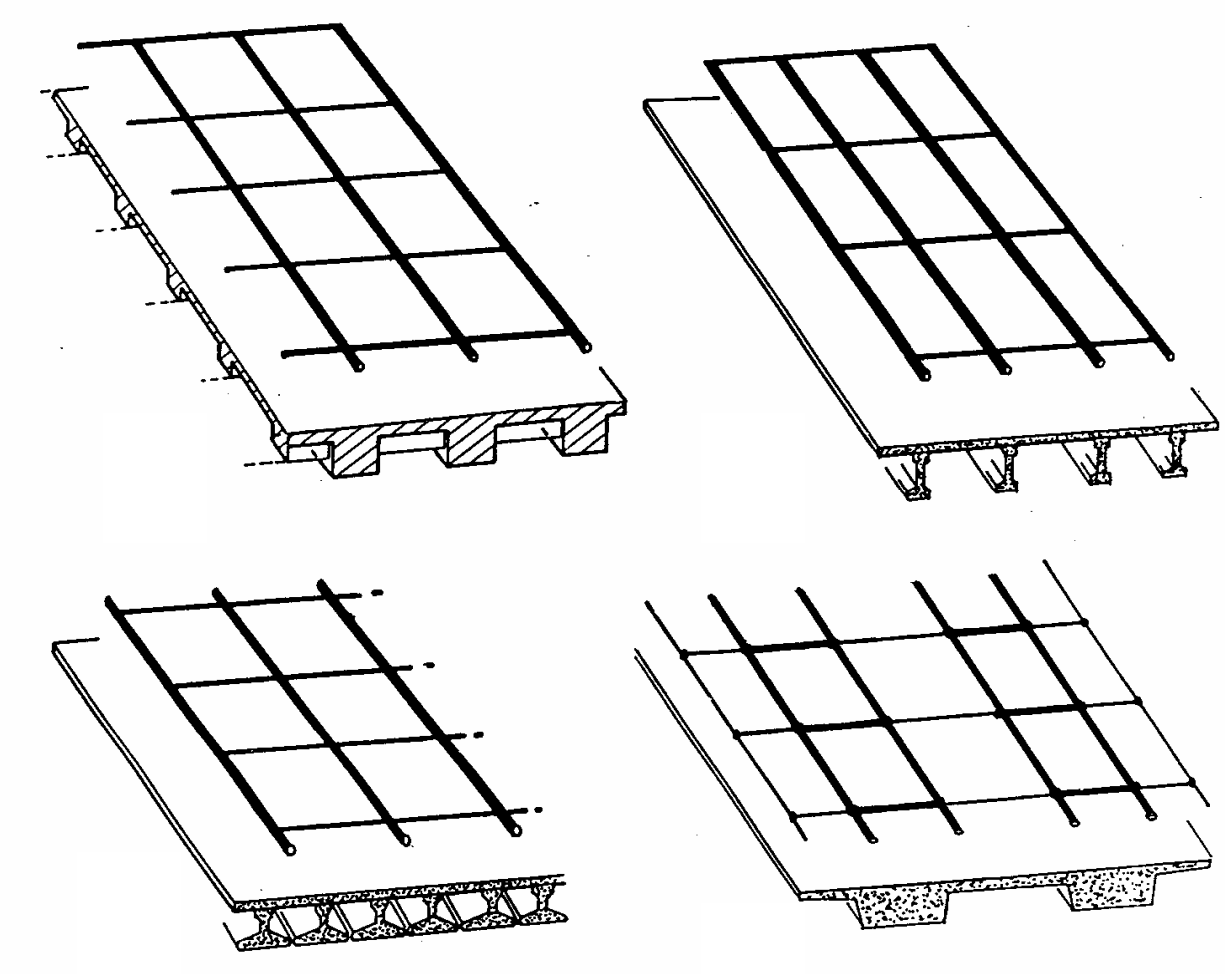


Figure 3.4 Grillage Analysis (Hambly 1991)

3.4 Selected Three-Dimensional FE Model

Finite element (FE) analysis enables bridge engineers to determine the distribution of wheel loads more accurately than using empirical or restricted code formulas. After reviewing several FE models, it has been concluded that the eccentric beam model provides a realistic idealization of bridge behavior while retaining simplicity, which is essential for the detailed analyses of these system (Chan and Chan 1999). The verification of the selected eccentric beam model is presented in Section 3.7. In particular, the results from the finite element analysis are compared to the results from the bridge field tests.

Compatibility Between Shell and Beam elements

The eccentric beam model ensures full composite action between the deck slab and the girders. This model utilizes the non-composite section properties of two elements to model composite action by applying the rigid links between the centroid of the girder and the midsurface of the slab. The concrete slab deck is usually modeled as shell elements, which combine plate bending and membrane elements. The effects of shear lag are automatically included since the elements used to model the slab consider membrane behavior as well as flexural behavior. Longitudinal girders are modeled using eccentrically connected beam elements.

Consider the eccentric beam model as shown in Figure 3.1 and Figure 3.5. As can be seen, the nodes of the beam do not coincide with the nodes of the plate. The beam and the plate should be connected in such a way that only plate degrees-of-freedom (DOFs) appear in the global structure. Imaginary weightless rigid links are added between the two pairs of nodes. Transformations are required to make beam DOFs ‘slave’ and plate DOFs ‘master’. The transformation equation between a plate node and its corresponding beam node is given by

$$\begin{Bmatrix} u^b \\ w^b \\ \theta_y^b \end{Bmatrix} = \begin{bmatrix} 1 & 0 & e \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u^p \\ w^p \\ \theta_y^p \end{Bmatrix}, \tag{3.1}$$

where the superscript *b* and *p* represent beam and plate, respectively. The *e* is the eccentricity between the plate element and the beam element. It should be noted that beam axial deformations are activated by the plate rotations.

Typical bending elements make use of a linear shape for the axial displacement and a cubic shape for the transverse displacement. It has been reported that displacement incompatibility occurs at the interface of two typical bending elements (Gupta and Ma, 1977). The axial displacement in the beam of eq. (3.1) is given as

$$u^b = u^p - e \cdot \theta_y^p . \tag{3.2}$$

This causes a quadratic expression of rotation in the plate (θ_y^p). The incompatibility is noticeable since the axial displacements (u^b and u^p) are linear but the rotation (θ_y^p) is quadratic in the axial (*x*) direction.

Even though this incompatibility error completely disappears as the mesh is refined, many studies have been proposed to eliminate this nonconforming error in the modeling of the eccentric beam model, which allow for the use of less refined meshes. Miller (1980) solved this problem by using the same elements, but including an extra axial DOF at the middle of each element so that the axial displacements becomes quadratic. Each term of the transformation equation given in eq. (3.2) is quadratic in the axial direction.

Various researchers (Marx, 1985) (Khaleel and Itani, 1990) (Chan and Chan, 1999) have proposed higher order elements based on the Mindlin theory. This theory automatically includes transverse shear deformation in element formulation and assumes that the normal to the midsurface remains straight after deformation, but not necessarily normal to the deformed midsurface. The slab is modeled as eight-noded serendipity elements or nine-noded Lagrangian elements, as shown in Figure 3.5. Three-noded Timoshenko beam elements are used to model girders. The shape of the element is quadratic for rotations and displacements separately. The elements used in modeling slab-

on-girder bridge are completely compatible with the quadratic expression at the interface of the slab and beam elements.

Selected ABAQUS Elements

The slected eccentric beam model utilizes the non-composite section properties of two elements to model composite action by applying rigid links (ABAQUS MPC) between the centroid of the girder and the mid-surface of the slab. The bridge deck slab is modeled by shear flexible eight-node shell elements (ABAQUS S8R elements), and the steel girder is idealized by three-node Timoshenko beam elements (ABAQUS B33 elements). This element selection has been made in order to eliminate a potential incompatibility along the element boundaries as discussed earlier.

Bearings are mechanical systems that transfer the reaction of superstructure components to the substructure. Since the main purpose of this study is to analyze the bridge superstructure, it is assumed that substructures, such as piers and abutments, do not influence the behavior of the superstructure. Although bearings are typically located below the beam element, many previous models neglected this fact and assumed bearings to be located at the centroid of the beam element or at the bottom flange of the beam. In this study, bearings are modeled by assigning boundary conditions to the zero-dimensional elements at their real location. For the simply supported beam, rotations in all directions are allowed in order to simulate the simply supported structure. Minimum restraints are assigned for longitudinal and transverse movement while vertical restraint is placed at the supports. Kinematic constraints are also supplied to nodes between the girders and the deck.

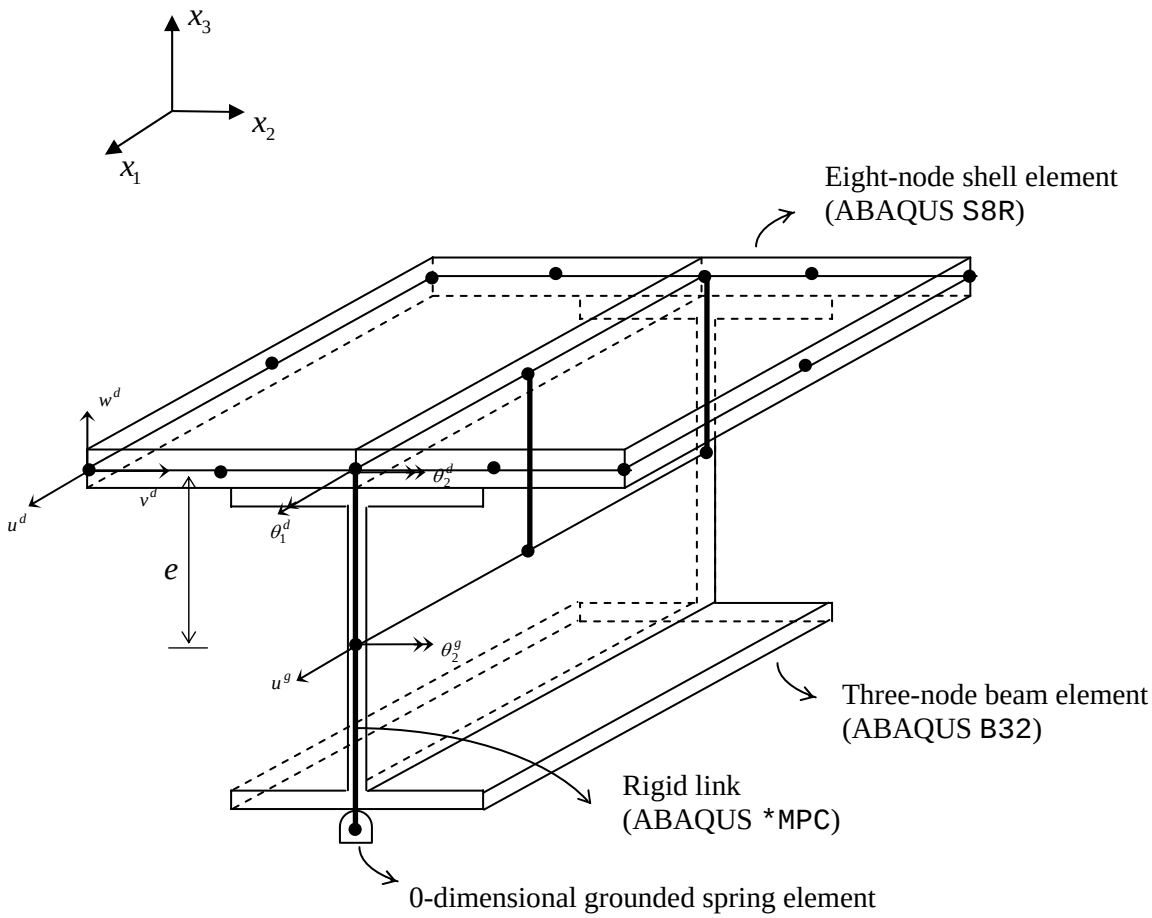


Figure 3.5 Improved Eccentric Beam Model

3.5 Pre-Processing

In this work, two pre-processors are developed for the finite element (FE) analysis of slab-on-girder bridges in order to expedite the analyses. They are the maximum longitudinal position generator (`Loadposition.m`) and the ABAQUS input file generator (`PrePro.for`). First, the truck position that produces the maximum moment or shear should be determined in the longitudinal direction. The transverse truck position is then positioned manually. In this procedure, it is required that a number of load cases be tested to determine the maximum effect to the specific girder. Whenever new load cases need to be placed, a new FE mesh must be created accordingly. The ABAQUS input file generator has been developed to minimize the preparation of the tedious FE data required by the program. The generator also uncouples the FE mesh generation from the load position by introducing the work equivalent nodal force algorithm. The flowchart of the procedure for determining the load distribution factor is shown in Figure 3.6.

ABAQUS Input Generator

An input generator was developed using the FORTRAN programming to automatically prepare the lengthy data required by the ABAQUS program. The key function of the input generator is to calculate the work equivalent nodal forces (ENF) and to place these loads at the proper nodal points. A complete list of the ABAQUS input file generator (prePro.for) is given in Appendix A.

The live load for bridge design is the AASHTO HS20 standard truck loading according to article 3.7.4 in AASHTO 1996 bridge specification and shown in Figure 3.7. The applied loading on a bridge deck consists of pressure loads applied through a tire patch. The AASHTO bridge code specifies a “tire contact area” in order to ensure a more exact analysis. The contact area is based on the wheel of a standard HS design vehicle. The ratio of the length in the direction of traffic to the tire width is given as 1:2.5, and the wheel load is assumed to be a uniform pressure, as shown in Figure 3.8.

In finite element modeling, this requirement imposes the need for a fine mesh in the deck so that the element is fitted to the patch size. In order to uncouple the patch load from the mesh size, equivalent nodal loads are employed. Both the Chen (1999) and Eom and Nowak (2001) studies utilized the Mindlin shape functions for four-node shell elements in their equivalent nodal force calculation. Each wheel load is considered as a single concentrated load on the shell elements instead of the patch load. Kim (2000) developed an algorithm that identifies the tire patch position and calculates equivalent nodal forces for the three-dimensional solid element in the application of FE analyses of pavement. The current FE model developed for this study requires an eight-node shell element for modeling of the bridge deck. The typical plate DOF of an eight-node shell element and the nodal numbering scheme are shown in the natural coordinate system, as shown in Figure 3.9.

The equivalent nodal load of the patch load can be calculated by the surface integral as follow:

$$\mathbf{R}_{\mathbf{e}} = \int_{S} \mathbf{N}^{T} \mathbf{t} dS \tag{3.3}$$

where **t** is the surface traction vector and **N** is the shape function matrix. Using this method, one must identify the nodes and elements that lie on the patch load. This approach is further complicated if the bridge deck is skewed. In this study, to expedite this computation, the patch load is discretized as a number of uniformly distributed sub-point loads, as shown in Figure 3.10. This method considers each sub-point load as a single concentrated load. If there are *K* sub-point loads applied to the tire patch on an element of an amount *p*, then the equivalent nodal forces are computed as:

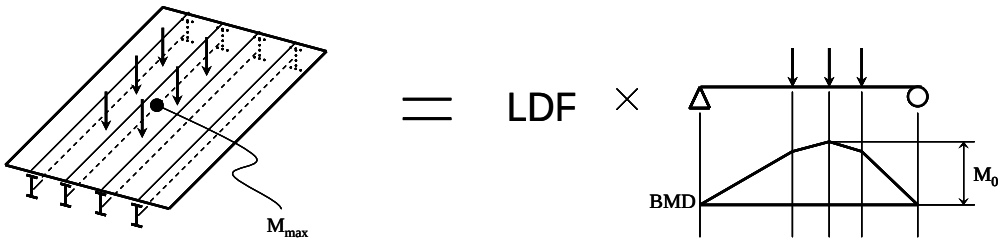
$$\mathbf{R}_{\mathbf{e}} = \sum_{i=1}^{K} \mathbf{N}_{i}^{T} \mathrm{p}_{i} \ . \tag{3.4}$$

The Mindlin plate shape functions are used for the calculation of the equivalent nodal forces, which are given by:

$$N_i = \frac{1}{4}(1+\xi\xi_i)(1+\eta\eta_i)(\xi\xi_i+\eta\eta_i-1) \text{ for node 1, 2, 3, and 4}$$
$$N_i = \frac{1}{2}(1+\eta\eta_i)(1-\xi^2) \text{ for node 5 and 7} \quad (3.5)$$
$$N_i = \frac{1}{2}(1+\xi\xi_i)(1-\eta^2) \text{ for node 6 and 8}$$

where subscript *i* represents the node number. The definition of the natural coordinate system is shown in Figure 3.9.

The element number and the corresponding node number that are subject to each sub-point load are identified. The saved equivalent nodal forces are assembled at the appropriate entry of the load vector. The detailed algorithm is shown in Figure 3.11. A major advantage of the discretized patch load algorithm is that it eliminates the cumbersome load boundary search problems and numerical integration, while retaining the accuracy of the FE solution when a sufficiently refined tire patch is used. The discretization error of the patch load is estimated using a square plate loaded under a distributed load represented by sub-point loads. Different levels of discretization are considered by increasing the number of sub-point loads. The exact equivalent nodal forces are calculated by Eq. (3.3). It is clear from Figure 3.12 that the equivalent nodal forces for both corner nodes and interior nodes using the proposed discretized algorithm converge to the exact value of equivalent nodal forces as the discretization level increases. It is observed that the use of approximately 100 sub-point loads results in less than 1% error for both corner and interior nodes.



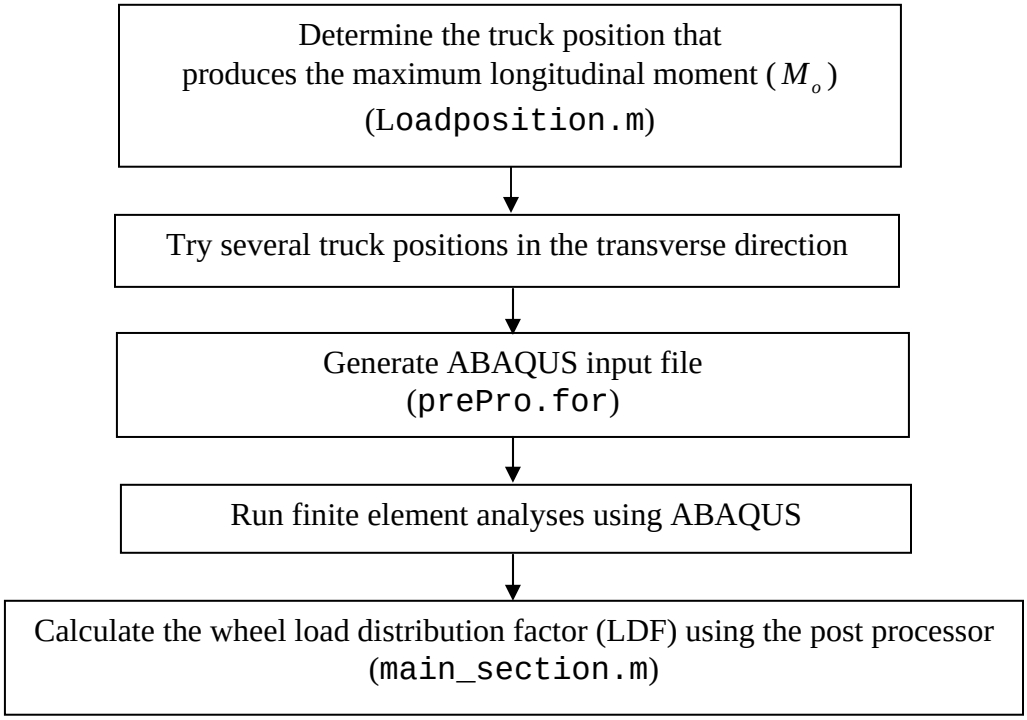


Figure 3.6 Flowchart of the Procedure Used for the Determination of the LDF.

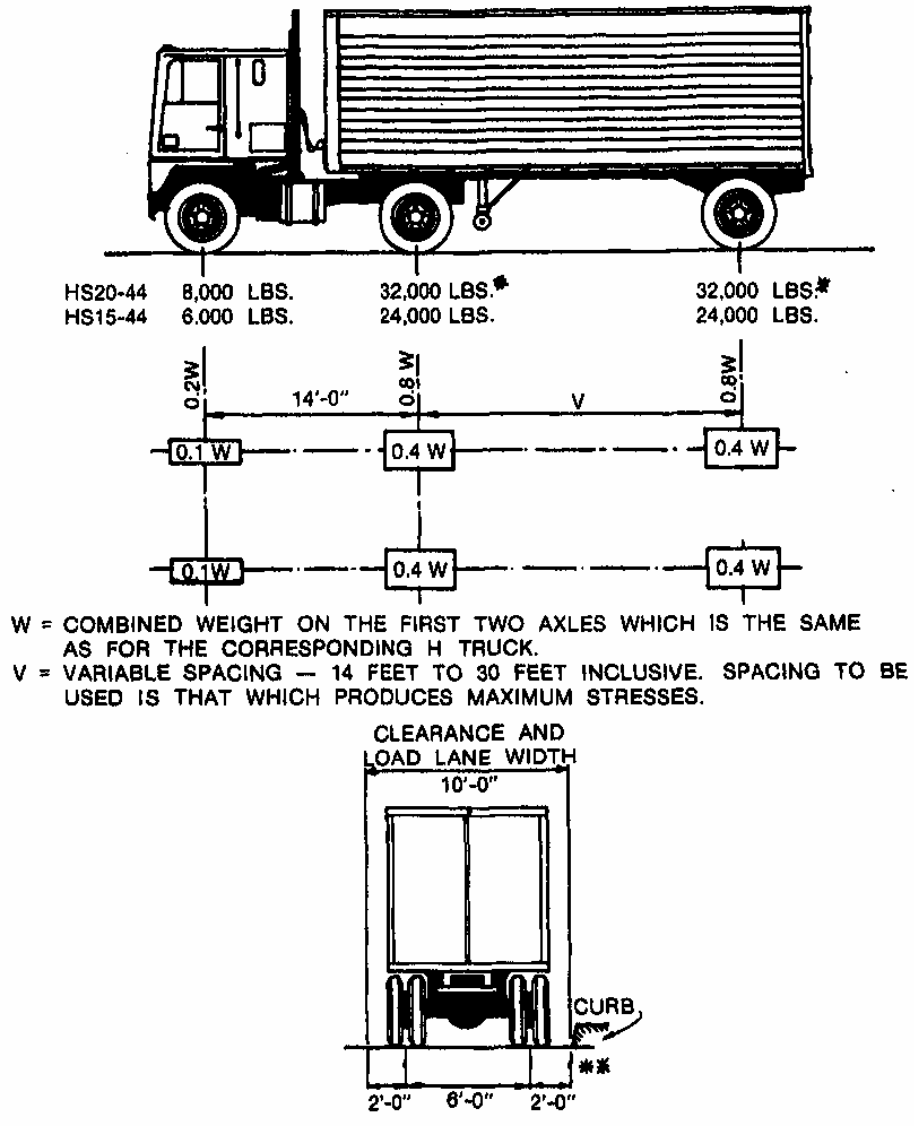


Figure 3.7 AASHTO HS-20 Design Truck (AASHTO 2002)

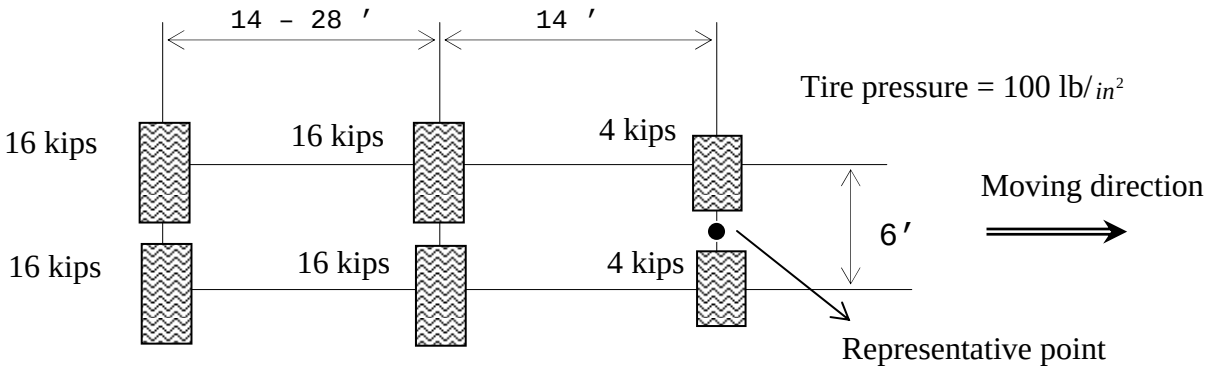
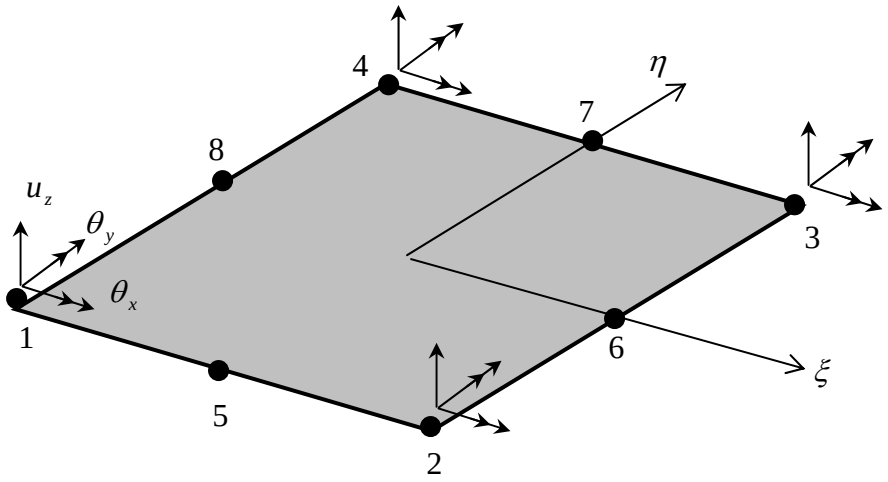


Figure 3.8 View of Tire Contact Area of AASHTO HS20 truck (Dimensions are 4 by 10 in. and 8 by 20 in. for the Front and Rear Tires, Respectively).



node	ξ	η
1	-1	-1
2	1	-1
3	1	1
4	-1	1

Figure 3.9 Typical Bending DOFs and Node Numbering of Shell Elements.

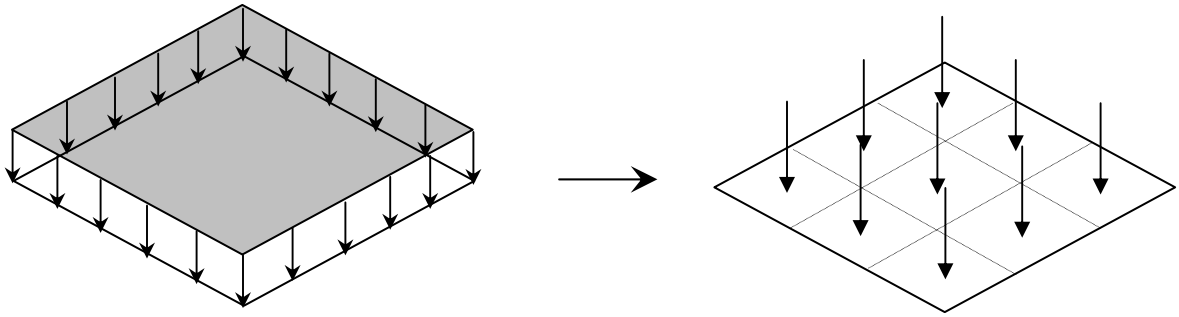


Figure 3.10 Discretization of Patch Load.

```
Do over the number of trucks
    Do over the number of wheels
        Do over the number of sub-point loads
            Identify loaded elements and nodes
            Calculate the position of sub-point load in the natural coordinate system
            Calculate the equivalent nodal loads
        End Do
    End Do
End Do
Assemble the equivalent nodal forces to appropriate entries in the load vector
Print the results to the ABAQUS loading block (*CLOAD)
```

Figure 3.11 Equivalent Nodal Force Computation Algorithm.

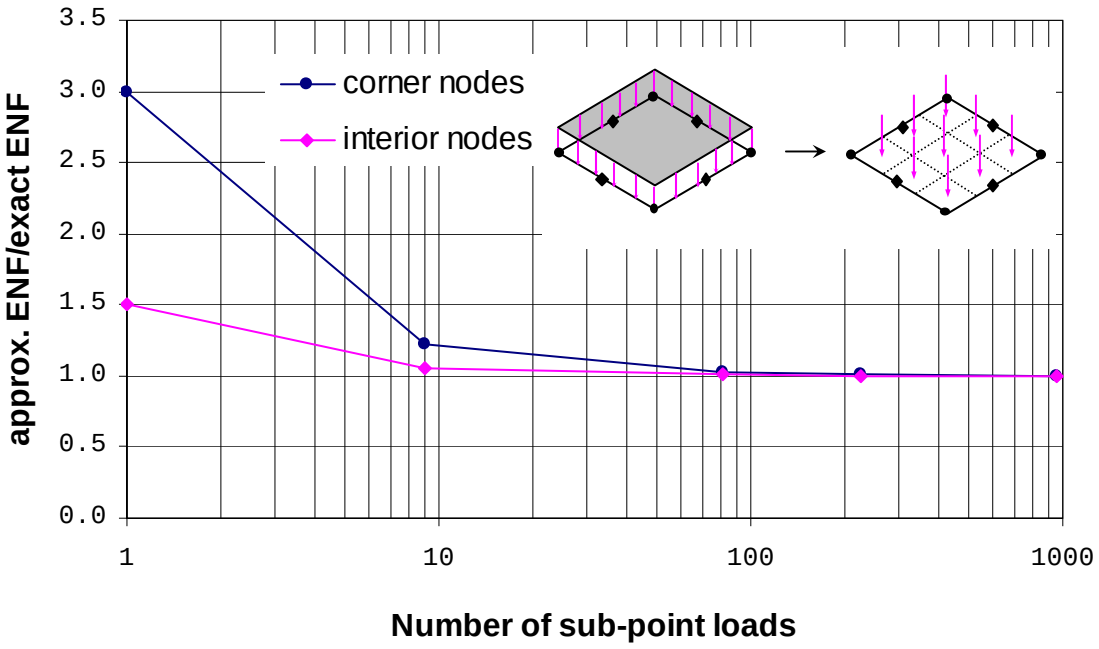


Figure 3.12 Discretization Error of Patch Load.

3.6 Post-Processing

In order to obtain the load distribution factor, the post-processing of the finite element results is required. The bridge finite element analysis yields the results only at the nodes and the element integration points. These locations depend on the meshing of the bridge. In order to obtain the results at other locations, the interpolation of the finite element results is necessary. Furthermore, the finite element analysis yields only forces and moments in the elements. In order to obtain meaningful results for bridge analysis, i.e. the load distribution factor, an appropriate interpretation of the finite element results is essential.

In the parametric study of the load distribution factor, several bridges are analyzed. Since the post-processing is performed repeatedly for each bridge analysis, a tool to alleviate a number of repetitive post-processing procedures is developed. The tool should be generalized and automated enough to perform the post-processing of several bridges with different bridge configurations. The MATLAB Software is used as the tool in this post-processing procedure since it provides rich graphical capability.

In this section the determination of the load distribution factor is demonstrated. Other issues related to the calculation of the load distribution factor, such as the ABAQUS result format, the effective width determination, the interpolation of results, the moment in the girder section, and the moment from beam analysis are presented. Finally, the program description, which includes the utilized algorithm, its limitation, and its manual, are discussed.

3.6.1 ABAQUS Result Format

The beam results from ABAQUS are illustrate in Figure 3.13, where the local n_1 direction is specified as <0,1,0>. In this figure, SF1 is the axial force, SM1 and SM2 are the bending moments about the local n_1-axis and local n_2-axis, respectively, and SM3 is the twisting moment about the beam axis. The local tangent along the beam element, t-axis, is defined as a vector from node i to node j. The local n_1-axis has to be specified in

the modeling procedure. The local n_2-direction is perpendicular to the local tangent t-axis and the local n_1-axis. These three axes follow the right-hand-rule. The results from the beam element are formatted by element number, point number, as SF1, SM1, SM2, and SM3, respectively. There are two integration points in the selected beam element. The required results for the post-processing are SF1 and SM1.

The ABAQUS shell results are shown in Figure 3.14, where SF1, SF2, and SF3 are the forces per unit width in the local 1-axis, local 2-axis, and local 1-2 plane, respectively, and SM1, SM2, and SM3 are the bending moments per unit width about local 2-axis, local 1-axis, and local 1-2 plane, respectively. The local-1 axis and local-2 axis lie on the plane of the shell element. The default direction of the local 1-axis is the projection of the global 1-axis onto the shell surface. If the global 1-axis is normal to the shell surface, the local 1-direction is defined as the projection of the global 3-axis onto the shell surface. The local 2-direction is perpendicular to the local 1-diretion on the plane of the shell element. The local 1-axis, the local 2-axis, and the local n-axis follow the right-hand-rule.

The results of the shell element are formatted by element number, Gauss point, SF1, SF2, SF3, SM1, SM2, and SM3, respectively. There are four Gauss points for the selected shell element. The required results for post-processing are SF1 and SM1.

3.6.2 Effective Width

The effective width of the deck can be determined using the AASHTO specification. For the interior girders as the smallest value of the following:

- One-fourth the span length
- Center-to-center distance between stringers
- Twelve times the average thickness of the slab, plus the greater of the girder web thickness or one-half the top flange width of the girder.

For the exterior girders, the effective width is specified as one-half the effective width of the adjacent interior girder, plus the smallest value of the following:

- One-eighth the span length

- Overhang width
- Six times the average thickness of the slab, plus the greater of half the girder web thickness or one-quarter of the top flange width of the girder.

3.6.3 Interpolation of Results

Finite element programs generally provide the results at the element nodes. However, the most accurate results from the finite element analysis are at the integration points. Since the eight-node thick shell elements with reduced integration are used to model the bridge deck, four integration points at Gauss Quadrature points of each shell element are available.

ABAQUS provides the shell element resultants, i.e., resulting force and moment, as the value per unit width. The total resultants along a section of shell element are achieved by multiplying the ABAQUS resultants to the section width. Since the location of the required resultants may not coincide with the integration points, the interpolation of the resultants at the integration points to the resultants at the specific location is necessary.

The method to interpolate the finite element resultants for shell elements is illustrated in Figure 3.15. In this figure, only the 2 by 3 mesh of the shell elements for the concrete deck is used as an example. The interpolated resultant at points 1, 2, 3, and 4 is required to establish the resultants along section B-B. The resultant at point 1 can be determined by interpolating the resultants at all integration points along line A-A. In the developed post-processor, spline interpolation is adopted. The advantage of this interpolation over the polynomial interpolation is that the interpolated value is not as sensitive to the remote values as it is in polynomial interpolation. The resultant at points 2, 3, and 4 can also be determined in the same manner. The resultant in section B-B is calculated by the summation of all products of each element resultant and half of its element width along the section B-B. It should be noted that the interpolation of the

resultant in the beam element is achieved in the same manner as the determination of the interpolated resultant at point 1.

3.6.4 Moment in the Girder Section

In order to calculate the LDF from the finite element analysis, the moment in the girder section has to be established. The moment in the girder section is the resultant of three components from the finite element results. These three components are girder moment in the beam elements, deck moment in the shell elements, and moment from the axial forces as shown in Figure 3.16.

a) Girder Moment

The girder moment is the moment in the beam element. The girder moment at a specific location is obtained by the interpolation of the girder moments at the integration points along that girder.

$$M_g = -SM_{1b}$$

where M_g is the girder moment and SM_{1b} is the interpolated resulting moment in the beam element at the required location. It should be noted that the minus sign is due to the sign convention used in ABAQUS result format.

b) Deck Moment

The deck moment is the moment in shell elements. The effective width described in the previous section is used to determine the boundary of the girder section. The deck moment is determined by the summation of all products of each shell moment and half of its element width within the girder section, i.e.,

$$M_s = -\sum_i \left(SM_{1s,i} \cdot \frac{b_i}{2} \right)$$

where M_s is the deck moment, $SM_{1s,i}$ is the interpolated moment in the shell element i at the required location, and b_i is the width of shell element i within the girder section

c) Moment Produced by Axial Forces

In the determination of the moment produced by the axial force, the neutral axis of the composite section needs to be determined. There are several methods to determine the moment from axial forces, as there are

i. *Neutral Axis of the Girder*

Since the force and moment in the girder are known, the neutral axis of the girder can be established. The idea of this method is that the neutral axis is where the strain and the corresponding stress in the beam element are zero. The eccentricity from the neutral axis to the centroid of the girder is calculated as follows:

$$\sigma = \frac{P}{A_g} + \frac{M \cdot y}{I_g} = \frac{SF_{1b}}{A_g} + \frac{(-SM_{1b}) \cdot (-e_g)}{I_g} = 0$$

$$e_g = -\frac{SF_{1b} \cdot I_g}{A_g \cdot SM_{1b}}; \qquad e_s = e - e_g$$

Then the moment due to the axial force can be calculated as follows:

$$M_{axial} = SF_{1b} \cdot e_g + \sum_i \left(- SF_{1s,i} \cdot \frac{b_i}{2} \right) \cdot e_s$$

Where, A_g and I_g are the girder area and moment of inertia, respectively, SF_{1b} is the girder axial force, $SF_{1s,i}$ is the axial force in the shell element i; e is the distance between the centroids of the girder and the deck; e_g and e_s are the eccentricity from the section's neutral axis to the centroids of the girder and the deck, respectively; and M_{axial} is the moment from the axial force.

ii. *Transformed Section*

In this method, the section's neutral axis is determined by transforming the area of concrete deck into an equivalent steel area using the modular ratio ($n=E_s/E_c$). The transformed section is used to determine the location of the neutral axis. Then the moment from the axial force can, thus, be obtained as: -

$$M_{axial} = SF_{1b} \cdot e_g + \sum_i \left(- SF_{1s,i} \cdot b_i \right) \cdot e_s$$

iii. *Average Force of Girder and Slab*

The idea of this method is that, in the girder section, the values of the girder force and the deck force are close to each other. The moment from the axial forces is determined by multiplication of the average of forces between the girder and the deck to the distance between the two forces as follows:

$$M_{axial} = \left(\frac{SF_{1b} + \sum_i \left(- SF_{1s,i} \cdot b_i \right)}{2} \right) \cdot e$$

The advantage of this method is that the neutral axis is not necessary to be determined.

iv. *Girder Force*

In this method, only the girder force is used in the determination of section moment. The idea is that the girder force is usually slightly greater than the deck force in the section. The moment from the axial force is:

$$M_{axial} = SF_{1b} \cdot e$$

The advantage of this method is that the neutral axis needs not to be determined and the moment from the axial force is slightly greater than other method and considered in the conservative side.

Since the force in the girder and the force in deck are typically close to each other, the moments from axial force from different methods are not significantly different. The transformed section method will be used herein.

3.6.5 Moment from Beam Analysis

In the determination of the load distribution factor, the maximum moment from the beam analysis is required. The one-dimensional beam analysis is analogous to the bridge configuration as illustrated in Figure 3.17. The length of beam is the same as the bridge span length. However, the loading on the beam analysis is one line of wheel loads placed at the position that produces the maximum moment. The maximum moment from the beam analysis will be used in the determination of the load distribution factor in the following section.

3.6.6 LDF Calculation

In the calculation of the load distribution factor, the moment in the girder section from the finite element results and the maximum moment from the beam analysis have to be known. The moment in the girder section from the finite element analysis composes of three parts that are the girder moment, the deck moment, and the moment from the axial force. The maximum moment from the beam analysis is the maximum moment due to a single line of wheel load at the position to produce the maximum effect in the beam analysis. These moments are described in detail in the previous section.

Consistent with the AASHTO specification, the LDF is determined by back-calculation from the maximum moment and the moment from one-dimensional beam analysis. The LDF can be calculated using the moment in the girder section divided by the maximum moment from the one-dimensional beam analysis.

The post-processing is used to obtain the section moment and the moment envelop for a specific load configuration. The MATLAB M-files for the post-processing are generalized enough to include the effect of skew angle and can also be easily modified to the need in the future research since each file is the stand-alone file. This post-processing eases tremendously the work in analyzing a number of bridges to obtain the load distribution factor.

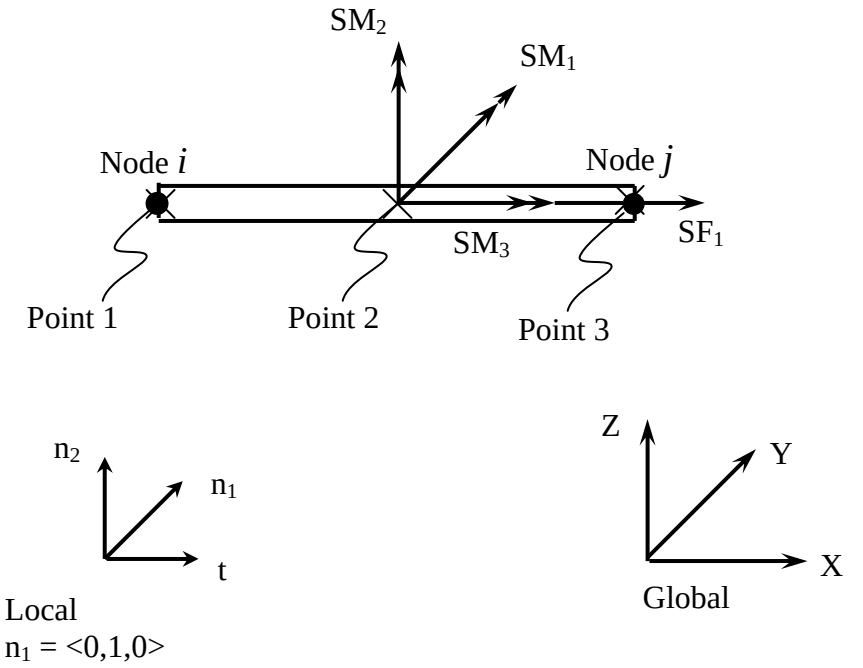


Figure 3.13 ABAQUS Notation of Beam Element.

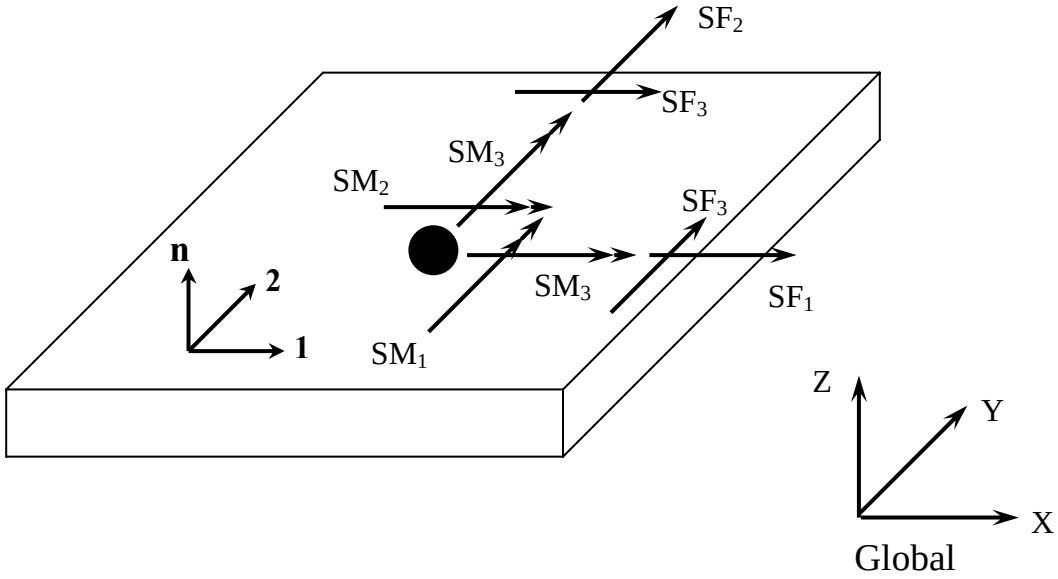


Figure 3.14 ABAQUS Notation of Shell Element.

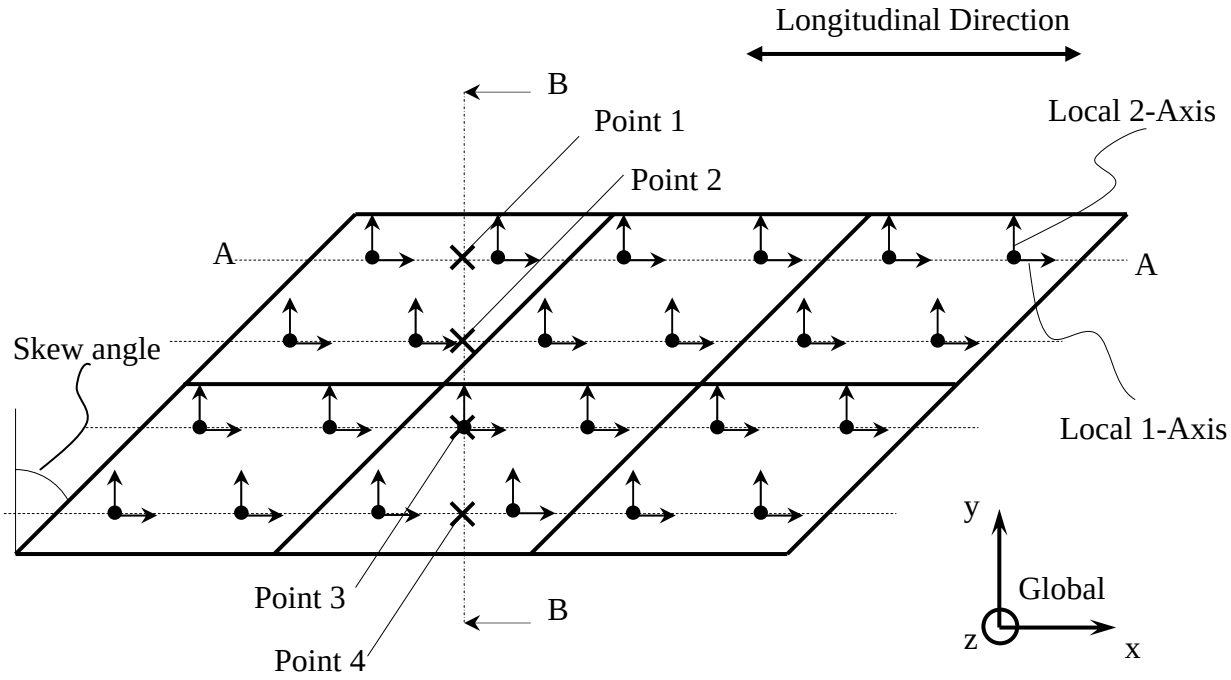


Figure 3.15 Interpolation of Finite Element Results for Shell Elements.

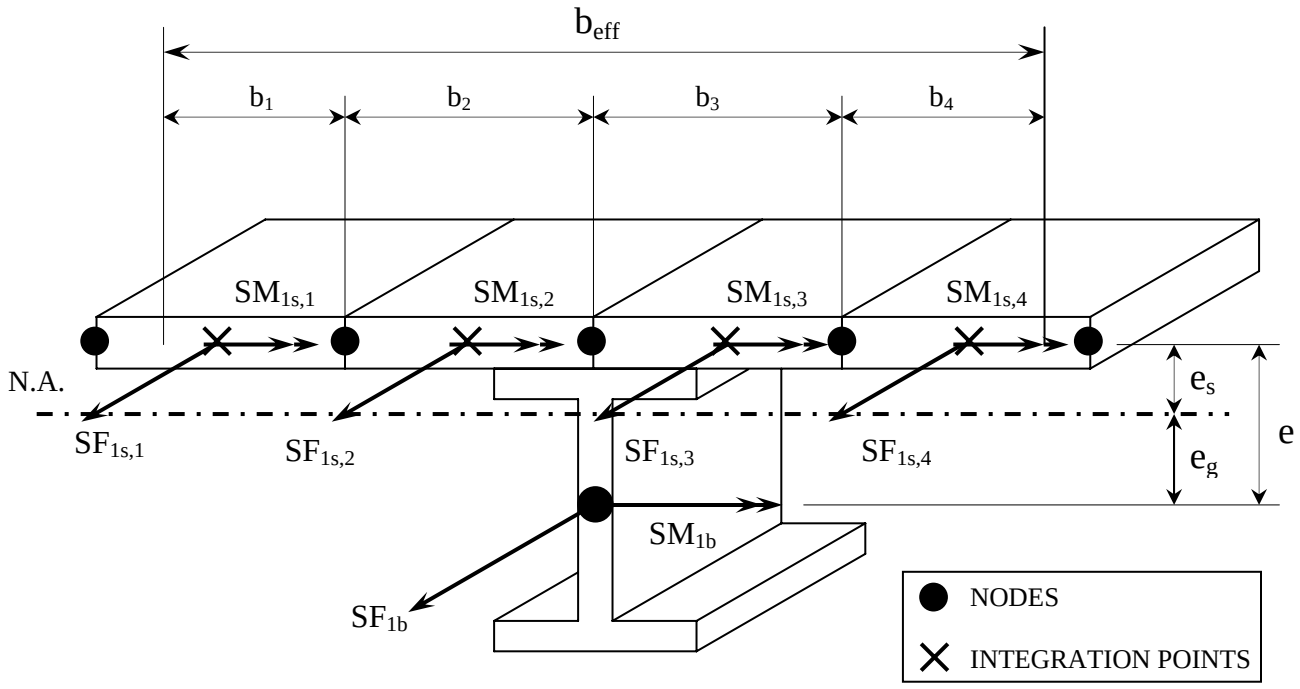


Figure 3.16 Moments in the Girder Section for the Determination of the LDF.

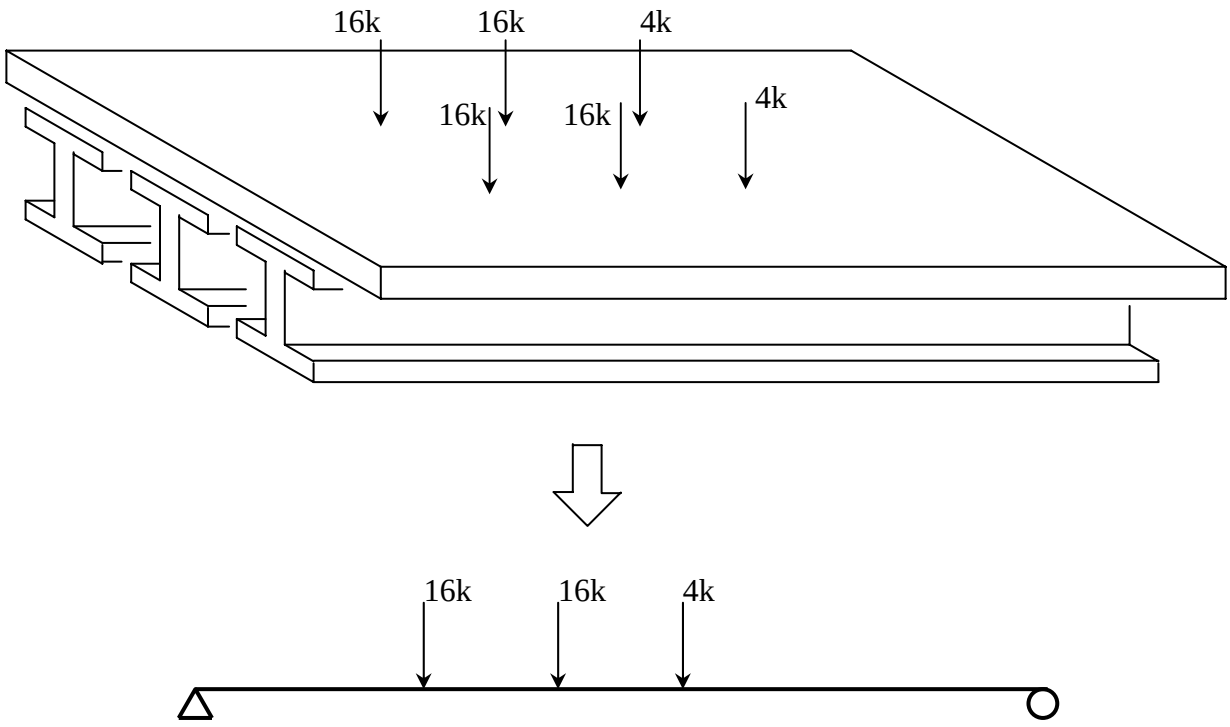


Figure 3.17 The Beam Analysis Analogous to the Bridge Geometry.

3.7 Comparison with Experimental Results

To verify that the selected finite element software, ABAQUS, is able to simulate the real behavior of the bridge, the results from the finite element analysis are compared to the results from the bridge field test.

The data used to compare the finite element model and the field test is the distribution of moment to each girder in the section of maximum moment. The selected finite element program is validated if it can simulate the bridge behavior, or in other words, if it can produce the distribution of moment to each girder close to the field-test results. When the finite element program is validated, the finite element model and finite element program will be confidently used as the level 3 analysis. Then, the finite element program will be confidently used to analyze Indiana bridges to obtain the girder distribution factors.

The results from two field tests are compared to the finite element results to verify the current finite element model. First bridge is tested by University of Tennessee in 1972. The other bridge is tested by University of Michigan.

3.7.1 Elk River Bridge

The Elk River Bridge in Tennessee is selected to verify the developed finite element model. This bridge is one of the bridge field tests used by Project 12-26 to verify their selected finite element programs. A full-scale bridge testing was reported by Burdette and Goodpasture (1971) from University of Tennessee. Figure 3.18 shows the Elk River Bridge before testing and the bridge failure after testing, respectively.

The bridge is on route 130 over Elk River in Tennessee. The bridge is s continuous with four-spans of 70, 90, 90 and 70 feet span length. Four longitudinal steel girders and seven inch deep reinforced concrete deck are used. This bridge was designed to carry two traffic lanes with 34.5 feet total width. The bridge has no-skew, horizontal tangent, and the field tests were carried out soon after the bridge construction. The girder spacing is 8.33 feet. The bridge cross section and the bridge longitudinal view are shown

in Figure 3.19 and Figure 3.20, respectively. The concrete modulus for bridge deck is 4415 ksi with Poisson's ratio of 0.2. The steel girder is W36x170 with modulus of 30000 ksi and Poisson ratio of 0.3.

The simulation of the truck wheel loads was accomplished by using a rock anchor system. The loads were applied at each of the eight rear wheel locations to simulate two trucks. The bending moments used in determining the lateral load distribution factor in each girder were determined by strain measurement. The strains were measured at the top and bottom of the flanges of the steel girder at the location of critical section, the maximum moment section. By knowing these strains, the neutral axis location could be established. As a result, the centroid of compression force in concrete slab could be located. The tensile force in steel could be obtained on the basis of the known steel stress-strain relation. Then, the moment was calculated as the product of the tensile force and the internal moment arm.

The bridge is modeled in the finite element software by using the technique described in the previous chapter. The concrete slab is idealized as quadrilateral shell elements and eccentrically connected by rigid links to the beam elements representing steel girders. The bearing or the support is modeled by grounded spring. The bearing is assumed to be 2 inch thick. The bridge deck is meshed into 256 longitudinal meshes by 32 transversal meshes. The shell element within the overhanging width has the dimension of 15 by 14.25 inches. The shell element between the girders has the dimension of 15 by 12.5 inches. The total number of nodes is 37,487 and the total number of elements is 9236. The truck loading is 16k positioned at 8 locations to represent two trucks. Figure 3.21 shows the moment envelope produced by the post-processing.

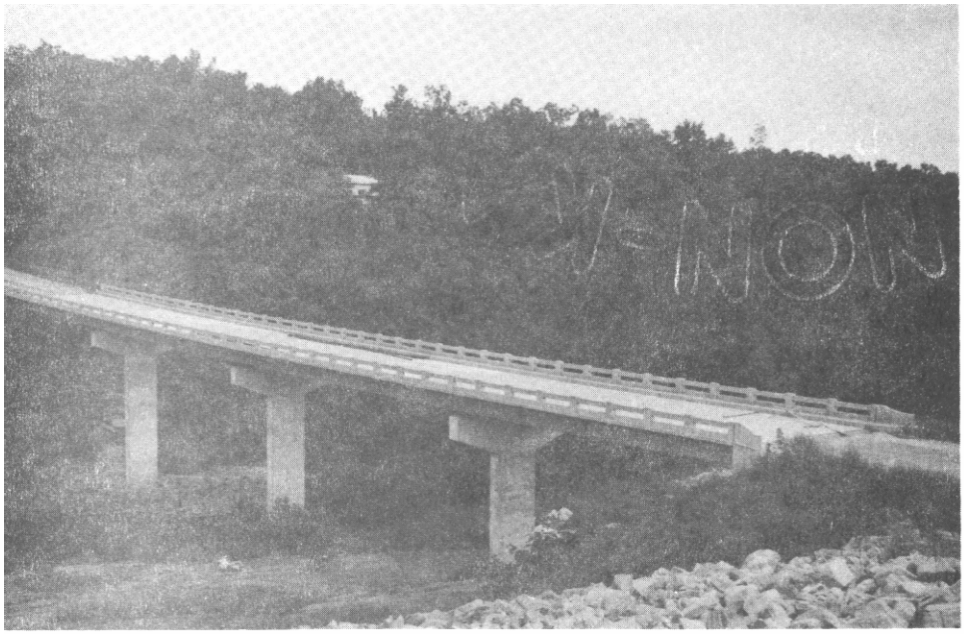
The data used to compare the finite element model and the field test is the distribution of moment to each girder in the section of maximum moment. The results of the distribution of moments to the middle girder from the field test, the finite element model, the AASHTO LRFD 1998 LDF equation, and the AASHTO Standard 1996 LDF equation are shown in Table 3.2. Compared to the field test result, the finite element software, ANSYS, is able to predict satisfactorily the load distribution. As predicted for

the moderately large girder spacing, the AASHTO standard 1996 formula overestimates the load distribution by 31 percent. AASHTO LRFD 1998 was able to predict the load distribution with good agreement and conservatively with respect to the experimental results.

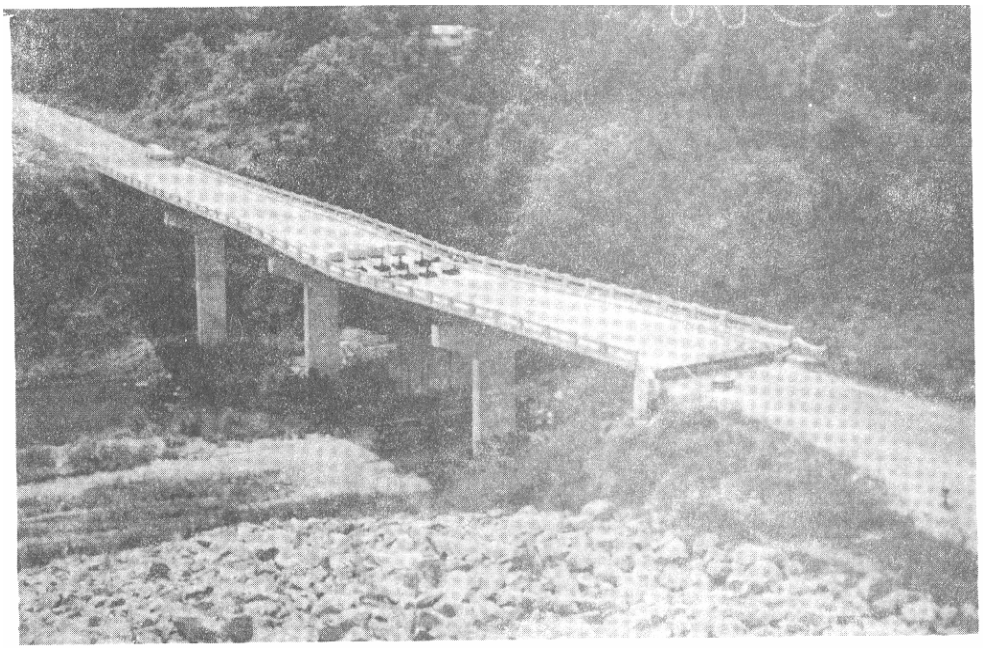
As a conclusion, the selected finite element model using ABAQUS program is able to predict well the distribution of moment. It should be noted that the application of the pre- and post-processing help dramatically reduce the time to perform level 3 analysis. The typical time to perform level 3 analysis for this bridge without pre- and post-processor is more than 100 hours. By using the customized pre- and post-processing software, the time used to model, analyze, and obtain the LDF is less than one hour once the required bridge information is obtained.

Table 3.2 Comparison of Distribution of Moments

	Field Test	Finite element model (ABAQUS)	AASHTO LRFD 1998	AASHTO Standard 1996
Distribution of moments to the middle girder	1.16-1.20	1.16	1.24	1.52



(a) Before Testing



(b) Bridge Failure after Testing

Figure 3.18 Elk River Bridge

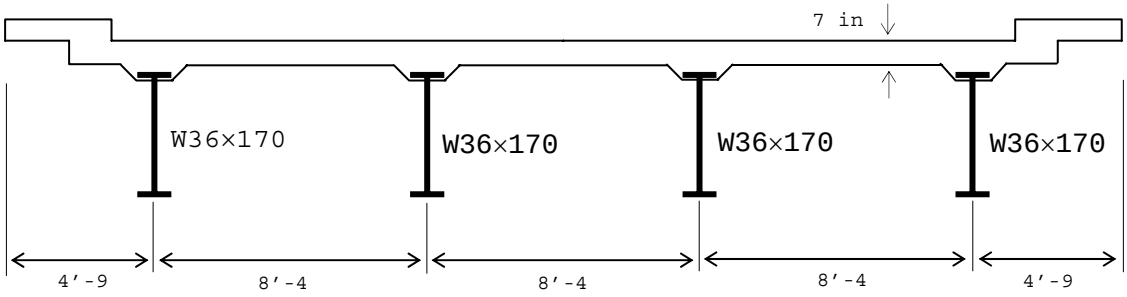


Figure 3.19 Cross Section of Elk River Bridge.

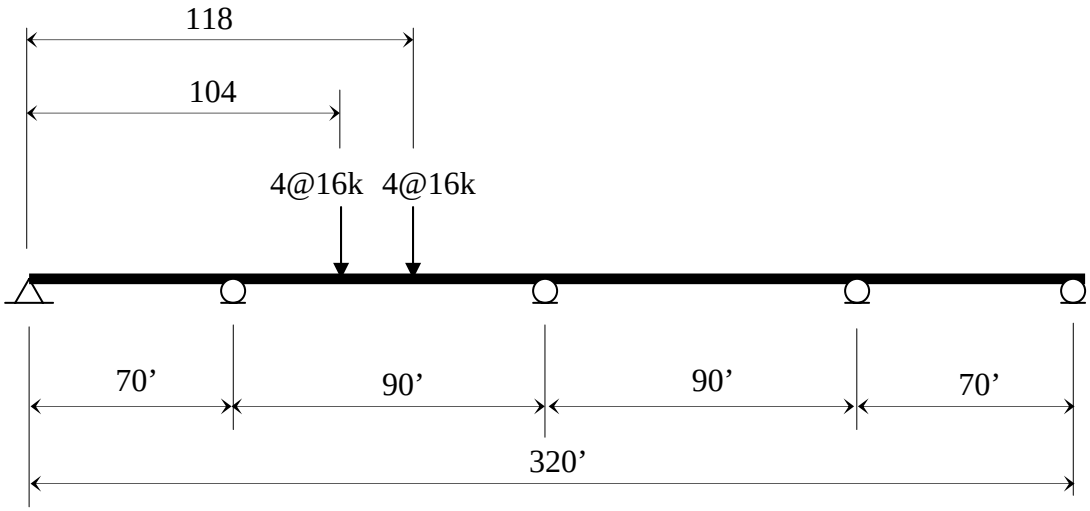


Figure 3.20 Longitudinal Dimensions and Loading Locations for Elk River Bridge.

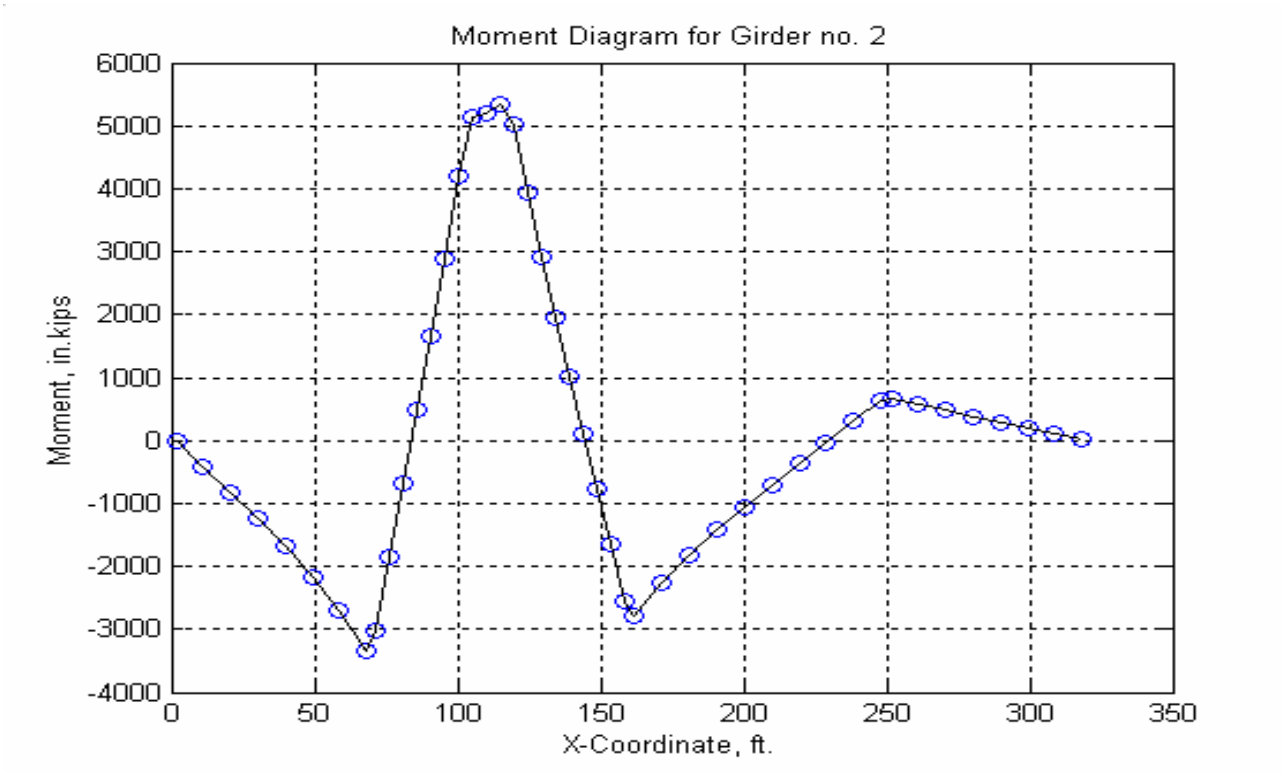


Figure 3.21 Moment Envelope of Girder Section no. 2.

3.7.2 Michigan Bridge

The finite element model developed for this study is further verified with the results of the field test conducted at the University of Michigan (Eom and Nowak 2001). The tested simple span bridge is located on Stanley Road over I-75 in Flint, Michigan. The span length is 126 ft (38.4 meters). There are 7 girders with girder spacing of 7.25 ft (2.21 meter) and an overhanging width of 2.45 ft (0.747 meter). The slab thickness is 8 in (203 mm). The cross section of the tested bridge is shown in Figure 3.22. General bridge information is given in Table 3.3 and girder section dimensions are given in Table 3.4.

Strain gauges were installed at the bottom flanges of the girders as shown in Figure 3.23. All strains were measured at mid-span. The load distribution factors were then calculated from the strains at the specific girders. The test load for field-testing is the Michigan three-unit, 11-axle truck. The weight and the axle configuration are given in Figure 3.24. The load test was performed with the truck at crawl speed to produce the maximum static strain at the steel girders.

For a simply supported bridge, the finite element model proposed for this study allows rotations along all directions and assigns minimum restraints for longitudinal and transverse movement while vertical restraints are placed at the supports. Since the details of reinforcement system is not available, the minimum amount of reinforcement is assumed according to the AASHTO specification.

The calculated load distribution factors for the selected FE model are compared to those obtained from the test results. As can be seen in Figure 3.25, good agreement between measured and calculated values is observed in all girders. The calculated values are conservative up to 8 %. This discrepancy may be due to the absence of the cross bracing in the FE model of the tested bridge. The results are consistent with the findings of previous studies (Eamon and Nowak 2002) (Mabsout et al. 1997) since the presence of secondary elements such as cross bracing and parapet carry more load by reducing the load effects in the interior girders. Therefore, the finite element model used for this study

is capable of accurately predicting the actual load distribution behavior of steel girder bridges.

Table 3.3 Michigan Test Bridge Information

Span length	126 feet
Number of girders	7
Girder Spacing	7.25 feet
Slab thickness	8 inch
Skew angle	0 degree
Total transverse width	48.4 feet
Overhang width	2.45 feet
Average daily traffic (ADT)	2000

Table 3.4 Girder Cross Section Dimensions for Michigan Test Bridge

Top flange	18” × 1⅞” (for 70’6” in the center) 18” × ⅞” (for 27’9” on each side)
Bottom flange	18” × 2¾” (for 84’ in the center) 18” × 1⅜” (for 24’ on each side)
Web	48” × ½”

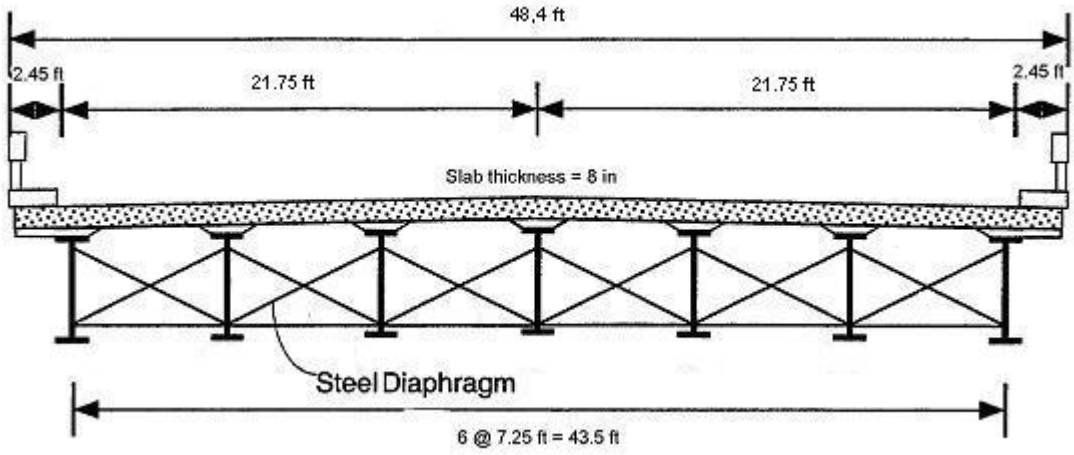


Figure 3.22 Cross Sectional View of Michigan Test Bridge (Eom and Nowak 2001).

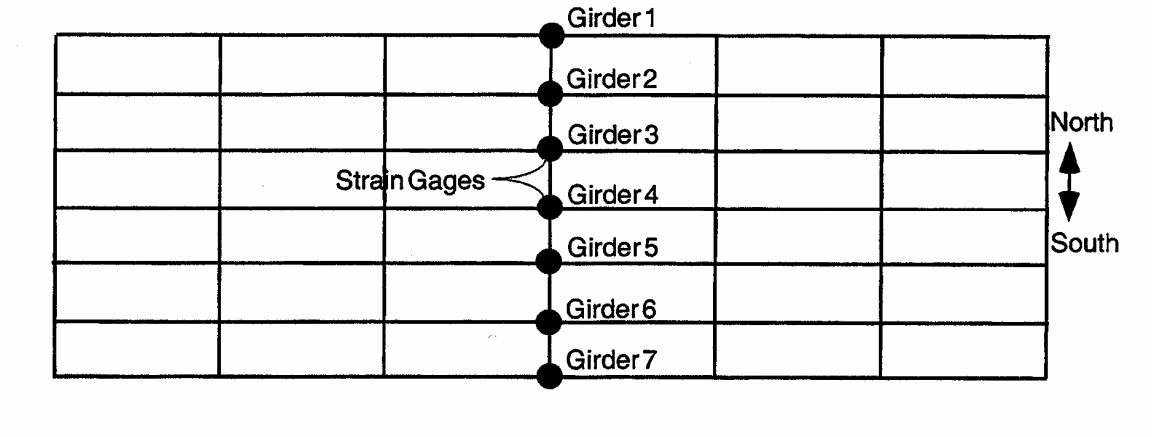
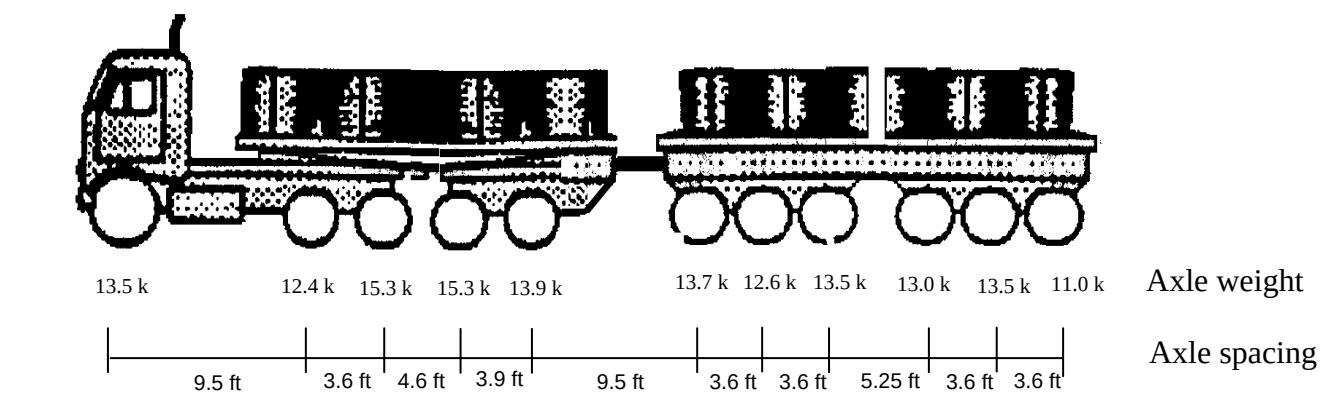


Figure 3.23 Layout of Strain Gauges of Michigan Test Bridge (Eom and Nowak 2001).



Gauge Width = 82 in (front axle), 70 in (rear axles)

Figure 3.24 Test Truck Configurations

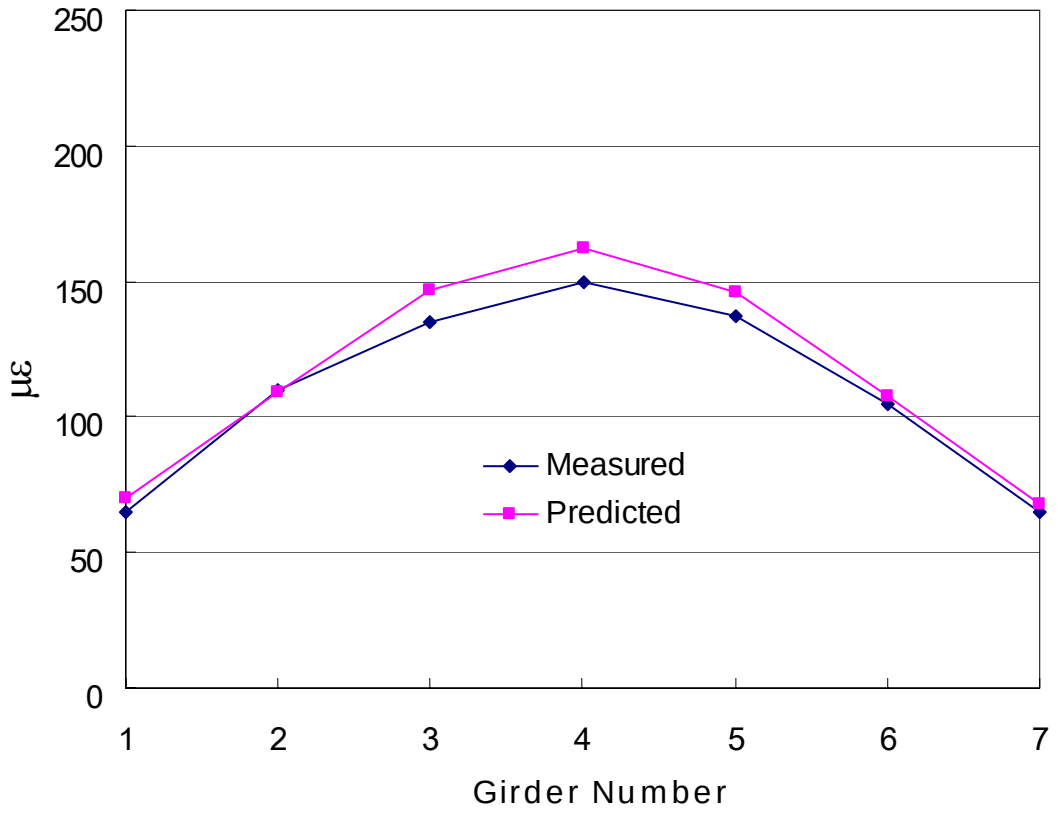


Figure 3.25 Bottom Flange Strains at Mid-span (Michigan Test).

3.8 Summary

The eccentric beam model has been selected for this study. This model utilizes the non-composite section properties of two elements to model composite action by applying rigid links between the centroid of the girder and the mid-surface of the slab. The bridge deck slab is modeled by shear flexible eight-node shell elements, and the steel girder is idealized by three-node Timoshenko beam elements. This element selection has been made in order to eliminate a potential incompatibility along the element boundaries. The bearings are modeled by assigning boundary conditions to the zero-dimensional elements at their actual location. To simulate the simply supported condition, rotations along all directions are allowed and minimum restraints are assigned for longitudinal and transverse movement while vertical restraints are placed at the supports. Kinematic constraints are also applied to nodal degree of freedom between the girders and the deck.

In this chapter, the developed equivalent nodal load algorithm with application to the finite element modeling of bridge deck is presented. In this method, the patch load is discretized into a number of uniformly distributed sub-point loads. This method uncouples the pressure load from the mesh size and provides an accurate representation of pressure load on the bridge deck.

It has been found that the developed finite element model is capable of predicting the behavior of steel girder bridges, including deflections, strains, and load distributions. Furthermore, the finite element model for slab on girder bridges provides a rational tool for the understanding of the behavior of bridge superstructures.

CHAPTER 4. MULTI-PARAMETER COMPARISON

4.1 General

The new equation for the wheel load distribution factor in the AASHTO-LRFD code involves many more parameters than the AASHTO-Standard code equation. This complicates the direct comparison of the two code equations. In order to facilitate the comparison, an applicable range for each parameter is defined based on the full range of bridge structures in Indiana. A database of existing Indiana bridges is constructed for this purpose. A number of bridge parameters are used and classified as follows:

1. NBI parameters

   Parameters available in NBI database. These include span length, transverse width, number of spans, total span length, skew angle, number of lanes, and construction year.

2. Girder parameters

   Parameters of girder information. These include girder spacing, number of girders, slab thickness, girder geometry, and material properties.

4.2 Indiana Bridge Database

In order to determine the effective range of parameters, the Indiana part of the National Bridge Inventory (NBI) database is extracted. This database has been prepared for use by state, federal, and other agencies in recording all details of bridge structures in the nation. The manual for inputting data into the database has been documented in a report, 'Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges' (FHWA 1995). The NBI database provides structural information as

well as management information for each specific bridge. Structural information includes data that is used in the code equation for wheel load distribution factor, such as bridge type, material, total span length, maximum span length, number of spans, transverse width, skew angle, number of traffic lanes, skew angle, and design load. Management information contains data on the identification of structures, owner, cost, year of construction, and inspection information. The NBI database, however, has no information regarding girder geometry and material properties.

The NBI database provides data for Indiana bridges currently in service. The total number of Indiana bridges available in the NBI is 19,321. Figure 4.1 through Figure 4.4 show the part of NBI data based on the following criteria: owner, type of service, material, and type of design. Figure 4.1 shows that 30% of the bridges in Indiana are managed by state agencies. Figure 4.2 shows that about 90% of the state owned bridges are serviced for highway bridges. Figure 4.3 shows the material used in state-owned highway bridges. Figure 4.4 shows that the majority of bridges present in Indiana are of the slab-on-girder configuration.

The database for this study includes steel I-girder bridges designed using the AASHTO HS-20 design truck. Only state-owned highway bridges are included. With this scope, the total number of steel girder bridges is reduced to 1,255.

4.3 Applicable Range

An applicable range for each NBI parameter is determined from the statistical analysis of the new database. The bridges that deviate from typical parameters are eliminated from the database and not considered further.

Figure 4.5 through Figure 4.10 show the results of the statistical study in the form of histograms. Figure 4.5 shows that two-span bridges are the most popular span in Indiana steel bridges, followed by three-span continuous bridges. The common ranges of span length and transverse deck width are 60 ft – 85 ft and 30 ft - 50 ft, respectively, as shown in Figure 4.6 and Figure 4.8. The histograms of the maximum span length in simple, two span, and three span bridges are given in Figure 4.7.

Figure 4.9 shows that a majority of bridges are constructed with right-angled support, and Figure 4.10 shows that most bridges are two-lane bridges.

The distribution of span length and transverse width as shown in Figure 4.6 and Figure 4.8 is assumed to be normal distribution. Those parameters are then considered "applicable" only if data is within 80 % (± 1.5×standard deviation) of the average value. The range of span lengths is from 44 ft to 122 ft, and the range of transverse widths is from 28 ft to 55 ft. The number of spans is selected from simple span to three spans. Only two and three lane bridges are chosen. The skew angle along the supports is selected from 0 degrees to 45 degrees. Statistical analysis is not required for other considered NBI parameters (number of spans, number of lanes, and skew angle) due to obvious trends in the data.

4.4 Representative Bridges

Because of the lack of girder parameters in the database, a total of 43 of the actual representative bridges in the applicable range are selected so that the girder parameters could be obtained from the actual plans. The NBI parameters of selected representative bridges are required to be distributed over all bridges in the applicable range. The list of 43 Indiana Representative Bridges are shown in

Table 4.1.

To check the validity of the representative bridges, the selected data points for several NBI parameters are plotted in the scattergrams shown in Figure 4.12 through Figure 4.15. If points in the scattergrams are well distributed, it means all representative bridges in the applicable range are covered. For example, Figure 4.12 shows the relationship between span length and transverse width. It is observed that the scattered points are equally distributed within the applicable range. The other scattergrams also show that the selected bridges are rationally distributed and that the representative bridges are valid.

The girder parameters of the selected 43 bridges are gathered with the cooperation of the Indiana Department of Transportation (INDOT). The information includes girder spacing, number of girders, slab thickness, and geometric dimension of girders. Figure 4.16 and Figure 4.17 show the variation of the girder spacing and the number of girders in the representative bridges. It is observed that the girder spacing ranges between 4.3 ft and 10.0 ft and that the number of girders varies from 5 to 11. Furthermore, several NBI parameters and girder parameters are plotted against each other in Figure 4.18 through Figure 4.21. It is observed that there is no significant relationship between the girder spacing and the span length or between the longitudinal stiffness and the girder spacing. However, as the span length increases, the longitudinal stiffness and the girder height also increase, as shown in Figure 4.20 and Figure 4.21. A summary of the statistical study is given in Table 4.2.

4.5 Sensitivity of Bridge Parameters

The AASHTO-LRFD wheel load distribution equations are compared with the AASHTO-Standard formulas. In order to investigate the sensitivity of each parameter used in the formulas, a "mean" bridge is selected for each span. The "mean" bridge is determined in such a way that all parameters of the bridge are close to the average value of each parameter. The specific procedure for selecting Indiana "mean" bridges is summarized as follows:

1. An applicable range of each NBI parameter is determined based on the statistical analysis using total Indiana steel girder bridges.
2. Representative bridges are selected within the applicable range.
3. Girder information of representative bridges is collected from actual plans.
4. The Indiana "mean" bridge is determined among the representative bridges.

The summary of "mean" bridges is given in Table 4.3. The mean three-span bridge, for example, is a 42-56-42 ft continuous bridge. All of the parameters provided in Table

4.3 are close to the mean value. The maximum span length is 56 ft, which is close to the common value of 60 ft as shown in Figure 4.7 (c). The transverse width of the mean bridge is 36 ft, which is the common value in Figure 4.8. The “mean” bridge is the real existing bridge whose parameters are closest to the mean values among representative bridges.

Wheel load distribution factors calculated from AASHTO specifications are compared within the applicable range of various bridge parameters from Figure 4.22 to Figure 4.27. Variations from the mean bridge of each span type are considered by changing values of bridge parameters one at a time. Parameter variations for which parameter sensitivity studies are performed are same as applicable ranges.

Figure 4.22 shows the wheel load distribution factors with varying girder spacing for simple span, two span, and three-span. AASHTO-Standard formula is close to AAHSTO-LRFD when girder spacing is short but becomes conservative as girder spacing increases. Figure 4.23 illustrates the sensitivity of span length to code equations. Wheel load distribution in the AASHTO-standard code is constant since the span length parameter is not included in the formula. It is observed that AASHTO-LRFD predicts lower distribution factor in any case of continuity when span length is larger than 60 ft. Other parameters compared are slab thickness, longitudinal stiffness, unitless inertia and skew angle and are included in AASHTO-LRFD. Figure 4.24 through Figure 4.27 show that these parameters are less sensitive in determining wheel load distribution factor.

It can be concluded from the multi-parameter comparisons that the girder spacing parameter in both AASHTO codes has the greatest influence on wheel load distribution in Indiana bridges. Next sensitive parameter is span length. Other parameters do not significantly influence the load distribution.

Finally, the wheel load distribution factors of Indiana representative bridges are plotted against girder spacing in Figure 4.28. AASHTO-Standard formula is more conservative for large spacing and less conservative for short spacing than AASHTO-LRFD specification.

Table 4.1 Representative Indiana Steel Girder Bridges

No.	NBI Structure Number	County	Location	Year built (rebuild)	Skew Angle	Number of Spans	Max. Length [ft]	Girder Spacing [ft]	Girder Type
1	5870	Newton	US 41	1953 (1999)	0	1	48	5.42	W24x162
2	6330	Tippecanoe	SR 28	1995	0	1	110	9.00	Plate girder
3	13330	Clinton	SR 38	1953 (1983)	45	1	60	5.83	W36x182
4	17030	Clay	SR 59	1933 (1984)	30	1	46	4.30	W27x114
5	18370	Martin	SR 550	1965 (1992)	36	1		10.00	Plate girder
6	23408	Vanderburgh	I-164	1988	0	1	122	6.58	Plate girder
7	27340	Porter	US 6	1955 (1999)	0	1	59	5.50	W30x132
8	27350	Porter	US 6	1955 (1986)	30	1	72	6.06	W36x170
9	29660	Montgomery	US 231	1936 (1983)	15	1	60	4.33	W33x201
10	960	Delaware	SR 67	1973 (1986)	0	2	67	5.75	W36x150
11	11658	Hendricks	SR 267	1986	15	2	120	7.66	Plate girder
12	32836	Allen	SR 1	1990	22	2	102	8.50	Plate girder
13	34960	Scott	SR 56	1985	0	2	112	5.67	Plate girder
14	35090	Jackson	US 31	1959 (1985)	0	2	65	5.25	W36x150
15	35760	Johnson	SR 44	1970	31	2	98	6.33	W36x230
16	35890	Johnson	SR 44	1971	14	2	88	8.00	W36x230
17	37270	Boone	US 52	1970	17	2	110	6.33	Plate girder
18	38160	Jasper	SR 14	1966 (1992)	0	2	98	5.25	W36x135
19	40120	Allen	US 24	1989	20	2	121	7.33	Plate girder
20	43860	Fountain	US 136	1959 (1986)	45	2	85	6.50	W36x182
21	75030	St Joseph	SR 331	1991	0	2	94	7.25	Plate girder
22	1620	Porter	SR 149	1985	45	3	78	7.00	W33x152
23	12350	Morgan	SR 44	1955 (1989)	0	3	75	5.42	W33x130
24	13057	Washington	SR 256	1992	0	3	103	8.50	Plate girder
25	14320	Vanderburgh	SR 66	1986	0	3	104	6.00	Plate girder
26	16080	Putnam	SR 236	1981	30	3	65	7.42	W36x150

Table 4.1 Representative Indiana Steel Girder Bridges (continued)

No.	NBI Structure Number	County	Location	Year built (rebuild)	Skew Angle	Number of Spans	Max. Length [ft]	Girder Spacing [ft]	Girder Type
27	17860	Ripley	SR 129	1985	0	3	56	6.50	W24x94
28	18314	Daviess	SR 57	1990	18	3	65	7.75	W33x118
29	23490	Spencer	US 231	1932 (1982)	0	3	85	6.17	W33x141
30	25000	Carroll	SR 218	1965 (1982)	0	3	78	5.33	W33x130
31	33162	Lake	I-80	1997	4	3	55	9.17	W36x150
32	34330	Floyd	SR 64	1972	27	3	97	6.67	W33x130
33	70430	Vanderburgh	I-64	1989	45	3	97	8.00	Plate girder
34	75310	Elkhart	US 33	1991	16	3	61	6.83	W30x116
35	76160	Allen	US 24	1992	2	3	122	7.50	Plate girder
36	210	Wayne	US 40	1962 (1987)	0	4	65	5.33	W30*124
37	11275	Cass	SR 25	1962 (1990)	17	4	57	5.83	W33x130
38	14620	Knox	US 50	1967 (1996)	42	4	56	4.17	W21x166
39	17980	Porter	SR 2	1987	44	4	106	8.00	W36x150
40	21260	Clay	SR 46	1955 (1984)	0	4	93	5.17	W36x232
41	33010	Martin	US 50	1956 (1984)	0	4	60	5.25	W30x108
42	33043	Lake	SR 912	1977	18	4	93	6.10	W36x194
43	31980	Knox	SR 67	1985	0	5	108	7.75	Plate girder

Table 4.2 Summary of Statistical Study

Item	Number of data	Min	Max	Mean	Standard deviation (σ)	Selected Range (± 1.5σ)
Number of spans	1255	1	17	2	-	1-3
Span length	1255	25 ft	217 ft	83 ft	26 ft	44 ft – 122 ft
Transverse width	1255	24 ft	124 ft	40 ft	15.8 ft	28 ft – 55 ft
Skew angle	1255	0 °	64 °	18 °	-	0-45
Number of lanes	1255	1	6	2	-	2-3

Table 4.3 Selected Mean Bridges

Span	No.	L (ft)	W (ft)	S (ft)	Number of girders	$K_g\,(in^4)$	Year
1	27340	60	46.6	5.5	8	182600	1955(1993)
2	38160	98-98	30	5.25	6	242644	1966(1992)
3	17860	42-56-42	36	6.5	6	90453	1991

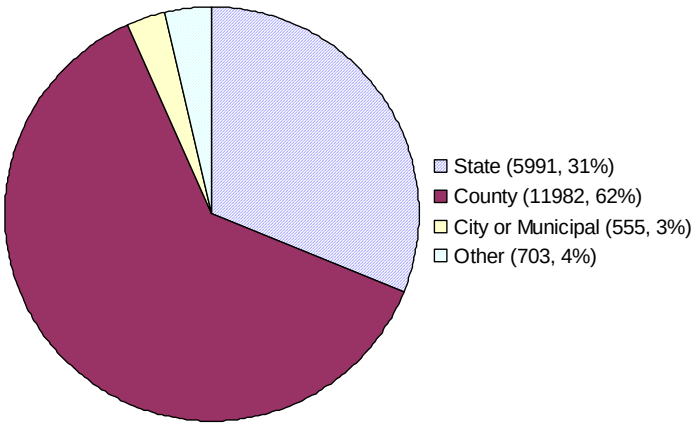


Figure 4.1 Indiana Bridge Database (Owner).

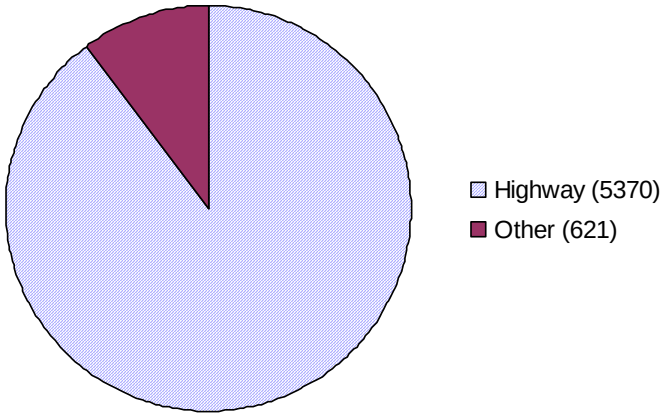


Figure 4.2 Indiana Bridge Database (Type of Service).

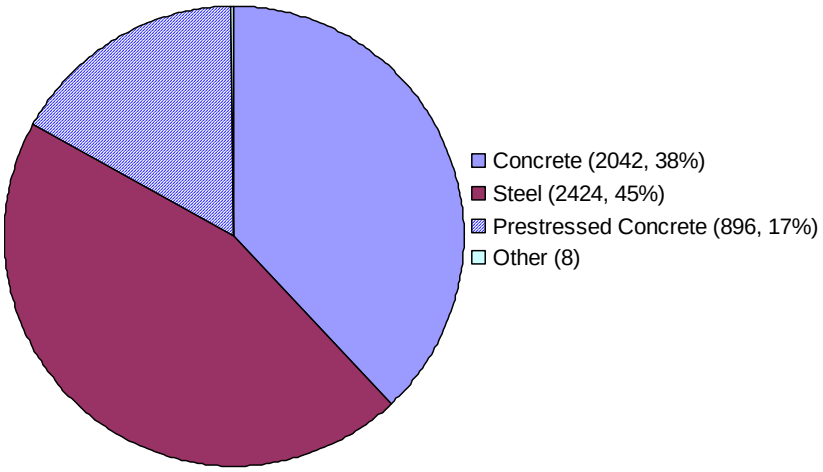


Figure 4.3 Indiana Bridge Database (Material).

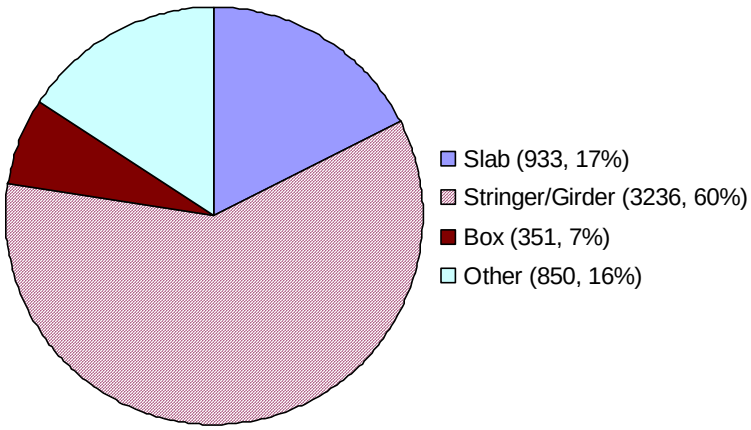


Figure 4.4 Indiana Bridge Database (Type of Design).

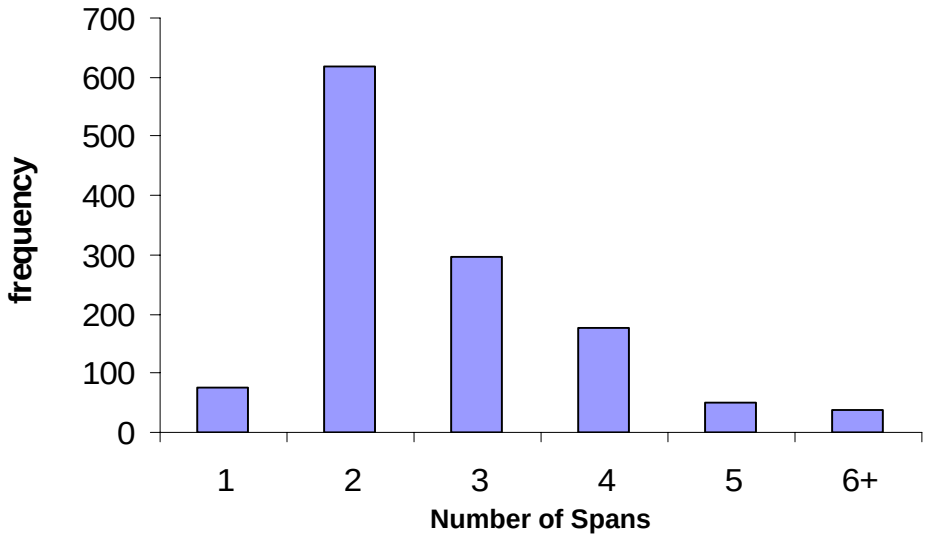


Figure 4.5 Indiana Bridge Histogram (Number of Spans).

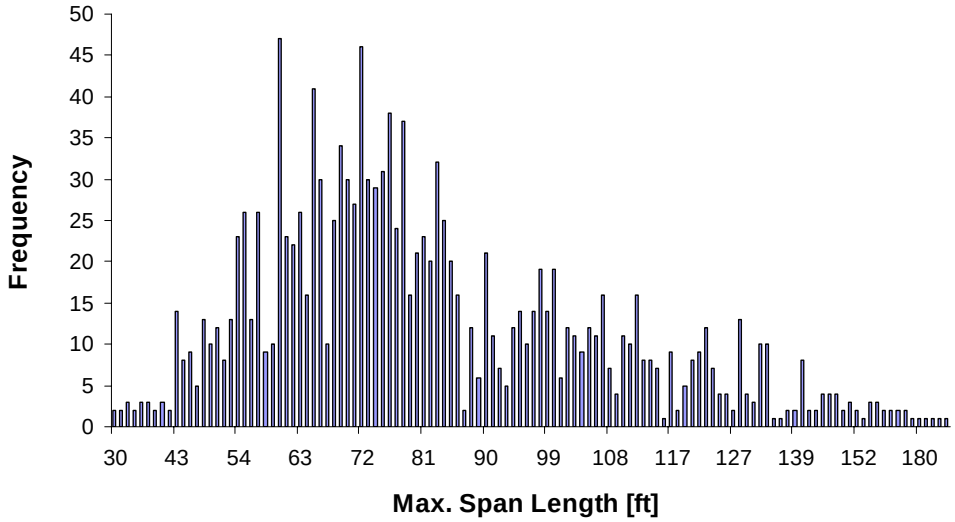
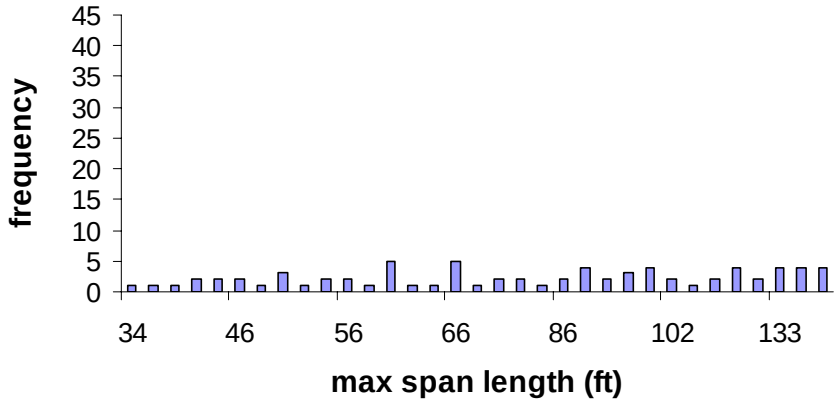
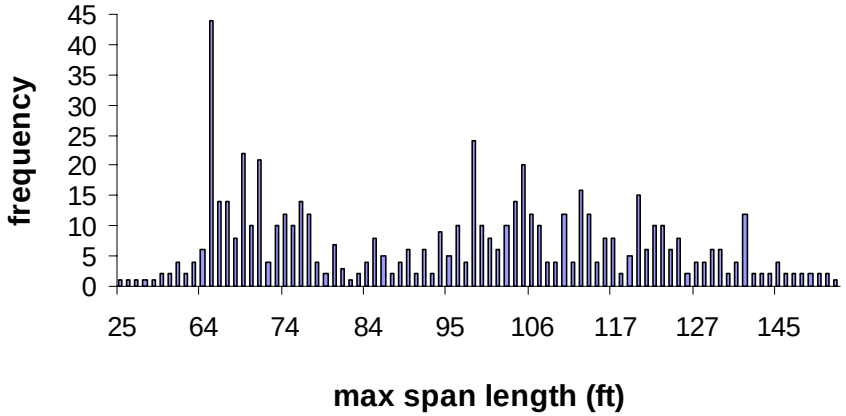


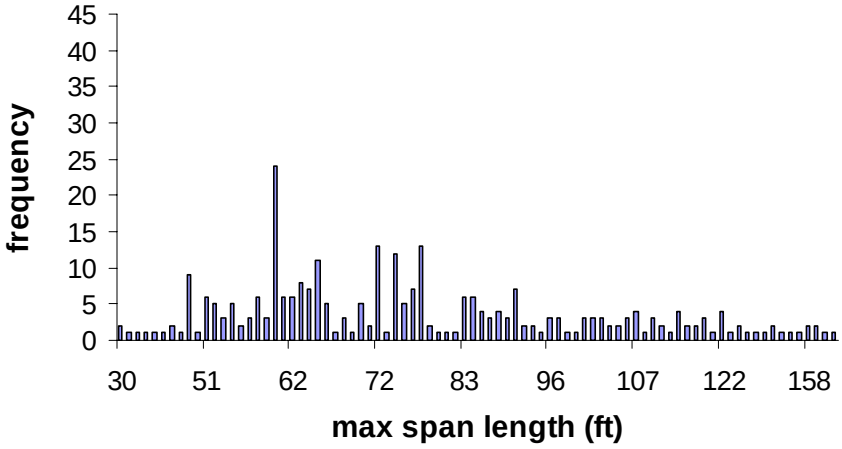
Figure 4.6 Indiana Bridge Histogram (Span Length).



(a) Simple Span Bridges



(b) Two Span Bridges



(c) Three Span Bridges

Figure 4.7 Maximum Span Length.

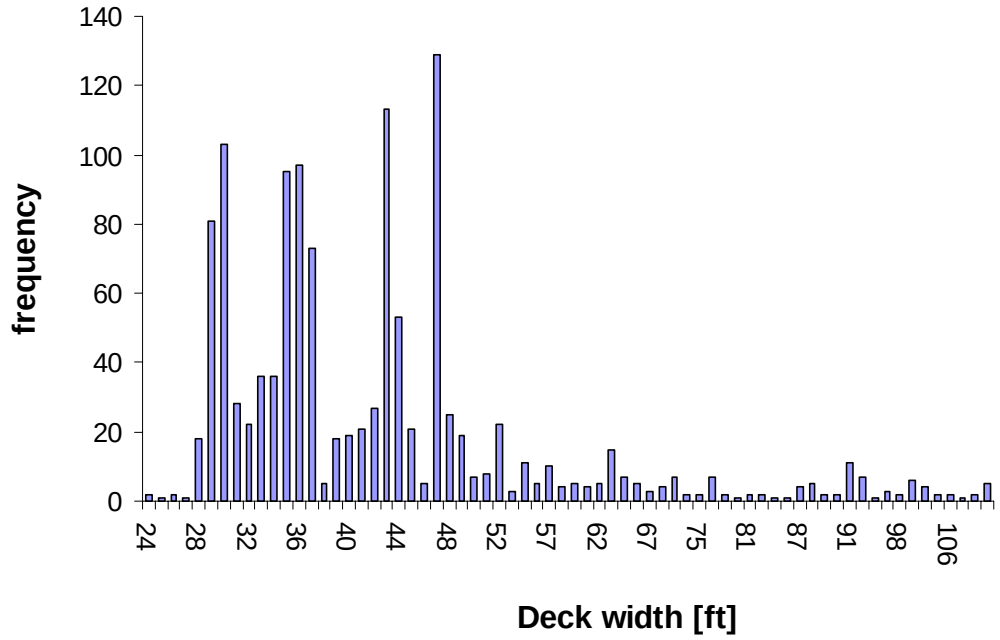


Figure 4.8 Indiana Bridge Histogram (Transverse Width),

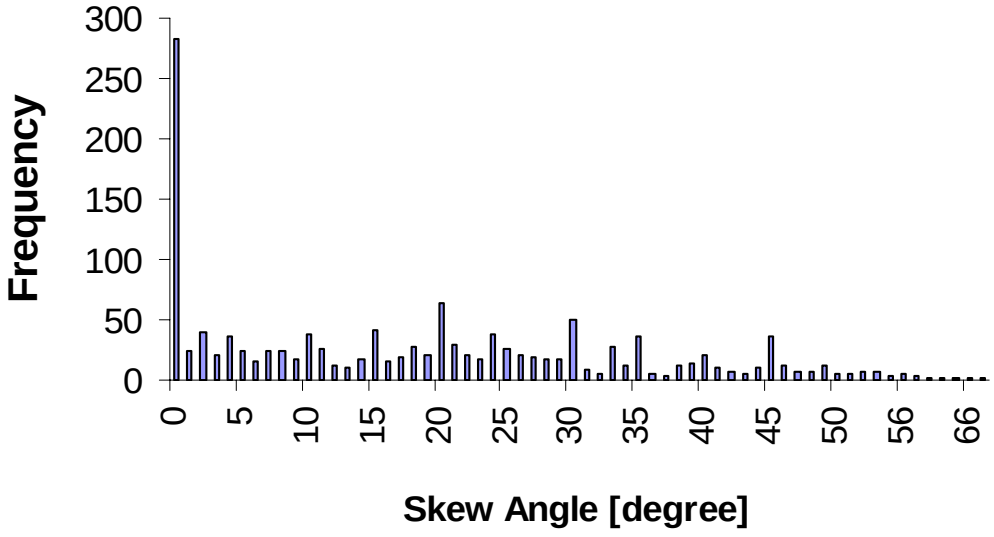


Figure 4.9 Indiana Bridge Histogram (Skew Angle).

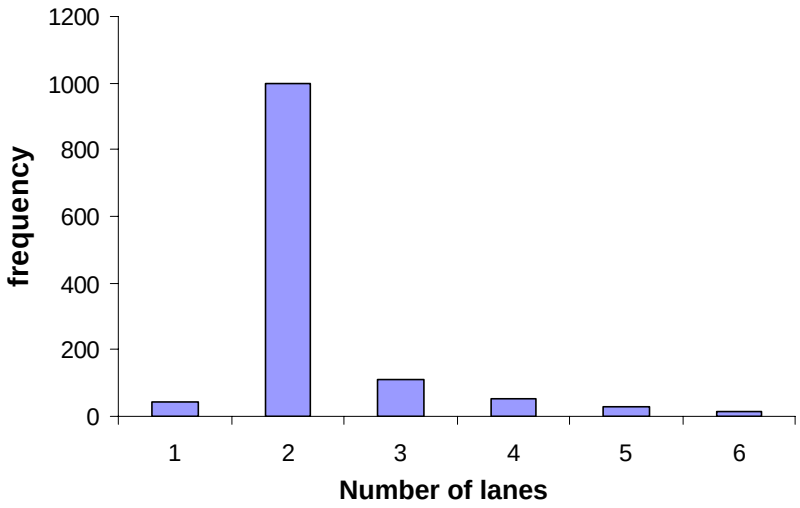


Figure 4.10 Indiana Bridge Histogram (Traffic Lane).

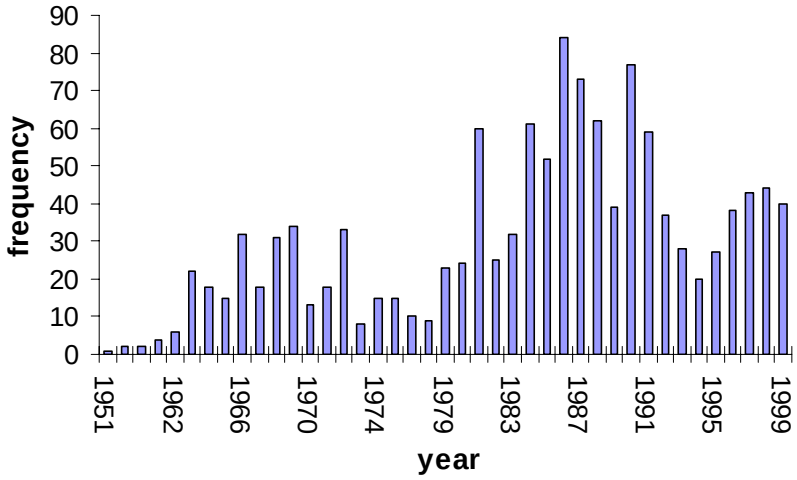


Figure 4.11 Indiana Bridge Histogram (Year Built or Reconstructed).

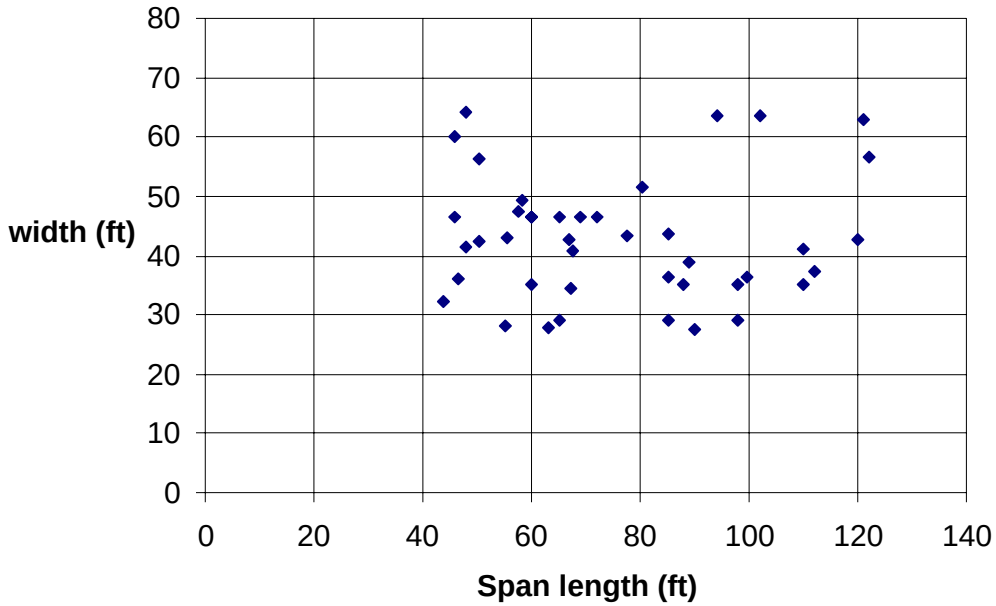


Figure 4.12 Representative Bridge Scattergram (NBI Parameters, Width vs. Span)

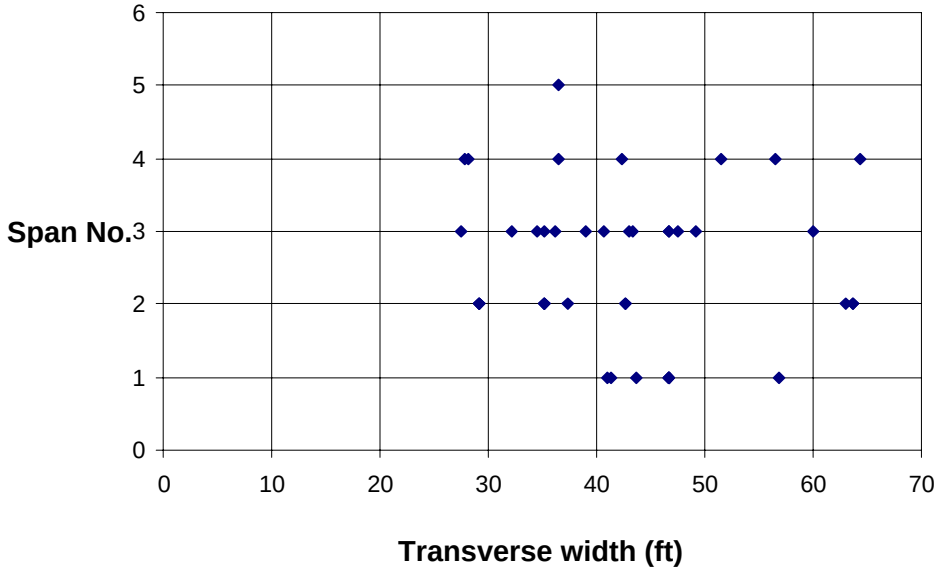


Figure 4.13 Representative Bridge Scattergram (NBI Parameters, Number of Spans vs. Width)

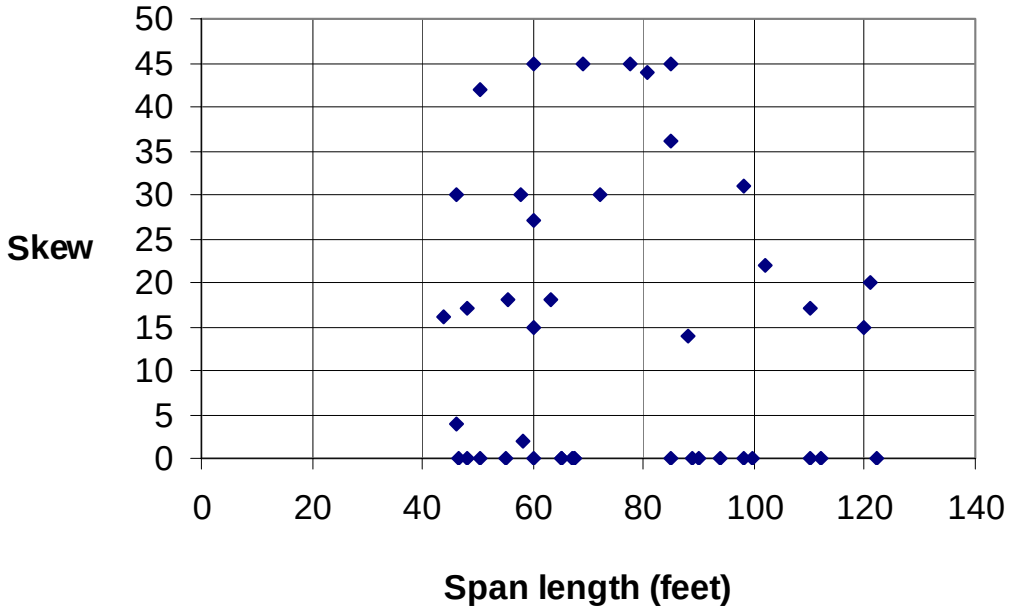


Figure 4.14 Representative Bridge Scattergram (NBI Parameters, Skew vs. Number of Spans).

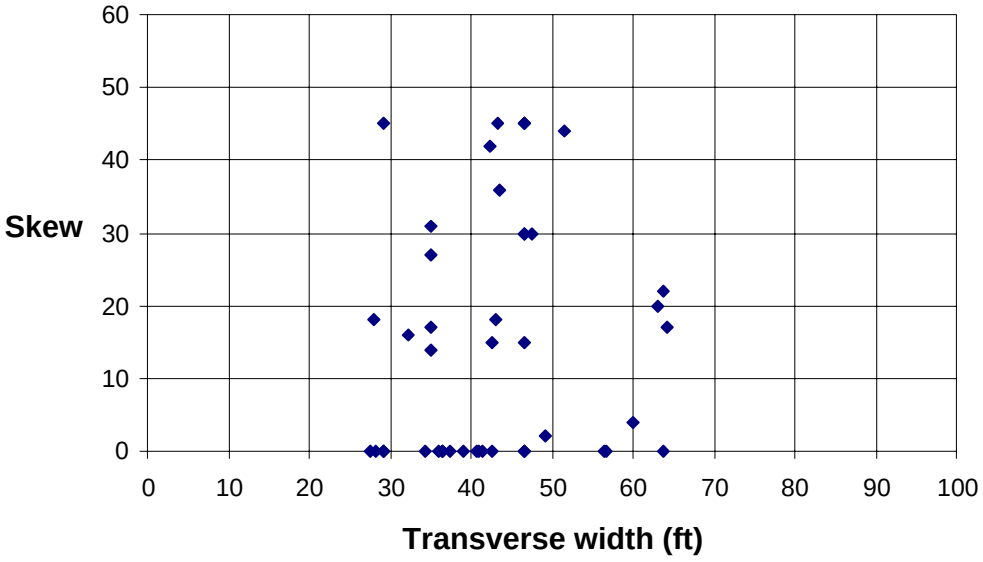


Figure 4.15 Representative Bridge Scattergram (NBI Parameters, Skew vs. Width).

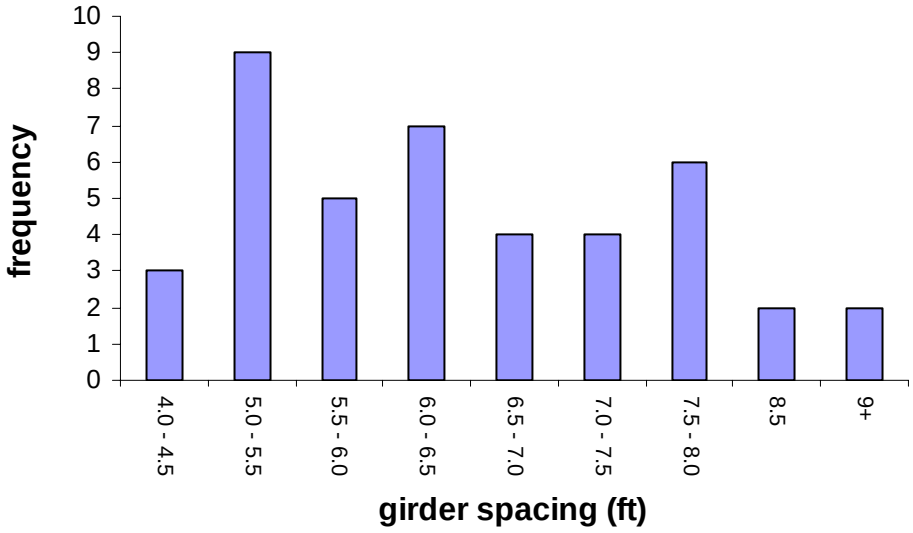


Figure 4.16 Representative Bridge Histogram (Girder Spacing)

Figure 4.17 Representative Bridge Histogram (Number of Girders).

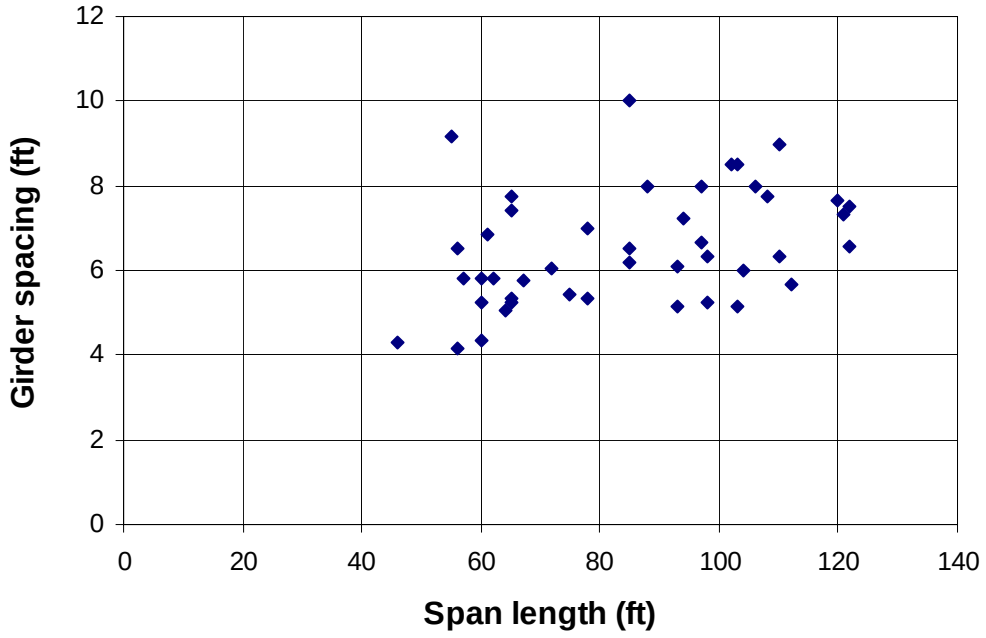


Figure 4.18 Representative Bridge Scattergram (Girder Spacing vs. Span).

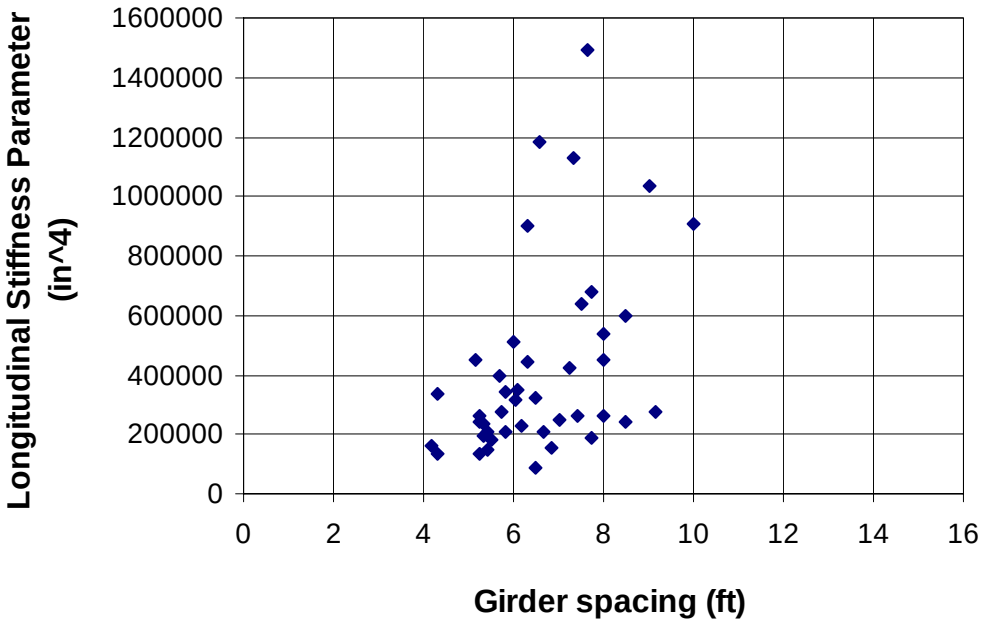


Figure 4.19 Representative Bridge Scattergram (Girder Spacing vs. Stiffness).

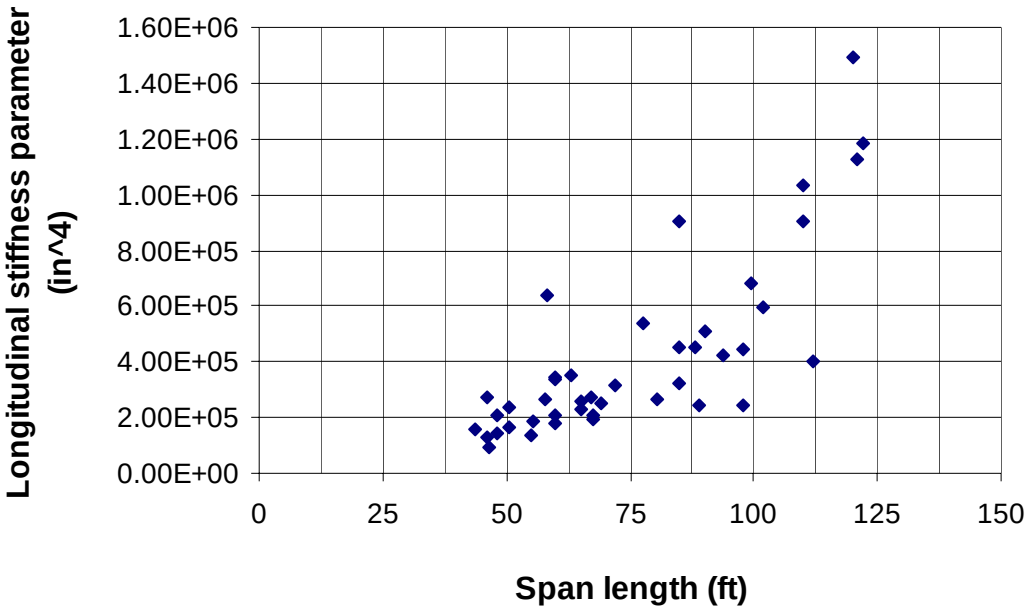


Figure 4.20 Representative Bridge Scattergram (Stiffness vs. Span).

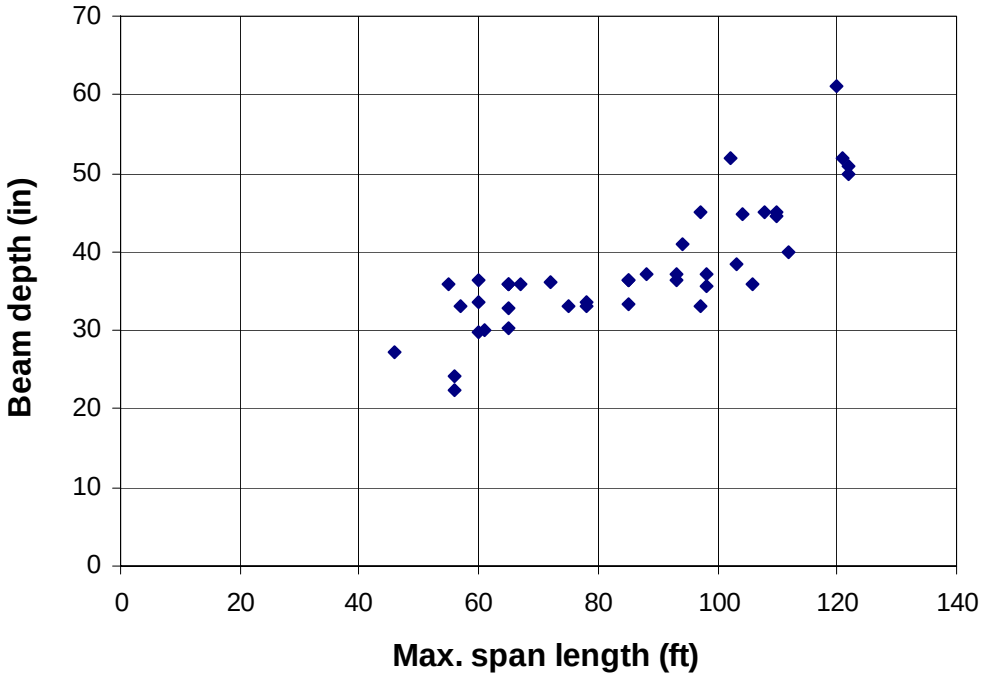
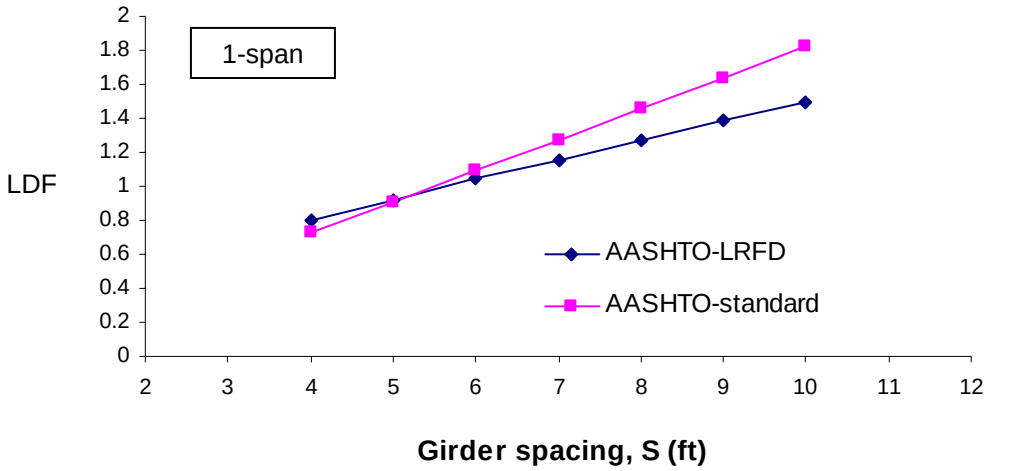
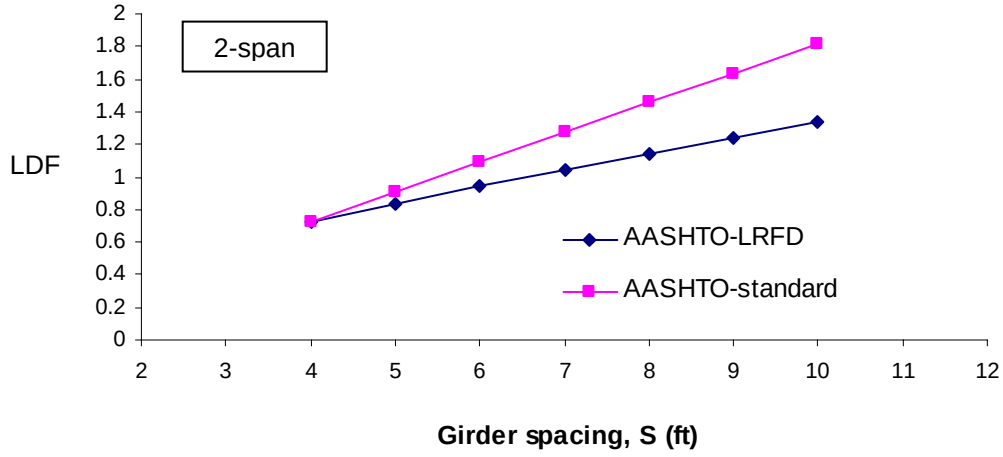


Figure 4.21 Representative Bridge Scattergram (Girder Depth vs. Span).





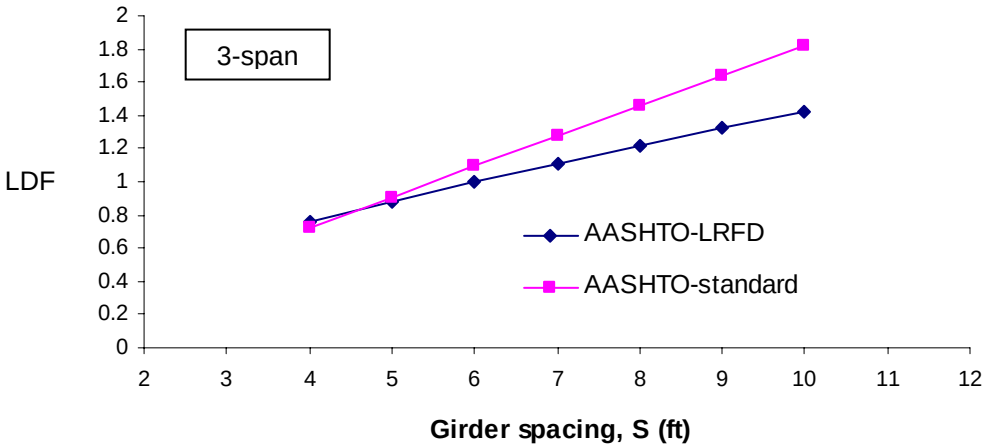
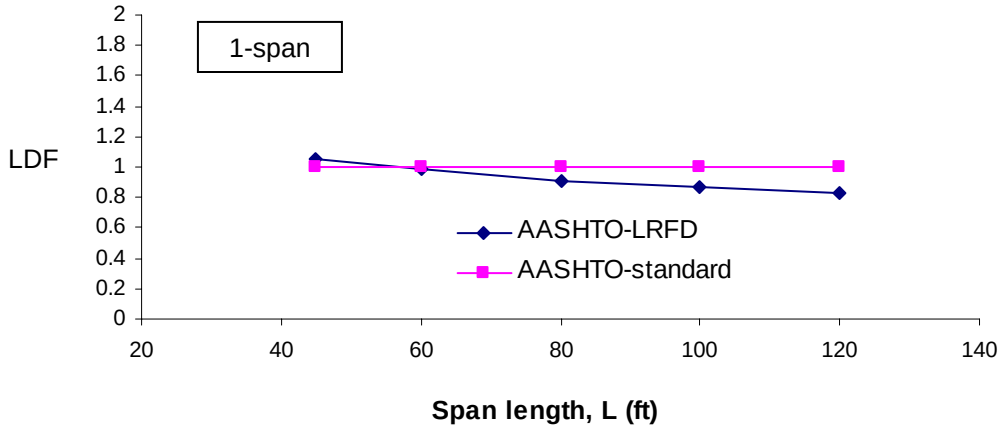
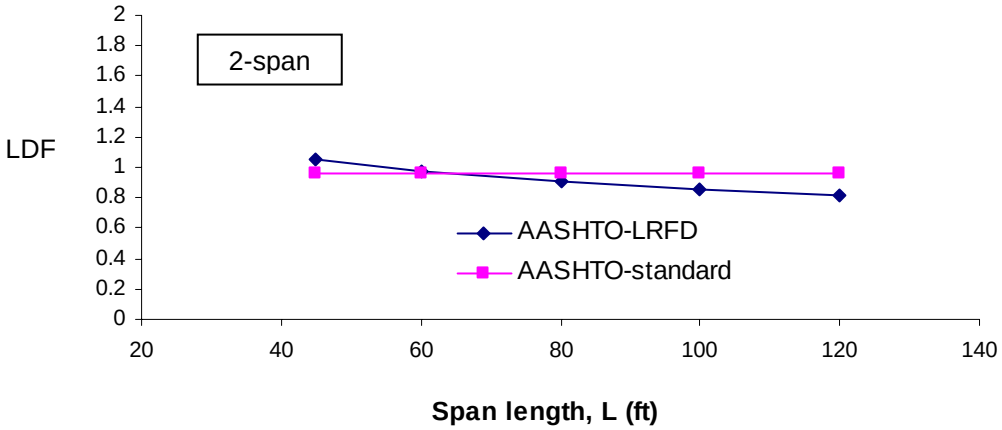


Figure 4.22 Comparison of Specifications (Girder Spacing).





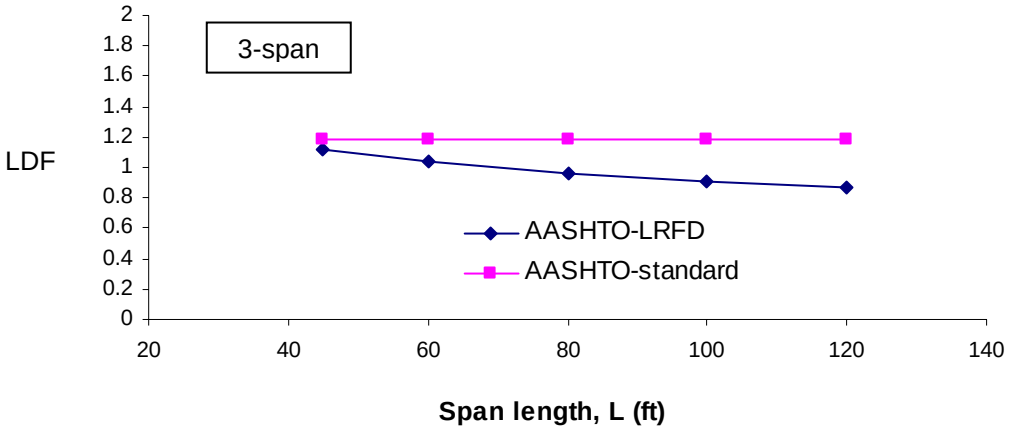
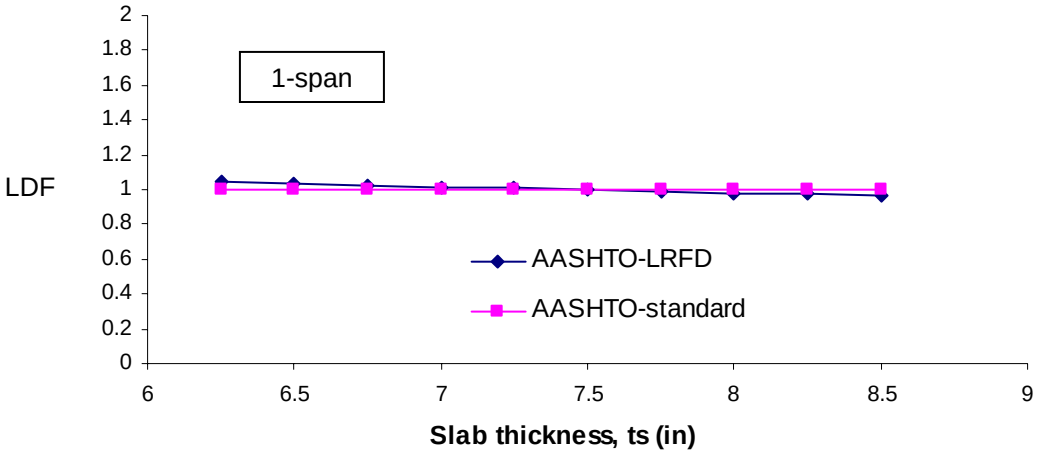
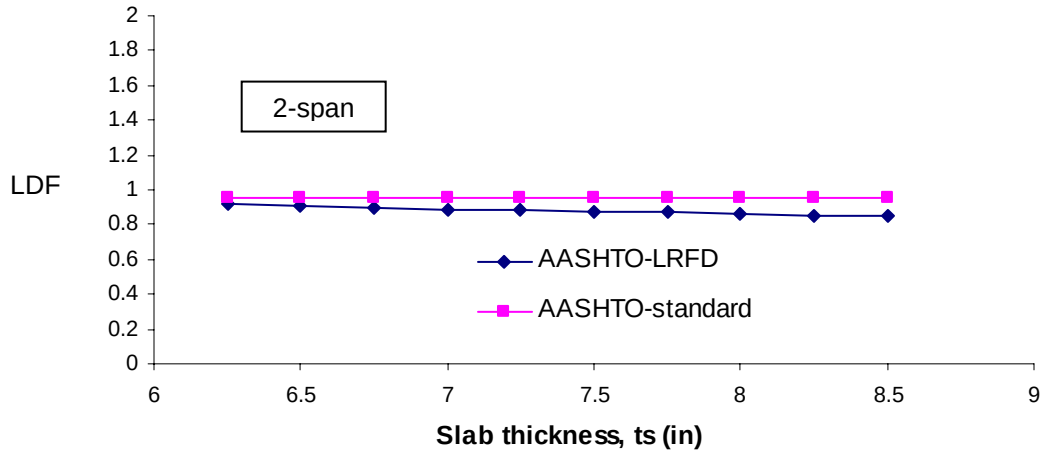


Figure 4.23 Comparison of Specifications (Span Length).





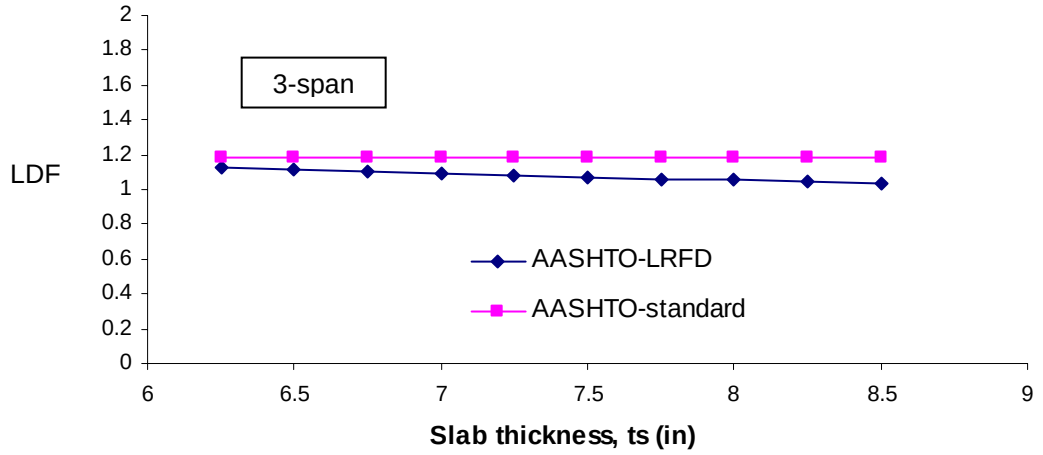
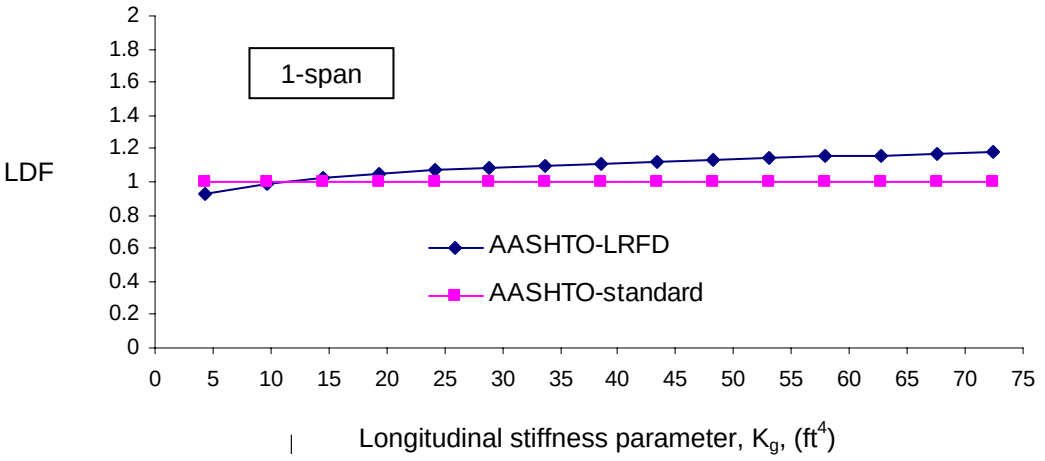
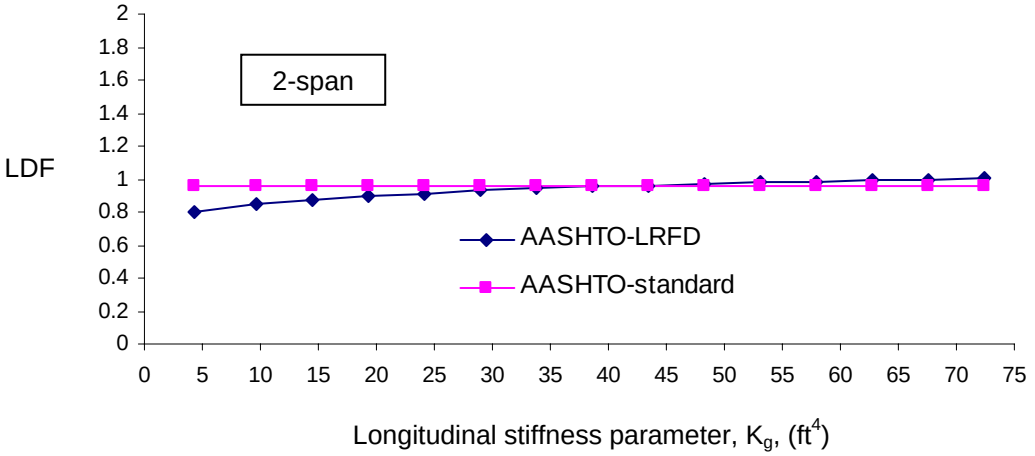


Figure 4.24 Comparison of Specifications (Slab Thickness)





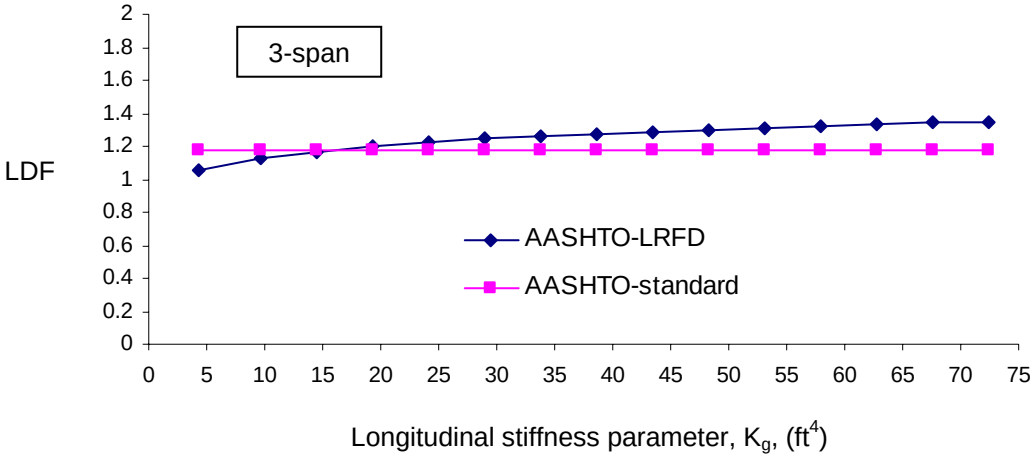


Figure 4.25 Comparison of Specifications (Longitudinal Stiffness Parameter).

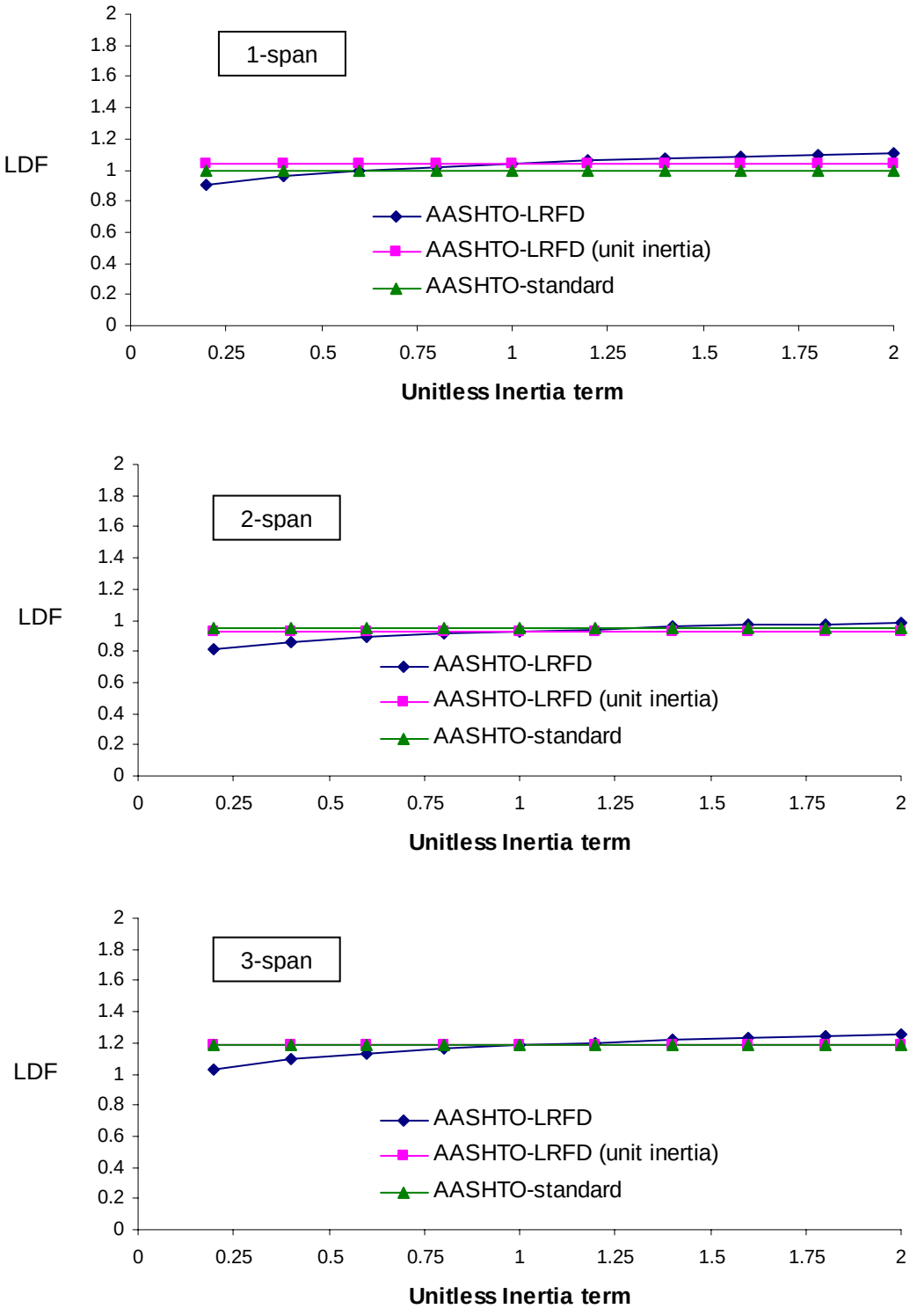
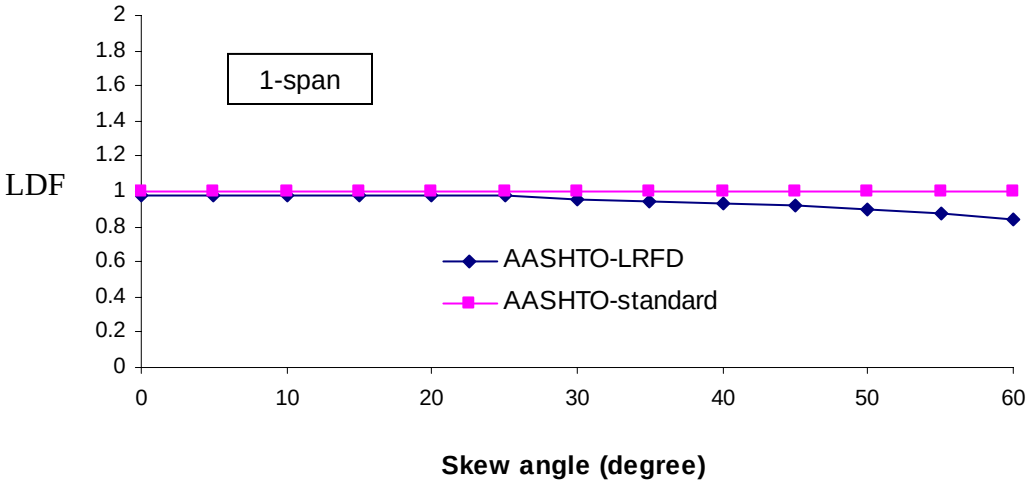
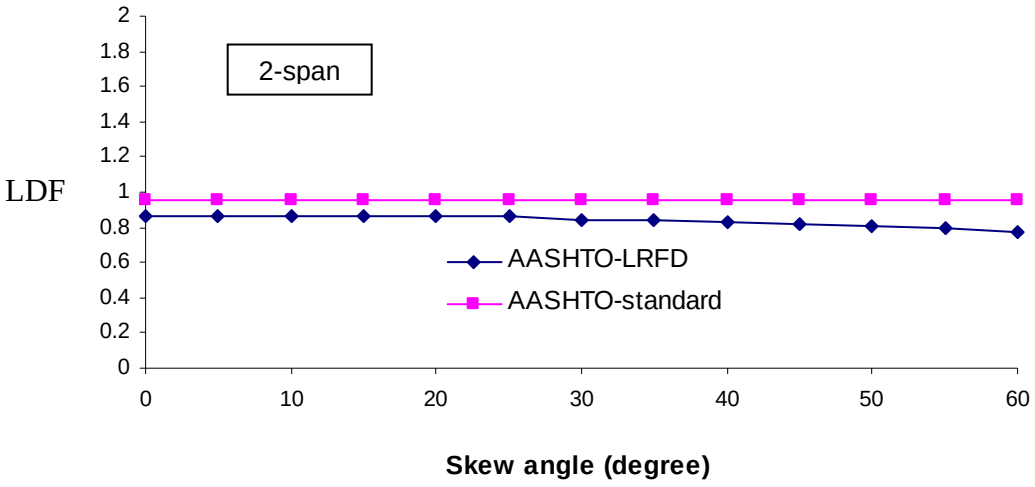


Figure 4.26 Comparison of Specifications (Unitless Inertia).





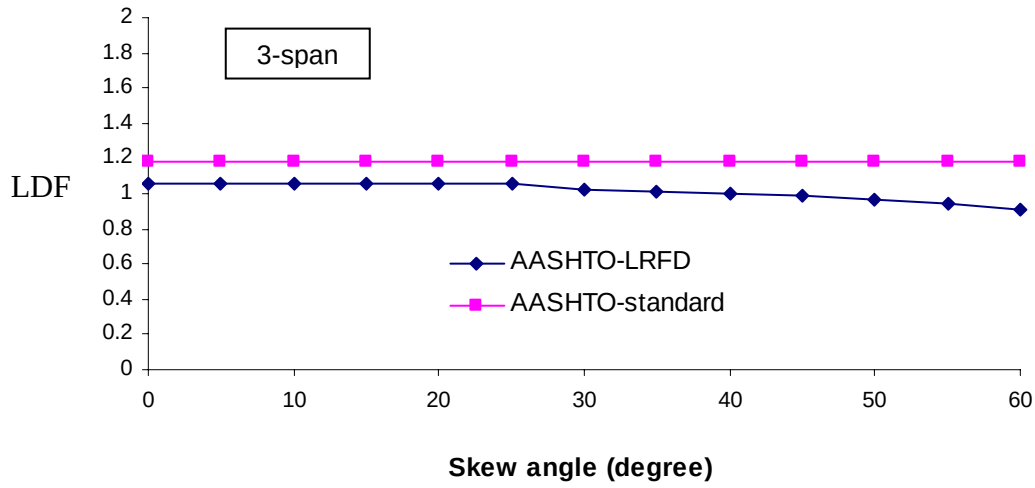


Figure 4.27 Comparison of Specifications (Skew Angle).

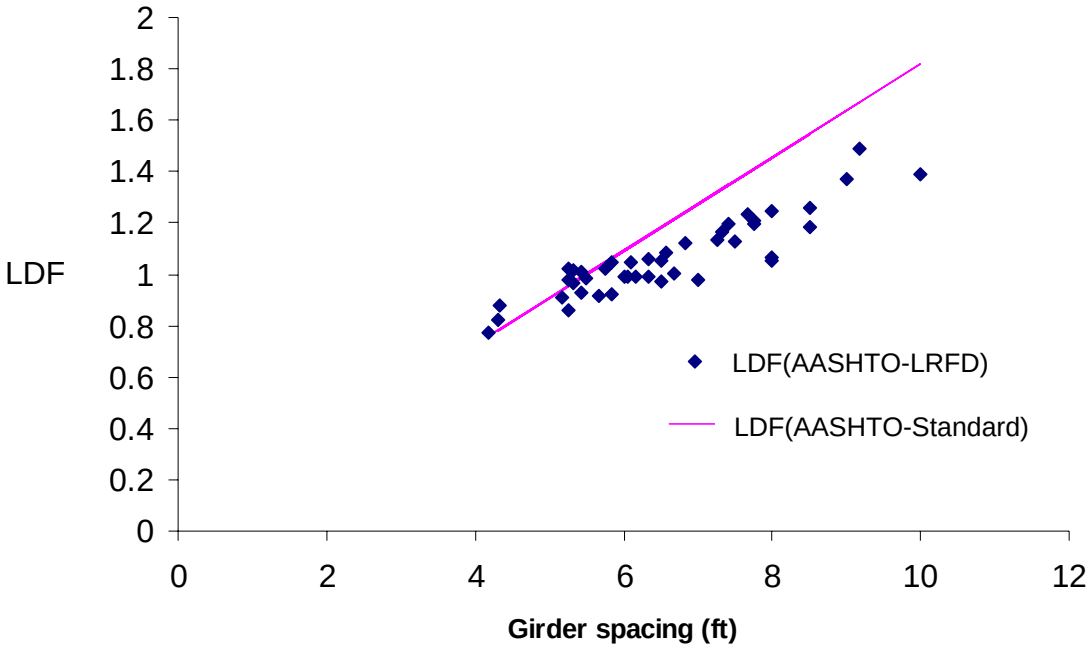


Figure 4.28 Wheel Load Distribution of Indiana Representative Bridges.

CHAPTER 5. POSTULATION OF SIMPLIFIED EQUATION

The main objective of this research is to provide the simplest, yet sufficiently accurate equation for calculation of load distribution. A new simplified wheel load distribution factor (LDF) equation, based on the current AASHTO-LRFD LDF formula is postulated in this chapter. The longitudinal stiffness parameter (K_g) and the slab thickness parameter (t_s) that appeared in LRFD equation are implicitly embedded in the simplified expression. This eliminates the iterative procedure introduced by LRFD formula. The accuracy and applicability of the simplified equation are demonstrated through comparisons of LDF calculated by AAHSTO-Standard, AASHTO-LRFD, and AASHTO level three analysis, namely finite element (FE) analysis.

5.1 Parameter Selection

The AASHTO-LRFD formula contains four parameters: girder spacing, span length, longitudinal stiffness, and slab thickness. In the formulation of the new simplified LDF equation, the sensitivity of the LDF to each parameter is considered. The goal is to eliminate parameters for which the LDF is not as sensitive as others, as well as those that require iterative design procedure.

According to the sensitivity studies performed both in the NCHRP 12-26 Project (Zokaie et al. 1991) and in the previous chapter, girder spacing (S) was the most sensitive parameter in computation of the LDF. Span length (L) is the next most sensitive parameter and longitudinal stiffness (K_g) somewhat influences the LDF. The LDF appears to be least sensitive to changes in slab thickness (t_s).

Based on the results from the sensitivity studies, some parameters were kept and others eliminated from the new simplified LDF equation. Girder spacing and span length are kept since from the sensitivity study these parameters are identified as having the most influence on the LDF value. The slab thickness of bridges in Indiana is typically 8 inches (200 mm), so this parameter is eliminated and the thickness is assumed to be 8 inches (200 mm).

Meanwhile, the longitudinal stiffness parameter is also to be eliminated. The longitudinal stiffness parameter is introduced in AASHTO-LRFD to increase the accuracy of the equation (Zokaie et al. 1991a, 1991b, 2000). Since the section properties of the girder are not known prior to determination of the LDF, the third term in the LRFD equation, which contains the longitudinal stiffness parameter, is assumed to be a unit value in the first calculation. After the girder section is determined, the LDF equation is reevaluated to check the strength criterion. This iterative procedure is cumbersome since redesign may be required. The intention is to eliminate the longitudinal stiffness parameter and the need for an iterative procedure in the LDF calculation.

The longitudinal stiffness parameter (K_g) is found to be related to the span length parameter (L). In Figure 5.1, the longitudinal stiffness of the Indiana representative bridges and the bridge database from the NCHRP project 12-26 are plotted versus the span length. The general trend of the relationship is that K_g increases as L increases, but the data are scattered in relatively wide range. One of the reasons may be the bridge engineer's preference on the selection of girder dimensions.

The K_g-L relationship is adjusted to account for the constant slab thickness of 8 inches in Indiana. The slab thickness of the bridges available in the database of the NCHRP 12-26 project varies from 4.5 inches to 12 inches. K_g values of the bridges are calibrated to a reference slab thickness 8 inches as shown in Figure 5.2. As it can be seen, the change of the slab thickness gives minor influence on the K_g-L relationship.

An upper bound relationship between K_g and L can be defined by the exponential trendline regression. This trendline covers all bridges in a conservative manner, as shown in Figure 5.3, and it is given by

$$K_g = 189940 \cdot e^{\left(\frac{L}{59}\right)} \quad \text{(US customary units)}$$
$$K_g = 7.91 \times 10^{10} \cdot e^{\left(\frac{L}{18000}\right)} \quad \text{(SI units)} \tag{5.1}$$

With this relationship almost all of the bridges are conservatively represented. The two bridges that do not fall below the trend line will be verified later by the finite element analysis. This K_g-L relationship will be used in the construction of the new simplified equation.

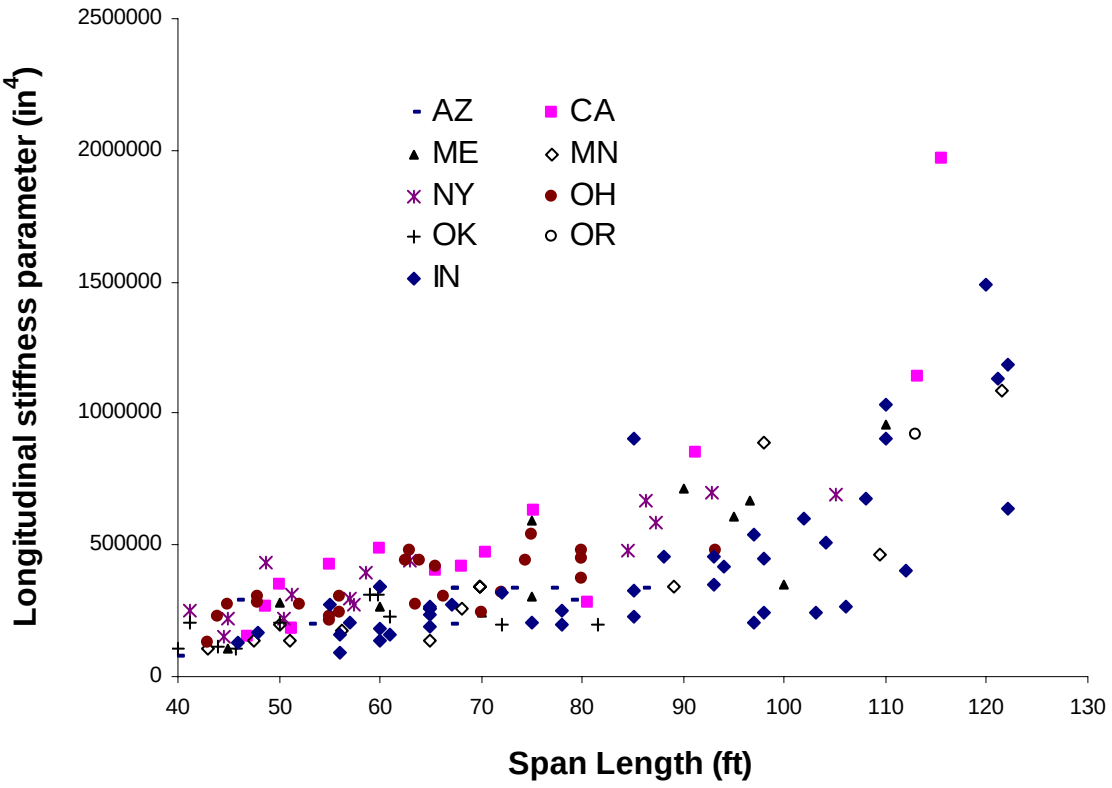


Figure 5.1 Scattergram of Longitudinal Stiffness Parameter and Span Length (from NCHRP 12-26 Database and Indiana Database).

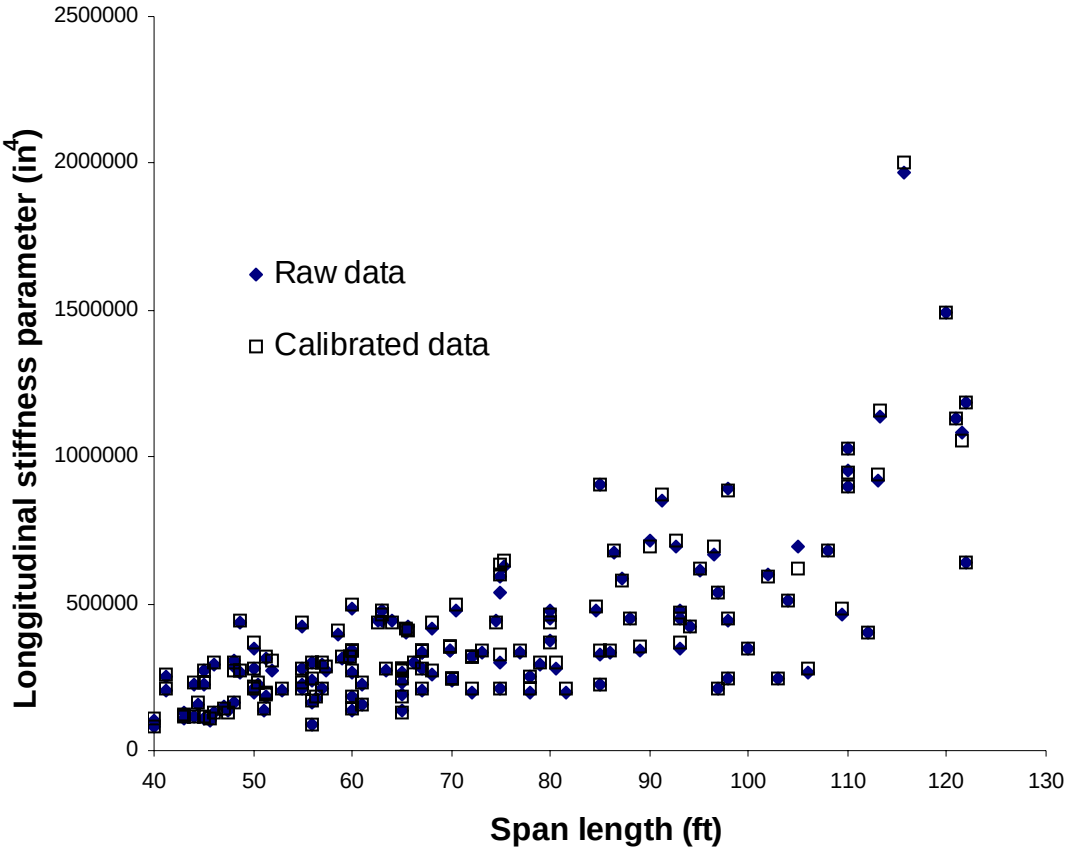


Figure 5.2 Calibrated Longitudinal Stiffness Parameter.

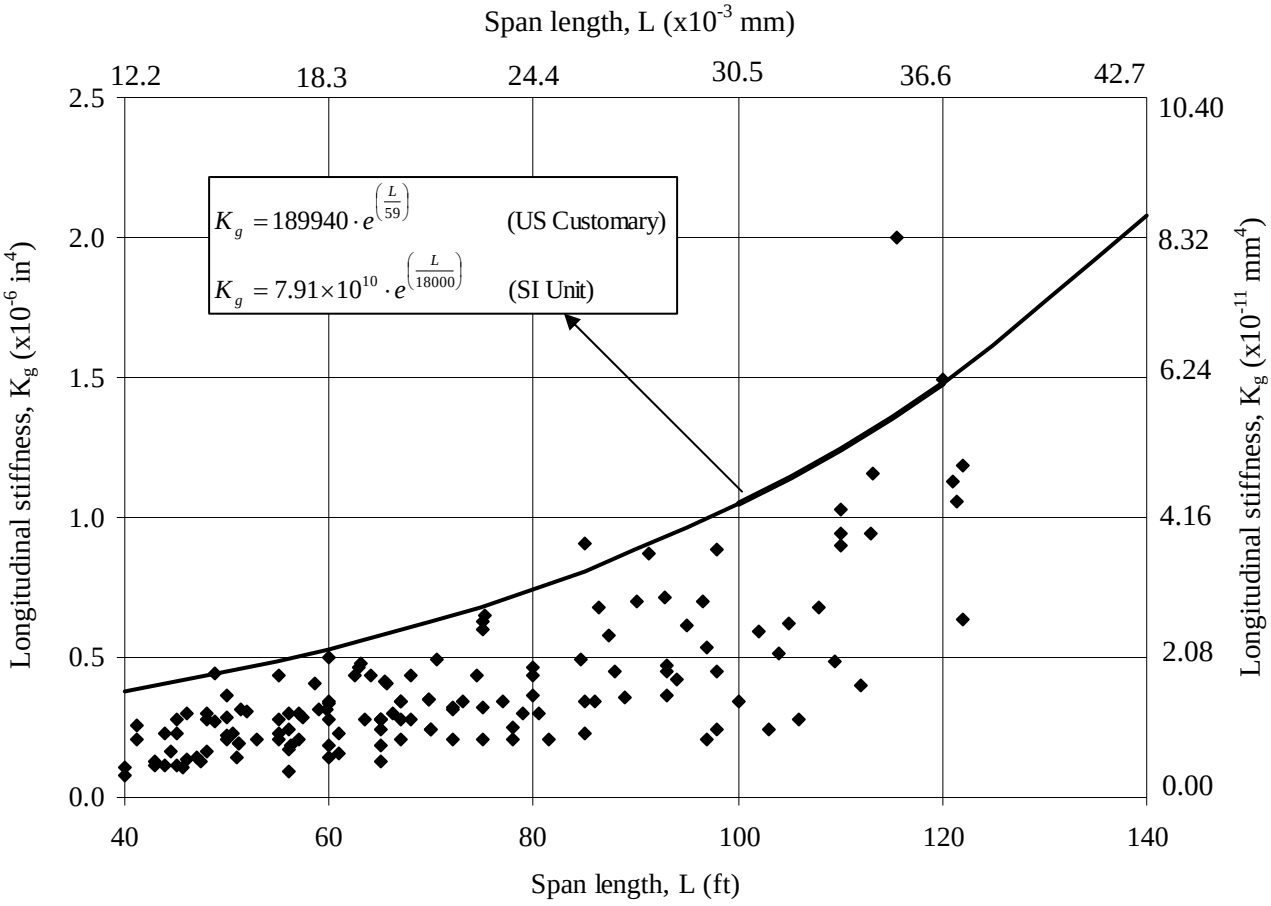


Figure 5.3 Exponential Trend Line.

5.2 Postulation of Simplified Formula

It is worthy to note that the applicable range for each parameter in the new simplified equation is summarized in Table 5.1.

Since the LRFD equation is presumed to be accurate, the postulation of the new LDF equation is constructed based on the current AASHTO-LRFD equation, which is presented again as follows:

$$LDF = 0.15 + \left(\frac{S}{3}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{12Lt_s^3}\right)^{0.1} \quad \text{(US customary units)}$$

$$LDF = 0.15 + \left(\frac{S}{914}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{Lt_s^3}\right)^{0.1} \quad \text{(SI units)} \qquad \text{(From 1.2)}$$

With a slab thickness equal to 8 inches (200 mm) and the relationship between the longitudinal stiffness and the span length from Equation (5.1), the new simplified formula for the wheel load distribution factor of concrete slab on steel girder bridges with two or more design lanes loaded is derived as

$$LDF = 0.15 + 0.73 \cdot \frac{S^{0.8}}{L^{0.3}} \cdot e^{\left(\frac{L}{590}\right)} \quad \text{(US customary units)}$$

$$LDF = 0.15 + 0.042 \cdot \frac{S^{0.8}}{L^{0.3}} \cdot e^{\left(\frac{L}{180{,}000}\right)} \quad \text{(SI units)} \qquad (5.2)$$

where S is the girder spacing (feet, mm) and L is the span length (feet, mm). The advantage of the new simplified equation is that it includes the most influential parameters (S, L, and K_g). The girder spacing and span length parameters are incorporated explicitly, while the longitudinal stiffness is built in implicitly through the relationship to the span length. In this fashion, the iterative procedure in the LDF determination is eliminated.

The base equation of Simplified LDF formula does not take into account the skew correction. The simplified LDF should be reduced by the skew correction factor as identified in Table 5.2 for US customary units and Table 5.3 for SI units, respectively. The skew correction factor for the new Simplified equation is also derived from one in the AASHTO-LRFD specification by using the Kg-L relationship (Equation 5.1).

It is important to note that this new simplified equation should be used within the applicable range as described in Table 5.1. Furthermore, the designer needs to check the final girder selection. The Simplified LDF equation works best if the selected girder produces a K_g less than the value obtained by Equation 5.1; a safe LDF value will certainly be obtained.

Table 5.1 Applicable Range for the New Simplified LDF Equation

Parameters	Girder Spacing: S, ft (mm)	Span Length: L, ft (mm)	Slab Thickness: t_s, in (mm)	Skew Angle (θ, degree)
Applicable Range	4 - 10 (1220 - 3050)	44 – 122 (13400 – 37200)	8 (200)	0 - 45

Table 5.2 Wheel Load Distribution Formulas for Concrete Slab on Steel Girder Bridges and Skew Correction Factor (US Customary Units*)

Specification	Basic LDF Formula	Skew correction factor
AASHTO Standard	$\frac{S}{5.5}$	N/A
AASHTO LRFD	$0.15+\left(\frac{S}{3}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{12Lt_s^3}\right)^{0.1}$	for $\theta \geq 30°$ $1-0.25\left(\frac{K_g}{12Lt_s^3}\right)^{0.25}\left(\frac{S}{L}\right)^{0.5}(\tan\theta)^{1.5}$
Simplified	$0.15+0.73\cdot\frac{S^{0.8}}{L^{0.3}}\cdot e^{\left(\frac{L}{590}\right)}$	for $\theta \geq 30°$ $1-0.59\cdot\frac{S^{0.5}}{L^{0.75}}\cdot(\tan\theta)^{1.5}\cdot e^{\left(\frac{L}{236}\right)}$

* Units of S, L, K_g, and t_s are ft, ft, in^4, and in, respectively.

Table 5.3 Wheel Load Distribution Formulas for Concrete Slab on Steel Girder Bridges and Skew Correction Factor (SI Units**)

Specification	Basic LDF Formula	Skew correction factor
AASHTO Standard	$\frac{S}{1676}$	N/A
AASHTO LRFD	$0.15+\left(\frac{S}{914}\right)^{0.6}\left(\frac{S}{L}\right)^{0.2}\left(\frac{K_g}{Lt_s^3}\right)^{0.1}$	for $\theta \geq 30°$ $1-0.25\left(\frac{K_g}{Lt_s^3}\right)^{0.25}\left(\frac{S}{L}\right)^{0.5}(\tan\theta)^{1.5}$
Simplified	$0.15+0.042\cdot\frac{S^{0.8}}{L^{0.3}}\cdot e^{\left(\frac{L}{180,000}\right)}$	for $\theta \geq 30°$ $1-2.5\cdot\frac{S^{0.5}}{L^{0.75}}\cdot(\tan\theta)^{1.5}\cdot e^{\left(\frac{L}{72,000}\right)}$

** Units of S, L, K_g, and t_s are mm, mm, mm^4, and mm, respectively.

5.3 Comparison of LDFs

The load distribution factors obtained using the proposed simplified equation are compared with those obtained using AASHTO LRFD and AASHTO Standard. They are also compared with those resulting from the finite element analysis (FEA). The definitions of the terms used in the LDF comparison are given in Table 5.2 and Table 5.3.

The comparisons of three LDF equations are shown in Figure 5.4 and Figure 5.5. The ratio of Simplified LDF to LRFD LDF for different span lengths and girder spacing is shown in Figure 5.4. Each data point represents a real bridge from the NCHRP 12-26 and Indiana databases. Simplified LDFs are greater than LRFD LDFs except for two data points which are located under the dotted line. These are data points for the two bridges beyond the upper limit defined in equation 5.1. Although at first glance these seem to be unconservative LDF values, further studies using the developed FE model reveal that the simplified LDFs are greater than those obtained using FEA results. Thus, it can be ascertained that the new equation always produces conservative LDFs within the applicable range of bridges.

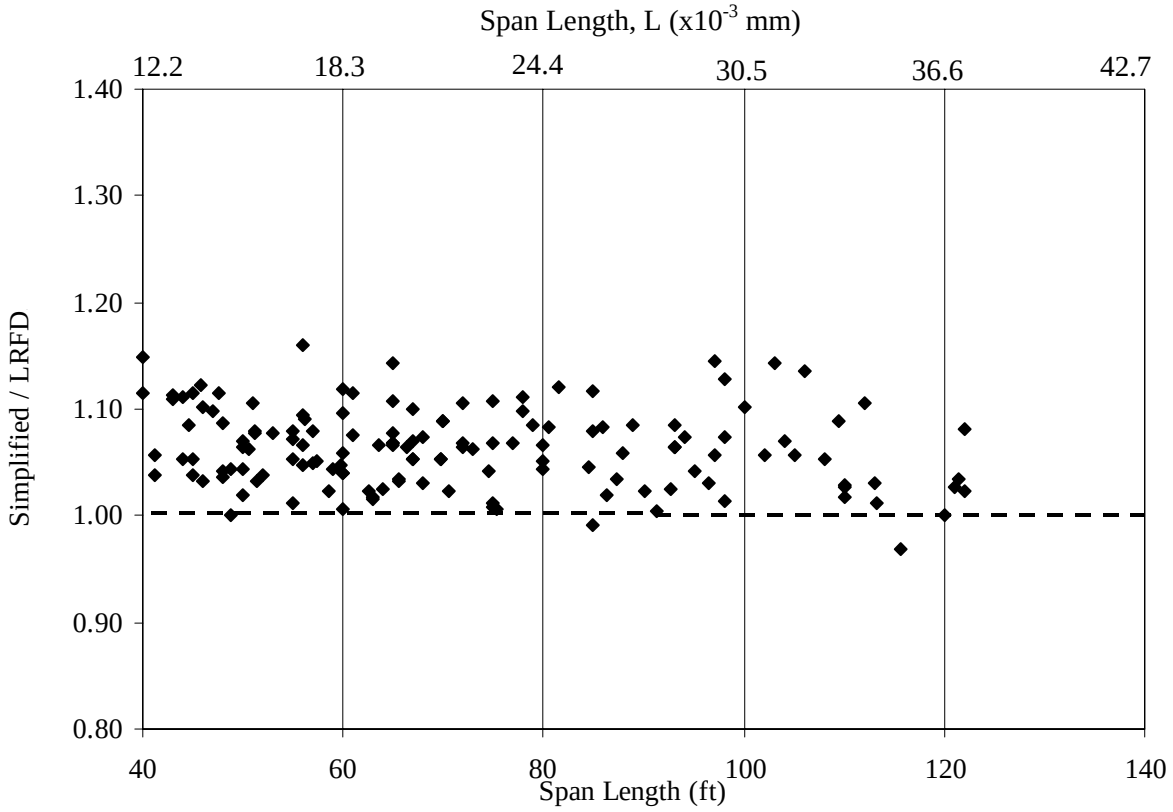
The ratio between Simplified LDF to Standard LDF is shown in Figure 5.5. Simplified LDFs are typically greater than Standard LDFs for small span length and small girder spacing, and lesser for large span length and large girder spacing. A similar trend is seen when LRFD LDF is compared to Standard LDF. In other words, the Simplified LDF equation has similar characteristics as the LRFD LDF equation.

In Figure 5.6 and Figure 5.7, the Simplified LDF and the LRFD LDF are compared to the FEM LDF for positive moment and negative moment, respectively. Each of the data points is an LDF ratio for one of the Indiana representative bridges. As mentioned earlier, the FEM LDF is considered to be the “exact” LDF. It is observed that LRFD LDFs are generally conservative, but for some bridges unconservative results are obtained. Simplified LDFs, on the other hand, are always conservative.

The Simplified LDF may be greater than the LRFD LDF by up to 16 percent depending on the bridge geometry. However, the Simplified LDF is always conservative, unlike the LRFD LDF. The Simplified and LRFD LDF are greater than the Standard

LDF when span length and girder spacing are relatively small. In other words, the Standard LDF is not conservative in those cases. Therefore, the simplified formula, which is similar to the LRFD formula in its ability to represent the load distribution. Finally, the new simplified equation is simple, requires no iteration, and produces LDF values that are at least as conservative as the ones obtained by the LRFD equation.

The limits of validity outside the applicable range of the Simplified LDF are further investigated. Very short and long span bridges are selected and analyzed. As can be seen in Figure 5.8, the Simplified LDF is always larger than the FEM LDF. Thus, the Simplifed formula can be applicable for the bridges whose span length ranges out of applicable range.



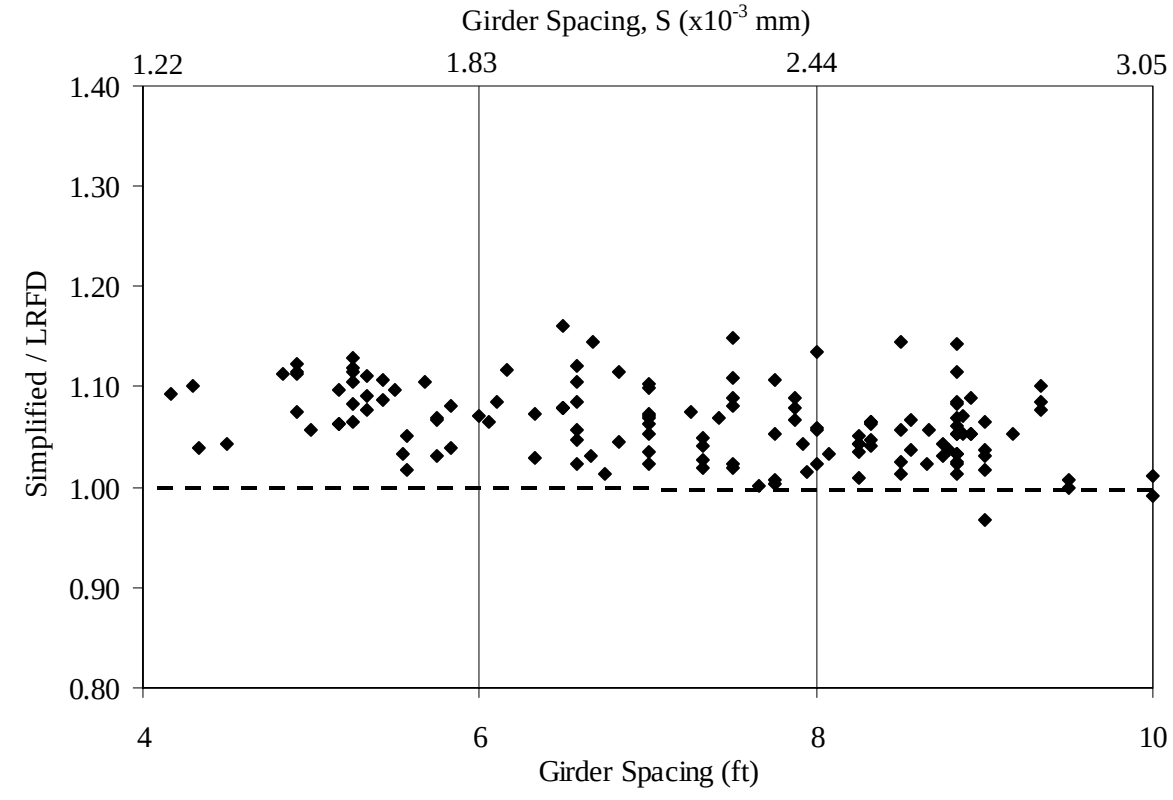


Figure 5.4 LDF Comparisons Between Simplified Equation and LRFD Equation.

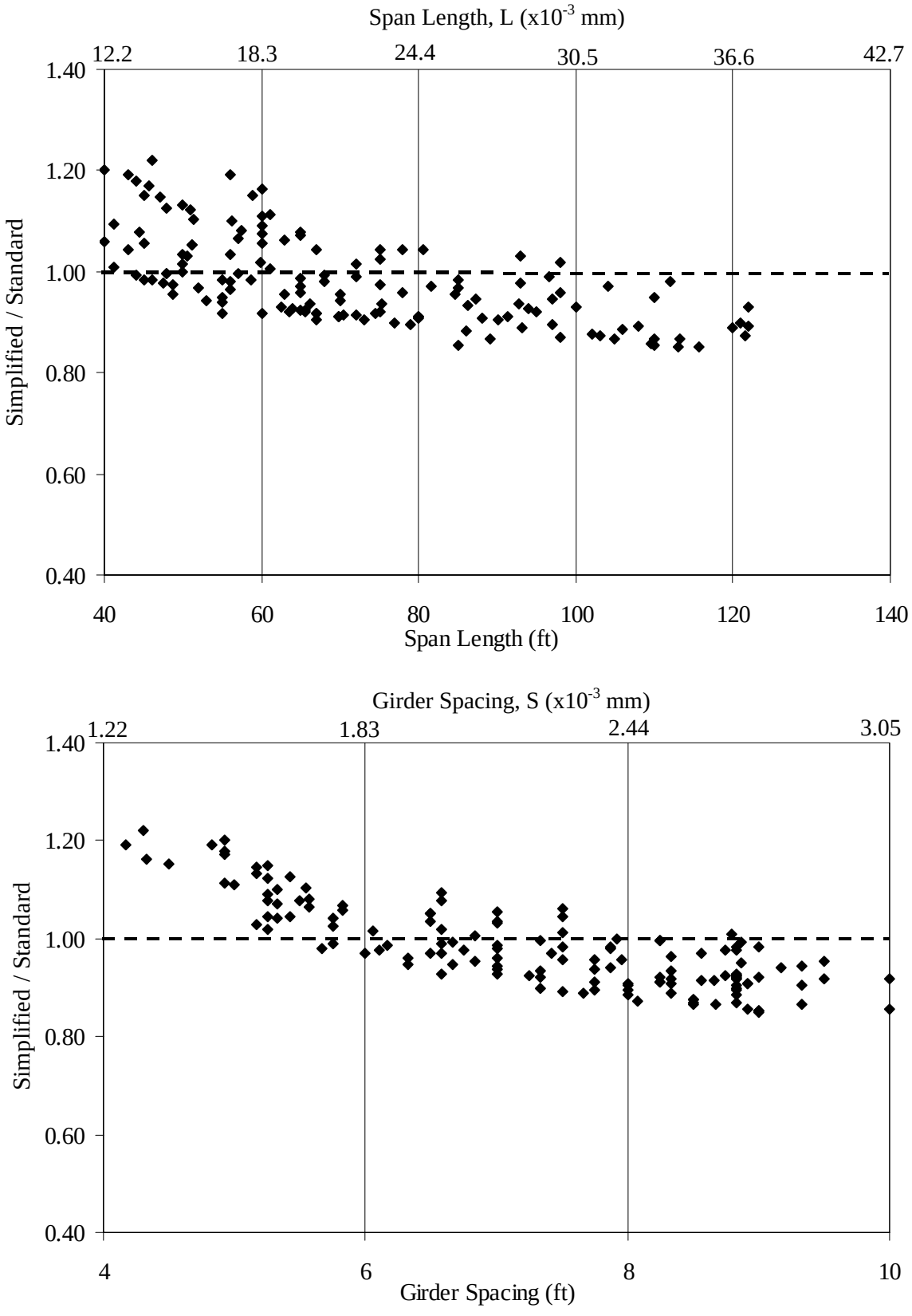


Figure 5.5 LDF Comparisons Between Simplified Equation and Standard Equation.

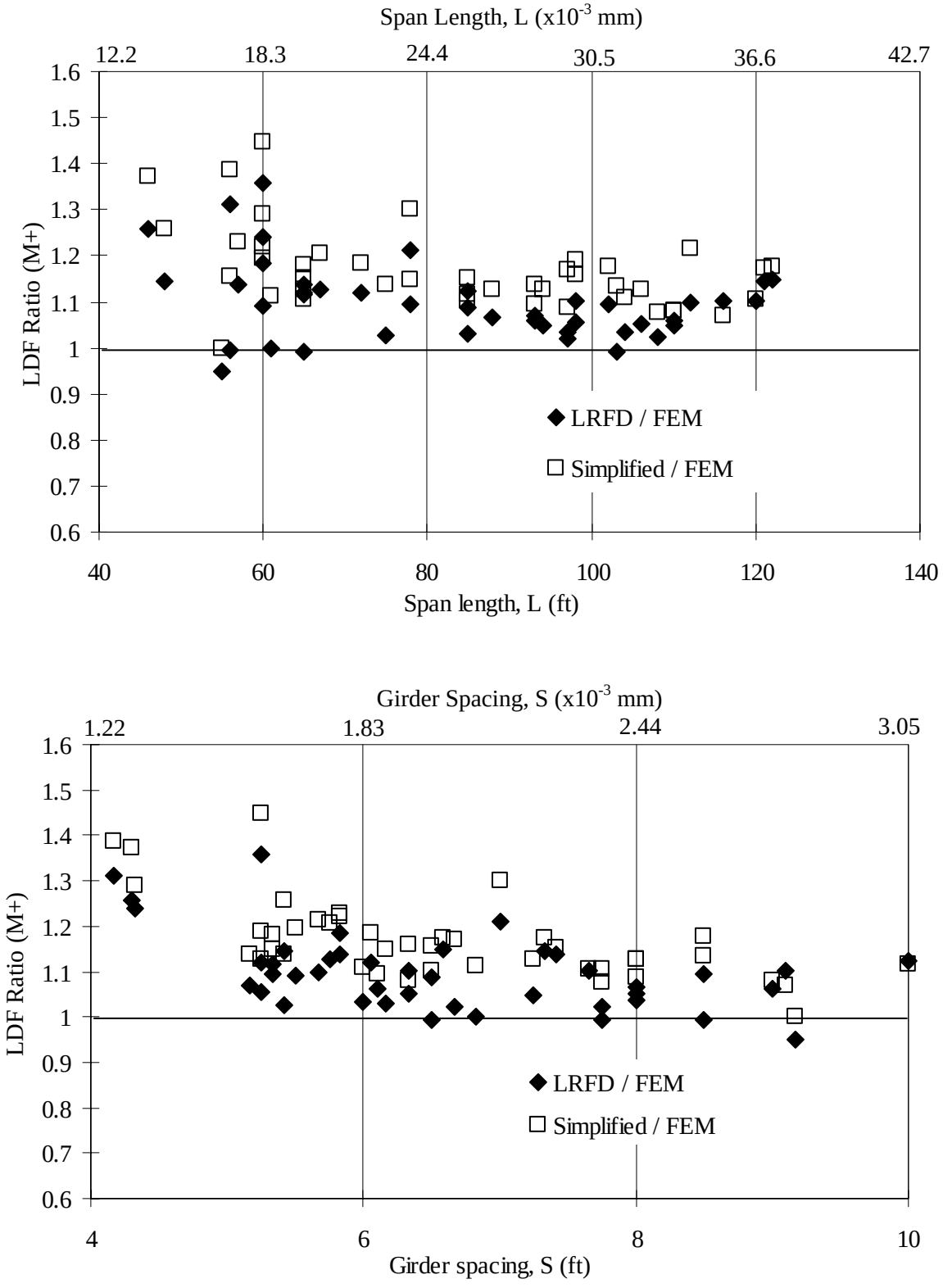


Figure 5.6 Positive Moment LDF Comparisons.

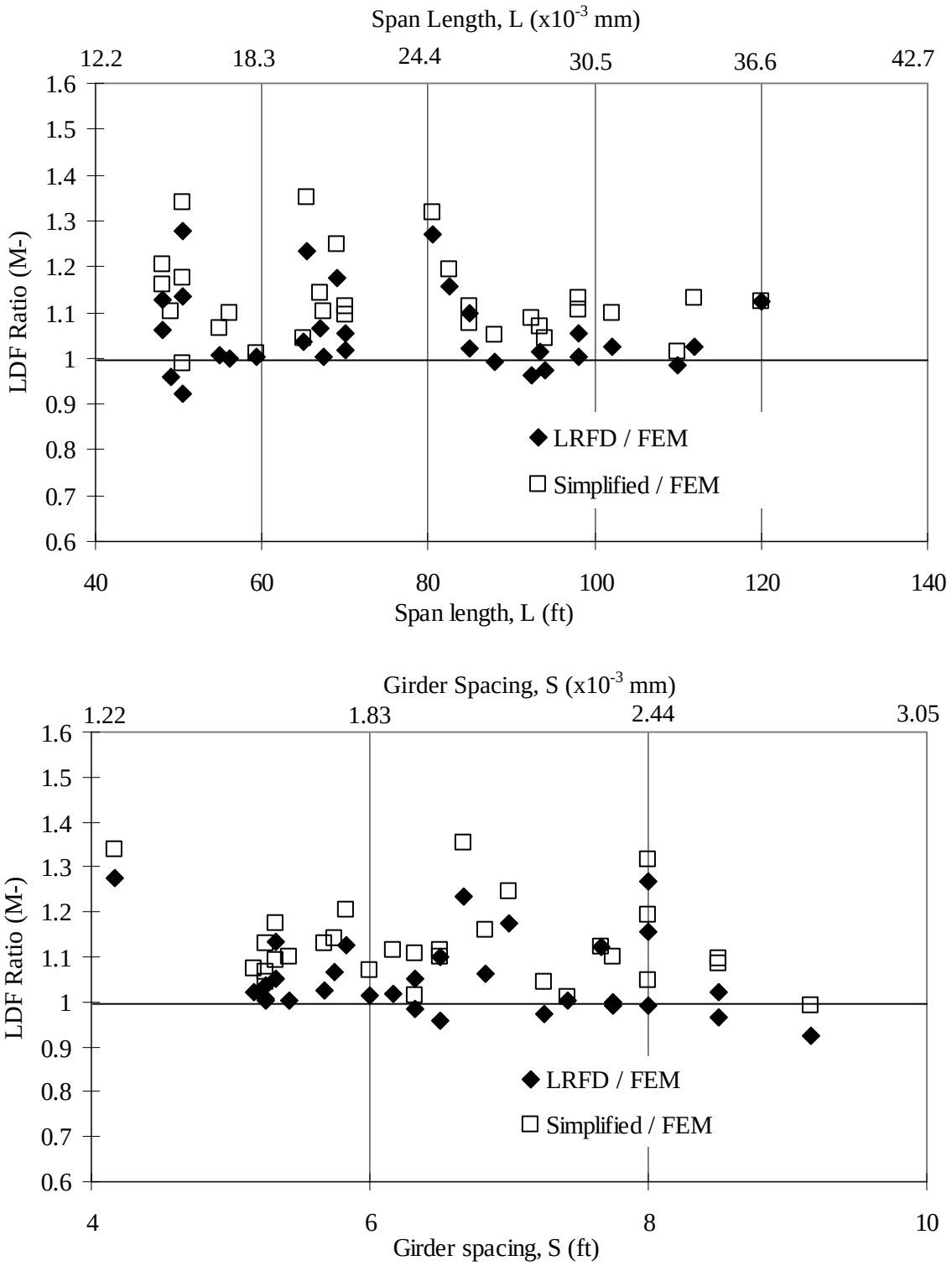
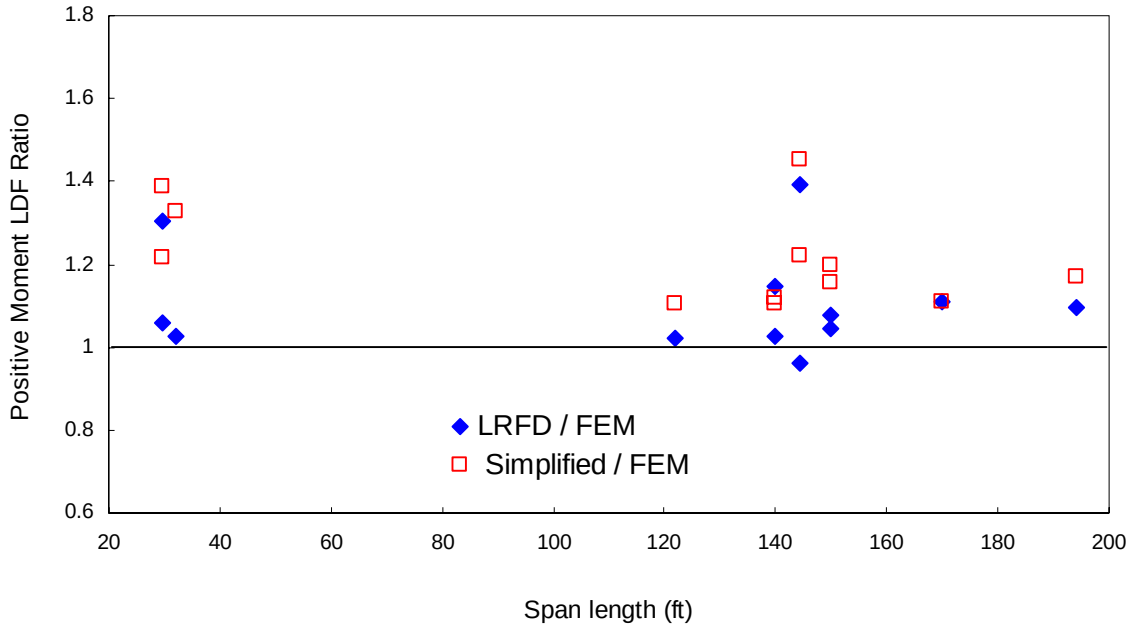


Figure 5.7 Negative Moment LDF Comparisons.



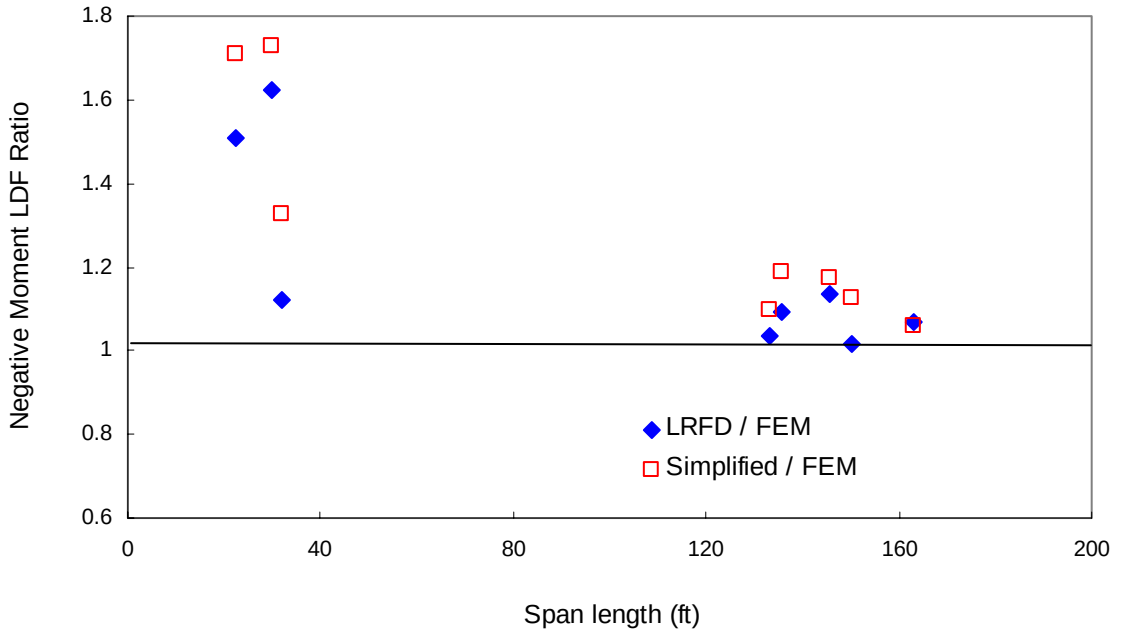


Figure 5.8 LDF Comparisons for Out of Range Bridges

CHAPTER 6. PRESTRESSED CONCRETE BRIDGES

6.1 Introduction

One of the popular types in bridge superstructure is the prestressed concrete (PC) girder bridge. About 17 % of Indiana highway bridges are PC girder bridges according to the NBI database. The applicability of the simplified load distribution factor (LDF) equation to PC girder bridges is examined in this chapter.

Current AASHTO specifications utilize the same LDF formula for both steel I-girder bridges and PC I-girder bridges. In this study, the simplified LDF equation has been developed based on data from steel girder bridges. This chapter further evaluates the simplified LDF equation with respect to PC girder bridges. First, several finite element modeling techniques are investigated to determine the exact LDF. The FE results are then compared with experimental tests done by other researchers. Next, a total of 17 Indiana PC girder bridges are selected and analyzed using a chosen finite element model. Finally, the applicability of the simplified LDF formula is evaluated through the comparisons with LDF values form finite element analyses and AASHTO-LRFD.

6.2 Finite Element Modeling of PC Girder Bridges

The eccentric beam model is selected for modeling of steel girder bridges due to its accuracy and efficiency as discussed in Chapter 3. In the finite element modeling of prestressed concrete (PC) girder bridges, eccentric truss elements are added in the framework of the eccentric beam model to idealize the prestressing tendons. Since the tendon profile varies along the length of the girder, the eccentricity between the centroid of the slab and the prestressing tendon varies from node to node as shown in Figure 6.1.

The eccentricity of the tendon is specified using the multi-point-constraint (MPC) feature available in ABAQUS.

In addition to modeling the profile of the prestressing tendons, the prestressing force must also be represented appropriately. Prestressing forces are specified through the initial stress of each truss element representing prestressing tendons. The magnitude of the initial stress can be determined by dividing the initial tension per strand by the area of each tendon. After the girders, slab, and prestressing tendons are all effective, live load cases are applied to the final configuration of PC girder bridges. This is referred to as a full load analysis (Hays et al. 1994).

In this study, however, only the effect of live loads is of interest since the load distribution factor (LDF) is due to live loads such as AASHTO HS20 trucks or lane loading. Three different modeling techniques for PC girder bridges are investigated to ensure their applicability for modeling live load effects. The first model (Model A) is the eccentric beam model. All prestressing elements, including prestressing tendons and prestressing force, are excluded from the finite element model as shown in Figure 6.2. The second model (Model B) is the same as Model A, but prestressing strands are modeled by eccentric truss elements and the prestressing force is not considered as shown in Figure 6.3. This model is included to examine the effect of tendon elements on live load distribution. The last model (Model C) is the most accurate and most rigorous model. First, the bridge is analyzed with the full load analysis (Figure 6.4 (a)). The LDF from the full load analysis is then subtracted by the LDF from the full load analysis without live loads (Figure 6.4 (b)). In this way, only live load effect is considered in the analysis. It should be noted that all FE models discussed above do not include the structure's self weight.

The LDFs obtained using the three modeling techniques discussed above are compared for two Indiana PC bridges. The first one is a single span bridge located on Indiana State Route 257 over Hurricane Creek in Davies County with a 72 ft span length. The bridge deck is supported by seven AASHTO Type III PC girders spaced at 6.5 ft. The other bridge is located on Indiana State Route 32 over White River in Randolph

County. It is a three span bridge with 55-65-55 ft span lengths. The bridge deck is supported by eight AASHTO Type II PC girders spaced at 6.0 ft.

Each bridge is modeled with the three developed FE models (Model A, Model B, and Model C). Figure 6.5 shows the load distribution factor (LDF) from the three finite element models, AASHTO-LRFD code, and Simplified equation. The LDF calculated by three different finite element models are essentially the same. The maximum difference is less than 0.5 %. However, it is clear that the LDF values from AASHTO-LRFD and Simplified equation are more conservative than the predicted LDF values. This indicates that the eccentric beam model (Model A) is just as accurate as the other models while being the simplest model. The eccentric beam model is thus selected for PC bridge modeling. It should be noted that the LRFD LDF equation is also derived based on the eccentric beam model (Zokaie 1991a & 1991b).

Table 6.1 Selected Indiana Bridges

Location	Indiana SR 257	Indiana SR 32
AASHTO Section Type	Type III	Type II
Skew (degree)	0	0
Overhang	2.125	2.25
Slab thickness (in)	8.0	6.25
Construction Year	1992	1981
Identification Number	NBI 18317	NBI 270

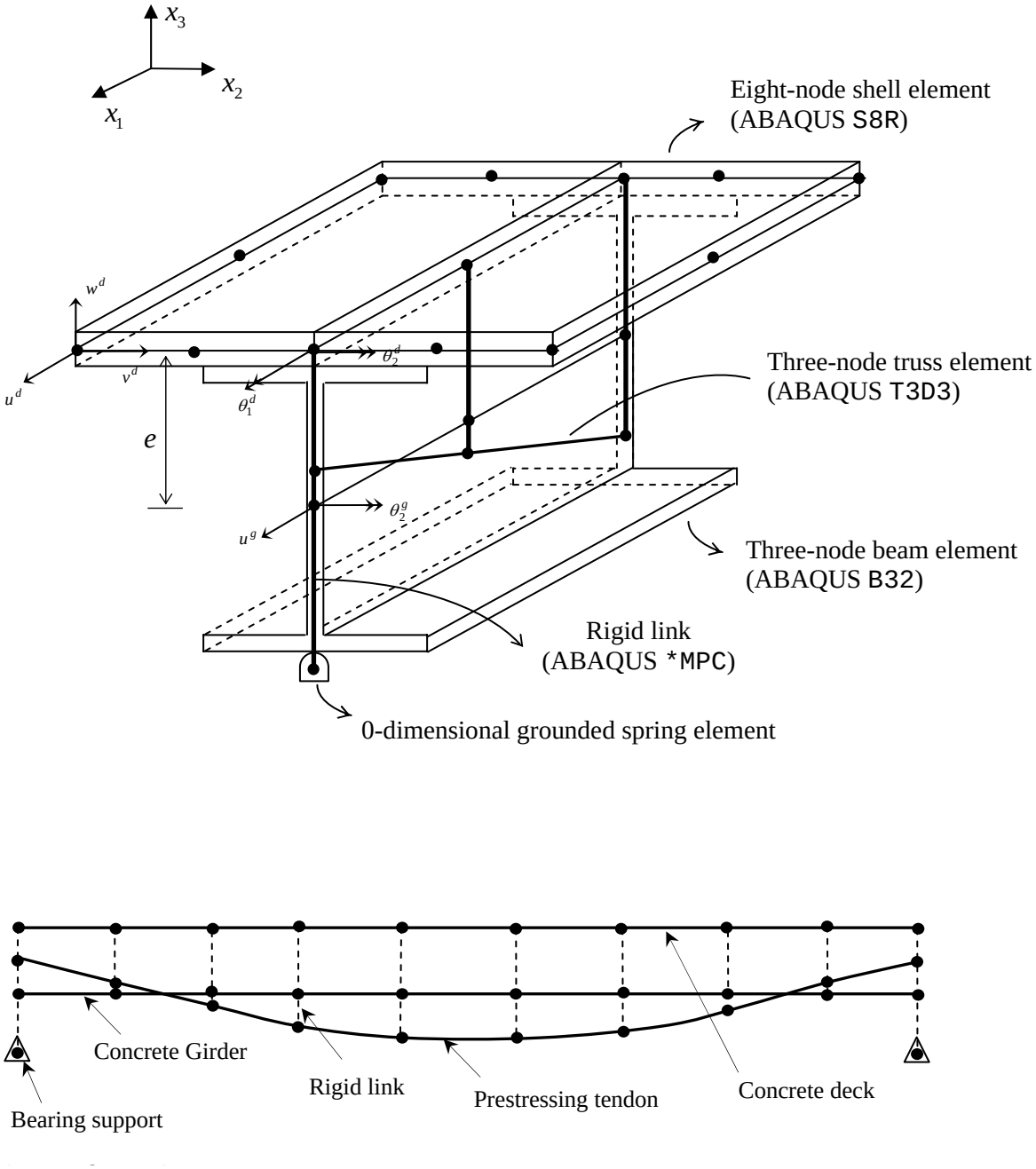


Figure 6.1 Eccentric Beam Model Including Prestressing Tendon

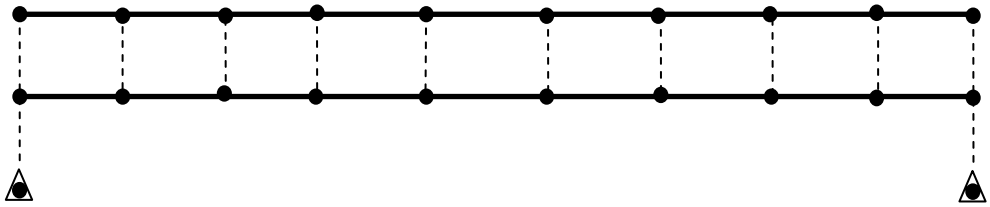


Figure 6.2 PC Bridge Model A

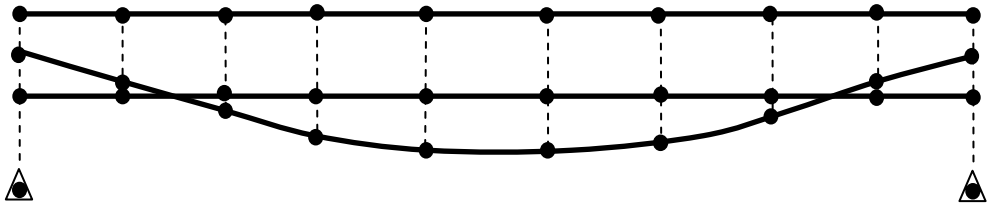
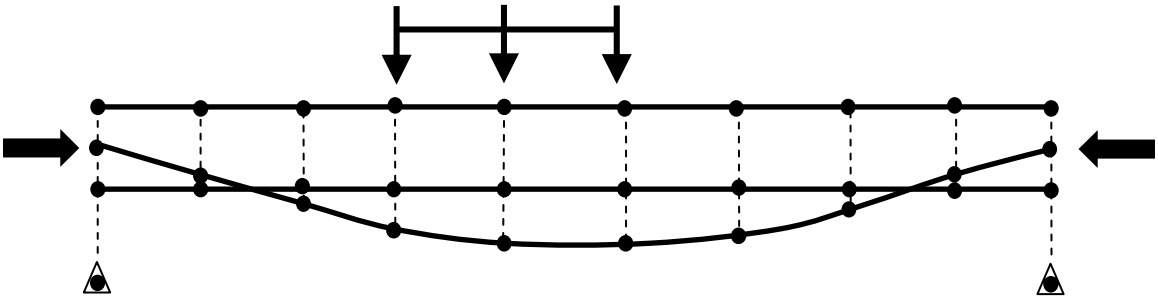
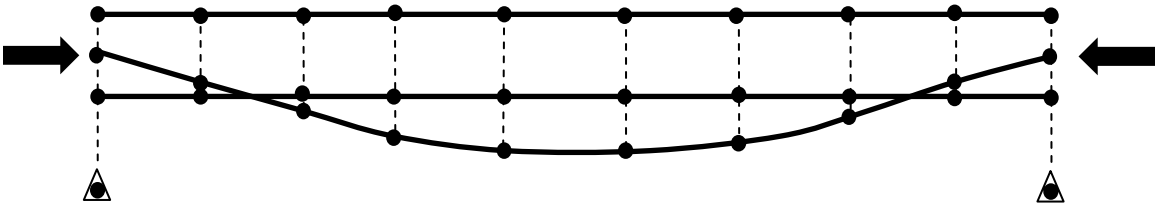


Figure 6.3 PC Bridge Model B



(a) Full Load Analysis



(b) Full Load Analysis Without Live Loads

Figure 6.4 PC Bridge Model C

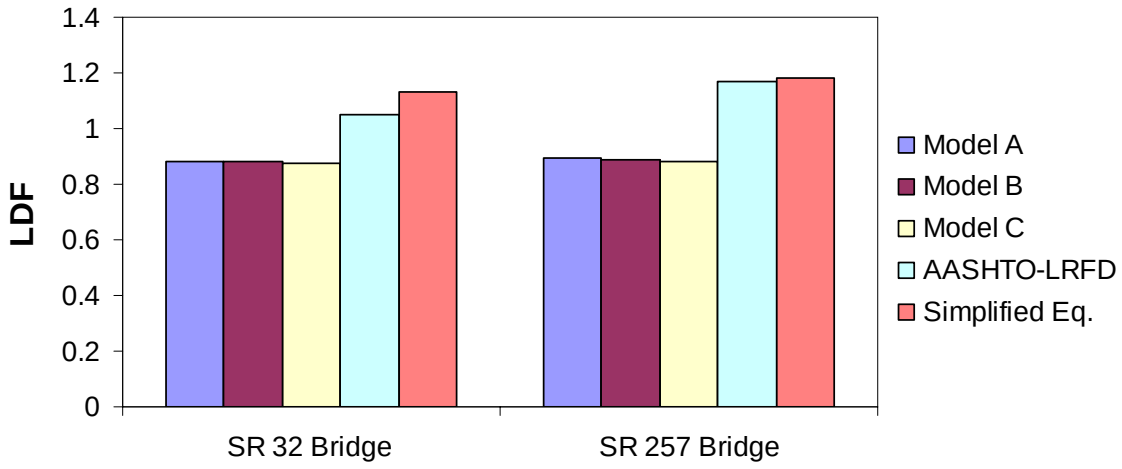


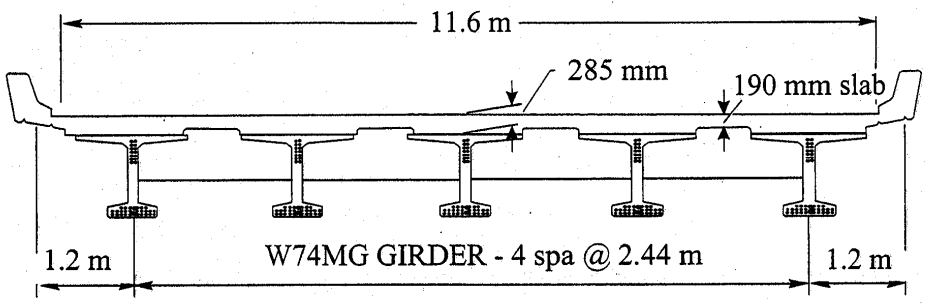
Figure 6.5 Comparisons of Live Load Distribution Factor

6.3 Washington Field Test

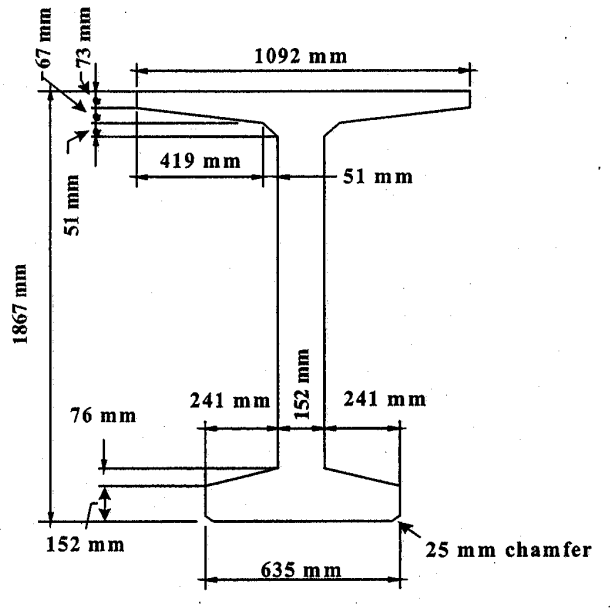
The finite element model developed for PC girder bridges is verified with the results of the field test conducted by University of Washington (Barr et al. 2001). This bridge is three span continuous with span lengths of 80 ft, 137 ft, and 80 ft. The cross sectional dimensions of PC girders are shown in Figure 6.6. The load for field testing is a tractor and semi-trailer unit approximating the ASSHTO HS-20 design loading.

The bridge deck is modeled using eight-node shell elements, and the girders are idealized using three-node beam elements. The deck thickness above the girder is 11.25 in and 7.5 in elsewhere. In the FE modeling of deck slabs, the thickness of the shell elements is assumed to be 9.125 in as an average value. Diaphragms are not modeled for this study. For the future comparisons, the finite element analysis based on the developed eccentric beam model is denoted as “Simplified FEA”.

The FE model used for this study is compared with experimental results and predicted results using the developed FE model by University of Washington (denoted “Washington FEA”) as shown in Figure 6.7 and Figure 6.8. Comparisons between calculated and measured moments at mid-span due to placement of a truck at mid-span of the exterior girder are made in Figure 6.7. It is observed that the moment from the Simplified FEA is generally larger than the results from Washington FEA and experiments. Washington FEA includes intermediate diaphragms and end diaphragms, which help the distribution of moment. It should be noted that moments from Simplified FEA are always larger than measured moments, while Washington FEA sometimes underestimates girder moments. Similarly, Figure 6.8 presents mid-span moments for a truck located at mid-span of the first interior girder. In general, good agreement between measured and calculated values is observed. It is concluded that the Simplified FEA based on the eccentric beam model is as accurate as the detailed Washington FEA and always produces conservative results.



(a) Bridge Cross Section



(b) W74MG girder dimension

Figure 6.6 Washington Bridge Girder (Barr et al. 2001)

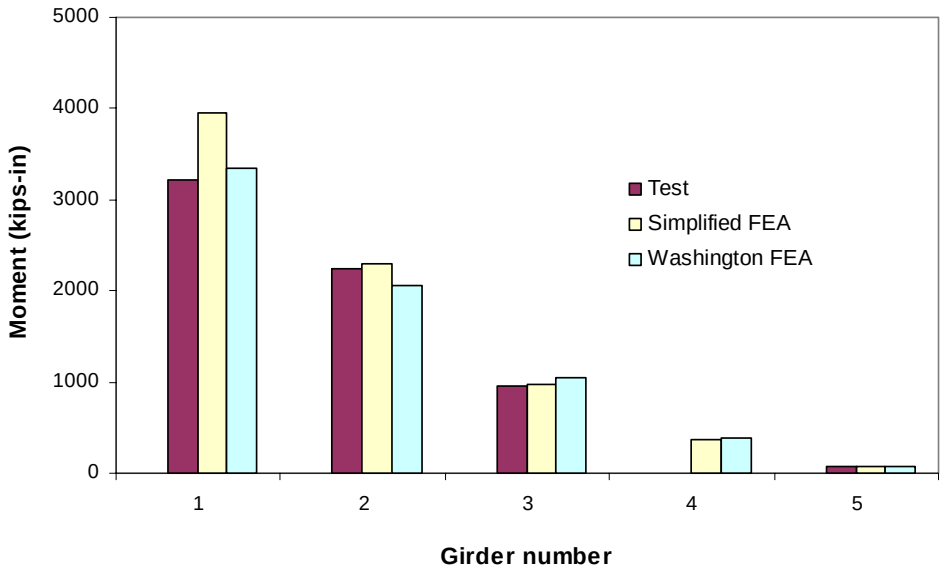


Figure 6.7 Mid-span Moment Due to Truck Load Located on the Exterior Girder

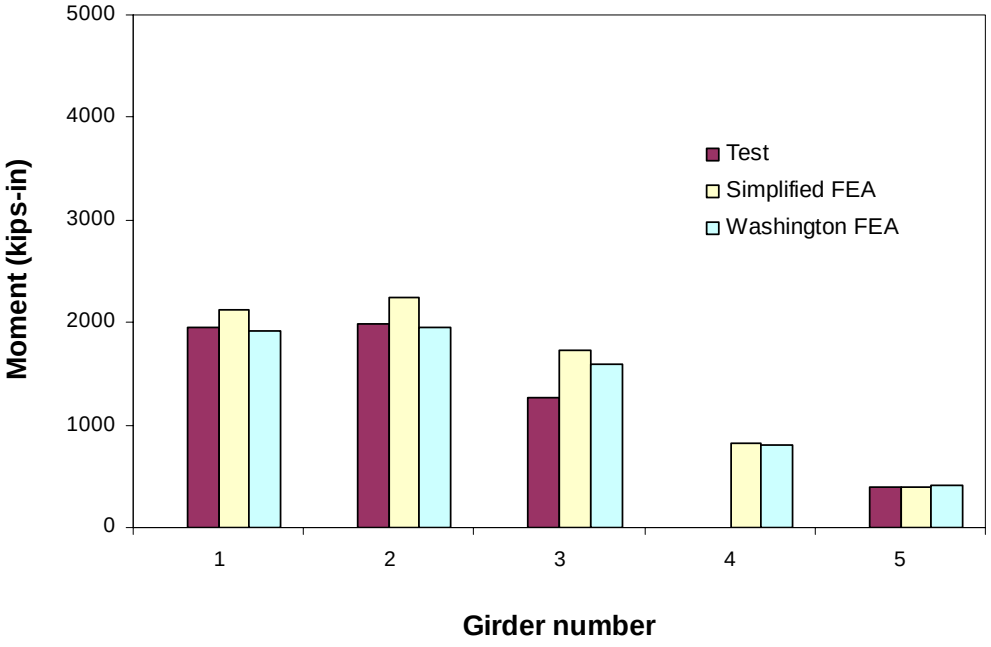


Figure 6.8 Mid-span Moment Due to Truck Load Located on the First Interior Girder

6.4 LDF Comparisons of PC Girder Bridges

From the data analysis of the NBI database it has been determined that the typical range of bridge span length for Indiana PC girder bridges is 45 to 80 feet. This range is thus assigned as the applicable range for the span length. The typical number of spans ranges from one span to three spans and the skew angle along the supports spans from 0 to 45 degrees.

From the applicable range, a total of 17 Indiana bridges are deliberately selected to cover the range. Theses bridges are referred to as "Indiana Representative PC girder Bridges" and are used later in the verification of the postulated formula. Table 6.2 shows the list of the Indiana Representative PC girder Bridges. The longest PC girder bridge (115 ft) among the available Indiana database is included to examine the range of validity of the simplified load distribution equation. Girder spacing of the representative bridges ranges from 6 ft to 7.5 ft.

In Figure 6.9, the longitudinal stiffness parameters for the Indiana representative bridges are plotted against the span length. The general trend of the relationship is that K_g increases as L increases. It should be noted that AAHSTO Type II, III, and IV are common in Indiana PC girder bridges. Type II bridges are used for span lengths from 45 ft to 70 ft, and Type III bridges are common in the range of 65 ft to 80 ft. Type IV bridges are used when the span length is larger than 80 ft. However, the data are scattered in relatively wide range. In other words, different girder types are used for relatively similar span lengths. One of the reasons for this may be the bridge engineer's preference on the selection of girder spacing.

In Figure 6.10 and Figure 6.11, the Simplified LDF and the LRFD LDF are compared to the FEM LDF for positive moment and negative moment, respectively. Each of the data points is an LDF ratio for one of the Indiana representative bridges. As mentioned earlier, the FEM LDF is considered to be the "exact" LDF. It is observed that the Simplified LDFs and LRFD LDFs are always conservative. Some simplified LDFs, however, are less than LRFD LDFs as shown in Figure 6.10 and Figure 6.11. These bridges do not fall below the exponential trend line used to derive the Simplified LDF,

which is an upper bound relationship between K_g and L for steel girder bridges. Even though Simplified LDFs do not guarantee the larger LDFs compared to LRFD LDFs, it is obvious from the results that simplified LDFs are always conservative with respect to "exact" LDFs (at least 10 % conservative) for the bridges analyzed.

Table 6.2 Representative Indiana Prestressed Concrete Girder Bridges

No	NBI Structure Number	County	Location	Year built (rebuilt)	Skew [°]	Max. length [ft]	Girder Spacing [ft]	AASHTO Girder Type
1	270	Randolph	SR 32	1981	0	65	6	II
2	530	Lake	US41	1995	0	54	7.25	II
3	590	Porter	SR 149	1994	20	115	6.75	IV
4	610	Porter	US 30	1988	30	49	7	II
5	1342	De Kalb	SR205	1990	0	87	6.33	IV
6	1690	La Porte	US 35	1996	35	78	8.5	IV
7	10400	Fountain	US 41	1982	16	71	6.5	III
8	11110	Wayne	SR 38	1984	20	76	7	III
9	12720	Hamilton	US31	1995	15	65	7	III
10	18253	Knox	SR 550	1999	15	55	6.5	II
11	18317	Daviess	SR257	1933 (1999)	0	70	6.5	III
12	19670	Lawrence	SR 58	1996	0	60	6.25	II
13	22850	Gibson	SR 65	1978	15	79	7	IV
14	23700	Knox	SR 550	1983	0	82	7	IV
15	24380	Gibson	US 41	1999	0	67	7.4	III
16	31100	Ohio	SR56	1967 (1999)	35	50	7.5	III
17	33020	Steuben	I-90	1986	0	48	7	II

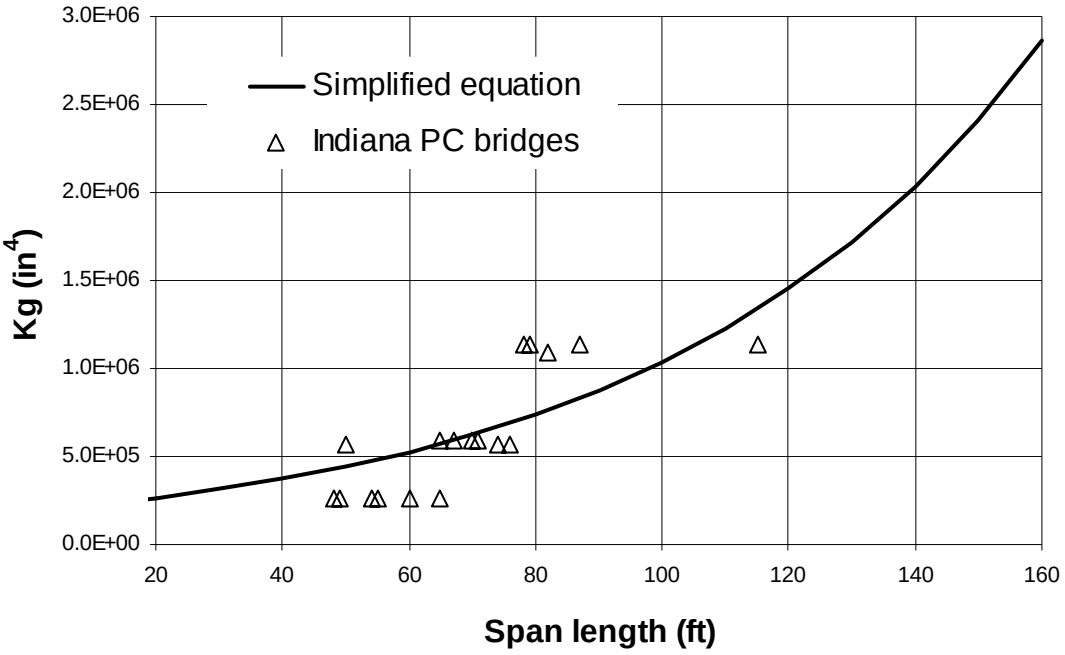
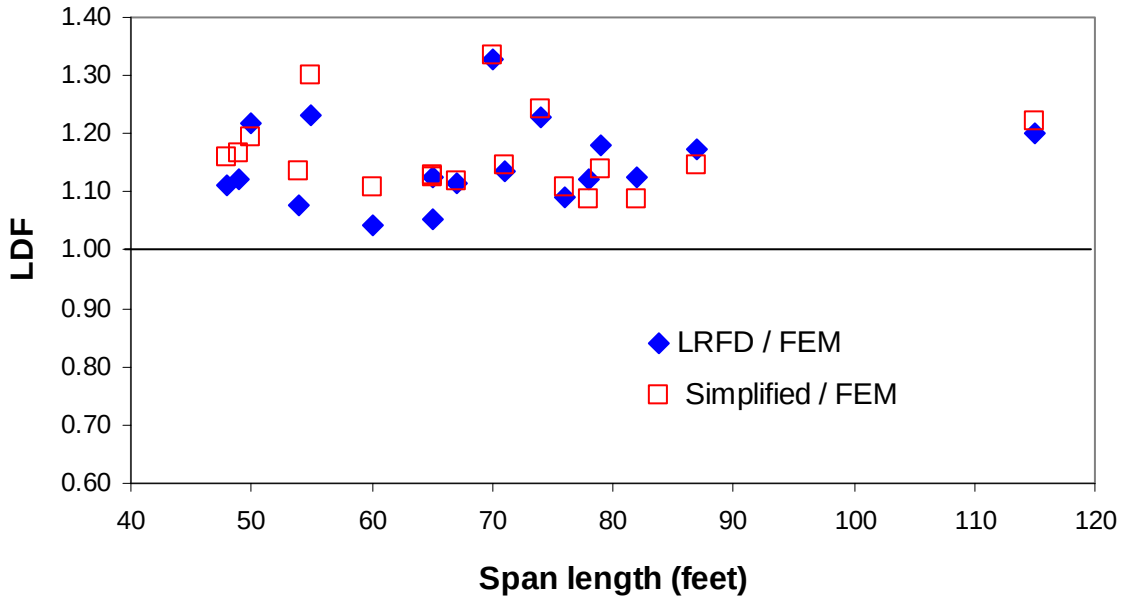


Figure 6.9 Scattergram of Longitudinal Stiffness Parameter and Span Length (Indiana PC girder bridges)



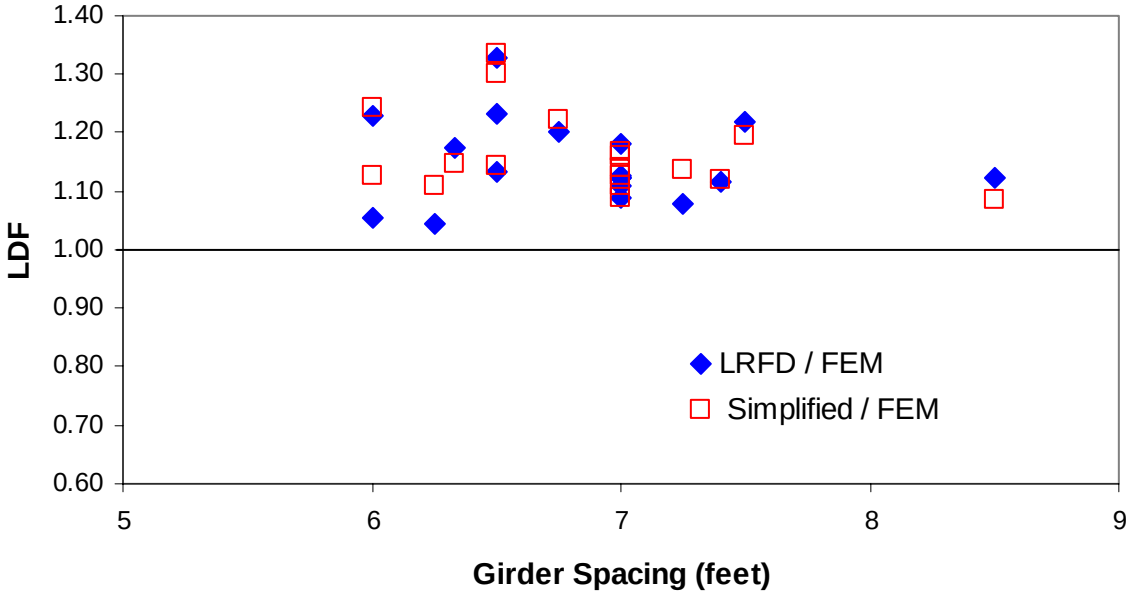
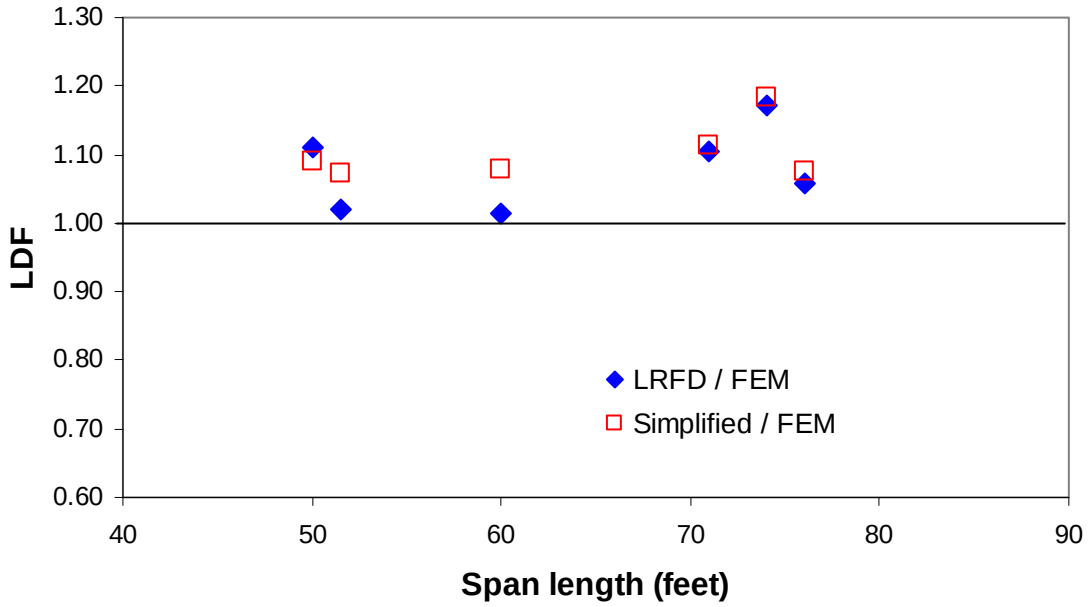


Figure 6.10 Positive Moment LDF Comparisons.



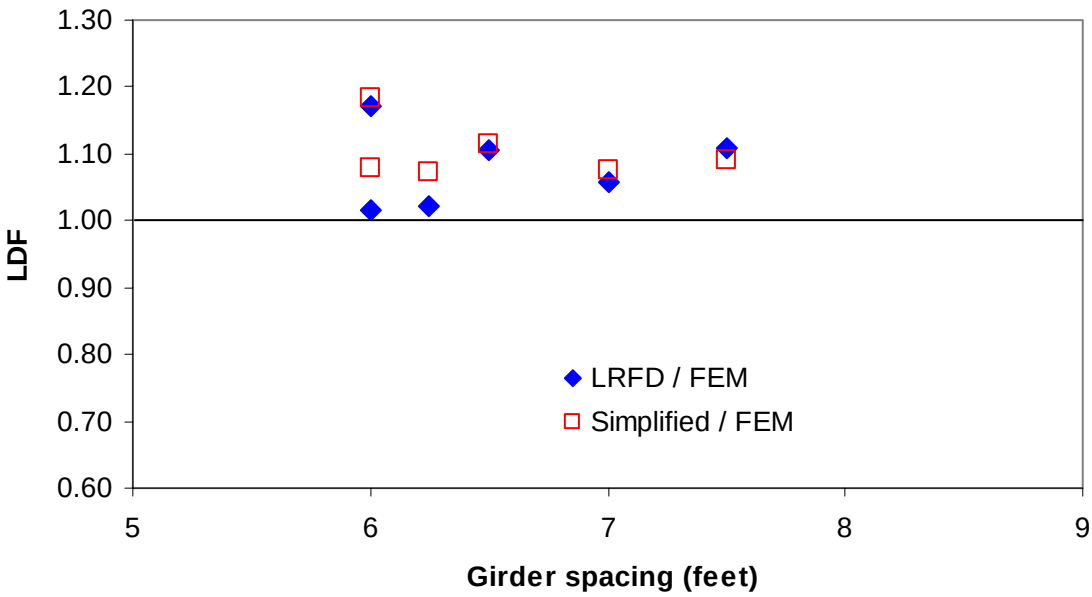


Figure 6.11 Negative Moment LDF Comparisons.

CHAPTER 7. EFFECTS OF SECONDARY ELEMENTS AND DECK CRACKING ON LOAD DISTRIBUTION FACTOR

7.1 Introduction

A new simplified equation based on the AASHTO LRFD formula has been developed using sophisticated finite element analyses of the 43 steel girder and 17 prestressed concrete girder bridges identified in the previous chapters. Even though the Simplified equation is considered to represent well the actual behavior of bridges, the FE model used in developing the Simplified LDF equation does not include some important features of bridges, which may affect lateral load distribution.

First, despite the presence of the secondary elements such as cross bracing, diaphragms, and parapets in bridges, these elements are not considered in the development of the AASHTO LRFD LDF equation. Previous parametric studies (Eamon and Nowak 2004; Eamon and Nowak 2002; Mabsout et al. 1997) have shown that consideration of secondary elements, which are typically present in steel girder bridges, has a significant effect on the lateral load distribution. Consequently, the AASHTO LRFD equation provides overly conservative results. Secondly, previous research (Frosch et al. 2003; French et al. 1999) revealed a widespread presence of pre-existing cracks in concrete bridge decks. These cracks are usually formed even before the bridge is open to traffic. Direction of cracking is typically transverse with respect to traffic direction, but longitudinal cracking has also been observed. Even though early-age deck cracking is a well-known phenomenon, the effect of deck cracking on the live load distribution has not yet been assessed.

The objectives of this study are (1) to investigate the influence of secondary elements on the lateral load distribution of typical steel girder bridges; and (2) to examine

the effects of deck cracking on the load distribution mechanism through nonlinear analyses. In order to examine these effects, a reliable three-dimensional finite element model including secondary elements and a concrete cracking constitutive model is developed. Then, nine Indiana bridges are selected and analyzed using the model. The load distribution factors obtained using this model are compared with those obtained using the AASHTO LRFD equation, AASHTO Standard equation, and Simplified equation.

7.2 Research Methodology

A total of nine Indiana steel girder bridges were deliberately selected to investigate the effect of secondary elements on load distribution factor. All the selected bridges are right-angled and service the state highway system. The two types of lateral bracing typically used in Indiana steel girder bridges, diaphragms and cross bracing, were considered. In general, the spacing of the lateral bracing ranges from 6.4 m (21 ft) to 7.5 m (24.5 ft). The Indiana 84 cm (33 in) concrete barrier, or parapet, was used for the analysis of each of the nine bridges. Figure 7.1 shows the dimensions of the common concrete barrier for Indiana. More details of the selected bridges are given in Table 7.1. The load distribution factor of each bridge was calculated using four different FE models. The influence of lateral bracing and parapets was investigated both separately and together, representing the bridge “as is”.

The influence of deck cracking on the load distribution of steel girder bridges was investigated through case studies of actual bridges. Nine Indiana bridges, which have experienced cracking in the deck slab, were identified. In previous research (Frosch et al. 2003), some bridges known to have experienced deck cracks were identified. As part of this study, field investigations were also carried out to determine the existence of cracking in the concrete deck. The list of identified bridges is given in Table 7.2. Nonlinear finite element analyses were performed using a previously developed nonlinear

finite element framework (Chung and Sotelino 2004). Through a number of simulations, typical crack types that have a major effect on load distribution were identified.

Table 7.1 Selected Indiana Bridges (Secondary Elements)

No.	NBI Number	County	Location	S ft (m)	Max. L ft (m)	K_g $in^4 (cm^4)$	Type of bracing	Lateral bracing	Bracing Spacing ft (m)
1	17843	Monroe	SR48	7.58 (2.31)	117 (35.7)	472,333 (19,659,984)	K frame	L4×3½×½ (L102×89×13)	23 (7.0)
2	50460	Hamilton	I-465	8.33 (2.54)	150 (45.7)	741,548 (30,865,558)	K frame	L4×3½×5/16 (L102×89×8)	22 (6.7)
3	16130	Montgomery	US231	9.66 (2.94)	155 (47.2)	2,303,292 (95,870,251)	K frame	L5×5×5/16 (L127×127×8)	21~23 (6.4~7.0)
4	1030	Delaware	US35	8.33 (2.54)	83 (25.3)	220,638 (9,183,647)	Diaphragm	W18×45 (W450×67)	22 (6.7)
5	12290	Morgan	SR37	5.50 (1.68)	84 (25.6)	220,638 (9,183,647)	Diaphragm	W14×43 (W360×64)	20~24 (6.1~7.3)
6	3600	Jasper	SR14	5.25 (1.60)	98 (29.9)	215,395 (8,965,417)	Diaphragm	W18×45 (W450×67)	24.5 (7.5)
7	49240	Porter	I-94	7.25 (2.21)	105 (32.0)	195,857 (8,152,184)	Diaphragm	W18×45 (W450×67)	21 (6.4)
8	37630	Tippecanoe	I-65 pass	6.17 (1.88)	97 (29.6)	289,719 (12,059,015)	Diaphragm	W18×45 (W450×67)	24.5 (7.5)
9	38300	Lake	I-65 pass	6.58 (2.01)	66 (20.1)	241,748 (10,062,311)	Diaphragm	W18×45 (W450×67)	22 (6.7)

Table 7.2 Selected Indiana Bridges (Pre-existing Cracks)

No.	County	Location	S ft (m)	Max. L ft (m)	K_g $in^4 (cm^4)$	Year built	Skew	Cracking
10	Lake	I-65 pass	5.83 (1.78)	65 (19.8)	186,521 (7,763,590)	1965	0	Transverse cracks
11	Lake	I-65 pass	6.58 (2.01)	66 (20.1)	70,688 (2,942,257)	1966	0	Transverse cracks
12	Carroll	IN75	5.33 (1.62)	78 (23.8)	51,359 (2,137,723)	1992	0	Transverse cracks
13	Marion	I-465	5.17 (1.58)	81 (34.7)	61,155 (2,545,463)	1997	0	Transverse cracks
14	Marion	I-465	7.5 (2.29)	82 (25.0)	46,345 (1,929,025)	1999	3	Transverse cracks
15	Knox	IN58	5.08 (1.55)	29 (8.8)	7,888 (328,323)	1996	0	Longitudinal cracks
16	Tippecanoe	IN25	8.0 (2.44)	110 (33.5)	298,772 (12,435,830)	1995	0	Longitudinal cracks
17	Marion	I-65	9.16 (2.79)	124 (37.8)	300,694 (12,515,829)	1996	20	Longitudinal cracks
18	Greene	IN-67	5.75 (1.76)	47 (14.3)	196,954 (8,197,844)	1994	20	Longitudinal cracks

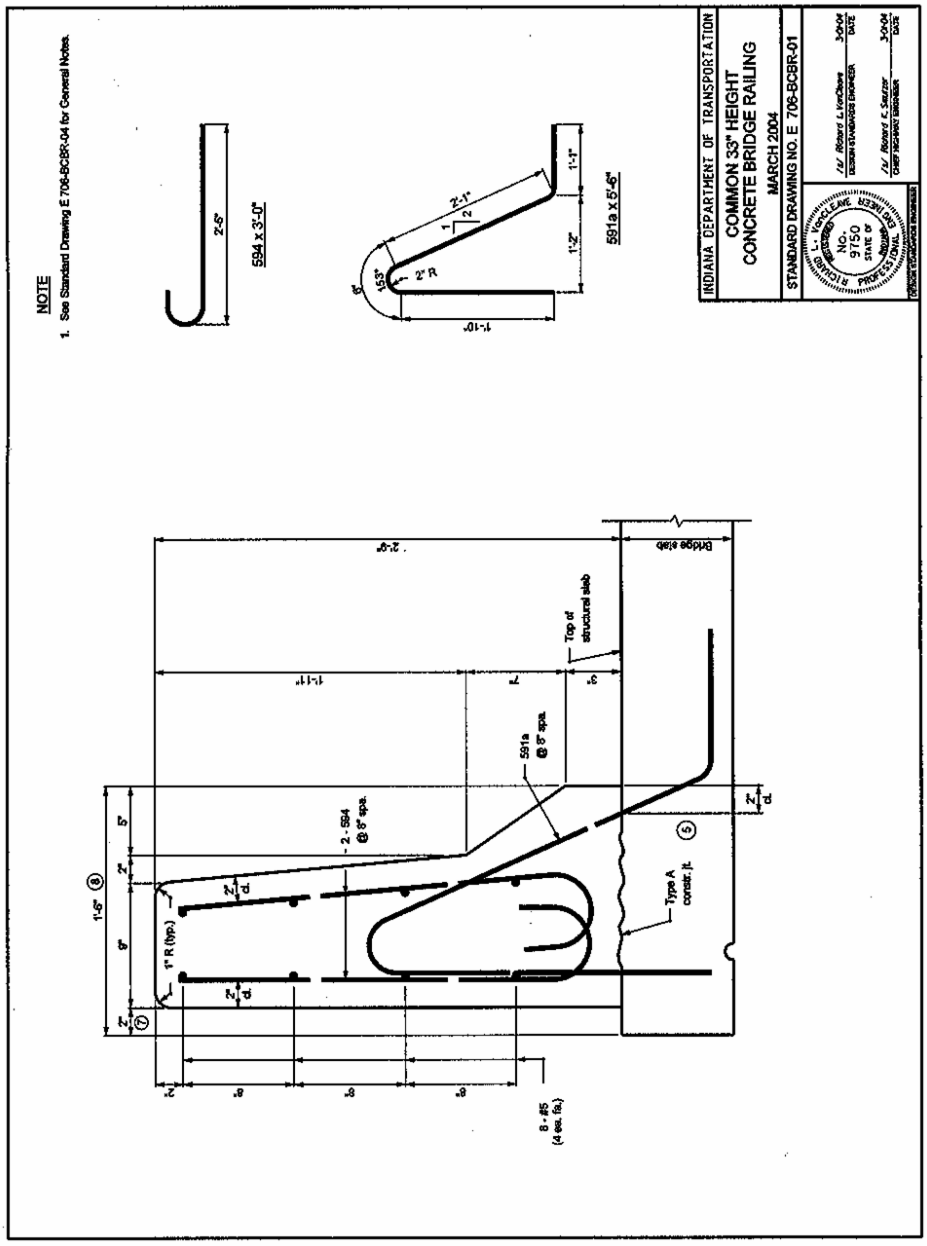


Figure 7.1 Standard Drawing of 33” Concrete Bridge Barrier

7.3 Finite Element Modeling of Secondary Elements

Several finite element bridge models were studied using the commercial finite element software, ABAQUS (2001). It was concluded that the eccentric beam model gives as close to real idealization as possible while retaining simplicity for practical use as discussed in Chapter 3. The eccentric beam model was extended to include the secondary elements in the bridge system.

In this study, the FE model of primary members such as bridge deck and steel girders is called “Base FEM”. The concrete deck is modeled by 8-node Mindlin shell elements (ABAQUS S8R), while the steel girder is modeled by 3-node Timoshenko beam elements (ABAQUS B32). The full composite action between the centroid of the girder and the mid-surface of the slab is modeled by rigid links (ABAQUS MPC). The bearings are modeled by assigning boundary conditions to the grounded spring elements (ABAQUS SPRING1) at their real location. For simply supported conditions, rotations in all directions are allowed. Minimum restraints are assigned for longitudinal and transverse movement while vertical restraint is placed at the supports. Rigid links are also applied to nodes between the girders and the deck.

For steel girder bridges, however, lateral bracing, such as diaphragms and cross bracings is typically used to prevent lateral movement of the girders. It also helps to distribute the load between girders. In this study, the lateral bracing is modeled by 3-node beam elements (ABAQUS B32). Diaphragms are assumed to be directly connected to the girders. The cross bracings are connected at the intersection of the flanges and the web. Rigid links are applied between the nodes to ensure full composite action.

The parapet (or barrier) is idealized using the beam elements (B32). The parapet is assumed to act as fully composite with the deck. Rigid links between the parapet and the deck provide this composite action in the model. Figure 7.2 shows the developed finite element model.

Two experiments are selected for comparison and verification of the developed finite element model. The first bridge is a full scale laboratory test conducted by Kathol

et al. (1995). This bridge is a 21.4 m (70 ft) simply supported steel girder bridge. The superstructure consists of three 137 cm (54 in) deep welded plate girders built compositely with a 19 cm (7.5 in) thick reinforced concrete deck. There are 3 girders with girder spacing of 3 m (10 ft). K frames are placed at approximately every 6.83 m (22.4 ft) along the span. The cross section of the bridge and the layout of strain gauges are shown in Figure 7.3. The test loading setup consisted of 12 post-tensioning rods simulating two side-by-side AASHTO HS-20 design trucks. Two sets of loads simulating two trucks were placed symmetrically with respect to the center girder in the transverse direction. For the elastic test, 2.5 times the HS20 truck load was applied on the rod plates. This load consisted of 362.5 kN (40 kips) for the center and rear wheels and 87.5 kN (10 kips) for the front wheels.

Figure 7.4 presents the measured and calculated bottom flange deflections and strains at various locations. As can be seen from this figure, the finite element models generally produce calculated deflections similar to the measured deflections. The maximum error between predicted and measured deflection is 9 % at the exterior girder of the mid-span. Good agreement between measured and calculated strains is also observed.

The finite element model including diaphragms is further verified with the results of the field test conducted by Canna and Bowman (2002). This bridge is located on Indiana SR 52 over 9th street in Lafayette, Indiana. It is a right angled five span continuous composite steel girder bridge with total length of 148.5 m (148 ft). Eight longitudinal steel girders and a 19 cm (7.5 in) deep reinforced concrete deck are used. The detailed geometry of the bridge model is shown in Figure 7.5. The diaphragms are connected to the longitudinal girders with fillet welds located on the tops of both flanges of the diaphragms and with intermittent welds along both sides of the web. The first 28 m (92 ft) span of the bridge was selected to be instrumented span. Strain gauges were placed at several locations on the bottom of the top and bottom flanges of the girders and mid-span of several diaphragms. The names and locations of the strain gauges, where comparisons with the finite element results are made, are given in Table 7.3. Several load cases are considered using a tandem axle dump truck weighting 232 kN. Table 7.4

shows the locations of the load cases considered. More descriptions on the field test of the bridge are available in the original report (Canna and Bowman 2002).

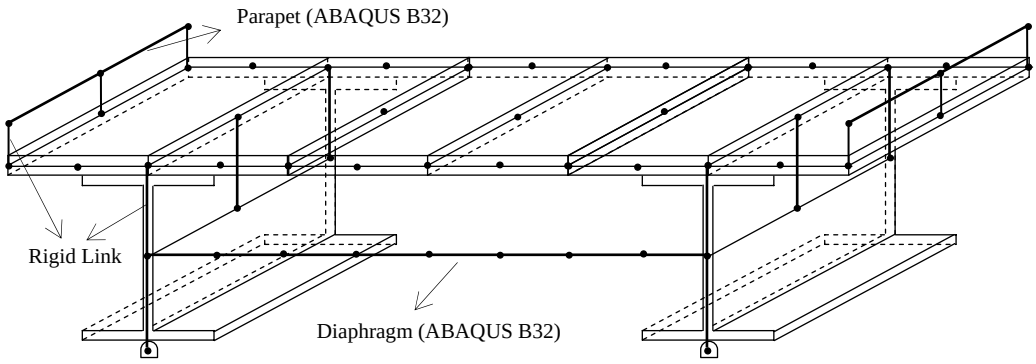
Figure 7.6 (a) shows the displacement comparisons between the finite element results and the test results. The finite element models generally predict the displacement results very well. The maximum displacement error at Girder 1 is 6 % for load case 3B. The predicted strains are compared to those obtained from the test results. Figure 7.6 (b) and (c) show the strain results at various locations. The correlations between calculated and measured strains are overall very good. The maximum error for girder strain is within 7 %. Based on the comparisons from the above, it can be concluded that the developed FE model used in this study is capable of predicting well the behavior of steel girder bridges.

Table 7.3 List of Strain Gauges (US 52 Bridge)

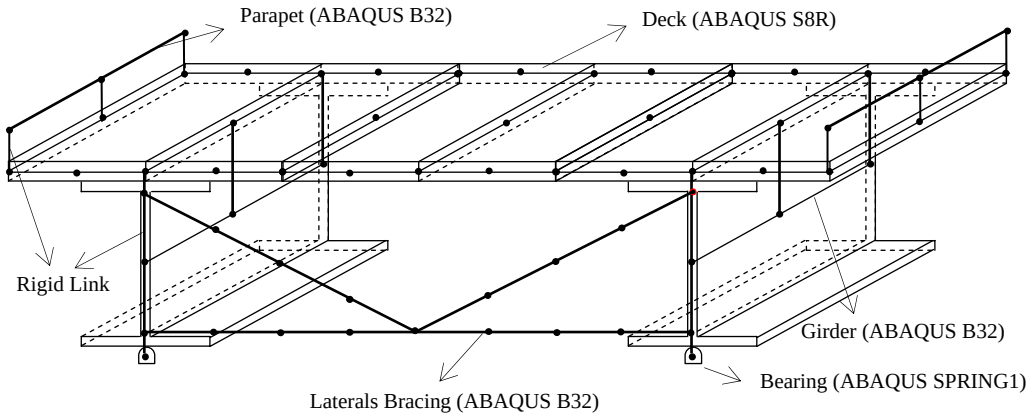
Strain Gauge	Location
G1A	Bottom of top flange (Girder #1)
G2A	Bottom of top flange (Girder #2)
G2D	Bottom of bottom flange (Girder #2)
D1A	Top of top flange (Diaphragm #1)
D1B	Bottom of bottom flange (Diaphragm #1)

Table 7.4 Load Cases for the US 52 Bridge Test

Load Case	Longitudinal Position	Transverse Position
2A	13 m (42’2”) from the end support	4 m (13’) from the curb
2B	13 m (42’2”) from the end support	3.3 m (10’10.5”) from the curb
3A	13.5 m (44’4”) from the end support	4 m (13’) from the curb



(a) Diaphragms + Parapet



(b) Cross Bracing + Parapet

Figure 7.2 Finite Element Model

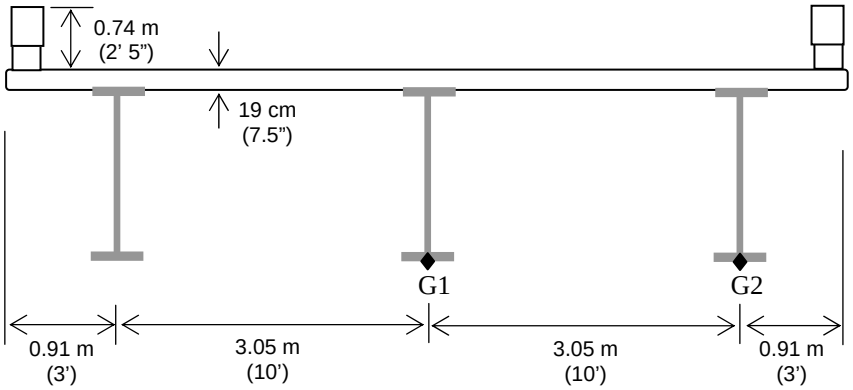
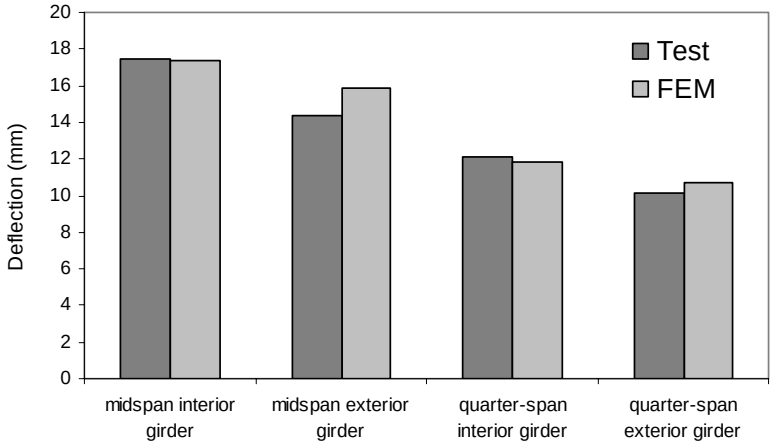
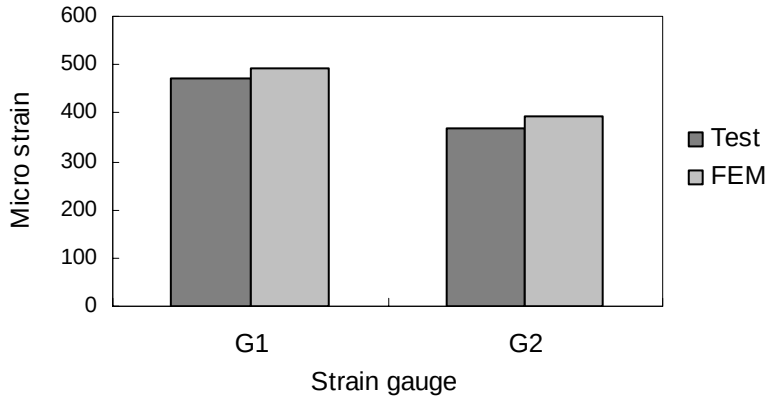


Figure 7.3 Layout of Strain Gauges at Mid-span (Nebraska Bridge)



(a) Bottom Flange Deflection



(b) Strain Results

Figure 7.4 Comparisons of FE results to Experimental Results (Nebraska Bridge)

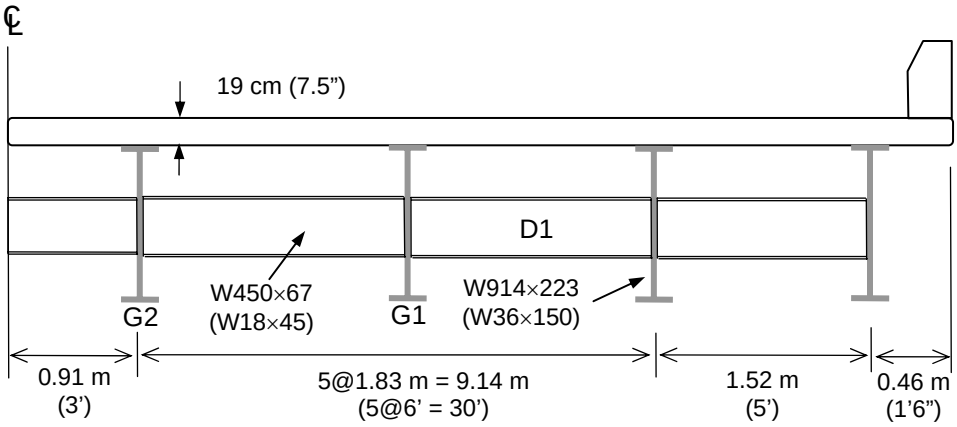
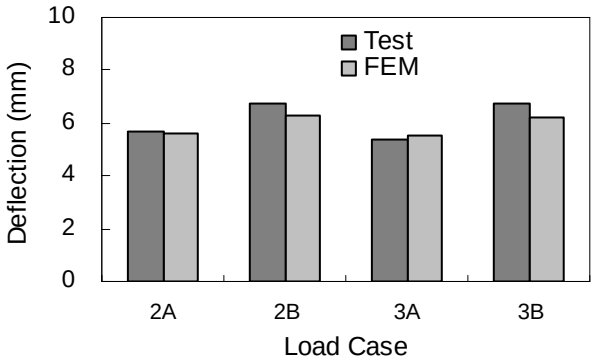
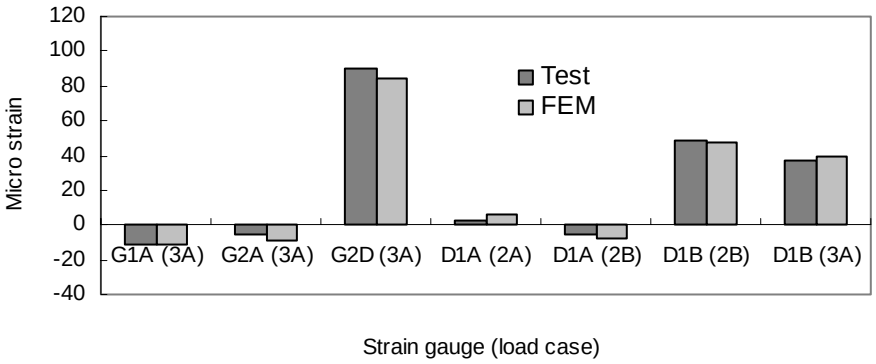


Figure 7.5 Cross Section of Instrumented Span (US 52 Bridge)



(a) Deflection at Girder 1



(b) Strain Results

Figure 7.6 Comparisons of FE Results to Experimental Results (US 52 Bridge)

7.4 Concrete Crack Model

The simplest method for considering the effect of deck cracking on the behavior of a bridge deck consists of reducing the deck's stiffness. Deck cracking can, then, be accounted for by using orthotropic material properties for the slab elements. In the cracked regions, the elastic modulus of the shell perpendicular to the crack direction is reduced, while the elastic modulus parallel to the crack direction and the actual thickness of the shell elements remains unchanged. In this manner, deck cracking can be taken into account in a simple way. However, this assumption oversimplifies the actual phenomenon across the crack surface.

In this study, a concrete crack model based on the strain decomposition technique (Chung and Sotelino 2004) has been adopted. This technique enables the explicit inclusion of physical behavior across the cracked concrete surface such as aggregate interlock and dowel action unlike other models that introduce these effects implicitly by means of a shear retention factor. The concrete material model has been extended to three-dimensional problems using a layered approach. The development and verification of the above model is discussed in detail in Chung and Sotelino (2004).

The above concrete crack model has been extended to include the modeling of pre-existing cracks in reinforced concrete bridge decks. The modification involves the state determination of pre-existing cracks. Since pre-existing cracks are not necessarily formed perpendicular to the principal stress direction, the crack surface can be subjected to a considerable amount of in-plane shear stress through dowel action and aggregate interlock. For this reason, the concrete crack model based on the strain decomposition technique is better able to represent the effect of pre-existing cracks in bridge decks than other modeling techniques.

The proposed crack model has been integrated into the ABAQUS shell elements through the user-defined material subroutine (UMAT). UMAT is called at all material calculation points of the elements. Thus, the user subroutine must update the stresses

($\boldsymbol{\sigma}_c$), concrete tangent stiffness ($\mathbf{C}_{conc}$), and solution dependent state variables at the end of the increment. The solution dependent variables for this case are crack displacements as defined by

$$\boldsymbol{\delta} = \{w \quad v\}^T . \tag{7.1}$$

This model consists of three routines, namely, "closed crack routine", "crack closing routine", and "open crack routine". Depending on the crack displacement, the appropriate routine is called. The detailed algorithm for the state determination and linearization is presented in Figure 7.7.

The first routine is called "Closed Crack Routine". This routine is used when the cracks in the bridge deck are closed, i.e., the crack displacement vector is zero at the beginning of the increment. In this case, the concrete is assumed to be an isotropic linear elastic material ($\mathbf{C}^{el}$). and , thus, strain decomposition does not need to be performed .

The second routine is the "Crack Closing Routine". This routine enables the modeling of the closing of pre-existing cracks in the bridge deck, which may occur due to bridge live loads. If the updated crack opening (w) becomes negative, the crack is closing within the current increment. Once the crack is closed, i.e., when the normal stress component across the crack changes from tension to compression, it is assumed that the concrete material recovers its linear elastic characteristics. The stiffness matrix of intact solid concrete is then reinserted for these crack closing states. The stress states just after the closing of the crack can be mathematically written as follows (de Borst and Nauta 1985):

$$\boldsymbol{\sigma}^c = \boldsymbol{\sigma}^o + \mathbf{C}^{el}\left(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^o\right) \tag{7.2}$$

where $\boldsymbol{\sigma}^o$ and $\boldsymbol{\varepsilon}^o$ are, respectively, the stress state and strain state at the moment the crack begins to close.

The last state determination routine is "Open Crack Routine". This routine is used when the crack displacement increases indicating that the crack is opening. In the present modeling technique, the cracks in the concrete material are assumed to be smeared into the shell element. The total strain increment is divided into two parts: the strain increment due to the cracks and the strain increment due to the intact concrete material between

cracks. Using the crack strain increment, a proper constitutive model for aggregate interlock and dowel action are included into the cracked concrete model through the average crack stress-strain relations. The intact solid concrete can be modeled using a linear-elastic isotropic constitutive material model, whereas this would not be appropriate for cracked concrete since it behaves highly nonlinearly due to the wedging effect between aggregate particles and due to the presence of the rebars. After mathematical manipulation (Chung and Sotelino 2004), the stiffness matrix of cracked concrete can be calculated by

$$\mathbf{C}^{cr} = \left(\frac{1}{s} \mathbf{\Omega} (\mathbf{B} + \mathbf{G})^{-1} \mathbf{\Omega}^T + \mathbf{S}^{el} \right)^{-1} \tag{7.3}$$

where s is the crack spacing between two adjacent cracks, $\mathbf{B}$ is the constitutive matrix of the aggregate interlock, $\mathbf{G}$ is the constitutive matrix of the dowel action, $\mathbf{S}^{el}$ is the tangential compliance matrix of linear elastic concrete and $\mathbf{\Omega}$ is the transformation matrix reflecting the orientation of the crack. Using the strain decomposition and the crack shear constitutive relations, more realistic results are expected to be obtained than those resulting from the commonly used method, which simply reduces the shear modulus after cracking.

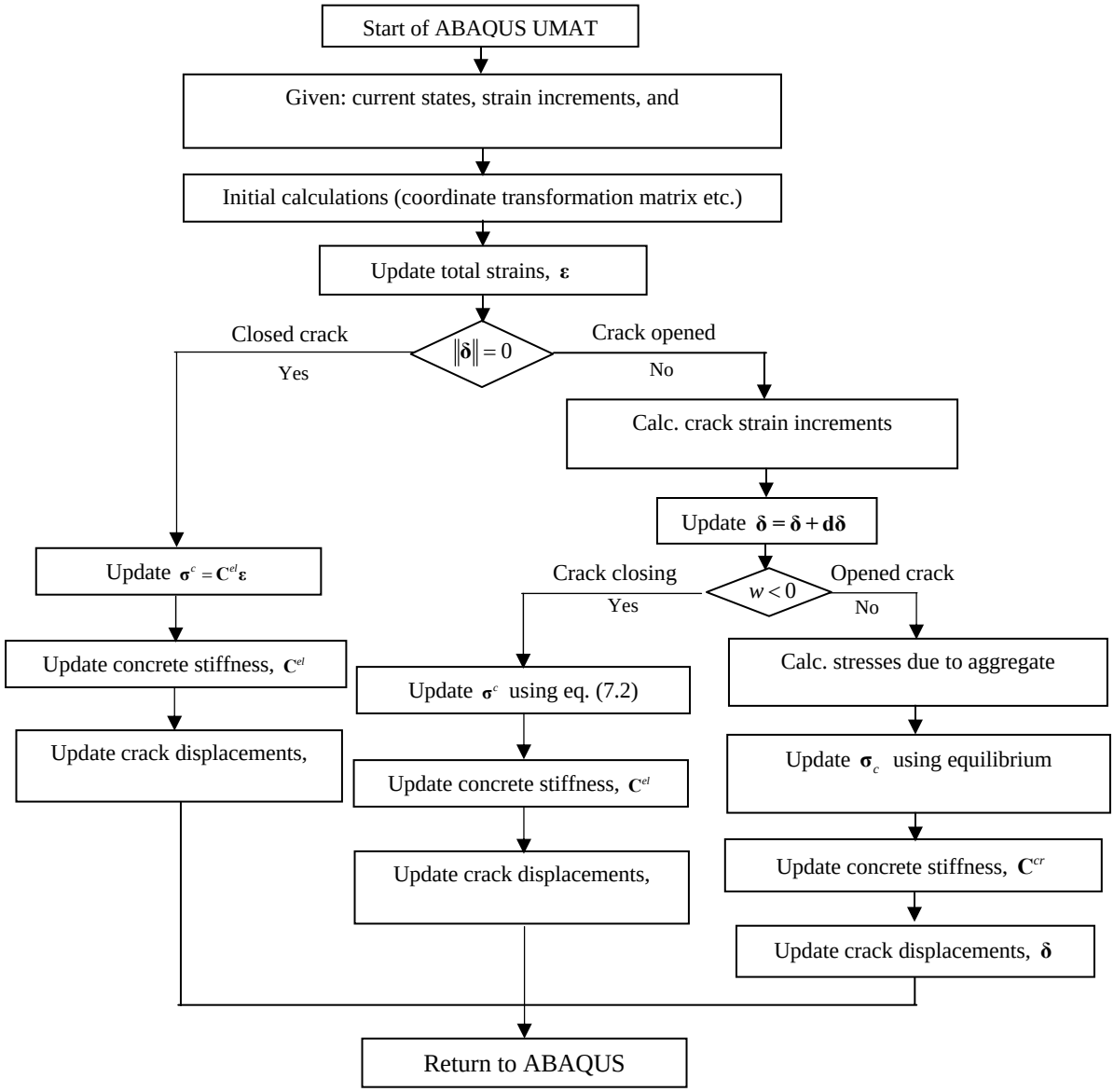


Figure 7.7 State Determination for Cracked Concrete

7.5 Load Distribution Factor Comparisons

7.5.1 Effects of Secondary Elements

The calculated load distribution factors considering the secondary elements are compared with those obtained using AASHTO LRFD equation and Simplified equation. Four different FE models are investigated. The first FE model includes primary members including deck, girders, and bearings. It is called “Base FEM”. The other three models are modifications of the Base FEM model. In the second model, lateral bracing is added in the FE model. In the third model, parapets are added, but no lateral bracing. The last model, which is the most comprehensive, both lateral bracing and parapets are added to the Base FEM. This model is referred to as the “As Is” model.

Two sets of AASHTO HS20 design truck wheel load are applied to produce the maximum effect on the live load distribution. For example, in the longitudinal direction, the maximum moment in simple span bridges occurs when the center line of the span is midway between the center of gravity of loads and the nearest concentrated load. The live load distribution factor is also greatly influenced by the transverse loading position. Several truck positions are investigated for the transverse direction to find the maximum effect after the longitudinal position is determined.

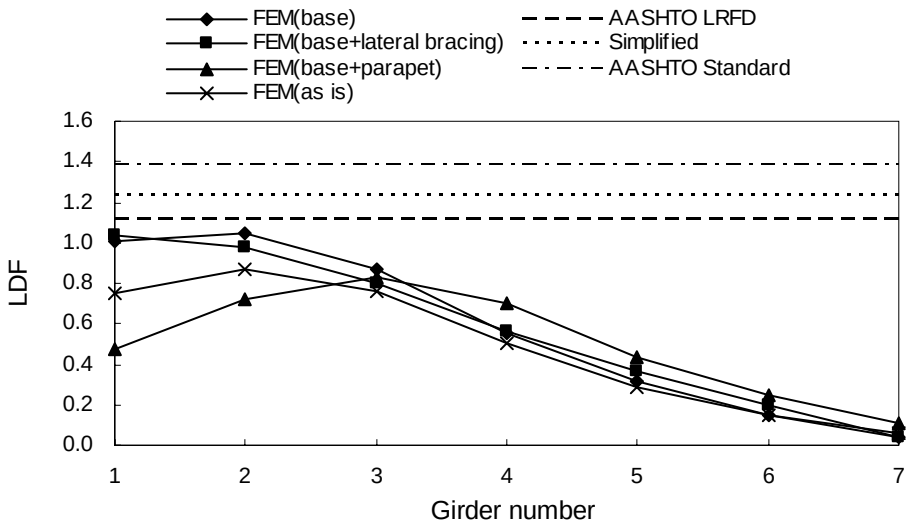
The maximum load distribution cases for two selected bridges are shown in Figure 7.8. Note that Girders 1 and 7 and 1 and 8 are the exterior girders for Bridge 1 and 4, respectively. The maximum interior LDF is found on the first interior girder when the Base FEM is considered or only the lateral bracing is modeled. However, the maximum LDF is found on the second interior girder for the model considering parapets. It is also found that the maximum LDF of the “As Is” model can occur on the first interior girder or the second interior girder for all considered bridges. This indicates that the presence of secondary elements significantly alters the load distribution for each bridge.

In Figure 7.9, the Base FEM LDF is compared to the FEM LDF considering secondary elements. The consideration of lateral bracing and parapet reduces the LDF by up to 11 % and 25 %, respectively. The LDFs obtained by the "As Is" FE model are 17 % to 38 % less than those obtained using the Base FEM LDF.

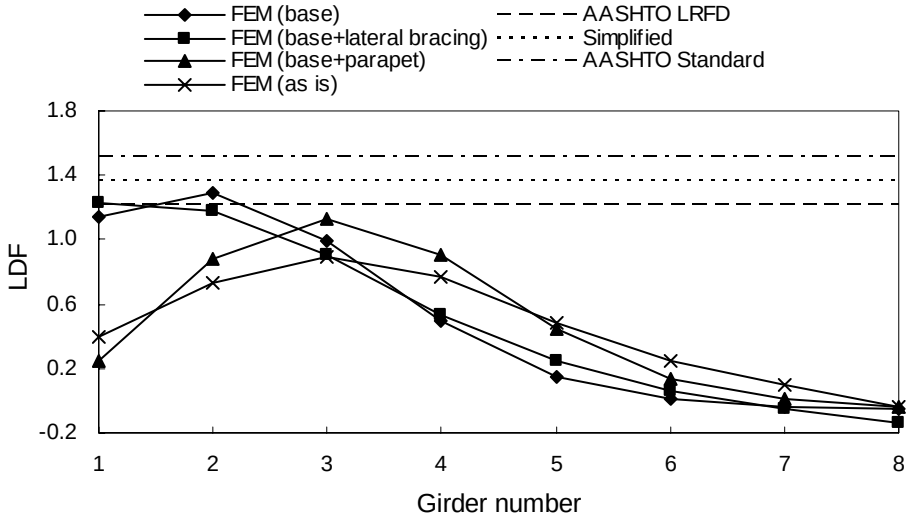
The comparisons of LDF obtained from code equations to LDF from FEM are shown in Figure 7.10 through Figure 7.12 for the nine selected bridges. Each of the data points is an LDF ratio for one of the considered bridges. In Figure 7.10, the LRFD LDF is compared to the FEM LDF. The Base FEM produces LDF greater than LRFD LDF for some of the considered bridges. In other words, the LRFD LDF is not conservative in these cases. However, the addition of lateral bracing or parapet decreases LDF when compared to Base FEM. The LDFs obtained from the "As Is" FEM are always less than LRFD LDF. It is observed that the consideration of lateral bracing and parapet separately produces an LDF up to 12 % and 27 % less than LRFD LDF, respectively. The "As Is" FE model gives LDF value less than up to 39 % than the LRFD value. There is no apparent trend in the LDF ratio with respect to either girder spacing or span length.

The ratios of FEM LDF to Simplified LDF are presented in Figure 7.11. All FE models produce lower LDF values than FEM LDFs. The modeling of lateral bracing and parapet produces LDF up to 20 % and 36 % less than the Simplified LDF. The "As Is" FEM LDF produces the LDF ranging from 20 % to 45 % less than the Simplified LDF. Similar trends are observed in the comparison between FEM LDF and Standard LDF as shown in Figure 7.12.

It is clear from the previous observation that the effect of secondary elements on the load distribution factor can be significant. In general, it is found that the presence of secondary elements helps the lateral load distribution of girder bridges. Furthermore, both the AASHTO LRFD and Simplified LDF formulas always produce conservative results when all secondary elements are considered.

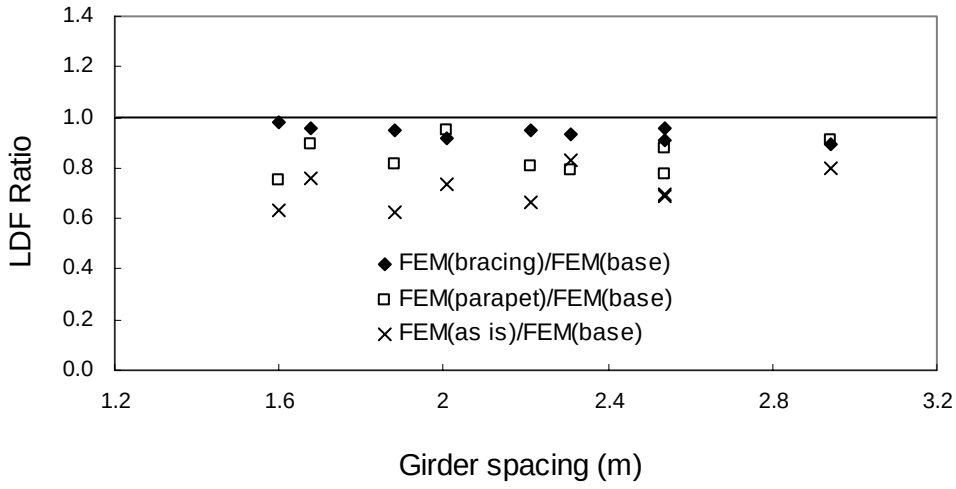


(a) Bridge 1



(b) Bridge 4

Figure 7.8 Lateral Distribution of LDF (Secondary Elements)



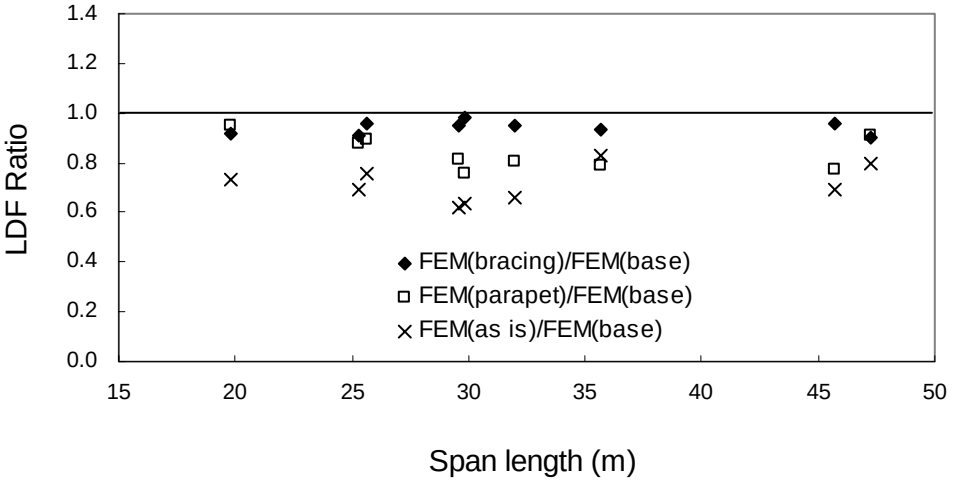
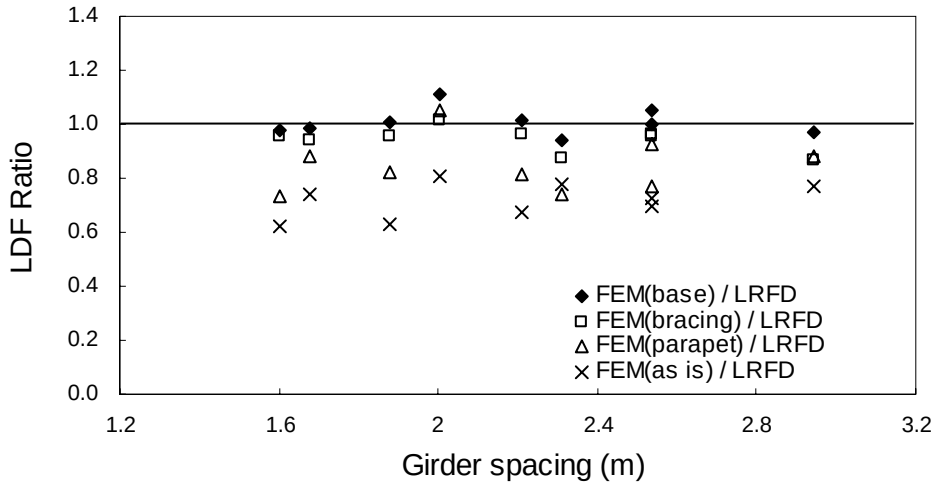


Figure 7.9 Effect of Secondary Elements on LDF



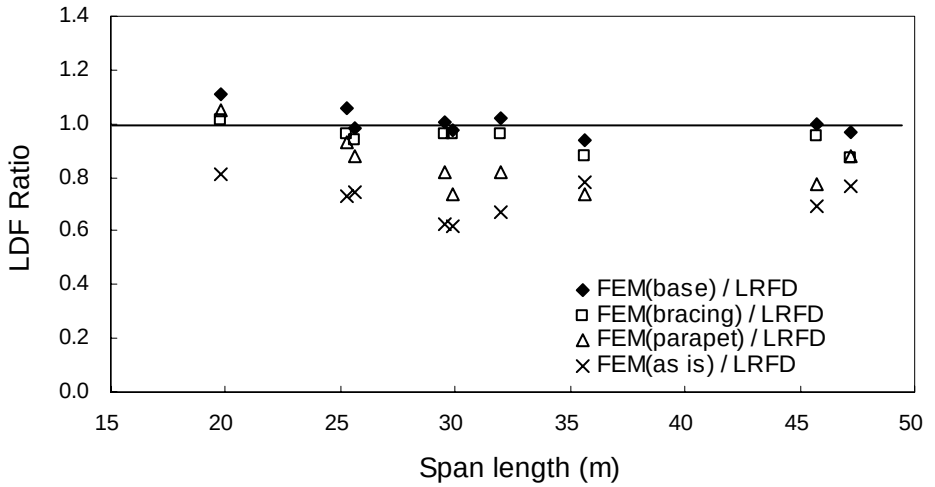
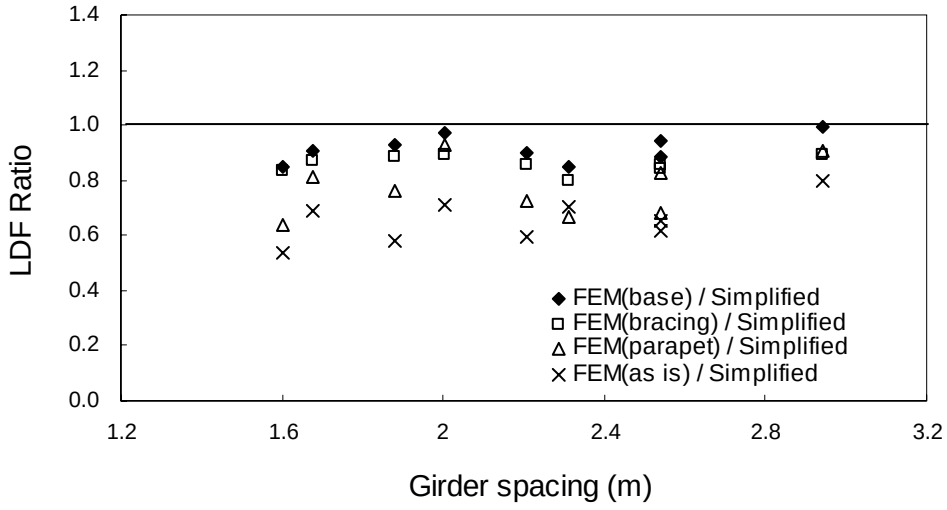


Figure 7.10 Comparisons of FEM LDF to LRFD LDF



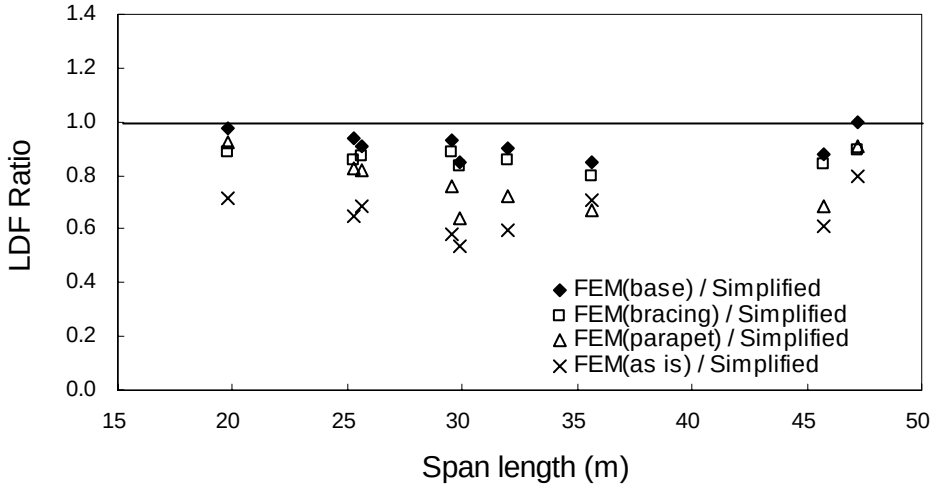
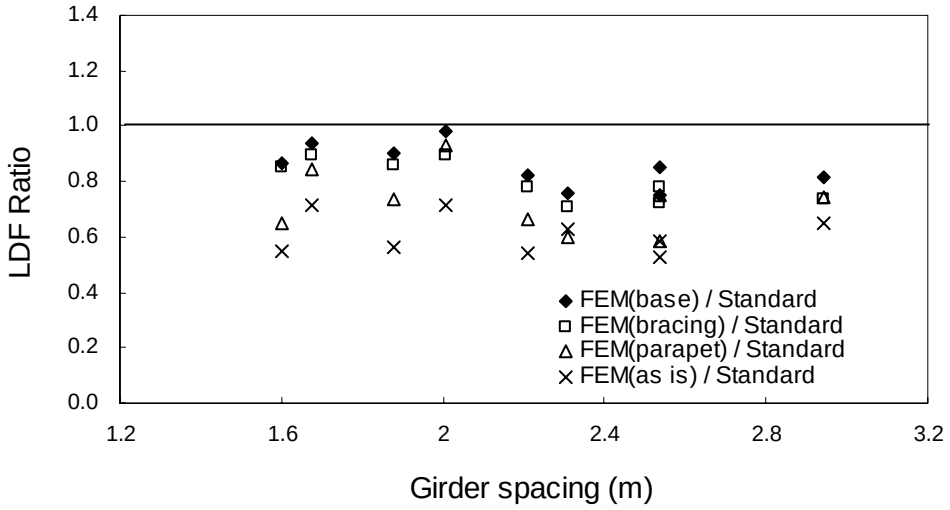


Figure 7.11 Comparisons of FEM LDF to Simplified LDF



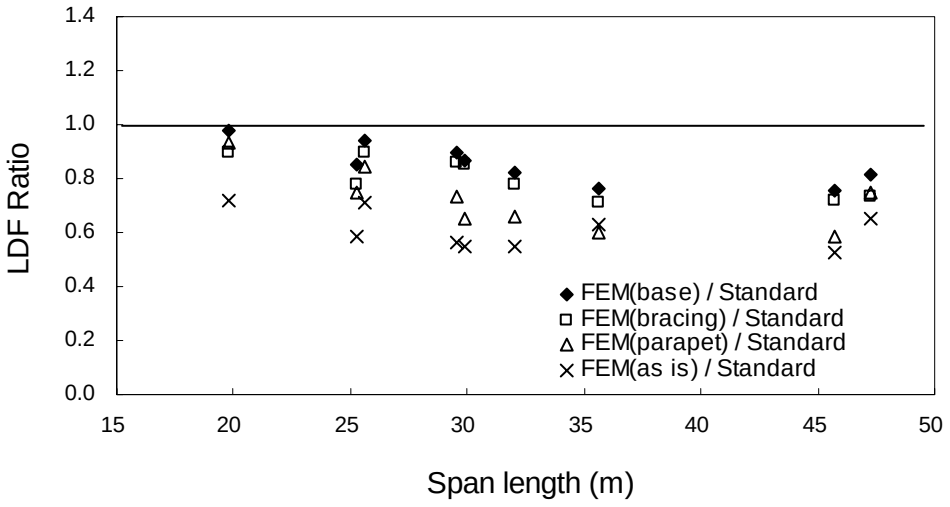


Figure 7.12 Comparisons of FEM LDF to Standard LDF

7.5.2 Effects of Bridge Deck Cracking

The characteristics of the nine selected Indiana bridges with pre-existing cracks are shown in Table 7.2. These bridges are analyzed within the developed nonlinear finite element (NLFE) framework to determine the effect of deck cracking on lateral load distribution.

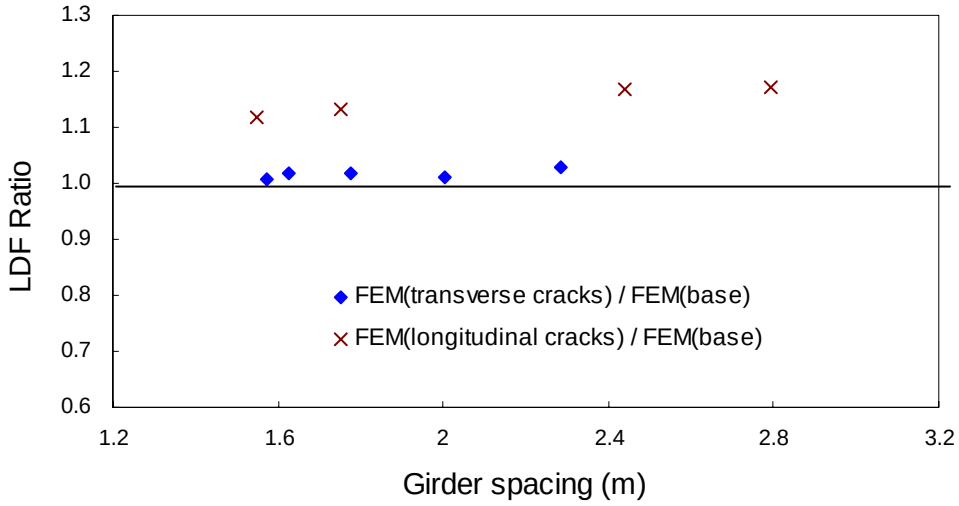
The parameters considered in the performed case studies are crack patterns (transverse cracks, longitudinal cracks) and crack depth (full depth, partial depth crack). The assumptions made in this study are discussed next. First, the smearing concept is used to model the pre-existing cracks. If cracks exist at an integration point, the cracking is modeled through an adjustment of the material properties, which effectively treats the cracking as a "smeared band" of cracks, rather than discrete cracks. Thus, the crack is idealized by an infinite number of parallel fissures across the element. Other assumptions include the area and length of the cracked elements and the depth of the cracking. A different number of cracked elements is assumed and only the case that produces the maximum effect is presented in this manuscript. The cracks are assumed to run the entire length of bridge in both the transverse and longitudinal directions. The last assumption is concerned with the depth of the cracks. It is difficult to measure the depth of a crack in the field since only the top surface of the crack is visible due to the use of stay-in-place formwork or due to lack of accessibility. In the simulations, however, the use of layered shell element enables the simulation of different crack depths.

It should be noted that the material properties of the beam elements are assumed to be linear elastic since the stresses in the bridge girders under service loads are much smaller than the steel yield limits. It should be noted that secondary elements are modeled in this case.

Figure 7.13 shows a comparison between LDF obtained from the nonlinear analyses considering deck cracking and the linear elastic analysis (Base FEM) for the selected bridges. As can be seen, transverse cracking of bridge deck slightly increases load distribution by 1 % to 3 % when compared to the values obtained from a linear elastic analysis. However, longitudinal cracking results in higher LDF by 12 % to 17 %.

The LDFs obtained from NLFE analyses are compared to the LDFs obtained from the LRFD LDF, Simplified LDF, and Standard LDF in Figure 7.14 through Figure 7.16. It is observed that the ratios of FEM LDF considering longitudinal cracking to LRFD LDF are larger than 1 for three of four bridges as shown in Figure 7.14. This indicates that the LRFD LDF, which is based on the linear elastic analysis, can result in an unconservative LDF with respect to a FEA considering longitudinal cracking of bridge deck. However, the Simplified LDF generally produces conservative results except for one bridge (Bridge 16) as shown in Figure 7.15. It is also observed that the FEM LDF considering transverse cracking is generally less than the LRFD LDF and the Simplified LDF.

The effect of both secondary elements and deck cracking is further investigated for Bridge 16. This bridge has been identified as having a higher calculated LDF than both the LRFD LDF and the Simplified LDF when longitudinal cracking is considered. This bridge has secondary elements including steel diaphragms and 84 cm (33”) oncrete parapet. Figure 7.17 shows the lateral distribution of LDF for Bridge 16. The consideration of both secondary elements and deck cracking produces LDF values lower than LRFD, Simplified, and Standard LDFs. Therefore, the effect of deck cracking, more specifically longitudinal cracking, increases the LDF, but the decrease in the LDF due to the presence of secondary elements results in LRFD, Simplified and Standard LDF values which are also conservative.



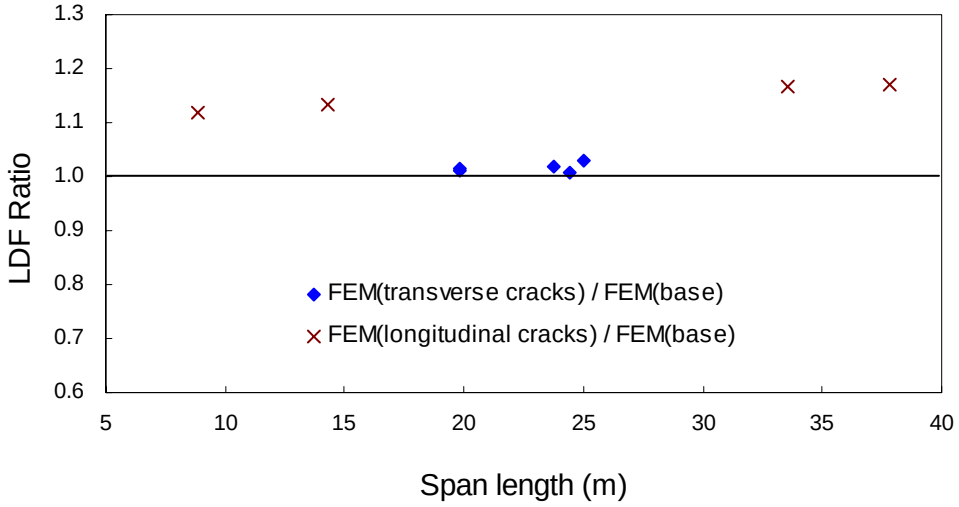
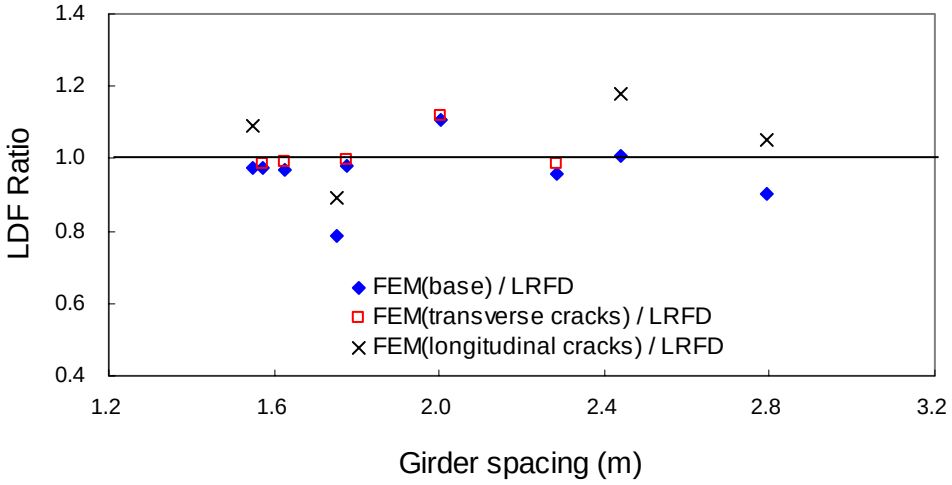


Figure 7.13 Effect of Deck Cracking on LDF



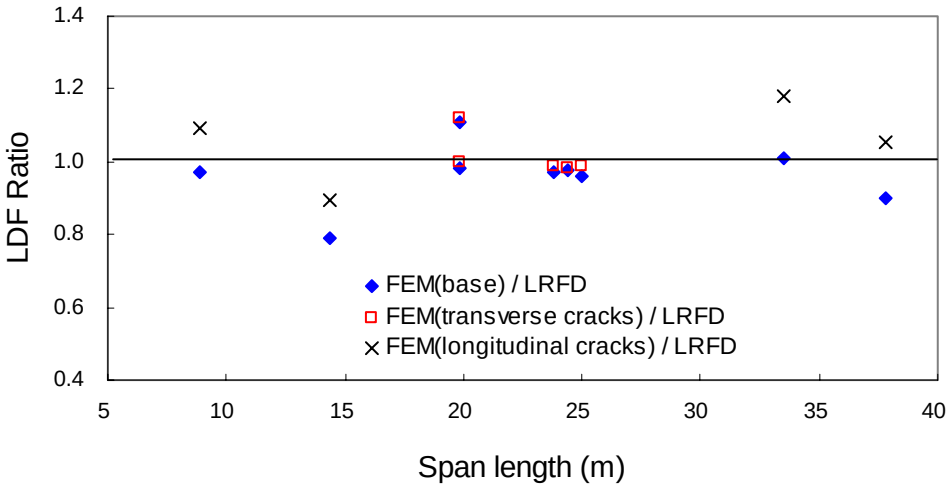
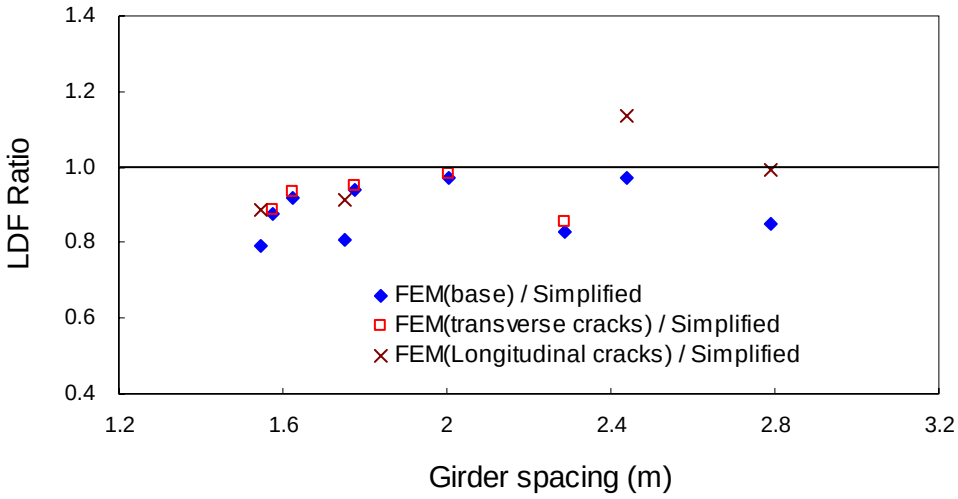


Figure 7.14 Comparisons of FEM LDF vs. LRFD LDF



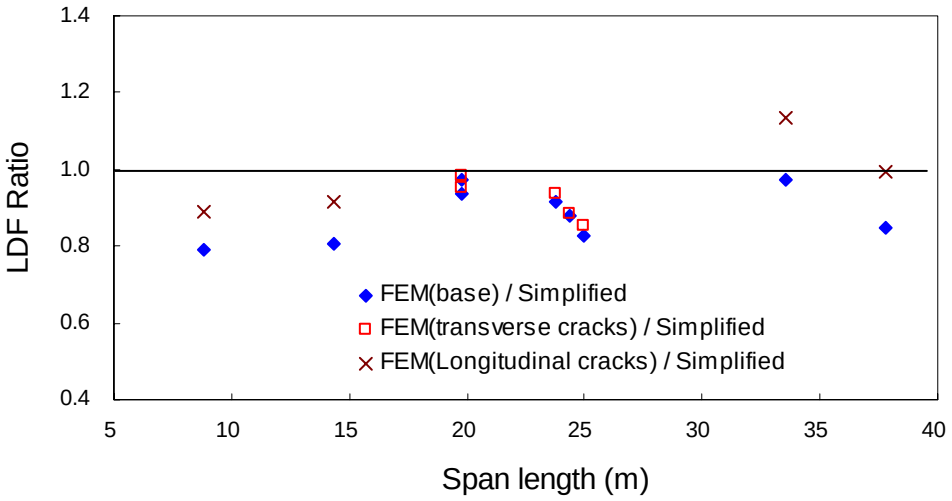
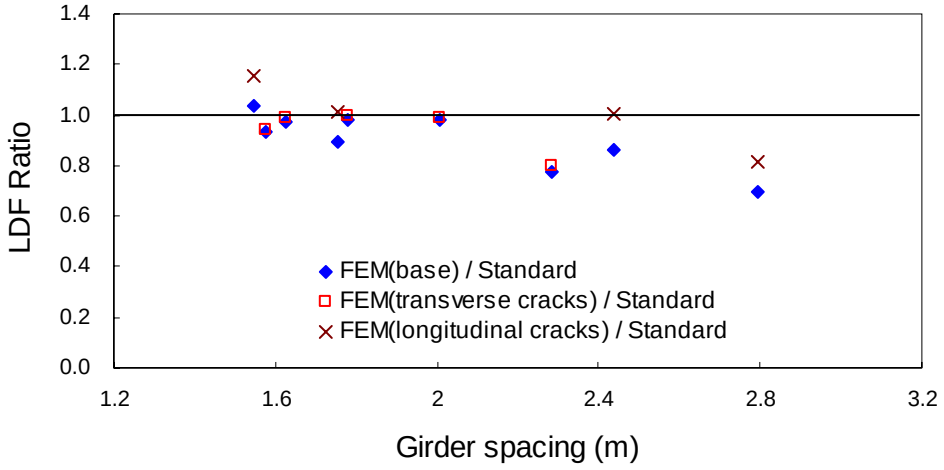


Figure 7.15 Comparisons of FEM LDF vs. Simplified LDF



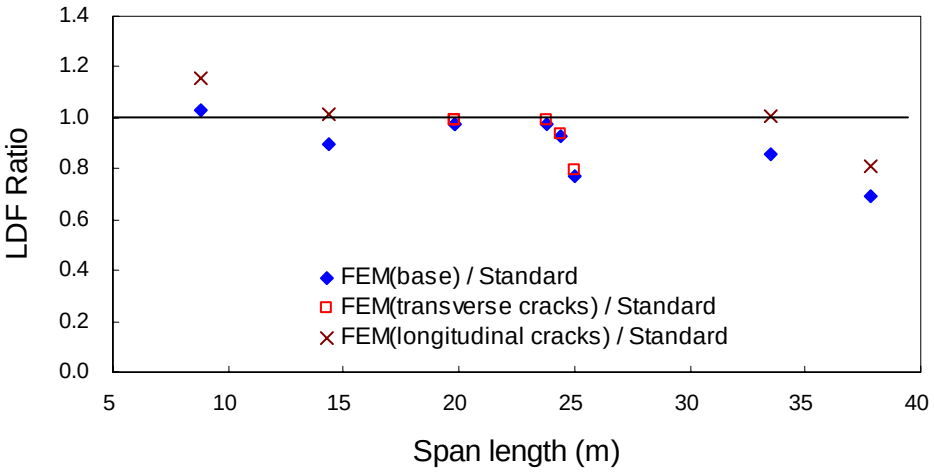


Figure 7.16 Comparisons of FEM LDF vs. Standard LDF

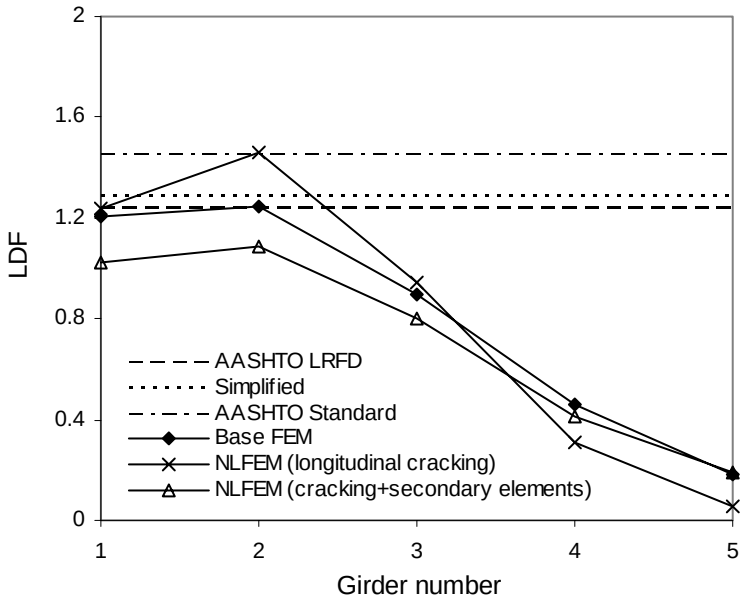


Figure 7.17 Lateral Distribution of LDF for Bridge 16

7.6 Summary

The AASHTO LRFD load distribution factor equation was developed based on linear elastic finite element analysis considering only primary members, i.e., the effects of secondary elements such as lateral bracing and parapets were not considered. Meanwhile, many bridges have been identified as having significant cracking in the concrete deck. Even though deck cracking is a well-known phenomenon, the significance of pre-existing cracks on the live load distribution has not yet been assessed.

The purpose of this chapter was to investigate the effect of secondary elements and deck cracking on the lateral load distribution of girder bridges. First, secondary elements such as diaphragms and parapet were modeled using the finite element method, and the calculated load distribution factors were compared with the code-specified values. Second, the effects of typical deck cracking and crack types that have a major effect on load distribution were identified through a number of nonlinear finite element analyses.

It was established that the presence of secondary elements can produce load distribution factors that are up to 40 % less than the AASHTO LRFD values. Longitudinal cracking was found to produce the load distribution factor up to 17 % greather than the LRFD value, while the transverse cracking was found to not significantly influence the transverse distribution of moment.

CHAPTER 8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Summary

The main objective of this research is to provide the simplest, yet sufficiently accurate equation for calculation of the load distribution factor. A new simplified wheel load distribution factor (LDF) equation is postulated based on the current AASHTO-LRFD specification. The longitudinal stiffness parameter (K_g) and the slab thickness parameter (t_s) that appear in the LRFD equation are implicitly embedded in the simplified expression. The simplified equation is developed based on the database of the steel girder bridges and eliminates the iterative procedure introduced by the LRFD formula. The accuracy and applicability of the simplified equation are demonstrated by comparisons to the AAHSTO-Standard, AASHTO-LRFD, and AASHTO level-three analysis, namely finite element (FE) analysis. Both the steel girder bridges and prestressed concrete girder bridges are investigated in this research.

Another objective of this study is to investigate the effects of secondary elements and deck cracking on the lateral load distribution of girder bridges. Several Indiana bridges are selected and analyzed considering each secondary element separately and in combination. The secondary elements in the bridge system considered in this study include lateral bracing and parapets. A concrete crack model is incorporated in the finite element model to account for the pre-existing cracks on bridge deck. Nonlinear FE analyses are performed to examine the influence of deck cracking on the load distribution factor. Finally, for one of the selected bridges, both concrete cracking and secondary elements are considered to investigate their combined effect on lateral load distribution.

8.2 Conclusions

Based on comparisons of the load distribution factor (LDF) from the Simplified equation, AASHTO-LRFD equation, AASHTO-Standard equation, and finite element analyses, the following conclusions can be drawn for steel girder bridges.

- The developed Simplified equation produces LDF values that are always conservative when compared to those obtained from the finite element analyses.
- The LDF values using the Simplified equation are generally greater than ones obtained using AASHTO-LRFD specification. Therefore, the Simplified equation provides a simple yet safe specification for LDF calculation.
- The Simplified and AASHTO-LRFD LDF are greater than the AASHTO-Standard LDF when span length and girder spacing are relatively small, in which case the Standard LDF tends to be unconservative. Therefore, the simplified formula is similar to the LRFD formula in its ability to represent the load distribution.
- The new simplified equation is simple, requires no iteration, and produces LDF values that are at least as conservative as the ones obtained by the LRFD equation.

The applicability of the Simplified LDF equation has been investigated for prestressed concrete (PC) girder bridges. The following conclusions are drawn based on the comparisons of the LDF from the Simplified equation, AASHTO-LRFD equation, AASHTO-Standard equation, and finite element analyses.

- The LRFD LDF equation for the PC girder bridges is the same as the one for the concrete slab on steel girder bridges. According to the finite element analysis of the 17 Indiana representative PC girder bridges, the LDF values obtained from the LRFD equation can be conservative by about 30%.

- According to the finite element analysis of the 17 Indiana representative PC girder bridges, the Simplified LDF equation, similarly to the LRFD LDF equation, is always conservative.
- AASHTO recommends a specific PC girder type for a given range of span length. If the recommended AASHTO type is used for the suggested span length, the Simplified equation represents the LDF values well. In other words, the Simplified LDF is generally more conservative than the LRFD LDF if the recommended AASHTO type is used.

The following conclusions can be drawn from the investigation of secondary elements and deck cracking on LDF.

- Secondary elements significantly help the transverse distribution of moment. It is found that the consideration of secondary elements produces LDF up to 39 % less than the AASHTO LRFD LDF. Both the AASHTO LRFD equation and Simplified equation always produce conservative LDF when secondary elements are considered.
- Transverse cracking in a bridge deck reduces its stiffness, thus resulting in higher deflection. However, it does not significantly influence the transverse distribution of moment.
- Longitudinal cracking increases the LDF significantly. For the limited number of tested cases, an LDF up to 17% higher than the AASHTO LRFD LDF has been observed. Thus, an increased LDF should be anticipated in the analysis and bridge rating when longitudinal cracking is present. However, it should be kept in mind that this increase is somewhat offset by the contributions from the secondary elements.

8.3 Recommendations for Simplified Specifications

Steel Girder Bridges

The new Simplified LDF equation should be used within the applicable range. Furthermore, the designer needs to check the final girder selection. The Simplified LDF equation works best if the selected girder produces a K_g less than the value obtained by

$$K_g = 189940 \cdot e^{\left(\frac{L}{59}\right)} \qquad \text{(US customary units)}$$

$$K_g = 7.91 \times 10^{10} \cdot e^{\left(\frac{L}{18000}\right)} \qquad \text{(SI units)}$$

In other words, a safe LDF value will be obtained as long as the final K_g is less than the upper bound K_g presented above.

Prestressed Concrete Girder Bridges

The Simplified LDF is generally greater than the LRFD LDF if the recommended AASHTO PC girder type for span length is used. If a larger PC section type is used, there is a risk of the Simplified LDF being less than the LRFD LDF. Although the LDF may still be conservative, it is recommended that the LRFD LDF be used in this case.

8.4 Simplified Load Distribution Factor Specification

The new simplified live load distribution factor equation is as follows:

$$0.15 + 0.73 \cdot \frac{S^{0.8}}{L^{0.3}} \cdot e^{\left(\frac{L}{590}\right)}$$

where S is the girder spacing (ft) and L is the span length (ft). The applicable range for span length is 44 ft to 122 ft. The applicable range for girder spacing is from 4 ft to 10 ft. The simplified equation is only valid when the slab thickness is 8 in. and the skew angle along the supports ranges from $0°$ to $45°$.

When the line supports are skewed and the difference between skew angles (θ) of two adjacent lines of supports exceeds $30°$, the load distribution factor shall be adjusted by the skew reduction factor:

$$1 - 0.59 \cdot \frac{S^{0.5}}{L^{0.75}} \cdot (\tan\theta)^{1.5} \cdot e^{\left(\frac{L}{236}\right)}$$

Commentary

The designer must check the final girder selection. The simplified load distribution factor equation works best if the selected girder produces a K_g less than the value obtained by

$$K_g = 189940 \cdot e^{\left(\frac{L}{59}\right)}$$

If a larger section is used, there is a risk of the simplified load distribution factor being less than the AASHTO-LRFD load distribution factor. In this case, it is recommended that the AASHTO-LRFD load distribution factor be used instead.

REFERENCES

AASHTO LRFD Bridge Specifications (1998). 2nd Edition, American Association of State Highway and Transportation Officials, Washington, D.C.

AASHTO Standard Specifications for Highway Bridges (1996). 16th Edition, American Association of State Highway and Transportation Officials, Washington, D.C.

ABAQUS/Standard User's Manual (2001). Version 6.2. Hibbitt, Karlsson & Sorensen, Inc.

Balmer H.A. (1978). "Another Aspect of Error in Eccentric Beam Formulation." *International Journal for Numerical Methods in Engineering*, Vol. 12, pp 1761-1763.

Barr, P.J., Eberhard, M.O., and Stanton, J.F. (2001). "Live-Load Distribution Factors in Prestressed Concrete Girder Bridges." *Journal of Bridge Engineering*, Vol. 6, No. 5, September/October, pp 298-306.

Bishara, A.G., Liu, M.C., and Nasser, D.E (1993). "Wheel Load Distribution on Simply Supported Skew I-Beam Composite Bridges." *Journal of Structural Engineering*, Vol. 119, No. 2, February, pp 399-419.

Brockenbrough, R.L. (1986). "Distribution Factors for Curved I-Girder Bridges." *Journal of Structural Engineering*, Vol. 112, No. 10, October, pp 2200-2215.

Burdette, E.G. and Goodpasture, D.W. (1971). "Full-scale Bridge Testing: An Evaluation of Bridge Design Criteria." University of Tennessee, December.

Canna, T.L and Bowman, M.D. (2002). "Fatigue Behavior of Beam Diaphragm Connections with Intermittent Fillet Welds; Part 1: Field Investigation." FHWA/IN/JTRP-2001/10-I-1, Indiana Department of Transportation. March.

Chan, T.H.T. and Chan, J.H.F (1999). "The Use of Eccentric Beam Elements in the Analysis of Slab-on-Girder Bridges." *Structural Engineering and Mechanics*, Vol. 8, No. 1, pp 85-102.

Chen, Y. (1995). "Refined and Simplified Methods of Lateral Load Distribution for Bridges with Unequally Spaced Girder: I. Theory." *Computers & Structures*, Vol.55, No. 1, pp 1-15.

Chen, Y. (1995). “Refined and Simplified Methods of Lateral Load Distribution for Bridges with Unequally Spaced Girder: II. Applications.” *Computers & Structures,* Vol.55, No. 1, pp 17-32.

Chen, Y. (1999). “Distribution of vehicular loads on bridge girders by the FEA using ADINA: modeling, simulation, and comparison.” *Computers & Structures*, Vol. 72, pp 127-139.

Chung, W. and Sotelino, E.D. (2004). “Nonlinear Finite Element Analysis of Composite Steel Girder Bridges.” ASCE *Journal of Structural Engineering*, October issue, in-press.

Cook, R.D., Malkus, D.S., Plesha, M.E. (1989). *Concepts and applications of finite element analysis*, Third Edition, Wiley, New York.

de Borst, R. and Nauta, P. (1985). “Non-Orthogonal Cracks in a Smeared Finite Element Model.” *Engineering Computation*, 2(3), pp 35-46.

Eamon, C.D. and Nowak, A.S. (2002). “Effect of Edge-Stiffening Elements and Diaphragms on Bridge Resistance and Load Distribution.” *Journal of Bridge Engineering*, Vol. 7, No. 5, September/October, pp 258-266.

Eamon, C.D. and Nowak A.S. (2004). “Effect of Secondary Elements on Bridge Structural System Reliability Considering Moment Capacity,” *Structural Safety*, Vol. 26, pp 29-47.

Ebeido, T. and Kennedy, J.B. (1996). “Girder Moments in Simply Supported Skew Composite Bridges.” *Canadian Journal of Civil Engineering*, Vol. 23, pp 904-916.

Eom, J. and Nowak, A.S. (2001). “Live Load Distribution for Steel Girder Bridges.” *Journal of Bridge Engineering*, Vol. 6, No. 6, November/December, pp 489-497.

Federal Highway Administration (1995), Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation’s Bridges, Report No. FHWA-PD-96-001, Washington D.C., 1995.

French, C., Eppers, L., Le, Q., and Hajjar, J.F. (1999). “Transverse Cracking in Concrete Bridge Decks,” *Transportation Research Record*, 1688, pp 21-29.

Frosch, R.J., Blackman, D.T., and Radabaugh, R.D. (2003). “Investigation of Bridge Deck Cracking in Various Bridge Superstructure Systems.” FHWA/IN/JTRP-2002/25, Indiana Department of Transportation. February.

Fu, K and Lu, F (2003). “Nonlinear Finite Element Analysis for Highway Bridge Superstructures.” *Journal of Bridge Engineering*, Vol. 8, No. 3, May, pp 173-179.

Goodrich, B.L. and Puckett, J.A. (2000). "Simplified load distribution for vehicles with nonstandard axle gauges." *Transportation Research Record*, No. 1696, pp158-170.

Gupta, A.K. and Ma, P.S. (1977). "Error in Eccentric Beam Formulation." *International Journal for Numerical Methods in Engineering*, Vol. 11, No. 9, pp 1473-1477.

Hays, C.O., Consolazio, G.R., Hoit, M.I., and Kakhandiki, A. (1994). "Bridge Rating of Girder – Slab Bridges Using Automated Finite Element Modeling Techniques." Structures Research Report No. 94-1, Department of Civil Engineering, University of Florida, May.

Hambly, E.C. (1991). Bridge Deck Behaviour, Second Edition, Chapman & Hall, London.

Kathol, S., Azizinamini, A., Luedke, J. (1995). "Strength Capacity of Steel Girder Bridges." NDOR Research Project No. RES1 90099, Nebraska Department of Roads, February.

Kennedy, J.B. and Grace, N.F. (1983). "Load Distribution in Continuous Composite Bridges." *Canadian Journal of Civil Engineering*, Vol. 10, pp 384-395.

Keogh, D.L. and O'Brien, E.J. (1996). "Recommendations on the Use of a 3-D Grillage Model for Bridge Deck Analysis." *Structural Engineering Review*, Vol. 8, No. 4, pp 357-366.

Khaleel, A.K. and Itani, R.Y. (1990). "Live-Load Moments for Continuous Skew Bridges." *Journal of Structural Engineering*, Vol. 116, No. 9, September, pp 2361-2373.

Kim, J. (2000). "Three-Dimensional Finite element Analysis of Multi-Layered Systems." Ph.D. Dissertation, Department of Civil Engineering, University of Illinois, Urbana-Champaign.

Kim, S. and Nowak, A.S. (1997). "Load Distribution and Impact Factors for I-Girder Bridges." *Journal of Bridge Engineering*, Vol. 2, No. 3, August, pp 97-104.

Mabsout, M.E., Tarhini, K.M., Frederick, G.R., and Tayar, C. (1997). "Finite-Element Analysis of Steel Girder Highway Bridges." *Journal of Bridge Engineering*, Vol. 2, No. 3, August, pp 83-87.

Mabsout, M.E., Tarhini, K.M., Frederick, G.R., and Kobrosly, M. (1997). "Influence of Sidewalks and Railings on Wheel Load Distribution in Steel Girder Bridges." *Journal of Bridge Engineering*, Vol. 2, No. 3, August, pp 88-96.

Mabsout, M.E., Tarhini, K.M., Frederick, G.R., and Kesserwan, A. (1998). "Effect of Continuity on Wheel Load Distribution in Steel Girder Bridges." *Journal of Bridge Engineering*, Vol. 3, No. 3, August, pp 103-110.

Marx, H.J. (1985). "Development of Design Criteria for Simply Supported Skew Slab-and-Girder Bridges." Ph.D. Dissertation, Department of Civil Engineering, University of Illinois, Urbana-Champaign.

Miller, R.E. (1980). "Reduction of the Error in Eccentric Beam Modeling." *International Journal for Numerical Methods in Engineering*, Vol. 15, pp 575-582.

Newmark, N.M. (1938). "A Distribution Procedure for the Analysis of Slabs Continuous Over Flexible Beams." Engineering Experiment Station, Bulletin 304, University of Illinois, Urbana, Illinois.

Nowak, A.S., Eom, J., Sanli, A., and Till, R. (1999). "Verification of Girder-Distribution Factors for Short-Span Steel Girder Bridges by Field Testing." *Transportation Research Record*, No. 1688, November, pp 62-67.

Prusty B.G. and Satsangi S.K. (2001). "Analysis of Stiffened Shell for Ships and Ocean Structures by Finite Element Method." *Ocean Engineering*, Vol. 28, No. 6, June, pp 621-638.

Sadek E.A. and Tawfik, S.A. (2000). "A Finite Element Model for the Analysis of Stiffened Laminated Plates." *Computers & Structures*, Vol. 75, No. 4, April, pp 369-383.

Schwarz, M. and Laman, J. (2001). "Response of Prestressed Concrete I-Girder Bridges to Live Load." *Journal of Bridge Engineering*, Vol. 6, No. 1, January/February, pp 1-8.

Shahawy, M. and Huang, D. (2001). "Analytical and Field Investigation of Lateral Load Distribution in Concrete Slab-On-Girder Bridges." *ACI Structural Journal*, Vol. 98, No. 4, July-August, pp 590-599.

Tabsh, S.W. and Tabatabai, M. (2001). "Live Load Distribution in Girder Bridges Subject to Oversized Trucks." *Journal of Bridge Engineering*, Vol. 6, No. 1, January/February, pp 9-16.

Tarhini, K.M., and Frederick, G.R. (1992). "Wheel load Distribution in I-Girder Highway Bridges." *Journal of Structural Engineering*, Vol. 8, No. 5, May, pp1285-1295.

Zokaie, T., Mish, K.D., and Imbsen, R.A. (1991). "Distribution of Wheel Loads on Highway Bridges." NCHRP 12-26/2 Final Report, National Cooperative Highway Research Program, Washington D.C.

Zokaie, T., Osterkamp, T.A., and Imbsen, R.A. (1991). “Distribution of Wheel Loads on Highway Bridges.” NCHRP 12-26/1 Final Report, National Cooperative Highway Research Program, Washington D.C.

Zokaie, T.(2000). “AASHTO-LRFD Live Load Distribution Specifications.” *Journal of Bridge Engineering*, Vol. 5, No. 2, May, pp 131-138.

APPENDIX A: PRE PROCESSOR CODE LIST

(prePro.for)

```
* ********************************************************************
*
*       Work Equivalent Nodal Force Calculation in ABAQUS format
*       <filename>.aba contains a loading block of ABAQUS input file.
*
*                             Wonseok Chung
*                        Purdue University
*                        Last modified Aug. 5, 2003
*
*   ELEMENT : S8R (thick only shell)
*
*   NOTE :  Default maximum number of truck(max) = 10
*           Number of element in longitidunal direction = 2000
*           Number of element in transverse direction = 200
*           Max. element number in deck = 400000
*
* WARNING !!!!!
*    output unit => kips, feet
* ********************************************************************
*
        PARAMETER (maxMesh1=200)
        PARAMETER (maxMesh2=2000)
* maxEle = maxMesh1*maxMesh2
        PARAMETER (maxEle=maxMesh1*maxMesh2)
        PARAMETER (maxTruck=10)
        PARAMETER (maxSpan=20)
        PARAMETER (nWheel=6)
** WARNING!! nShort and nLong should be "odd" number.
        PARAMETER (nShort=31)
        PARAMETER (nLong=31)
* DECLARATION of NEW VARIABLES from SUBROUTINE INPUT
        REAL span(maxSpan),widOH,widGir
        REAL sELSpan(maxSpan),spaceG,sWidOH,sWidGir,skew
        REAL xLoad(maxTruck),yLoad(maxTruck),trailer(maxTruck)
        REAL Es,Ec,PoissonS,PoissonC
        REAL area,strongI,weakI,TJ,h,thick,thickOHs,thickOHe
        REAL paraH,paraWid
        INTEGER nSpan,nElSpan(maxSpan),nGirder,nElG,nWidOH,nWidGir
        INTEGER nTotSpan,nTotWid
        INTEGER nTruck,direct(maxTruck)
        INTEGER nUnit
        INTEGER Nmat, Npara
* DECLARATION of NEW VARIABLES from SUBROUTINE HS20
        REAL xHS20(maxTruck,nWheel,nShort,nLong)
        REAL yHS20(maxTruck,nWheel,nShort,nLong)
* DECLARATION of NEW VARIABLES from SUBROUTINE meshGen
       INTEGER nElement(maxMesh1,maxMesh2)
       INTEGER nNode(maxEle,9)
* DECLARATION of NEW VARIABLES from SUBROUTINE loadElem
       INTEGER nXPos(maxTruck,nWheel,nShort,nLong)
       INTEGER nYPos(maxTruck,nWheel,nShort,nLong)
       REAL localX(maxTruck,nWheel,nShort,nLong)
       REAL localY(maxTruck,nWheel,nShort,nLong)
       REAL sElX(maxTruck,nWheel,nShort,nLong)
       REAL sElY(maxTruck,nWheel,nShort,nLong)
       INTEGER iElement(maxTruck,nWheel,nShort,nLong)
       INTEGER iNode(maxTruck,nWheel,nShort,nLong,9)
* DECLARATION of NEW VARIABLES from SUBROUTINE loadElem
        REAL N(maxTruck,nWheel,nShort,nLong,9)
* DECLARATION of NEW VARIABLES from SUBROUTINE enfCalc
        REAL enf(maxTruck,nWheel,nShort,nLong,9)
*
* Input data
*

      CALL INPUT(maxTruck,maxSpan,nSpan,span,widGir,widOH,
     +           nElSpan,nWidGir,nWidOH,sELSpan,sWidGir,sWidOH,
     +           nTotWid,nTotSpan,
     +           nGirder,spaceG,nElG,Ec,Es,PoissonC,PoissonS,
     +           area,strongI,weakI,TJ,h,thick,thickOHs,thickOHe,
     +           paraH, paraWid, Nmat,Npara,
     +           nTruck,direct,xLoad,yload,trailer,skew,nUnit)
*
* Open abaqus output file (enf.aba)
*
      OPEN(6,FILE='abaqus.inp',STATUS='REPLACE')
*
```

```
* Generate AASHTO HS 20 Truck Loading
*    xHS20(i), yHs20(i)
*
       CALL HS20(nTruck,nWheel,nShort,nLong,xLoad,yLoad,
     +             trailer,direct,xHS20,yHS20,skew)
*
* Generate element number and corresponding node number
*    nElement(number of tran element,number of long element)
*    nNode(nElement,4)
*
       CALL meshGen(nTotSpan,nTotWid,nElement,nNode)
*
* Identify loaded elements
*
       CALL loadElem(nSpan,nTruck,nWheel,span,
     +      nShort,nLong,widGir,widOH,sELSpan,sWidOH,sWidGir,
     +      nElSpan,nWidOH,nWidGir,nTotSpan,nTotWid,
     +     xHS20,yHS20,nElement,nNode,
     +      nXPos,nYPos,localX,localY,sElX,sElY,iElement,iNode)
*
*
* Calculate Shape function of the Kirchhoff plate element
*       N(i,j,k)
*  i = each truck
*  j = each HS20 wheel load
*  k = each dof (3 dof * 4 nodes)

       CALL shapeFun(nTruck,nWheel,nShort,nLong,
     +         localX,localY,sElX,sElY,N)
*
* Calculate Equivalent Nodal Force
*       enf(i,j,k)
*  i = each truck
*  j = each HS20 wheel load
*  k = each dof (3 dof * 4 nodes)
*
       CALL enfCalc(nTruck,nWheel,nShort,nLong,N,enf)
*
* File Output w/ ABAQUS format
*
       CALL OUTPUT(nSpan,span,widGir,widOH,nElSpan,nWidGir,nWidOH,sELSpan,sWidGir,
     +             sWidOH,nGirder,spaceG,nElG,Ec,Es,PoissonC,PoissonS,
     +             area,strongI,weakI,TJ,h,thick,thickOHs,thickOHe,
     +             paraH, paraWid, Nmat,Npara,skew,nUnit)
*
* Assemble equivalent nodal force to each node
*
       CALL assNout(nTruck,nWheel,nShort,nLong,nTotSpan,nTotWid,iNode,enf)
*
* File Output w/ ABAQUS format
*
       CALL OUTPUT2()
*
* Close the opened file
*
       CLOSE(6)
*
       write(*,*) '-----------------------------------------------------'
       write(*,*) 'The pre-processor has completed successfully.'
       write(*,*) 'You can find <abaqus.inp> file in the same directory '
       write(*,*) '-----------------------------------------------------'
*
       STOP
       END
```

```
* ************************************************************************
*                              SUBROUTINE HS20
*  In
*        maxTruck
*        nTruck: Number of Truck
*        xLoad(nTruck): x-co of the representative loading point for HS20 truck
*        yLoad(nTruck): y-co of the representative loading point for HS20 truck
*       trailer(nTruck): Variable length of trailer
*        direct(nTruck): Direction of the HS20 truck
*  Out
*        xHS20(nTruck,6,3,21): x-co of HS20 (each truck, each wheel)
*        yHS20(nTruck,6,3,21): y-co of HS20 (each truck, each wheel)
* ************************************************************************
*
       SUBROUTINE HS20 (nTruck,nWheel,nShort,nLong,xLoad,yLoad,
     +                  trailer,direct,xHS20,yHS20,skew)
*
* VARIABLE DECLARATION (IN)
*
       INTEGER nTruck, direct(nTruck)
       REAL xLoad(nTruck),yLoad(nTruck),trailer(nTruck),skew
*
* VARIABLE DECLARATION (OUT)
*
       REAL xHS20(nTruck,nWheel,nShort,nLong)
       REAL yHS20(nTruck,nWheel,nShort,nLong)
*
* ************************************************************************
*                                        |
*  HS 20 truck configuration             |
*                                        |
*                                        |
*  direct = 2                            |   direct = 1
*                                        |
*                                        |
*    o(2)      O(4)          O(6)        |     O(5)          O(3)      o(1)
*    |         |             |           |     |             |         |
*    x---------|-------------|           |     |-------------|---------x
*    |         |             |           |     |             |         |
*    o(1)      O(3)          O(5)        |     O(6)          O(4)      o(2)
*                                        |
*       ( ) : wheel number               |
*       x : representative point         |
*                                        |
*  PATCH LOADING                         |
*     at wheel number 1                  |
*                                        |
*       -------------------              |            --------------------
*      |  x 5|  x  |  x  |               |           |  x  |  x 6|  x 1|
*      |-----|-----|-----|               |           |-----|-----|-----|
*      |  x 4|  x  |  x  |               |           |  x  |  x  |  x 2|
*      |-----|-----|-----|               |           |-----|-----|-----|
*      |  x 3|  x  |  x  |               |           |  x  |  x  |  x 3|
*      |-----|-----|-----|               |           |-----|-----|-----|
*      |  x 2|  x  |  x  |               |           |  x  |  x  |  x 4|
*      |-----|-----|-----|               |           |-----|-----|-----|
*      |  x 1|  x 6|  x  |               |           |  x  |  x  |  x 5|
*      |-----|-----|-----|               |           |-----|-----|-----|
*
* ** WARNING : NShort and NLong should be odd number
* ** WARNING : Unit => inch !!
*
       do i=1,nTruck
         if(direct(i).EQ.1) then
               do j=1,nShort
                 do k=1,nLong
                       yHS20(i,1,j,k) = yLoad(i) + 36.
     +                                  -10./nLong*(k-(nLong/2+1))
                       yHS20(i,2,j,k) = yLoad(i) - 36.
     +                                  -10./nLong*(k-(nLong/2+1))
                       yHS20(i,3,j,k) = yLoad(i) + 36.
     +                                  -20./nLong*(k-(nLong/2+1))
                       yHS20(i,4,j,k) = yLoad(i) - 36.
     +                                  -20./nLong*(k-(nLong/2+1))
                       yHS20(i,5,j,k) = yHS20(i,3,j,k)
                       yHS20(i,6,j,k) = yHS20(i,4,j,k)
```

```
                xHS20(i,1,j,k) = xLoad(i)-yHS20(i,1,j,k)*tan(skew)
+                                      - 4./nShort*(j-(nShort-nShort/2))
                xHS20(i,2,j,k) = xLoad(i)-yHS20(i,2,j,k)*tan(skew)
+                                      - 4./nShort*(j-(nShort-nShort/2))
                xHS20(i,3,j,k) = xLoad(i)-yHS20(i,3,j,k)*tan(skew)-168.
+                                      - 8./nShort*(j-(nShort-nShort/2))
                xHS20(i,4,j,k) = xLoad(i)-yHS20(i,4,j,k)*tan(skew)-168.
+                                      - 8./nShort*(j-(nShort-nShort/2))
                xHS20(i,5,j,k) = xLoad(i)-yHS20(i,5,j,k)*tan(skew)-168.

+                                - trailer(i)- 8./nShort*(j-(nShort-nShort/2))

          enddo
        enddo
  else
        do j=1,nShort
          do k=1,nLong
                yHS20(i,1,j,k) = yLoad(i) - 36.
+                                      +10./nLong*(k-(nLong/2+1))
                yHS20(i,2,j,k) = yLoad(i) + 36.
+                                      +10./nLong*(k-(nLong/2+1))
                yHS20(i,3,j,k) = yLoad(i) - 36.
+                                      +20./nLong*(k-(nLong/2+1))
                yHS20(i,4,j,k) = yLoad(i) + 36.
+                                      +20./nLong*(k-(nLong/2+1))
                yHS20(i,5,j,k) = yHS20(i,3,j,k)
                yHS20(i,6,j,k) = yHS20(i,4,j,k)

                xHS20(i,1,j,k) = xLoad(i)-yHS20(i,1,j,k)*tan(skew)
+                                      + 4./nShort*(j-(nShort-nShort/2))
                xHS20(i,2,j,k) = xLoad(i)-yHS20(i,2,j,k)*tan(skew)
+                                      + 4./nShort*(j-(nShort-nShort/2))
                xHS20(i,3,j,k) = xLoad(i)-yHS20(i,3,j,k)*tan(skew)+168.
+                                      + 8./nShort*(j-(nShort-nShort/2))
                xHS20(i,4,j,k) = xLoad(i)-yHS20(i,4,j,k)*tan(skew)+168.
+                                      + 8./nShort*(j-(nShort-nShort/2))
                xHS20(i,5,j,k) = xLoad(i)-yHS20(i,5,j,k)*tan(skew)+168.
+                                + trailer(i)+ 8./nShort*(j-(nShort-nShort/2))
                xHS20(i,6,j,k) = xLoad(i)-yHS20(i,6,j,k)*tan(skew)+168.
+                                + trailer(i)+ 8./nShort*(j-(nShort-nShort/2))

          enddo
        enddo
  endif
enddo


return
end
```

```
* ************************************************************************
*                              SUBROUTINE meshGen
*  In
*       nTotSpan:      "total" number of element in long direction
*         nTotWid: "total" number of element in transverse direction
*  Out
*         nElement(nTotWid,nTotspan): array of element number
*       nNode(element#,nodeNumber): 4 node numbers correspond to element number
*
*  NOTE: In mesh design, the node number and element number scheme
*              should be same as followings.
*
*       - node number
*                29 __30__31___32__33___34___35
*                |    |    |    |    |    |    |
*                22|   23   24|  25  26|   27   28|
*                |    |    |    |    |    |    |
*                15|__16__17|__18__19|__20__21|
*                |    |    |    |    |    |    |
*                 8|    9   10|   11   12|   13   14|
*                |    |    |    |    |    |    |
*                 1|___2___3|___4___5|___6___7|
*
*
*       - element number
*                        _____________________
*                       |     |     |     |
*                       |  3  |  4  |
*                       |_____|_____|_____|__
*                       |     |     |     |
*                       |  1  |  2  |
*                       |_____|_____|_____|__
*
* ************************************************************************
*
       SUBROUTINE meshGen(nTotSpan,nTotWid,nElement,nNode)
*
* VARIABLE DECLARATION (IN)
*
       INTEGER nTotSpan,nTotWid
*
* VARIABLE DECLARATION (OUT)
*
       INTEGER nElement(nTotWid,nTotSpan),nNode(nTotWid*nTotSpan,9)
*
* Generate element number array
*
       do i=1,nTotWid
          do j=1,nTotSpan
               nElement(i,j) = j + (i-1)*nTotSpan
          enddo
       enddo
*
* Generate node number array
*
*      Definition of node number array
*
*                4 ___ 7___ 3
*                |     |    |
*                8|     9    |6    <= nElement
*                |     |    |
*                |_____|____|
*                1     5    2
*
       iVary=2*nTotSpan+1

       do i=1,nTotWid
          do j=1,nTotSpan
             nNode(nElement(i,j),1)= iVary*2*(i-1) + (2*j-1)
             nNode(nElement(i,j),2)= iVary*2*(i-1) + (2*j+1)
             nNode(nElement(i,j),3)= iVary*2*(i-1) + (iVary*2+2*j+1)
             nNode(nElement(i,j),4)= iVary*2*(i-1) + (iVary*2+2*j-1)
             nNode(nElement(i,j),5)= iVary*2*(i-1) + 2*j
             nNode(nElement(i,j),6)= iVary*2*(i-1) + (iVary+2*j+1)
             nNode(nElement(i,j),7)= iVary*2*(i-1) + (iVary*2+2*j)
             nNode(nElement(i,j),8)= iVary*2*(i-1) + (iVary+2*j-1)
             nNode(nElement(i,j),9)= iVary*2*(i-1) + (iVary+2*j)
          enddo
       enddo
```

```
       return
       end
* ************************************************************************
*                             SUBROUTINE loadElem
*  In
*        maxMesh: Maximum number of mesh matrix size(??)
*        maxTruck: maximum number of trucks
*        maxSpan: Maximum number of span
*        nSpan: Number of span
*        nTruck: Number of Truck
*        span(nSpan): Span length in ith span
*        nELSpan(nSpan): Number of element in ith span
*        sELSpan(nSpan): Size of element in ith span
*        widGir: Transverse width (girder part only)
*        widOH: Overhanging width
*        nWidGir: Number of element in girder transverse direction
*        nWidOH: Number of element in overhang
*        sWidGir: size of element in girder transverse direction
*        sWidOH: size of element in overhang
*        xHS20(nTruck,6,nShort,nLong): x-co of HS20 (each truck, each wheel)
*        yHS20(nTruck,6,nShort,nLong): y-co of HS20 (each truck, each wheel)
*        nElement(nTotWid,nTotspan): array of element number
*         nNode(element#,nodeNumber): 4 node numbers correspond to element number
*  Out
*        nXPos(nTruck,6,nShort,nLong): (N-1)th element in longitudinal(x) direction
*        nYPos(nTruck,6,nShort,nLong): (N-1)th element in transverse(y) direction
*        localX(nTruck,6,nShort,nLong): Local position in element natural (xi) direction
*         localY(nTruck,6,nShort,nLong): Local position in element transverse(eta)
direction
*        sElX(nTruck,6,nShort,nLong): Element length in the current span
*        sElY(nTruck,6,nShort,nLong):        Element width in the current span
*        iElement(nTruck,6,nShort,nLong): Loaded element number
*        iNode(nTruck,6,nShort,nLong,9): node number of the loaded element
* ************************************************************************

       SUBROUTINE loadElem(nSpan,nTruck,nWheel,span,nShort,nLong,widGir,widOH,
     +                     sELSpan,sWidOH,sWidGir,nElSpan,nWidOH,nWidGir,nTotSpan,
     +                     nTotWid,xHS20,yHS20,nElement,nNode,
     +                     nXPos,nYPos,localX,localY,sElX,sElY,iElement,iNode)
*
* VARIABLE DECLARATION (IN)
*
       REAL span(nSpan),widOH,widGir
       REAL sELSpan(nSpan),sWidOH,sWidGir
       INTEGER nTruck,nSpan,nElSpan(nSpan),nWidOH,nWidGir
       INTEGER nTotWid,nTotSpan
       REAL xHS20(nTruck,nWheel,nShort,nLong)
       REAL yHS20(nTruck,nWheel,nShort,nLong)
       INTEGER nElement(nTotWid,nTotSpan),nNode(nTotWid*nTotSpan,9)
*
* VARIABLE DECLARATION (OUT)
*
       INTEGER nXPos(nTruck,nWheel,nShort,nLong)
       INTEGER nYPos(nTruck,nWheel,nShort,nLong)
       REAL localX(nTruck,nWheel,nShort,nLong)
       REAL localY(nTruck,nWheel,nShort,nLong)
       REAL sElX(nTruck,nWheel,nShort,nLong)
       REAL sElY(nTruck,nWheel,nShort,nLong)
       INTEGER iElement(nTruck,nWheel,nShort,nLong)
       INTEGER iNode(nTruck,nWheel,nShort,nLong,9)
*
* Local Variables
*
       INTEGER nTotX
      REAL preSpan, totSpan
*
* Calc. total span length
*
      totSpan=0
      do i=1,nSpan
         totSpan = totSpan + span(i)
      enddo
*
* Find (N-1)th element & local position in longitudinal(x) direction
* Find the each element length in the current span
*
       do i=1,nTruck
```

```
          do j=1,nWheel
            do l=1,nShort
                  do m=1,nLong
                       preSpan = 0.
                       nTotX = 0
                       k=1
10                     continue
                       if(xHS20(i,j,l,m)/(span(k)+preSpan).GT.1.0) then
                           nTotX = nTotX + nELSpan(k)
                           preSpan = preSpan + span(k)
                           k=k+1
                           if(k.GT.nSpan) then
                               write(*,*) '**Warning in loadElem'
                           endif
                           goto 10
                       else
                           CALL quot(xHS20(i,j,l,m)-preSpan, sELSpan(k),
     +                                     nXPos(i,j,l,m),localX(i,j,l,m))
                           nXPos(i,j,l,m) = nXPos(i,j,l,m) + nTotx
                           localX(i,j,l,m) = localX(i,j,l,m) - sELSpan(k)/2.
                           sElX(i,j,l,m) = sELSpan(k)
                       endif

                       if(xHS20(i,j,l,m).GT.totSpan.OR.
     +                                 xHS20(i,j,l,m).LT.0.0) then
                           write(*,*) '**WARNING: X Loading is out of structure.'
                           pause
                       endif
                   enddo
                enddo
            enddo
          enddo

*
* Find (N-1)th element & local position in transverse(y) direction
* Find the each element width in the current span
*
          do i=1,nTruck
            do j=1,nWheel
              do l=1,nShort
                   do m=1,nLong
                      if(yHS20(i,j,l,m).LE.widOH) then
                          CALL quot(yHS20(i,j,l,m),sWidOH,nYPos(i,j,l,m),localY(i,j,l,m))
                          localY(i,j,l,m) = localY(i,j,l,m) - sWidOH/2.
                          sElY(i,j,l,m) = sWidOH
                      else if (yHS20(i,j,l,m).GT.widOH.AND.
     +                                    yHS20(i,j,l,m).LE.widOH+widGir) then
                          CALL quot(yHS20(i,j,l,m)-widOH,sWidGir,nYPos(i,j,l,m),
     +                                       localY(i,j,l,m))
                          nYPos(i,j,l,m) = nYPos(i,j,l,m) + nWidOH
                          localY(i,j,l,m) = localY(i,j,l,m) - sWidGir/2.
                          sElY(i,j,l,m) = sWidGir
                      else
                          CALL quot(yHS20(i,j,l,m)-(widOH+widGir), sWidOH,
     +                                     nYPos(i,j,l,m), localY(i,j,l,m))
                          nYPos(i,j,l,m) = nYPos(i,j,l,m) + (nWidOH+nWidGir)
                          localY(i,j,l,m) = localY(i,j,l,m) - sWidOH/2.
                          sElY(i,j,l,m) = sWidOH
                      endif

                       if(yHS20(i,j,l,m).GT.2*widOH+widGir.OR.
     +                                   yHS20(i,j,l,m).LT.0.0) then
                          write(*,*) '**WARNING: Y Loading is out of structure.'
                          pause
                       endif
                   enddo
               enddo
            enddo
          enddo


*
* Identify loading element
*
          do i=1,nTruck
            do j=1,nWheel
              do l=1,nShort
                   do m=1,nLong
```

```
              iElement(i,j,l,m)=nElement(nYPos(i,j,l,m)+1,nXPos(i,j,l,m)+1)
                enddo
              enddo
          enddo
        enddo
*
* Identify the node numbers of loading element
*

        do i=1,nTruck
         do j=1,nWheel
          do l=1,nShort
           do m=1,nLong
            do k=1,9
                  iNode(i,j,l,m,k) = nNode(iElement(i,j,l,m),k)
              enddo
           enddo
          enddo
         enddo
        enddo

        return
        end




* **********************************************************************
*                              SUBROUTINE shapeFun
*  In
*         maxTruck: maximum number of trucks
*         nTruck: Number of Truck
*         localX(nTruck,6,nShort,nLong): Local position in element natural (xi) direction
*        localY(nTruck,6,nShort,nLong): Local position in element transverse(eta)
direction
*         sElX(nTruck,6,nShort,nLong): Element length in the current span
*         sElY(nTruck,6,nShort,nLong): Element width in the current span
*  Out
*         N(nTruck,6,nShort,nLong,dof)
* **********************************************************************
        SUBROUTINE shapeFun(nTruck,nWheel,nShort,nLong,
        +                                    localX,localY,sElX,sElY,N)
*
* VARIABLE DECLARATION (IN)
*
        INTEGER nTruck
        REAL localX(nTruck,nWheel,nShort,nLong)
        REAL localY(nTruck,nWheel,nShort,nLong)
        REAL sElX(nTruck,nWheel,nShort,nLong)
        REAL sElY(nTruck,nWheel,nShort,nLong)
*
* VARIABLE DECLARATION (OUT)
*
        REAL N (nTruck,nWheel,nShort,nLong,9)
*
* Local Variables
*
        REAL xi,eta
*
* Shape Function of the 8-node Serendipity element
*  N(i,j,k)
*               i = number of truck
*               j = HS20 wheel number
*               k = dof number
*

        do i=1,nTruck
          do j=1,nWheel
            do l=1,nShort
              do m=1,nLong
                xi = localX(i,j,l,m)/(sElX(i,j,l,m)/2.)
                eta = localY(i,j,l,m)/(sElY(i,j,l,m)/2.)
                N(i,j,l,m,1) = -0.25*(1.-xi)*(1.-eta)*(xi+eta+1)
                N(i,j,l,m,2) = 0.25*(1.+xi)*(1.-eta)*(xi-eta-1)
                N(i,j,l,m,3) = 0.25*(1.+xi)*(1.+eta)*(xi+eta-1)
                N(i,j,l,m,4) = 0.25*(1.-xi)*(1.+eta)*(-xi+eta-1)
```

```
                N(i,j,l,m,5) = 0.5*(1.-eta)*(1.-xi**2)
                N(i,j,l,m,6) = 0.5*(1.+xi)*(1.-eta**2)
                N(i,j,l,m,7) = 0.5*(1.+eta)*(1.-xi**2)
                N(i,j,l,m,8) = 0.5*(1.-xi)*(1.-eta**2)
              enddo
            enddo
          enddo
        enddo

        return
        end


* **********************************************************************
*                              SUBROUTINE enfCalc
*  In
*         maxTruck: maximum number of trucks
*         nTruck: Number of Truck
*         N(nTruck,nWheel,nShort,nLong,dof)
*  Out
*         enf(nTruck,nWheel,nShort,nLong,dof)
* **********************************************************************
*
        SUBROUTINE enfCalc(nTruck,nWheel,nShort,nLong,N,enf)
*
* VARIABLE DECLARATION (IN)
*
        INTEGER nTruck
        REAL N(nTruck,nWheel,nShort,nLong,9)
*
* VARIABLE DECLARATION (OUT)
*
        REAL enf(nTruck,nWheel,nShort,nLong,9)
*
        do k=1,nTruck
         do i=1,nWheel
          do l=1,nShort
           do m=1,nLong
              do j=1,9
                  if(i.LE.2) then
                        enf(k,i,l,m,j) =  N(k,i,l,m,j)*4./(nShort*nLong)
                  else
                        enf(k,i,l,m,j) = N(k,i,l,m,j)*16./(nShort*nLong)
                  endif
              enddo
           enddo
          enddo
         enddo
        enddo

        return
        end



* **********************************************************************
*                              SUBROUTINE OUTPUT
*  In
*         maxTruck: maximum number of trucks
*         maxSpan: Maximum number of span
*         nSpan: Number of span
*         span(nSpan): Span length in ith span
*         nELSpan(nSpan): Number of element in ith span
*         sELSpan(nSpan): Size of element in ith span
*         nGirder: Number of girders
*         spaceG: Girder spacing
*         nElG: Number of elements between tow adjacent girder (even number)
*         widGir: Transverse width (girder part only)
*         widOH: Overhanging width
*         nWidGir: Number of element in girder transverse direction
*         nWidOH: Number of element in overhang
*         sWidGir: size of element in girder transverse direction
*         sWidOH: size of element in overhang
*
*         Ec: Concrete modulus
*         Es: Steel modulus
*         PoissonC: Poisson's ratio concrete
```

```
*         PoisoonS: Poisson's ratio steel
*
*         area: Sectional area of girder
*         strongI: Moment of inertia about strong axis
*         weakI: Moment of inertia about weak axis
*         TJ: Tortional rigidity
*         h: Girder height
*         thick: slab thickness
*         thickOHs: slab thickness of starting OH
*         thickOHe: slab thickness of edge
*
*         nTruck: Number of Truck
*         direct(nTruck): Direction of the HS20 truck
*         xLoad(nTruck): x-co of the representative loading point for HS20 truck
*         yLoad(nTruck): y-co of the representative loading point for HS20 truck
*          trailer(nTruck): Variable length of trailer
*         skew: skew angle (radian)
* ************************************************************************
*
        SUBROUTINE OUTPUT(nSpan,span,widGir,widOH,nElSpan,nWidGir,nWidOH,
     +                sELSpan,sWidGir,sWidOH,
     +                 nGirder,spaceG,nElG,Ec,Es,PoissonC,PoissonS,
     +                area,strongI,weakI,TJ,h,thick,thickOHs,thickOHe,
     +                 paraH, paraWid, Nmat,Npara,skew,nUnit)
*
* VARIABLE DECLARATION (IN)
*

        REAL span(nSpan),widOH,widGir
        REAL sELSpan(nSpan),spaceG,sWidOH,sWidGir,skew
        REAL Es,Ec,PoissonS,PoissonC
        REAL area,strongI,weakI,TJ,h,thick,thickOHs,thickOHe
        REAL paraH,paraWid
        INTEGER nSpan,nElSpan(nSpan),nGirder,nElG,nWidOH,nWidGir
        INTEGER nUnit
        INTEGER Nmat, Npara
*
* VARIABLE DECLARATION (Local)
*
        INTEGER ntmp1(nSpan+1),ntmp2(nSpan+1)
        INTEGER ntmp3(nSpan+1),ntmp4(nSpan+1)
        INTEGER kGtmp(nSpan+1,nGirder)
        INTEGER kStmp(nSpan+1,nGirder), kSEl(nSpan+1,nGirder)
        INTEGER kPAtmp(nSpan+1,nGirder)
        INTEGER nSpanTmp(nSpan+1)
        REAL xCor1(nSpan+1),xCor2(nSpan+1),xCor3(nSpan+1),xCor4(nSpan+1)
        REAL yCor1,yCor2,yCor3,yCor4
        REAL xGCor(nSpan+1,nGirder),yGCor(nSpan+1,nGirder)
        REAL xSCor(nSpan+1,nGirder),ySCor(nSpan+1,nGirder)
        REAL xPACor(nSpan+1,nGirder),yPACor(nSpan+1,nGirder)
*
* ---------------------------------------------------------------------
*                                                 Heading
* ---------------------------------------------------------------------
*
        write(6,10) '*HEADING'
        write(6,10) 'Automatic ABAQUS input generator'
*
* ---------------------------------------------------------------------
*                          MESH GENERATION (Bridge Slab)
* ---------------------------------------------------------------------
*
* *NODE part
*
        nTotSpan=0
        do i=1,nSpan
           nTotSpan = nTotSpan + nElSpan(i)
        enddo
*
* Node number
*
        iVary = nTotSpan*2 + 1
        nTmp=1
        do i=1,nSpan
           nTmp = nTmp + nElSpan(i)*2
           ntmp1(i) = nTmp - nElSpan(i)*2
           ntmp2(i) = ntmp1(i) + iVary*2*nWidOH
           ntmp3(i) = ntmp2(i) + iVary*2*nWidGir
```

```
          ntmp4(i) = ntmp3(i) + iVary*2*nWidOH
        enddo
        ntmp1(nSpan+1) = nTotSpan*2+1
        ntmp2(nSpan+1) = ntmp1(nSpan+1) + iVary*2*nWidOH
        ntmp3(nSpan+1) = ntmp2(nSpan+1) + iVary*2*nWidGir
        ntmp4(nSpan+1) = ntmp3(nSpan+1) + iVary*2*nWidOH
*
* X and Y coordinate
*
        yCor1 = 0.
        yCor2 = widOH
        yCor3 = widOH + widGir
        yCor4 = 2*widOH + widGir

        totSpan1=0.
        totSpan2=0.
        totSpan3=0.
        totSpan4=0.
        do i=1,nSpan
           totSpan1 = totSpan1 + span(i)
           totSpan2 = totSpan2 + span(i)
           totSpan3 = totSpan3 + span(i)
           totSpan4 = totSpan4 + span(i)
           xCor1(i) = totSpan1 - span(i) + yCor1*tan(skew)
           xCor2(i) = totSpan2 - span(i) + yCor2*tan(skew)
           xCor3(i) = totSpan3 - span(i) + yCor3*tan(skew)
           xCor4(i) = totSpan4 - span(i) + yCor4*tan(skew)
        enddo
        xCor1(nSpan+1) = totSpan1 + yCor1*tan(skew)
        xCor2(nSpan+1) = totSpan2 + yCor2*tan(skew)
        xCor3(nSpan+1) = totSpan3 + yCor3*tan(skew)
        xCor4(nSpan+1) = totSpan4 + yCor4*tan(skew)

        write(6,10) '** '
        write(6,10) '** ******************** Bridge Slab Nodal Coordinate'
        write(6,10) '** '
        write(6,10) '*NODE'
        do i=1,nSpan+1
           write(6,20) ntmp1(i),xCor1(i),yCor1
           write(6,20) ntmp2(i),xCor2(i),yCor2
           write(6,20) ntmp3(i),xCor3(i),yCor3
           write(6,20) ntmp4(i),xCor4(i),yCor4
        enddo
*
* *NGEN part
*
        write(6,30) '*NGEN', ' NSET=BOTTOM'
        do i=1,nSpan
           write(6,40) ntmp1(i),ntmp1(i+1)
        enddo

        write(6,30) '*NGEN', ' NSET=GIRDERS'
        do i=1,nSpan
           write(6,40) ntmp2(i),ntmp2(i+1)
        enddo
        write(6,60) '*NSET', ' NSET=GIRDERS1', 'GENERATE'
        do i=1,nSpan
           write(6,40) ntmp2(i),ntmp2(i+1)
        enddo

        write(6,30) '*NGEN', ' NSET=GIRDERL'
        do i=1,nSpan
           write(6,40) ntmp3(i),ntmp3(i+1)
        enddo
        write(6,60) '*NSET', ' NSET=GIRDERL1', 'GENERATE'
        do i=1,nSpan
           write(6,40) ntmp3(i),ntmp3(i+1)
        enddo

        write(6,30) '*NGEN', ' NSET=TOP'
        do i=1,nSpan
           write(6,40) ntmp4(i),ntmp4(i+1)
        enddo
*
* *NFILL part
*
        write(6,10) '*NFILL'
        write(6,50) 'BOTTOM','GIRDERS',nWidOH*2, iVary
```

```
       write(6,50) 'GIRDERS','GIRDERL',nWidGir*2, iVary
       write(6,50) 'GIRDERL','TOP',nWidOH*2, iVary
*
* *ELEMENT part
*
       write(6,60) '*ELEMENT', 'TYPE=S8R', 'ELSET=DECK'
       write(6,71) '1','1','3',iVary*2+3,iVary*2+1,'2',
      +        iVary+3,iVary*2+2,iVary+1

71     format(a,',',2x,a,',',2x,a,',',i5,',',i5,',',2x,a,',',
      +       i5,',',i5,',',i5)

*
* *ELGEN part
*
       write(6,30) '*ELGEN', 'ELSET=DECK'
       write(6,80) '1',nTotSpan,'2','1',2*nWidOH+nWidGir,iVary*2,nTotSpan
*
* ----------------------------------------------------------------------
*                         MESH GENERATION (Girders)
* ----------------------------------------------------------------------
*
* Node number
*
       nEndN1 = iVary * ( (nWidOH*2+nWidGir)*2+1 )
       nTmpEl = 0
       do i=1,nSpan
          nTmpEl=nTmpEl+nElSpan(i)*2
          do j=1,nGirder
             kGtmp(i,j) = (nEndN1+1)+ nTmpEl-nElSpan(i)*2 + (j-1)*iVary
          enddo
       enddo
       do j=1,nGirder
          kGtmp(nSpan+1,j) = (nEndN1+1)+ nTmpEl + (j-1)*iVary
       enddo
*
* X and Y coordinate
*
      totSpan=0.
       do i=1,nSpan
          totSpan = totSpan + span(i)
          do j=1,nGirder
             if(j.EQ.1) then
                yGCor(i,j) = widOH
             else if (j.EQ.nGirder) then
                yGCor(i,j) = widOH + widGir
             else
                yGCor(i,j) = widOH + spaceG*(j-1)
             endif
             xGCor(i,j) = totSpan - span(i) + yGCor(i,j)*tan(skew)
          enddo
       enddo
       do j=1,nGirder
          if(j.EQ.1) then
             yGCor(nSpan+1,j) = widOH
          else if (j.EQ.nGirder) then
             yGCor(nSpan+1,j) = widOH + widGir
          else
             yGCor(nSpan+1,j) = widOH + spaceG*(j-1)
          endif
          xGCor(nSpan+1,j) = totSpan + yGCor(nSpan+1,j)*tan(skew)
       enddo

       write(6,10) '** '
       write(6,10) '** ************************ Girder Nodal Coordinate'
       write(6,10) '** '
       write(6,10) '*NODE'
       do i=1,nSpan+1
          do j=1,nGirder
             write(6,25) kGtmp(i,j),xGCor(i,j),yGCor(i,j),-(thick/2+h/2.)
          enddo
       enddo
*
* *NGEN part
*
       do j=1,nGirder
          write(6,35) '*NGEN', ' NSET=GIRDER',j
          do i=1,nSpan
```

```
            write(6,40) kGTmp(i,j),kGTmp(i+1,j)
         enddo
      enddo
*
* *ELEMENT part
*
      nEndEl1 = nTotSpan*(2*nWidOH+nWidGir)
      write(6,60) '*ELEMENT', 'TYPE=B32', 'ELSET=GIRDER'
      write(6,75) nEndEl1+1,nEndN1+1,nEndN1+2,nEndN1+3

*
* *ELGEN part
*
      write(6,30) '*ELGEN', 'ELSET=GIRDER'
      write(6,85) nEndEl1+1,nTotSpan,'2','1',nGirder,
     +            iVary,nTotSpan


*
* ----------------------------------------------------------------
*                        MESH GENERATION (Support)
* ----------------------------------------------------------------
*
* Node number & coordinate
*
      nEndN2 = nEndN1 + iVary*nGirder

*
* Node number
*
      do i=1,nSpan+1
         do j=1,nGirder
            kStmp(i,j) = (nEndN2+j) + nGirder*(i-1)
         enddo
      enddo
*
* X and Y coordinate
*
     totSpan=0.
      do i=1,nSpan
         totSpan = totSpan + span(i)
         do j=1,nGirder
            if(j.EQ.1) then
               ySCor(i,j) = widOH
            else if (j.EQ.nGirder) then
               ySCor(i,j) = widOH + widGir
            else
               ySCor(i,j) = widOH + spaceG*(j-1)
            endif
            xSCor(i,j) = totSpan - span(i) + ySCor(i,j)*tan(skew)
         enddo
      enddo
      do j=1,nGirder
         if(j.EQ.1) then
            ySCor(nSpan+1,j) = widOH
         else if (j.EQ.nGirder) then
            ySCor(nSpan+1,j) = widOH + widGir
         else
            ySCor(nSpan+1,j) = widOH + spaceG*(j-1)
         endif
         xSCor(nSpan+1,j) = totSpan + ySCor(nSpan+1,j)*tan(skew)
      enddo
      write(6,10) '** '
      write(6,10) '** ********************** Support Nodal Coordinate'
      write(6,10) '** '
      write(6,10) '*NODE'
      do i=1,nSpan+1
         do j=1,nGirder
            write(6,25) kStmp(i,j),xSCor(i,j),ySCor(i,j),-(thick/2+h+2.)
         enddo
      enddo
*
* *ELEMENT part
*
      nEndEl2 = nEndEl1 + nTotSpan*nGirder
      do i=1,nSpan
         do j=1,nGirder
            kSEl(i,j)= (nEndEl2+j) + nGirder*(i-1)
```

```
         enddo
       enddo
       do j=1,nGirder
          kSEl(nSpan+1,j)= nEndEl2 + nGirder*nSpan +j
       enddo
       write(6,60) '*ELEMENT', 'TYPE=SPRING1', 'ELSET=SUPPORT'
       do i=1,nSpan+1
          do j=1,nGirder
             write(6,77) kSEl(i,j), kStmp(i,j)
          enddo
       enddo
77     format(i7,',',i7)
*
* ----------------------------------------------------------------------
*                         MESH GENERATION (Paraphet, OPTIONAL)
* ----------------------------------------------------------------------
*
*
       if(Npara.EQ.1) then
*
* Node number
*
       nEndN3 = nEndN2 + nGirder*(nSpan+1)
       nTmpEl = 0
       do i=1,nSpan
          nTmpEl=nTmpEl+nElSpan(i)*2
          do j=1,2
             kPAtmp(i,j) = (nEndN3+1)+ nTmpEl-nElSpan(i)*2 + (j-1)*iVary
          enddo
       enddo
       do j=1,2
          kPAtmp(nSpan+1,j) = (nEndN3+1)+ nTmpEl + (j-1)*iVary
       enddo
*
* X and Y coordinate
*
      totSpan=0.
       do i=1,nSpan
          totSpan = totSpan + span(i)
          do j=1,2
             if(j.EQ.1) then
                yPACor(i,j) = 0.
             else
                yPACor(i,j) = 2*widOH + widGir
             endif
             xPACor(i,j) = totSpan - span(i) + yPACor(i,j)*tan(skew)
          enddo
       enddo
       do j=1,2
          if(j.EQ.1) then
             yPACor(nSpan+1,j) = 0.
          else
             yPACor(nSpan+1,j) = 2.*widOH + widGir
          endif
          xPACor(nSpan+1,j) = totSpan + yPACor(nSpan+1,j)*tan(skew)
       enddo

       write(6,10) '** '
       write(6,10) '** ********************** Paraphet Nodal Coordinate'
       write(6,10) '** '
       write(6,10) '*NODE'
       do i=1,nSpan+1
          do j=1,2
             write(6,25) kPAtmp(i,j),xPACor(i,j),yPACor(i,j),
     +              (thick/2+paraH/2.)
          enddo
       enddo
*
* *NGEN part
*
       do j=1,2
          write(6,35) '*NGEN', ' NSET=PARAPHET',j
          do i=1,nSpan
             write(6,40) kPATmp(i,j),kPATmp(i+1,j)
          enddo
       enddo
*
* *ELEMENT part
```

```
*
       nEndEl3 = nEndEl2 + nGirder*(nSpan+1)
       write(6,60) '*ELEMENT', 'TYPE=B32', 'ELSET=PARAPHET'
       write(6,75) nEndEl3+1,nEndN3+1,nEndN3+2,nEndN3+3
*
* *ELGEN part
*
       write(6,30) '*ELGEN', 'ELSET=PARAPHET'
       write(6,88) nEndEl3+1,nTotSpan,'2','1','2',
     +             iVary,nTotSpan
88     format(i6,',',i6,',',1x,a,',',1x,a,',',a,',',1x,i6,',',i6)
*
       endif
*

*
* ---------------------------------------------------------------------
*                            MATERIAL & SECTION PROPERTIES
* ---------------------------------------------------------------------
*
* Material
*
       write(6,10) '** '
       write(6,10) '** **************************** Material Property'
       write(6,10) '** Concrete'
       write(6,10) '** '
       write(6,30) '*MATERIAL', 'NAME=CONC'
       write(6,30) '*ELASTIC', 'TYPE=ISO'
       write(6,31) Ec, PoissonC

       write(6,10) '** '
       write(6,10) '** Steel'
       write(6,10) '** '
       write(6,30) '*MATERIAL', 'NAME=STEEL'
       write(6,30) '*ELASTIC', 'TYPE=ISO'
       write(6,31) Es, PoissonS
31     format(f9.2,',',2x, f5.2)
*
* Shell section
*
       write(6,10) '** '
       write(6,10) '** ************************** Sectional Property'
       write(6,10) '** '
       write(6,10) '** Shell'
       write(6,10) '** '
* Variable thickness definition in OH
       write(6,10) '*NODAL THICKNESS '
       write(6,79)    'BOTTOM, ', thickOHe
       write(6,79)    'GIRDERS1, ', thickOHs
       write(6,30) '*NODAL THICKNESS ', 'GENERATE'
       write(6,81) 'BOTTOM, ', 'GIRDERS1, ', 2*nWidOH, iVary

       write(6,10) '*NODAL THICKNESS '
       write(6,79)    'GIRDERS, ', thick
       write(6,79)    'GIRDERL, ', thick
       write(6,30) '*NODAL THICKNESS ', 'GENERATE'
       write(6,81) 'GIRDERS, ', 'GIRDERL, ', 2*nWidGir, iVary

       write(6,10) '*NODAL THICKNESS '
       write(6,79)    'GIRDERL1, ', thickOHs
       write(6,79)    'TOP, ', thickOHe
       write(6,30) '*NODAL THICKNESS ', 'GENERATE'
       write(6,81) 'GIRDERL1, ', 'TOP, ', 2*nWidOH, iVary

       write(6,82) '*SHELL SECTION', 'NODAL THICKNESS ',
     +             'ELSET=DECK', 'MATERIAL=CONC'
79     format(a,1x,f6.3)
81     format(a,1x,a,1x,i5,',',2x,i9)
82     format(a,',',2x,a,',',2x,a,',',2x,a)
*
* Beam Section
*
       write(6,10) '**'
       write(6,10) '** Girder'
       write(6,10) '** '
       write(6,34) '*BEAM GENERAL SECTION','ELSET=GIRDER',
     +               'SECTION=GENERAL'
       write(6,30)    'DENSITY=0.', 'ZERO=0.'
```

```
       write(6,32) area, strongI, '0.0', weakI, TJ, '0.0', '0.0'
       write(6,60) '0.', '1.', '0.'
       write(6,33) Es, Es/(2*(1.+PoissonS)), '0.'
*
* Support section
*
       write(6,10) '** '
       write(6,10) '** Spring'
       write(6,10) '** '
       write(6,30) '*SPRING', 'ELSET=SUPPORT'
       write(6,10) '3,'
       write(6,10)    '200.,'
*
32     format(f7.2,',',f8.2,',',2x,a,',',2x,f8.2,',',f8.2,','2x,
     +        a,',',2x,a)
33     format(f15.2,',',f15.2,',',2x,a)
34     format(a,',',2x,a,',',2x,a,',')
36     format(f6.2,',',3x,a)
*
* Paraphet section (optional)
*
       if(Npara.EQ.1) then
         write(6,10) '**'
         write(6,10) '** Paraphet'
         write(6,10) '** '
         write(6,34) '*BEAM SECTION','ELSET=PARAPHET',
     +                'SECTION=RECT'
         if (nMat.EQ.1) then
            write(6,30)       'MATERIAL=CONC', 'POISSON=0.'
         else
            write(6,30)       'MATERIAL=STEEL', 'POISSON=0.'
         endif
         write(6,63) paraWid, paraH
         write(6,60) '0.', '1.', '0.'
       endif
63    format(f10.2,',', 2x,f10.2)
*
* ------------------------------------------------------------------
*                           MPC (Constraints)
* ------------------------------------------------------------------
*
* Support MPC
       write(6,10) '** '
       write(6,10) '** ************************* Multi Point Constraint'
       write(6,10) '** '
       write(6,10) '** Support MPC'
       write(6,10) '** '
       write(6,10) '*MPC '
       do i=1,nSpan
          nSpanTmp(1) = 0
          nSpanTmp(i+1) = nElSpan(i)
       enddo
       nTmp = 0
       do i=1,nSpan+1
       ntmp = ntmp + nSpanTmp(i)
          do j=1,nGirder
             write(6,100)'BEAM,',(iVary*2*nWidOH+1)+iVary*2*nElG*(j-1)
    +                         +nTmp*2, kStmp(i,j)
             write(6,100)'BEAM,',kGtmp(i,j), kStmp(i,j)
          enddo
       enddo
* Rest of MPC
       write(6,10) '** '
       write(6,10) '** Rest of MPC'
       write(6,10) '** '
       write(6,10) '*MPC '
       nTmp = 0
       do i=1,nSpan
       ntmp = ntmp + nElSpan(i)
          do j=1,nGirder
             do k=1,2*nElSpan(i)-1
        write(6,100)'BEAM,',(nEndN1+1)+iVary*(j-1)+(ntmp-nElSpan(i))*2+k,
    +   (iVary*2*nWidOH+1)+iVary*2*nElG*(j-1) + (ntmp-nElSpan(i))*2 + k
                enddo
          enddo
       enddo
* Paraphet MPC (optional)
       if (nPara.EQ.1) then
```

```
          write(6,10) '** '
          write(6,10) '** Paraphet MPC'
          write(6,10) '** '
          write(6,10) '*MPC '
          do j=1,2
             do k=1,iVary
                write(6,100)'BEAM,',nEndN3+iVary*(j-1)+k,
     +                      (nEndN1-iVary)*(j-1)+k
              enddo
          enddo
       endif
*
* ----------------------------------------------------------------
*                         Boundary Conditions
* ----------------------------------------------------------------
*
       write(6,10) '** '
       write(6,10) '** ***************************** Boundary Condition'
       write(6,10) '** '
* Bottom left (x,y,z restraint)
       write(6,10) '** bottom left node'
       write(6,10) '*BOUNDARY'
       write(6,51) nEndN2+1, '1,,', '0.'
       write(6,51) nEndN2+1, '2,,', '0.'
       write(6,51) nEndN2+1, '3,,', '0.'
* Rest of left end(x,z restraits)
       write(6,10) '** rest of left node'
       write(6,10) '*BOUNDARY'
       do i=1,nGirder-1
          write(6,51) (nEndN2+1)+i, '1,,', '0.'
          write(6,51) (nEndN2+1)+i, '3,,', '0.'
       enddo
* Rest of bottom (y,z restraint)
       write(6,10) '** rest of bottom supports'
       write(6,10) '*BOUNDARY'
       do i=1,nSpan
          write(6,51) (nEndN2+1)+nGirder*i, '2,,', '0.'
          write(6,51) (nEndN2+1)+nGirder*i, '3,,', '0.'
       enddo

* Rest of left
       write(6,10) '** rest of supports'
       write(6,10) '*BOUNDARY'
       do i=1,nSpan
          do j=1,nGirder-1
             write(6,51) (nEndN2+2)+nGirder*i+(j-1), '3,,', '0.'
          enddo
       enddo
*
* ----------------------------------------------------------------
*                         Step
* ----------------------------------------------------------------
*
       write(6,10) '** '
       write(6,10) '** ***************************************** Step'
       write(6,10) '** '
       write(6,30) '*STEP', 'PERTURBATION'
       write(6,10) '*STATIC '

*
* ----------------------------------------------------------------
*                         Loading
* ----------------------------------------------------------------
*
       write(6,10) '** '
       write(6,10) '** ********************************** Truck Load'
       write(6,10) '** '
       write(6,10) '*CLOAD'
*
* format sentences
*
10     format(a)
20     format(i6,',',F9.2,',',F8.2,',',3x,'0.0')
25     format(i6,',',F9.2,',',F8.2,',',3x,F6.2)
30     format(a,',',2x,a)
35     format(a,',',2x,a,i1)
40     format(i6,',',i6,',',2x,'1')
50     format(a,',',2x,a,',',2x,i6,',',2x,i6)
```

```
51      format(i7,',',2x,a,3x,a)
60      format(a,',',2x,a,',',2x,a)
70      format(a,',',2x,a,',',2x,a,',',i5,',',i5,',',2x,a,',',
        +       i5,',',i5,',',i5,',',i5)
75      format(i7,',',i7,',',i7,',',i7)

80      format(a,',',i6,',',1x,a,',',1x,a,',',i6,',',i6,',',i6)
85      format(i6,',',i6,',',1x,a,',',1x,a,',',i6,',',i6,',',i6)
100     format(a,2x,i7,',',2x,i7)
        return
      end
```

```
* ************************************************************************
*                              SUBROUTINE assemble
*  In
*        maxTruck: maximum number of trucks
*        nTruck: Number of Truck
*       nTotSpan:      "total" number of element in long direction
*        nTotWid: "total" number of element in transverse direction
*        enf(nTruck,6,nShort,nLong,dof)
*        iNode(nTruck,6,nShort,nLong,#of nodes)
*  Out
*        file output to <filename>.aba
* ************************************************************************
*
       SUBROUTINE assNout(nTruck,nWheel,nShort,nLong,
     +                    nTotSpan,nTotWid,iNode,enf)
*
* VARIABLE DECLARATION (IN)
*
       INTEGER nTruck,nTotSpan,nTotWid
       INTEGER iNode(nTruck,nWheel,nShort,nLong,9)
       REAL enf(nTruck,nWheel,nShort,nLong,9)
*
* Local Variables
*
       REAL Genf((2*nTotSpan+1)*(2*nTotWid+1))
       INTEGER nCode((2*nTotSpan+1)*(2*nTotWid+1))
       INTEGER line
       REAL temp
*
* Assemble enf in terms of nodal quantity
*

       do i=1, (2*nTotSpan+1)*(2*nTotWid+1)
         line=0
         temp = 0.
         do j=1,nTruck
            do k=1,nWheel
               do ll=1,nShort
                  do mm=1,nLong
                     do l=1,8
                        if(i.EQ.iNode(j,k,ll,mm,l)) then
                           line = 1
                           temp = temp - enf(j,k,ll,mm,l)
                        endif
                     enddo
                  enddo
               enddo
            enddo
         enddo
         Genf(i) = temp
         nCode(i) = line
       enddo

       do i=1,(2*nTotSpan+1)*(2*nTotWid+1)
          if(nCode(i).EQ.1) then
             write(6,157) i, 3, Genf(i)
          endif
       enddo

157   format(i5, 2x, ',', i3,3x, ',', f20.4)

       return
       end
```

```
* ************************************************************************
*                          Output Request
* ************************************************************************


       SUBROUTINE OUTPUT2()

       write(6,10) '** '
       write(6,10) '** ********************************* Output request'
       write(6,10) '** '
       write(6,30) '*NODE PRINT', 'FREQ=1'
       write(6,10) 'U,'
       write(6,30) '*NODE FILE', 'FREQ=1'
       write(6,10) 'U,'
       write(6,60) '*EL PRINT', 'POS=INTEG', 'FREQ=1'
       write(6,10) 'SF,'
       write(6,60) '*EL FILE', 'POS=INTEG', 'FREQ=1'
       write(6,10) 'SF,'
*
* End of step
*
       write(6,10) '** '
       write(6,10) '*END STEP'
*
* format sentences
*
10     format(a)
30     format(a,',',2x,a)
60     format(a,',',2x,a,',',2x,a)
       return
       end



*
* ********************************************************
* Subroutine quot provides "%" operator in JAVA
* ********************************************************
* Input: a, b
* Output: q = quotient
*              r = remainder
*
       SUBROUTINE quot(a,b,q,r)

       REAL a,b,r
       INTEGER q

       q=a/b
       r=a-q*b

       return
       end
```

APPENDIX B: POSTPROCESSOR MANUAL

Program Algorithm

Programming in the MATLAB Software is considered as the tool in this post-processing procedure. The post-processing program is written in MATLAB M-files. Four M-files involve in the determination of the section moment, which are 'main_envelop.m', 'GetBeam.m', 'GetShell.m', and 'Qd.m'. Figure A.1 displays the flowchart of post-processing procedure. The four M-files are described as follows:

a) main_envelop.m

This M-file is the main file that will call other sub-files. The bridge configuration is needed as the input. This main file will determine the location to calculate the section moment and invoke the sub-files, 'GetBeam.m' to get the force and moment in beam, and 'Qd.m & GetShell.m' to get the force and moment in shell. Then, the effective width and the neutral axis location of the section are determined in the sub-functions, 'Getbeff' and 'GetNA', respectively. The moment in the section is consecutively calculated by the method as described in the previous section.

a) GetBeam.m

This M-file is the sub-file that is called by 'main_envelop.m'. It is used to determine the force and moment in the beam element at the requested location using the one-dimensional spline interpolation.

b) GetShell.m

This M-file is the sub-file that is called by 'Qd.m'. It is used to determine the force and moment in the shell element at the requested location using the two-dimensional spline interpolation.

c) Qd.m

This M-file is the sub-file that is called by 'main_envelop.m'. It is used to determine the integration of force and moment in shell along the section within the

effective width because the shell element outputs are results as “per unit width”. This file is the modification of ‘Quad.m’ in the MATLAB tool box directory by mean to serve the need of the post-processing.

Program Restrictions and Limitations

The assumptions used in the post-processing are as follows:

1. The order of element number is the same as the format for the pre-processor. The element number is ascending from left to right (x-direction) and then bottom to top (y-direction).
2. Origin of coordinate (0,0,0) is at the lower left node of first shell element of reinforced concrete deck.

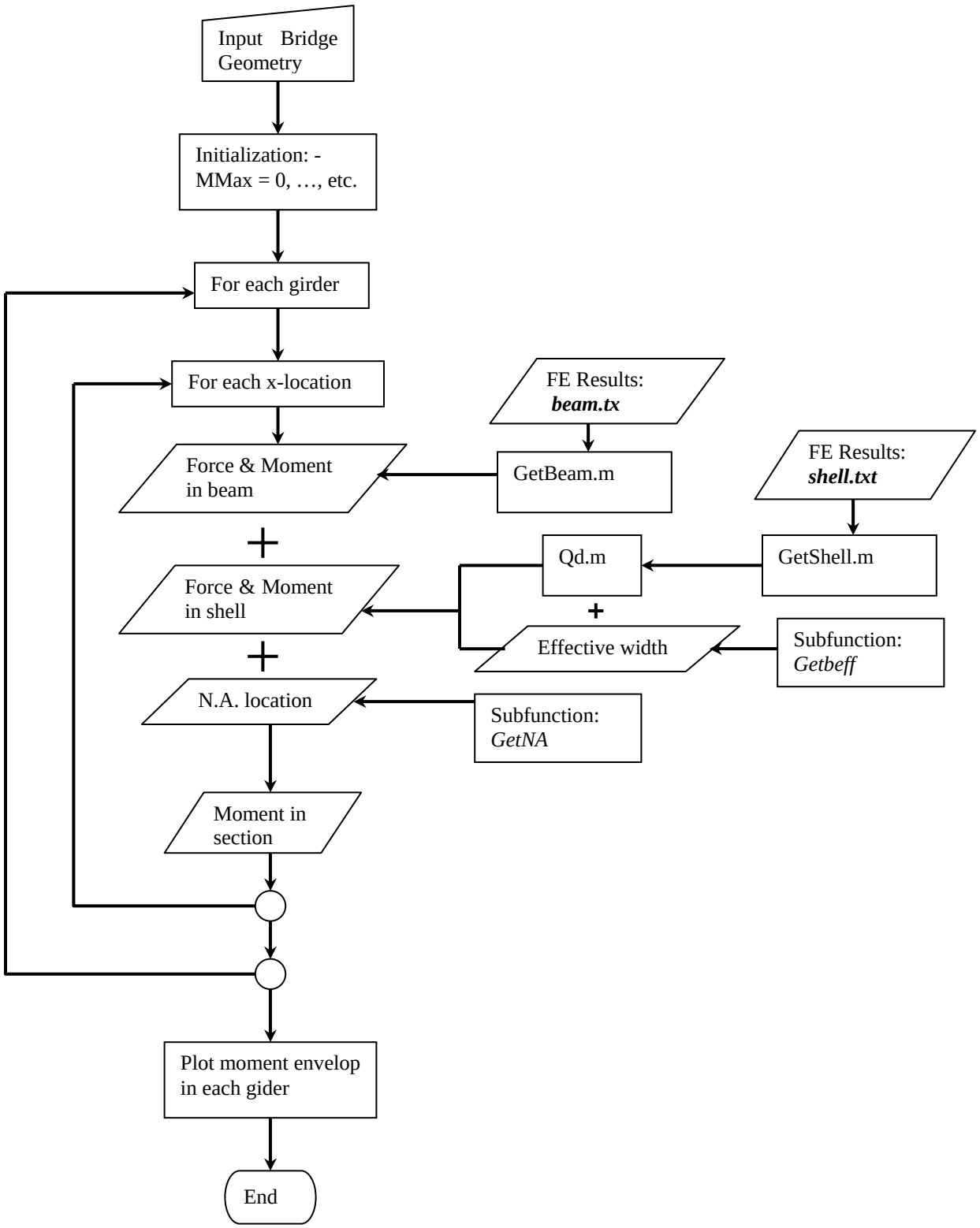


Figure A.1 Flowchart of the post-processor MATLAB program (moment_envelop.m).

Program Manual

The required inputs are as follows:

S	= Girder spacing (ft)	
widthOH	= Width of overhanging (ft), assume same for both side of bridge	
noGirder	= Total number of girders in bridge	
noElmtG	= Number of shell element between two adjacent girders	
noElmtOH	= Number of shell element in each overhanging	
noSpan	= Number of span in bridge	
lengthSpan	= Length of each span (ft)	(input as array format)
noElmtSpan	= Number of shell/beam element in each span	(input as array format)
skew	= Skew angle from transversal y-axis (degree)	
ts	= Slab thickness (in)	
tf	= Girder flange thickness (in)	
bf	= Girder flange width (in)	
tw	= Girder web thickness (in)	
bw	= Girder web width (in)	
n	= Ratio of steel to concrete modulus (=Es/Ec)	
lengthTotal	= Summation of all span lengths	
widthTotal	= Total width of bridge	
noElmtLong	= Total number of element in longitudinal direction (X-direction)	
noElmtTran	= Total number of element in transversal direction (Y-direction)	
interval	= Interval to display moment envelop in each span	
tol	= Tolerance in the adaptive quadrature integration for shell [default=10e-6]	
d	= Distance from support to exclude in the result (ft) [default=ts/2+2*tf+bw)/2/12]	
girder	= Girder number to be considered	(only needed in main_section.m)
xLocation	= Section location in X-coordinate (ft)	(only needed in main_section.m)
span	= Section location in which span	(only needed in main_section.m)

Program Procedure:

After running the finite element analysis for the specific loading locations, the ABAQUS result file is obtained as “filename.dat”. This result file is composed of shell and beam element outputs. In order to use the post-processor program, these outputs are required to extract into 2 files, “shell.txt” and “beam.txt”. Each file is in the format as described previous section (ABAQUS Result Format) and contains only numeric data, i.e., without text data. The bridge configuration is then required as the input into the

MATLAB M-files, “main_envelop.m”, “GetBeam.m” and “GetShell.m”. After running “main_envelop.m”, the results of the maximum and minimum moments and their corresponding locations are obtained in the MATLAB command window. The moment envelop of all girders are plotted in other windows. An example of the moment envelope is shown in Figure A.2.

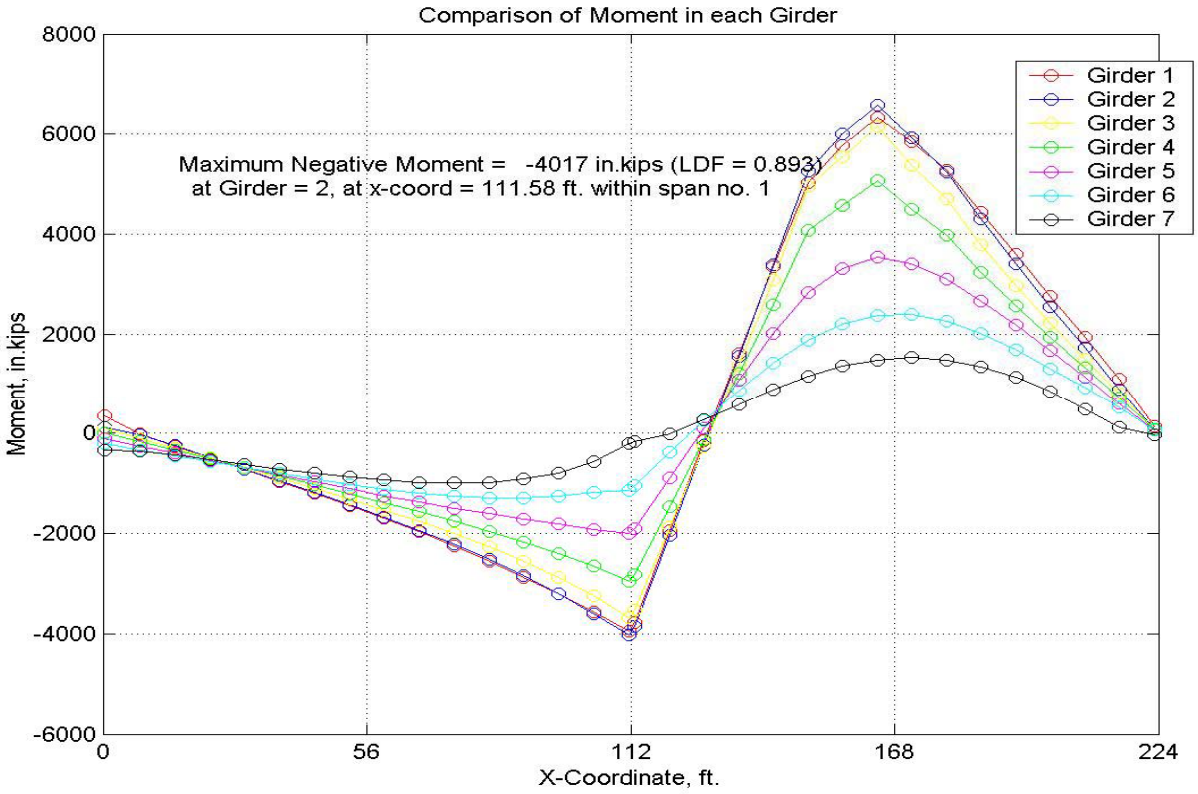


Figure A.2 Output example from post-processing (moment_envelop.m).

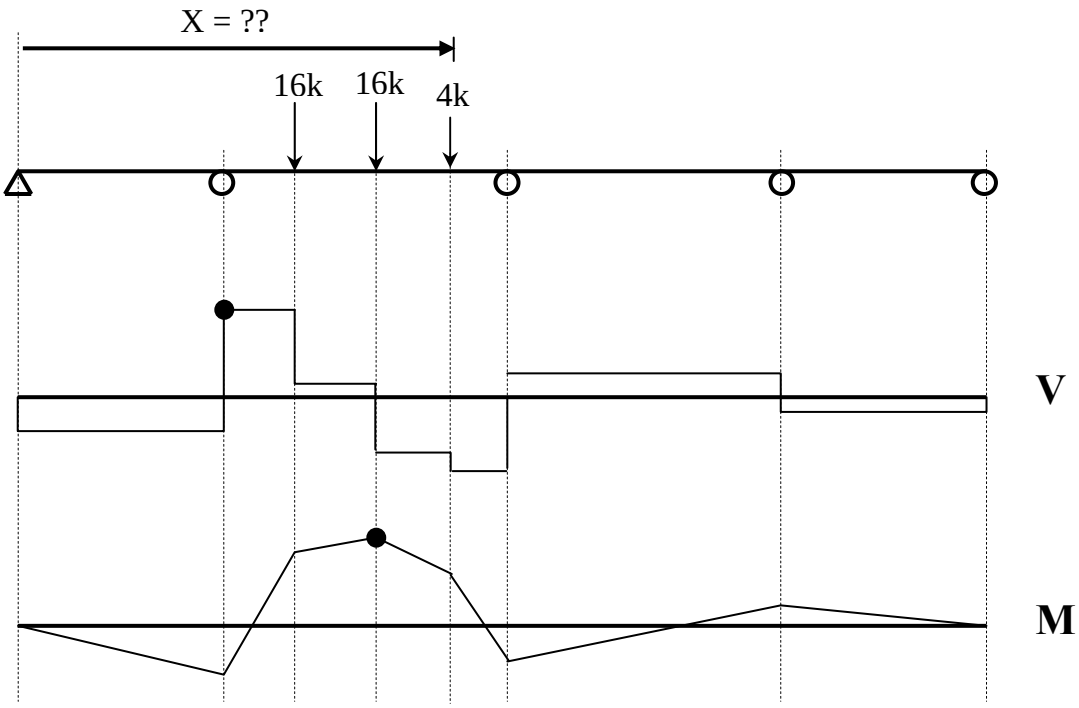


Figure A.3 Determination of the longitudinal truck position (loadposition.m).

Program list of `main_section.m`

```
function Moment_Section;
% Find section moment at specific 'girder', 'xLocation' and 'span'

% INPUT
% girder = 3;         % girder number to be considered
xLocation = 216.58;       % section location in X-coordinate (ft)
span = 2;            % section location in which span

S        = 8.67;     % girder spacing (ft)
widthOH  = 2.5;     % width of overhanging (ft), assume same for both side of bridge
noGirder = 4;       % total number of girders in bridge
noElmtG  = 4;       % number of shell element between two adjacent girders
noElmtOH = 1;       % number of shell element in each overhanging
noSpan     = 2;       % number of span in bridge
lengthSpan = [126,140]; % length of each span (ft)
noElmtSpan = [63,70]; % number of shell/beam element in each span
skew  = 24;          % skew angle from transversal y-axis (degree)
ts    = 8.0;        % slab thickness (in)
Ec    = 3182;       % Modulus of concrete (ksi)
Es    = 29000;      % Modulus of steel (ksi)
dd    = 46.5;      % girder thickness (in)
tw    = 0.4375;      % girder web thickness (in)
bf    = 20;     % girder flange width (in)
tf    = 1.75;      % girder flange thickness (in)

Mo    = 10846.9;        % Moment from 1-D analysis (in.k)
Vo    = 0;      % Shear from 1-D analysis (kips)

% Variable to output the results (can adjust)
GPlot = 0;          % girder in quesion (if want to plot all girder or have no idea,
enter 0;)
interval = [1,1];   % interval (between x1Plot and x2Plot) to display moment envelop in
each span
tol   = 10e-1;      % tolerance in the adaptive quadrature integration for shell
[default=10e-6]
% END INPUT


% Calculate data to be used in program
n = Es/Ec;
bw = dd-2*tf;        % girder web width (in)
lengthTotal = 0;
noElmtLong  = 0;
for i=1:noSpan
    lengthTotal = lengthTotal+lengthSpan(i);    % summation of all span lengths
    noElmtLong  = noElmtLong+noElmtSpan(i);     % total number of elmt in long. direction
(X-direction)
end;
widthTotal = 2*widthOH + (noGirder-1)*S;        % total width of bridge
noElmtTran = 2*noElmtOH + (noGirder-1)*noElmtG; % total number of elmt in trans.
direction (Y-direction)

% Open files 'beam.txt' and store them in array variables 'beam'
f1 = fopen('beam.txt','r');
beam = fscanf(f1,'%i %i %g %g %g %g %g %g',[8,inf]);  % "beamTemp" has 8 rows now.
beam = beam';                                  % Transpose to get the same format as
beam.txt
if (size(beam,1)~=(2*noElmtLong*noGirder))
    disp('Error in input or in beam.txt')
end;
fclose(f1);

% Find the X-coordinate of each beam data point
XB = 0;
index = 0;
temp = widthOH*tan(skew*pi/180);
for i=1:noSpan
    elmtSize = lengthSpan(i)/noElmtSpan(i);
    temp = temp+elmtSize/2*(1-1/sqrt(3));
    for j=1:noElmtSpan(i)
        index = index+1;
        XB(1,index) = temp;
        temp = temp+elmtSize/sqrt(3);
```

```
        index = index+1;
        XB(1,index) = temp;
        temp = temp+elmtSize*(1-1/sqrt(3));
    end;
    temp = temp-elmtSize/2*(1-1/sqrt(3));
end;
xDifferent = S*tan(skew*pi/180);
for i=2:noGirder
    for j=1:(2*noElmtLong)
        XB(i,j) = XB(i-1,j)+xDifferent;
    end;
end;

% Obtain beam data (SF1,SF2,SM1) in matrix form
for i=1:noGirder
    for j=1:(2*noElmtLong)
        index1 = noElmtLong*(i-1)*2+j;
        SF1B(i,j) = beam(index1,3);
        SM1B(i,j) = beam(index1,6);
        SF2B(i,j) = beam(index1,4);
    end;
end;

f2 = fopen('XB.txt','w');
f3 = fopen('SF1B.txt','w');
f4 = fopen('SM1B.txt','w');
f5 = fopen('SF2B.txt','w');
for i=1:noGirder
    fprintf(f2,'%g ',XB(i,:));
    fprintf(f2,'\n');
    fprintf(f3,'%g ',SF1B(i,:));
    fprintf(f3,'\n');
    fprintf(f4,'%g ',SM1B(i,:));
    fprintf(f4,'\n');
    fprintf(f5,'%g ',SF2B(i,:));
    fprintf(f5,'\n');
end;
fclose(f2);
fclose(f3);
fclose(f4);
fclose(f5);


% Open files 'shell.txt' and store them in array variables 'shell' and 'beam'
f6 = fopen('shell.txt','r');
shell = fscanf(f6,'%i %i %g %g %g %g %g %g %g %g',[10,inf]);   % "shell" has 8 rows now.
shell = shell';                                                % Transpose to get the same
format as shell.txt
if (size(shell,1)~=(4*noElmtLong*noElmtTran))
    disp('Error in input or in shell.txt')
end;
fclose(f6);

% Find the X- and Y- coordinate of each data point
xS = 0;
xindex = 0;
temp = 0;
for i=1:noSpan
    elmtSize = lengthSpan(i)/noElmtSpan(i);
    temp = temp+elmtSize/2*(1-1/sqrt(3));
    for j=1:noElmtSpan(i)
        xindex = xindex+1;
        xS(xindex) = temp;
        temp = temp+elmtSize/sqrt(3);
        xindex = xindex+1;
        xS(xindex) = temp;
        temp = temp+elmtSize*(1-1/sqrt(3));
    end;
    temp = temp-elmtSize/2*(1-1/sqrt(3));
end;

y = 0;
yindex = 0;
elmtSizeOH = widthOH/noElmtOH;
elmtSizeG = S/noElmtG;
temp = elmtSizeOH/2*(1-1/sqrt(3));
for i=1:noElmtOH
    yindex = yindex+1;
```

```
    yS(yindex) = temp;
    temp = temp+elmtSizeOH/sqrt(3);
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH*(1-1/sqrt(3));
end;
temp = temp-elmtSizeOH/2*(1-1/sqrt(3))+elmtSizeG/2*(1-1/sqrt(3));
for i=1:(noGirder-1)
    for j=1:noElmtG
        yindex = yindex+1;
        yS(yindex) = temp;
        temp = temp+elmtSizeG/sqrt(3);
        yindex = yindex+1;
        yS(yindex) = temp;
        temp = temp+elmtSizeG*(1-1/sqrt(3));
    end;
end;
temp = temp-elmtSizeG/2*(1-1/sqrt(3))+elmtSizeOH/2*(1-1/sqrt(3));
for i=1:noElmtOH
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH/sqrt(3);
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH*(1-1/sqrt(3));
end;

for i=1:yindex      % X and Y are matrices dimension '(2xnoElmtTran)*(2xnoElmtLong)'
    for j=1:xindex
        YS(i,j) = yS(i);
        XS(i,j) = xS(j)+yS(i)*tan(skew*pi/180);
    end;
end;

% Obtain shell data (SF1,SM1) in matrix form
for i=1:noElmtTran
    for j=1:noElmtLong
        index1 = noElmtLong*(i-1)+j;
        SF1S(2*i-1,2*j-1) = shell(4*index1-3,3);
        SF1S(2*i-1,2*j)   = shell(4*index1-2,3);
        SF1S(2*i  ,2*j-1) = shell(4*index1-1,3);
        SF1S(2*i  ,2*j)   = shell(4*index1  ,3);

        SF4S(2*i-1,2*j-1) = shell(4*index1-3,6);
        SF4S(2*i-1,2*j)   = shell(4*index1-2,6);
        SF4S(2*i  ,2*j-1) = shell(4*index1-1,6);
        SF4S(2*i  ,2*j)   = shell(4*index1  ,6);

        SM1S(2*i-1,2*j-1) = shell(4*index1-3,8);
        SM1S(2*i-1,2*j)   = shell(4*index1-2,8);
        SM1S(2*i  ,2*j-1) = shell(4*index1-1,8);
        SM1S(2*i  ,2*j)   = shell(4*index1  ,8);
    end;
end;

f7 = fopen('XS.txt','w');
f8 = fopen('YS.txt','w');
f9 = fopen('SF1S.txt','w');
f10 = fopen('SF4S.txt','w');
f11 = fopen('SM1S.txt','w');
for i=1:yindex
    fprintf(f7,'%g ',XS(i,:));
    fprintf(f7,'\n');
    fprintf(f8,'%g ',YS(i,:));
    fprintf(f8,'\n');
    fprintf(f9,'%g ',SF1S(i,:));
    fprintf(f9,'\n');
    fprintf(f10,'%g ',SF4S(i,:));
    fprintf(f10,'\n');
    fprintf(f11,'%g ',SM1S(i,:));
    fprintf(f11,'\n');
end;
fclose(f7);
fclose(f8);
fclose(f9);
fclose(f10);
fclose(f11);
```

```
% **** Begin MAIN PROGRAM **** %
if (Vo==0)
    fprintf('<Mo = %g in.k, xLocation = %g ft>\n',Mo,xLocation);
       for girder=1:noGirder
              % Get moment from beam and shell
              forceB  = -GetBeam(girder,xLocation,'SF1');
              momentB = -GetBeam(girder,xLocation,'SM1');
              [beff1,beff2] =
Getbeff(girder,noGirder,span,lengthSpan,S,ts,tw,bf,widthOH);
              yLocation = widthOH+(girder-1)*S;
              a = yLocation-beff1;
              b = yLocation+beff2;
              forceS  = -12*Qd(@GetShell,a,b,xLocation,'SF1',tol);
              momentS = -12*Qd(@GetShell,a,b,xLocation,'SM1',tol);
              % Moment calculation
              bEff = 12*(beff1+beff2);
              [eNA] = GetNA(tf,bf,tw,bw,bEff,ts,n);
              momentForce = -(forceB*eNA)+(forceS*(ts/2+tf+bw/2-eNA));
              mSection = momentB + momentS + momentForce;
        fprintf('Girder No. %i, moment = %7.1f in.k, M-LDF =
%5.3f\n',girder,mSection,mSection/Mo);
       end;
end
if (Mo==0)
    fprintf('<Vo = %g k, xLocation = %g ft>\n',Vo,xLocation);
       for girder=1:noGirder
              % Get shear from beam and shell
              shearB  = -GetBeam(girder,xLocation,'SF2');
              [beff1,beff2] =
Getbeff(girder,noGirder,span,lengthSpan,S,ts,tw,bf,widthOH);
              yLocation = widthOH+(girder-1)*S;
              a = yLocation-beff1;
              b = yLocation+beff2;
              shearS  = -12*Qd(@GetShell,a,b,xLocation,'SF4',tol);
              % Shear calculation
              bEff = 12*(beff1+beff2);
              [eNA] = GetNA(tf,bf,tw,bw,bEff,ts,n);
              vSection = shearB+shearS;
        fprintf('Girder No. %i, shear = %6.2f k(%6.2fB,%5.2fS), S-LDF =
%5.3f\n',girder,vSection,shearB,shearS,vSection/Vo);
       end;
end
% **** END MAIN PROGRAM **** %


function [beff1,beff2] = Getbeff(girder,noGirder,span,lengthSpan,S,ts,tw,bf,widthOH);
% Get the effective width of the girder section
a = lengthSpan(span)/4/2;
b1 = S/2;                          % for interior girder
b2 = widthOH;                      % for exterior girder
c = (12*ts+max(tw,bf/2))/2;
if (girder~=1)
    beff1 = min(a,b1);            % beff1 is eff. width in minus y-direction
    beff1 = min(beff1,c);
else
    beff1 = min(a,b2);
    beff1 = min(beff1,c);
end;
if (girder~=noGirder)
    beff2 = min(a,b1);            % beff2 is eff. width in plus y-direction
    beff2 = min(beff2,c);
else
    beff2 = min(a,b2);
    beff2 = min(beff2,c);
end;


function [eNA] = GetNA(tf,bf,tw,bw,bEff,ts,n);
% Get the nuetral axis location of the transformed section
area = (2*tf*bf)+(tw*bw)+(ts*bEff/n);
moment_area =
(tf*bf*tf/2)+(tw*bw*(tf+bw/2))+(tf*bf*(1.5*tf+bw))+(ts*bEff/n*(2*tf+bw+ts/2));
y = moment_area/area;
eNA = y-tf-bw/2;
```

Program list of moment_envelop.m

```
function Moment_Envelop;
% Determine the maximum moment and coresponding location, and draw moment envelop for all
girders

% INPUT
S        = 8.67;     % girder spacing (ft)
widthOH  = 2.5;     % width of overhanging (ft), assume same for both side of bridge
noGirder = 4;       % total number of girders in bridge
noElmtG  = 4;       % number of shell element between two adjacent girders
noElmtOH = 1;       % number of shell element in each overhanging
noSpan     = 2;       % number of span in bridge
lengthSpan = [126,140]; % length of each span (ft)
noElmtSpan = [63,70]; % number of shell/beam element in each span
skew  = 24;          % skew angle from transversal y-axis (degree)
ts    = 8.0;        % slab thickness (in)
Ec    = 3182;       % Modulus of concrete (ksi)
Es    = 29000;      % Modulus of steel (ksi)
dd    = 46.5;      % girder thickness (in)
tw    = 0.4375;      % girder web thickness (in)
bf    = 20;     % girder flange width (in)
tf    = 1.75;      % girder flange thickness (in)

Mo    =-6006.2;        % Moment from 1-D analysis (in.k)
Vo    = 0;      % Shear from 1-D analysis (kips)

% Variable to output the results (can adjust)
GPlot = 3;          % girder in quesion (if want to plot all girder or have no idea,
enter 0;)
interval = [1,1];   % interval (between x1Plot and x2Plot) to display moment envelop in
each span
tol   = 10e-1;      % tolerance in the adaptive quadrature integration for shell
[default=10e-6]
% END INPUT


% Calculate data to be used in program
n = Es/Ec;
bw = dd-2*tf;        % girder web width (in)
lengthTotal = 0;
noElmtLong  = 0;
for i=1:noSpan
    lengthTotal = lengthTotal+lengthSpan(i);    % summation of all span lengths
    noElmtLong  = noElmtLong+noElmtSpan(i);     % total number of elmt in long. direction
(X-direction)
end;
widthTotal = 2*widthOH + (noGirder-1)*S;        % total width of bridge
noElmtTran = 2*noElmtOH + (noGirder-1)*noElmtG; % total number of elmt in trans.
direction (Y-direction)

% Open files 'beam.txt' and store them in array variables 'beam'
f1 = fopen('beam.txt','r');
beam = fscanf(f1,'%i %i %g %g %g %g %g %g',[8,inf]);  % "beamTemp" has 8 rows now.
beam = beam';                                  % Transpose to get the same format as
beam.txt
if (size(beam,1)~=(2*noElmtLong*noGirder))
    disp('Error in input or in beam.txt')
end;
fclose(f1);

% Find the X-coordinate of each beam data point
XB = 0;
index = 0;
temp = widthOH*tan(skew*pi/180);
for i=1:noSpan
    elmtSize = lengthSpan(i)/noElmtSpan(i);
    temp = temp+elmtSize/2*(1-1/sqrt(3));
    for j=1:noElmtSpan(i)
        index = index+1;
        XB(1,index) = temp;
        temp = temp+elmtSize/sqrt(3);
        index = index+1;
        XB(1,index) = temp;
        temp = temp+elmtSize*(1-1/sqrt(3));
    end;
```

```
    temp = temp-elmtSize/2*(1-1/sqrt(3));
end;
xDifferent = S*tan(skew*pi/180);
for i=2:noGirder
    for j=1:(2*noElmtLong)
        XB(i,j) = XB(i-1,j)+xDifferent;
    end;
end;

% Obtain beam data (SF1,SF2,SM1) in matrix form
for i=1:noGirder
    for j=1:(2*noElmtLong)
        index1 = noElmtLong*(i-1)*2+j;
        SF1B(i,j) = beam(index1,3);
        SM1B(i,j) = beam(index1,6);
        SF2B(i,j) = beam(index1,4);
    end;
end;

f2 = fopen('XB.txt','w');
f3 = fopen('SF1B.txt','w');
f4 = fopen('SM1B.txt','w');
f5 = fopen('SF2B.txt','w');
for i=1:noGirder
    fprintf(f2,'%g ',XB(i,:));
    fprintf(f2,'\n');
    fprintf(f3,'%g ',SF1B(i,:));
    fprintf(f3,'\n');
    fprintf(f4,'%g ',SM1B(i,:));
    fprintf(f4,'\n');
    fprintf(f5,'%g ',SF2B(i,:));
    fprintf(f5,'\n');
end;
fclose(f2);
fclose(f3);
fclose(f4);
fclose(f5);


% Open files 'shell.txt' and store them in array variables 'shell' and 'beam'
f6 = fopen('shell.txt','r');
shell = fscanf(f6,'%i %i %g %g %g %g %g %g %g %g',[10,inf]);   % "shell" has 8 rows now.
shell = shell';                                                % Transpose to get the same
format as shell.txt
if (size(shell,1)~=(4*noElmtLong*noElmtTran))
    disp('Error in input or in shell.txt')
end;
fclose(f6);

% Find the X- and Y- coordinate of each data point
xS = 0;
xindex = 0;
temp = 0;
for i=1:noSpan
    elmtSize = lengthSpan(i)/noElmtSpan(i);
    temp = temp+elmtSize/2*(1-1/sqrt(3));
    for j=1:noElmtSpan(i)
        xindex = xindex+1;
        xS(xindex) = temp;
        temp = temp+elmtSize/sqrt(3);
        xindex = xindex+1;
        xS(xindex) = temp;
        temp = temp+elmtSize*(1-1/sqrt(3));
    end;
    temp = temp-elmtSize/2*(1-1/sqrt(3));
end;

y = 0;
yindex = 0;
elmtSizeOH = widthOH/noElmtOH;
elmtSizeG = S/noElmtG;
temp = elmtSizeOH/2*(1-1/sqrt(3));
for i=1:noElmtOH
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH/sqrt(3);
    yindex = yindex+1;
    yS(yindex) = temp;
```

```
    temp = temp+elmtSizeOH*(1-1/sqrt(3));
end;
temp = temp-elmtSizeOH/2*(1-1/sqrt(3))+elmtSizeG/2*(1-1/sqrt(3));
for i=1:(noGirder-1)
    for j=1:noElmtG
        yindex = yindex+1;
        yS(yindex) = temp;
        temp = temp+elmtSizeG/sqrt(3);
        yindex = yindex+1;
        yS(yindex) = temp;
        temp = temp+elmtSizeG*(1-1/sqrt(3));
    end;
end;
temp = temp-elmtSizeG/2*(1-1/sqrt(3))+elmtSizeOH/2*(1-1/sqrt(3));
for i=1:noElmtOH
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH/sqrt(3);
    yindex = yindex+1;
    yS(yindex) = temp;
    temp = temp+elmtSizeOH*(1-1/sqrt(3));
end;

for i=1:yindex      % X and Y are matrices dimension '(2xnoElmtTran)*(2xnoElmtLong)'
    for j=1:xindex
        YS(i,j) = yS(i);
        XS(i,j) = xS(j)+yS(i)*tan(skew*pi/180);
    end;
end;

% Obtain shell data (SF1,SM1) in matrix form
for i=1:noElmtTran
    for j=1:noElmtLong
        index1 = noElmtLong*(i-1)+j;
        SF1S(2*i-1,2*j-1) = shell(4*index1-3,3);
        SF1S(2*i-1,2*j)   = shell(4*index1-2,3);
        SF1S(2*i  ,2*j-1) = shell(4*index1-1,3);
        SF1S(2*i  ,2*j)   = shell(4*index1  ,3);

        SF4S(2*i-1,2*j-1) = shell(4*index1-3,6);
        SF4S(2*i-1,2*j)   = shell(4*index1-2,6);
        SF4S(2*i  ,2*j-1) = shell(4*index1-1,6);
        SF4S(2*i  ,2*j)   = shell(4*index1  ,6);

        SM1S(2*i-1,2*j-1) = shell(4*index1-3,8);
        SM1S(2*i-1,2*j)   = shell(4*index1-2,8);
        SM1S(2*i  ,2*j-1) = shell(4*index1-1,8);
        SM1S(2*i  ,2*j)   = shell(4*index1  ,8);
    end;
end;

f7 = fopen('XS.txt','w');
f8 = fopen('YS.txt','w');
f9 = fopen('SF1S.txt','w');
f10 = fopen('SF4S.txt','w');
f11 = fopen('SM1S.txt','w');
for i=1:yindex
    fprintf(f7,'%g ',XS(i,:));
    fprintf(f7,'\n');
    fprintf(f8,'%g ',YS(i,:));
    fprintf(f8,'\n');
    fprintf(f9,'%g ',SF1S(i,:));
    fprintf(f9,'\n');
    fprintf(f10,'%g ',SF4S(i,:));
    fprintf(f10,'\n');
    fprintf(f11,'%g ',SM1S(i,:));
    fprintf(f11,'\n');
end;
fclose(f7);
fclose(f8);
fclose(f9);
fclose(f10);
fclose(f11);


% **** Begin MAIN PROGRAM **** %
mMax = 0; mMin = 0;
gMax = 0; gMin = 0;
```

```
xMax = 0; xMin = 0;
spanMax = 0; spanMin = 0;
for i=1:noGirder
    d1(i) = S/2*tan(skew*pi/180);
    if (i==noGirder)
        d1(i) = widthOH*tan(skew*pi/180);
    end;
    d2(i) = S/2*tan(skew*pi/180);
    if (i==1)
        d2(i) = widthOH*tan(skew*pi/180);
    end;
end;

if (GPlot==0)
    girder1 = 1;
    girder2 = noGirder;
else
    girder1 = GPlot;
    girder2 = GPlot;
end;
for girder=girder1:girder2
    fprintf('Girder No.   : %i\n',girder);

    d1(girder) = S/2*tan(skew*pi/180);
    if (girder==noGirder)
        d1(girder) = widthOH*tan(skew*pi/180);
    end;
    d2(girder) = S/2*tan(skew*pi/180);
    if (girder==1)
        d2(girder) = widthOH*tan(skew*pi/180);
    end;

    index = 0;
    yLocation = widthOH+(girder-1)*S;
    xLocation = yLocation*tan(skew*pi/180);
    for span=1:noSpan
        fprintf('  Span       : %i\n',span);

        d1(girder) = d1(girder) + lengthSpan(span)/noElmtSpan(span)/2*(1-1/sqrt(3));
        d2(girder) = d2(girder) + lengthSpan(span)/noElmtSpan(span)/2*(1-1/sqrt(3));
%        d1(girder) = 0
%        d2(girder) = 0

        xLocation = xLocation + d1(girder);
        size = (lengthSpan(span)-d1(girder)-d2(girder))/interval(span);
        fprintf('    Interval :\n');
               [beff1,beff2] =
Getbeff(girder,noGirder,span,lengthSpan,S,ts,tw,bf,widthOH);
               a = yLocation-beff1;
               b = yLocation+beff2;
               bEff = 12*(beff1+beff2);
               [eNA] = GetNA(tf,bf,tw,bw,bEff,ts,n);
        for i=1:(interval(span)+1)
            fprintf('               %i\n',i);
            index = index+1;

%            if (i>=190 & span==1)
                               % Get moment from beam and shell
                               forceB  = -GetBeam(girder,xLocation,'SF1');
                               momentB = -GetBeam(girder,xLocation,'SM1');
                               forceS  = -12*Qd(@GetShell,a,b,xLocation,'SF1',tol);
                               momentS = -12*Qd(@GetShell,a,b,xLocation,'SM1',tol);
                               % Moment calculation
                               momentForce = -(forceB*eNA)+(forceS*(ts/2+tf+bw/2-eNA));
                               mSection = momentB + momentS + momentForce;
 %            else
 %                mSection = 0;
 %            end;

            if (mSection>mMax)
                mMax = mSection;  gMax = girder;
                xMax = xLocation; spanMax = span;
            end;
            if (mSection<mMin)
                mMin = mSection;  gMin = girder;
                xMin = xLocation; spanMin = span;
            end;
            x(girder,index) = xLocation;
```

```
            moment(girder,index) = mSection;
            xLocation = xLocation+size;
        end;
        xLocation = xLocation-size + d2(girder);
    end;
%    switch girder
%        case 1, string = 'ro';
%        case 2, string = 'bo';
%        case 3, string = 'yo';
%        case 4, string = 'go';
%        case 5, string = 'mo';
%        case 6, string = 'co';
%        otherwise, string = 'ko';
%    end;
%    figure(girder)
%    plot(x(girder,:),moment(girder,:),string,x(girder,:),moment(girder,:),'k-')
%    grid
%    xlabel('X-Coordinate, ft.')
%    ylabel('Moment, in.kips')
%    title(['Moment Diagram for Girder no. ',int2str(girder)])
end;
%figure(noGirder+1)
for girder=girder1:girder2
    switch girder
        case 1, string = 'ro-';
        case 2, string = 'bo-';
        case 3, string = 'yo-';
        case 4, string = 'go-';
        case 5, string = 'mo-';
        case 6, string = 'co-';
        otherwise, string = 'ko-';
    end;
    plot(x(girder,:),moment(girder,:),string)
    hold on
end;
grid
xlabel('X-Coordinate, ft.')
ylabel('Moment, in.kips')
title('Comparison of Moment in each Girder')
hold off
fprintf('\nMaximum Positive Moment = %7.0f in.kips (LDF = %5.3f)',mMax,mMax/Mo);
fprintf('\n  at Girder = %i\n  at x-coord = %5.2f ft. within span no.
%i',gMax,xMax,spanMax);
fprintf('\n             = %5.2f ft. from left support\n\n',xMax-((gMax-
1)*S+widthOH)*tan(skew/180*pi));
fprintf('\nMaximum Negative Moment = %7.0f in.kips (LDF = %5.3f)',mMin,mMin/Mo);
fprintf('\n  at Girder = %i\n  at x-coord = %5.2f ft. within span no.
%i',gMin,xMin,spanMin);
fprintf('\n             = %5.2f ft. from left support\n\n',xMin-((gMin-
1)*S+widthOH)*tan(skew/180*pi));
% **** End MAIN PROGRAM **** %



function [beff1,beff2] = Getbeff(girder,noGirder,span,lengthSpan,S,ts,tw,bf,widthOH);
% Get the effective width of the girder section
a = lengthSpan(span)/4/2;
b1 = S/2;                          % for interior girder
b2 = widthOH;                      % for exterior girder
c = (12*ts+max(tw,bf/2))/2;
if (girder~=1)
    beff1 = min(a,b1);            % beff1 is eff. width in minus y-direction
    beff1 = min(beff1,c);
else
    beff1 = min(a,b2);
    beff1 = min(beff1,c);
end;
if (girder~=noGirder)
    beff2 = min(a,b1);            % beff2 is eff. width in plus y-direction
    beff2 = min(beff2,c);
else
    beff2 = min(a,b2);
    beff2 = min(beff2,c);
end;


function [eNA] = GetNA(tf,bf,tw,bw,bEff,ts,n);
% Get the nuetral axis location of the transformed section
```

```
area = (2*tf*bf)+(tw*bw)+(ts*bEff/n);
moment_area =
(tf*bf*tf/2)+(tw*bw*(tf+bw/2))+(tf*bf*(1.5*tf+bw))+(ts*bEff/n*(2*tf+bw+ts/2));
y = moment_area/area;
eNA = y-tf-bw/2;
```

Program list of Qd.m

```
function [Q,fcnt] = Qd(funfcn,a,b,x,string,tol,trace,varargin)
% Subfuction called from 'main_envelop.m' or 'main_section.m'

%************************IMPORTANT
NOTE*****************************************************
% This M-file is modified from M-file 'quad' in
D:\MATLAB6p1\toolbox\MATLAB\funfun\quad.m
% It includes variable, 'x' for the use in moment_envelop ABAQUSin Post-processing
% Also, string is additional variable indicate 'SF1' (force) or 'SM1' (moment)
% Example:
%    Q = Qd(@myfun,0,2,10,'SF1');
%    where myfun.m is an M-file:
%          function z = myfun(x,y)
%          z = y./(x.^3-2*x-5);
%*******************************************************************************
****

%QUAD   Numerically evaluate integral, adaptive Simpson quadrature.
%   Q = QUAD(FUN,A,B) tries to approximate the integral of function
%   FUN from A to B to within an error of 1.e-6 using recursive
%   adaptive Simpson quadrature.  The function Y = FUN(X) should
%   accept a vector argument X and return a vector result Y, the
%   integrand evaluated at each element of X.
%
%   Q = QUAD(FUN,A,B,TOL) uses an absolute error tolerance of TOL
%   instead of the default, which is 1.e-6.  Larger values of TOL
%   result in fewer function evaluations and faster computation,
%   but less accurate results.  The QUAD function in MATLAB 5.3 used
%   a less reliable algorithm and a default tolerance of 1.e-3.
%
%   [Q,FCNT] = QUAD(...) returns the number of function evaluations.
%
%   QUAD(FUN,A,B,TOL,TRACE) with non-zero TRACE shows the values
%   of [fcnt a b-a Q] during the recursion.
%
%   QUAD(FUN,A,B,TOL,TRACE,P1,P2,...) provides for additional
%   arguments P1, P2, ... to be passed directly to function FUN,
%   FUN(X,P1,P2,...).  Pass empty matrices for TOL or TRACE to
%   use the default values.
%
%   Use array operators .*, ./ and .^ in the definition of FUN
%   so that it can be evaluated with a vector argument.
%
%   Function QUADL may be more efficient with high accuracies
%   and smooth integrands.
%
%   Example:
%       FUN can be specified three different ways.
%
%       A string expression involving a single variable:
%          Q = quad('1./(x.^3-2*x-5)',0,2);
%
%       An inline object:
%          F = inline('1./(x.^3-2*x-5)');
%          Q = quad(F,0,2);
%
%       A function handle:
%          Q = quad(@myfun,0,2);
%          where myfun.m is an M-file:
%             function y = myfun(x)
%             y = 1./(x.^3-2*x-5);
%
%   See also QUADL, DBLQUAD, INLINE, @.

%   Based on "adaptsim" by Walter Gander.
%   Ref: W. Gander and W. Gautschi, "Adaptive Quadrature Revisited", 1998.
%   http://www.inf.ethz.ch/personal/gander
%   Copyright 1984-2001 The MathWorks, Inc.
%   $Revision: 5.22 $  $Date: 2001/04/15 11:59:20 $

f = fcnchk(funfcn);
if nargin < 6 | isempty(tol), tol = 1.e-6; end;
if nargin < 7 | isempty(trace), trace = 0; end;
```

```
% Initialize with three unequal subintervals.
h = 0.13579*(b-a);
y = [a a+h a+2*h (a+b)/2 b-2*h b-h b];
for i=1:7
    z(i) = feval(f, y(i), x,string, varargin{:});
end;
fcnt = 7;

% Fudge endpoints to avoid infinities.
if ~isfinite(z(1))
   z(1) = feval(f,a+eps*(b-a),x,string,varargin{:});
   fcnt = fcnt+1;
end
if ~isfinite(z(7))
   z(7) = feval(f,b-eps*(b-a),x,string,varargin{:});
   fcnt = fcnt+1;
end

% Call the recursive core integrator.
hmin = eps/1024*abs(b-a);
[Q(1),fcnt,warn(1)] = ...
   quadstep(f,y(1),y(3),x,string,z(1),z(2),z(3),tol,trace,fcnt,hmin,varargin{:});
[Q(2),fcnt,warn(2)] = ...
   quadstep(f,y(3),y(5),x,string,z(3),z(4),z(5),tol,trace,fcnt,hmin,varargin{:});
[Q(3),fcnt,warn(3)] = ...
   quadstep(f,y(5),y(7),x,string,z(5),z(6),z(7),tol,trace,fcnt,hmin,varargin{:});
Q = sum(Q);
warn = max(warn);

switch warn
   case 1
      warning('Minimum step size reached; singularity possible.')
   case 2
      warning('Maximum function count exceeded; singularity likely.')
   case 3
      warning('Infinite or Not-a-Number function value encountered.')
   otherwise
      % No warning.
end

% ------------------------------------------------------------------------

function [Q,fcnt,warn] = quadstep
(f,a,b,x,string,fa,fc,fb,tol,trace,fcnt,hmin,varargin)
%QUADSTEP  Recursive core routine for function QUAD.

maxfcnt = 10000;

% Evaluate integrand twice in interior of subinterval [a,b].
h = b - a;
c = (a + b)/2;
if abs(h) < hmin | c == a | c == b
   % Minimum step size reached; singularity possible.
   Q = h*fc;
   warn = 1;
   return
end
y = [(a + c)/2, (c + b)/2];
for i=1:2
    z(i) = feval(f, y(i), x, string,varargin{:});
end;
fcnt = fcnt + 2;
if fcnt > maxfcnt
   % Maximum function count exceeded; singularity likely.
   Q = h*fc;
   warn = 2;
   return
end
fd = z(1);
fe = z(2);

% Three point Simpson's rule.
Q1 = (h/6)*(fa + 4*fc + fb);

% Five point double Simpson's rule.
Q2 = (h/12)*(fa + 4*fd + 2*fc + 4*fe + fb);

% One step of Romberg extrapolation.
```

```
Q = Q2 + (Q2 - Q1)/15;

if ~isfinite(Q)
   % Infinite or Not-a-Number function value encountered.
   warn = 3;
   return
end
if trace
   disp(sprintf('%8.0f %16.10f %18.8e %16.10f',fcnt,a,h,Q))
end

% Check accuracy of integral over this subinterval.
if abs(Q2 - Q) <= tol
   warn = 0;
   return

% Subdivide into two subintervals.
else
   [Qac,fcnt,warnac] =
quadstep(f,a,c,x,string,fa,fd,fc,tol,trace,fcnt,hmin,varargin{:});
   [Qcb,fcnt,warncb] =
quadstep(f,c,b,x,string,fc,fe,fb,tol,trace,fcnt,hmin,varargin{:});
   Q = Qac + Qcb;
   warn = max(warnac,warncb);
end
```

Program List of **GetBeam.m**

```
function Data = GetBeam(girderi,xi,string);
% Subfuction called from 'main_envelop.m' or 'main_section.m'
% This function can also be used alone to plot beam data

% INPUT
S        = 8.67;     % girder spacing (ft)
widthOH  = 2.5;     % width of overhanging (ft), assume same for both side of bridge
noGirder = 4;       % total number of girders in bridge
noElmtG  = 4;       % number of shell element between two adjacent girders
noElmtOH = 1;       % number of shell element in each overhanging
noSpan     = 2;       % number of span in bridge
lengthSpan = [126,140]; % length of each span (ft)
noElmtSpan = [63,70]; % number of shell/beam element in each span
skew  = 24;          % skew angle from transversal y-axis (degree)
ts    = 8.0;        % slab thickness (in)
Ec    = 3182;       % Modulus of concrete (ksi)
Es    = 29000;      % Modulus of steel (ksi)
dd    = 46.5;      % girder thickness (in)
tw    = 0.4375;      % girder web thickness (in)
bf    = 20;     % girder flange width (in)
tf    = 1.75;      % girder flange thickness (in)

Mo    = 10846.9;        % Moment from 1-D analysis (in.k)
Vo    = 0;      % Shear from 1-D analysis (kips)

% Variable to output the results (can adjust)
GPlot = 0;          % girder in quesion (if want to plot all girder or have no idea,
enter 0;)
interval = [1,1];   % interval (between x1Plot and x2Plot) to display moment envelop in
each span
tol   = 10e-1;      % tolerance in the adaptive quadrature integration for shell
[default=10e-6]
% END INPUT


% Calculate data to be used in program
n = Es/Ec;
bw = dd-2*tf;        % girder web width (in)
lengthTotal = 0;
noElmtLong  = 0;
for i=1:noSpan
    lengthTotal = lengthTotal+lengthSpan(i);    % summation of all span lengths
    noElmtLong  = noElmtLong+noElmtSpan(i);     % total number of elmt in long. direction
(X-direction)
end;
widthTotal = 2*widthOH + (noGirder-1)*S;        % total width of bridge
noElmtTran = 2*noElmtOH + (noGirder-1)*noElmtG; % total number of elmt in trans.
direction (Y-direction)

%Open necessary files and interpolate to obtain the acquired data
f1 = fopen('XB.txt','r');
XB = fscanf(f1,'%g',[2*noElmtLong,noGirder]);
XB = XB';
fclose(f1);

switch string
    case 'SF1', str = 'SF1B.txt';
    case 'SM1', str = 'SM1B.txt';
    case 'SF2', str = 'SF2B.txt';
    otherwise, disp('Unknown Required Data in Shell')
end;
f3 = fopen(str,'r');
DataB = fscanf(f3,'%g',[2*noElmtLong,noGirder]);
DataB = DataB';
fclose(f3);

XB(girderi,:);
DataB(girderi,:);
Data = interp1(XB(girderi,:),DataB(girderi,:),xi,'spline');

%% Plot the spline interpolation graph of beam SF1 or SM1 for all girders
%for girder=1:noGirder
%       Dat = 0;
%       for i=1:noElmtLong
%         index1 = noElmtLong*(girder-1)+i;
```

```
%        Dat(i) = beam(index1,index2);
%    end;
%    xtemp = 1.75:.25:54.25;
%       Datatemp = interp1(x,Dat,xtemp,'spline');
%       figure(girder)
%    plot(x,Dat,'o',xtemp,Datatemp);
%       grid
%       xlabel('X-Coordinate, ft.')
%       switch string
%    case 'SF1',
%        ylabel('SF1 in beam, in.kips')
%              title(['Spline Interpolation of SF1 in beam for girder
no.',int2str(girder)])
%    case 'SM1',
%        ylabel('SM1 in beam, in.kips')
%              title(['Spline Interpolation of SM1 in beam for girder
no.',int2str(girder)])
%    otherwise, disp('Unknown Required Data in Beam')
%    end;
%end;
```

Program List of `GetShell.m`

```
function Data = GetShell(yi,xi,string);
% Subfuction called from 'main_envelop.m' or 'main_section.m'
% This function can also be used alone to plot shell data

% INPUT
S        = 8.67;     % girder spacing (ft)
widthOH  = 2.5;     % width of overhanging (ft), assume same for both side of bridge
noGirder = 4;       % total number of girders in bridge
noElmtG  = 4;       % number of shell element between two adjacent girders
noElmtOH = 1;       % number of shell element in each overhanging
noSpan     = 2;       % number of span in bridge
lengthSpan = [126,140]; % length of each span (ft)
noElmtSpan = [63,70]; % number of shell/beam element in each span
skew  = 24;          % skew angle from transversal y-axis (degree)
ts    = 8.0;        % slab thickness (in)
Ec    = 3182;       % Modulus of concrete (ksi)
Es    = 29000;      % Modulus of steel (ksi)
dd    = 46.5;      % girder thickness (in)
tw    = 0.4375;       % girder web thickness (in)
bf    = 20;     % girder flange width (in)
tf    = 1.75;      % girder flange thickness (in)

Mo    = 10846.9;        % Moment from 1-D analysis (in.k)
Vo    = 0;      % Shear from 1-D analysis (kips)

% Variable to output the results (can adjust)
GPlot = 0;          % girder in quesion (if want to plot all girder or have no idea,
enter 0;)
interval = [1,1];   % interval (between x1Plot and x2Plot) to display moment envelop in
each span
tol   = 10e-1;      % tolerance in the adaptive quadrature integration for shell
[default=10e-6]
% END INPUT


% Calculate data to be used in program
n = Es/Ec;
bw = dd-2*tf;        % girder web width (in)
lengthTotal = 0;
noElmtLong  = 0;
for i=1:noSpan
    lengthTotal = lengthTotal+lengthSpan(i);    % summation of all span lengths
    noElmtLong  = noElmtLong+noElmtSpan(i);     % total number of elmt in long. direction
(X-direction)
end;
widthTotal = 2*widthOH + (noGirder-1)*S;        % total width of bridge
noElmtTran = 2*noElmtOH + (noGirder-1)*noElmtG; % total number of elmt in trans.
direction (Y-direction)

%Open necessary files and interpolate to obtain the acquired data
f1 = fopen('XS.txt','r');
XS = fscanf(f1,'%g',[2*noElmtLong,2*noElmtTran]);
XS = XS';
fclose(f1);

%f2 = fopen('YS.txt','r');
%YS = fscanf(f2,'%g',[2*noElmtLong,2*noElmtTran]);
%YS = YS';
%fclose(f2);

switch string
    case 'SF1', str = 'SF1S.txt';
    case 'SF4', str = 'SF4S.txt';
    case 'SM1', str = 'SM1S.txt';
    otherwise, disp('Unknown Required Data in Shell')
end;
f3 = fopen(str,'r');
DataS = fscanf(f3,'%g',[2*noElmtLong,2*noElmtTran]);
DataS = DataS';
fclose(f3);

%Data = interp2(XS,YS,DataS,xi,yi,'spline');
%%Data = griddata(XS,YS,DataS,xi,yi,'cubic');
```

```
if (yi<widthOH)
    elmtSize = widthOH/noElmtOH/2;
    index1 = fix(yi/elmtSize)+1;
elseif (yi<(widthOH+S*(noGirder-1)))
    elmtSize = S/noElmtG/2;
    index1 = noElmtOH*2 + fix((yi-widthOH)/elmtSize) + 1;
else
    elmtSize = widthOH/noElmtOH/2;
    index1 = noElmtOH*2 + noElmtG*(noGirder-1)*2 + fix((yi-widthOH-S*(noGirder-
1))/elmtSize) + 1;
    if (index1>size(XS,1))
        index1 = size(XS,1);
    end;
end;
Data = interp1(XS(index1,:),DataS(index1,:),xi,'spline');



%% Plot the surface at data point
%nFig = 1;
%figure(nFig);
%meshc(XS,YS,DataS);
%xlabel('X-Coordinate, ft.')
%ylabel('Y-Coordinate, ft.')
%switch string
%case 'SF1',
%    zlabel('SF1 in shell, in.kips')
%       title('Surface of SF1 data points in shell')
%case 'SM1',
%    zlabel('SM1 in shell, in.kips')
%       title('Surface of SM1 data points in shell')
%otherwise, disp('Unknown Required Data in Beam')
%end;
%axis([0 lengthTotal 0 widthTotal -3 1])
%
%% Plot the spline interpolation surface of shell SF1 or SM1
%xinterval = 100;
%yinterval = 50;
%ysize = widthTotal/yinterval;
%x1 = 0;
%for i=1:noSpan
%       nFig = nFig+1;
%    figure(nFig);
%    sizeelmt = lengthSpan(i)/noElmtSpan(i);
%    x1 = x1+sizeelmt/2;
%    xsize = (lengthSpan(i)-sizeelmt)/xinterval;
%    x2 = x1+lengthSpan(i)-sizeelmt;
%       [xi,yi] = meshgrid(x1:xsize:x2,0:ysize:widthTotal);
%    Datatemp = interp2(XS,YS,DataS,xi,yi,'spline');
%       meshc(xi,yi,Datatemp);
%       xlabel('X-Coordinate, ft.')
%       ylabel('Y-Coordinate, ft.')
%       switch string
%    case 'SF1',
%        zlabel('SF1 in shell, in.kips')
%              title('Spline Interpolation surface of SF1 in shell')
%    case 'SM1',
%        zlabel('SM1 in shell, in.kips')
%              title('Spline Interpolation surface of SM1 in shell')
%    otherwise, disp('Unknown Required Data in Beam')
%    end;
%       axis([0 lengthTotal 0 widthTotal -3 1])
%
%    nFig = nFig+1;
%    figure(nFig)
%    Datatemp2 = griddata(XS,YS,DataS,xi,yi,'linear');
%       meshc(xi,yi,Datatemp2);
%       xlabel('X-Coordinate, ft.')
%       ylabel('Y-Coordinate, ft.')
%       switch string
%    case 'SF1',
%        zlabel('SF1 in shell, in.kips')
%              title('Griddata Interpolation surface of SF1 in shell')
%    case 'SM1',
%        zlabel('SM1 in shell, in.kips')
%              title('Griddata Interpolation surface of SM1 in shell')
%    otherwise, disp('Unknown Required Data in Beam')
%    end;
```

```
%       axis([0 lengthTotal 0 widthTotal -3 1])
%
%
%    x1 = x2+sizeelmt/2;
%end;
```

Program List of Loadposition.m

```
function Loading_Position;
%Input the span lengths of bidge and distance between second and third axial to get
%the position of truck that produce maximum positive moment, max. neg. moment and
%max. shear
% - Same as 'loadposition3.m'

%INPUT
vDistance = 14;     % Distance between second and third axial (14-30 ft)
span = 2;
lengthSpan = [140,126];    % unit = ft.
interval = 0.1;     % Each analysis, move truck "interval" ft to the right
[recommend=0.1]
%END INPUT


%Start main program
truckPosition = 14.05+vDistance;    % Position of front axial of truck
[recommend=14.05+vDistance]

maxV = 0; x1MaxV = 0; x2MaxV = 0; truckMaxV = 0;
minV = 0; x1MinV = 0; x2MinV = 0; truckMinV = 0;
maxM = 0; xMaxM = 0; truckMaxM = 0;
minM = 0; xMinM = 0; truckMinM = 0;
for i=1:span-1
    for j=1:span-1
        a(i,j) = 0;
    end;
    b(i) = 0;
end;

temp = 0;
load(1,2) = 0;
for i=1:span
    temp = temp+lengthSpan(i);
    load(i+1,2) = temp;
end;
lengthTotal = load(span+1,2);

while (truckPosition<lengthTotal)
    load(span+2,1) = -16;
    load(span+2,2) = truckPosition-14-vDistance;
    load(span+3,1) = -16;
    load(span+3,2) = truckPosition-14;
    load(span+4,1) = -4;
    load(span+4,2) = truckPosition;

    if (span~=1)
        for i=1:span-1
            for j=1:span-1
                a(i,j) = Deflection(load(j+1,2),lengthTotal,load(i+1,2));
            end;
            b(i) =  4*Deflection(load(span+4,2),lengthTotal,load(i+1,2)) +
16*Deflection(load(span+3,2),lengthTotal,load(i+1,2)) +
16*Deflection(load(span+2,2),lengthTotal,load(i+1,2));
        end;

        R = GaussianElimination(a,b);
        for i=1:span-1
            load(i+1,1) = R(i);
        end;
    end;

    mTemp = 0;
        for i=1:span+4
        if (i~=1 & i~=span+1)
            mTemp = mTemp+load(i,1)*load(i,2);
        end;
        end;
        load(span+1,1) = -mTemp/load(span+1,2);
```

```
        temp = 0;
        for i=2:span+4
            temp= temp+load(i,1);
    end;
        load(1,1) = -temp;

        % Sort load
        xSort(:,:) = load(:,:);         % variable 'xSort' is same as 'load' but sort in
order of x
        for i=2:span+4
         for j=1:i-1
             if (xSort(i,2)<xSort(j,2))
                 temp = xSort(j,:);
                 xSort(j,:) = xSort(i,:);
                 xSort(i,:) = temp;
             end;
         end;
        end;

        for i=1:span+4
         shearTemp  = 0;
         momentTemp = 0;
         for j=1:i
             shearTemp  = shearTemp+xSort(j,1);
             momentTemp = momentTemp+xSort(j,1)*(xSort(i,2)-xSort(j,2));
         end;
         shear(i)  = shearTemp;
         moment(i) = momentTemp;
         if (shear(i)>maxV)
             maxV = shear(i);
             x1MaxV = xSort(i,2);
             x2MaxV = xSort(i+1,2);
             truckMaxV = truckPosition;
         end;
         if (shear(i)<minV)
             minV = shear(i);
             x1MinV = xSort(i,2);
             x2MinV = xSort(i+1,2);
             truckMinV = truckPosition;
         end;
         if (moment(i)>maxM)
             maxM = moment(i);
             xMaxM = xSort(i,2);
             truckMaxM = truckPosition;
         end;
         if (moment(i)<minM)
             minM = moment(i);
             xMinM = xSort(i,2);
             truckMinM = truckPosition;
         end;
        end;

    truckPosition = truckPosition+interval;
end;

fprintf('\nINPUT:-\n');
fprintf('\n   Bridge span length = ');
for i=1:span
    fprintf('%5.2f  ',lengthSpan(i));
end;
fprintf('ft.');
fprintf('\n   vDistance = %5.2f ft.\n',vDistance);
fprintf('\nRESULT:-\n');
fprintf('\n   Maximum moment = %10.2f kips-in, at location x = %5.2f ft, when truck
position = %6.2f ft\n',maxM*12,xMaxM,truckMaxM);
fprintf('   Minimum moment = %10.2f kips-in, at location x = %5.2f ft, when truck
position = %6.2f ft\n\n',minM*12,xMinM,truckMinM);
fprintf('   Maximum shear  = %10.2f kips   , at location x = %5.2f - %5.2f ft, when truck
position = %6.2f ft\n',  maxV,x1MaxV,x2MaxV,truckMaxV);
fprintf('   Minimum shear  = %10.2f kips   , at location x = %5.2f - %5.2f ft, when truck
position = %6.2f ft\n\n',minV,x1MinV,x2MinV,truckMinV);
%End MAIN PROGRAM


function [y] = Deflection(a,L,x);
%Get the deflection of beam
b = L-a;
if (x<=a & x>=0)
```

```
    y = b*x*(L^2-x^2-b^2)/(6*L);
elseif (x<=L)
    y = b*(L/b*(x-a)^3+(L^2-b^2)*x-x^3)/(6*L);
else
    fprintf('Truck Position Error!');
end;


function x = GaussianElimination(A,f);
%Solve linear system of equations
n = size(A,1);
m = zeros(n,1);
for k=1:n-1
    m(k+1:n) = A(k+1:n,k)/A(k,k);
    for i=k+1:n
        A(i,k+1:n) = A(i,k+1:n)-m(i)*A(k,k+1:n);
    end;
    for j=k+1:n
        f(j) = f(j)-m(j)*f(k);
    end;
end;
U = triu(A);

f(n) = f(n)/U(n,n);
for k=n-1:-1:1
    for j=1:k
        f(j) = f(j)-U(j,k+1)*f(k+1);
    end;
    f(k) = f(k)/U(k,k);
end;
x = f;
```

Program List of `location.m`

```
function Location(xLocation);
%Input similar to "loadposition.m" but require the specific x-location
%This function will find the truck position that produce the maximum moment and shear
%         at single specific location
%xLocation must be inside the bridge [0,lengthTotal] (ft)
% - Same as 'location3.m'

%INPUT
vDistance = 14;     % Distance between second and third axial (14-30 ft)
span = 1;
lengthSpan = [102,102];
interval = 0.1;     % Each analysis, move truck "interval" ft to the right
[recommend=0.1]
%END INPUT


%Start main program
truckPosition = 14.05+vDistance;    % Position of front axial of truck
[recommend=14.05+vDistance]

maxV = 0; truckMaxV = 0;
minV = 0; truckMinV = 0;
maxM = 0; truckMaxM = 0;
minM = 0; truckMinM = 0;
for i=1:span-1
    for j=1:span-1
        a(i,j) = 0;
    end;
    b(i) = 0;
end;

temp = 0;
load(1,2) = 0;
for i=1:span
    temp = temp+lengthSpan(i);
    load(i+1,2) = temp;
end;
lengthTotal = load(span+1,2);

if (xLocation>=0 & xLocation<=lengthTotal)
        while (truckPosition<lengthTotal)
        load(span+2,1) = -16;
        load(span+2,2) = truckPosition-14-vDistance;
        load(span+3,1) = -16;
        load(span+3,2) = truckPosition-14;
        load(span+4,1) = -4;
        load(span+4,2) = truckPosition;

        if (span~=1)
            for i=1:span-1
                for j=1:span-1
                    a(i,j) = Deflection(load(j+1,2),lengthTotal,load(i+1,2));
                end;
                b(i) =  4*Deflection(load(span+4,2),lengthTotal,load(i+1,2)) +
16*Deflection(load(span+3,2),lengthTotal,load(i+1,2)) +
16*Deflection(load(span+2,2),lengthTotal,load(i+1,2));
            end;

            R = GaussianElimination(a,b);
            for i=1:span-1
                load(i+1,1) = R(i);
            end;
        end;

        mTemp = 0;
               for i=1:span+4
            if (i~=1 & i~=span+1)
                mTemp = mTemp+load(i,1)*load(i,2);
            end;
               end;
               load(span+1,1) = -mTemp/load(span+1,2);
               temp = 0;
               for i=2:span+4
                   temp= temp+load(i,1);
        end;
```

```
               load(1,1) = -temp;

               % Sort load
               xSort(:,:) = load(:,:);         % variable 'xSort' is same as 'load' but
sort in order of x
               for i=2:span+4
            for j=1:i-1
                if (xSort(i,2)<xSort(j,2))
                    temp = xSort(j,:);
                    xSort(j,:) = xSort(i,:);
                    xSort(i,:) = temp;
                end;
            end;
               end;

        index = 1;
        shearTemp  = 0;
        momentTemp = 0;
        while (xSort(index,2)<xLocation)
            shearTemp  = shearTemp+xSort(index,1);
            momentTemp = momentTemp+xSort(index,1)*(xLocation-xSort(index,2));
            index = index+1;
        end;
        shear  = shearTemp;
        moment = momentTemp;
        if (shear>maxV)
            maxV = shear;
            truckMaxV = truckPosition;
        end;
        if (shear<minV)
            minV = shear;
            truckMinV = truckPosition;
        end;
        if (moment>maxM)
            maxM = moment;
            truckMaxM = truckPosition;
        end;
        if (moment<minM)
            minM = moment;
            truckMinM = truckPosition;
        end;

        truckPosition = truckPosition+interval;
    end;
else
    fprintf('\n ********** ERROR ********** \n');
    fprintf('xLocation is outside the bridge\n\n');
end;

fprintf('\nINPUT:-\n');
fprintf('\n   Bridge span length = ');
for i=1:span
    fprintf('%5.2f  ',lengthSpan(i));
end;
fprintf('ft.');
fprintf('\n   vDistance = %5.2f ft.\n',vDistance);
fprintf('   xLocation = %5.2f ft.\n',xLocation);
fprintf('\nRESULT:-\n');
fprintf('\n   Maximum moment = %10.2f kips-in, when truck position = %6.2f
ft\n',maxM*12,truckMaxM);
fprintf('   Minimum moment = %10.2f kips-in, when truck position = %6.2f
ft\n\n',minM*12,truckMinM);
fprintf('   Maximum shear  = %10.2f kips   , when truck position = %6.2f ft\n',
maxV,truckMaxV);
fprintf('   Minimum shear  = %10.2f kips   , when truck position = %6.2f
ft\n\n',minV,truckMinV);
%End MAIN PROGRAM


function [y] = Deflection(a,L,x);
%Get the deflection of beam
b = L-a;
if (x<=a & x>=0)
    y = b*x*(L^2-x^2-b^2)/(6*L);
elseif (x<=L)
    y = b*(L/b*(x-a)^3+(L^2-b^2)*x-x^3)/(6*L);
else
    fprintf('Truck Position Error!');
```

```
end;


function x = GaussianElimination(A,f);
%Solve linear system of equations
n = size(A,1);
m = zeros(n,1);
for k=1:n-1
    m(k+1:n) = A(k+1:n,k)/A(k,k);
    for i=k+1:n
        A(i,k+1:n) = A(i,k+1:n)-m(i)*A(k,k+1:n);
    end;
    for j=k+1:n
        f(j) = f(j)-m(j)*f(k);
    end;
end;
U = triu(A);

f(n) = f(n)/U(n,n);
for k=n-1:-1:1
    for j=1:k
        f(j) = f(j)-U(j,k+1)*f(k+1);
    end;
    f(k) = f(k)/U(k,k);
end;
x = f;
```