DEVELOPMENT OF A GRAPHICAL USER INTERFACE FOR COMPOSITE BRIDGE FINITE ELEMENT ANALYSIS

DENİZ GÜVEN

JANUARY 2007

DEVELOPMENT OF A GRAPHICAL USER INTERFACE FOR COMPOSITE BRIDGE FINITE ELEMENT ANALYSIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DENİZ GÜVEN

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

JANUARY 2007

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

Prof. Dr. Güney ÖZCEBE
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

Assoc. Prof. Dr. Cem TOPKAYA
Supervisor

Examining Committee Members

Prof. Dr. Çetin YILMAZ	(METU, CE)	_____
Assoc. Prof. Dr. Cem TOPKAYA	(METU, CE)	_____
Asst. Prof. Dr. Alp CANER	(METU, CE)	_____
Dr. Özgür KURÇ	(METU, CE)	_____
Ateeq AHMAD (M.S.)	(PROYA)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Deniz GÜVEN

Signature :

ABSTRACT

DEVELOPMENT OF A GRAPHICAL USER INTERFACE FOR COMPOSITE BRIDGE FINITE ELEMENT ANALYSIS

Güven, Deniz
M.S., Department of Civil Engineering
Supervisor: Associate Professor Dr. Cem Topkaya

January 2007, 91 pages

Curved bridges with steel/concrete composite girders are used frequently in the recent years. Analysis of these structural systems presents a variety of challenges. Finite element method offers the most elaborate treatment for these systems, however its use is limited in routine design practice due to modeling requirements. In recent years, a finite element program named UTrAp was developed to analyze construction stages of curved/straight composite bridges. The original Graphical User Interface could not be used with the modified computation engine. It is the focus of this thesis work to develop a brand new Graphical User Interface with enhanced visual capabilities compatible with the engine. Pursuant to this goal a Graphical User Interface was developed using C++ programming language together with OPENGL libraries. The interface is linked to the computational engine to enable direct interaction between two programs. In the following thesis work the development of the GUI and the modifications to the computational engine are presented. Moreover, the analysis results pertaining to the newly added features are checked against analytical solutions and recommendations presented in design specifications.

Keywords: Composite Curved Bridge, Finite Element Method, Software, Graphical User Interface

ÖZ

KOMPOZİT KÖPRÜ SONLU ELEMAN ANALİZİ İÇİN KULLANICI GRAFİK ARAYÜZÜ GELİŞTİRİLMESİ

Güven, Deniz

Yüksek Lisans, İnşaat Mühendisliği Bölümü

Tez Yöneticisi : Doç. Dr. Cem Topkaya

January 2007, 91 sayfa

Son yıllarda, çelik/beton kompozit gövdeli kavisli köprüler sıklıkla kullanılmaktadır. Bu yapı sistemlerinin analizi bir çok zorluğu beraberinde getirmektedir. Sonlu elemanlar metodu, modelleme gereksinimlerinden dolayı rutin tasarım uygulamalarında sınırlı kullanımı olmasına karşın, bu tür sistemler için detaylı çözümler sunmaktadır. Son yıllarda, kompozit gövdeli kavisli ve düz köprülerin yapım aşamalarını analiz etmek için UTrAp adında bir sonlu elemanlar programı geliştirilmiştir. Orijinal grafik ara yüzü, geliştirilmiş hesap modülü ile kullanılamamaktadır. Bu tez çalışmasının amacı, hesap modülü ile uyumlu, ileri seviyede görsel yetenekleri olan yeni bir kullanıcı grafik arayüzü geliştirmektir. Bu amaç doğrultusunda, OPENGL kütüphaneleri ile beraber C++ programlama dili kullanılarak bir kullanıcı grafik arayüzü geliştirilmiştir. Arayüz, iki modül arasında doğrudan etkileşim kurmak için hesap modülüne direk olarak bağlanmıştır. Aşağıdaki tez çalışmasında, kullanıcı grafik arayüzünün geliştirilmesi ve hesap modülüne yapılan değişiklikler sunulmuştur. Bunlara ek olarak, yeni eklenmiş özelliklere ait olan analiz sonuçları, analitik çözümlerle ve tasarım şartnamelerinde belirtilen değerlerle karşılaştırılarak doğrulukları kontrol edilmiştir.

Anahtar Kelimeler: Kavisli Köprü, Sonlu Eleman Metodu, Yazılım, Kullanıcı Arayüzü

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Background

Due to the advances in fabrication technology steel/concrete composite curved bridge systems are being used frequently in the recent years. Based on the shape of the steel section used there are basically two types of steel/concrete composite girder systems. Traditionally the steel section is composed of a monosymmetric I shape as shown in Fig.1.1 The disadvantage of this system is that the I-girders are torsionally and laterally weak and need to be braced along its length. The alternative to I girder systems is the box girder system as shown in Fig.1.2 and Fig.1.3.

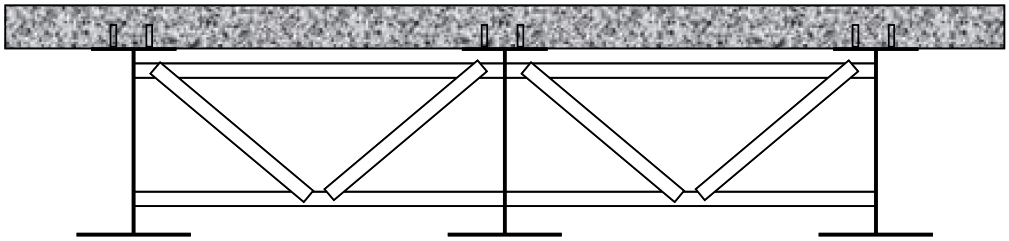


Fig.1.1. A Typical I-Girder System

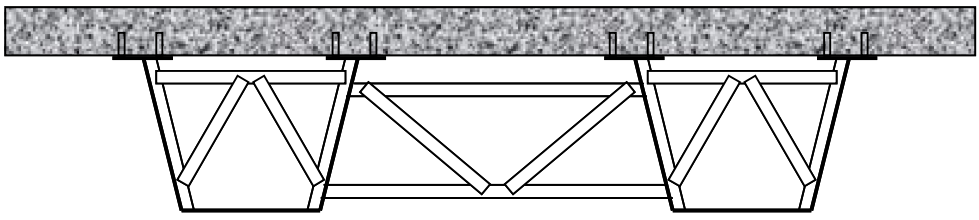


Fig.1.2. A Typical Box Girder System

Fig.1.3. A Photo of a Curved Box Composite Bridge

Both in I-girder and box girder systems, composite action between steel and concrete is achieved by using connectors such as welded shear studs. Critical loads in design develop usually at construction loads and service time. During construction, steel girders are placed at top of the piers over the bearings. A permanent metal deck form (PMDF) is installed in between girders and PMDF is basically used as a formwork for reinforced concrete deck. Usually the reinforced concrete deck is placed in a number of segments to reduce shrinkage. Depending on the time of concrete curing some parts of the bridge may be partially composite. After the completion of the bridge, the system is typically opened to heavy truck loads. Among the two, construction loads are critical compared to the live loads, since 60 to 70 percent of maximum cross section stresses occur during construction.

Analysis of horizontally curved girders presents a variety of challenges. Approximate hand methods, grid analysis method and finite element method are generally used for the analysis of these systems. Among the analysis methods mentioned finite element method provides the most elaborate treatment. However, performing finite

element analysis requires the knowledge of this method by the designer and significant amount of computer resources. Although there are well developed general purpose finite element programs, their use in curved steel/composite bridge design is limited.

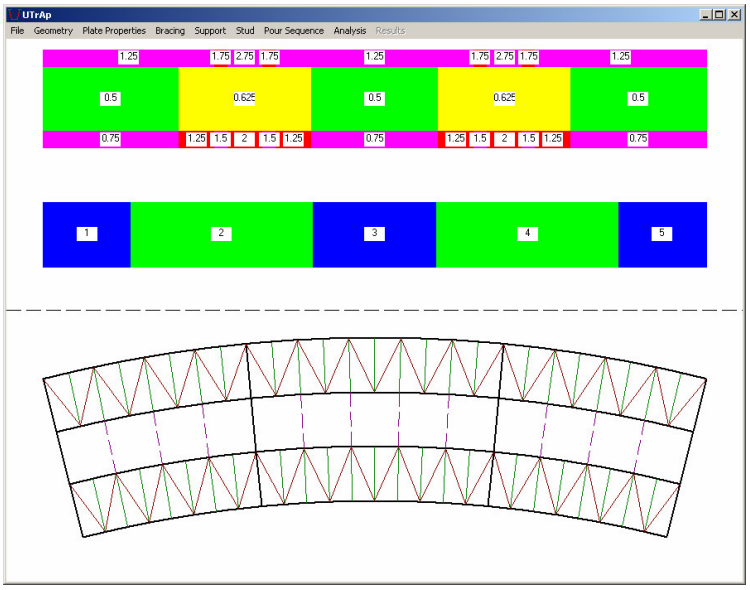
In the recent years there have been some bridge failures. These failures range from buckling of a single member in a box girder system (Fig.1.4) to the complete collapse of an I-girder system (Fig.1.5). These failures were mainly during the construction stage and were attributable to the lack of adequate analysis tools for these types of systems.

Fig.1.4. Buckling of a Brace Member in a Curved Box Girder Bridge

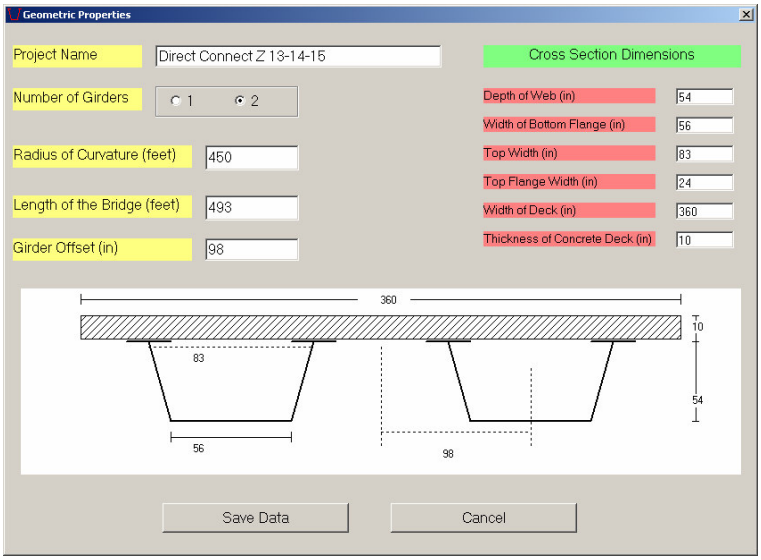
Fig.1.5. Complete Collapse of a Curved I-girder Bridge

In order to circumvent the problems related with curved bridge analysis a study has been undertaken at the University of Texas at Austin in early 2000. The study aimed to develop an easy to use finite element computer program for the analysis of curved composite box girder systems under construction loads. The study was completed in the year 2002 and as a deliverable a program named UTrAp was released.

The first version of UTrAp (Topkaya and Williamson, 2003) is a computer program capable of analyzing curved composite box girder systems in an optimal manner. The program includes a computational engine written in FORTRAN and a graphical user interface (GUI) written in Visual Basic (Fig.1.6). The GUI prepares the text files necessary for the input to the computational engine. After the analysis of the bridge system important information such as deflections, stresses, brace forces are written to text files which later on are read by the GUI for display.



a)



b)

Fig.1.6. View of the Graphical User Interface of UTrAp a) Main Form b) Cross Sectional Properties Form

The computational engine has linear three dimensional finite element program. Based on the input for geometry the computational engine automatically prepares the mesh by producing nodes, elements, element properties, loading and etc. After preprocessing, the element stiffness matrices are formed and assembled into a global stiffness matrix. The unknown displacements are found using a sparse solver which is optimized for PCs. After the processing stage, the displacements are post processed to get stresses, stress resultants, member forces, rotations and etc. In UTrAp, the steel girders and concrete deck are modeled with 9-node shell elements, and the bracing members are modeled with truss elements. These elements can be used to model both thin and thick shells. Results obtained from UTrAp have been shown to compare well with published solutions and observations from field studies (Topkaya and Williamson 2003, Topkaya et al. 2004).

After success of the first version of UTrAp original developers independently modified the program for new capabilities. At the University of Texas at Austin an analysis module has been added to the program that is capable of performing an eigenvalue buckling analysis (Popp 2004). This enabled for the solution of lateral buckling loads for curved composite box girder systems. At the Middle East Technical University side a totally different direction has been taken. The computational engine has been restructured so that new capabilities can be added with minimal effort (Kalaycı 2005). Apart from the restructuring of the code some capabilities were also added. The developments as of year 2005 were summarized by Kalaycı (2005) and are given here in Table 1. As can be seen from this table several important features were added. With the new additions users are capable of analyzing straight as well as curved bridges with variable radius of curvature. There is a possibility of choosing different element types depending on the desired accuracy. Element size along the bridge length can be input by the user which was set to a constant value of two feet in the earlier program. In addition, users can analyze I-girder systems using this updated version.

Table 1.1. Improvements to the Computational Engine as of 2005

UTrAp (2003)	UTrAp (2005)
Has a rigid structure and changes cannot be easily implemented	Has a flexible structure
Element size is constant	Element size is variable
Utilizes 9-node shell elements	Utilizes both 9-node and 4-node shell elements
Box Girders can be analyzed	Box Girders and I-girders can be analyzed
Constant curvature and straight bridges are analyzed	Variable curvature and straight bridges are analyzed
The adopted solver is a direct sparse solver	In addition to the direct solver an iterative solver is adopted into the program
Uses Imperial system of units	Uses imperial and metric system of units

1.2. Problem Statement and Organization of the Thesis

All the improvements made to the computational engine bring the necessity of updating of the Graphical User Interface. Input requirements and output due to these changes must be incorporated into the GUI. It is the focus of this thesis work to update the GUI to meet the current needs. In order to have a program with enhanced graphical capabilities and with a flexible code structure a new direction in the GUI programming has been taken. A GUI was developed using Borland C++ and object oriented programming rather than updating the old GUI. In addition to the newly developed GUI some improvements to the computational engine were made also. Chapter 2 of this thesis presents the additions and improvements to the computational engine. Chapter 3 details in the development of the Graphical User Interface. Chapter 4 presents some computational results obtained using the new software and finally Chapter 5 presents the conclusions. A detailed User's Manual along with an example problem is given in the Appendix.

CHAPTER 2

IMPROVEMENTS TO THE COMPUTATIONAL ENGINE

2.1. Summary of the Improvements

Several new features were added to the program to improve its capabilities. In Table 2.1 modifications and additions to the 2005 version can be found.

Table 2.1: Summary of Improvements to the Computational Engine

UTrAp (2005)	UTrAp (2007)
Only single and dual girder systems with box and I- cross section are supported	Infinitely many girders with box or I-section can be modeled and analyzed
Has only one type of internal brace for box girders	Has two types of internal brace configurations for box girders
Has only one type of external brace configuration for box and I-girder systems	Has three types of external brace configurations
Lateral braces can be connected to the top flanges of I-girders and box girders	Lateral braces can be connected to top flanges of box girders and top and bottom flanges of I-girders
Does not post process support reactions	Outputs support reactions
Does not post process shear flow at the steel/concrete interface	Outputs shear flow at the steel/concrete interface
Shell element stiffness matrices are reformed for every placement sequence analysis and for post processing	Shell element stiffness matrices are stored in the physical memory and retrieved for each analysis and post processing
No live load analysis capabilities	Has live load analysis capabilities
External braces are kept for all analysis	External braces can be removed for live load analysis

2.2. Details of the Improvements

2.2.1. Number of Girders

In earlier versions of the UTrAp the number of girders was limited to two girders in analysis due to limitations in computational resources. In a recent version, the element size is changed to a variable in computational engine so that bridges can be analyzed with coarser meshes. Now it is possible to analyze systems with more than two girders on personal computers. In the current version, users can model infinitely many girders having an I-shape or a box shape cross section next to each other. The structure of the program was completely renewed to enable meshing that is a function of the number of girders. Girders can be modeled using either 9-node or 4-node shell elements.

2.2.2. Internal Brace Types

In earlier versions only one internal brace type (type 1) (Fig. 2.1) can be specified for box girders. Following the design practice another brace type (type 2) (Fig. 2.1) is implemented into the computational engine.

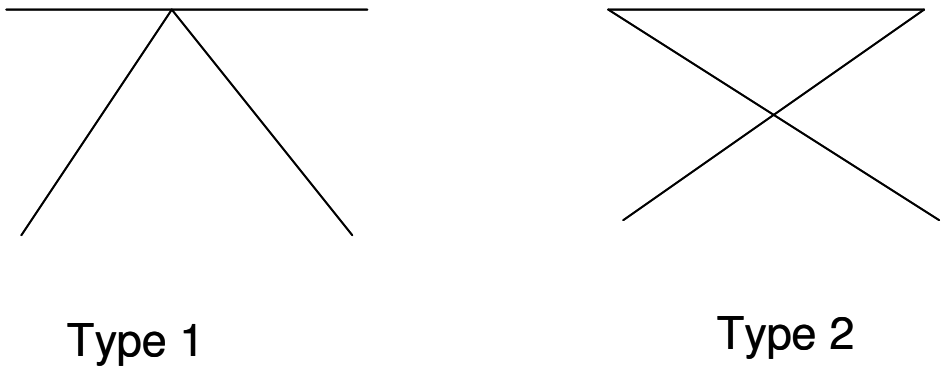


Fig. 2.1: Internal Brace Types Supported by the Computational Engine

2.2.3. External Brace Types

In the earlier versions only one external brace type (type 1) (Fig. 2.2) can be specified for box and I- girders. Following the design practice two other brace types (type 2 and type 3) (Fig. 2.2) are implemented into the computational engine.

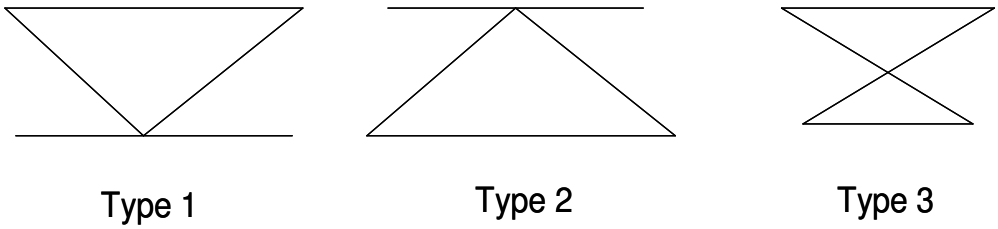


Fig. 2.2: External Brace Types Supported by the Computational Engine

2.2.4. Lateral Braces

In the earlier versions of the program lateral braces can be connected to the top flanges of box and I- girders. Following the design practice it was necessary to connect the lateral braces to the bottom flanges of the I-girders. Due to the geometry there is no need of lateral braces at bottom flange location in box girder systems. This new feature is implemented into the computational engine.

2.2.5. Post-Processing of Support Reactions and Shear Flow

In earlier versions of the program support reactions and shear flow are not output during the post processing stage. New subroutines are added to the computational engine to calculate the support reactions for each analysis case. UTrAp uses a penalty formulation to simulate the support conditions. Stiff springs are placed at the support locations to prevent vertical movement. After obtaining the nodal displacements the spring forces are calculated and summed up for each girder. The program reports support reactions for each girder as well as for the entire structure.

UTrAp models shear studs as spring elements. Using the nodal displacements the stud force at every location is calculated in this new version of the program. Later on depending on the number of studs and their spacing the force value is converted to a shear flow value. This shear flow represents the amount of shear force per length developing at the steel/concrete interface. In Chapter 4 a detailed discussion of the shear flow and slip at the interface is given.

2.2.6. Shell Element Stiffness Matrix Formulation and Storage

Due the changes in material properties in between the construction stages shell element stiffness matrices must be formed repeatedly for each analysis case. This is also true for the post processing stage. Formulation of shell element stiffness matrices is a

major process in terms of computation time. In order to reduce the time required for analysis another approach is taken in the current version of the computational engine. Shell elements stiffness matrices are formed only once throughout an analysis and stored in the physical memory. After meshing is complete, program forms the shell element stiffness matrices for concrete sections using a modulus of unity. Later on for each analysis case during the assembly phase element stiffness matrices are retrieved from the memory and their entries are multiplied by the corresponding modulus or elasticity value specified by the user. Although this approach requires more physical memory there is a significant amount of savings in terms of the computation time.

2.2.7. Live Load Analysis Capability

This is the most significant improvement to the computational engine. In the older versions the program can only perform analysis for placement sequence. In order to have a complete package a live load analysis capability is added to the program. The live analysis capability is based on generating influence lines for the desired output quantities. Therefore, the computational engine does not require any type of live load as in input. The program has been tailored to generate loading and produce influence lines for the desired quantities. In the older version of the program only single right hand side can be processed. The computational engine has been structured to process multiple right hand sides that correspond to multiple load cases. The CXML solver implemented into the program has the capability of performing solutions for systems having multiple right hand sides.

It is impractical and computational very intensive to place loading to every single node on the deck segment and produce a detailed influence surface. In order to circumvent this problem it was decided to produce influence lines based on lane loads. Therefore, user has to specify the number of lanes and their corresponding geometry as an input. The program automatically develops distribution factors for the deck nodes based on the geometric location of the nodes. Fig. 2.3 presents a generic load distribution for a lane defined by a start and end location.

The distributed lane load has a magnitude of unity when integrated along the width of the deck. This unit force is distributed to the nodes by statical equilibrium. This is performed by a subroutine added to the program to find out the distribution factors. After finding the distribution factors for each lane the load is traveled along the bridge length. At this point the loading interval is conceived to be a variable. User can specify

the live load placement interval as a function of the element size. That is user can space the loads at every element, two elements etc. The most accurate case would be to place loading at every element; however, this has the drawback of producing a large number of right hand sides thereby increasing the computational cost significantly. After the right hand side vectors are formed the program calculates displacement vector for each right hand side vector and performs post processing of the results. The output from post processing is stored into arrays which are going to be processed further by the Graphical User Interface. The Graphical User Interface generates influence lines for the desired quantity. Making use of these influence lines the GUI calculates the final results due to distributed or traveling truck loads. Details of the GUI are given in Chapter 3.

Another feature that is added to the program is the external brace removal capability. Usually external braces are removed after the bridge is constructed due to fatigue issues. Users can choose to remove external braces from the model during live load analysis.

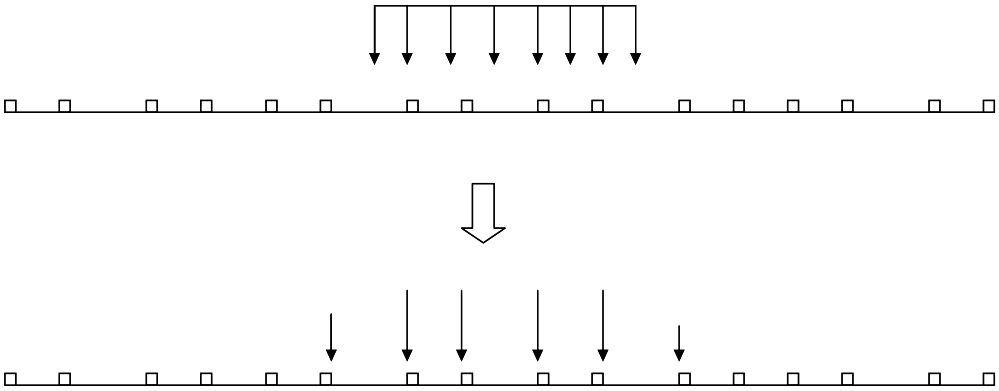


Fig. 2.3: Distribution of Lane Load to Individual Deck Nodes

CHAPTER 3

DEVELOPMENT OF A GRAPHICAL USER INTERFACE

In the first version of UTrAp, there was a computational engine in FORTRAN and a graphical user interface developed in Visual Basic.

In current version, a new windows user interface is developed using C++ language. Unlike the former version, this interface is in direct interaction with the computational engine. To accomplish this direct interaction, the FORTRAN part (computational engine) of UTrAp is reformed as a DLL (Dynamic Link Library) file. Hence, the program is composed of mainly two parts, the user interface part (EXE file) and computational engine part (DLL file).

The main working principle is the fact that the interface calls the required function in the DLL and passes the inputs to DLL part, and DLL part makes the calculations and returns the results via this function.

In this chapter, structure of the user interface part is explained. Interface part is composed of mainly two groups. These are visual user interface components and object classes. Former group is composed of main frame of the interface, view window class and number of dialog box classes used for input and output. Latter group consists of object classes and database class.

3.1. User Interface Components

3.1.1. Main Frame Class

This class establishes the main window of the program. All the menus, toolbars and view windows are created on it. In Fig. 3.1, the components on the main frame of the program are shown. In the program, only one instance of this class is created.

3.1.2. View Window Class

This class creates the view window of program and handles all the drawing operations and the interaction with user such as zoom, rotate and pan-move of model. In the program multiple instance of this view window can be created. Each window has own viewing settings. These settings include viewing position, scale of the model, and visibility of the objects drawn. Therefore, with multiple views, model can be viewed from different positions and with different draw states.

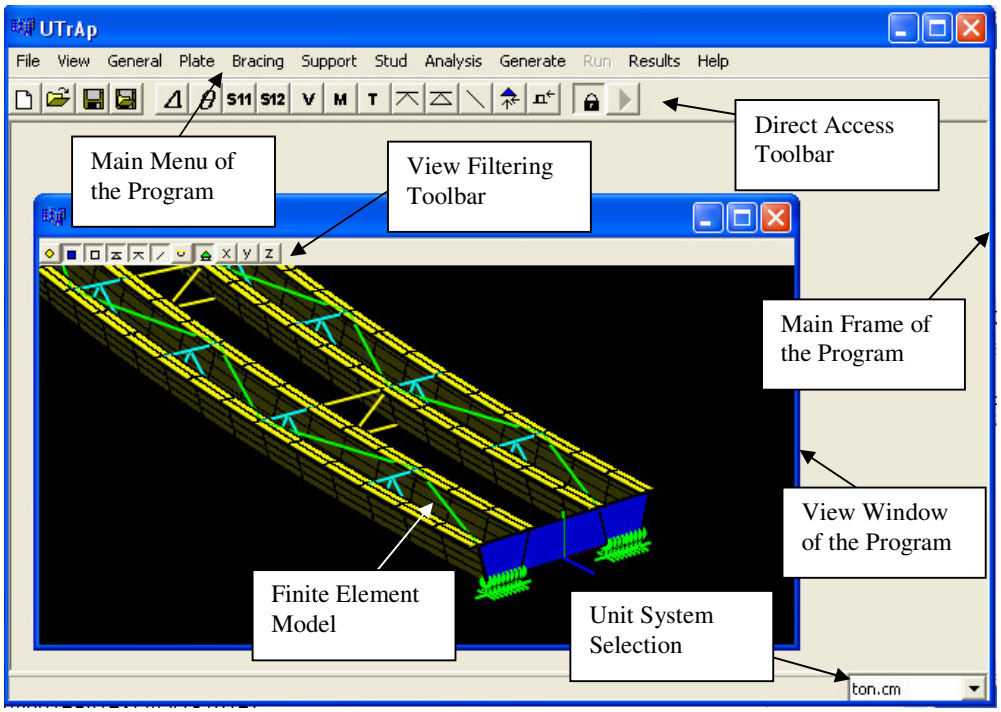


Figure 3.1. General View of the Program

3.1.3. Dialog Box Classes

The dialog boxes used in the program are listed below with their functions.

General Properties: Shows for bridge geometric properties and material constants.

Typical Sections: Shows typical sections i.e. box girder and I girder.

Plate Properties: Shows plate properties used in bridge.

Bracing Properties: Shows brace properties used in bridge.

Brace Types: Shows brace types for external, internal and lateral braces.

Support Positions: Shows support positions for the bridge.

Stud Properties: Shows stud properties used in the bridge.

Analysis Options: Shows analysis options for the analysis

Placement Sequence: Shows placement sequence for the bridge

Live Loading Properties: Shows live loading parameters defined for the bridge.

Live Load Cases: Shows live load cases defined for analysis.

Load Combination: Shows load combinations defined for analysis

Run and Girder Selection: For run number and girder selection.

Stress Point Selection: For stress point selection at the section of the bridge.

Brace Item Selection: Dialog box for selection of sub item of external and internal braces.

Load Case and Combination Selection: Used for selection of load case and/or load combinations.

Result Type Selection: Used for result type selection i.e. Placement Sequence, Live Load, Influence Line.

Log Viewer: Shows the report of the analysis immediately after its completion.

Graph and Table Result: A versatile graph and table visualization dialog box used for analysis results.

About Box: Shows brief information about program

View Adjust: Displayed when user double clicked on an empty place on view screen. Used for adjust of the view orientation and background color.

3.2. Object Classes in the Program

There are number of object classes used for the database and components of bridge. Main visual objects on a bridge are Shell, Node, External Brace, Internal Brace, Lateral Brace, Support, Pin elements. These visual objects are shown in Fig. 3.2.

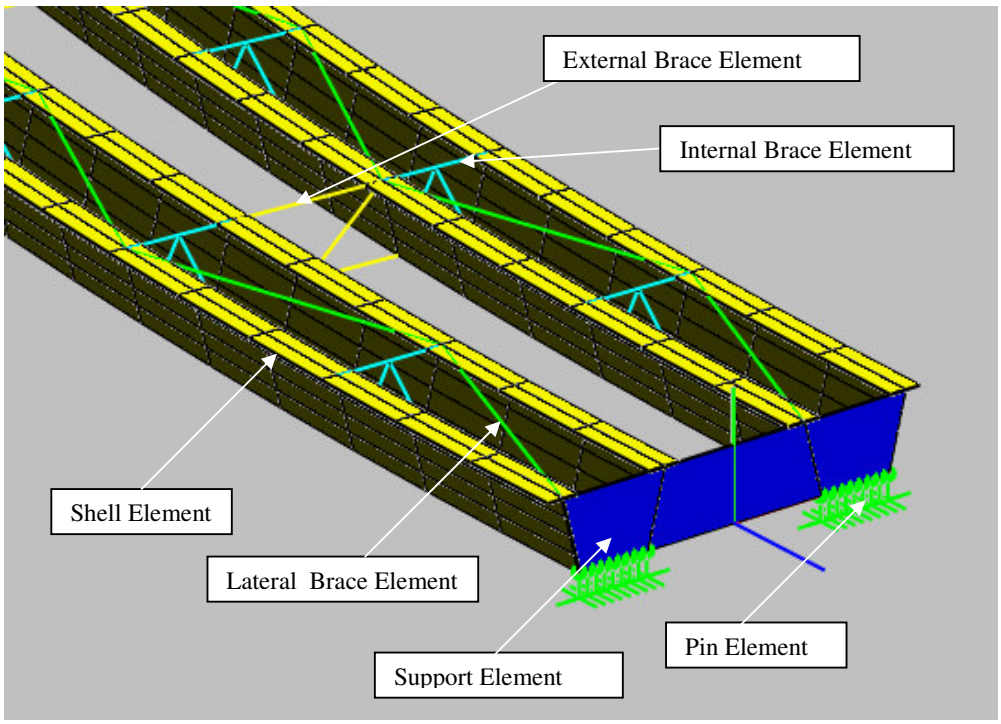


Figure 3.2. Visual Objects on a Conventional Bridge

3.2.1. AxObject Class

This class is the base class for the entire database resident objects. This class has no parent class. It only includes basic object properties and functions. This class is an abstract class therefore no instance of this class can be created.

3.2.2. AxEntity Class

This class is the base class for all database resident and visual objects. It derives from Object class. This class is also an abstract class. It includes basic functions with visual operations.

3.2.3. AxNode Class

This class derives from Entity class and represents the finite element nodes in the FE model of bridge. For every node in the model, an instance of this class is created an appended to the database.

3.2.4. AxShell Class

AxShell Class is derived from Entity class like all visual objects. It represents Shell objects in the model. This class supports variable number of nodes for shell elements.

3.2.5. AxExtBrFrame Class

AxExBrFrame Class is derived from Entity Class. It represents the external braces in the bridge.

3.2.6. AxIntBrFrame Class

AxIntBrFrame Class is derived from Entity Class. It represents the internal braces in the bridge.

3.2.7. AxLatFrame Class

AxLatBrFrame Class is derived from Entity Class. It represents the lateral braces in the bridge.

3.2.8. AxSupport Class

AxSupport Class is derived from Entity Class. It represents the support objects in the bridge.

3.2.9. AxPin Class

AxPin Class is derived from Entity Class. It represents the pin objects in the bridge.

3.2.10. AxFiler Class

AxFiler class is implemented to make read and write operations of data. This class makes possible to read and write of number of predefined data types. It can load from and save to a memory stream as well as a file. This class writes to and reads from the .axd file which holds the all project data.

Every database resident object writes and reads its data via this class.

3.2.11. AxPoint3d Class

AxPoint3d class is a geometric class representing a point in three dimensional space. It has three coordinates of double type. Database resident objects having a position data uses this class to store that point.

3.2.12. AxGraph Class

AxGraph mainly carries out the drawing operations of database resident objects. AxGraph class also handles all the operations for initialization and destruction of the OpenGL context and rotation, pan slide and zoom operations of the three dimensional model. Each view window has its own AxGraph instance and uses this instance to execute the drawing operations.

3.2.13. AxDatabase Class

AxDatabase class is the conductor of the program since it holds the entire project information, and does all the operations related with the project. In the program there is always one instance of database class and in the global scope, it can be accessed from everywhere.

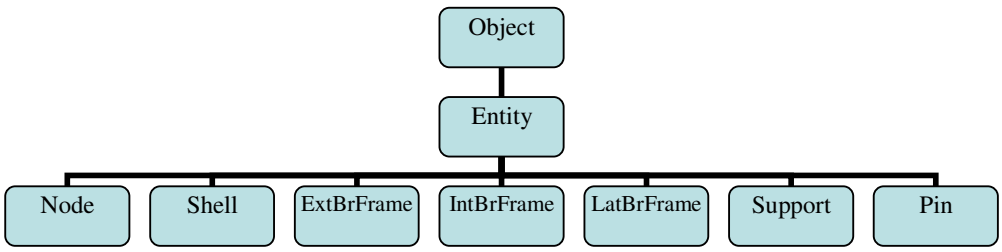


Figure 3.3. Class Hierarchy of Interface Objects

In Fig3.3, class hierarchy of database resident objects is presented. Basic database resident object class is AxObject. This class has not drawing capability. AxEntity class is derived from AxObject class. This class is the base class for the database objects with drawing capabilities. In current version of UTrAp, there is no non-visual database resident object class, but in future versions, new classes such as a material class can be derived from AxObject.

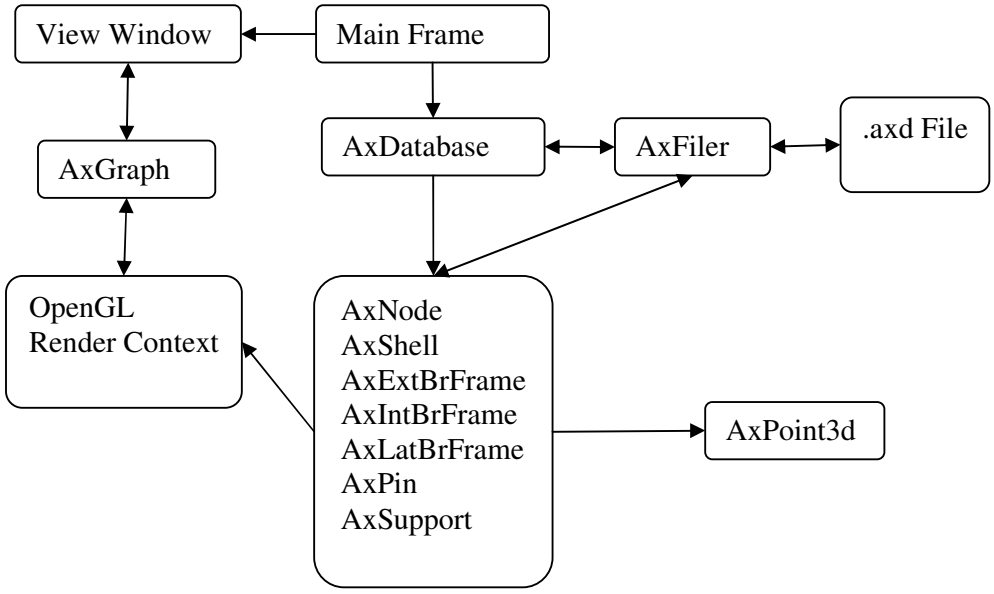


Figure 3.4. Interaction Diagram of Objects

In Fig 3.4, Interaction diagram of the graphical user interface is presented. In this diagram, mainly, object storage, drawing and save/load operations of the program are demonstrated. The detailed information about these operations is given below.

Storage of Database Resident Objects:

In database, database resident objects are stored in a vector type array. A vector type array allocates a memory space for storage. Unlike link lists, all the items in the array are sequentially stored in the allocated memory block. With the addition of newly created objects, if the current size is not enough to store, it allocates a new memory block to satisfy the required store size. To perform this reallocation, it reserves a memory block of new size and copy the previously allocated memory block to newly allocated space. This is an important drawback of vectors. To eliminate this drawback, the reallocation size should be selected enough to store all objects.

In database array, the pairs of identity numbers (id) and pointers (memory addresses) of objects are stored. Every object has a unique id. Upon the creation of a new object, a new id is assigned to it. The database array is always kept sorted with respect to

ids (For sorting algorithm, refer to Appendix B). This provides fast access to objects by ids. To access any id in the array, a quick search algorithm is utilized (For quick search algorithm, refer to Appendix B). This quick sort algorithm runs on only sorted arrays.

Save/Load Operations:

For save and load operations, the database object creates an instance of AxFiler class and associates it with an UTrAp project file specified by the user. And this instance of AxFiler object is passed to the all objects by FileIn and FileOut virtual functions to allow them to read and write their data. For write operations FileOut, for read operations FileIn functions are called. The important point is to exactly match the count of bytes written and read by any object. Since the entire project file is read and written sequentially. Therefore any inconsistency in read and write procedures of an object will fail the read and write operations of all project.

Drawing Operations:

Drawing operations are performed basically by the OpenGL (Open Graphics Library, Segal, M, Akeley, K, 1999). OpenGL is a software interface to graphics hardware. The interface consists of a set of several hundred procedures and functions that allow a programmer to specify the objects and operations involved in producing high-quality graphical images, specifically color images of three-dimensional objects.

To use OpenGL commands, a rendering context has to be created. To create a rendering context, OpenGL has to be associated with a drawing area (i.e. rendering window). AxGraph class performs this association and the required initializations. Every view window has its own AxGraph instance. This class prepares the OpenGL for drawing operations.

Upon the request of drawing of the model, the OpenGL rendering context of the active view window is made current by the AxGraph object, and then the database object calls *Draw* functions of all the objects derived from AxEntity. In Draw functions, objects calls directly OpenGL commands. By means of these commands, required drawing is done on the current view window.

Analysis Operations:

Operations undertaken by database involve also loading and calling FORTRAN analysis engine to make analysis, retrieving results and processing them.

Database class holds three sets of analysis data. These are the user inputs of project, the input data structure to send to FORTRAN analysis engine and the analysis results data structure obtained after the analysis.

The user inputs of project data are all the information entered by the user via dialog boxes or an input file.

Input data structure includes all the data sets to be passed to the analysis engine. In fact, this data set has the same information with the user inputs data set, but the format is different. The variables in the input data structure are listed in Appendix C.

Analysis results data set includes all the analysis result arrays obtained from the analysis. During analysis results visualization, this data set is processed.

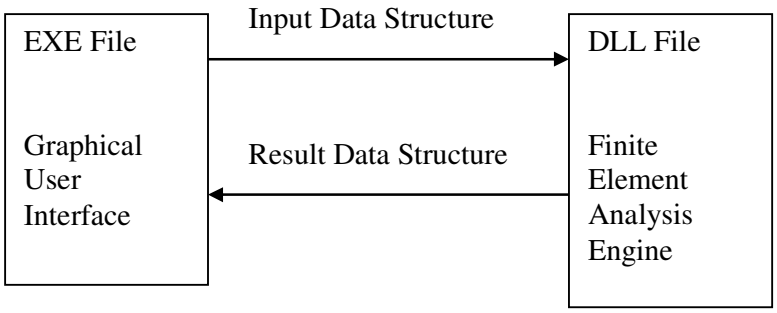


Fig 3.5. Interaction EXE and DLL Parts

Parameters passed to and taken from the analysis engine are shown in Fig.3.5.

Steps of Analysis:

- Before the analysis is conducted, FORTRAN dynamic link library is loaded and address of required function to make the analysis is taken.
- Then, the input data structure is created and filled by the database using the user inputs.
- Following the preparation of inputs, analysis function is called with these inputs.
- After the analysis is done, a number of memory addresses pointing result arrays are taken as analysis results. These memory addresses can not be used by C++ arrays directly, since FORTRAN and C++ array indexing are different. FORTRAN constitutes multi dimension array in a single thread memory stream in column wise ordering, whereas, C++ does the same job using multi thread

memory streams. Therefore FORTRAN arrays can be copied to C++ arrays as segments. This process is explained below on an example.

In Fig 3.5, for example, there is an array of n numbers and this array is same array with a FORTRAN array of a x b. To calculate the index of (r,s) element of FORTRAN array: index in memory stream= ((r – 1) x b + s)

1	2	3	.	.	.	.	n-2	n-1	n

A memory stream taken from FORTRAN

1,1	1,2	1,3	.	1,b	.	2,b	.	.	a,b

Corresponding multi-dimension array of (a x b) in FORTRAN

Figure 3.6 Memory Stream Correspondences

This array copying operation is valid for all number of dimensions of arrays. After filling the result data structure as described above, the results are ready to be used by database.

Result Quantity Calculation for Live Loading

In the live load analysis, any analysis result like force, displacement or stress at a position or on an object such as shell, brace, support, can be obtained by using the integration of influence line for that position of element under the given load.

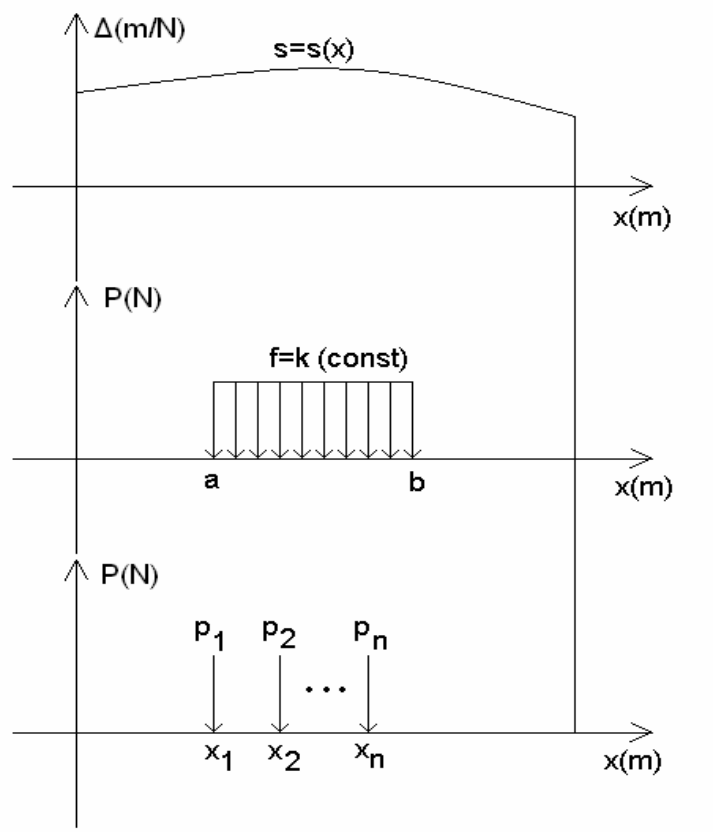


Fig 3.7 Continuous Influence Line and Loading

Say that for a position along the bridge, we have a displacement influence line s=s(x) as in Fig 3.7. The displacement quantity has to be found at that position under a constant distributed load f=k extending from x=a to x=b, and n point loads p_1, p_2, …, pn shown in Fig3.7.

Under the distributed load, the displacement value can be found as follows;

$$\Delta = \int_a^b f \cdot s(x) \cdot dx \qquad (1)$$

Under the point load, the displacement value can be found as follows;

$$\Delta = \sum_{i=1}^{n} p_i \cdot s(x_i) \qquad (2)$$

In this program, influence line is obtained as discrete lines, therefore, instead of analytical integration, numerical integration is used.

For distributed loads, to find the displacement value, the area corresponding to portion x=a, b (shaded area in Fig 3.8) is calculated and scaled with f=k value.

For point loads, using linear interpolation, to find the displacement value, the y values at load positions on influence line graph are found and scaled with the p_i values and summed up.

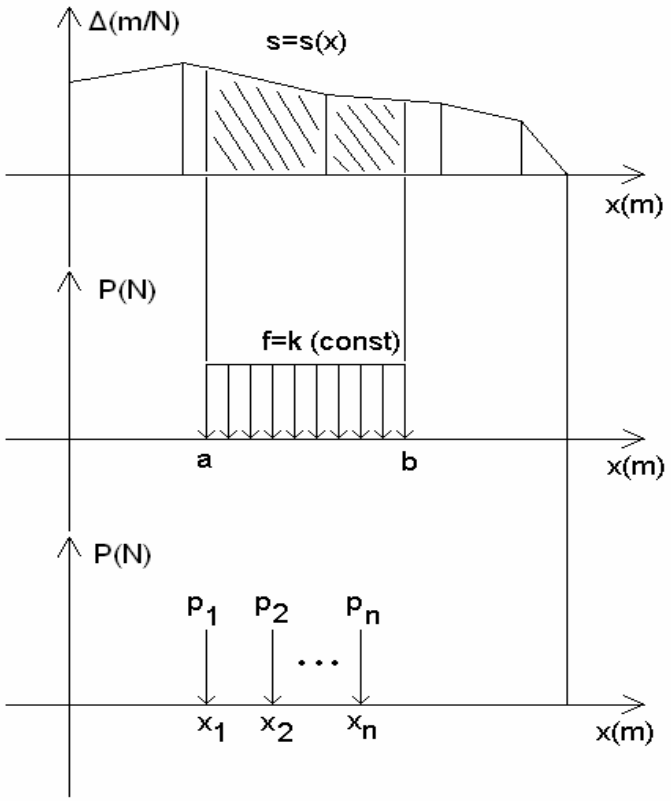


Fig 3.8 Discrete Influence Line and Loading

3.3. Difficulties Encountered and Recommendations for Future Improvements

Main difficulties encountered during development of this program are basically due to graphical operations, storage of database resident objects, save/load operations. In the following paragraphs, the difficulties encountered during programming and the future recommendations for improvements of the program are explained.

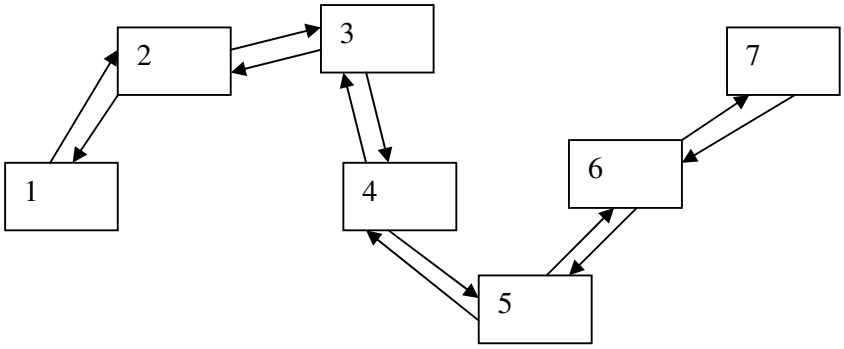
In graphical operations, the important point is to make the drawing routines fast. Since, with the increasing size of the project, graphical operations such as rotation, zoom, etc. will get slower. Therefore developer always has to optimize the drawing functions. Any unnecessary operations inside drawing functions should be removed. The total render of entire project has to be made within tens of milliseconds.

The other important point is the storage data structure of database resident objects. There are two well known data structures, i.e. vector and link list. The general memory occupations of them are shown in Fig.3.9. The main advantage of vector is to have direct access to any index in the vector. But important drawback of this data structure is reallocation necessity as mentioned sections above. Unlike vector, link list has no reallocation necessity. Since, it does not have to put entire data sequentially in a memory block. But in link list data structure, direct access to a required index is impossible. To reach any index in the list, all the nodes from start to required index has to be passed. All adjacent nodes have links to each other.

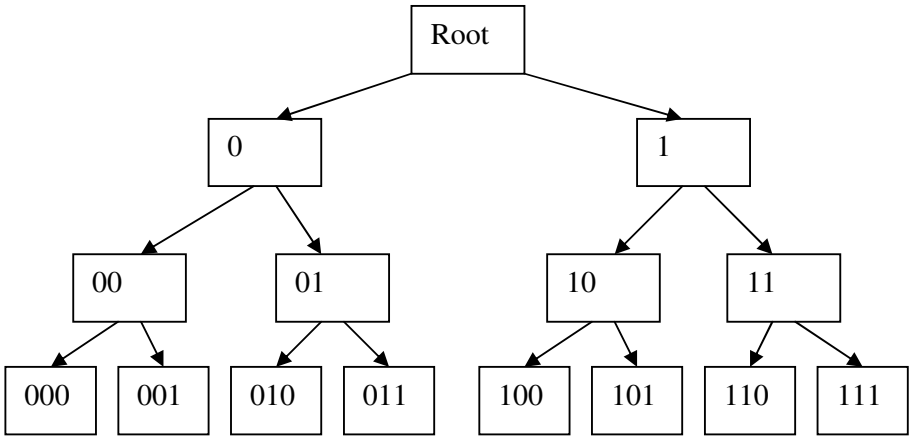
For future improvements to the program, a binary tree data structure can be embedded into the program. Binary trees have the advantages of both vector and link list. They do not have to reallocate memory, since it works like a link list but in a tree formation. Also access speed is much faster than a link list. With n step of search, 2^n number of items can be accessed.

For save load operations, the important point is to put a version counter in every save and load functions. Since, with modifications to the program data structures of objects will change and the presaved projects can not be read anymore. To avoid this situation, a version counter has to be inserted. This counter will orient the save load procedures.

A vector structure



A link list structure



A binary tree structure

Fig.3.9. Structures of Vector, Link List and Binary Tree

CHAPTER 4

PROGRAM VERIFICATION

In this chapter, the newly added features of the program were tested against closed form solutions and design charts. In the first part of the verification, slip at the interface for simple beams were compared to the closed form solutions. In the second part, maximum moment and end reaction forces due to traveling truck load were computed and are compared with the values presented in AASHTO design chart.

4.1. Part1: Interface Slip

The design of composite beams is controlled primarily by the magnitude of horizontal shear force transferred between the concrete slab and the steel beam. This force transfer is usually provided for by headed stud or other types of shear connectors, all of which are characterized by similar load-slip behavior.

The force transfer in a composite beam can be visualized by drawing free-body diagrams of the beam and the slab (Fig. 4.1) and writing equations of equilibrium of the components and the overall system. For the linear elastic case, i.e. barring any nonlinear behavior of materials and assuming that the slope of the load-slip curve for studs is constant, it is possible to write a closed form solution. The derivation of interface slip is presented by Viest et al. (1997). Following section focuses on the theory of incomplete action as explained by Viest et al. (1997).

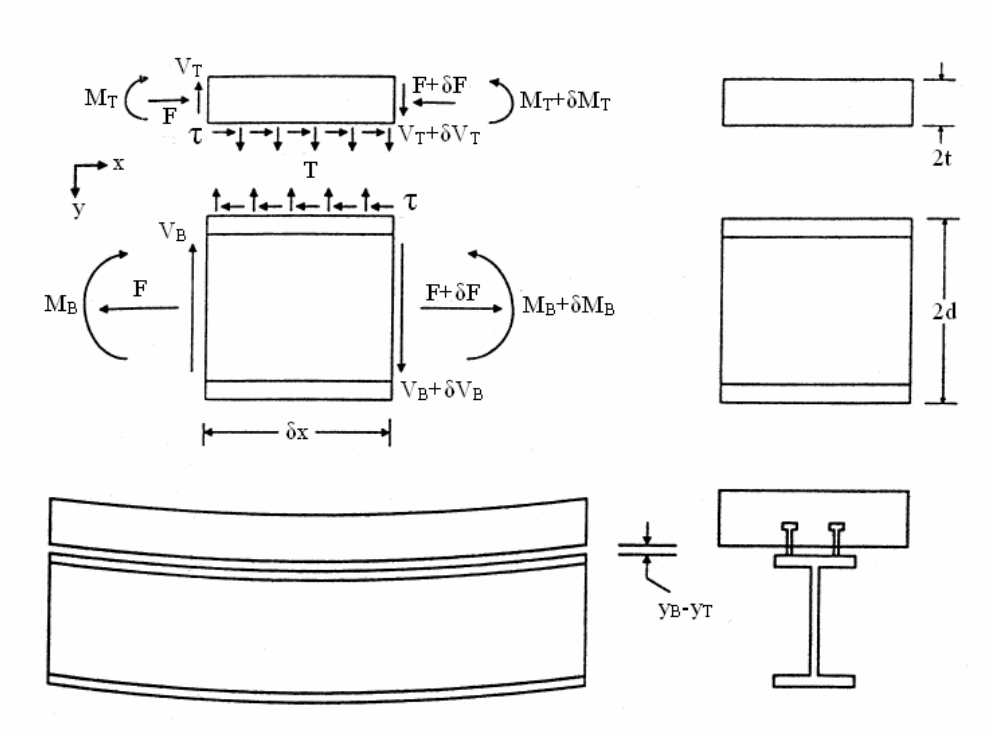


Figure 4.1 Forces Acting on a Composite Section (Ref: Viest et al., 1997)

Theory of Incomplete Interaction:

The force transfer in a composite beam with partial shear connection can be visualized by separating the problem into its constituents, the beam and the slab, and drawing their free-body diagrams (Fig. 4.1). For each of these constituents, the following equations of equilibrium can be written.

For concrete slab:

$$\partial V_T = -T \cdot \partial x \qquad (4.1)$$

$$\partial F = \tau \cdot \partial x \qquad (4.2)$$

$$\partial M_T = V_T - t \cdot \partial F \qquad (4.3)$$

For steel beam:

$$\partial V_B = -T \cdot \partial x \qquad (4.4)$$

$$\partial F = \tau \cdot \partial x \qquad (4.5)$$

$$\partial M_B = V_B - d \cdot \partial F \qquad (4.6)$$

The moment at any cross section is given by

$$M = M_T + M_B + h \cdot F \tag{4.7}$$

Where; h : distance between centroids of concrete and steel sections.

Differentiating Eqns. (4.3) and (4.6) twice with respect to x yields

$$\frac{d^2 M_T}{dx^2} = \frac{dV_T}{dx} - t\frac{d^2 F}{dx^2} \tag{4.8}$$

$$\frac{d^2 M_B}{dx^2} = \frac{dV_B}{dx} - d\frac{d^2 F}{dx^2} \tag{4.9}$$

$$M = -EI\frac{d^2 y}{dx} \tag{4.10}$$

Defining also;

$$EA_{eq} = \frac{(EA)_T (EA)_B}{(EA)_T + (EA)_B} \tag{4.11}$$

$$EI_{abs} = (EA)_T + (EA)_B \tag{4.12}$$

$$EI_{full} = EI_{abs} + EA_{eq} h^2 \tag{4.13}$$

$$\alpha^2 = \frac{k_s}{EA_{eq}} \frac{EI_{full}}{EI_{abs}} \tag{4.14}$$

where;

k_s : shear stiffness of a shear connector per unit length.

$(EA)_T$: axial stiffness of slab.

$(EA)_B$: axial stiffness of beam.

$(EI)_T$: flexural stiffness of concrete slab.

$(EI)_B$: flexural stiffness of steel section.

After rigorous derivations starting with the equations written above, following result is obtained (Ref: Viest et al., 1997).

The slip, for the case of a uniformly distributed load q, is given by

$$s(x) = \frac{qh}{EI_{abs}\alpha^3} \cdot \left[\frac{1-\cosh(\alpha \cdot l)}{\sinh(\alpha \cdot l)} \cosh(\alpha x) + \sinh(\alpha x) + \frac{\alpha \cdot l}{2} - \alpha x \right] \quad (4.15)$$

where; q : applied distributed force.

Example Problem:

The following problem was solved using UTrAp and the interface slip values are compared against closed form solutions in Fig. 4.4 and Fig. 4.5.

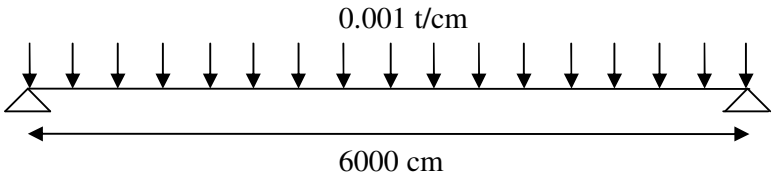


Figure 4.2 Loading of bridge for example problem

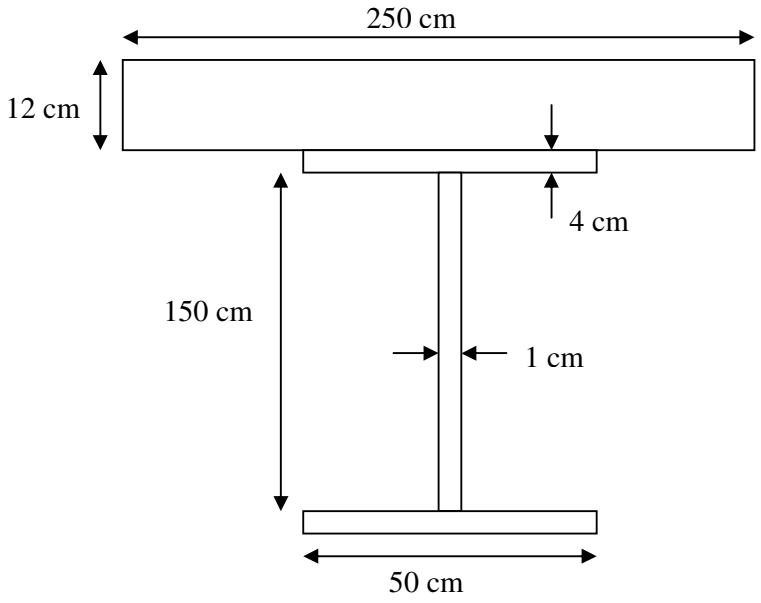


Figure 4.3 Cross-sectional properties of bridge for example problem.

$$E_c = 200t/cm^2$$

$$E_s = 2000t/cm^2$$

$$A_s = 550cm^2$$

$$I_s = \frac{(150)^3}{12} \cdot 2 \cdot 50 \cdot 4 \cdot (75)^2 = 2531250cm^2$$

$$A_c = 2500cm^3$$

$$I_c = \frac{(12)^3 \cdot 250}{12} = 36000cm^4$$

Studs every 100cm and 1 stud per flange.

$$(EA)_T = (EA)_C = (200 \cdot 2500) = 500000t$$

$$(EA)_B = (EA)_S = (2000 \cdot 550) = 1.1 \cdot 10^6 t$$

$$(EI)_T = (EI)_C = (200 \cdot 36000) = 7.2 \cdot 10^6 t \cdot cm^2$$

$$(EI)_B = (EI)_S = (2000 \cdot 2531250) = 5.06 \cdot 10^9 t \cdot cm^2$$

$$EA_{eq} = \frac{(EA)_T \cdot (EA)_B}{(EA)_T + (EA)_B} = \frac{500000 \cdot 1.1 \cdot 10^6}{500000 + 1.1 \cdot 10^6} = 343750t$$

$$EI_{abs} = (EI)_T + (EI)_B = 7.2 \cdot 10^6 + 5.06 \cdot 10^9 t \cdot cm^2$$

$$EI_{full} = EI_{abs} + EA_{eq} \cdot h^2 = 5 \cdot 067 \cdot 10^9 + 343750 \cdot (81)^2 = 7.32 \cdot 10^9 t \cdot cm^2$$

$$k_s = \frac{100t/cm}{100cm} = 1t/cm^2$$

$$\alpha^2 = \frac{k_s}{EA_{eq}} \cdot \frac{EI_{full}}{EI_{abs}} = \frac{1}{343750} \cdot \frac{7.32}{5.067} = 4.202 \cdot 10^{-6} cm^{-2}$$

$$s(x) = \frac{q \cdot h}{EI_{abs} \cdot \alpha^3} \cdot \left[\frac{1 - \cosh(\alpha L)}{\sinh(\alpha L)} \cosh(\alpha x) + \sinh(\alpha x) + \frac{\alpha L}{2} - \alpha x \right]$$

The results found from the Eqn. 4.15 and UTrAp are plotted in Fig. 4.4 and 4.5.

K_s=100 t/cm 100 cm stud spacing 1 stud per flange

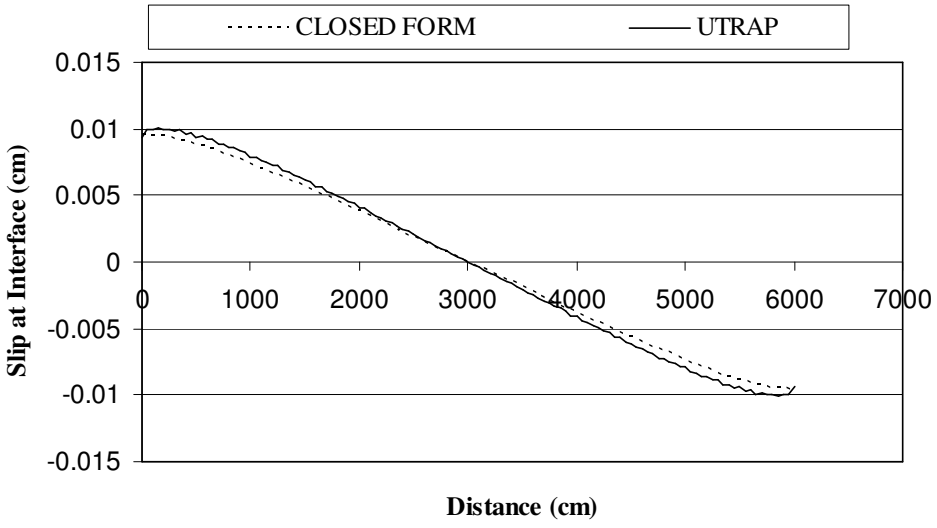


Fig.4.4 Results of slip at the interface for Ks=100t/cm

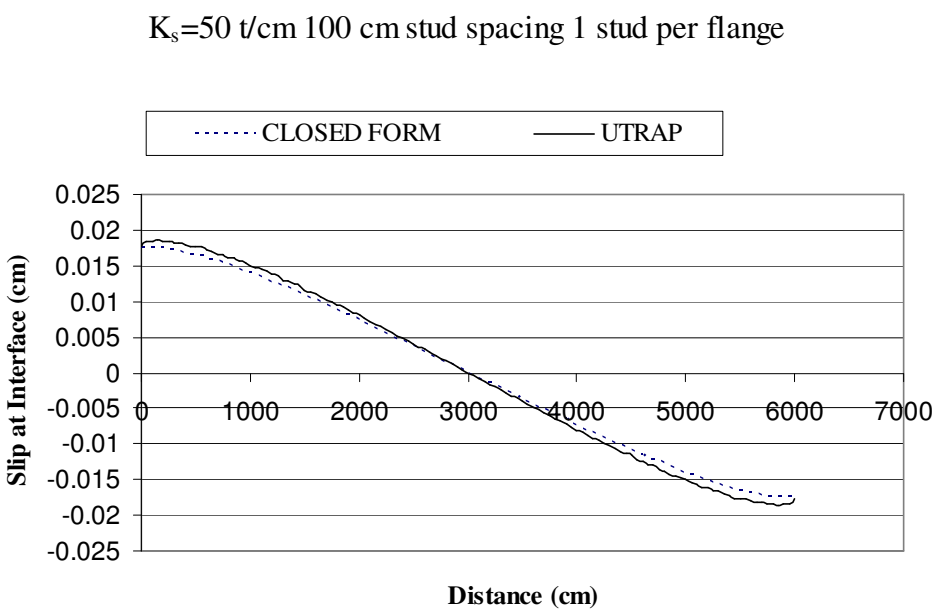


Fig.4.5 Results of slip at the interface for Ks=50t/cm

As seen in Fig. 4.4 and Fig. 4.5, there is excellent agreement between the computational results and the theoretical solution.

4.2. Part2: Comparison of Moment and End Reaction Values for Simply Supported Beams

In this part, moment and end reaction values recommended by AASHTO specifications are compared with the results obtained from UTrAp. For this comparison, simple span bridges in varying lengths are analyzed. For loading, AASHTO HS20 type loading is selected. The results are tabulated and drawn below. In Table 4.1, moment and end reaction values given in the AASHTO specifications are tabulated. Also in this table, the analysis results from UTrAp are presented along with the percentage differences. In addition to Table 4.1, Figs. 4.6 through 4.9 present the results in a graphical format.

Table 4.1. Maximum Moments and End Reactions for Given Bridge Lengths

	HS20 Loading					
	AASHTO		UTrAp			
Span Length(ft)	Moment (kip.ft)	End Reac. (kip)	Moment (kip.ft)	% difference	End Reac. (kip)	% difference
4	32.00	32.00	32.00	0.00	32.00	0.00
8	64.00	32.00	64.00	0.00	32.00	0.00
14	112.00	32.00	112.00	0.00	32.00	0.00
18	144.00	39.10	144.00	0.00	39.11	0.03
24	192.70	45.30	192.00	0.36	45.33	0.07
28	252.00	48.00	251.43	0.23	48.00	0.00
34	343.50	52.20	342.59	0.26	52.24	0.08
38	414.30	54.30	413.47	0.20	54.32	0.04
46	556.50	57.30	555.83	0.12	57.39	0.16
56	735.10	60.00	734.86	0.03	60.00	0.00
66	914.00	61.90	913.45	0.06	61.82	0.13
90	1344.40	64.50	1344.00	0.03	64.53	0.05
120	1883.30	66.40	1883.20	0.01	66.40	0.00
*170	3077.10	80.40	3076.57	0.02	80.40	0.00
*240	5688.00	102.80	5688.00	0.00	102.80	0.00
*300	8550.00	122.00	8548.80	0.01	122.00	0.00

(*) Governing loading is lane loading, otherwise truck loading governs.

Fig.4.6 Maximum moments obtained by UTrAp and recommended by AASHTO

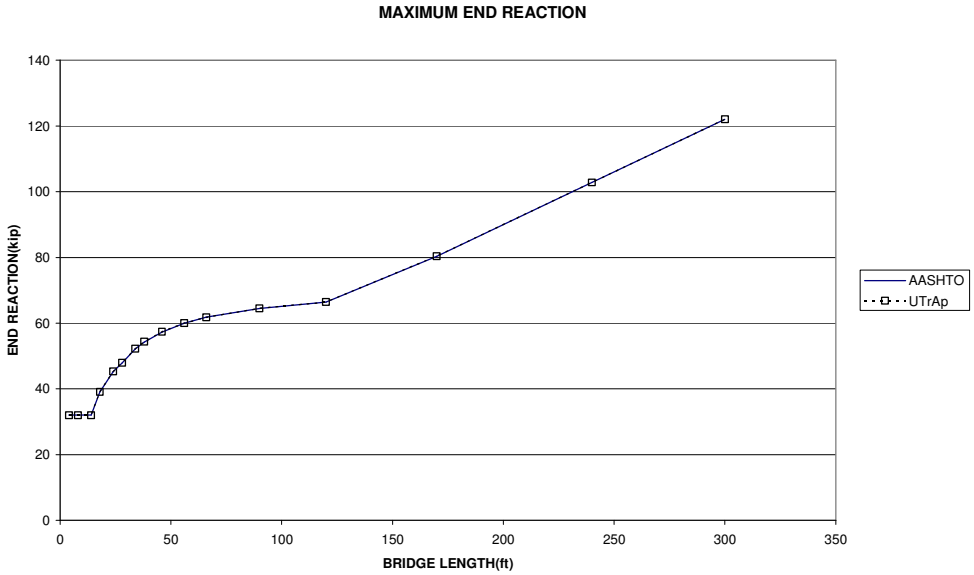


Fig.4.8 Maximum end reactions obtained by UTrAp and recommended by AASHTO

As can be seen from figures above, there is an excellent overlap between the values obtained from UTrAp and recommended by AASHTO. Maximum percentage difference is 0.36% for moment as shown in Fig.4.7. For end reaction values, maximum percentage difference is 0.15% as can be seen in Fig.4.9. These differences are quite small and in acceptable limits.

CHAPTER 5

SUMMARY, CONCLUSIONS AND FUTURE RECOMMENDATIONS

Due to the advances in fabrication technology steel/concrete composite bridges are used more frequently in complex highway interchanges. Long span capability, economics and aesthetics of these systems make them more favorable compared to the other structural systems. Analysis of curved composite bridges presents a variety of challenges. Approximate hand methods, finite strip method, grid analysis method, and finite element method can be used to analyze these systems. Among the methods presented finite element method provides the most elaborate treatment. However, this analysis method had limited use in the past due to the vast amount of computational resources needed and the knowledge of this analysis method on the designers part. In order to circumvent these problems a research study has been undertaken at the University of Texas in early 2000. In this research study a computer program named UTrAp was developed to analyze these systems using the finite element method. The first release UTrAp was capable of analyzing curved composite trapezoidal box girders under construction loads. The program had a computational engine written in FORTRAN and a Graphical User Interface written in Visual Basic.

Over the following years the computational engine of the program was enhanced. As of year 2005 the computational engine was capable of analyzing I-girder systems. In addition, the number of finite element types were increased and meshing options were added. In order to come up with a complete software package there was a need to enhance the capabilities of the computational engine further and to develop a brand new Graphical User Interface.

Pursuant to this goal several new features are added to the program as a part of this thesis work. The modifications include the addition of live load analysis capability, relaxing the limitations on the number of girders, post processing of support reactions and

shear flow, addition of new internal and external brace types. In addition several parts of the computational engine is restructured to shorten computation time.

In addition to the modifications to the computational engine a new Graphical User Interface was developed. The Graphical User Interface is written in C++ language using OPENGL libraries for visualization. The GUI is in direct interaction with the computational engine. Data is transferred between two modules using the Random Access Memory, thereby providing no loss of accuracy. By making use of the GUI users can input data easily into the program and be able to post process the results. The GUI has the capability of displaying graphical and tabulated output. Users can post process for deflections, rotations, support reactions, shear flow, stresses, cross sectional forces and brace member forces. These quantities are given for placement sequence analysis as well as live load analysis. GUI also has the capabilities of displaying influence lines for desired quantities. Users can add truck and distributed live loads and can define load combinations.

Results obtained from the software are compared against closed form solutions and recommendations given by design specifications. Comparisons revealed that UTrAp provides excellent solutions for the test cases.

In conclusion, the developed software can be easily used in routine design practice. Complex bridges can be analyzed with minimal effort using this computer program. As is true for most of the software new capabilities can be added in the future. Some of the features that can be added to the program are as follows:

- Addition of user defined trucks
- Unit conversions
- Dynamic analysis capabilities
- Eigenvalue buckling analysis capability
- Addition of skew supports
- Addition of dapped ends
- Modeling of elevation difference between supports
- AASHTO design checks

REFERENCES

Topkaya, C., Williamson, E. B., Development of Computational Software for Analysis of Curved Girders under Construction Loads, Computers and Structures, 2003, Vol. 81, pp.2087-2098.

Topkaya, C., Williamson, E.B., and Frank, K.H., Behavior of curved steel trapezoidal box-girders during construction, Engineering Structures, 2004, No. 26, 721-733.

Popp, D. R., UTrAp 2.0: Linearized Buckling Analysis of Steel Trapezoidal Girders, M.S. Thesis, 2004, The University of Texas at Austin.

Kalaycı, A. S., Improvement of Computational Software for Composite Curved Bridge Analysis, M.S. Thesis, 2005, Middle East Technical University.

Viest , I. M., Colaco, J. P., Furlong, R.W., Griffis, L. G., Leon, R. T., Wyllie, L. A., Composite Construction Design For Buildings, 1997, ASCE Publications, McGraw-Hill.

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

Segal, M, Akeley, K., OpenGL Graphics System: A Specification, 1999, Silicon Graphics Inc., CA

APPENDIX A

USER'S MANUAL AND EXAMPLE PROBLEM FOR UTrAp

UTrAp is a computer program developed for analysis of curved/straight steel/concrete composite bridges having box or I section. Multiple girder systems with multiple segments with different curvatures can be analyzed with this program. The program consists of a Graphical User Interface (GUI) and an analysis module. The analysis module relies on the finite element method to compute the response of the three-dimensional bridge structure. Input data is supplied to the program by making use of the GUI. The program can handle multiple analysis cases and has graphics capability to visualize the output. In the following sections, details of the program are presented along with an example problem.

Example Problem Definition

The example problem presented herein is a 3-span, dual girder system with a centerline radius of curvature of 137m. The bridge is named as "Direct Connect Z" and has a centerline arc length of 156m. The plan view of the bridge is given in Fig. A.1.

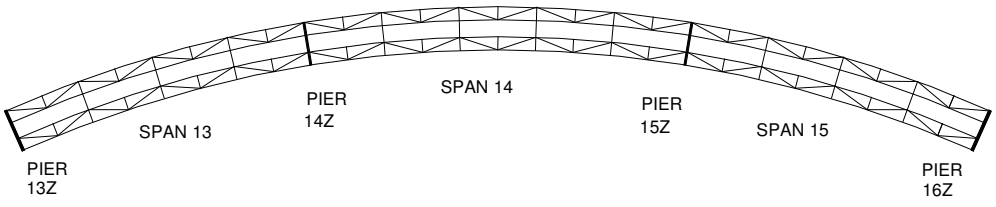


Figure A.1: Plan View of Direct Connect Z

In UTrAp, the radius of curvature can be variable along the bridge length. Positions along the bridge are defined by the distance along the arc length relative to the start end. In order to specify a bridge with a variable radius of curvature user has to input the segment length and the radius of curvature of the segment. Positive radius of curvature results in a concavity pattern shown in Fig. A.1 where the start end is pier 13Z. Negative values for the radius of curvature result in an opposite concavity pattern. A straight segment can be specified by assigning zero to the curvature.

Cross-sectional dimensions of the Direct Connect Z are given in Fig. A.2. Web depth is measured between the centerline of top and bottom flanges. Centerline of each girder is offset by 250 cm from the bridge centerline. The concrete deck width and thickness are 900 cm and 25 cm, respectively.

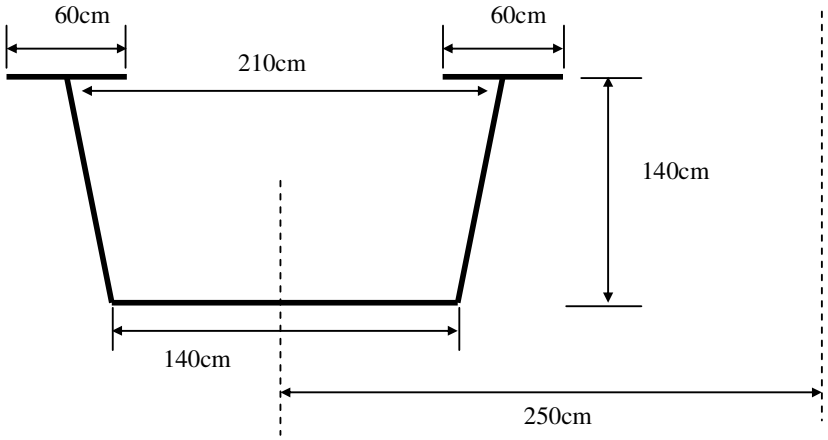


Figure A.2: Cross-sectional Dimensions

The steel plates that make up the girder have variable thickness along the length of the bridge. Table A.1 provides the details of the plate thickness. Lengths given in this table are the centerline arc lengths. Properties are listed beginning from the start end of the bridge. In this program all girders must have the same plate thickness properties.

Table A.1: Plate Properties

WEB		BOTTOM FLANGE		TOP FLANGE	
Length(m)	Thickness(cm)	Length(m)	Thickness(cm)	Length(m)	Thickness(cm)
32	1.2	32	2.0	40	3.2
31	1.6	8	3.2	3	4.5
30	1.2	4	4.0	9	7.0
31	1.6	8	5.0	3	4.5
32	1.2	3	4.0	46	3.2
		8	3.2	3	4.5
		30	2.0	9	7.0
		8	3.2	3	4.5
		3	4.0	40	3.2
		8	5.0		
		4	4.0		
		8	3.2		
		32	2.0		
Σ = 156 m		**Σ = 156 m**		**Σ = 156 m**	

Bracing members are provided throughout the girder. Internal, external and lateral braces are present. Locations of the braces are given in Table A.2. For internal and external braces, only one location value is required. For lateral braces, the start and end location of each brace is needed.

Table A.2: Location of Braces

	Internal Bracing	**External Bracing**	**Lateral Bracing**	
Brace Number	Location (m)	Location (m)	Start Location (m)	End Location (m)
1	6	12	0	6
2	12	24	6	12
3	18	36	12	18
4	24	60	18	24
5	30	72	24	30
6	36	84	30	36
7	42	96	36	42
8	54	120	42	48
9	60	132	48	54
10	66	144	54	60
11	72		60	66
12	78		66	72
13	84		72	78
14	90		78	84
15	96		84	90
16	102		90	96
17	114		96	102
18	120		102	108
19	126		108	114
20	132		114	120
21	138		120	126
22	144		126	132
23	150		132	138
24			138	144
25			144	150
26			150	156

There are 23 internal and 26 lateral braces per girder. In addition, there are 10 external braces between the two girders. Internal braces are in the form of K-trusses, which have members with cross-sectional area of 25.0 cm^2. All lateral braces have a cross-sectional area of 40 cm^2, and their orientation is given in Fig .A.1. External braces are comprised of truss members with a cross-sectional area of 30 cm^2. Details of their configuration are provided in Fig. A.1.

The bridge has four supports which are located 0, 48, 108, and 156 meter away from the start end. Studs are spaced every 30 cm at both ends of the bridge for a distance of 3 m from the pier. For the remainder of the bridge, studs are spaced at every 60 cm. There are 3 studs per flange over the entire length of the bridge.

The concrete deck is placed in 5 segments. The lengths and the sequence of placements are given in Fig. A.3.

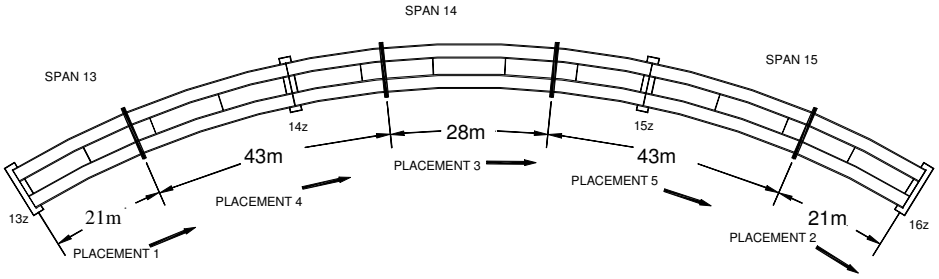


Figure A.3: Concrete Placement Sequence

This analysis example will consider all placement sequences. The program requires the lengths of the placements and number of analysis to be performed. In this example, 5 analysis cases will be considered.

In the first analysis, the concrete deck is placed on the first segment, and a uniform loading of 0.055t/cm is applied on that segment to account for the concrete self-weight. In the second analysis, it is assumed that the concrete on the first segment has cured and attained a stiffness of 70t/cm^2 with a corresponding stud stiffness of 45t/cm. A uniform loading of 0.055t/cm is applied to the second segment due to the concrete weight. In the third analysis, it is assumed that the concrete modulus and stud stiffness have reached to 140t/cm^2 and 90t/cm, respectively, for the first segment. For the second

segment, the concrete and stud stiffness values are assumed to attain values of 70t/cm^2 and 45t/cm, respectively. A uniform loading of 0.055t/cm is applied to the third segment to account for concrete weight. Finally for the fourth and fifth analysis, a uniform load of 0.055t/cm is applied to the respective segments. It is assumed that segments one, two and three have concrete stiffness of 200t/cm^2 and stud stiffness of 130t/cm.

The summary of analysis parameters are given in Table A.3.

Table A.3: Pour Sequence Analysis Parameters

		Analysis 1			Analysis 2			Analysis 3			Analysis 4			Analysis 5		
Deck	Length	Con. Mod.	Std. Stf.	Load	Con. Mod.	Std. Stf.	Load	Con. Mod.	Std. Stf.	Load	Con. Mod	Std. Stf.	Load	Con. Mod.	Std. Stf.	Load
1	21	0	0	0.055	70	45	0	140	90	0	200	130	0	200	130	0
2	43	0	0	0	0	0	0	0	0	0	0	0	0.055	0	0	0
3	28	0	0	0	0	0	0	0	0	0.055	200	130	0	200	130	0
4	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.055
5	21	0	0	0	0	0	0.055	70	45	0	200	130	0	200	130	0
	Σ=156m															

In addition to placement sequence analysis, live load analysis is also performed. For live loading analysis, two input sets are required; these are lane definitions and deck properties for live loading analysis. In this example problem, only one lane having a width of 3m is defined (Extends from -1.5m to 1.5m from the centerline of the bridge deck). For the deck properties, stud stiffness and concrete modulus are entered for the user defined bridge segments. In this case, only one segment along the bridge is defined with stud stiffness 130 t/cm, and concrete modulus 200t/cm^2.

Table A.4: Lane Definitions

Lane No	Start Location (m)	End Location (m)
1	-1.5	1.5

Table A.5: Live Loading Parameters

Segment Number	Length(m)	Concrete Modulus (t/cm^2)	Stud Stiffness (t/cm)
1	156	200	130

To perform live load analysis, live load cases should be defined. A live load case has two parameters: the first one is the load definition and the other one is the lane on which the load applied. Load type can be moving load (i.e. truck type) or distributed load over the lane defined.

In this analysis, two live load cases, LL1 and LL2, will be defined. One for truck type and one for distributed load. For truck type live load case LL1, standard HS20 truck is selected. For second live load case LL2, 0.01t/cm distributed load extending from 0m to 108m along the bridge length.

The following representative load combinations are going to be defined for this bridge example.

LC1 = (PS1 + PS2 + PS3 + PS4 + PS5) * 1.0

LC2 = LC1 + 1.5 * LL1

LC3 = LC1 + 1.5 * LL2

Where, PS stands for nth placement sequence.

USER'S GUIDE AND SOLUTION OF THE EXAMPLE PROBLEM

The Graphical User Interface of UTrAp has a total of 12 menus. This section will explain each of these menus in detail. Use of these menus will be presented along with the example problem.

File Menu: This menu has seven submenus and is used for data management. Files can be stored and retrieved by making use of this menu. Details of each submenu are as follows:

New: This submenu starts a blank project. If a new bridge model is going to be formed, this option should be selected.
Open: This submenu is used to open an existing project. The UTrAp project files have an extension of **.axd*. When the existing project submenu is invoked, an open file box will appear which is used to select the existing project file.
Save: This submenu is used to save a project to the hard disk. It can be used to save the changes made to an existing project or the contents of a newly developed project. When the *Save* submenu is invoked, if project is previously saved, project is saved with current state, if not saved previously then save file box will appear which is used to name or rename the project file.
Save As: This submenu has a similar function to the *Save* submenu. However, this button always opens the save dialog box and gets a new project name.
Import: This submenu imports UTrAp input file in text format and apply these inputs to the current project.
Export: This submenu exports the current project to UTrAp input file in text format.
Exit: This submenu is used to exit the program.

Example Problem: A new project is formed by making use of the *New* submenu.

View Menu: This menu has two submenus,
Create View: This submenu is used to create a new view of the finite element model.
Arrange: This submenu has 3 submenus; it is basically used to arrange multiple views that are present. Three options are available; Cascade, Tile Horizontal, Tile Vertical.

Example problem: When a new project is selected, the program automatically generates a view window; therefore this menu does not have to be used unless multiple views are required.

General Menu: This menu is used to input the geometric properties of the bridge. Values should be typed in the boxes provided. After entering all the required data, the user must press *OK* button to save and exit the form. To exit without saving the data entered, *Cancel* button must be pressed. This data saving process is valid for all subsequent forms.

Example Problem: Geometric property values are entered on the form and saved by making use of *OK* button. Fig. A.4 shows the Geometric Properties form with the entered data.

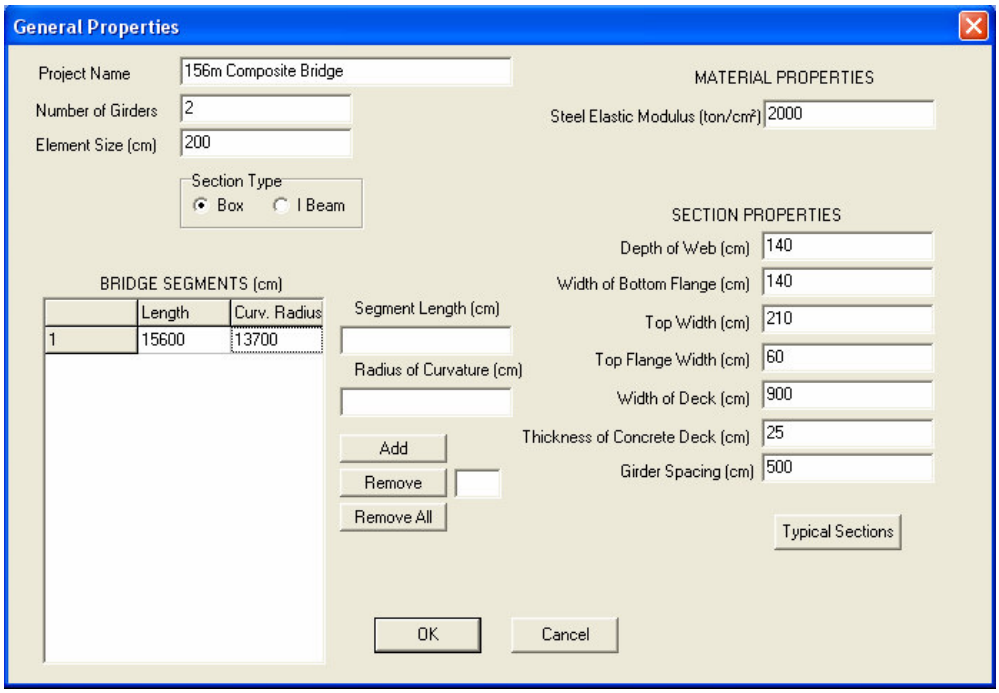


Figure A.4: General Properties Form

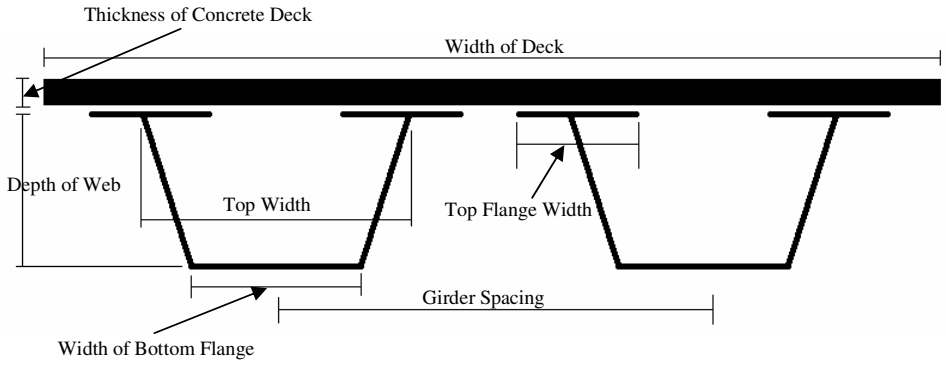


Figure A.5: Box Girder Dimensions

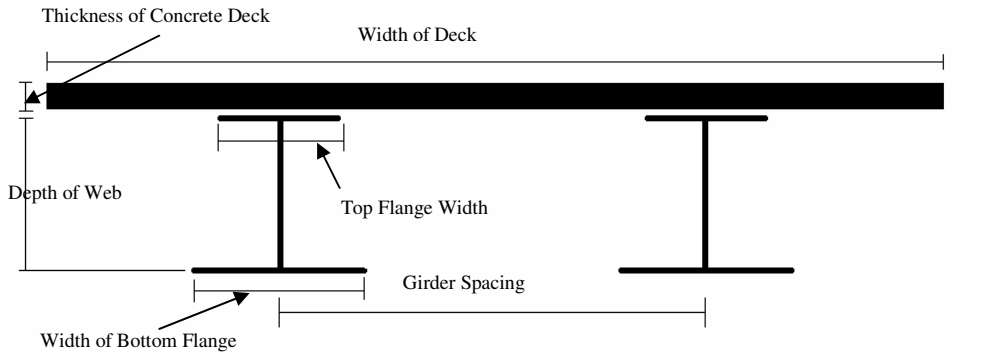


Figure A.6: I Girder Dimensions

The element size along the bridge length needs to be entered in the general properties dialog box. It is recommended to have an element size less than 150cm and values less than 100cm are desired. Is is also recommended that the element size be chosen so as to ensure that the distance between internal brace panels be an integer multiple of the element size. Users can input segment properties by using the *Add/Remove* buttons provided on the form. To remove all the segments, *Remove All* button can be used.

The typical box and I girder dimensions can be viewed via pressing the *Typical Sections* button in the dialog. In Fig. A.5 and Fig. A.6 these typical sections are shown.

Plate Properties Menu: This menu is used to input the plate properties related with the bridge. The plate properties form has three folders. Each folder is reserved either for the web, the bottom flange or the top flange properties. Properties are input in a tabular form. The length of the plate and its thickness should be entered from the start to the end of the bridge. There are two buttons used to add and remove properties. Their function is explained below.

Add: This button is used to add properties. A change in plate thickness requires the user to specify a new property. The user should enter the number of properties that will be needed to characterize the bridge. After, the number of rows in the table is increased by the total number of properties specified by the user.

Remove: This button is used to remove properties. The property number that is going to be removed should be specified in the box next to the *Remove* button.

Example Problem: In each folder, the number of properties is increased by the *Add* button. All plate properties are entered in a tabular format. A representative input for bottom flange plate properties are given in Fig. A.7.

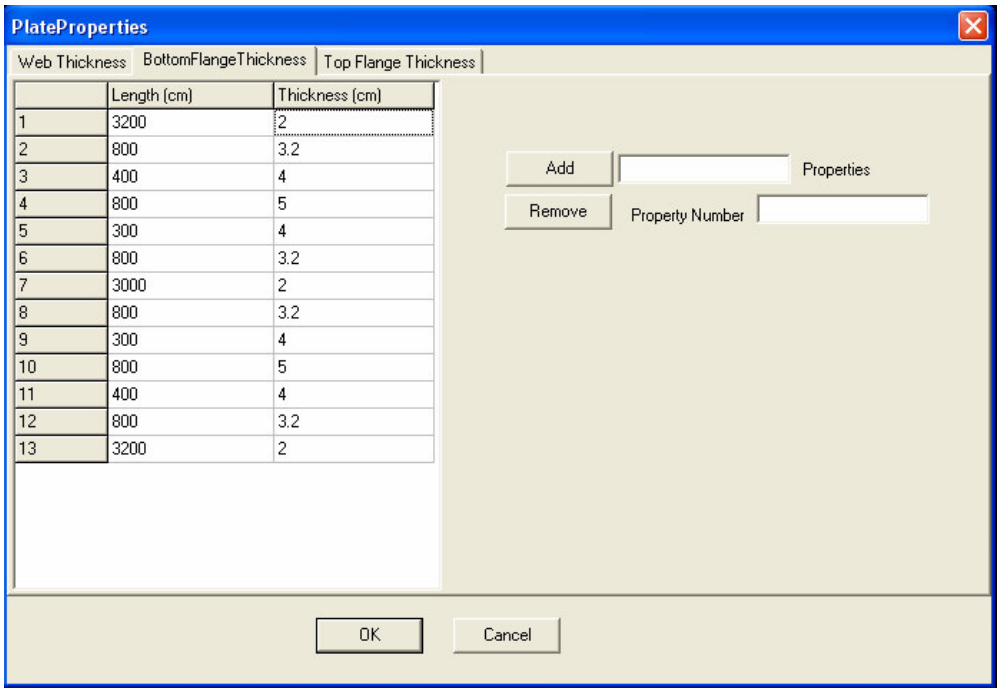


Figure A.7: Plate Properties Form

Bracing Menu: This menu is used to input bracing information related with the bridge. The brace properties form has three folders. Each folder is reserved for either the internal, external, or the lateral brace properties. Properties are input in a tabular form. Depending on the version of the program, different geometrical types of braces can be specified for internal and external braces. For internal braces, location, type, gird/spacing (For box type girders, girder index, for I type girders, spacing index) and member cross-sectional area information are required. Location, type, spacing and member cross-sectional area information are required for external braces. The type, spacing, start location, end location, and cross-sectional area are required for the lateral braces. There are buttons provided to add and remove braces. Functions of the buttons are explained below.

Add: This button is used to add braces. The user should enter the number of braces that will be added to the box next to the *Add* button. The number of rows in the table is increased by the corresponding number entered by the user.

Equally Space: This button is used to add braces at equally spaced intervals. The number of braces to be added is specified in the box next to the button. For this button to function properly, two more location values must be entered. Braces are placed at equal intervals between these values.

Remove: This button is used to remove braces. The brace number that is going to be removed should be specified in the box next to the *Remove* button.

Remove All Braces: This button is used to remove all the braces specified previously in a certain folder.

Type: This button is displayed in the internal and external braces folder. It is used to assign the same type to all braces. The type of the brace should be entered into the box next to this button. The available bracing types and their configurations are displayed in a separate form using *Show Internal/External Brace Types* buttons.

Area: This button is used to assign the same cross-sectional area value to all brace members. The cross-sectional area value should be entered into the box next to this button.

Show Internal/External/Lateral Brace Types: These buttons are used to display the types of braces that a user can specify in the program. When this button is pressed, a form that shows the geometry and types of braces are displayed on the screen. Fig. A.8 shows the types of internal and external braces supported by the current version of the program.

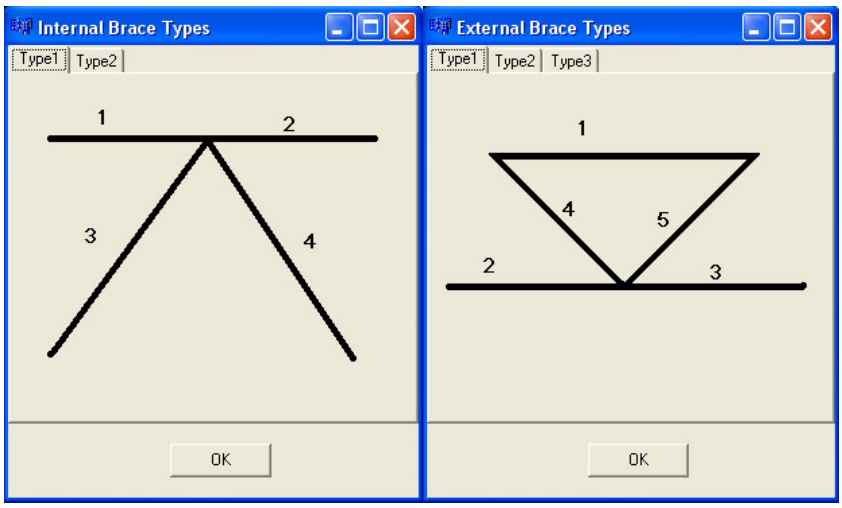


Figure A.8: Internal and External Brace Types

All Type: This button is displayed only in the lateral braces folder. It is used to assign a type written in the edit box just right of the button to all lateral braces. Lateral braces can have only four orientations. Therefore, there are four types of lateral braces which are shown in Fig. A.9. Lateral braces of type 1 and 2 are connected to top flange. This type bracing can be applied to both box and I type girders. Type 3 and 4 braces are connected to bottom flange. Unlike type 1 and 2, these types of lateral braces can only be applied to I girders.

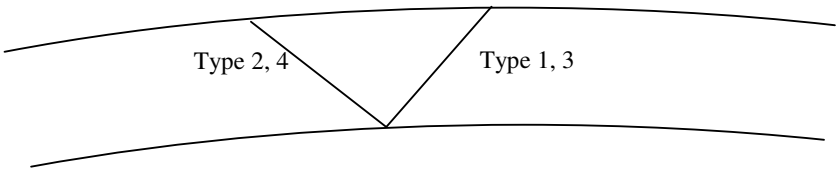


Figure A.9: Lateral Brace Types

Alternating Lateral Brace Type: This button is displayed only in the lateral braces tab. It is used to assign alternating types to consecutive braces. The first brace will be of type written in the edit box just right of the button, and second brace will be of type written in the other edit box, etc.

Example Problem: 23 internal braces and 26 lateral braces for each girder (i.e. total of 46 internal and 52 internal braces), 10 external braces are added to the folders by making use of the *Add* button. Brace locations, types, spacing and cross sectional areas are entered into the folders according to the information given in Table A.2. All internal and external braces are type 1. Lateral braces have alternating types starting with type 2. In Fig A.10 and Fig. A.11, two tabs of the bracing properties form are shown.

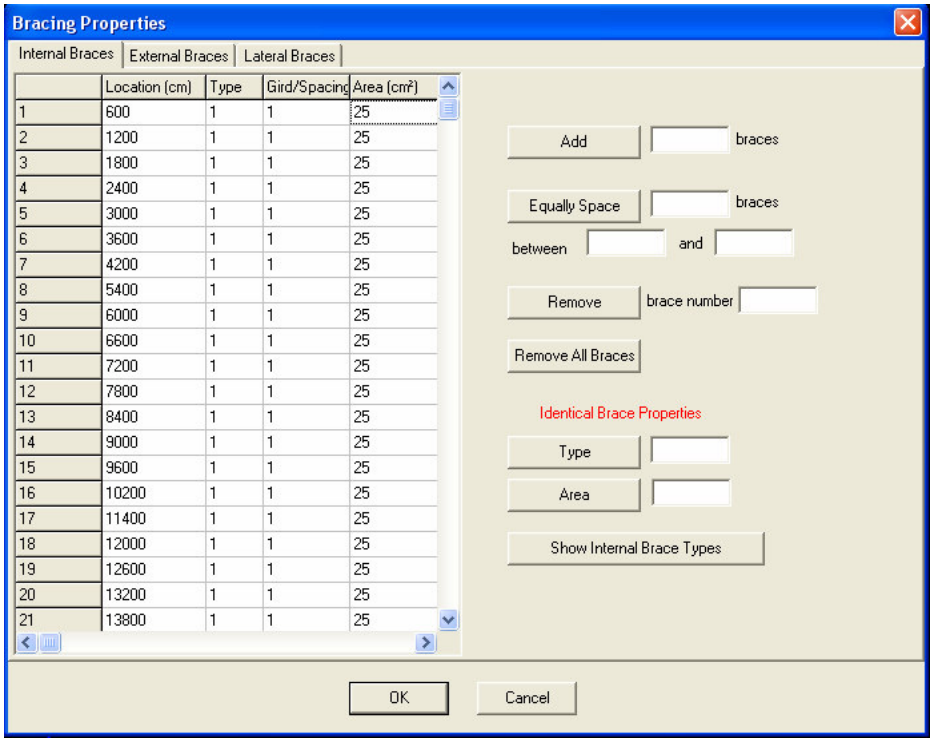


Figure A.10: Bracing Properties Form - Internal Braces Tab

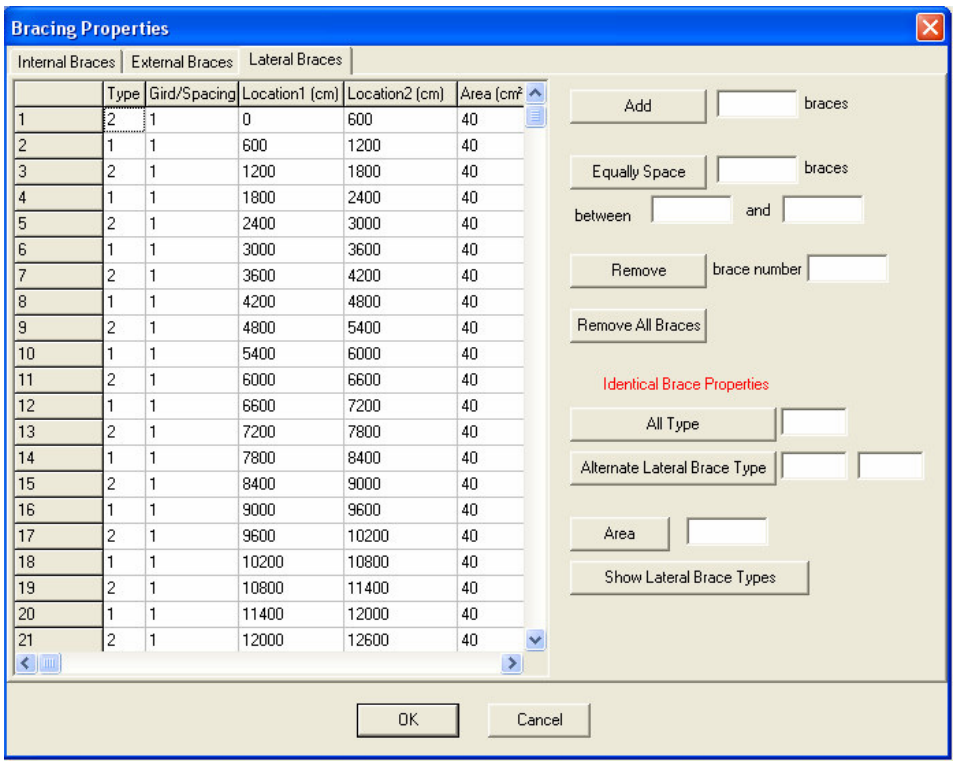


Figure A.11: Bracing Properties Form – Lateral Braces Tab

Support Menu: This menu is used to input support locations. Locations are input in a tabular form. The program assumes that only one of the supports is pinned and the rest are rollers. The first support specified is considered to be the pinned one. Number of rows of the tabular input form is controlled by the *Add* and *Remove* buttons. Functions of the buttons are explained below.

Add: This button is used to add supports. The user should enter the number of supports that will be added to the box next to the *Add* button. The number of rows in the table is increased by that specific amount.

Remove: This button is used to remove supports. The support number that is going to be removed should be specified in the box next to the *Remove* button.

Example Problem: Four supports are added to the table by making use of the *Add* button. Support locations given in the description of the bridge are entered on the table. Fig. A.12 shows the support locations form along with the entered data.

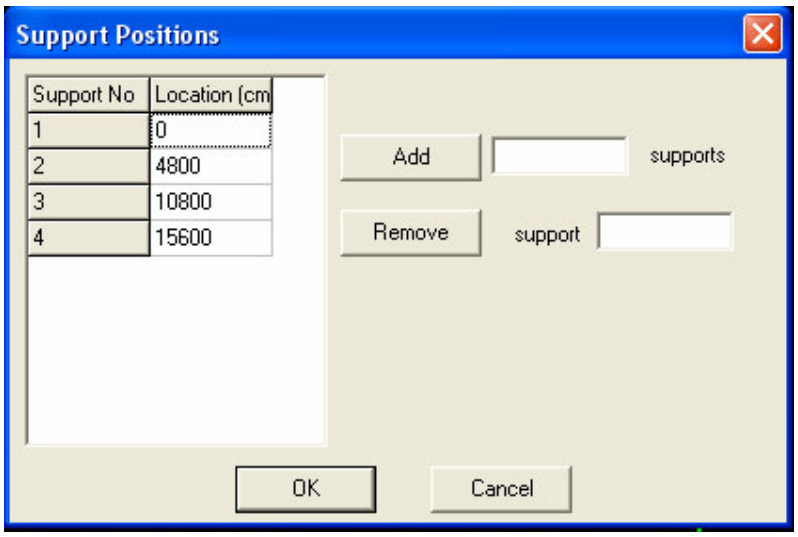


Figure A.12: Support Locations Form

Stud Menu: This menu is used to input stud properties. Properties are input in tabular form. Spacing of the studs and the number of studs per flange should be supplied to the program along the bridge length. The number of rows of the tabular input form is controlled by the *Add* and *Remove* buttons. Functions of these buttons are explained below.

Add: This button is used to add properties. The user should enter the number of properties that will be added to the box next to the *Add* button. Number of rows in the table is increased by that specific amount.

Remove: This button is used to remove properties. The property number that is going to be removed should be specified in the box next to the *Remove* button.

Example Problem: For this problem, stud properties change three times along the bridge length. Therefore, three rows are added to the table by making use of the *Add* button. Cells of the table are filled according to the geometry information given in the bridge description. Fig. A.13 shows the stud properties form along with the entered data.

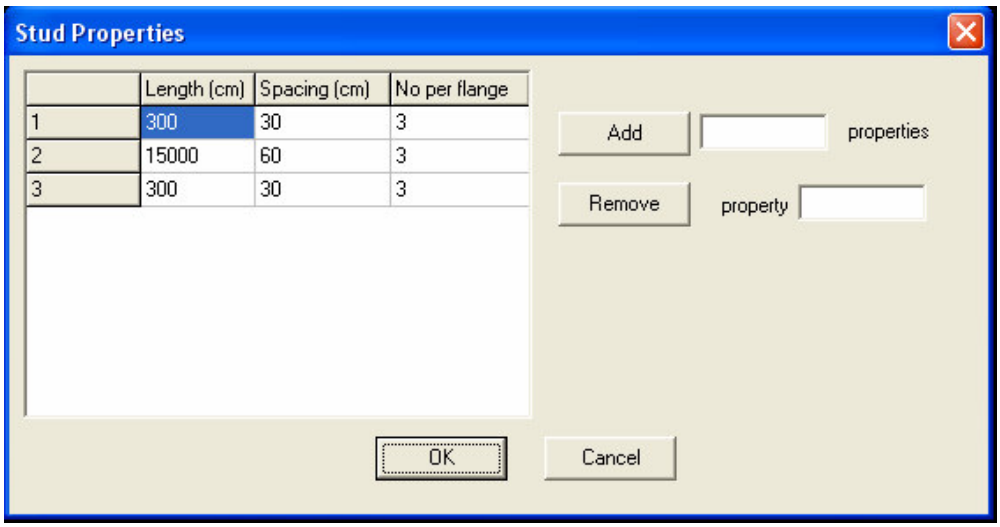


Figure A.13: Stud Properties Form

Analysis Menu: This menu has five submenus. These are *Options, Placement Sequence, Live Load Parameters, Live Load Cases and Load Combinations*.

1. **Options:** This menu is used to input element type, solver type analysis type, live load placement interval, moving load step and the exclusion flag for external braces in live loading analysis.

   All this information must be supplied by the user. Bridges can be modeled using 9-node shell element (Element type 1) or 4-node shell element (Element type 2). It is recommended to use Type1 elements. Depending on the types of analysis user has to choose between "placement sequence" analysis and "live load analysis". Both analyses can be performed in the same run also.

   The precision of the influence lines can be controlled by the live load interval parameter. The most accurate one is to have load interval at every element. If higher values are used the run time will be shorter, however, the results for live load analysis may be imprecise.

   The moving load step variable is used to consider the different locations of truck that is passing over the bridge. The axle loads of truck are displaced by this value along the bridge.

   As it can be understood from the caption, *Exclude external braces in Live Load Analysis* check box is used to remove external braces during live loading analysis.

Example problem: In this problem, element type 1 and solver type 1 are selected. Moving load step and element interval for live load placement are chosen as 50cm and 2 elements respectively as shown in Fig. A.14. External braces in live load analysis are excluded. Analysis type is chosen as both that is live load and placement sequence analysis are going to be performed.

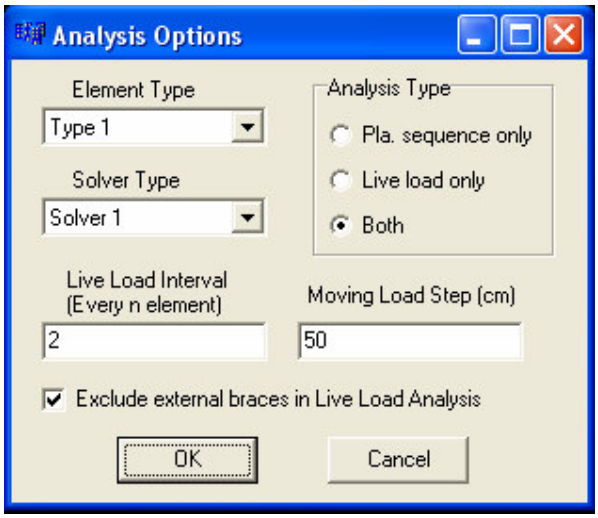


Figure A.14: Analysis Options Form

2. **Placement Sequence Menu**: This menu is used to input placement sequence analysis parameters. Parameters are input in tabular form. The concrete deck can be divided into segments corresponding to each placement, and there can be multiple analyses that are independent from each other. For each analysis, properties of deck segments and loading on the segments should be provided as input. Properties for a deck segment include the stiffness of concrete and the stiffness of studs. Lengths of the deck segments are the same for all analyses and their values should be given as input. The tabular form is controlled by four buttons. These buttons are used to add and remove columns and rows to the table. Functions of the buttons are explained below.

Add Analysis Case: This button is used to add a new analysis case to the table. Three columns for analysis parameters are added to the right of the table each time a new analysis is added.

Remove Analysis Case: This button is used to remove a specific analysis case. The analysis number that is going to be removed should be entered into the box next to this button. Three columns related with the analysis number specified are removed from the table.

Add Deck Property After: As mentioned before, the concrete deck can be divided into segments. At least one deck property must be specified. This button is used to add a new deck property row to the table. The new deck property is added after the deck number specified in the box next to this button. If no deck has been defined in the table previously, a value of zero should be used. Specifying a value of zero adds blank cells to the first row.

Remove Deck Property: This button is used to remove a deck property row. The number of the deck property to be removed should be entered into the box next to this button. The specified row is deleted from the table.

Example Problem: In this problem, the concrete deck is divided into five segments. These deck segments are added to the table by making use of the *Add Deck Property After* button. There are a total of five analyses to be performed. These analysis cases are added to the table by using the *Add Analysis Case* button. The table is filled with parameters specified in Table A.3. Fig. A.15 shows the placement sequence form together with the input data.

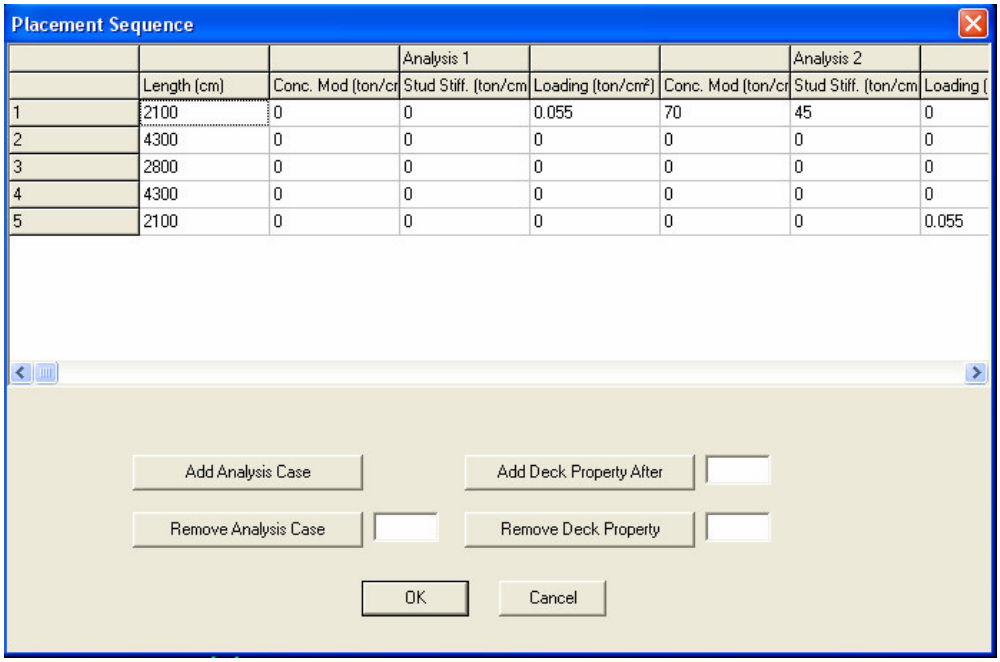


Figure A.15: Placement Sequence Form

3. **Live Load Parameters:** This menu is used to input the live loading parameters. In live loading parameters dialog, two tabs are present; one is the lane definition, the other one is the live loading parameters. In both tabs, using *Add* button new lane and deck properties can be added and with *Remove* button, chosen rows can be deleted.

Example Problem: In this problem, one lane and one live loading segment are defined as shown in Fig. A16 and Fig. A.17.

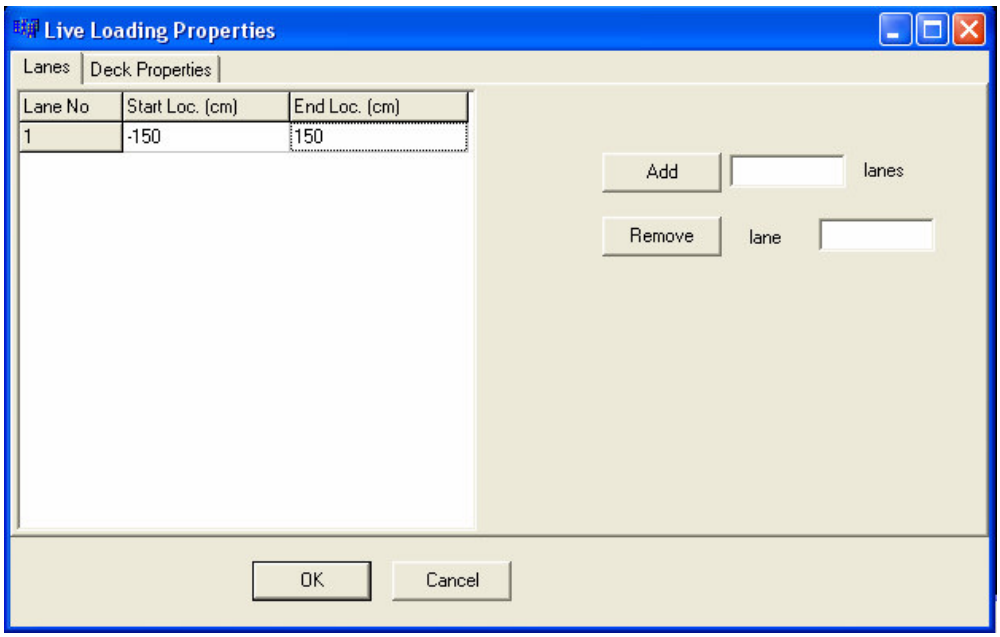


Figure A.16: Lane Definition Form

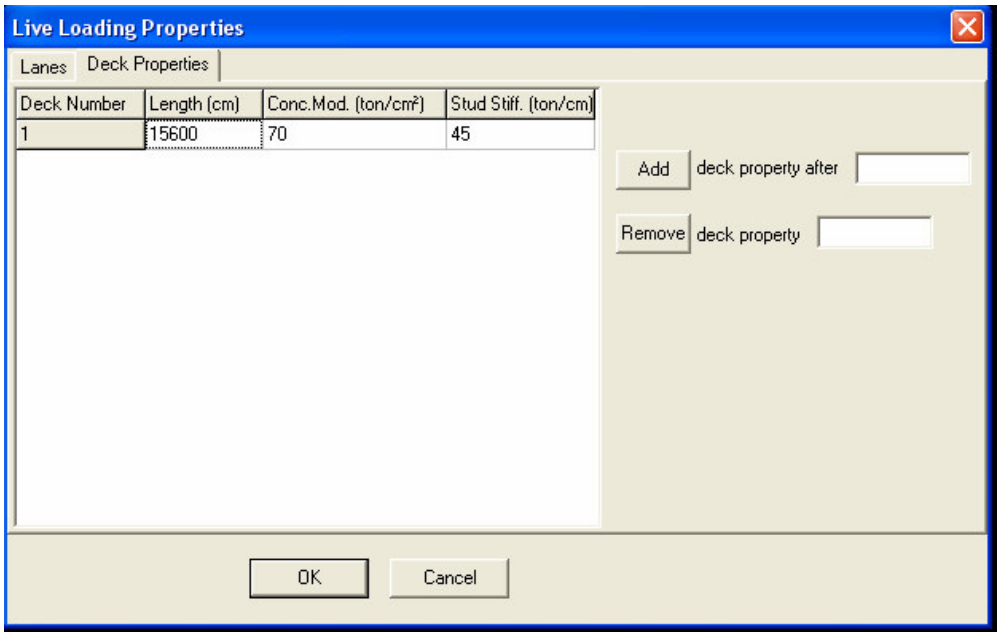


Figure A.17: Live Loading Parameters Form

4. **Live Load Cases:** This menu is used to define live load cases. Live load case dialog has three sections; name of the live load case, lane on which live loading is applied, loading type. In the loading section three radio buttons are used to define whether live load case is a distributed load, truck type loading or point load. For distributed loading, three edit boxes are present. These are for distributed load quantity, start location and end location of distributed load. For truck type loading, a combo box is present to select the type of truck. For point load, an edit box is present to get the quantity of point load.

   Add: This button is used to add new live load case after entering the required inputs.

   Remove: This button removes the previously defined live load case selected in live load list box.

   Update: Update button updates the properties of the selected live load case.

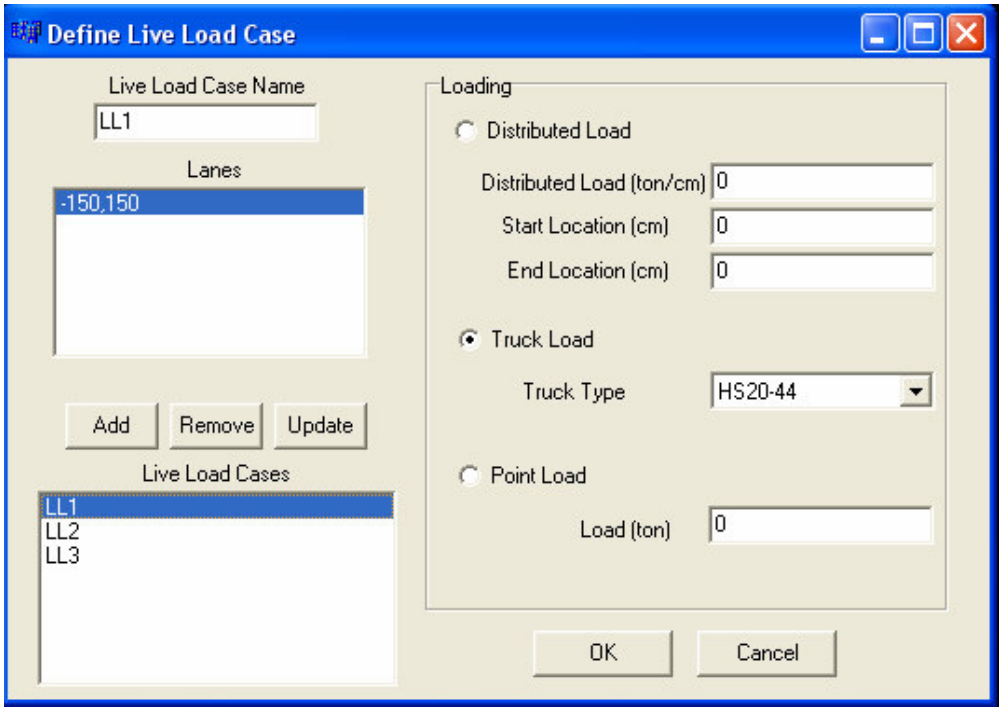


Figure A.18: Live Load Case Definition Form (For Truck Type)

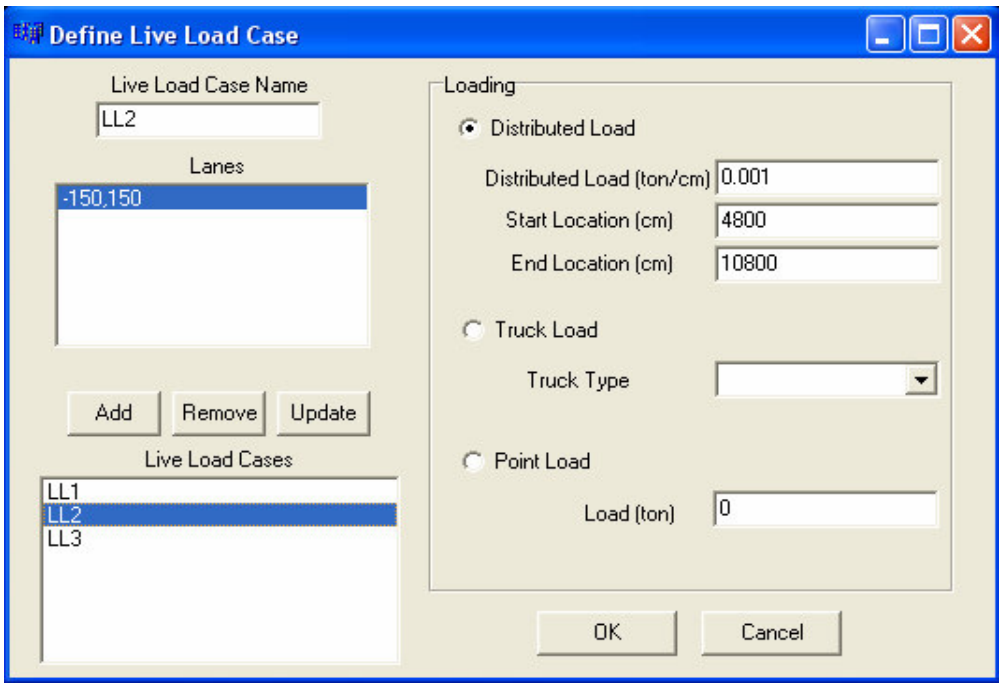


Figure A.19: Live Load Case Definition Form (For Distributed Load Type)

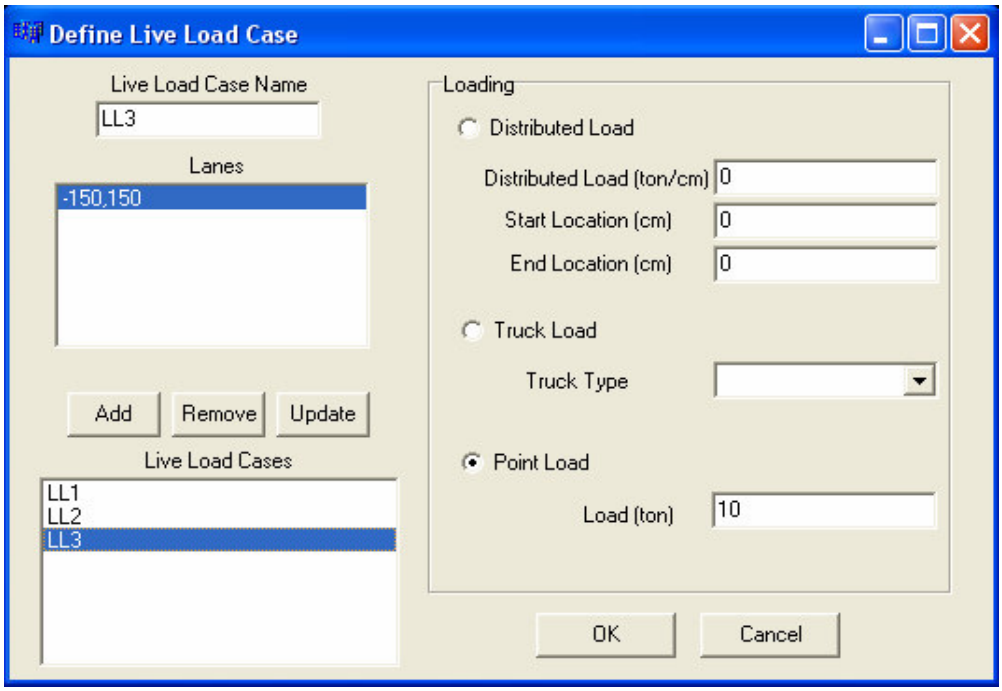


Figure A.20: Live Load Case Definition Form (For Point Load Type)

Example Problem: In this example, three live load cases named LL1, LL2, and LL3 are defined. To create a live load case, first, *Live Load Case Name* edit box has to be entered and Add button is pressed. Then, lane should be selected. At last, type of live load is selected. For LL1, *Truck Load* radio box is checked and HS20 is selected from *Truck Type* combo box. After these operations, *Update* button is pressed to assign these properties to LL1. In a same manner, LL2 is created and assigned a distributed type of load with a value of 0.01t/cm extending from 48 to 108m along the bridge. To assign a distributed load to a live load case, user should check the *Distributed Load* radio box. LL3 is created like the other live load cases but *Point Load* radio box is selected. For quantity of point load, 10tons is entered in the *Load* edit box.

5. **Load Combinations:** This menu is used to define load combinations incorporating placement sequences and live load cases with individual magnifying factors.
   Create: This button creates a new load combination with the name entered in the *Combo Name* edit box.
   Remove: Remove button deletes the load combination selected from the list.
   Update: This button updates the name of load combination selected from the list.
   Add/Remove: After creating a load combination, selected load case can be added to that combination using add/remove buttons at the left of the dialog box. To add a load case, the target load combination should also be selected. While adding a load case to a combination, magnification factor should be entered.

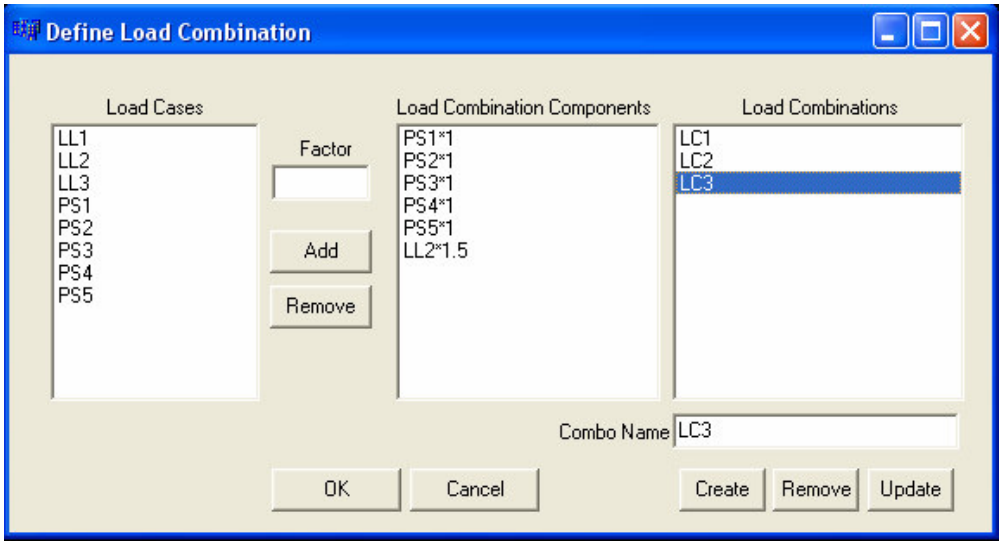


Figure A.21: Load Combination Definition Form

Example Problem: To define a load combination, user should enter a name into *Combo Name* edit box and then press *Create* button. To modify a created load combination, the load combination should be selected from the *Load Combinations* list box and then by using the *Add/Remove* buttons at the left of the dialog box, load cases can be added to or removed from the load combination. For this problem, three load combinations defined. First load combination incorporates all placement sequence load cases. Second one includes all placement sequence too, but in addition, LL1 load case with a factor of 1.5. Third load combination is same with the second one but, instead of LL1, LL2 is included.

Generate Menu: This menu generates the three dimensional finite element model for the current project. All the geometric and material properties have to be supplied before invoking the *Generate m*enu.

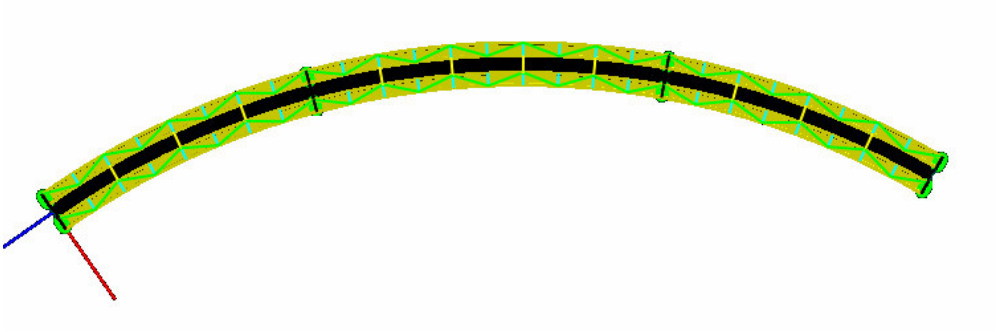


Figure A.22: General View of Finite Element 3d Model

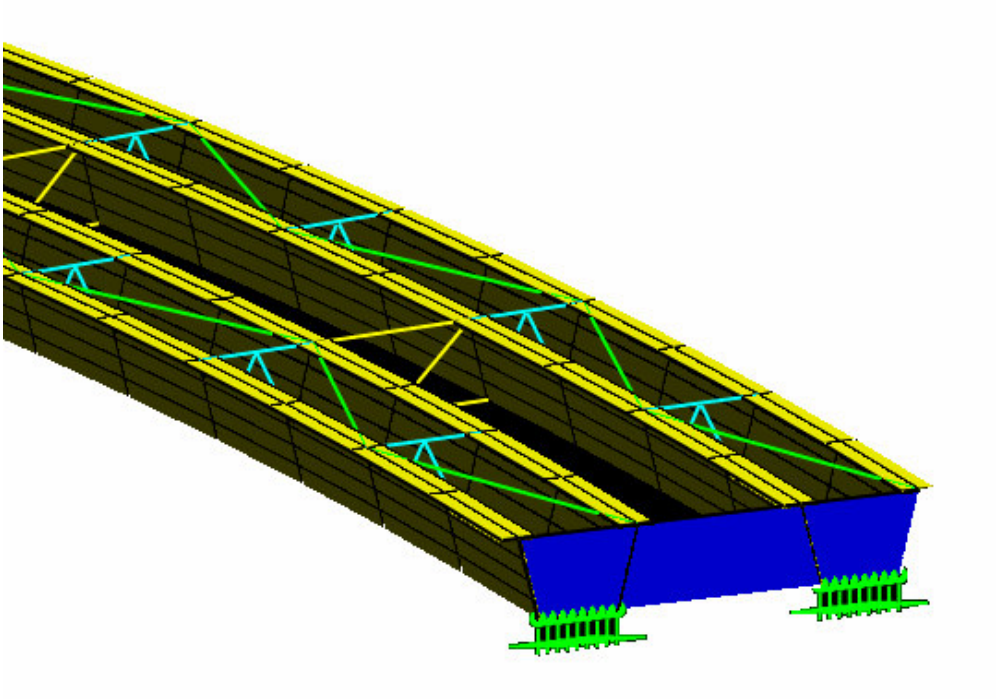


Figure A.23: Perspective View of Finite Element 3d Model (Without Deck)

In the view window, there are view buttons to toggle of visibility of visual objects and to look from axes.

Table A.6. Tool Buttons of View Window and Functions

Tool Button Icon	Function of the Button
	Toggle for node visibility
	Toggle for shell visibility
	Toggle for shell outline visibility
	Toggle for external brace visibility
	Toggle for internal brace visibility
	Toggle for lateral brace visibility
	Toggle for deck visibility
	Toggle for support visibility
	Sectional view
	Side view
	Top view

In the graphical user interface, user can manipulate the view using mouse buttons.

User can rotate the model by pressing the left mouse button and moving the mouse to desired rotate direction at the same time. To activate the rotate mode, user must press F8 button of keyboard before rotate operation. To deactivate rotate mode, F7 button must be pressed.

Using mouse roller button, user can zoom in and out the model. If mouse roller button is rolled to front, the model will be zoomed out, i.e. it will get smaller. On the contrary, to look at the model closer, user must roll the roller button to back.

To slide the model, user must press the middle mouse button and move the mouse to the desired slide direction.

Also, there is a view adjustment dialog box to adjust the view settings. Appearance of the dialog box is shown in Fig. A.23. In the dialog box, the upper three edit boxes are for the rotation degree of the model in the view (i.e. model is first rotated

about X axis, as much as the value written in *X* edit box, then *Y* value is applied for rotation about Y axis. At last model is rotated about Z axis as much as the Z value.

The lower two exit boxes are used for the definition of the clipping plane positions perpendicular to the screen. The objects only in the range between NearZ and FarZ values are drawn in the view.

At the bottom, there is a button to change the background color. User can change the background color by clicking on the button and picking a color from the dialog box appearing immediately after clicking.

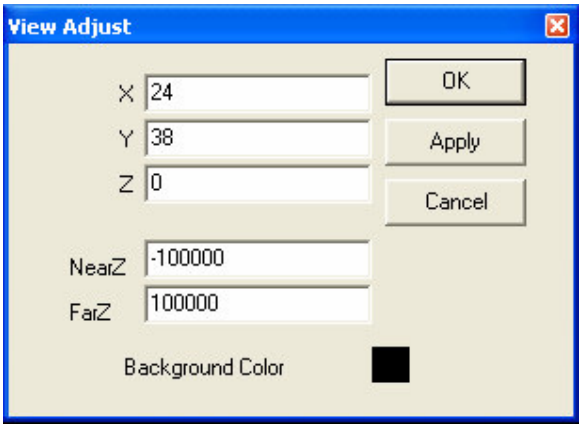


Figure A.24: View Adjust Dialog Box

Run Menu: This menu runs the analysis with the all given parameters. This run button will be disabled after the run is completed. Following the end of the analysis, a log viewer form as in Fig. A.25 will appear. In that log viewer, steps of analysis can be followed. User has to make sure that the log viewer finally reports that the "Analysis has been completed successfully". After the end of the analysis, project is also locked to avoid any changes in the input variables of the project. At the lock state, all the input boxes (i.e. edit boxes, grid controls, etc.) are set to read-only mode.

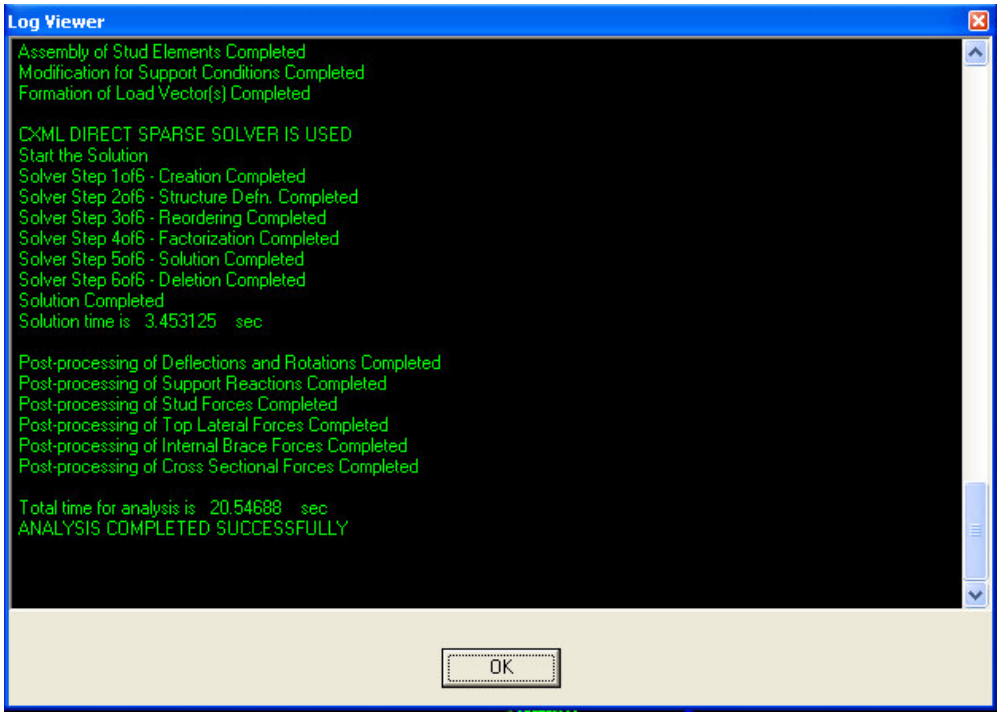


Figure A.25: Analysis Log Viewer

Results Menu: Using this menu, results of the analysis can be obtained. This menu has 4 submenus; P*lacement Sequence*, *Live Load*, *Influence Lines*, *Load Combinations*. Under these submenus, a number of submenus are present, such as, displacements, rotations, stresses, support reactions, etc.

- Placement sequence analysis results gives the displacements, rotations, forces and stresses in the bridge upon the placement of the concrete deck according to the predefined placement sequence scenario.
- Live load menu provides the results of the automatically generated unit loads on lanes along the bridge. These results are only used for program verification checks.
- Using Influence line menu, user can get the influence lines of displacements, rotations, cross-sectional forces, stresses, braces, support reactions and shear flow at specified positions or elements.

- Load Combination Results menu gives the results of the placement sequence cases, live load cases and linear combination of the results according to the defined load combinations.

The results of the analysis are displayed by graphs and/or tables. If the desired analysis result is a result of a placement sequence or a load combination including no live load case with truck that result will be shown with only one line graph and one column of tabulated result. Whereas, for the results of live load case or load combinations including at least one live load with truck, there will be two graphs and two columns in the table for maxima and minima of the result. These maxima and minima values are obtained while the truck is moved along the bridge.

In this manual, only placement sequence results will be presented. Results for live loading, influence lines, load combinations can be obtained in the same manner.

Placement Sequence: Using this sub menu, results of placement sequence can be viewed. Under this menu, there are eight submenus as shown in Fig. A.26.

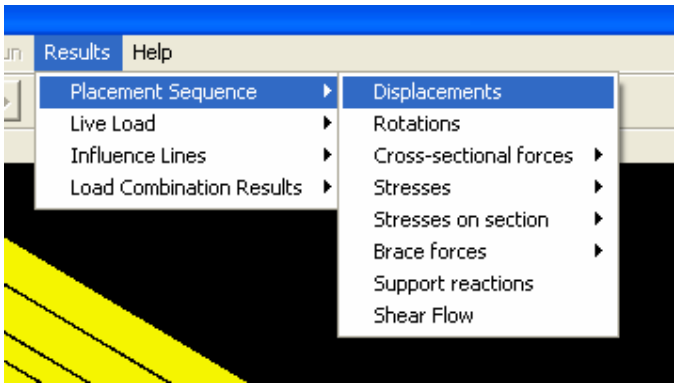


Figure A.26: Placement Sequence Submenu

Under Cross-Sectional forces menu, there are three sub items such as V (Section Shear), M (Section Moment), and T (Section Torsion). For Stresses and Stresses on section menus, there are two sub items; S11 (Stress normal to section), S12 (Stress

perpendicular to the section). Brace forces menu also has three sub items for Internal Braces, External Braces, and Lateral Braces.

Displacements: Using this menu, vertical deflections of the girders can be obtained. After clicking the menu, a dialog for selecting the run number and girder will appear as in Fig. A.27. Using this dialog, results for multiple runs and girders can be displayed. In this example, only first run and first girder are selected.

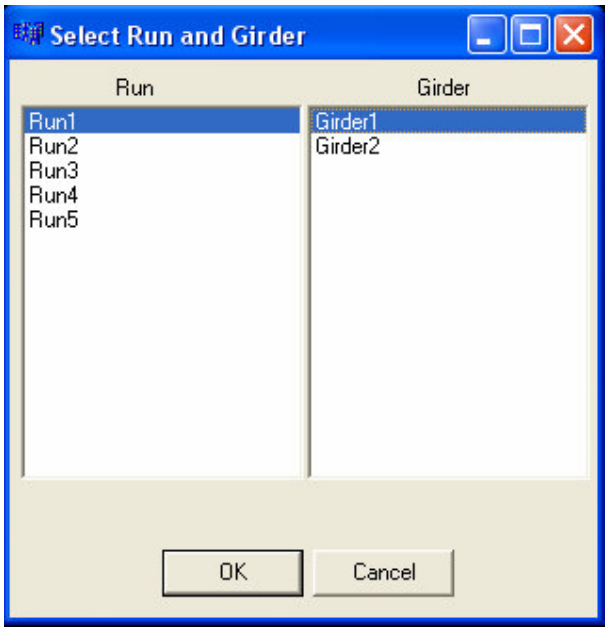


Figure A.27 Run and Girder Selection for Displacements

After pressing *OK* button, a dialog box as in Fig. A.28 showing the displacement of the selected girder under first placement sequence is displayed. In this dialog box two tab pages are present. First tab page is showing the graph of the displacement under given loading along the bridge. On the second tab page, there are tabulated results of the girder displacements as in Fig. A.29.

At the bottom side of the dialog box, there are precision adjusters for x and y axes. Using these controls, desired precisions can be adjusted on both the graph and the table. At the bottom right of the dialog box, *Save* button is used to write out the results to

a text file. If *Show after save* check box is selected, after save operation, the saved file will be opened. This output file can be either in text format or excel file format.

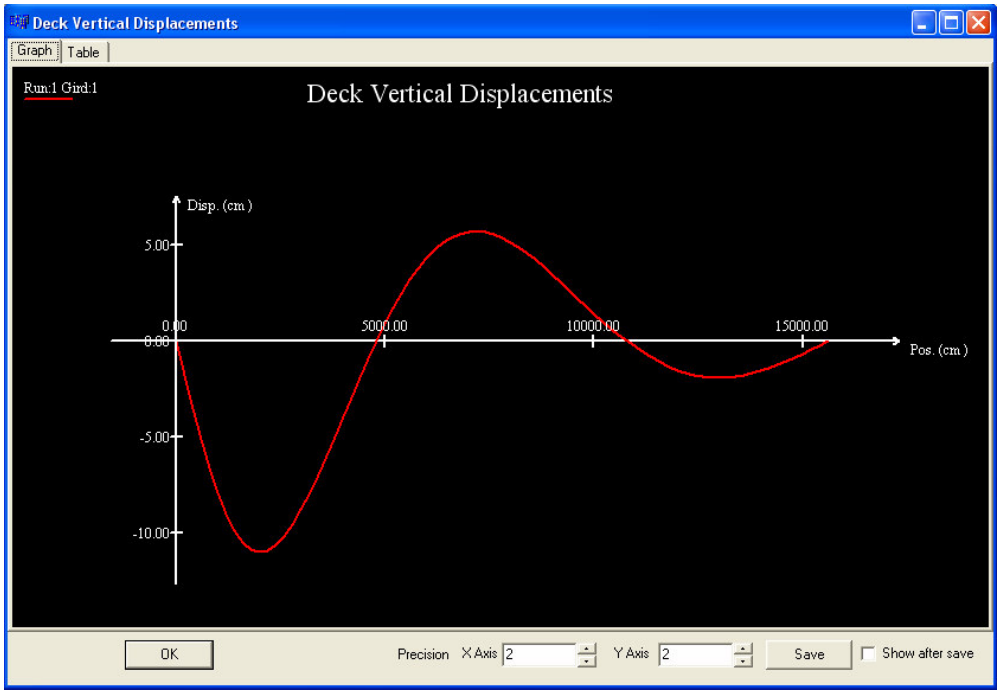


Figure A.28 Graph of Displacement Results

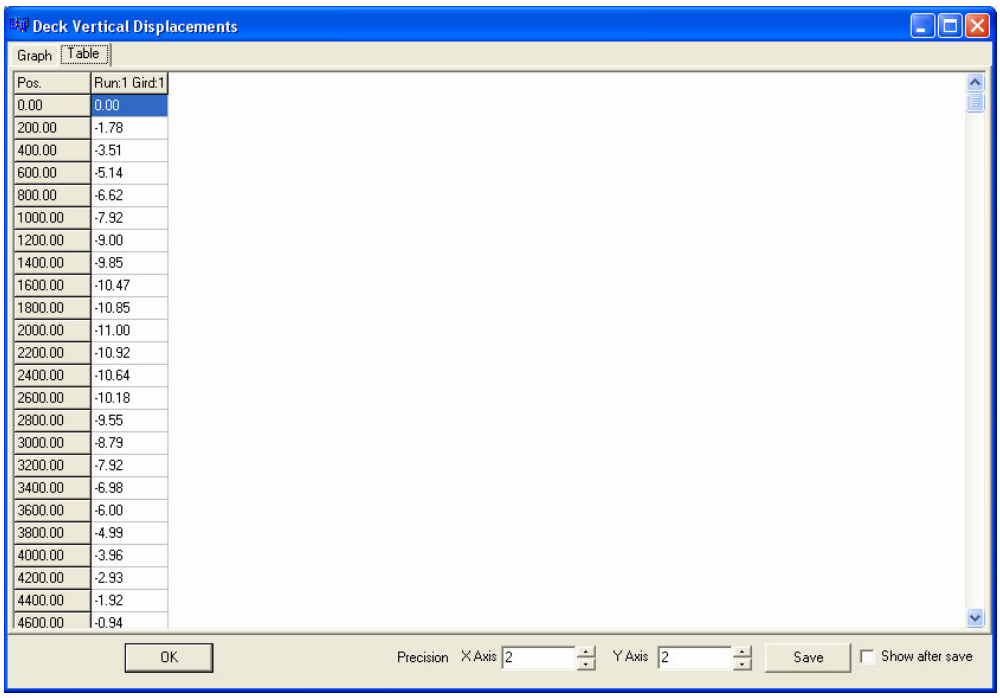


Figure A.29 Table of Displacement Results

These dialog boxes shown in Fig. A.28 and Fig. A.29 are conventionally utilized for all the other analysis results.

Rotations: This menu is used the get the girder rotations. After selecting this menu, like in displacement results, a dialog box will appear for selecting the girder and run number. For rotations, for example, first girder and third run are selected as in Fig. A.30.

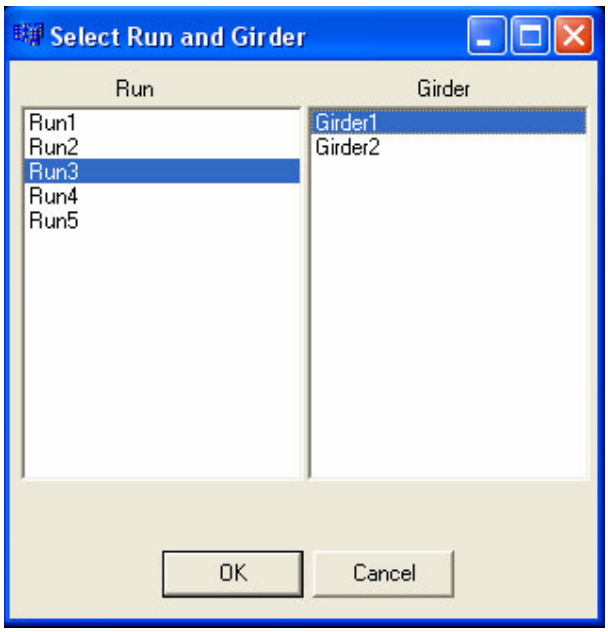


Figure A.30 Run and Girder Selection for Rotations

After pressing ok button, the rotation results will be presented in the results dialog box as in Fig. A.31 and Fig. A.32.

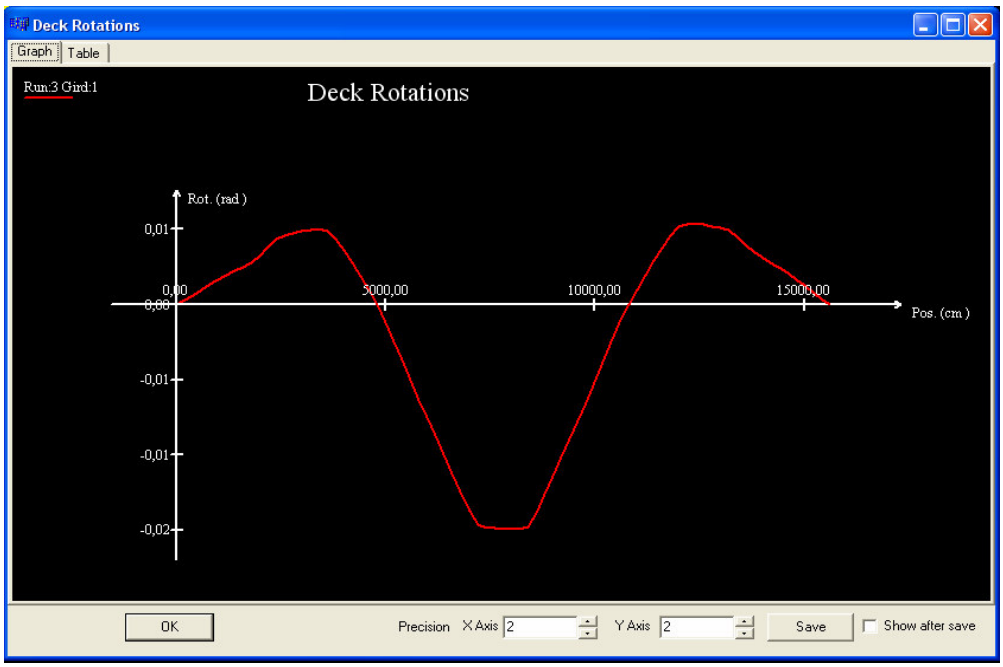


Figure A.31 Graph of Rotation Results

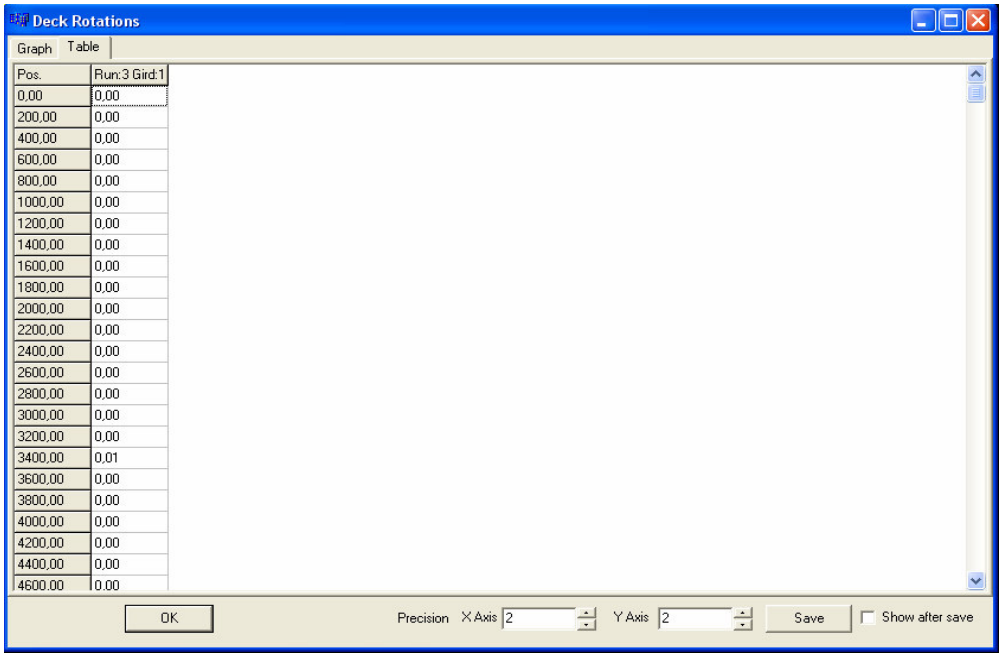


Figure A.32 Table of Rotation Results

Cross-sectional Forces: Under this submenu there are three submenu i.e. V (section shear force), M (Section bending moment), T (Section torsion moment). These force and moments are obtained by integration of end forces of the finite elements at the sections.

In this manual, section bending moments will be presented. To obtain the moment value, first the run number and girder must be selected. Unlike the displacements and rotations, for section force results, sum of the results for all girders can also be selected in Fig. A.33. First run and sum option are selected for this example.

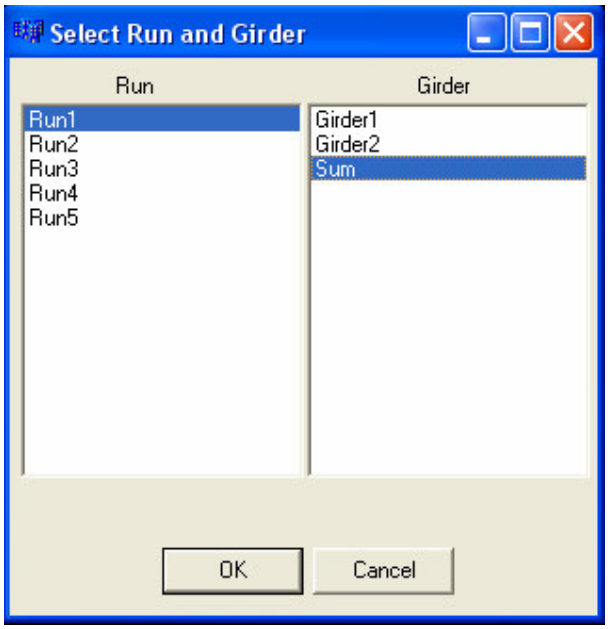


Figure A.33 Run and Girder Selection for Section Moment

As a result, the graph in Fig. A.34 is obtained for the cross-sectional moment for the first run and sum of the all girders.

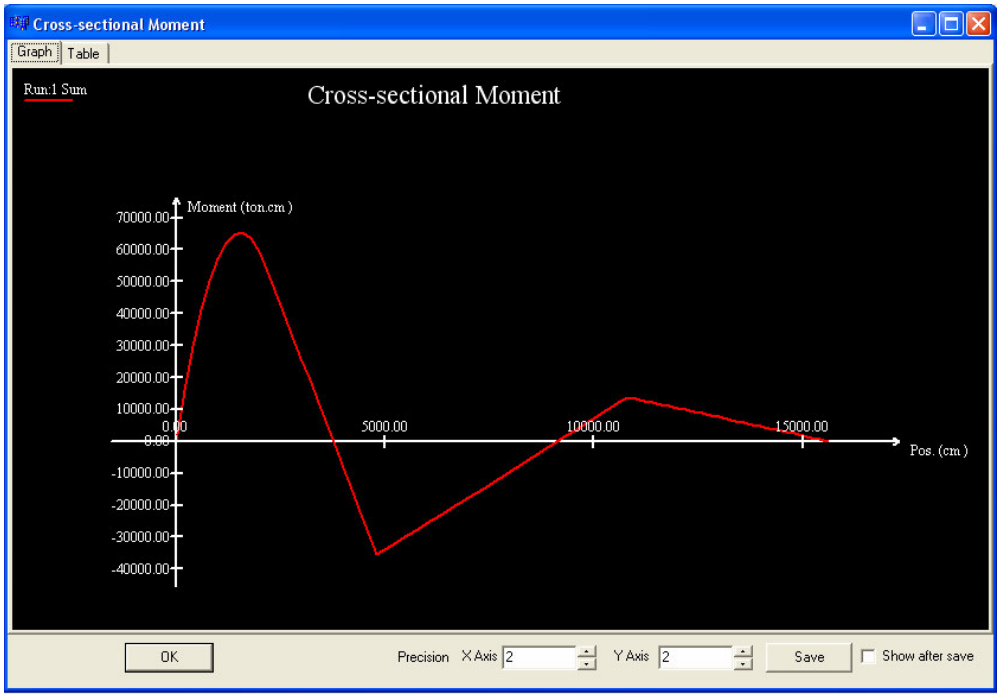


Figure A.34 Graph of Cross-sectional Moment Results

Stresses: This menu is used to obtain the stress values of the selected point on every cross section along the bridge. Under this menu S11, S12 submenus are present. S11 and S12 are the stresses normal and parallel to the sections, respectively. To obtain the stress graph of one of them, first, run and girder selection should be done; afterwards, stress point on a typical cross section should be selected from the stress selection dialog box as in Fig. A.36. Stress point selection dialog box will show the typical cross-section of Box or I Beam girders according to the girder type of the current project.

For this case first run and first girder are selected and for the stress point, the point at the bottom mid face of the girder is selected.

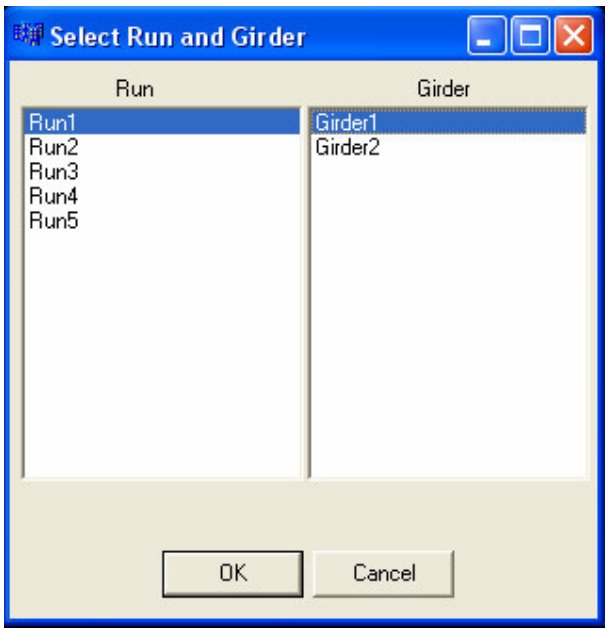


Figure A.35 Run and Girder Selection for Stresses

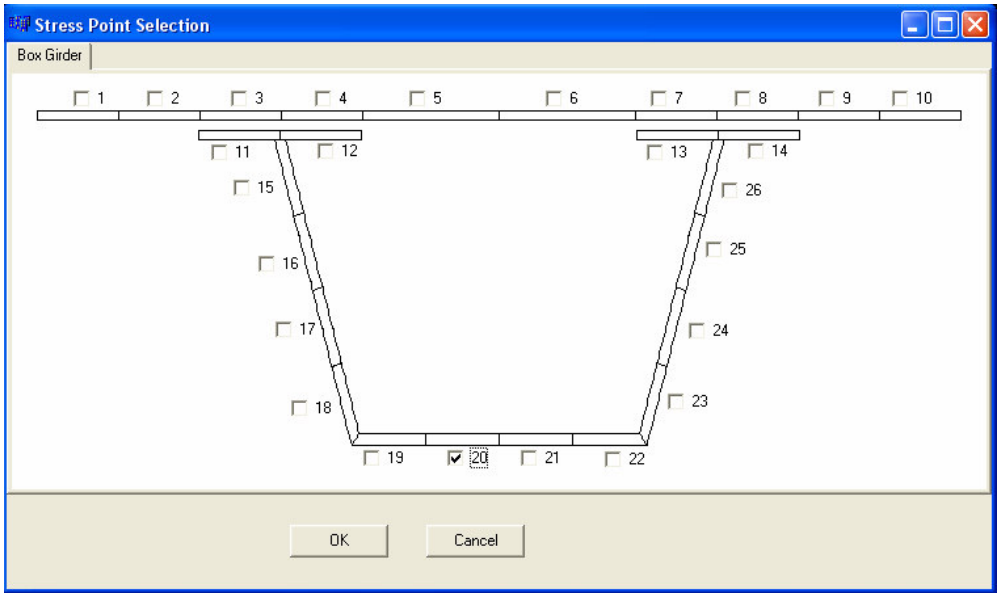


Figure A.36 Stress Point Selection for Box Girder

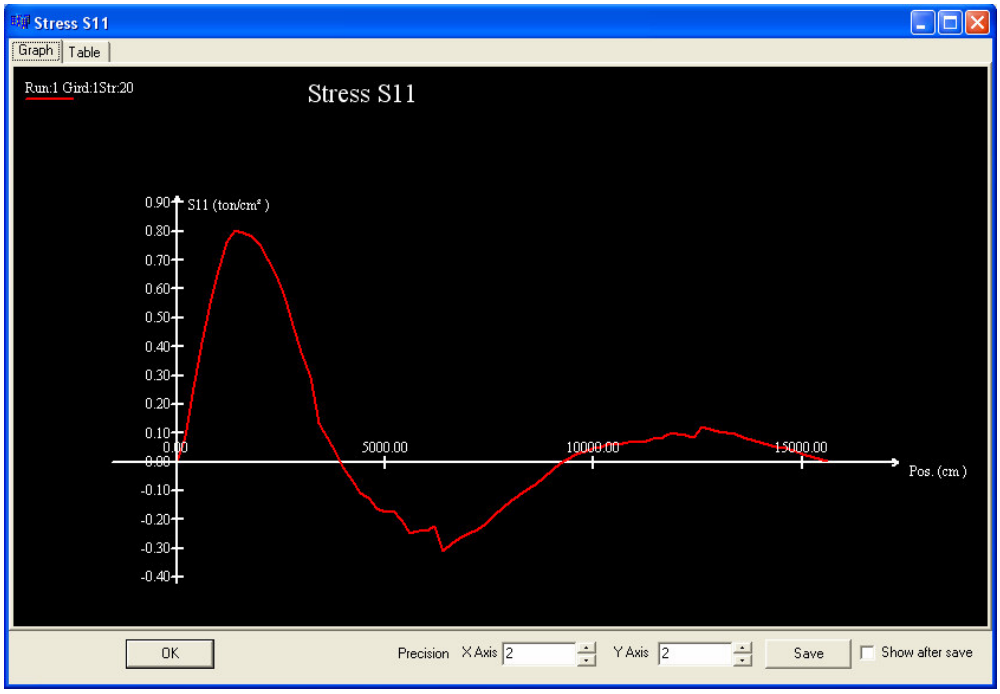


Figure A.37 S11 Stress Results along the Bridge

Stresses on Section: This menu is used to obtain the stress values on bridge as the *Stresses* menu above does but this button shows all the stress values on a typical cross section rather than reporting stress at points along the bridge length.

On the R*un and Girder Selection* dialog box, in addition to the run number and girder number, user must select the position of the section on which the stress values taken. Run1, Girder1 and section at 6000 cm is selected for S11 result as in Fig. A.38.

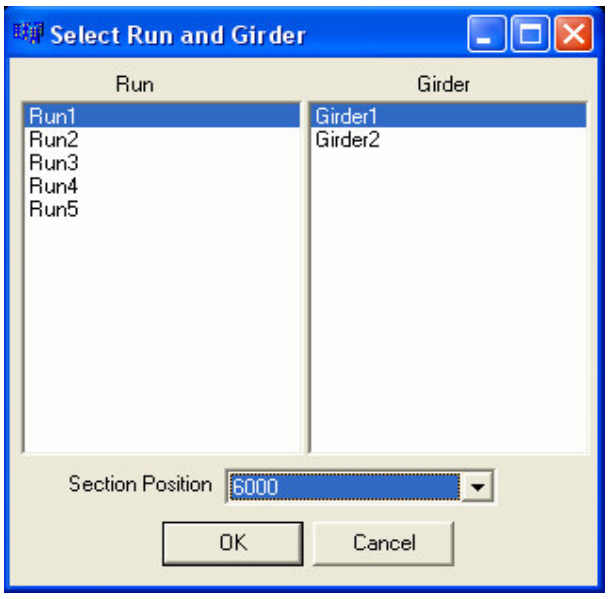


Figure A.38 Run, Girder and Section Position Selection Dialog Box

After selecting the run, girder and section position, the results for the stress values will be shown on a typical cross section of the bridge as in Fig. A.39.

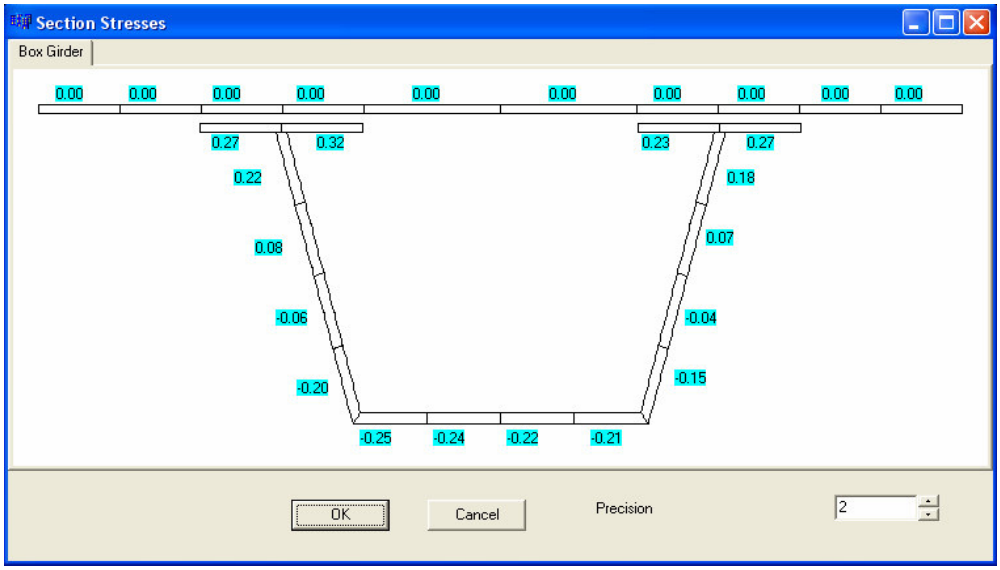


Figure A.39 Stress Results on the Section Selected

Brace Forces: Brace forces menu is used to obtain the forces on the braces under given loading. There are three submenus under this menu, internal braces, external braces, lateral braces.

To obtain the brace forces result, run number should be selected. Run1 is selected for this example as in the Fig. A.40.

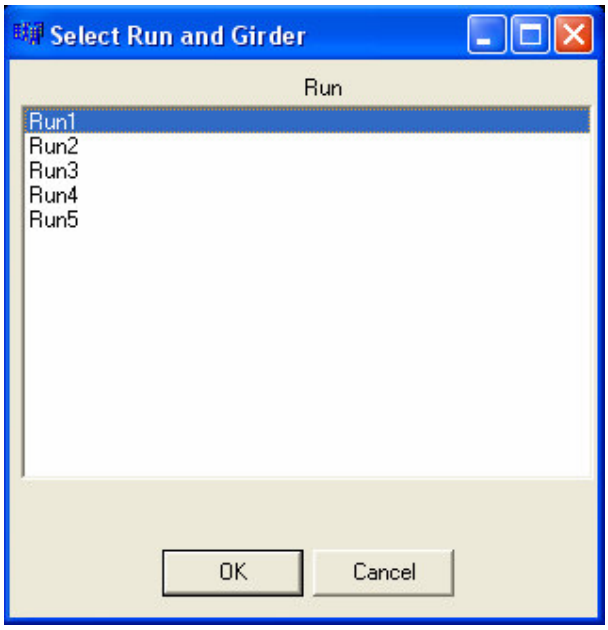


Figure A.40 Run Number Selection for Braces

After the selection of the run number, if external or internal brace forces are required, a dialog box will appear to select the brace items as in Fig. A.41. The check box count in the dialog box will be 5 and 4 for external and internal braces, respectively. If lateral brace forces are desired, this brace item selection dialog box will not be shown. In this example, internal brace is selected as brace type and first item is selected as brace sub item.

Figure A.41 Brace Item Selection Dialog Box

Following the brace item selection dialog, the brace forces result dialog box will appear as in the Fig. A.42.

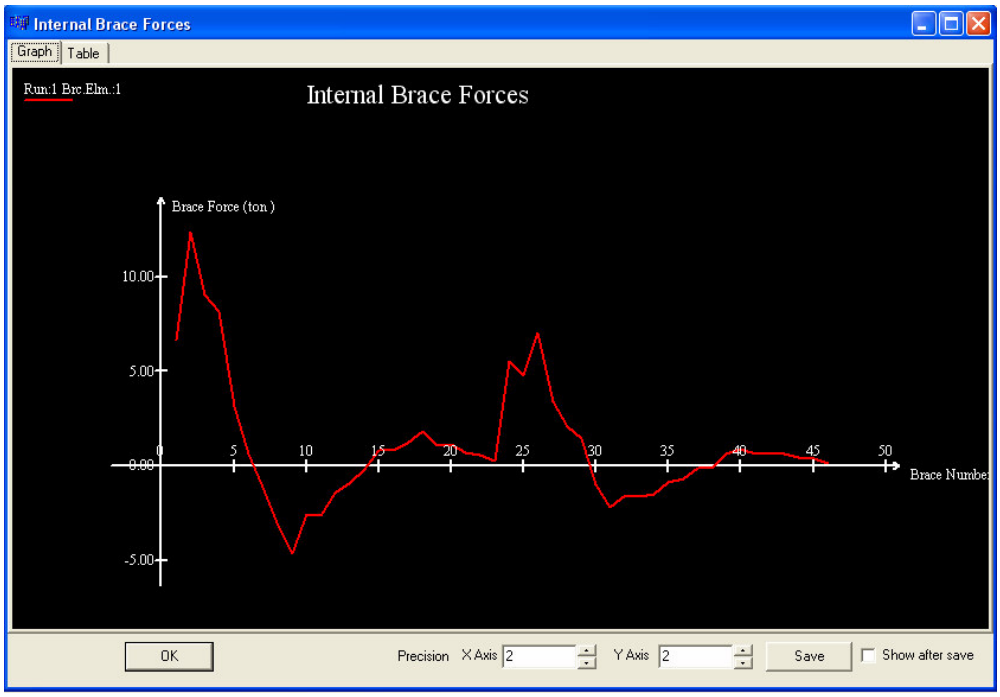


Figure A.43 Brace Forces Result Dialog Box

Support Reactions: Support reactions menu will give the resultant reactions under the supports of the bridge. To obtain the reactions, run number and girder numbers or the sum of all girders should be selected as shown in Fig. A.43. For this case, Run1 and Sum choices are selected. After pressing the *OK* button, the result dialog box will immediately appear. The result dialog box for reactions has only tabulated tab as in Fig. A.44.

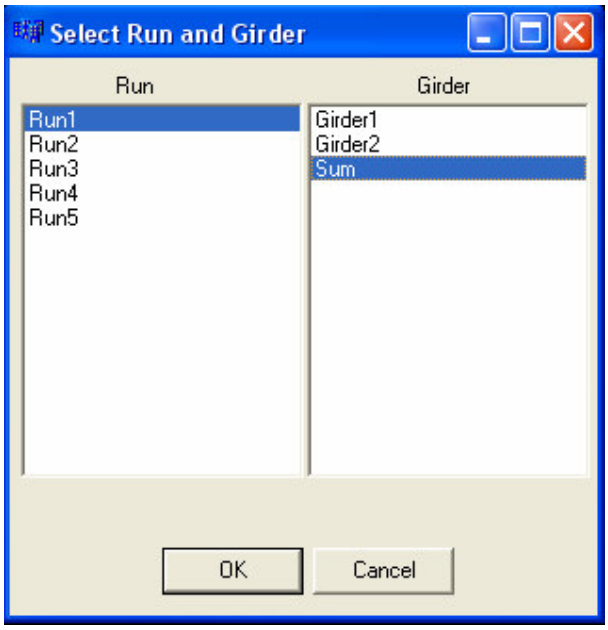


Figure A.43 Run and Girder Selection for Support Reactions

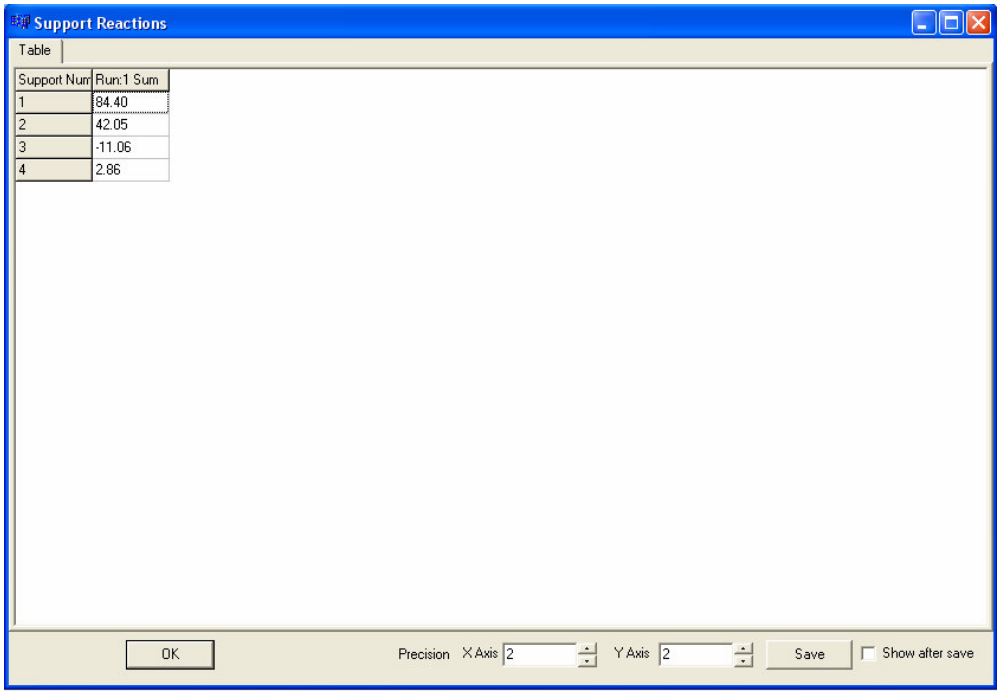


Figure A.44 Result Dialog Box for Support Reactions

Shear Flow: Using this menu, one can get the shear flow at the interface between concrete deck and girders along the bridge. To obtain the shear flow result, run number and girders should be selected as in Fig. A.45. In the girder list box, sum option is present to get the total shear flow of all girders. For this example, Run5 and Sum are selected.

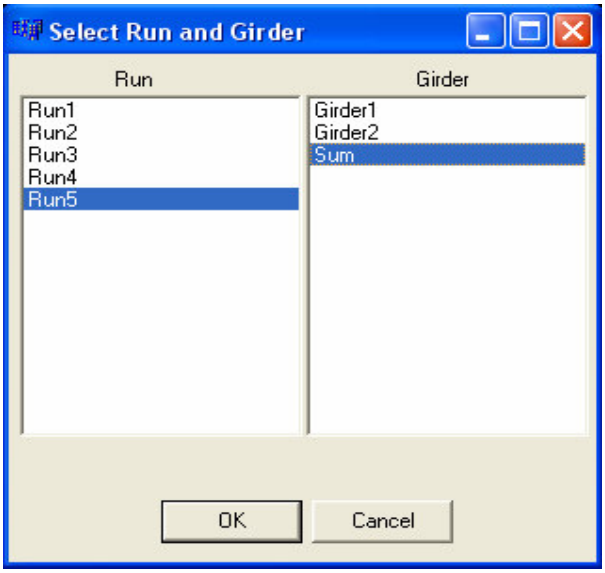


Figure A.45 Run and Girder Selection Dialog Box for Shear Flow

Following the run and girder selection, the shear flow result dialog box will be shown. In the dialog box, shear flow along the bridge is shown as graph in Fig. A.46.

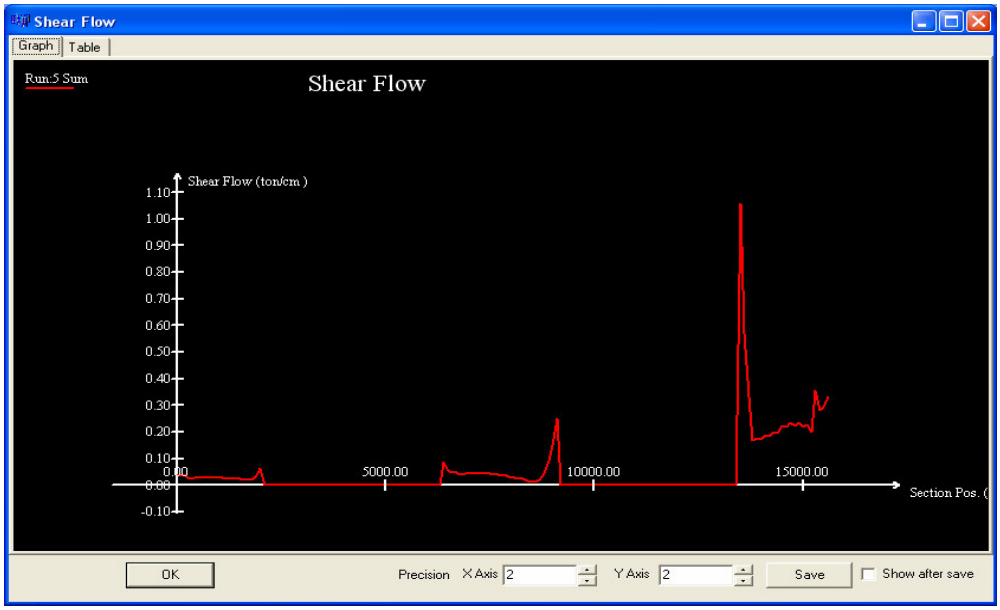


Figure A.46 Shear Flow Result

There is also a toolbar for direct access to results on the UTrAp main frame. After pressing these buttons, a result type selection dialog box will appear as in Fig. A.47. On this dialog box, there are three options present; *Placement Sequence, Live Load, Influence Line*. After selecting and pressing *OK* button, the required dialog boxes for the type of result will be shown.

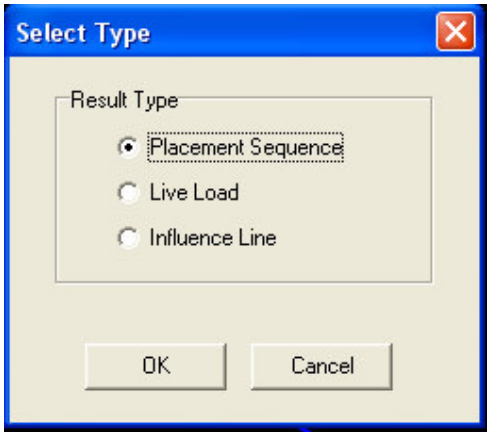


Figure A.47 Result Type Selection Dialog Box

There are also other buttons on this toolbar other than result buttons. These buttons are mainly for file operations.

All the buttons on this toolbar are tabulated in Table A.7.

Table A.7. Tool Buttons of Main Frame Toolbar and Functions

Tool Button Icon	Function of the Button
	Button to obtain the displacement results.
	Button to obtain the rotation results.
	Button to obtain the stress results of type S11.
	Button to obtain the stress results of type S12.
	Button to obtain the sectional shear force.
	Button to obtain the sectional moment.
	Button to obtain the sectional torsion force.
	Button to obtain the internal braces forces.
	Button to obtain the external braces forces.
	Button to obtain the lateral braces forces.
	Button to obtain the support reactions.
	Button to obtain the shear flow.
	Button to start a new project
	Button to open a presaved project from a file
	Button to save the active project to a file
	Button to save the active project with a different name
	Button to run the analysis. (Disabled in analyzed state)
	This button shows the lock state of the active project. First and second labels show that project is in unlocked and locked state respectively. In the unlocked state, pressing this button will lock the project and label of the button will change to second type. Similarly, in locked state, pressing this button will unlock the project and label of the button will change back to first type

In the program, user can set the unit system by changing the selected item in the unit system combo box at the lower right corner of the UTrAp main frame. This unit system is only for the fact that user enters the inputs in a consistent unit system. No user inputs are changed according to the adopted unit system. In the program, only predefined standard truck loads are converted to adopted unit system before used in the calculations.

Final Comments

The program works under Windows XP operating systems. A physical memory of 1 GB is recommended for problems involving twin girders. In cases where the physical memory is not enough, the program uses the virtual memory to solve the problem. However, using virtual memory significantly increases the time for solution.

APPENDIX B

USEFUL ALGORITHMS USED IN THE PROGRAM

Quick Sort Algorithm:

This algrotihm has three input parameters, these are a vector to be sorted and two integer values for sorting bounds. To make a total sort of given array, *start=0* and *end=vector.size()-1*. This function sorts the *array* from smallest to largest number. It returns true, unless it encounter same values in the array.

```
bool Sort(vector<int>& array, int start, int end){
    if(start < end) {
        //Take the pivot as the first value.
        int pivot = start, pivotValue = array[start];
        int temp;
        //Move the values less than pivot to below pivot.
        for(int i=start+1; i<=end; i++){
            if (array[i] < pivotValue){
                temp = array[i];
                array[i] = array[pivot+1];
                array[pivot+1] = temp;
                pivot++;
            }
            else if(array[i] == pivotValue){
               //There are two entries having same Id
                return false;
            }
        }
        //Put the pivot into its proper place.
        temp = array[start];
        array[start] = array[pivot];
        array[pivot] = temp;
        //Do the next quicksort.
        Sort(array, start, pivot-1);
        Sort(array, pivot+1, end);
    }
    return true;
}
```

Quick Search Algorithm:

This function has two input and one output parameters, these are a vector in which searching performed, a searched value of integer, and the index of the searched value in the array. It returns true, unless the given array has no entries.

```
bool Search(vector<int>& array, int id,unsigned& index)const
{
    if(array.size()==0)
        return false;
    int tempIndex;
    int temp,l,r;
    l=0;
    r=array.size()-1;
    do
    {
        temp=(l+r)/2;
        if(id<array[temp])
               r=temp-1;
        else
               l=temp+1;
    }
    while(id!=array[temp] && l<=r);

    if(id==array[temp])
       tempIndex=temp;
    else
        return false;
    index=(unsigned)tempIndex;
    return true;
}
```

APPENDIX C

VARIABLES OF INPUT DATA STRUCTURE

Table A.8. Variables of Input Data Structure

Variable Type	Variable Name	Variable Definition
char*	project	Name of current project
int	ngird	Number of the girder
double	elemsize	Width of an element along the bridge
double	steelmodulus	Elastic Modulus of Steel
int	isec_type	Section type (I or Box)
int	ioutelem	Placement of unit load for every n element
double	webdC	Web depth of section
double	botflC	Width of bottom flange
double	toplC	Top width of section
double	tfwC	Width of top flange
double	deckwC	Deck width of bridge
double	decktC	Deck thickness of bridge
double	girdspacingC	Girder spacing
int	nsegmC	Segment count in the bridge
int	nwebt	Number of web plate thicknesses
int	nbotft	Number of bottom flange thicknesses
int	ntft	Number of top flange thicknesses
int	n_int_brc	Number of internal braces
int	n_ext_brc	Number of external braces
int	n_top_ltr	Number of lateral braces
int	n_support	Number support in the bridge
int	ielemtype	Shell element type used in the FE model
int	n_studsp	Number of stud properties
int	ianalysistype	Type of the analysis performed
int	isolvertype	Solver type used in the analysis
int	n_runs	Number of different runs for placement sequence
int	n_deck	Number of deck properties for placement sequence
int	nlanes	Number of lanes defined in the project
int	n_deck_ll	Number of deck prop. for live loading analysis
int	iremextbr	Flag for removal of external braces in live loading analysis
double*	al_segm	Array of size *nsegmC* for length of segments
double*	al_rcurv_segm	Array of size *nsegmC* for radius of segments

continuation of the table is on the next page

double*	alwebt	Array of size nwebt for range of web thicknesses along the bridge
double*	webt	Array of size nwebt for web thicknesses
double*	albotft	Array of size nbotft for range of bottom flange thicknesses along the bridge
double*	botft	Array of size nbotft for bottom flange thicknesses
double*	altft	Array of size ntft for range of top flange thicknesses along the bridge
double*	tft	Array of size ntft for top flange thicknesses
double*	aloc_int_brc	Array of size n_int_brc for location of internal braces
double*	area_int_brc	Array of size n_int_brc for section area of internal braces
int*	ktype_int_brc	Array of size n_int_brc for type of internal braces
int*	kgird_int_brc	Array of size n_int_brc for index of the girder where internal braces are placed.
double*	aloc_ext_brc	Array of size n_int_brc for location of external braces
double*	area_ext_brc	Array of size n_ext_brc for sectional area of external braces
int*	ktype_ext_brc	Array of size n_ext_brc for type of external braces
int*	kspa_ext_brc	Array of size n_ext_brc for index of the spacing where external braces are placed.
double*	aloc1_top_ltr	Array of size n_top_ltr for start location of lateral braces
double*	area_top_ltr	Array of size n_top_ltr for sectional area of lateral braces
double*	aloc2_top_ltr	Array of size n_top_ltr for end location of lateral braces
int*	kspa_top_ltr	Array of size n_top_ltr for index of the spacing where lateral braces are placed.
int*	ktype_top_ltr	Array of size n_top_ltr for type of lateral braces
double*	aloc_support	Array of size n_support for support locations
double*	al_studsp	Array of size n_studsp for range of studs along the bridge
double*	stud_sp	Array of size n_studsp for stud spacing
int*	n_s_flange	Array of size n_studsp for number of stud per flange
double*	al_pour	Array of size n_deck for length of deck segments of placement sequence analysis
double*	conc_mod	Array of size n_runs*n_deck for elastic modulus of concerete for placement sequence analysis
double*	stud_stf	Array of size n_runs*n_deck for stud stiffnesses for placement sequence analysis
double*	dist_load	Array of size n_runs*n_deck for loading of placement sequence analysis
double*	st_lane	Array of size nlanes for start position of lanes
double*	ed_lane	Array of size nlanes for end position of lanes
double*	al_live	Array of size n_deck_ll for length of deck segments for live loading analysis
double*	conc_mod_ll	Array of size n_deck_ll for elastic modulus of concrete for live loading analysis
double*	stud_stf_ll	Array of size n_deck_ll for stud stiffnesses for live loading analysis