

Object-Oriented LRFD Design of Steel Frames

by

Feng Huang

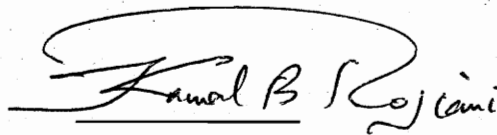
Project Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

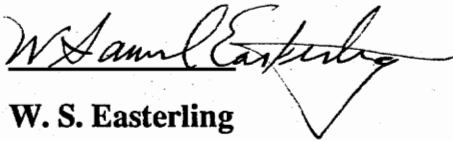
in

Civil Engineering

Approved:



K. B. Rojiani



W. S. Easterling



D. A. Garst

June, 1996

Blacksburg, Virginia

C.2

LD
5655
V851
1996
H836
c.2

Object-Oriented LRFD Design of Steel Frames

by

Feng Huang

Committee Chairman: Dr. Kamal B. Rojiani

Charles E. Via, Jr. Department of Civil Engineering

Virginia Polytechnic Institute and State University

Abstract

A program for the design of steel frames was developed. The program can be used to design beams and columns in a frame in accordance with the Load and Resistance Factor Design method. The program was coded in C++ using the object oriented programming method. The program runs under the windows graphical operating environment. The Microsoft Visual C++ compiler was used for developing the program. The major classes in the program are joints, members, beams, columns, and loads, and correspond to objects in a real structure. The architecture of the program is based on the relationship between these objects. The accuracy of the program was verified by comparing design results obtained from the program with results from a commercial structural analysis and design program (DAST version 11.0), as well as with results obtained from hand calculations. It was concluded that the object oriented approach has many advantages that makes it a desirable approach for developing computer applications for structure engineering for today's graphical windows operating environments.

Acknowledgments

I would like to thank Dr. K. B. Rojiani for his continuous support and guidance throughout this research. I would like to thank Professors W. S. Easterling and D. A. Garst for serving as committee members.

I would also like to express my deep gratitude to my husband for his love and encouragement throughout all these years. Without his moral and financial help, I could not have completed this project.

Table of Contents

Acknowledgments	iii
List of Figures	viii
List of Tables	x
1.0 Introduction	1
1.1 introduction	1
1.2 Purpose	2
1.3 Organization	2
2.0 Overview of Object Oriented Programming	4
2.1 Introduction	4
2.2 Object Oriented Programming	4
2.3 Classes and Objects	6
2.4 Characteristics of Objects	6
2.5 Object Oriented languages	8
2.6 Visual C++ Integrated Development Environment	8
2.7 Development of a Windows Application using MFC	9
2.8 Program Design Strategy	12
3.0 Program Structure	14
3.1 Introduction	14
3.2 The Program Development Process	14
3.3 Overview of the Classes	15

3.4	Structural Classes	16
3.4.1	<u>Structure</u> Class	17
3.4.2	<u>Joint</u> Class	17
3.4.3	<u>LoadComb</u> Class	17
3.4.4	<u>Member</u> Class	18
3.4.5	<u>JointLoad</u> Class	23
3.4.6	<u>MembLoad</u> Class	23
3.5	Interface Classes	24
3.5.1	Document Class	24
3.5.2	View Class	26
3.5.3	Dialog Box Classes	26
4.0	Design Procedure and Program Description	29
4.1	Introduction	29
4.2	Overview of the Structural Design Procedure	29
4.2.1	Structure Features	30
4.2.2	Beam Design Parameters	30
4.2.3	Column Design Parameters	31
4.2.4	Mark Groups	31
4.2.5	Load Combinations	32
4.2.6	Beam Design	32
4.2.7	Column Design	33
4.2.8	Design Results	35
4.3	Program Description	36
4.3.1	Main Window Menus	36

4.3.2 Main Application Window Menu	36
5.0 Design Results and Discussions	48
5.1 Introduction	48
5.2 Continuous Beam	48
5.2.1 Geometry and Loading	48
5.2.2 Input and Output Files	49
5.2.3 Discussion of Design Results	49
5.3 Two Story Plane Frame	51
5.3.1 Geometry and Loading	51
5.3.2 Input and Output Files	53
5.3.3 Discussion of Design Results	54
5.4 Two Story Space Frame	55
5.4.1 Geometry and Loading	55
5.4.2 Input and Output Files	59
5.4.3 Discussion of Design Results	59
6.0 Summary and Conclusions	64
References	66
Appendix A. Class Definitions	70
Appendix B. Input and Output Files for Test Cases	100
Appendix C. Hand Verification	146

List of Figures

Figure 2.1 Class Inheritances.....	11
Figure 3.1 Hierarchy of Member Classes.....	18
Figure 3.2 Hierarchy of Load Classes.....	23
Figure 3.3 Classes for Dialog Boxes.....	28
Figure 4.1 Main Application Window.....	37
Figure 4.2 Structure Features Dialog Box for Plane Frame	39
Figure 4.3 Structure Features Dialog Box for Space Frame	39
Figure 4.4 Load Combination Dialog Box	40
Figure 4.5 Beam Design Parameters Dialog Box	41
Figure 4.6 Position of the Internal Brace Points Dialog Box	41
Figure 4.7 Column Design Parameters Dialog Box for Plane Frame	42
Figure 4.8 Column Design Parameters Dialog Box for Space Frame	42
Figure 4.9 Beam Mark Group Dialog Box.....	43
Figure 4.10 Column Mark Group Dialog Box.....	43
Figure 4.11 A Message Box for Checking Design Parameters.....	44
Figure 4.12 Beam Number Dialog Box.....	45
Figure 4.13 Beam Design Results Window.....	46
Figure 4.14 Column Design Results Window.....	47
Figure 5.1 Continuous Beam.....	50
Figure 5.2 Two Story Plane Frame.....	52
Figure 5.3 Dead Loads and Live Loads on the Frame.....	52
Figure 5.4 Wind Loads on the Frame.....	53

Figure 5.5 Two Story Space Frame.....	56
Figure 5.6 Dead loads on the Frame.....	57
Figure 5.7 Live Loads on the Frame.....	57
Figure 5.8 Wind Loads (front) on the Frame.....	58
Figure 5.9 Wind Loads (side) on the Frame.....	58
C. 1 Loads On Continuous Beam	146
C. 2 Moment Diagram of the Continuous beam for $1.2D+1.6L$	147
C. 3 Two Story Plane Frame with Applied Loads	149
C. 4 Moment Diagrams for Member 7 and 8 for $1.2D+1.6L$	156

List of Tables

Table 3.1 Interface Classes.....	27
Table 5.1 Design Results for Continuous Beam.....	51
Table 5.2 Beam Design Results for Plane Frame	54
Table 5.3 Column Design Results for the Plane Frame.....	55
Table 5.4 Comparison of Beam Design Results for Space Frame.....	61
Table 5.5 Column Design Results for the Space Frame.....	62
Table 5.6 Code Check by DAST for Column Design Results	63
Table C.1 Design Results from the Program	149
Table C.2 G Values for Calculating K_x	150
Table C.3 Effective Length Factor K_x	151
Table C.4 Member End Forces for Dead Load	151
Table C.5 Member End Forces for Live Load	152
Table C.6 Member End Forces for Wind Load	152
Table C.7 Required Flexural Moment M_{ux} and P_u for 1.2D+1.6L	153
Table C.8 Required Flexural Moment M_{ux} and P_u for 1.2D+1.3W+0.5L	154
Table C.9 Unity Checks for LRFD Beam-Column Interatin Formulas for 1.2D+1.6L.	155
Table C.10 Unity Checks for LRFD Beam-Column Interatin Formulas for 1.2D+1.3W+0.5L	155

Chapter I

Introduction

1.1 Introduction

The concept of object oriented programming was first introduced in the mid 60's [Yu & Adeli 1991]. However, it did not draw much attention until the late 80's. In recent years, object oriented programming has become one of the most popular programming methodologies in software engineering. One reason for this surge of interest is that the characteristics of the object oriented programming paradigm, such as data encapsulation, abstraction, inheritance and reusability, make it easier to develop programs that can model intricate real world objects. Another reason is that most of the popular object oriented languages provide an extensive collection of class libraries that greatly reduce the effort of developing applications for graphical operating environments such as Microsoft windows.

C++ is the most popular object oriented programming language. As a superset of the popular ANSI C language, it supports both procedural and object oriented programming. To many C programmers, it is easier to learn a superset of C than a "pure" object oriented language [Petel,1994]. This is one reason for the growing popularity of C++. Many C++ compilers provide large class libraries for performing routine programming tasks and for developing applications for graphical operating environments. These classes serve as the basic building blocks of an application.

Structural engineering software development has made great progress in recent years. Many structural engineering programs have recently been updated to take advantage of the graphical windows environment. In the light of the many potential

benefits of object oriented programming, structural engineers and software developers are exploring the applications of the object oriented programming methodology to various areas of structural engineering [Forde, 1990], [Biedermann, 1993], [Aouad, 1994]. The object oriented programming approach has the potential to be a powerful tool for developing integrated structural engineering applications.

1.2 Purpose

The objective of this project is to develop a program for the design of continuous steel beams, plane frames and space frames. In order to concentrate on the design process, analysis results are read from the output file of the CADKEY frame analysis program. Beams and columns in the plane frames and space frames are designed in accordance with the AISC Load and Resistant Factor Design Specification(LRFD)[AISC, 1986]. The design sections used in this program are standard W-sections. The program was written in C++, and was developed using the Microsoft Visual C++ compiler. The program runs under the Microsoft Windows graphical operating environment.

1.3 Organization

Chapter 2 of this project report presents an overview of the object oriented programming paradigm, and the Microsoft Visual C++ compiler. In Chapter 3, the program development process and the structure of the program is discussed. The organization of the structural classes and the interface classes is also described. Chapter 4 provides the major features of the program and also serves as an user's manual for the program. Design results are verified and discussed in Chapter 5. Chapter 6 gives a

summary and major conclusions. A complete definition of the classes used in the program is given in Appendix A. Appendix B provides input and output files for three test problems. A hand verification of the first two test problems is presented in Appendix C.

Chapter 2

Overview of Object Oriented Programming and Visual C++

2.1 Introduction

In this chapter, a brief description of the object oriented programming technique is presented. The essential concepts of object oriented programming, such as data abstraction, inheritance, polymorphism and encapsulation are discussed. A brief description of the Microsoft Visual C++ compiler and the basic concepts regarding the development of a windows application are given. Finally, the object-oriented program design strategy is presented.

2.2 Object Oriented Programming

In the history of the development of computer programming techniques, three techniques, procedural programming, knowledge based programming and object oriented programming, are among the most popular. The procedural programming technique is well suited for simple computational type applications. The procedural programming technique gradually evolved into the so called structured programming technique and could handle more complicated numerical oriented problems. One well known procedural programming language is FORTRAN. In structured programming, a complex problem is broken down into smaller units called functions or subroutines. Several functions are grouped together to form a module. Structured programming lacks the flexibility required

for handling large complex problems [Lee and Arora, 1991], [Biedermann and Grierson, 1993].

In the mid 80's, another popular programming technique called knowledge-based programming emerged. In knowledge-based programming, a series of rules are used to control processing. The rules are usually organized in the form of conditional "IF-THEN" logical decisions. Although this event based programming technique can efficiently deal with heuristics intensive problems where judgment is important, it is not well suited for solving complex problems involving both numerical processing and dynamic events based representation.

Object oriented programming is a relatively new programming technique. An object is an independent unit that contains two elements: data, and functions that operate on data [Fordeand Foschi, 1990]. Objects communicate with each other by sending messages. In the object oriented programming method, the problem is usually divided into objects that correspond to real world objects [Aouad and Kirkham, 1994]. A good description of object oriented programming is the one given by Miller, "A program thus builds an environment containing objects (such as beams, floors and columns) with attributes analogous to the real world counterparts (such as weight, length and material properties) and analogous behaviors (deformation under loads). Furthermore, objects can communicate with each other and with the user, and can be given necessary skills and intelligence for carrying out requested tasks." [Miller, 1991]. Compared with the procedural programming technique, object oriented programming has brought about a revolutionary change in the way in which real world objects can be modeled on a computer. It has many attractive features that make it a useful programming technique [An-Nashif and Powell, 1991].

2.3 Classes and Objects

As the name indicates, the key concept behind object oriented programming is the object [Yu and Adeli, 1991]. An object is an instance of a class. A class is a data structure that contains both data attributes and functions for manipulating the data. The class acts like a "blueprint" for each object of that particular class. The behavior of an object is based on its class definition. When an object is created, memory is allocated for the data elements of the object [Kulkarni, 1993]. When an object needs to perform a given task, it calls the corresponding function to operate upon its own data. The object's data can be accessed from outside the object through messaging functions. The functions in a class are usually divided into two types: those that operate on the object's data and those that provide the interface between the object and the outside world.

2.4 Characteristics of Objects

The most important characteristics of objects are [Budd, 1991]:

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

An object contains both data and functions. Encapsulation means preventing data and functions from being modified by other parts of the program. Data members can be defined as private, which means only functions inside the object can access the data. Data members can also be declared as public, which means these member can be accessed by other objects. The object oriented paradigm provides total control on the access of data

member in an object. Encapsulation makes object oriented programming significantly different from conventional programming where data are often stored on a global basis.

Abstraction means that the details of the implementation are hidden from other parts of the program. When a function from outside an object requests an action, it does not need to know how this request is fulfilled. It only needs to know what information has to be sent to the object and what information is returned by the object. For example, consider an object of class Rectangle for storing and manipulating rectangular entities in a program. When the program needs to draw a rectangle, a message is sent to this object to draw itself. The programmer just sends a message to the object to perform the desired task and does not have to care about the details of how the task is implemented inside the object.

In the real world, many similar objects can be grouped into a general class, which contains the common properties of its subset objects. In object oriented programming, the general class is called the base class or the parent class. New classes can be derived from this base class. The new classes inherit the characteristics of the base class but can also have their own special properties. This unique feature of object oriented programming is called inheritance. Inheritance significantly reduces the effort required to design a new class. With inheritance, it is possible to visualize the relations between classes. As Miller pointed out, "one of the more powerful features of object oriented programming is the capability of sharing attributes and skills by means of inheritance" [Miller,1991].

Polymorphism, a definition from biology, means the expression of different shapes of the same organism. In object oriented programming, this feature is usually implemented by using virtual functions. Because of the encapsulation of data and functions, functions in different classes do not interfere with each other. Thus, the programmer can use the same function name which is declared in base class as virtual

function to call multiple functions in the subclasses. For example, consider a function called Draw which is defined as virtual in a base class. When the Draw function of its subclass Rectangle or Circle is called, the corresponding object of the class will draw a rectangle or a circle.

2.5 Object Oriented Languages

There are many object oriented programming languages in use today. A partial list of the popular object oriented languages includes C++, SmallTalk, Objective-C, and Object Pascal [Budd, 1991].

SmallTalk is the oldest object oriented language and is now the most popular of the pure object oriented languages. In SmallTalk, every element is an object. Like the other object oriented programming languages, SmallTalk has a rich collection of class libraries.

Both C++ and Objective-C are extensions of the C programming language. They are called hybrid programming languages. Object Pascal is also a hybrid programming language derived from Pascal. C++ is a superset of the C programming language. C++ is the most popular of all the object oriented programming languages. The reason for its popularity is due to the fact that many programmers are familiar with the C programming language. The hybrid nature of C++ gives programmers the flexibility to develop both procedural and object-oriented programs. Previously written C code can be embedded in the new object-oriented environment. This can significantly reduce the time required for developing new C++ software or updating existing software. This ability has contributed to the success of C++.

2.6 Visual C++ Integrated Development Environment

Visual C++, from Microsoft Corporation, is a comprehensive integrated program development environment that consists of an editor, compiler, linker, debugger, object brewer and resource. [Microsoft ,1993][Young, 1993].

The integrated development environment is called the Visual Workbench. It has an editor for entering program source code and wizards (App Wizard and Class Wizard) for automatically generating code for Windows programs, as well as a project manager for managing program files. It also contains an integrated debugger which makes it possible to quickly and easily debug the program. The C/C++ compiler, linker, and other tools are integrated with Visual Workbench and are called directly from the Workbench.

Microsoft Foundation Classes (usually referred to as the MFC) is an extensive C++ class library designed for creating Windows applications. MFC has many high level features that can save considerable coding effort. The Visual C++ has several other tools which make it easier to develop applications using MFC. The App Wizard generates a starter application using MFC classes. The Microsoft App Studio provides a place for creating and editing resources for user interface elements such as menus, dialog boxes, toolbars and icons. The Class Wizard connects interface objects to the application code and provides a mechanism for writing message handling functions.

2.7 Development of a Windows Application using MFC

The Microsoft Foundation Class (MFC) Library is an object oriented class library which significantly reduces the effort required for designing and implementing applications for Windows. The MFC Library contains a large number of C++ classes and provides an

application development framework which contains the fundamental components of a Windows application.

The classes for the new application are derived from classes contained in the MFC library. The derived classes inherit all the behavior and functionality of their base classes. The functionality of the classes is extended by adding new member variables and functions and modify their existing behavior by overriding inherited member functions.

The key concepts in the development of a program using the MFC Application Framework are the following [Microsoft, 1993]: (1) The object derived from the **application** is the heart of the Windows application. The **application** object manages a list of documents and forwards commands to other objects in the program. There is only one **application** object in each application. (2) The **document** class is responsible for manipulating data. It maintains, loads and stores the application's data. It is possible to create an application that can work with either a single document or multiple documents. Multi-document applications can handle many documents in the same main frame window. (3) A view window provides a window for displaying the document and for interface between the document and the user. The **view** class displays the document's data or responds to the events such as mouse and keyboard input. (4) User interface objects, such as menus, buttons, toolbars, send commands to the documents, views and other objects in the application. Various types of dialog boxes provide places for data input.

The main tasks for developing an MFC application are the following:

(1) Defining the application's data in its document class. No matter how complicated the program's application classes, all data manipulation of these classes needs to be done through the application's document class.

(2) Defining how the user views and interacts with the data. The View classes show different parts of the document, for example, the graphical representation of the structure or the output of the program.

(3) Connecting menus, buttons and other user interface elements to commands, and defining message handling functions to execute these commands.

Figure 2.1 from the MFC User's Guide [Microsoft, 1993] shows how the new application code is derived from the MFC framework class. In Figure 2.1, the CYourApp class is an manager class which takes care of initializing the frame windows and opening or closing document files. The CYourFrame class builds the empty window frames for the program. The CYourView classes provide view windows for displaying certain parts of the document. The CYourDlg classes are created as user interface classes for data input or interaction with the user. The CYourDoc class provides a place for loading, storing and managing application data.

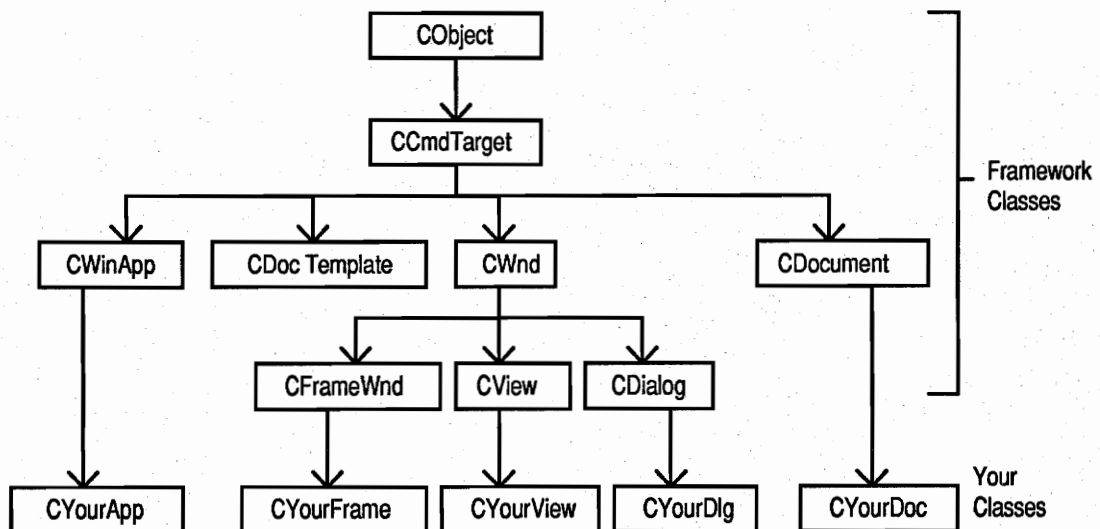


Figure 2.1 Class Inheritances

2.8 Program Design Strategy

To take full advantage of object oriented programming, a special effort has to be made when developing the applications [Powell and Abdalla, 1989].

In the conventional top-down design, the programmer first outlines the major steps of the program using pseudocode. The procedural decomposition is processed by breaking the major steps down into smaller steps. Object oriented design technique is significantly different from the procedural program design technique. It is neither a top-down technique nor a bottom-up process. Actually, object oriented design is a process for object interactions at both high and low levels of abstraction throughout the project development process.

Microsoft Tutorial [Microsoft, 1993] gives the following general steps for object-oriented design:

- (1) Identify the classes.
- (2) Define attributes and behavior for the class.
- (3) Find relationships among the classes.
- (4) Arrange the classes into hierarchies.

The first step is to define classes for the program. To identify classes for modeling physical objects is sometimes much easier than to identify the conceptual objects on which the program manipulates. It is more difficult to define the events and the interactions between events in the program and arrange them into classes.

Attributes represent the features of the class. The information that an object of a class must hold are attributes. Behavior is about how an object interacts with itself and how it interacts with other objects. After identifying attributes and behavior, one has a clear idea of the components of a useful class. If it is difficult to identify the responsibilities of a class, it probably does not represent a well-defined entity in the

program. If a class does nothing more than encapsulate processes, that class may be replaced by a function. If a class does not define any behavior, it may just need to be defined as a data structure.

While identifying relationships among the classes, it is necessary to consider how a class performs its assigned behavior. Does this class depend on other class' behavior? Or does this class contain the general information of other classes? Does these classes need to work together?

By performing the above steps, one will understand what classes are needed by the program, their similarities and differences, as well as their relationship with other classes. The classes can now be organized using inheritance. Inheritance makes the program clearer and more logical. Some of above steps may need to be repeated during the design process. Finally, it will be possible to develop a clear picture of these classes: the attributes and behavior of every class, the relationship between classes, the hierarchies of the classes. The final task is to write the detailed implementation of each class in the programming language.

Chapter 3

Program Structure

3.1 Introduction

The object oriented programming technique was used to develop the program. This chapter presents an overview of the classes that were developed. These classes can be divided into two types: structural classes and interface classes. The major classes in the program are described. In order to identify the classes in the program and the corresponding objects in a real structure, the names of all classes are underlined. Also the names of the major functions in the classes are italicized.

3.2 Program Development Process

As mentioned in Chapter 2, the process of designing an object oriented program is quite different from that for a procedural program. The design process requires both high level and low level abstractions. The first step is to abstract the program's main features and create the major classes which outline the structure of the program. In the process of developing this program, the MFC classes were used to generate the framework of application. The MFC library provides all of the functionality for the user interface. Thus, it was only necessary to concentrate on developing the classes needed for steel design and to connect these with the application framework.

The structural classes represent objects in a real structure. The class names and the relationship between the classes are also in accordance with these found in the real

world. For designing steel frames, the major classes are Joint, Member, LoadComb (load combination), MembLoad (member load), and JtLoad (joint load).

The document class which is derived from the base class CDocument in the MFC library is responsible for loading, storing, and processing data. The basic connection between the application framework and the structural design classes is through the document class. There are several collection classes in MFC which facilitate the process of connecting of the structural classes with application framework classes, such as CObArray. An object of CObArray can store an array of objects of a class derived from the base class CObject. Using objects of these collection classes makes it possible to manipulate and serialize data stored in objects of classes inherited from CObject. In this project, all objects of the Joint, Member, LoadComb, MembLoad and JtLoad classes are stored using the objects of CObArray class. These CObArray objects are declared as data members in the document class. This makes the structure of the program and the connection between the interface classes and the structural classes easy to understand.

Another important task is to design dialog boxes and control bars for user interface. Although substantial support for these tasks is built into the MFC library. Considerable effort is still required in developing these dialog boxes and controls.

3.3 Overview of Classes

The classes in the program are divided into two groups: structural classes and interface classes. Structural classes represent objects in a real structure. The class hierarchies are built according to the relationships that exist between real-world objects. The names of these classes are the same as in a real structure. The Member class has two derived classes Beam and Column. Each object of class Beam or Column represents a

member in the structure. Loads on the structure are represented by MembLoad and JtLoad classes. The MembLoad class has several subclasses such as ConcLoad, UniLoad and LinearLoad classes for representing different types of member loads. An object of class Structure stores general information about the structure.

The interface classes are developed from the skeleton classes generated by the Visual C++ compiler. These classes include the application class which manages the entire application, the main frame class which represents the application's main window, the document class which stores application data, the view class which displays results and various dialog classes for obtaining input data.

3.4 Structural Classes

The major structural classes used in this program are Structure, Joint, Member, Beam, Column, JtLoad, MembLoad and LoadComb. The concept of inheritance is used to define these classes. The program also defines several data structure variables, such as MIncid for representing member incidences, MemberEndForce for representing member end forces, and Displacement for representing joint displacements. These data structures are declared as data members in the structural classes mentioned above.

All of the structural classes are defined as subclasses of the MFC base class CObject. Each structural class contains a function called "serialize". This function overrides the serialize function defined in the base class CObject. The serialize function provides the capability for reading and writing data to disk. Most functions in the structural classes are declared as public.

3.4.1 Structure Class

The class Structure contains the general description of the structure. The data stored in the Structure class includes: structure type (plane frame or space frame), frame type (braced frame, unbraced frame, braced in X direction, braced in Y direction), total number of joints, total number of members, total number of supports, total number of load cases and total number of load combinations. The data members in the Structure class are all private. A series of functions are used for assigning values to the member variables and for transferring information from the Structure class to other objects.

3.4.2 Joint Class

The Joint class stores the joint number, joint type (fixed, free, or pinned for supports), joint coordinates and joint constraints. The data structure Displacement is declared as a data member of the Joint class. The data structure Displacement contains the following members: load case number, displacement and rotation in each direction.

3.4.3 LoadComb Class

The LoadComb class stores information on load combinations. Each object of the class represents one load combination. The total number of the load combinations for design and the description of each combination is obtained using a dialog box. The data stored in the class includes: number of load cases in this combination, load case number for each load case, and load factors corresponding to each load case. The load combinations are defined according to the LFRD specification. The load factors in each load combination can not be changed. The SetArray function initializes the load factors and load case arrays, and the Get Data functions provide the necessary support for transferring loading information to other objects.

3.4.4 Member Class

The Member class is the base class for both the Beam and Column classes (see Fig. 3.1). It contains the information that is common to both the Beam and Column classes, such as member type (beam or column), member length, material properties, design section constraints and section properties. All of the data members are declared as protected, which means that they can only be accessed by derived classes.

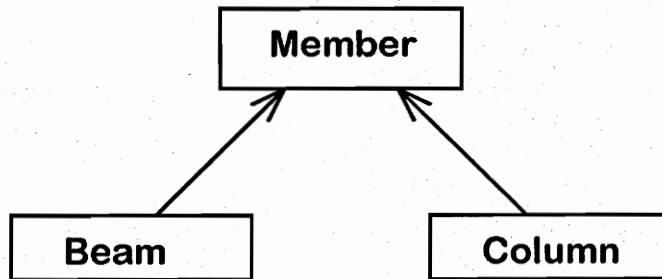


Fig. 3.1 Hierarchy of Member Classes

The Member class also contains two data members which point to the member's two end joints. This provides access to information regarding the two end joints. Another pointer points to the Structure object and allows access to information regarding the structure. Member end forces that are read from the analysis file are stored in the MembEndForce data structure. The members of the data structure MembEndForce are: load case number, joint numbers for end joints, axial forces, shear forces and moments at the ends of the member.

A series of Set data functions are declared in Member class to transfer information to the object. A partial list of these functions includes: SetWShape -- sets constraints on W sections; SetLength -- sets length of the member; SetMembForceArray -- initializes an array for storing objects of MembEndForce. A series of Get data functions transfer data

values to other objects. Examples of these functions are: `GetLength` -- returns the length of this member, `GetResults` -- returns design results.

3.4.4.1 Beam Class

The Beam class inherits all public and protected data members and functions of its parent class Member. In addition to storing information about its base class, the Beam object also contains special data members for analyzing and designing individual beams. These data members are private. A beam can be totally laterally braced, braced at the ends, or braced at several points between the two ends. Additional information needed by the Beam class includes:

1. The total number of equally spaced sections where moments and shears are calculated along the beam;
2. The number of concentrated loads, uniform loads, and/or linear loads on a beam;
3. The design moment and shear. These values are computed after an analysis of the beam is performed;
4. Moments at different brace points;
5. Section properties of the W sections.

The section properties are stored in the data structure aiscw which is declared as a private data member in class Beam. The shear and moment at each section due to each load combination are stored in two dimensional arrays. The corresponding maximum values are stored in one dimensional arrays.

The function *Analysis* computes moments and shears at fifteen sections along the beam. The pointers contained in the *Analysis* function provide access to loads, load combinations and load cases. The *Analysis* function calls several other functions, such as

Cal_Mom, and *Cal_Unild*, for computing moments and shears for a given load case, and *Impose* for superimposing the results.

The function *DesignForce* determines the critical load combination and the maximum shear and moment values. The function *SetBraceMom* calculates moments at each brace point while the *ComputeCb* function computes C_b values between brace points. The function *ComputeMn* calculates the nominal strength of a given trial W section according to the LRFD specification. The W section will be selected only when the section's flange and web local buckling criteria are limited to compact section only. The *OpenDataFile* function opens the AISCW binary data file and reads section properties of the trial W section. The *Design* function designs the beam. It calls the *OpenDataFile* function to read section properties, calculates the nominal strength and checks the section's weight. The function *CheckShear* verifies that the trial W section meets shear strength requirements. The design section is the W section which meets the strength requirements according to the LRFD manual section F, and has the least weight per foot.

3.4.4.2 Column Class

As with the Beam class, the Column class inherits member properties from its parent class Member. The additional data members in this class are K_x , K_y , the effective length factors for strong and weak axis buckling, and L_x and L_y , the unbraced lengths in the strong and weak directions.

The analysis file read by the program contains only first order analysis results. P- Δ effects are considered according to Section H-1 of the LRFD Specification. B_1X and B_1Y are B_1 moment magnifiers in X and Y direction respectively. B_2X and B_2Y are B_2 magnifiers in X and Y direction respectively. These factors are calculated first in order to

calculate the required design moments which take into account the second-order effects in both the in-plane (global X) and out-of-plane (global Y) directions. B_2X and B_2Y are calculated outside of the Column class. This is because the function for calculating B_2 requires consideration of in-plane and out-of-plane forces for the entire structure.

$$B_2 = \frac{1}{1 - \Sigma P_u \frac{\Delta_{oh}}{\Sigma HL}}$$

where ΣP_u is the factored axial compression load for all columns in the frame in a story subject to sway, Δ_{oh} is the translation deflection of the story under consideration under service horizontal loads, ΣH is the sum of all story horizontal service forces producing Δ_{oh} , L is the story height.

The effective length factors, K_x , K_y are also calculated outside of the Column class. For designing plane frames, K_x , L_x correspond to the in-plane direction while K_y , L_y correspond to the out of plane direction. The default value of K_y set equal to 1 and L_y is set to the length of the member. However, these default values can be changed from the Column Parameters dialog box. For designing space frames, the program provides four frame types. These are braced frame, unbraced frame, frame braced only in the X direction, and frame braced only in the Y direction. For an unbraced frame, the program calculates K_x and K_y values for each column. For braced frames, both K_x and K_y values are set equal to 1 for each column. If the frame is only braced in the X direction, K_x is set equal to 1 and K_y is calculated. For a frame braced in the Y direction K_y is set equal to 1 and K_x is calculated. The K_x and K_y values for each column are computed using the nomograph equation [Adeli, 1989].

$$\frac{G_A G_B (\pi / K)^2 - 36}{6(G_A + G_B)} = \frac{\pi / K}{\tan(\pi / K)}$$

Here A and B refer to the two ends of the column, and

$$G = \frac{\sum \frac{I_c}{L_c}}{\sum \frac{I_g}{L_g}}$$

where I_c is the moment inertia and L_c is the unbraced length of a column section, I_g the moment inertia and L_g the unbraced length of a girder. The I_c , I_g values are taken from the preliminary W sections read from the analysis file. The summation is for all members connected to the joint under consideration and lying in the plane of buckling. The *Cal_K* function in the document class solves this equation using the bisection technique and computes the K_x and K_y values for each column.

The *SetKxLx* and *SetKyLy* functions set the values of K_x , L_x , K_y , L_y for each column. The *SetB2* function enters B_2 values while the *SetCombEndForce* function sets arrays of the factored axial force and factored moments at the ends of the columns for different load combinations. The *ComputePn* function calculates the nominal axial capacity of a trial W section, while the *ComputeMn* function calculates the nominal flexural capacity of a trial W section for strong axis bending. The *ComputeMny* function calculates the nominal capacity for weak axis bending. The *CalPe* function calculates Euler load for the trial W section. The *OpenDataFile* function reads the section properties of the trial section from the data file. The *Design* function calls *OpenDataFile* to get a trial W section, checks its strength and weight, and checks if the W section satisfies both the interaction equations for each load combination and the least weight

requirement. The final W section selected is the heaviest W section among these sections selected for different load combinations.

3.4.5 JointLoad Class

The JointLoad class stores information about a joint load. The information stored includes load case number, joint number, type of force (force or moment), load direction, and load magnitude. All the above data values are private. The *SetJointLoad* function assigns values to the data members and a series of Get Data functions transfer data to other objects.

3.4.6 MembLoad Class

The MembLoad class is the base class for all member load classes. The data members shared by all classes inherited from the MembLoad class are: load case number, member number, and load type. The base class MembLoad provides functions for setting and returning these data values. Figure 3.2 shows the hierarchy of the load classes.

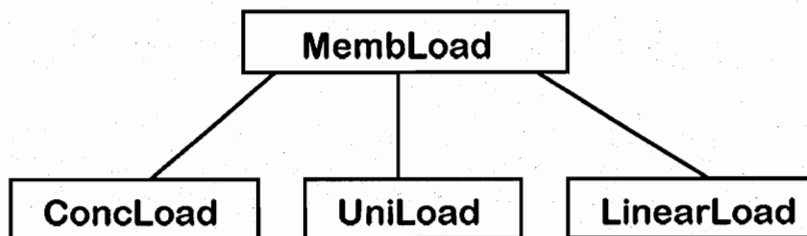


Fig. 3.2 Hierarchy of Load Classes

3.4.6.1 ConcLoad Class

The ConcLoad class inherits the properties of its base class. Its own data members are magnitude of the load and the distance to the left end of the member. The *SetConcLoad* function assigns the data to ConcLoad objects and the *GetConcLoad* function transfers the data to other objects.

3.4.6.2 UniLoad Class

Similar to the ConcLoad class, the UniLoad class inherits the properties of the base class MembLoad. Its own data members include the intensity of the uniform load, the distances to the start and end of the load from the left end of the member. The *SetUniLoad* function assigns the values to the private data members of the objects of UniLoad class and the *GetUniLoad* function transfers the data to other objects.

3.4.6.2 Linearload Class

The LinearLoad class also inherits the properties of the base class MembLoad. Its own data members include the intensity of loads at the start and end point, and distances from the left end of the member to the start and end point of the linear load.

3.5 Interface Classes

As described in Chapter 2, the Visual C++ compiler, AppWizard can create the basic framework for the application. The four basic classes in this application framework are: CSteelfmDoc, CSteelfmView, CMainFrame and CSteelfmApp. Skeletons for the dialog classes corresponding to the various dialog boxes are generated by the ClassWizard. Table 3.1 lists the interface classes in the program.

3.5.1 Document Class

The document class, CSteelfmDoc, is responsible for creating, and maintaining instances of the various structural classes and other data in the program. Almost all of the command functions for operating dialog boxes and for controlling the program are also in the document class. The *SetBeamGroup* and *SetColumnGroup* functions set the beam and column mark groups. These groups are created through the BeamMark Group and the ColumnMark Group dialog boxes. The maximum number of beam mark groups is ten. The maximum number of column mark groups is also ten. The maximum number of beams or columns in a mark group is limited to thirty. Objects of the Beam and Column classes are stored in MembArray which is an object of the collection class CObArray. In the same way, JtArray, ULdArray, CLdArray, JtLdArray are also objects of CObArray. They store objects of Joint, UniLoad, ConcLoad, JointLoad classes respectively. The MIIncid object array stores objects of the MembIncid data structure which represent member incidences. The LdComb object array stores objects of the load combination class.

The function *ReadAnFile* reads in the analysis file and creates the objects of the Beam, Column, Joint, and MembLoad and JtLoad classes, and then stores them into the corresponding objects of the CObArray class. The *Cal_K* function calculates the effective length factors K_x , K_y for each column. The *Cal_B2* calculates the sideways magnification factors for each column. The *InitDocument* function is responsible for initializing the document object while the *DeleteContent* function provides a place for deleting these dynamically created objects and arrays. The *BeamOutput* and *ColumnOutput* functions create text display for beam and column results in the view window. The *Draw* function is responsible for drawing the structure on the screen. The

Serialize function serializes the data members in the document class and calls the *serialize* function for the objects of the CObArray class.

3.5.2 View Class

The CSteelfmView class is responsible for displaying the data of the document class. The *OnDraw* function can display the structure or show design results for beams or columns on the screen.

3.5.3 Dialog Box Classes

Fig 3.3 shows the hierarchy of the dialog classes. All the classes for creating dialog boxes are generated from the base class CDialog. These classes are responsible for obtaining user input. A brief description of these classes is given in Table 3.1.

Table 3.1 Interface Classes

Interface Class	Base Class	Purpose
CMainFrame	CMDIFrameWnd	Main application window
CSteelfmApp	CWinApp	Initializes all windows, Manages commands
CSteelfmView	CScrollView	Displays document data, also processes mouse and keyboard input
CSteelfmDoc	CDocument	Manages and stores application data
BeamMarkDlg	CDialog	MarkGroup dialog box allows user to assign mark groups for beams
BmNumDlg	CDialog	Beam member number dialog box for specifying the beam for which design results are displayed
BmParaDlg	CDialog	Beam Parameters dialog box for choosing beam design parameters
BracePosDlg	CDialog	Brace Position dialog box for setting the positions of brace points
ColNumDlg	CDialog	Column member number dialog box for specifying the column for which design results are displayed
ColMarkDlg	CDialog	MarkGroup dialog box for specifying mark groups for columns
ColParaDlg	CDialog	Column Parameters dialog box for entering column design parameters
CombLdDlg	CDialog	LoadComb dialog box for entering load combinations
StructDlg	CDialog	Structure dialog box for selecting structure type

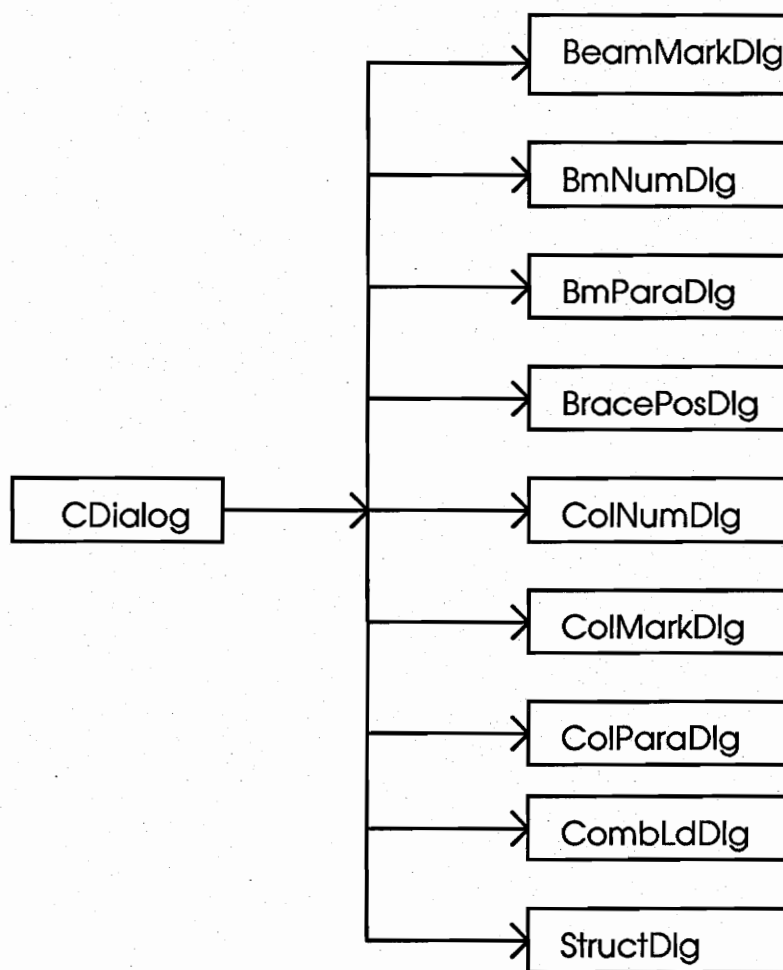


Fig 3.3 Classes for Dialog Boxes

In this chapter, the major classes in the program were described. The application contains two types of classes: structural classes and interface classes. The description of the structural and interface classes was presented and the interaction between the various classes was described.

Chapter 4

Design Procedure and Program Description

4.1 Introduction

In this chapter, an overview of the procedure for the design of steel frames is presented. This is followed by a description of the program. The various menus and dialog boxes are described. This chapter serves as a user's guide to the program.

4.2 Overview of Design Procedure

This program can be considered as a post processor for the design of steel frame, since its major purpose is to design the members in the frame. Beams and columns in the frame are designed based on the LRFD steel design specification [AISC 1986]. The program reads in the output files from a frame analysis program. It then modifies these files to include second order effects in accordance with Section H-1 of the LRFD steel design specification. The program can handle as many members in the frame as available computer memory. Default values for beam and column design parameters are provided to minimize data entry. The menus and dialog boxes are designed to make the program easy to use and understand.

When the program begins execution, an open file dialog box is displayed. This allows the user to select the analysis input file. The information read from the analysis input file is the following: the type of structure (plane frame or space frame), number of joints, number of members, number of supports, number of load cases, joint coordinates,

member incidences, member properties (section area, moments of inertia), joint releases if any, applied loads and member end forces for each load case. The number of load combinations is not read, since the load information is obtained later.

The steel frame design program contains two separate modules: one for the design of plane frames, and one for the design of space frames. The major difference between these two modules is that effective length factors, K_y (out-of-plane direction) and unbraced lengths, L_y , can be entered in the first module, but are calculated in the second module.

4.2.1 Structure Features

For a plane frame, it is necessary to specify whether the frame is braced or unbraced. For a space frame structure, the user can select from one of four frame types: braced frame (braced in both in-plane and out-of-plane directions), unbraced frame (unbraced in both directions), braced in X direction (in-plane direction), and braced in Y direction (out-of-plane direction).

4.2.2 Beam Design Parameters

The default beam design parameters include the modulus of elasticity of steel, and the steel strength. The default values are 29000 ksi and 36 ksi respectively, and can be modified later. The beam and column sections are limited to W-shapes. A specific group of W sections, for example, "W10", can be specified in which case the section will only be selected from the W10 sections, or "All" in which case all W sections will be searched.

Brace conditions for the beams are also required. Three brace conditions are provided in this program: fully laterally supported, laterally unbraced except at the two

ends, and braced at several internal brace points. If the last option is chosen, the locations of the internal brace points have to be entered.

4.2.3 Column Design Parameters

In the design of columns in a plane frame, the strong bending axis of the column is always set to resist in-plane bending. The effective length factor K_x for strong bending is calculated by solving the nomograph equation mentioned in Chapter 3. If the frame is a braced frame, K_x is equal to 1 for all columns. The corresponding unbraced length L_x is set equal to the length of the member in both cases. The default effective length factor K_y for weak-axis bending is set equal to 1 by assuming that the frame is braced out-of-plane. The default value for the unbraced length L_y is equal to the length of the member. However, the K_y , L_y values can be changed as needed.

In the design of space frames, K_x , K_y are calculated or set to one according to the type of frame selected. The beta angle for the column orientation is set to 0.0 by default, which means that the strong axis of the W section is in the out-of-plane direction. If beta is set to 90.0, the strong axis is in the plane. Only these two values of beta are considered in the program.

4.2.4 Mark Groups

All members in a mark group have the same section. Beams or columns with similar properties and loading can be placed in mark groups in the program. The design section for a mark group is the heaviest W section of all of the individual sections designed for the members in the mark group. The maximum number of beam mark groups is ten. The maximum number of column mark groups is also ten. The maximum number of members in each group is thirty.

4.2.5 Load Combinations

The load combinations considered for the design are decided by the user. As specified in the LRFD Steel Design Manual [AISC 1986], the program lists the following six load combinations:

- (1) $1.4 D$
- (2) $1.2 D + 1.6 L$
- (3) $1.2 D + 1.6 L + 0.5 S$
- (4) $1.2 D + 1.3 W + 0.5 L$
- (5) $1.2 D + 1.5 E + 0.5 L$
- (6) $0.9 D - 1.3 W$

where D = dead load, L = live load, S = snow load, W = wind load, E = earthquake load.

Load combinations are specified by first selecting from a list of the basic load cases considered in the analysis. This list is displayed in the load combination dialog box. The load combinations should be obtained by only using these basic load cases. For example, if no earthquake load case is displayed in the load combination dialog box, the fifth load combination can not be selected. An error message is displayed if the wrong load combination is selected.

4.2.6 Beam Design

When the "Design Beams..." menu item is selected, the program designs all the beams in the frame. Shear forces and bending moments at fifteen equally spaced sections along the beam are computed for all load combinations. The maximum negative or positive moment is selected as the design moment.

For beams that are braced at several points, the program determines the W section for each segment between two consecutive brace points and then selects the heaviest W

section among these as the design section for the whole beam. If the selection of the beam section is restricted to a specific group of W-section, for example, W-10's, the program will only search for a W section from this group. If a section is not found from this group, a message is displayed in dialog box that the chosen W-section group is too small. If the smallest W section in the group satisfies the required design strength, the program displays a message informing the user that a smaller group may be more economical. The design section is the W section that meets the strength requirements and has the least weight per foot. The shear strength of the selected W section is also checked. If the shear check fails, the program continues searching to find a larger section that meets both flexural and shear requirements.

4.2.7 Column Design

The program calculates the effective length factors K_x , K_y (about both bending axes) using the data read from the analysis input file. In order to design a column, the program needs to calculate the required compression strength and required flexural strength for each column. Since the analysis input file contains only first order elastic analysis results, the approximate design moments including second order effects have to be determined if the frame is unbraced. The required design moments are calculated in accordance with Section H-1 of the LRFD Specification [AISC 1986]:

$$M_u = B_1 * M_{nt} + B_2 * M_{lt}$$

where M_{nt} is the required flexural strength of the member assuming there is no lateral translation of the frame, M_{lt} is the required flexural strength of the member as a result of lateral translation of the frame only, and B_1 , B_2 are the moment magnification factors.

The value of $B_1 * M_{nt} + B_2 * M_{lt}$ is calculated at each end of the column. The required design moment M_u is the larger of these two combined end moments. M_{lt} is the larger of

the two factored end moments under lateral load case only, and the moment caused by the asymmetric gravity loading is neglected. M_{nt} equals the larger of the two end moments obtained by combining all the factored moments caused by gravity loads for a given load combination. The required design moments about both axes M_{ux} , M_{uy} , and the factored axial forces P_u , are calculated for each load combination. B_1 , B_2 are also calculated for both bending directions. The W section selected for the column under each load combination is the trial section which satisfies the interaction equations and also has the least weight per foot. The interaction equations in Section H-1 of the AISC LRFD Specification [AISC 1986] are:

For $\frac{P_u}{\phi P_n} \geq 0.2$:

$$\frac{P_u}{\phi P_n} + \frac{8}{9} \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}} \right) \leq 1.0 \quad (H1-1a)$$

For $\frac{P_u}{\phi P_n} \leq 0.2$

$$\frac{P_u}{2\phi P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}} \right) \leq 1.0 \quad (H1-1b)$$

where P_u is the required axial strength, M_{ux} , M_{uy} are the required flexural strengths about both X and Y axes, P_n is the nominal axial strength, M_{nx} and M_{ny} are the nominal flexural strengths about X and Y axes.

The final W section selected for a column is the largest section from among the section designed for each load combination. For column mark groups, the W section

selected for the mark group is the heaviest section among all of the columns in the mark group.

4.2.8 Design Results

Design results can be displayed on the screen. They can also be saved as a file or printed directly.

4.2.8.1 Beam Design Results

Beam results include member number, member length, W section selected and its properties, total weight of the beam, nominal flexural strength, nominal shear strength, modulus of elasticity of steel E , yield strength of steel F_y , lateral bracing conditions, critical load combination for design, and design moments and shear forces at fifteen sections along the beam.

4.2.8.2 Column Design Results

Column design results consist of member number, member length, modulus of elasticity of steel E , yield strength of steel F_y , effective length factors K_x and K_y , unbraced lengths L_x and L_y , selected W section and its properties, beta angle, critical load combination for design, applied end moments and axial forces for design, nominal axial strength, and nominal flexural strength about both axes.

4.3 Program Description

The program runs under the Windows environment. Menus and toolbars are designed to facilitate the operation of the program. Various dialog boxes are used to for communicating with the user. The toolbars allow easy access to the corresponding menus and are easier to manipulate than menus. For example, by clicking the corresponding icons on the toolbar, it is possible to view the structure, beam design results, or column design results. The program uses the multi-document MFC template, which makes it possible to open several files and work on several structures at the same time. A brief description of the menus is given below.

4.3.1 Main Menu

When the program begins execution, an Open File dialog box is displayed to allow the selection of the analysis input file. All analysis files have an extension of “.anl”. These input files have the same format as the output file that is generated from the CADKEY frame analysis program. The main menu consists of three pull down menus, File, View and Help. The File menu has four menu items: New, Open, Recent Files, Exit. When the New menu item is selected, the Open File dialog box is displayed for analysis input file to be read in . The Open menu item from the File menu also opens an Open File dialog box. However this dialog box is for opening files with an extension of “.des”, which are design files previously saved by the program.

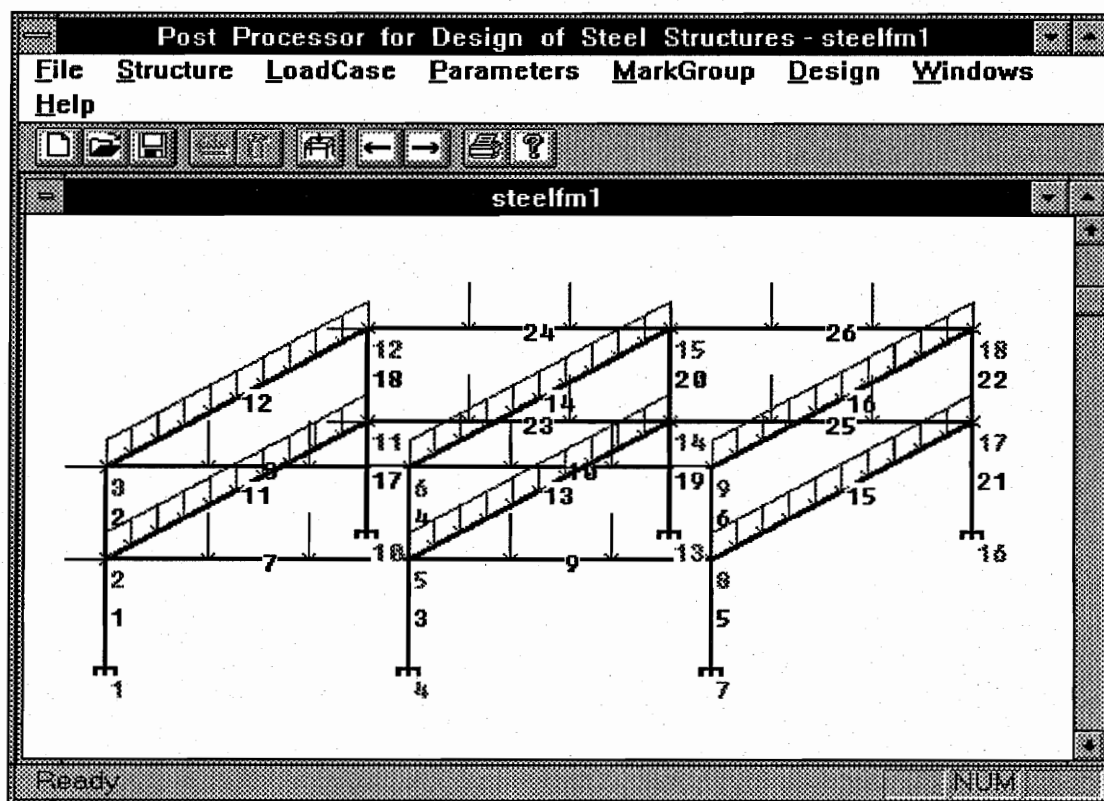


Fig. 4.1 Main Application Window

4.3.2 Application Window Menu

After reading an input file or a design file from disk, the main frame window is replaced by the application window. The view window in the application window displays the structure. Fig 4.1 shows the main application window with the main menu and the toolbars. The main menu contains eight pull down menus: File, Structure, LoadCase, Parameters, MarkGroup, Windows, and Help. These pull down menus allow access to different parts of the program. The toolbar icons have the same functionality as the corresponding menu items and are more convenient. From left to right, the first three toolbar icons correspond to the File menu items: New, Open, Save. The fourth and fifth toolbar icons are grayed (disabled) initially and become active after the Design Beams or

the Design Columns menu items are selected. The fourth toolbar icon is for displaying beam design results and the fifth toolbar icon is for displaying column design results. The sixth toolbar icon is for displaying the structure, while the seventh and eighth toolbar buttons select the previous and next beam or column. The ninth and tenth toolbar buttons correspond to the Print and About menu items.

4.3.2.1 File Menu

The File menu has ten pull down menu items: New, Open, Close, Save, Save As, Print, Print Preview, Print Setup, Recent Files and Exit. The New, Open perform the same functions as these described earlier. The Save As menu item opens a Save File dialog box to allow the contents of the current document to be saved in a file with the extension “.des”. When Close is selected, the program displays a dialog box to notify the user if the current document needs to be saved. If the file needs to be saved, the save File dialog box is displayed. The program then closes the currently active document view window. The program prints design results when Print menu item is chosen.

4.3.2.2 Structure Menu

The Structure menu contains only one pull down item: Features. When selected, a Structure Features dialog box opens to allow the user to define the type of structure, that is, braced frame or unbraced frame. The Structure Features dialog box for a plane frame is shown in Fig. 4.2. The Structure Features dialog box for a space frame is shown in Fig. 4.3.

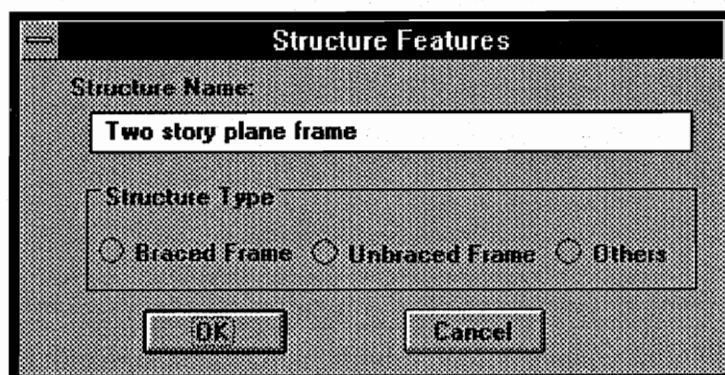


Fig. 4.2 Structure Features Dialog Box for Plane Frame

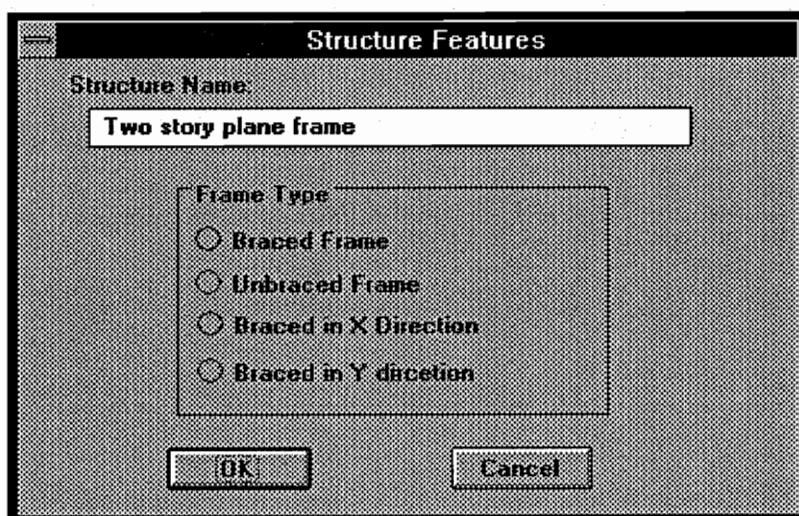


Fig. 4.3 Structure Features Dialog Box for Space Frame

4.3.2.3 LoadCase Menu

The LoadCase menu has one menu item: LdCombCase. When selected, the Load Combination dialog box is displayed. On the left side of the dialog box, is a list of the basic load cases read from the analysis file. Load combinations can be formed by selecting from these listed basic load cases. Fig. 4.4 shows the Load Combination dialog box.

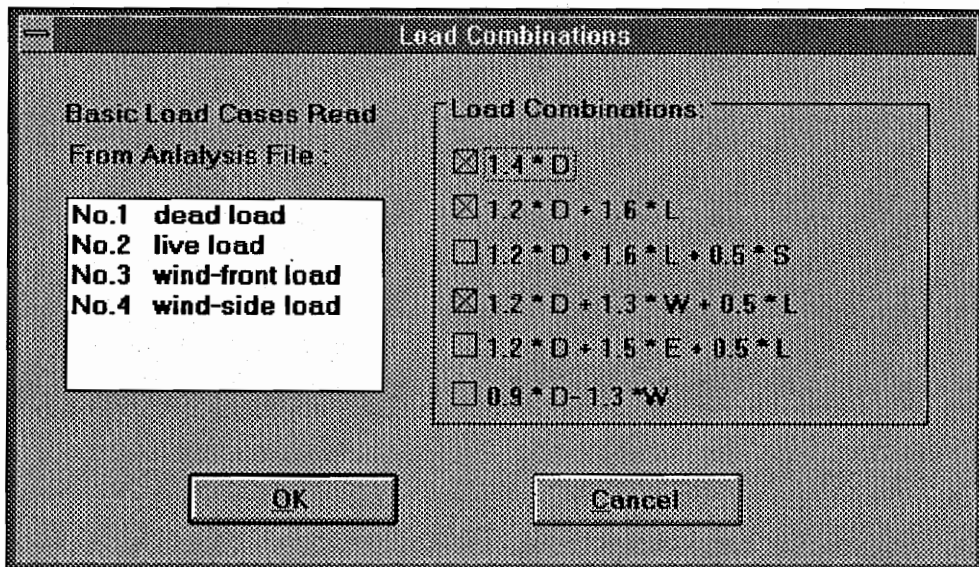


Fig. 4.4 Load Combinations Dialog Box

4.3.2.4 Parameters Menu

The Parameters menu contains two menu items: Beam Design Parameters and Column Design Parameters. When Beam... is selected, the Beam Design Parameters dialog box is displayed (see Fig. 4.5). The default brace condition is set to Total Laterally Supported. If the Internal Brace Points option is selected, it is necessary to enter the number of internal brace points. The default number of internal brace points is zero. A message is displayed to input the number of the internal brace points. After the number of the braces is entered, the locations of the brace points are entered through the Brace Positions dialog box. The positions of the brace points must be entered in order from left to right along the beam. A message is displayed if the positions are entered in the wrong order. Fig 4.6 shows the Brace Positions dialog box.

When the Column... menu item is chosen, the Column Design Parameters dialog box is displayed. Column parameters are entered in this dialog box. The dialog box allows the user to enter the effective length factor K_y , and unbraced length L_y for a plane frame, and beta angle for a space frame. Fig. 4.7 shows the Column Design Parameters dialog box for plane frames. Fig. 4.8 shows the Column Design Parameters dialog box for space frames.

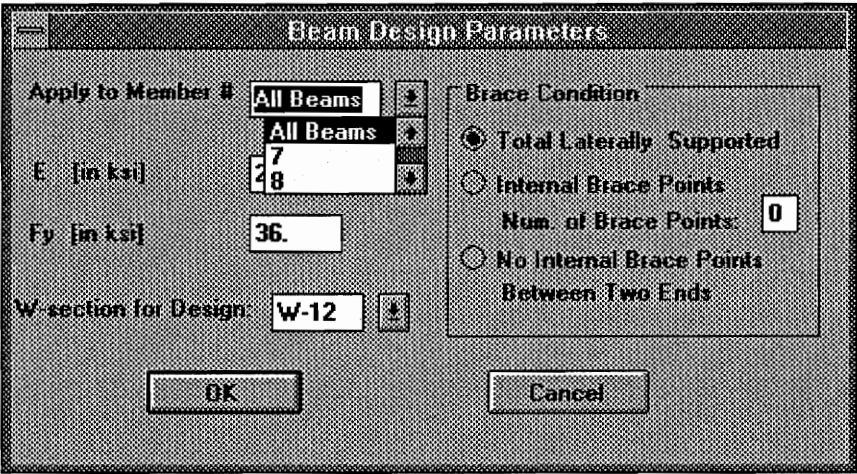


Fig. 4.5 Beam Design Parameters Dialog Box

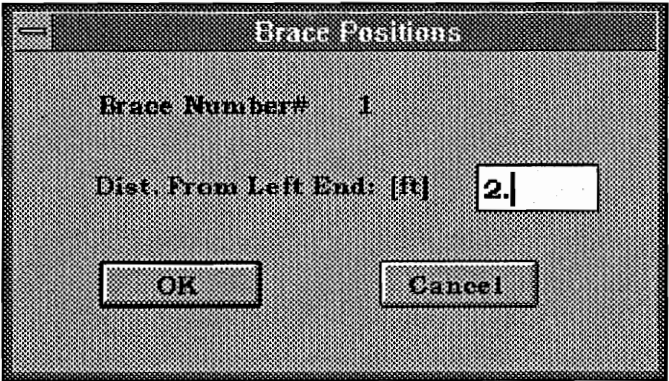


Fig. 4.6 Position of the Internal Brace Points Dialog Box

Column Design Parameters

Apply to member No: **All Columns**

E (in ksi) **29000.**

Fy (in ksi) **36.**

W-Sections : **ALL**
ALL
W-4
W-5

Effective Length in Out Plane Direction
 Ky = **1.**

Unbraced Length
 Ly (in ft.) = **Length**

Fig. 4.7 Column Design Parameters Dialog Box for Plane Frames

Column Design Parameters

Apply to member No: **All Columns**
All Columns
1
2

E (in ksi) **29000.**

Fy (in ksi) **36.**

W-Sections for Design: **ALL**

Initial Cross Section:
 Beta Angle : **0.**

Note:
 Beta Angle can be 0.0 (weak axis in the plane) or 90.0 (weak axis out-of-plane)

Fig 4.8 Column Design Parameters Dialog Box for Space Frames

4.3.2.5 MarkGroup Menu

The MarkGroup menu has two pull down menu items: Beam Mark Groups and Column Mark Groups. The Beam Mark Groups dialog box is shown in Fig. 4.9 and the Column Mark Groups dialog box is shown in Fig. 4.10. The Add button in the dialog box is for adding members to the mark group and Del button allows a member to be deleted from the mark group. The Next button is for entering members to the next mark group. The maximum number of mark groups is ten and the maximum number of the members in a mark group is thirty.

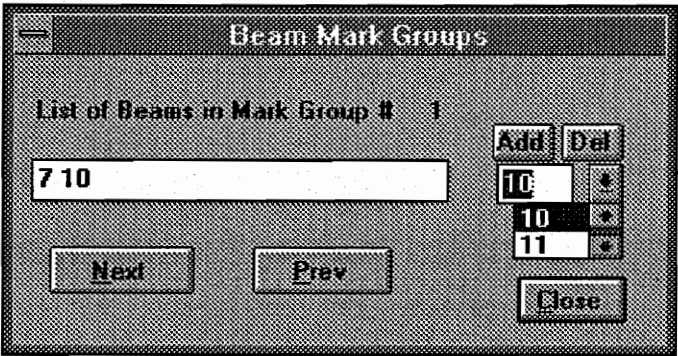


Fig. 4.9 Beam Mark Group Dialog Box

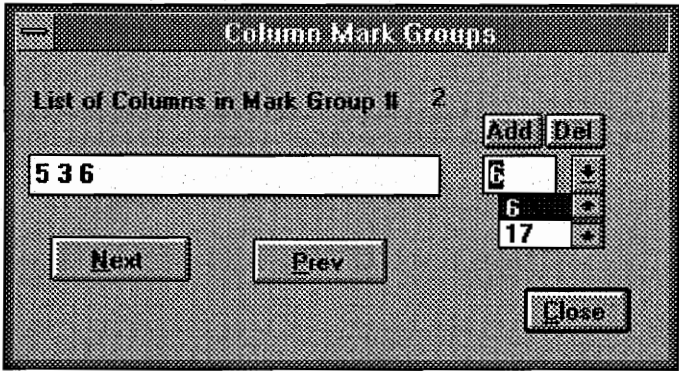


Fig. 4.10 Column Mark Group Dialog Box

4.3.2.6 Design Menu

The Design menu has two menu items: DesignBeam and DesignColumn. When DesignBeam is selected, the program designs all of the beams in the structure using the design process described in Section 4.2.5. Before designing the beams, the program first checks to make sure that the structure type, load combinations, and design parameters for each beam have been entered. If the necessary data has not be entered, message boxes are displayed to notify the user (see Fig. 4.11).

When the DesignColumn menu item is selected, the program designs all the columns in the structure following the procedure described in Section 4.2.6. Message boxes are displayed if the required data has not been entered.



Fig. 4.11 Message Box for Checking Design Parameters

4.3.2.7 Windows Menu

The Windows menu contains two items: Results and Display. The Results menu item has four pull down menu items: ShowBeams, ShowColumns, Show Next, Show Previous. When the ShowBeams menu item is selected, a dialog box is displayed to allow the user to specify the beam for which design results are to be viewed. Fig. 4.12 shows the Beam Number dialog box. The ShowColumns provides a Column Number dialog box

for selecting the column whose data are to be viewed. When Show Previous is selected or the corresponding toolbar icon is clicked, the results of the previous beam or column member are displayed.

Beam design results are shown in Fig 4.13. The design results shown are beam length, selected W section and its properties, brace conditions, critical load combination, and design moments and shear forces at the fifteen equally spaced points along the beam.

Column design results are shown in Fig. 4.14. The design results for the columns are length, W section selected and its properties, critical load combination, beta angle, effective length factors and unbraced lengths, applied axial forces and moments, and nominal flexural and axial strength of the column.

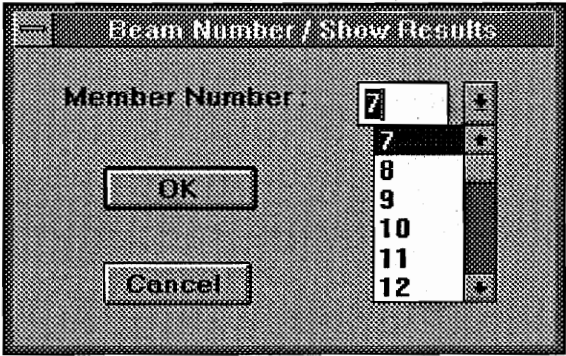


Fig. 4.12 Beam Number Dialog Box

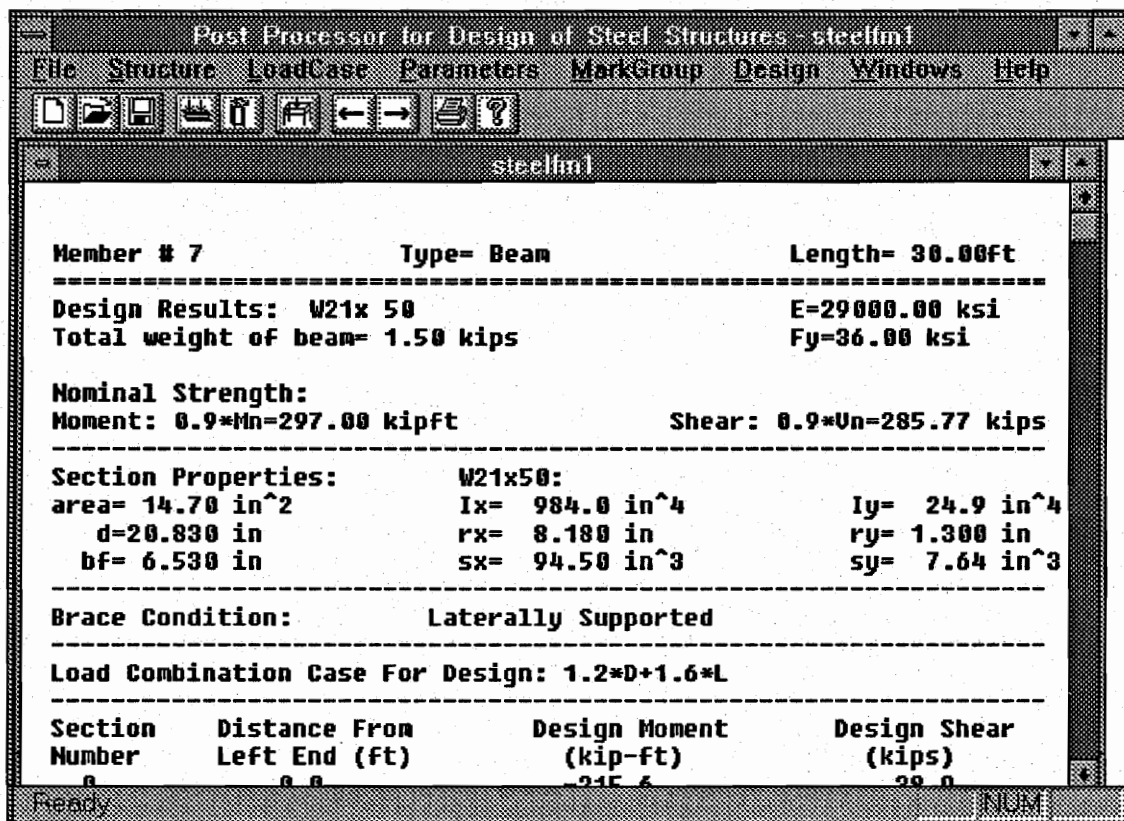


Fig. 4.13 Beam Design Results Window

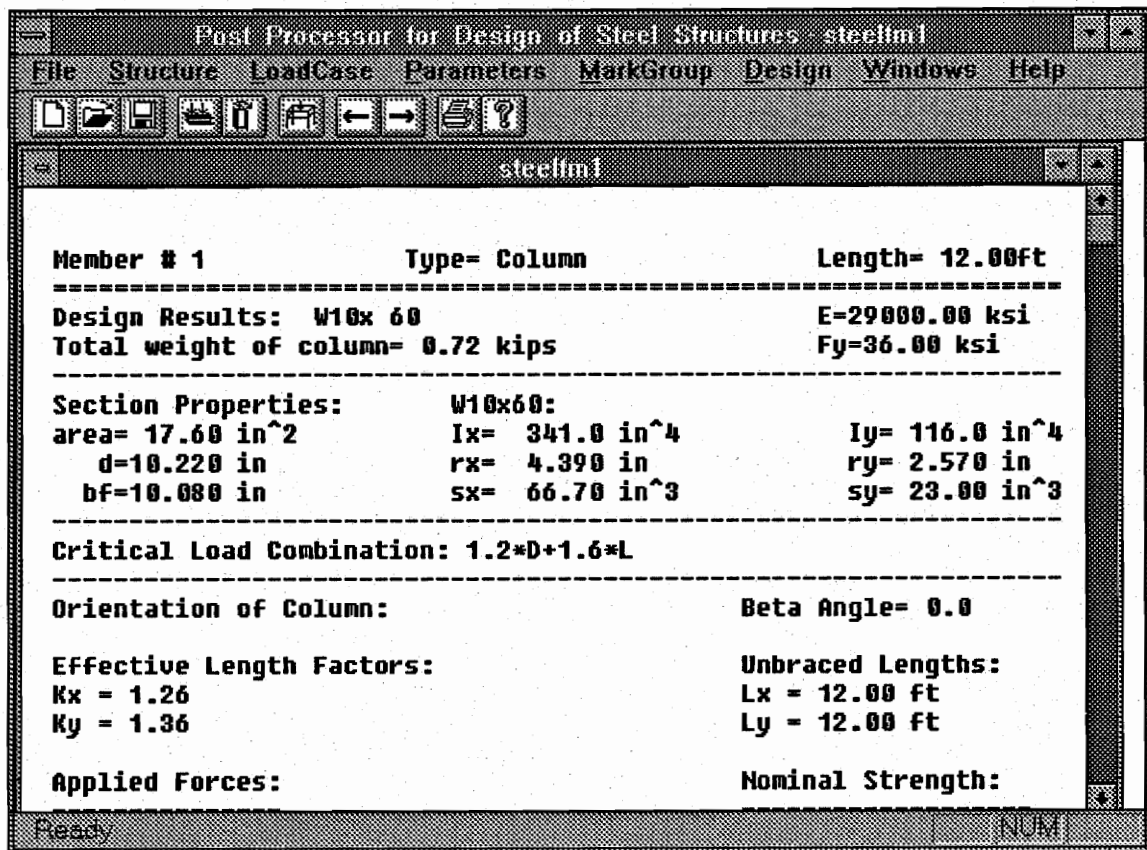


Fig. 4.14 Column Design Results Window

In this chapter, a description of the program was presented. The procedure used for the design of beams and columns was described. The operation of the program was discussed and results obtained from the program were described.

Chapter 5

Design Results and Discussions

5.1 Introduction

The program developed in this project was evaluated by comparing the design results obtained from the program for three test structures with a commercial program and by hand calculations. The first structure is a four span continuous beam. The second structure is a two bay, two story plane frame. The third structure is a two story space frame. In this chapter, the design results are presented and compared with those obtained from DAST (Version 11.0) [DAS Consulting, Inc.], which is an integrated analysis and design program running under DOS. The results of the hand verification for the first two test structures are presented in Appendix C.

5.2 Continuous Beam

5.2.1 Geometry and Loading

The first test structure is a four span continuous beam. A W-section was selected for each span. The geometry and loading are shown in Fig. 5.1. The exterior spans are 23.5 feet long, and the interior spans are 25 feet long. The exterior supports are pinned, and all interior supports are roller supports. The loading on the beam includes two load cases: dead load and live load. The dead load consists of an uniformly distributed load of

2.52 kips/ft plus the weight of the beam. The live load consists of an uniformly distributed load of 3 kips/ft and concentrated loads of 20 kips acting at the center of each span. Two load combinations are considered:

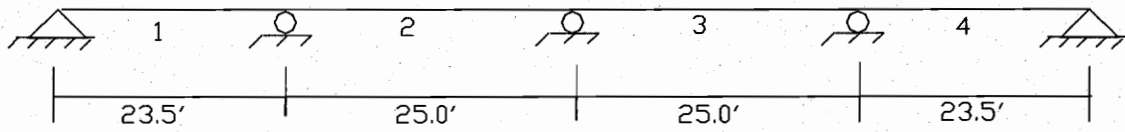
- 1) $1.4 D$,
- 2) $1.2 D + 1.6 L$.

5.2.2 Input and Output Files

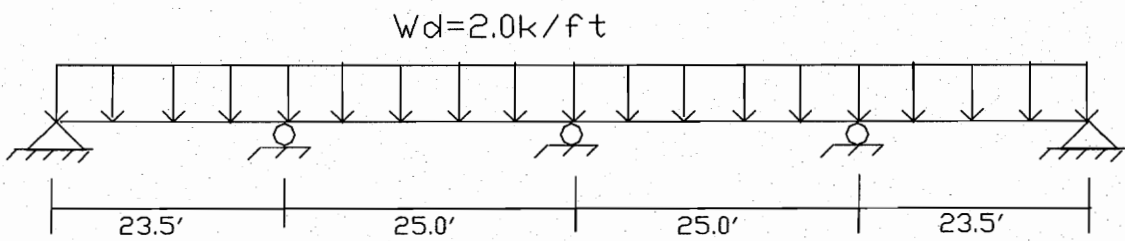
The input data file, which is an output file from the CADKEY frame analysis program, as well as the output file from the program are given in Appendix B.

5.2.3 Discussion of Design Results

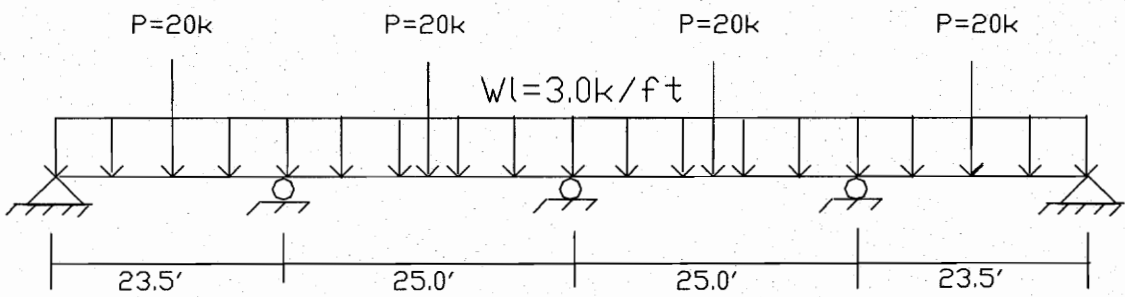
A commercial program, DAST (Version 11.0), is used to verify the results. The W sections selected by the program and by DAST are shown in Table 5.1. From Table 5.1, it is seen that the results for both programs are identical. The design sections obtained by hand calculations also support this conclusion. The results of the hand calculations are given in Appendix C.



(a) Beam Geometry



(b) Dead Load



(c) Live Load

Fig. 5.1 Continuous Beam

Table 5.1 Design Results for Continuous Beam

Beam Number	Program	DAST
1	W24x104	W24x104
2	W24x104	W24x104
3	W24x104	W24x104
4	W24x104	W24x104

5.3 Two Story Plane Frame

5.3.1 Geometry and Loading

Fig. 5.2 shows the two story plane frame designed by the program. The first story is 12 feet high, the second story is 10 feet high. The frame has two 30 feet wide bays. The frame is a moment resisting unbraced frame. All supports are fixed and all other joints are rigid. For column design, the effective length factors K_x for in-plane bending were calculated by the method described in Chapter 3. The effective length factors K_y for out-of-plane bending were taken equal to 1.

Fig.5.3 shows the dead and live loads acting on the frame. Two concentrated dead loads of 23.5 kips are applied at one third points of each beam. The live load consists of concentrated loads of 7.5 kips which are also applied at one third points. Fig. 5.4 shows the wind loads acting on the frame.

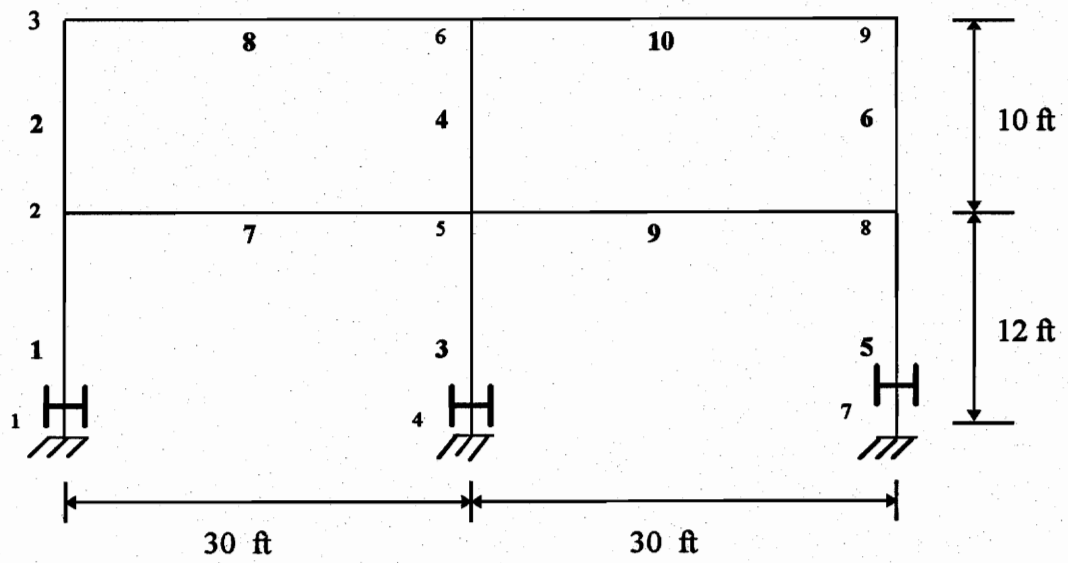


Fig. 5.2 Two Story Plane Frame

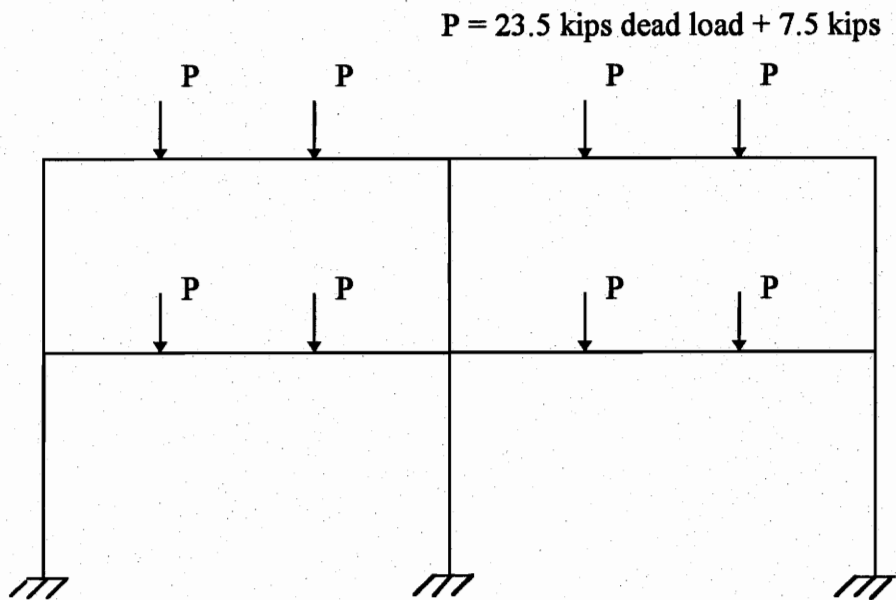


Fig. 5.3 Dead and Live Loads on Two Story Frame

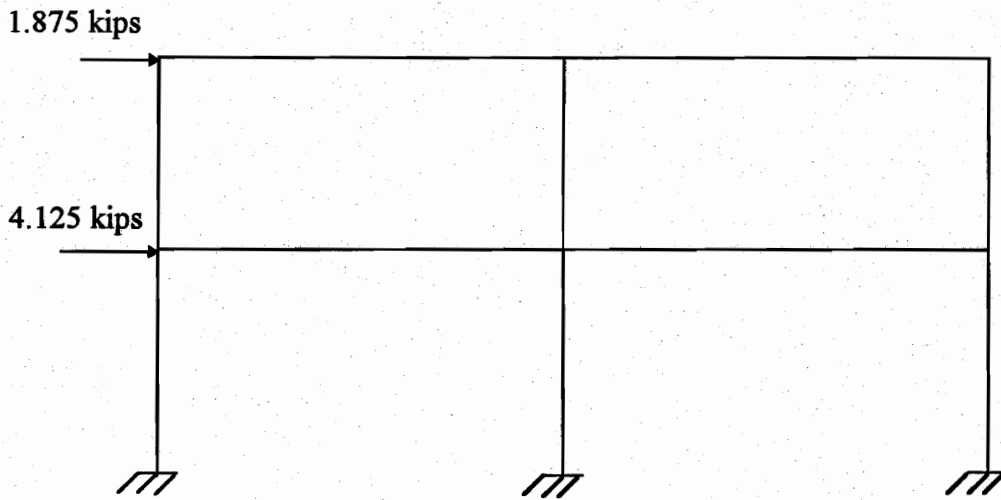


Fig. 5.4 Wind Loads on the Frame

The following load combinations are considered for the analysis and design:

- (1) $1.4 D$;
- (2) $1.2 D + 1.6 L$;
- (3) $1.2 D + 0.5 L + 1.3 W$;

where D = dead load, L = live load, W = wind load.

5.3.2 Input and Output Files

Appendix B shows the input data for the program, and the output obtained from the program.

5.3.3 Discussion of Design Results

A comparison of the beam and column sections obtained using the program with those obtained using DAST is given in Table 5.2 and Table 5.3 respectively. The results were obtained by searching through all W-sections in the data file without specifying any constraint on section selection. The beam sections given by both programs are identical. The column sections differ slightly for Columns #2 and #6. For both cases, DAST gave a slightly lighter section (W14x34 versus W12x40). One possible reason for this difference is that analysis results read in by this program are slightly different from those calculated by DAST. For practical purposes, the slight difference between column design results from the two programs is not significant. The results of the hand calculations agree with those obtained from the program, as shown in Appendix C.

Table 5.2 Beam Design Results for the Plane Frame

Member Number	Program	DAST
7	W18x86	W18x86
8	W18x86	W18x86
9	W18x86	W18x86
10	W18x86	W18x86

Table 5.3 Column Design Results for the Plane Frame

Member Number	Program	DAST
1	W8x31	W8x31
2	W12x40	W14x34
3	W8x31	W8x31
4	W5x19	W5x19
5	W8x31	W8x31
6	W12x40	W14x34

5.4 Two Story Space Frame

5.4.1 Geometry and Loading

Figure 5.5 shows the two story space frame designed by the program. The height from ground to first story is 12 feet, and from first story to second story is 10 feet. The frame has two 30 feet by 30 feet bays. The total length of the structure is 60 feet and the total width is 30 feet. All supports are fixed and all other joints are rigid. The frame is considered as a rigid frame. The effective length factors K_x , K_y are calculated by the program respectively. The beta angles for all the columns are zero, which means that the W section's weak bending axis is parallel to the global X axis (Fig. 5.5).

Figures 5.6 and 5.7 show the dead load and live load acting on the frame. The wind loads acting on the left side wall and on the front wall are represented by equivalent joint loads shown in Figures 5.8 and 5.9. The following four load combinations were considered in designing the structure:

- (1) $1.4 D$
- (2) $1.2 D + 1.6 L$
- (3) $1.2 D + 0.5 L + 1.3 W$ (side)
- (4) $1.2 D + 0.5 L + 1.3 W$ (front)

where D = dead load, L = live load, W = wind load.

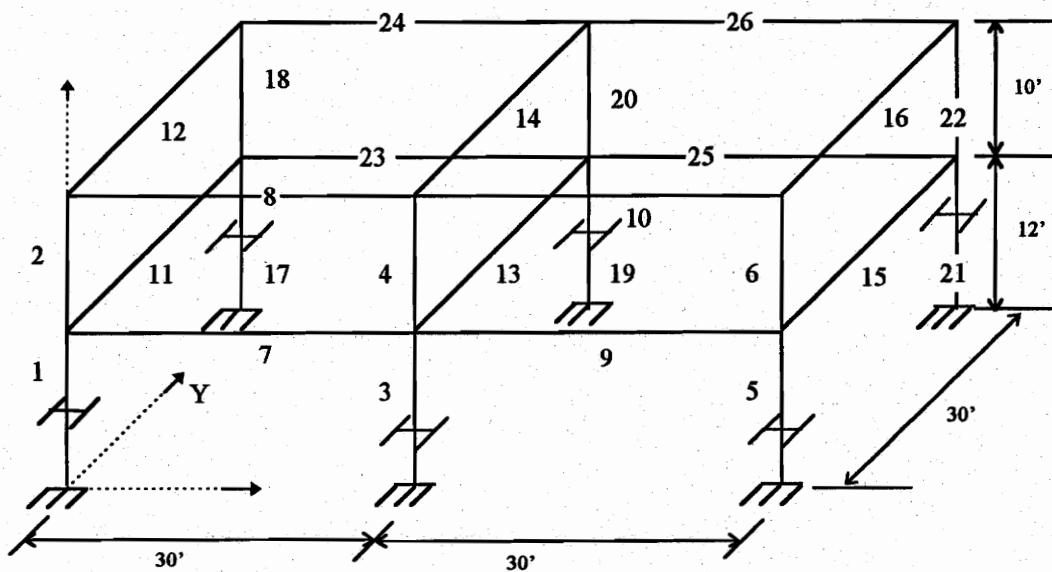


Fig. 5.5 Two Story Space Frame

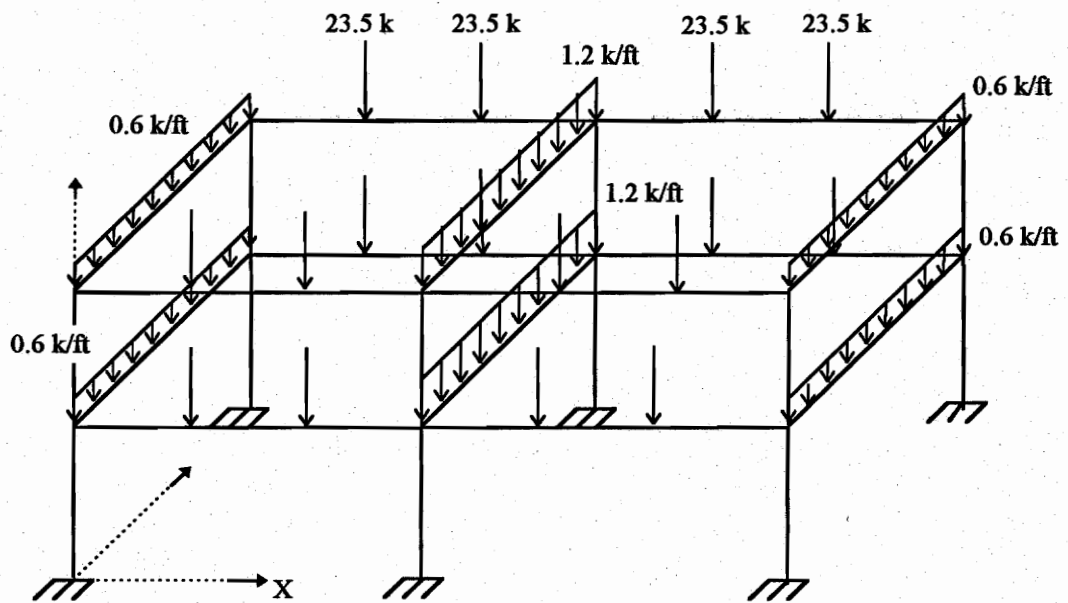


Fig. 5.6 Dead Loads on Two Story Space Frame

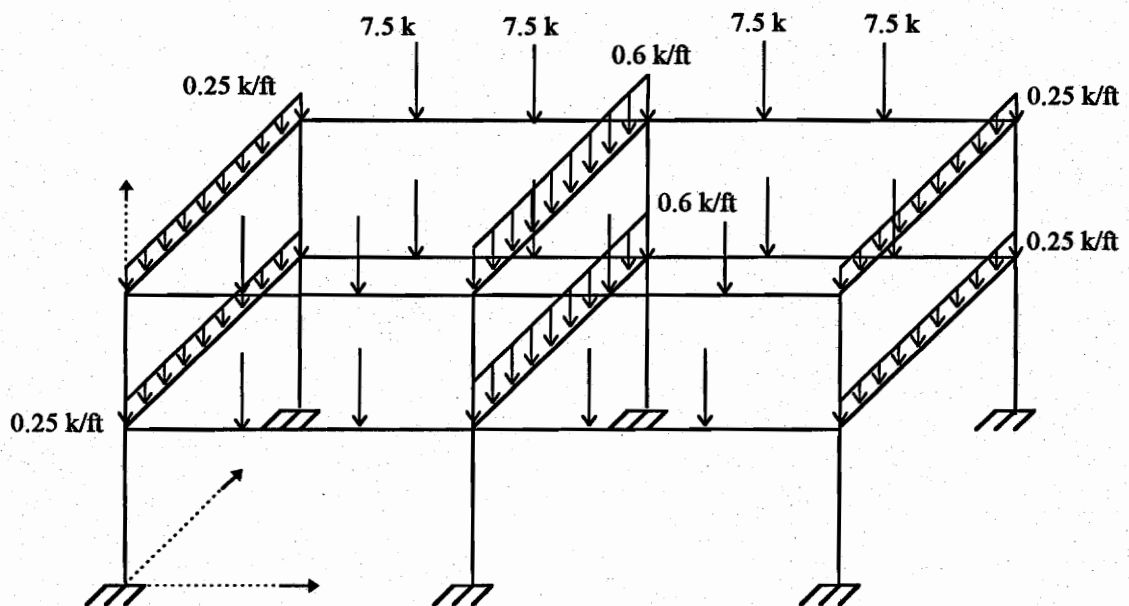


Fig. 5.7 Live Loads on Two Story Space Frame

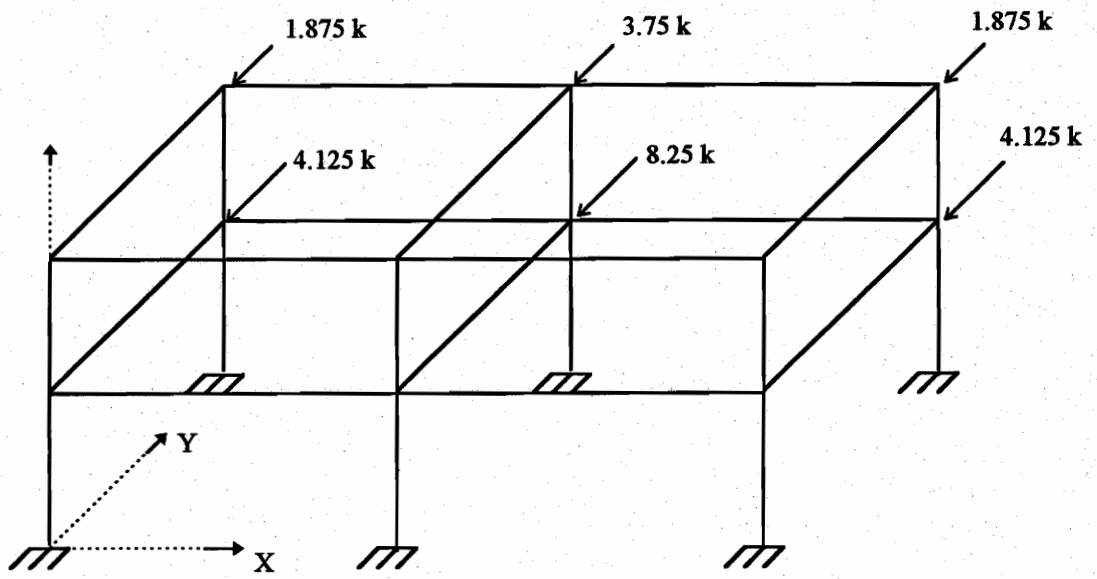


Fig. 5.8 Wind Loads (from Front) on Two Story Space Frame

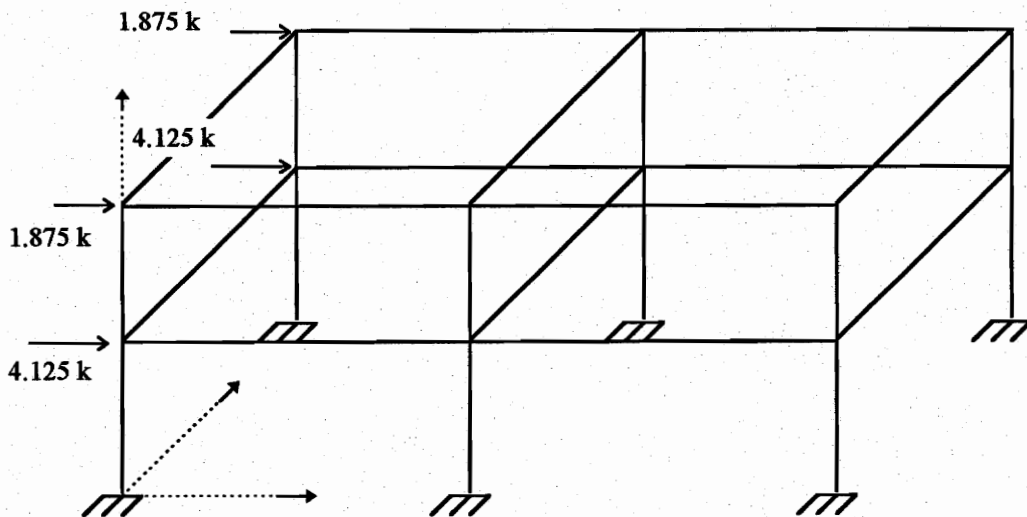


Fig. 5.9 Wind Loads (from Side) on Two Story Space Frame

5.4.2 Input and Output Files

Appendix B shows the input data file, and the output from the program.

5.4.3 Discussion of Design Results

The same structure was analyzed and designed using DAST. It is noticed that the DAST (Version 11.0) gives accurate results for the design of beams, as well as the columns in a plane frame. However, the design results for columns in space frames obtained from DAST are not reliable. It does provide accurate results for the code check for both beams and columns. It checks beams or columns by dividing the element into 10 sections and checking the required strengths (such as axial strength and flexural strength) at each section according to the LRFD criteria. It then determines the critical section for each element for the critical load combination. The code check feature of the DAST program was used for verifying column design results obtained by the program of this study.

5.4.3.1 Beam Design Results

Table 5.4 shows beam design results obtained from the program and from DAST. It is seen that design results for most beams are the same. The only beams which had different section sizes were beams 12, 14, 16. For these three beams, the difference in

beam sizes were not significant. One possible explanation for the difference could be the difference in analysis results.

5.4.3.2 Column Design Results

Table 5.5 shows the column design results obtained from the program. Table 5.6 shows the code check results obtained using DAST for the same column sizes as were designed by the program. From Table 5.6, it is seen that the unity results for the interaction functions $H1.2$ [LRFD 86] are close to 1 (varying from 0.88 to 0.98) indicating that the columns are well designed. The unity ratios for column No.1, 3, 5, 17, 19, 21 are slightly lower (around 0.88), but are still good enough for practical design. The reason for the lower ratio may be due to slight differences in analysis results between the CADKEY analysis program and DAST, or due to different procedures for determining the required flexural strength M_u based on second order analysis. The unity ratios for the other columns of the frame are greater than 0.9, which means the design results from this program are economical.

Table 5.4 Comparison of Beam Design Results for Space Frame

Beam Number	The Program	DAST
7	W 18x76	W 18x76
8	W 18x86	W 18x86
9	W 18x76	W 18x76
10	W 18x86	W 18x86
11	W 10x39	W 10x39
12	W 8x35	W 10x39
13	W 12x58	W 12x58
14	W 12x53	W 12x58
15	W 10x39	W 10x39
16	W 8x35	W 10x39
23	W 18x76	W 18x76
24	W 18x86	W 18x86
25	W 18x76	W 18x76
26	W 18x86	W 18x86

Table 5.5 Column Design Results

Column Number	Design Results
1	W10x49
2	W12x87
3	W12x65
4	W12x87
5	W10x49
6	W12x87
17	W14x48
18	W12x87
19	W12x65
20	W12x87
21	W14x48
22	W12x87

Table 5.6 Code Check by DAST for Column Design Results

Column Number	Design Results from The program	Unity Ratio of H1.2 (LRFD)	Critical Load Combination
1	W10x49	0.887	1.2D+0.5L+1.3W
2	W12x87	0.968	1.2D+1.6L
3	W12x65	0.896	1.2D+0.5L+1.3W
4	W12x87	0.972	1.2D+1.6L
5	W10x49	0.887	1.2D+0.5L+1.3W
6	W12x87	0.968	1.2D+1.6L
17	W14x48	0.885	1.2D+1.6L
18	W12x87	0.959	1.2D+1.6L
19	W12x65	0.882	1.2D+1.6L
20	W12x87	0.975	1.2D+1.6L
21	W14x48	0.885	1.2D+1.6L
22	W12x87	0.959	1.2D+1.6L

Chapter 6

Summary and Conclusions

The purpose of this study was to develop a program for the design of steel plane frames and space frames using object oriented programming techniques. This program reads analysis results from the output file of the CADKEY frame analysis program and designs beams and columns in the structure in accordance with the LRFD steel design specification [1986].

For beam design, the program calculates moments and shear forces at fifteen equally spaced sections along the beam under different load combination cases. The critical load combination and design moments are determined. The beam can be lateral braced or unbraced. The locations of the brace points can be specified for each beam. The lateral bracing for each beam is specified by the user. The beams are designed by finding the W section which satisfies both the strength requirement [LRFD 1986 Sec. F-1] and least weight requirement. Similar beams can be assigned to a mark group and designed with the same size.

The program designs columns in accordance with the LRFD steel design specification section H1-2. Both braced and unbraced frames can be designed. Effective length factors for each column are computed by solving the nomograph equation. The program has the capability for considering second order analysis using the procedure provided in the LRFD specification. Columns can also be assigned to column mark groups. The columns are designed to satisfy both the beam-column interaction equations [LRFD H1-2] and least weight requirement.

The program was developed using object oriented programming approach. Both the structural design and user interface were developed using object oriented techniques. The advantages of the object oriented approach are reflected in the program. The program is easy to understand and debug since objects in the program represents corresponding objects in a real structure. The classes for structural design aspects are well defined and organized using inheritance. The use of MFC [Microsoft Foundation Classes] library saved considerable time and effort in developing the graphical user interface.

This study illustrates the use of the object oriented programming for developing steel structural design applications. Further work may include: 1) Developing an integrated program in which both structural analysis and design are included. 2) Extending the program to include concrete design. 3) Considering other advanced and practical aspects of steel frame design such as "leaning" columns.

References

An-Nashif, H.N., "An Object Oriented Algorithm for Automated Modeling of Frame Structures: Stiffness Modeling", *Engineering with Computers*, Vol. 7, 1991, pp 121-128.

Adeli, H., *Interactive Microcomputer-Aided Structural Steel Design*, Prentice Hall, New Jersey, 1988.

Aouad, G.F. and Kirkham, J. A., "An Object-Oriented Model of Building Procedurement", *Microcomputers in Civil Engineering*, Vol.6, 1994, pp 401-416.

Biedermann, J.D. and Girerson, D.E. "An Object Oriented Approach to Detailed Structural Design", *Microcomputers in Civil Engineering*, Vol. 8, 1993, pp 225-231.

Budd, T., *An Introduction To Object Oriented Programming*, Addison-Wesley Publishing Co. Inc., New Jersey, 1991.

Cheney, W. and Kincaid, D., *Numerical Mathematics and Computing*, Third Edition, Brook/Cole Publishing Co., New York, 1994.

Crawley, S.W. and Dillon, R.M., *Steel Building Analysis and Design*, Prentice Hall, New Jersey, 1984.

DAST, *DAST Users Manual*, Version 11.0, Das Consulting, Inc., Boston, 1994.

Forde, B.W.R., "Object Oriented Finite Element Analysis", *Computers and Structures*, Vol. 34, No. 3, 1990, pp 355-374.

Gaylord, E.H.; Gaylord, C.N. and Stallmeyer, J.E., *Design of Steel Structure*, Third Edition, McGraw Hill, Inc., New York, 1992.

Jaiswal, S. S. and Riggd, H. R., "Design of An Object-Oriented Structural Analysis Program", *Proceedings of First International Offshore Polar Engineering Conference*, 1991.

Kulkarni, A. B., *Object Oriented Programming for Reinforced Concrete Design*, M.S. Thesis, Virginia Polytechnic and State University, 1993.

Lee, H.H.and Arora, J. S., "Object-Oriented Programming for Engineering Applications", *Engineering with Computers*, Vol. 7, 1991, pp225-235.

Lorenz, M., *Object Oriented Software Development: A Practical Guide*, Prentice Hall, New Jewsey, 1993.

AISC, *Load and Resistance Factor Design Manual of Steel Construction*, First Edition, AISC, 1986.

Microsoft, *Programmer's Guides*, Microsoft Visual C++ Development System for Windows, Version 1.0, Microsoft Corp., 1993.

Microsoft, *Microsoft Foundation Classes User's Guides*, Microsoft Visual C++ Development System for Windows, Version 1.0, Microsoft Corp., 1993.

Miller, G.R., "An Object Oriented Approach To Structural Analysis and Design", *Computers and Structures*, Vol. 40, No.1, 1991, pp75-82.

Miller, G.R., "Object-Oriented, Concurrent Structural Analysis", *Computing in Civil Engineering*, Proceedings of the Sixth Conference, September 11-13, Atlanta, 1989.

Oloufa, A., "An Object Oriented Approach to Integrated Design Information", *Microcomputers in Civil Engineering*, Vol. 9, 1994, pp 425-435.

Patel, J., *Post Processor for Design of Reinforced Concrete Space Frames Using Object-Oriented Programming*, M.S. Thesis, Virginia Polytechnic and State University, 1994.

Powell, G.H., Abdalla, G.A. and Sause, R., "Object Oriented Knowledge Representations: Cute Things and Caveats", *Computing in Civil Engineering*, Proceedings of the Sixth Conference, September 11-13, Atlanta, 1989.

Salmon, C. G. and Johnson, J. E., *Steel Structures Design and Behavior*, McGraw Hill, Inc., New York, 1990.

Young, M.J., *Mastering Microsoft Visual C++ Programming*, SYBEX Inc., 1993

Yu, G. and Adeli, H., "Computer-Aided Design Using Object-Oriented Programming Paradigm and Blackboard Architecture", *Microcomputers in Civil Engineering*, Vol. 6, 1991, pp177-190.

Appendix A

Class Definitions for the Program

In this Appendix, complete definitions of the classes used in this program are presented. Structural classes are described first. This is followed by the interface classes which include classes derived from the MFC application framework classes, such as the classes for creating dialog boxes.

A. Structural Classes:

1) Class **Structure** stores the description of the structure. It also contains the definition of the data structure **MembIncid** for storing member incidences.

```
// structur.h : Header file for Structure class
/////////////////////////////////////////////////////////////////

#ifndef _STRUCTURE_H_
#define _STRUCTURE_H_

struct MembIncid          // member incidences
{
    int MembNum;           // member number
    int INum;              // start joint number of the member
    int JNum;              // end joint number of the member
};

class Structure : public CObject
{
public:
    int strType;
    // structure type, plane frame=0, space frame=1

private:
    CString strName;      // structure name

    int frameType;
    // structure type, braced frame=0, unbraced frame=1, braced
    // in X direction=2, braced in Y direction=3

    int numJoints;        // total number of joints
    int numMembers;       // total number of members
}
```

```

    int numSupts;        // total number of supports
    int numPinSupts;     // total number of pinned supports
    int numFixSupts;     // total number of fixed supports

    int numLoadCase;     // total number of load cases,

    int numLoadComb;     // total number of load combinations

    float Width, Height, Length;
    // total width (Z dir.), height and length(X dir.) of
    //the structure

protected:
    DECLARE_SERIAL(Structure)

public:
    //constructors
    Structure ( );
    Structure ( CString Name, int SType, int Joints, int Memb,
                int Supts, int PinSupts, int FixSupts, int
                adcase, int LoadComb);

    // operations
    void SetStrData ( CString Name, int SType, int Joints,
                    int Memb, int Supts, int PinSupts,
                    int FixSupts, int Loadcase);

    void SetFrameType ( int FType);        // set frame type

    // set number of load combinations
    void SetLdComb (int LoadComb);

    // set total length, width and height of the struture
    void SetStrDim (float l, float wid, float hgt);

    CString GetStrName( ) { return strName; };
    int GetFrameType( )   { return frameType; };
    int GetNumJoints ( )  { return numJoints; };
    int GetNumMemb ( )    { return numMembers; };
    int GetNumSupts ( )   { return numSupts; };
    int GetNumLoadCase ( ) { return numLoadCase; };
    int GetNumLoadComb ( ) { return numLoadComb; };
    void GetSupts ( int pinned, int fixed );
    void GetStrDim (float &l, float &wid, float &hgt );

    //serialize the data to disk file
    virtual void Serialize(CArchive & ar);

};

#endif // _STRUCTURE_H_

```

2) Class Joint stores joint data.

```

// joint.h : interface of class "joint"
////////////////////////////////////

```

```

#ifndef _JOINT_H_
#define _JOINT_H_

struct displacement
// stores joint displacements for different load cases
{
    int LoadCaseNum ;           // load case number
    float dispX;                // displacement in X direction
    float dispY;                // displacement in Y direction
    float dispZ;                // displacement in Z direction
    float Rot_X;                // Rotation in X direction
    float Rot_Y;                // Rotation in Y direction
    float Rot_Z;                // Rotation in Z direction
};

class Joint : public CObject
{
private:
    int JtNum;
    float CoordX, CoordY, CoordZ; // joint coordinates
    int JointType;                // free=0, fixed=1, hinged=2
    int X_Restri;                // restraint in x direction
    int Y_Restri;                // restraint in y direction
    int R_Restri;                // rotational restraint
    int NumLoadCase;             // loadcase number

public:
    displacement *Disp;
    // store joint displacements under different load cases

protected:
    DECLARE_SERIAL(Joint)

public:
    //constructor
    Joint ( );
    Joint ( int jtNum, float x, float y, float z, int JType,
            int X_Res, int Y_Res, int R_Res);

    // operations
    void SetJtNum ( int jtNum );
    void SetCoord ( float x, float y , float z );
    void SetJointType ( int JType );
    void SetJointRes ( int X_Res, int Y_Res, int R_Res);

    // set displacement array
    void SetDispArray ( int NumCase);

    // set Disp object Array
    void SetDisp( int casenum, float dispx, float dispy,
                  float dispz, float rotx, float roty, float rotz );

    void GetCoord      ( float &x, float &y, float &z );
    int GetJointType( ) { return JointType;};
    void GetJointRes   ( int &X_Res, int &Y_Res, int &R_Res);

```

```

public:
    ~Joint() ;          //destructor

    //serialize the data to disk file
    virtual void Serialize ( CArchive & ar );

};

#endif // _JOINT_H_

```

3) Class LoadComb stores load combination data.

```

//loadcomb.h: interface of LoadComb class
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#ifndef ( _LOADCOMB_H_)
#define _LOADCOMB_H_

class LoadComb : public CObject
{
private:
    int i;                // number of load cases

    float *factor;        // load factors
    int *LoadCase;        // base load cases: live load, dead load

protected:
    DECLARE_SERIAL(LoadComb)

public:
    //constructors and destructor
    LoadComb( );
    LoadComb ( int ii, float fact[ ], int Case[ ] );
    ~LoadComb ( );
    void operator= ( const LoadComb &ldcomb );

    //operations
    void SetArrays ( int ii, float fact[ ], int Case[ ] );
    //set arrays for storing load factors and each load cases

    int GetNumCases ( ) {return i;};
    // get the number of cases in a combination

    void GetCombData ( float fact[ ], int Case[ ] );
    // get comb factors and load cases

    virtual void Serialize ( CArchive & ar );
    // serialize data to disk file
};

#endif // _LOADCOMB_H_

```

4) Class Member stores the description of the members. It is a base class which stores general information about a member. The Beam and Column classes are its derived classes, which store specific information about a beam or a column.

```
//member.h: interface of class "member"
////////////////////////////////////

#ifndef _MEMBER_H_
#define _MEMBER_H_

#include "structur.h"
#include "joint.h"
#include "loadcomb.h"
#include "membload.h"

// Data structure used for storing section properties of a W section
struct aiscw
{
    char dsn[3]; designation
    float dn;                // nominal depth
    float wgt;               // weight per foot
    char jshape[3];         // Jumbo shape
    float a;                // area
    float d;
    float tw;
    float bf;
    float tf;
    float xk;
    float btf;
    float fyp;
    float htw;
    float dtw;
    float fypppl;
    float fypppa;
    int x1;
    float x2;
    float rt;
    float daf;
    float ri;
    float ra;
    int nt;
    float ix;
    float sx;
    float rx;
    float iy;
    float sy;
    float ry;
    float zx;
    float zy;
    float xj;
    float cw;
    float wno;
    float sw;
    float qf;
    float qw;
    float ro;
    float h;
}
```

```

    } ;

// Data structure MembEndForce for storing member end forces for
// different load cases

struct MembEndForce
{
    int LoadCaseNum;           // base load case number

    int JointA, JointB;
    // two joint numbers of the member

    float AxialForceA;
    float AxialForceB;         // axial forces

    float ShearAX, ShearAY, ShearBX, ShearBY; // shear forces
    float MomAX, MomAY, MomBX, MomBY;         // end moments
};

////////////////////////////////////
// Member.h: interface of Member class , which is the base class
// for both Beam and Column class

enum{ beam,column};

class Member : public CObject
                // base class for Beam and Column class
{
    // attributes
public:
    int MembType;
    // member type : Beam=0 or Column=1

protected:
    float Length;           // member length

    float Fy, E;
    // steel strength and elasticity modulus of W-sections

    CString WShape;
    // choice of design sections entered by user

    Joint *JtA, *JtB;
    // two pointers to two end joint objects

    float Ix, Iy;
    // section property, first read from input file

    Structure *myStr;
    // pointer to structure object

    MembEndForce *MembEndF;
    //array of struct MembEndForce objects

    int NumCase;
    // total number of load cases

```

```

int DesignI;
// design result: designed W section's position in data file

float wdn;
// nominal depth of the designed W section

float weight;
// weight of designed W section

float a,d,sx,sy,rx,ry,bf,wgt;           // W section data

// implementation
protected:
    DECLARE_SERIAL(Member)

public:
    // constructors
    Member ( );
    Member ( Joint *JointA, Joint *JointB,   Structure *newStr );
    ~Member();                               // destructor

    //operations
    void SetWShape ( CString shape );
    // set W section restraints
    void SetEFy ( float MemE, float memFy );

    void SetLength ( float l );
    // set length according to member's joint coordinates

    void SetMemForceArray ( );
    // set MemEndForce object arrays to store member end forces
    // under different load cases

    void SetMemForce( int CaseNum, int JA, int JB ,
                      float ForceA, float ForceB, float ShAX,
                      float ShAY, float ShBX, float ShBY,
                      float MoAX, float MoAY, float MoBX,
                      float MoBY);

    // set moment of inertia of the member
    void SetPropty ( float Ixx, float Iyy);

    void GetPropty ( float &Ixx, float &Iyy );
    float GetLength ( ) { return Length;};
    void GetEFy ( float &e, float &fy ) ;

    // get the selected W section's data
    void GetResults ( float &a1, float &d1, float &sx1, float
                      &sy1, float &rx1, float &ry1, float &ix,
                      float &iy, float &wdn1, float &wgt1,
                      float &bf1);

    // serialize member data to disk file
    virtual void Serialize(CArchive & ar);
};

```

```

////////////////////////////////////
// interface of Beam class

class Beam : public Member
{
    // attributes
private:
    // number of sections for calculating moments and shears
    int numSection;

    // brace condition of the beam, 0 - lateral supported, 1 -
    // several brace points, 2 - no brace points except two ends
    int BraceCond;

    int numPtLd;        // total number of point loads on a beam
    int numUniLd;        // total number of uniform loads
    int numLinLd;        // total number of linear loads

    // array to store shear at each section on the beam for a
    // load case
    float *Shear;

    // arrays to store moment at each section on the beam for a
    // load case
    float *Moment;

    // two dimensional arrays for storing total moments and
    // shears for different load combinations
    float **Mom_Tot;
    float **Shear_Tot;

    // attributes about brace points
private:
    // unbraced lengths between brace points
    float *Lb;

    // total number of internal brace points
    int numBracePts;

    float *BracePt;        // array of brace positions

    // moments at brace points for design
    float *BraceMom;

    // attributes for design results
    float DesignMom;        // maximum moment for design
    float DesignSh;        // maximum shear for design

    // location of maximum moment on the beam

```

```

    int MaxJ;

    int DesignComb;          // critical load combination

public:
    aiscw wout;

//implementation
protected:
    DECLARE_SERIAL(Beam)

public:
    // constructors and destructor
    Beam ( );
    Beam ( int numSect, Joint *JointA, Joint *JointB,
           Structure *newStr);
    ~Beam ( );

    // operations

    // set the number of sections
    void SetNumSect( int numSect );

    // set the number of point loads
    void SetNumPtLd( int numPLd );

    // set the number of uniform loads
    void SetNumUniLd( int numULd );

    // set Moment[ ], Shear[ ] arrays
    void SetArrays ( );

    // set Shear_Tot[ ][ ], Mom_Tot[ ][ ] arrays
    void SetDoubleArrays ( );

    // set brace condition
    void SetBraceCond ( int BCond );    // set brace condition

    // set number of internal brace points
    void SetNumBracePt ( int numBracePt );

    void SetBracePt ( float Braces[ ] );
    // set brace positions
    void SetUnbracedL ( );
    // set unbraced length

    // calculate moments and shears for a single case

    // compute moments due to left end moment
    void Cal_LtMom ( float LtMom );

    // compute moments due to right end moment
    void Cal_RtMom ( float RtMom );

```

```

// compute moments due to point load
void Cal_PtLd ( float p, float a );

// compute moments and shears due to full or partial uniform
// load
void Cal_UniLd ( float w, float a=0, float b=0 );

// analyse the beam

// superimpose factored moments, shears to corresponding
// positions in Mom_Tot, Shear_Tot arrays
void Impose( float Fact, int CombNum);

// get load combination cases for point loads, uniform loads
// and linear loads. Calculate moments and shears under
// single load case and superimpose them. Store them in two-
// dimension arrays.
void Analyse ( LoadComb LdComb[ ], ConcLoad *PLd[ ],
              UniLoad *UniLd[ ], LinearLoad *LiLd[ ] );

// find maximum moment and shear. Find critical combination
// case as design case and put its moments and shears in
// Moment[ ], shear[ ] arrays. Delete the double dimension
// arrays
void DesignForce ( );

// calculate moment at brace points
void SetBraceMom ( );

// design the beam

// compute Cb between brace points
void ComputeCb ( float Cb[ ] );

// compute the nominal strength of a trial W section
float ComputeMn ( float Lb, float Cb, float bf_2tf, float
                  hc_tw, int X1, float X2, float Sx, float
                  Zx, float Sy, float Zy, float ry );

// open "wdata.dat" data file to get W section
void OpenDataFile ( int i );

// design beam according to brace conditions and open data
// file to find required W sections. Check strength of trial
// sections.
void Design ( );

```

```

void CheckShear(float A);           //check shear strength

// return the depth and weight of designed W section
void GetDesignSect( float &dn, float &wgt);

// get design moments and shears
void GetForces ( float mom[ ], float sh[ ] );

int GetNumSect ( ) { return numSection;};
int GetBraceCond ( ) { return BraceCond;};
int GetNumBrace ( ) { return numBracePts;};
void GetBracepts ( float brace[ ] );
int GetDesignComb ( ) {return DesignComb; };

//serialize beam data to disk file
virtual void Serialize(CArchive & ar);

};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//interface of column class

class Column : public Member
{
    // attributes
private:
    float Ky, Kx;           // effective length factors
    float Lx, Ly;           // unbraced length in x, y directions

    // factors for computing required bending moment in both
    // strong X axis and weak Y axis
    float B1X, B1Y, B2X, B2Y;

    // member end moments under different load combinations
    // about strong axis
    float *MxA, *MxB;

    float *MyA, *MyB;           // about weak axis

    // member end Axial Forces under different load combinations
    float *Pu;

    // cross section's position, Beta =0 weak axis in plane
    // Beta =90.0 weak axis off the plane
    float Beta;

    CString CaseNm[6];           // store load case name

    // factored axial force under comb case containing side
    // wind load

```

```

float WindPuX;

// factored axial force under comb case containing front
// wind load
float WindPuY;

// critical comb. case for column design
int DesignComb;

// Factored member end forces corresponding to the critical
// combination case
float Mx1, Mx2, My1, My2, P;

float Mnxx, Mnxy, Pnn;          // nominal strengths

public:
    aiscw wout;

//implementation
protected:

    DECLARE_SERIAL(Column)

public:
    //constructors
    Column ( );                  // default constructors
    Column ( Joint *JointA, Joint *JointB, Structure *newStr );
    ~Column ( );                 // destructors

    //operations

    void SetKxLx ( float kx );   //set Kx, Lx, Lx = Length
    void SetKyLy ( float ky );   //set Ky, Ly, Ly = Length

    // set MxA[ ],MxB[ ],MyA[ ], MyB[ ],Pu[ ] arrays under
    // different load combination cases
    void SetCombEndForce ( LoadComb LdComb[ ] );

    void SetCaseName (char casenm[6][25] );
    void SetB2 (float b2x, b2y ); //set B2

    // calculate axial strength of the column
    float ComputePn(float ag, float rx, float ry);

    // calculate flexural strength of the column about strong
    // axis
    float ComputeMn ( float m1, float m2, float bf_2tf, float
                      hc_tw, int X1, float X2, float Sx, float
                      Zx, float Sy, float Zy, float ry );

```

```

// calculate flexural strength of the column about weak axis
float ComputeMny ( float bf_2tf, float Zy );

// calculate Euler Load
float CalPe(float a, float rx, float ry);

void GetDesignSect(float &dn, float &wgt);

// open "wdata.dat" data file to get W section
void OpenDataFile( int i );

// design the column according to interaction equations and
// least weight requirement
void Design ( LoadComb LdComb[ ] );

// get critical combination case
int GetDesignComb ( ) { return DesignComb;};

// return Kx, Lx, Ky, Ly
void GetKL ( float &kx, float &ky, float &lx, float &ly );

// return design end forces
void GetDesignForce ( float &mx1, float &mx2, float &p );

// return nominal strength
void GetStrength ( float &mnx , float &pn );

// return beta values
float GetBeta ( ) { return Beta; }

// get axial force under comb case containing wind load
// cases
void GetWindPu ( float &pux, float &puy );

// serialize data to disk file
virtual void Serialize ( CArchive & ar );

};

#endif //__MEMBER_H_

```

5) Load classes include classes for joint load and member loads. Classes for member loads consist of the base class `MembLoad`, and the derived classes `ConcLoad`, `UniLoad` and `LinearLoad`.

```

// jointld.h: header file for JointLoad class
////////////////////////////////////

#ifndef _JOINTLD_H_
#define _JOINTLD_H_

class JointLoad : public CObject

```

```

{
    private:
        int LoadCaseNum;        // base load case number
        int JointNum;            // joint number
        int dir;                 // direction of load, xdir =0, ydir=1
        int forceType;           // force=0, moment=1
        float JtLoad;            // joint load
        float JtMomLd;           // joint moment load

    protected:
        DECLARE_SERIAL( JointLoad )

    public:
        // constructor
        JointLoad ( );
        JointLoad ( int CaseNum, int JtNum, int direct, int f, float
                    Load, float MomLd );
        ~JointLoad ( ) {};        //destructor

        //operations
        void SetJointLoad ( int CaseNum, int JtNum, int direct,
                           int f, float Load, float MomLd);

        int GetJointNum ( )      { return JointNum;};
        int GetJointDir ( )      { return dir;};
        int GetCaseNum ( )       { return LoadCaseNum; };
        float GetJointLoad ( )   { return JtLoad;};
        float GetJtMomLoad ( )   { return JtMomLd;};

    public:

        // serialize data to disk file
        virtual void Serialize(CArchive& ar);

};

#endif // _JOINTLD_H_

```

// membload.h: header file for MembLoad classes

```

////////////////////////////////////
#ifndef MEMBLOAD_H_
#define MEMBLOAD_H_

class MembLoad : public CObject
{
    protected:

        int LoadCaseNum; // load case number
        int MembNum;      // member number
        int LoadType;     // concload=0, uniload=1, linearload =2

    protected:

        // for serialization only
        DECLARE_SERIAL( MembLoad )

```

```

public:
    // constructor and destructor
    MembLoad ( );
    MembLoad ( int CaseNum, int MemNum, int LType );
    ~MembLoad ( ) { };

    // operations
    void SetMembLoad ( int CaseNum, int MemNum, int LType );
    int GetCaseNum ( ) ;
    int GetLoadType ( ) { return LoadType; };
    int GetMembNum ( ) { return MembNum; };

    // serialize data to disk file
    virtual void Serialize ( CArchive & ar );

};

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class ConcLoad : public MembLoad
{
private:
    float PtLoad;        // concentrated load
    float PDist;         // distance from left end of the beam

protected:
    // for serialization only
    DECLARE_SERIAL(ConcLoad)

public:
    // constructor and destructor
    ConcLoad ( );
    ConcLoad ( float PLoad, float Dist, int CaseNum,
               int MemNum, int LType);
    ~ConcLoad ( ) { };

    // operations
    void SetConcLoad ( float PLoad, float Dist );
    void GetConcLoad ( float &PPLoad, float &DDist );

    // serialize data to disk file
    virtual void Serialize ( CArchive & ar );

};

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class UniLoad : public MembLoad
{
private:

```

```

        float WLoad;           // intensity of uniform load
        float aPos;           // start point of uniform load
        float bPos;           // end point of uniform load

protected:
    // for serialization only
    DECLARE_SERIAL(UniLoad)

public:
    // constructor and destructor
    UniLoad ( );

    // constructor for uniform load over entire length of the
    // beam
    UniLoad ( float w, int CaseNum, int MemNum, int LType );

    // constructor for partially uniform load
    UniLoad ( float w, float a, float b, int CaseNum,
              int MemNum, int LType);

    ~UniLoad ( ) {};

    // operations
    void SetUniLoad ( float w, float a, float b );

    float GetUniLoad ( ) { return WLoad;};
    void GetPosition ( float &a, float &b );
    // serialize data to disk file
    virtual void Serialize( CArchive & ar);

};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class LinearLoad : public MembLoad
{
private:
    float TriLoad1;           // intensity of load at start
    float TriLoad2;           // intensity of load at end
    float TDist1;             // start point of linear load
    float TDist2;             // end point of linear load

protected:
    DECLARE_SERIAL ( LinearLoad )
    // for serilization only

public:
    // constructor and destructor
    LinearLoad ( );
    LinearLoad ( float TLoad1, float TLoad2, float Dist1, float
                Dist2, int CaseNum, int MemNum, int LType);
    ~LinearLoad ( ) { };

    // operations

```

```

        void SetLinearLoad ( float TLoad1, float TLoad2, float
                               Dist1, float Dist2 );
        void GetLinearLoad ( float &TLoad1, float &TLoad2, float
                               &Dist1, float &Dist2 );

        virtual void Serialize ( CArchive & ar );
        // serialize data to disk file
};

#endif          //_MEMBLOAD_H_

```

B. Interface Classes

Interface classes include MFC application framework classes and classes for creating and managing dialog boxes.

1) CMainFrame class. Class for the main application window.

```

// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////
//
class CMainFrame : public CMDIFrameWnd
{
    DECLARE_DYNAMIC(CMainFrame)

public:
    CMainFrame();

// Attributes
public:

// Operations
public:

// Implementation
public:
    virtual ~CMainFrame( );
#ifdef _DEBUG
    virtual void AssertValid( ) const;

```

```

        virtual void Dump ( CDumpContext& dc)    const;
#endif

protected:    // control bar embedded members
               CStatusBar  m_wndStatusBar;
               CToolBar   m_wndToolBar;

// Generated message map functions
protected:
   //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

    DECLARE_MESSAGE_MAP()
};

```

2) CSteelfmApp class. This is the application class.

```

// steelfm.h : main header file for the STEELFM application
//
/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"    // main symbols

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
// CSteelfmApp:
// See steelfm.cpp for the implementation of this class
//

class CSteelfmApp : public CWinApp
{
public:
    CSteelfmApp();

// Overrides
    virtual BOOL InitInstance ( );

// Implementation

   //{{AFX_MSG(CSteelfmApp)
    afx_msg void OnAppAbout();

    DECLARE_MESSAGE_MAP ( )
};

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////

```

3) CSteelfmDoc class. This class stores, maintains, and processes all data in the program.

```

// steeldoc.h : interface of the CSteelfmDoc class
//
/////////////////////////////////////////////////////////////////
////
#include "member.h"
#include "jointld.h"

const int NUMLINES = 60;
// number of lines stored in document

const int MARGIN=15;          // margin for displaying information

class CSteelfmDoc : public CDocument
{
protected:                    // create for serialization only
    CSteelfmDoc ( );
    DECLARE_DYNCREATE ( CSteelfmDoc )

// Attributes
public:
    CString AFileName;        // analysis file name
    CString strName;          // structure description
    int strType;              // structure type, 0-plane frame, 1-space frame
    int frameType;            // frame type

    // modulus of elasticity and yield strength of steel
    float E, Fy;

    // pointers for storing beam numbers and column numbers
    int *BmIndex, *ColIndex;

    int NumBeam, NumCol;      // total number of beams or columns
    int NumMemb;              // total number of members
    int NumLdComb;            // number of load combination cases
    char CaseName [ 6 ][ 25 ]; // load case name
    int NumLoadCase;          // number of load cases

    //string array to store the lines shown in the view window
    CString mLineTable[NUMLINES];

    // check current ID for showing document data or graphics
    UINT CurrentTool;

    // choice for displaying results or displaying graphics
    int DrawFlag;

protected:

    // data for load combinations
    BOOL comb1,comb2, comb3, comb4,comb5, comb6;

    // counter for loadcomb dialog box

```

```

int count;

int BmNum;           // currently selected beam number
int ColNum;          // currently selected column number

// data for setting beam mark groups
int BeamGroup[10][30]; // set beam mark groups
CString BeamList[10];  // strings for storing beam list in beam
                        // mark group dialog box
int NumGroup;          // number of beam groups selected
int ctmk;              // count times of opening beam dlg box
int NumList[10];       // number of members in each group

// data for setting column mark groups
int ColGroup[10][30]; // set column mark groups
CString ColList[10];  // strings for storing column list in
                        // column mark group dialog box
int NumColGroup;      // number of column mark groups
int NumColList[10];   // number of columns in each group

protected:

Structure theStr;      // object of the structure class

// objects of CObArray class for storing object arrays of member
// objects and joint objects
CObArray MembArray, JtArray;

// objects of CObArray class for storing object arrays of member
// uniformload, point load, linear load and joint load objects
CObArray ULdArray, CLdArray, LiLdArray, JtLdArray;

// object array for storing objects of member incidences
MembIncid *MIncid;

// object array for storing loadcomb objects
LoadComb *LdComb;

public:

// Operations

void ReadAnlFile ( );           // read analysis file

// calculate effective length factor kx and ky for columns
void Cal_K ( );

// calculate B2 factor for sidesway consideration in x and y dir.
void Cal_B2();

// initialize data in document
void InitDocument ( );

// delete dynamically created objects and arrays

```

```

void DeleteContents ( );

void SetBmGroup (int CriticMem[ ] );
// determine critical member in the beam group
// determine critical member in the column group
void SetColGroup ( int CritiMem[ ] );

// create output strings for showing results
void BeamOutput ( int MembNum );

// create output strings for showing results
void ColOutput (int MembNum );

public:

    void Draw(CDC *pDC);                // draw graphics

// Implementation

public:

    virtual ~CSteelfmDoc( );
    virtual void Serialize(CArchive& ar);

    // overridden for document i/o
#ifdef DEBUG
    virtual void AssertValid( ) const;
    virtual void Dump(CDumpContext& dc) const;
    CMemoryState  oldState, newState,diffState;
#endif

protected:

    virtual BOOL OnNewDocument ( );
    virtual BOOL OnOpenDocument(const char* pszPathName);
    virtual void OnCloseDocument ( );

// Generated message map functions
protected:
   //{{AFX_MSG(CSteelfmDoc)
    afx_msg void OnBeampara ( );
    afx_msg void OnColumnpara ( );
    afx_msg void OnStrfeature ( );
    afx_msg void OnDesignbm ( );
    afx_msg void OnLdcombcase ( );
    afx_msg void OnDesigncol ( );
    afx_msg void OnShowbeam ( );
    afx_msg void OnShowcol ( );
    afx_msg void OnUpdateShowbeam (CCmdUI* pCmdUI);
    afx_msg void OnUpdateShowcol (CCmdUI* pCmdUI);
    afx_msg void OnDisplay ( );
    afx_msg void OnNext ( );
    afx_msg void OnPrev ( );
    afx_msg void OnBeamark ( );
    afx_msg void OnColmark ( );
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP( )

```

```
};
```

```
////////////////////////////////////  
////
```

4) CSteelfmView class. This class is responsible for displaying document data in the view window.

```
// steelvw.h : interface of the CSteelfmView class
```

```
//
```

```
////////////////////////////////////  
////
```

```
class CSteelfmView : public CScrollView  
{
```

```
protected:          // created for serialization only  
    CSteelfmView();  
    DECLARE_DYNCREATE(CSteelfmView)
```

```
// Attributes
```

```
public:  
    CSteelfmDoc* GetDocument();
```

```
// Operations
```

```
protected:  
    virtual void OnUpdate (CView* pSender, LPARAM lHint, CObject*  
                           pHint);  
    virtual void OnInitialUpdate ( );
```

```
// Implementation
```

```
public:
```

```
    virtual ~CSteelfmView ( );  
    virtual void OnDraw ( CDC* pDC );  
    // overridden to draw this view  
    virtual void OnPrint(CDC *pDC, CPrintInfo * pInfo);
```

```
#ifdef _DEBUG
```

```
    virtual void AssertValid ( ) const;  
    virtual void Dump (CDumpContext& dc) const;
```

```
#endif
```

```
protected:
```

```
    // Printing support  
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);  
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);  
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
```

```
// Generated message map functions
```

```
protected:  
    //{AFX_MSG(CSteelfmView)  
    afx_msg void OnOpenAnlfile();  
    //}AFX_MSG  
    DECLARE_MESSAGE_MAP()
```

```
};
```

```

#ifndef _DEBUG // debug version in steelvw.cpp
inline CSteelfmDoc* CSteelfmView::GetDocument( )
    { return (CSteelfmDoc*)m_pDocument; }
#endif

```

```

////////////////////////////////////
////

```

5) Dialog classes for creating dialog boxes.

5.1) BeamMark class manages the beam mark group dialog box.

```

// beammark.h : header file
//
////////////////////////////////////
////
// beamMark dialog

class BeamMark : public CDialog
{
// Construction
public:
    beamMark(CWnd* pParent = NULL);    // constructor

    // Dialog Data
    //{AFX_DATA(BeamMark)
    enum { IDD = IDD_BEAMGROUP };
    CEdit m_MemStr;
    CComboBox m_MemCombList;
    //}AFX_DATA

    int *BeamInx;           // beam index
    int NumBeam;           // number of beams
    int BmNum;             // beam number chosen
    CString m_GroupList[10]; // list of members in mark group
    int GroupNum;          // group number

    int ind;
    int numgroup;          // total number of mark groups

    // total number of beams in each mark group
    int NumList[10];

// Implementation
public:
    void SetCombList(int bmInx[]);
    void InitMembList();

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    // Generated message map functions
    //{AFX_MSG(BeamMark)

```

```

    afx_msg void OnClickedAddnum();
    afx_msg void OnClickedDelnum();
    afx_msg void OnClickedNext();
    afx_msg void OnClickedPrevious();
    virtual void OnOK();
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.2) BmNumDlg class manages the dialog box for the selection of the beam for which results are to be displayed.

```

// bnumdlg.h : header file
//

/////////////////////////////////////////////////////////////////
/////
// BmNumDlg dialog

class BmNumDlg : public CDialog
{
// Construction
public:
    BmNumDlg(CWnd* pParent = NULL);    // constructor

// Dialog Data
    //{{AFX_DATA(BmNumDlg)
    enum { IDD = IDD_BEAMNUM };
    CComboBox m_BeamList;
    //}}AFX_DATA

    int *BeamInx;           // beam index
    int NumBeam;            // number of beams
    int BmNum;              // beam number chosen

// Implementation
public:
    void SetCombList ( int bmInx[ ] );
    void InitMembList ( );

protected:
    virtual void DoDataExchange(CDataExchange* pDX);
                                                // DDX/DDV support

    // Generated message map functions
    //{{AFX_MSG(BmNumDlg)
    afx_msg void OnSelchangeMember();
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.3) BmParaDlg class manages the beam parameters dialog box.

```

// bmpara.h : header file
//

```

```

/////////////////////////////////////////////////////////////////
/////
// BMPARADLG dialog

class BMPARADLG : public CDialog
{
// Construction
public:
    BMPARADLG(CWnd* pParent = NULL);    // constructor
    ~BMPARADLG();                      // destructor

// Dialog Data
    //{AFX_DATA(BMPARADLG)
    enum { IDD = IDD_BEAMPARA };
    CComboBox    m_MembComb;
    float    m_E;
    float    m_Fy;
    int    BraceCond;
    int    NumBrace;
    CString    WShape;
    //}AFX_DATA
    float *Bracepts;    // positions of brace points
    int *BeamInx;    // beam index array
    int NumBeam;    // number of beams
    int BmNum;    // beam number chosen

// Implementation
public:
    void SetCombData (int bmInx[ ] );
    void InitMembList ( );

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    // Generated message map functions
    //{AFX_MSG(BMPARADLG)
    afx_msg void OnChangeNumbrace();
    afx_msg void OnBracepts();
    virtual BOOL OnInitDialog();
    afx_msg void OnSelchangeMember();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.4) BracePos class manages the brace position dialog box for entering position of lateral bracing.

```

// bracepos.h : header file
//
/////////////////////////////////////////////////////////////////
/////
// Bracepos dialog

```

```

class Bracepos : public CDialog
{
// Construction
public:

    int    m_BraceNum;

    Bracepos(CWnd* pParent = NULL);    // constructor

// Dialog Data
//{{AFX_DATA(Bracepos)
enum { IDD = IDD_BRACEPOS };
float m_Dist;

//}}AFX_DATA

// Implementation
protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    // Generated message map functions
//{{AFX_MSG(Bracepos)
virtual BOOL OnInitDialog();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

```

5.5) ColMkDlg class manages the column mark group dialog box for entering column mark groups.

```

// colmkdlg.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////
// ColMkDlg dialog

class ColMkDlg : public CDialog
{
// Construction
public:
    ColMkDlg(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
//{{AFX_DATA(ColMkDlg)
enum { IDD = IDD_COLMARK };
CComboBox    m_MemComb;
CEdit m_ColList;
//}}AFX_DATA
int *ColInx;                // column index
int NumCol;                 // number of columns
int ColNum;                 // column number chosen
CString m_GroupList[10];    // list of the members

```

```

        int GroupNum;                // group number

        int ind;
        int numgroup;                // total number of mark groups
        int NumList[10];
        // total number of columns in each group

// Implementation
public:
    void SetCombList(int colinx [ ] );
    void InitMembList ( );

protected:
    virtual void DoDataExchange(CDataExchange* pDX);
                                                // DDX/DDV support

    // Generated message map functions
   //{{AFX_MSG(ColMkDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnClickedAddnum();
    afx_msg void OnClickedDelnum();
    afx_msg void OnClickedNext();
    afx_msg void OnClickedPrevious();
    virtual void OnOK();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.6) ColNumDlg class manages the dialog box for selecting the column for which results are displayed.

```

// colnumdlg.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// ColNumDlg dialog

class ColNumDlg : public CDialog
{
// Construction
public:
    ColNumDlg(CWnd* pParent = NULL);    // constructor

// Dialog Data
    {{{AFX_DATA(ColNumDlg)
    enum { IDD = IDD_COLNUM };
    CComboBox    m_ColList;
    //}}AFX_DATA
    int *ColInx;                // column index
    int NumCol;                 // number of columns
    int ColNum;                 // column number chosen

// Implementation
public:
    void InitMembList();

```

```

        void SetCombList ( int cInx[ ] );

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    // Generated message map functions
   //{{AFX_MSG(ColNumDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSelchangeMember();
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.7) ColParaDlg class manages the dialog box for entering column data.

```

// colpadlg.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////
// Colparadlg dialog

class Colparadlg : public CDialog
{
// Construction
public:
    Colparadlg(CWnd* pParent = NULL);    // constructor
    ~Colparadlg();

// Dialog Data
    {{{AFX_DATA(Colparadlg)
    enum { IDD = IDD_COLPARA };
    CComboBox    m_Memlist;
    float m_E;
    float m_Fy;
    CString      m_WShape;
    float m_Beta;
    }}}AFX_DATA
    int *ColInx;
    int NumCol;           // number of columns
    int ColNum;           // column chosen

public:

    void InitMemblList();
    void SetCombList(int cInx[]);

// Implementation
protected:
    virtual void DoDataExchange(CDataExchange* pDX);
                                                // DDX/DDV support

    // Generated message map functions
    {{{AFX_MSG(Colparadlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSelchangeMember();
    }}}AFX_MSG

```

```

        DECLARE_MESSAGE_MAP()
};

```

5.8) CCombLdDlg class manages the load combination dialog box.

```

// combdlg.h : header file
//

/////////////////////////////////////////////////////////////////
/////
// CCombLdDlg dialog

class CCombLdDlg : public CDialog
{
// Construction
public:
    CCombLdDlg(CWnd* pParent = NULL);    // standard constructor
    ~CCombLdDlg ( );

// Dialog Data
    //{AFX_DATA(CCombLdDlg)
    enum { IDD = IDD_LOADCOMB };
    CListBox      m_CaseList;
    BOOL  m_Comb1;
    BOOL  m_Comb2;
    BOOL  m_Comb3;
    BOOL  m_Comb4;
    BOOL  m_Comb5;
    //}AFX_DATA
    CString CaseName[6];    // load case names
    int NumLoadCase;        // number of load cases shown in list box
    int ct;                 // count number
    int NumLoadComb;        // number of load combination cases chosen

// Implementation
public:
    void InitCaseList();
    void SetCombCase();

protected:
    virtual void DoDataExchange(CDataExchange* pDX);
                                    // DDX/DDV support

// Generated message map functions
    //{AFX_MSG(CCombLdDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnClickedComb1();
    afx_msg void OnClickedComb2();
    afx_msg void OnClickedComb3();
    afx_msg void OnClickedComb4();
    afx_msg void OnClickedComb5();
    virtual void OnOK();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

5.9) StructDlg class manages the structure data dialog box:

```
// structdlg.h : header file
//

/////////////////////////////////////////////////////////////////
/////
// Structdlg dialog

class Structdlg : public CDialog
{
// Construction
public:
    Structdlg(CWnd* pParent = NULL);    // constructor

// Dialog Data
    //{AFX_DATA(Structdlg)
    enum { IDD = IDD_STRUCTURE };
    int     m_StrType;           //structure type
    CString m_StrName;          //structure name
    //}AFX_DATA

// Implementation
protected:
    virtual void DoDataExchange(CDataExchange* pDX);

// Generated message map functions
    //{AFX_MSG(Structdlg)

    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

Appendix B

Input and Output Files for Test Problems

In this Appendix, the input and output files for the three test problems discussed in Chapter 5 are provided. The first test structure was a continuous beam while the second test problem was a two story plane frame. The third test structure was a two story space frame. The input and output files for the continuous beam are presented first, followed by those for the two story plane frame, and the two story space frame.

B.1 Input and Output Files for Continuous Beam

B.1.1 Input File for Continuous Beam

```
STRUCTURE Continuous Beam - Test Problem 1
TYPE CONTINUOUS BEAM
METHOD STIFFNESS
TABULATE ALL
NUMBER OF JOINTS          5
NUMBER OF MEMBERS         4
NUMBER OF SUPPORTS       5
NUMBER OF LOADINGS        2
NUMBER OF LOAD COMBINATIONS 1
JOINT COORDINATES
  1   0.000   0.000 S
  2 282.000   0.000 S
  3 582.000   0.000 S
  4 882.000   0.000 S
  5 1164.000  0.000 S
MEMBER INCIDENCES
  1   1   2
  2   2   3
  3   3   4
  4   4   5
MEMBER PROPERTIES PRISMATIC
  1 AX IZ 30.6 3100.00
  2 AX IZ 30.6 3100.00
  3 AX IZ 30.6 3100.00
  4 AX IZ 30.6 3100.00
CONSTANTS E 29000 ALL
JOINT RELEASES
  1 MOMENT Z
  2 MOMENT Z
  3 MOMENT Z
  4 MOMENT Z
```

5 MOMENT Z
LOADING 1 - dead load
MEMBER LOADS
1 FORCE Y UNIFORM -0.21
2 FORCE Y UNIFORM -0.21
3 FORCE Y UNIFORM -0.21
4 FORCE Y UNIFORM -0.21
LOADING 2 - live load
MEMBER LOADS
1 FORCE Y UNIFORM -0.25
2 FORCE Y UNIFORM -0.25
3 FORCE Y UNIFORM -0.25
4 FORCE Y UNIFORM -0.25
1 FORCE Y CONCENTRATED -20. 141.
2 FORCE Y CONCENTRATED -20. 150.
3 FORCE Y CONCENTRATED -20. 150.
4 FORCE Y CONCENTRATED -20. 141.
SOLVE

STRUCTURE Continuous Beam - Test Problem 1

LOADING 1 - dead load

MEMBER FORCES

MEMBER	JOINT	AXIAL FORCE	SHEAR FORCE	MOMENT
1	1	0.000	23.014	0.00
1	2	0.000	36.206	-1860.06
2	2	0.000	32.925	1860.06
2	3	0.000	30.075	-1432.47
3	3	0.000	30.075	1432.47
3	4	0.000	32.925	-1860.06
4	4	0.000	36.206	1860.06
4	5	0.000	23.014	0.00

SUPPORT JOINT DISPLACEMENTS

JOINT	X-DISPLACEMENT	Y-DISPLACEMENT	ROTATION
1	0.0000E+00	0.0000E+00	-0.7950E-05
2	0.0000E+00	0.0000E+00	0.1562E-05
3	0.0000E+00	0.0000E+00	0.8704E-20
4	0.0000E+00	0.0000E+00	-0.1562E-05
5	0.0000E+00	0.0000E+00	0.7950E-05

STRUCTURE Continuous Beam - Test Problem 1

LOADING 2 - live load

MEMBER FORCES

MEMBER	JOINT	AXIAL FORCE	SHEAR FORCE	MOMENT
1	1	0.000	34.132	0.00
1	2	0.000	56.368	-3135.39
2	2	0.000	50.052	3135.39
2	3	0.000	44.948	-2369.80
3	3	0.000	44.948	2369.80

3	4	0.000	50.052	-3135.39
4	4	0.000	56.368	3135.39
4	5	0.000	34.132	0.00

SUPPORT JOINT DISPLACEMENTS

JOINT	X-DISPLACEMENT	Y-DISPLACEMENT	ROTATION
1	0.0000E+00	0.0000E+00	-0.1356E-04
2	0.0000E+00	0.0000E+00	0.2797E-05
3	0.0000E+00	0.0000E+00	0.1138E-19
4	0.0000E+00	0.0000E+00	-0.2797E-05
5	0.0000E+00	0.0000E+00	0.1356E-04

B1.2 Output File for Continuous Beam

Member # 1	Type = Beam	Length = 23.50 ft	
=====			
Design Results:	W24x104	E = 29000.00 ksi	
Total weight of beam =	2.44 kips	Fy = 36.00 ksi	
Nominal strength:			
Moment:	0.9*Mn = 643.67 kip.ft	Shear: 0.9*Vn = 594.86 kips	
=====			
Section Properties:	W24x104		
area= 30.60 in^2	Ix= 3100.0 in^4	Iy= 259.0 in^4	
d= 24.060 in	rx= 10.100 in	ry= 2.910 in	
bf= 12.750 in	sx= 258.00 in^3	sy= 40.70 in^3	
=====			
Brace Condition :	No Lateral Supports Between Two Ends		
=====			
Load Combination Case For Design:	1.2*D+1.6*L		
=====			
Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	0.0	85.5
1	1.6	124.0	72.8
2	3.1	228.2	60.2
3	4.7	312.7	47.6
4	6.3	377.3	35.0
5	7.8	422.2	22.3
6	9.4	447.3	9.7
7	11.0	452.7	-2.9
8	12.5	412.4	48.5
9	14.1	326.5	-61.1
10	15.7	220.9	-73.7
11	17.2	95.5	-86.4
12	18.8	-49.8	-99.0
13	20.4	-214.7	-111.6
14	21.9	-399.5	-124.2
15	23.5	-604.1	-136.9

Member # 2	Type = Beam	Length = 25.00 ft	
=====			
Design Results:	W24x104	E = 29000.00 ksi	
Total weight of beam =	2.60 kips	Fy = 36.00 ksi	
Nominal strength :			
Moment: 0.9*Mn=	625.65 kip.ft	Shear: 0.9*Vn = 594.86 kips	
=====			
Section Properties:	W24x104		
area= 30.60 in^2	Ix= 3100.0 in^4	Iy= 259.0 in^4	
d= 24.060 in	rx= 10.100 in	ry= 2.910 in	
bf= 12.750 in	sx= 258.00 in^3	sy= 40.70 in^3	
=====			
Brace Condition :	No Lateral Supports Between Two Ends		
=====			
Load Combination Case For Design:	1.2*D+1.6*L		
=====			
Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-604.1	123.0
1	1.6	-410.2	109.6
2	3.1	-238.8	96.1
3	4.7	-89.8	82.7
4	6.3	36.9	69.3
5	7.8	141.2	55.9
6	9.4	223.1	42.4
7	11.0	282.6	29.0
8	12.5	292.2	-17.4
9	14.1	252.0	-30.8
10	15.7	189.5	-44.3
11	17.2	104.5	-57.7
12	18.8	-2.9	-71.1
13	20.4	-132.6	-84.6
14	21.9	-284.7	-98.0
15	23.5	-459.2	-111.4

Member # 3	Type = Beam	Length = 25.00 ft	
=====			
Design Results:	W24x104	E = 29000.00 ksi	
Total weight of beam =	2.60 kips	Fy = 36.00 ksi	
Nominal strength :			
Moment:	0.9*Mn= 625.65 kipft	Shear: 0.9*Vn = 594.86 kips	

Section Properties:	W24x104		
area= 30.60 in^2	Ix= 3100.0 in^4	Iy= 259.0 in^4	
d= 24.060 in	rx= 10.100 in	ry= 2.910 in	
bf= 12.750 in	sx= 258.00 in^3	sy= 40.70 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-459.2	111.4
1	1.6	-284.7	98.0
2	3.1	-132.6	84.6
3	4.7	-2.9	71.1
4	6.3	104.5	57.7
5	7.8	189.5	44.3
6	9.4	252.0	30.8
7	11.0	292.2	17.4
8	12.5	282.6	-29.0
9	14.1	223.1	-42.4
10	15.7	141.2	-55.9
11	17.2	36.9	-69.3
12	18.8	-89.8	-82.7
13	20.4	-238.8	-96.1
14	21.9	-410.2	-109.6
15	23.5	-604.1	-123.0

Member # 4	Type = Beam	Length = 23.50 ft	
=====			
Design Results:	W24x104	E = 29000.00 ksi	
Total weight of beam =	2.44 kips	Fy = 36.00 ksi	
Nominal strength :			
Moment:	0.9*Mn = 643.67 kipft	Shear: 0.9*Vn = 594.86 kips	
=====			
Section Properties:	W24x104		
area= 30.60 in^2	Ix= 3100.0 in^4	Iy= 259.0 in^4	
d= 24.060 in	rx= 10.100 in	ry= 2.910 in	
bf= 12.750 in	sx= 258.00 in^3	sy= 40.70 in^3	
=====			
Brace Condition :	No Lateral Supports Between Two Ends		
=====			
Load Combination Case For Design:	1.2*D+1.6*L		
=====			
Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-604.1	136.9
1	1.6	-399.5	124.2
2	3.1	-214.7	111.6
3	4.7	-49.8	99.0
4	6.3	95.5	86.4
5	7.8	220.9	73.7
6	9.4	326.5	61.1
7	11.0	412.4	48.5
8	12.5	452.7	2.9
9	14.1	447.3	-9.7
10	15.7	422.2	-22.3
11	17.2	377.3	-35.0
12	18.8	312.7	-47.6
13	20.4	228.2	-60.2
14	21.9	124.0	-72.8
15	23.5	0.0	-85.5

B. 2 Input and Output Files for Two Story Plane Frame

B 2.1 Input File for Two Story Plane Frame

```
STRUCTURE      Two story plane frame
TYPE PLANE FRAME
METHOD STIFFNESS
TABULATE ALL
NUMBER OF JOINTS      9
NUMBER OF MEMBERS     10
NUMBER OF SUPPORTS    3
NUMBER OF LOADINGS    3
NUMBER OF LOAD COMBINATIONS  3
JOINT COORDINATES
  1    0.00    0.00 S
  2    0.00   144.00
  3    0.00   264.00
  4  360.00    0.00 S
  5  360.00   144.00
  6  360.00   264.00
  7  720.00    0.00 S
  8  720.00   144.00
  9  720.00   264.00
MEMBER INCIDENCES
  1   1   2
  2   2   3
  3   4   5
  4   5   6
  5   7   8
  6   8   9
  7   2   5
  8   3   6
  9   5   8
 10   6   9
MEMBER PROPERTIES PRISMATIC
  1 AX IZ  9.13 110.0
  2 AX IZ 11.8 310.0
  3 AX IZ  9.13 110.0
  4 AX IZ  5.87 41.3
  5 AX IZ  9.13 110.0
  6 AX IZ 11.8 310.0
  7 AX IZ 25.3 1530.0
  8 AX IZ 25.3 1530.0
  9 AX IZ 25.3 1530.0
 10 AX IZ 25.3 1530.0
CONSTANTS E 29000 ALL
LOADING 1 - dead load
MEMBER LOADS
  7 FORCE Y CONCENTRATED -23.5 120.
  7 FORCE Y CONCENTRATED -23.5 240.
  8 FORCE Y CONCENTRATED -23.5 120.
  8 FORCE Y CONCENTRATED -23.5 240.
  9 FORCE Y CONCENTRATED -23.5 120.
  9 FORCE Y CONCENTRATED -23.5 240.
 10 FORCE Y CONCENTRATED -23.5 120.
 10 FORCE Y CONCENTRATED -23.5 240.
```

LOADING 2 - live load

MEMBER LOADS

```

7 FORCE Y CONCENTRATED -7.5 120.
7 FORCE Y CONCENTRATED -7.5 240.
8 FORCE Y CONCENTRATED -7.5 120.
8 FORCE Y CONCENTRATED -7.5 240.
9 FORCE Y CONCENTRATED -7.5 120.
9 FORCE Y CONCENTRATED -7.5 240.
10 FORCE Y CONCENTRATED -7.5 120.
10 FORCE Y CONCENTRATED -7.5 240.

```

LOADING 3 wind load (side)

JOINT LOADS

```

2 FORCE X 4.125
3 FORCE X 1.875

```

SOLVE

STRUCTURE Two story plane frame

LOADING 1 - dead load

MEMBER FORCES

MEMB	JOINT	AXIAL	FORCE	SHEAR	MOMENT
1	1	39.46	-4.36	-211.90	
1	2	-39.46	4.36	-416.28	
2	2	19.14	-14.79	-828.00	
2	3	-19.14	14.79	-947.28	
3	4	104.30	0.00	0.00	
3	5	-104.30	0.00	0.00	
4	5	53.39	0.00	0.00	
4	6	-53.39	0.00	0.00	
5	7	39.46	4.36	211.90	
5	8	-39.46	-4.36	416.28	
6	8	19.14	14.79	828.00	
6	9	-19.14	-14.79	947.28	
7	2	-10.43	20.54	1244.28	
7	5	10.43	25.56	-2147.76	
8	3	14.80	19.32	947.28	
8	6	-14.80	26.79	-2291.88	
9	5	-10.43	25.56	2147.76	
9	8	10.43	20.54	-1244.28	
10	6	14.79	26.79	2291.88	
10	9	-14.79	19.32	-947.28	

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z-ROTAT
2	-0.01473	-0.03736	-0.00819
3	0.02089	-0.05248	-0.01218
5	0.00000	-0.09856	0.00000
6	0.00000	-0.14064	0.00000
8	0.01473	-0.03736	0.00819
9	-0.02089	-0.05248	0.01218

STRUCTURE Two story plane frame

LOADING 2 - live load

=====

MEMBER FORCES

MEMB	JOINT	AXIAL	FORCE	SHEAR	MOMENT
1	1	12.98	-1.41		-68.64
1	2	-20.63	1.41		-134.76
2	2	6.29	-4.77		-268.08
2	3	-6.29	4.77		-306.72
3	4	34.04	0.00		0.00
3	5	-34.04	0.00		0.00
4	5	17.42	0.00		0.00
4	6	-17.42	0.00		0.00
5	7	12.98	1.41		68.64
5	8	-20.63	-1.41		134.76
6	8	6.29	4.77		268.08
6	9	-6.29	-4.77		306.72
7	2	-3.38	6.69		402.84
7	5	3.38	8.31		-695.40
8	3	4.79	6.29		306.72
8	6	-4.79	8.71		-742.08
9	5	-3.38	8.31		695.40
9	8	3.38	6.69		-402.84
10	6	4.79	8.71		742.08
10	9	-4.79	6.29		-306.72

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z-ROTAT
2	-0.00477	-0.01225	-0.00265
3	0.00677	-0.01720	-0.00394
5	0.00000	-0.03214	0.00000
6	0.00000	-0.04584	0.00000
8	0.00477	-0.01225	0.00265
9	-0.00677	-0.01720	0.00394

STRUCTURE Two story plane frame

=====

LOADING 3 wind load (side)

=====

MEMBER FORCES

MEMB	JOINT	AXIAL	FORCE	SHEAR	MOMENT
1	1	-0.85	1.90		154.08
1	2	0.85	-1.90		120.12
2	2	-0.20	0.41		9.84
2	3	0.20	-0.41		38.88
3	4	-0.01	2.22		168.79
3	5	0.01	-2.22		152.04
4	5	-0.01	1.03		55.56
4	6	0.01	-1.03		68.39
5	7	0.86	1.85		151.20
5	8	-0.86	-1.85		117.48
6	8	0.21	0.44		12.00
6	9	-0.21	-0.44		40.32
7	2	2.63	-0.65		-129.96
7	5	-2.63	0.65		-103.92
8	3	1.47	-0.20		-38.88

8	6	-1.47	0.20	-33.84
9	5	1.43	-0.65	-103.68
9	8	-1.43	0.65	-129.48
10	6	0.44	-0.21	-34.56
10	9	-0.44	0.21	-40.32

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z-ROTAT
2	0.36174	0.00080	-0.00136
3	0.49894	0.00096	-0.00039
5	0.35803	0.00000	-0.00068
6	0.49686	0.00001	-0.00025
8	0.35601	-0.00081	-0.00135
9	0.49625	-0.00097	-0.00040

B 2.2 Output File for Two Story Plane Frame

Member # 7	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-178.1	37.1
1	2.0	-104.0	37.1
2	4.0	-29.8	37.1
3	6.0	-44.4	37.1
4	8.0	118.6	37.1
5	10.0	192.8	-4.3
6	12.0	184.2	-4.3
7	14.0	175.6	-4.3
8	16.0	166.9	-4.3
9	18.0	158.3	-4.3
10	20.0	149.7	-45.7
11	22.0	58.2	-45.7
12	24.0	-33.2	-45.7
13	26.0	-124.6	-45.7
14	28.0	-216.1	-45.7
15	30.0	-307.5	-45.7

Member # 8	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-135.6	35.0
1	2.0	-65.6	35.0
2	4.0	4.3	35.0
3	6.0	74.3	35.0
4	8.0	144.3	35.0
5	10.0	214.3	-6.4
6	12.0	201.4	-6.4
7	14.0	188.6	-6.4
8	16.0	175.8	-6.4
9	18.0	162.9	-6.4
10	20.0	150.1	-47.8
11	22.0	54.5	-47.8
12	24.0	-41.2	-47.8
13	26.0	-136.8	-47.8
14	28.0	-232.5	-47.8
15	30.0	-328.1	-47.8
=====			

Member # 9	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-307.5	45.7
1	2.0	-216.1	45.7
2	4.0	-124.6	45.7
3	6.0	-33.2	45.7
4	8.0	58.2	45.7
5	10.0	149.7	4.3
6	12.0	158.3	4.3
7	14.0	166.9	4.3
8	16.0	175.6	4.3
9	18.0	184.2	4.3
10	20.0	192.8	-37.1
11	22.0	118.6	-37.1
12	24.0	44.4	-37.1
13	26.0	-29.8	-37.1
14	28.0	-104.0	-37.1
15	30.0	-178.1	-37.1

Member # 10	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-328.1	47.8
1	2.0	-232.5	47.8
2	4.0	-136.8	47.8
3	6.0	-41.2	47.8
4	8.0	54.5	47.8
5	10.0	150.1	47.8
6	12.0	162.9	6.4
7	14.0	175.8	6.4
8	16.0	188.6	6.4
9	18.0	201.4	6.4
10	20.0	214.3	-35.0
11	22.0	144.3	-35.0
12	24.0	74.3	-35.0
13	26.0	4.3	-35.0
14	28.0	-65.6	-35.0
15	30.0	-135.6	-35.0

Member # 1	Type = Column	Length = 12.00 ft
=====		
Design Results:	W8x31	E = 29000.00 ksi
Total weight of beam =	0.37 kips	Fy = 36.00 ksi

Section Properties:	W8x31	
area= 9.13 in ²	Ix= 110.0 in ⁴	Iy= 37.1 in ⁴
d= 8.000 in	rx= 3.470 in	ry= 2.020 in
bf= 7.995 in	sx= 27.50 in ³	sy= 9.27 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Effective Length Factors:	Unbraced Lengths:	
Kx = 1.29	Lx = 12.00 ft	
Ky = 1.00	Ly = 12.00 ft	
Applied Forces:	Nominal Strength:	
Axial Force Pu = 68.12 kips	Pn = 251.53 kips	
Smaller End Moment Mx1 = 30.34 kip-ft	Mnx = 91.20 kip-ft	
Larger End Moment Mx2 = 59.60 kip-ft		
=====		

Member # 2	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x40	E = 29000.00 ksi
Total weight of beam =	0.40 kips	Fy = 36.00 ksi

Section Properties:	W12x40	
area= 11.80 in ²	Ix= 310.0 in ⁴	Iy= 44.1 in ⁴
d= 11.940 in	rx= 5.130 in	ry= 1.930 in
bf= 8.005 in	sx= 51.90 in ³	sy= 11.00 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Effective Length Factors:	Unbraced Lengths:	
Kx = 1.23	Lx = 10.00 ft	
Ky = 1.00	Ly = 10.00 ft	
Applied Forces:	Nominal Strength:	
Axial Force Pu = 33.03 kips	Pn = 346.58 kips	
Smaller End Moment Mx1 = 118.54 kip-ft	Mnx = 172.50 kip-ft	
Larger End Moment Mx2 = 135.62 kip-ft		
=====		

Member # 3	Type = Column	Length = 12.00 ft
=====		
Design Results:	W8x31	E = 29000.00 ksi
Total weight of beam = 0.37 kips		Fy = 36.00 ksi
=====		
Section Properties:	W8x31	
area= 9.13 in ²	Ix= 110.0 in ⁴	Iy= 37.1 in ⁴
d= 8.000 in	rx= 3.470 in	ry= 2.020 in
bf= 7.995 in	sx= 27.50 in ³	sy= 9.27 in ³
=====		
Load Combination Case For Design:	1.2*D+1.6*L	
=====		
Effective Length Factors:	Unbraced Lengths:	
Kx = 1.22	Lx = 12.00 ft	
Ky = 1.00	Ly = 12.00 ft	
=====		
Applied Forces:	Nominal Strength:	
Axial Force Pu = 179.62 kips	Pn = 251.53 kips	
Smaller End Moment Mx1 = 0.00 kip-ft	Mnx = 91.20 kip-ft	
Larger End Moment Mx2 = 0.00 kip-ft		
=====		

Member # 4	Type = Column	Length = 10.00 ft
=====		
Design Results:	W6x15	E = 29000.00 ksi
Total weight of beam = 0.15 kips		Fy = 36.00 ksi
=====		
Section Properties:	W6x15	
area= 4.43 in ²	Ix= 29.1 in ⁴	Iy= 9.3 in ⁴
d= 5.990 in	rx= 2.560 in	ry= 1.450 in
bf= 5.990 in	sx= 9.72 in ³	sy= 3.11 in ³
=====		
Load Combination Case For Design:	1.4*D	
=====		
Effective Length Factors:	Unbraced Lengths:	
Kx = 1.12	Lx = 10.00 ft	
Ky = 1.00	Ly = 10.00 ft	
=====		
Applied Forces:	Nominal Strength:	
Axial Force Pu = 74.75 kips	Pn = 111.20 kips	
Smaller End Moment Mx1 = 0.00 kip-ft	Mnx = 0.00 kip-ft	
Larger End Moment Mx2 = 0.00 kip-ft		
=====		

Member # 5	Type = Column	Length = 12.00 ft
=====		
Design Results:	W8x31	E = 29000.00 ksi
Total weight of beam =	0.37 kips	Fy = 36.00 ksi

Section Properties:	W8x31	
area= 9.13 in ²	Ix= 110.0 in ⁴	Iy= 37.1 in ⁴
d= 8.000 in	rx= 3.470 in	ry= 2.020 in
bf= 7.995 in	sx= 27.50 in ³	sy= 9.27 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Effective Length Factors:	Unbraced Lengths:	
Kx = 1.21	Lx = 12.00 ft	
Ky = 1.00	Ly = 12.00 ft	
Applied Forces:	Nominal Strength:	
Axial Force Pu = 68.12 kips	Pn = 251.53 kips	
Smaller End Moment Mx1 = 30.34 kip-ft	Mnx = 91.20 kip-ft	
Larger End Moment Mx2 = 59.60 kip-ft		
=====		

Member # 6	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x40	E = 29000.00 ksi
Total weight of beam =	0.40 kips	Fy = 36.00 ksi

Section Properties:	W12x40	
area= 11.80 in ²	Ix= 310.0 in ⁴	Iy= 44.1 in ⁴
d= 11.940 in	rx= 5.130 in	ry= 1.930 in
bf= 8.005 in	sx= 51.90 in ³	sy= 11.00 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Effective Length Factors:	Unbraced Lengths:	
Kx = 1.13	Lx = 10.00 ft	
Ky = 1.00	Ly = 10.00 ft	
Applied Forces:	Nominal Strength:	
Axial Force Pu = 33.03 kips	Pn = 346.58 kips	
Smaller End Moment Mx1 = 118.54 kip-ft	Mnx = 172.50 kip-ft	
Larger End Moment Mx2 = 135.62 kip-ft		
=====		

B. 3 Input and Output Files for Two Story Space Frame

B 3.1 Input File for Two Story Space Frame

STRUCTURE Two story space frame - Test Problem 2

TYPE SPACE FRAME

METHOD STIFFNESS

TABULATE ALL

NUMBER OF JOINTS 18

NUMBER OF MEMBERS 26

NUMBER OF SUPPORTS 6

NUMBER OF LOADINGS 4

NUMBER OF LOAD COMBINATIONS 4

JOINT COORDINATES

1	0.00	0.00	0.00	S
2	0.00	144.00	0.00	
3	0.00	264.00	0.00	
4	360.00	0.00	0.00	S
5	360.00	144.00	0.00	
6	360.00	264.00	0.00	
7	720.00	0.00	0.00	S
8	720.00	144.00	0.00	
9	720.00	264.00	0.00	
10	0.00	0.00	360.00	S
11	0.00	144.00	360.00	
12	0.00	264.00	360.00	
13	360.00	0.00	360.00	S
14	360.00	144.00	360.00	
15	360.00	264.00	360.00	
16	720.00	0.00	360.00	S
17	720.00	144.00	360.00	
18	720.00	264.00	360.00	

MEMBER INCIDENCES

1	1	2
2	2	3
3	4	5
4	5	6
5	7	8
6	8	9
7	2	5
8	3	6
9	5	8
10	6	9
11	2	11
12	3	12
13	5	14
14	6	15
15	8	17
16	9	18
17	10	11
18	11	12
19	13	14
20	14	15

21 16 17
 22 17 18
 23 11 14
 24 12 15
 25 14 17
 26 15 18

MEMBER PROPERTIES PRISMATIC

1	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
2	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
3	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
4	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
5	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
6	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
7	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
8	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
9	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
10	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
11	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
12	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
13	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
14	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
15	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
16	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
17	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
18	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
19	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
20	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
21	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
22	AX	IX	IY	IZ	9.13	0.00	37.1	110.0
23	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
24	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
25	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00
26	AX	IX	IY	IZ	25.3	0.00	175.0	1530.00

CONSTANTS E 29000 ALL

LOADING 1 - dead load

MEMBER LOADS

7	FORCE	Y	CONCENTRATED	-23.5	120.
7	FORCE	Y	CONCENTRATED	-23.5	240.
8	FORCE	Y	CONCENTRATED	-23.5	120.
8	FORCE	Y	CONCENTRATED	-23.5	240.
9	FORCE	Y	CONCENTRATED	-23.5	120.
9	FORCE	Y	CONCENTRATED	-23.5	240.
10	FORCE	Y	CONCENTRATED	-23.5	120.
10	FORCE	Y	CONCENTRATED	-23.5	240.
23	FORCE	Y	CONCENTRATED	-23.5	120.
23	FORCE	Y	CONCENTRATED	-23.5	240.
24	FORCE	Y	CONCENTRATED	-23.5	120.
24	FORCE	Y	CONCENTRATED	-23.5	240.
25	FORCE	Y	CONCENTRATED	-23.5	120.
25	FORCE	Y	CONCENTRATED	-23.5	240.
26	FORCE	Y	CONCENTRATED	-23.5	120.
26	FORCE	Y	CONCENTRATED	-23.5	240.
11	FORCE	Y	UNIFORM	-0.05	
12	FORCE	Y	UNIFORM	-0.05	
13	FORCE	Y	UNIFORM	-0.10	
14	FORCE	Y	UNIFORM	-0.10	
15	FORCE	Y	UNIFORM	-0.05	
16	FORCE	Y	UNIFORM	-0.05	

LOADING 2 - live load

MEMBER LOADS

7 FORCE Y CONCENTRATED -7.5 120.
 7 FORCE Y CONCENTRATED -7.5 240.
 8 FORCE Y CONCENTRATED -7.5 120.
 8 FORCE Y CONCENTRATED -7.5 240.
 9 FORCE Y CONCENTRATED -7.5 120.
 9 FORCE Y CONCENTRATED -7.5 240.
 10 FORCE Y CONCENTRATED -7.5 120.
 10 FORCE Y CONCENTRATED -7.5 240.
 23 FORCE Y CONCENTRATED -7.5 120.
 23 FORCE Y CONCENTRATED -7.5 240.
 24 FORCE Y CONCENTRATED -7.5 120.
 24 FORCE Y CONCENTRATED -7.5 240.
 25 FORCE Y CONCENTRATED -7.5 120.
 25 FORCE Y CONCENTRATED -7.5 240.
 26 FORCE Y CONCENTRATED -7.5 120.
 26 FORCE Y CONCENTRATED -7.5 240.
 11 FORCE Y UNIFORM -0.0208
 12 FORCE Y UNIFORM -0.0208
 13 FORCE Y UNIFORM -0.0500
 14 FORCE Y UNIFORM -0.0500
 15 FORCE Y UNIFORM -0.0208
 16 FORCE Y UNIFORM -0.0208

LOADING 3 - wind load (front)

JOINT LOADS

11 FORCE Z -4.125
 12 FORCE Z -1.875
 14 FORCE Z -8.250
 15 FORCE Z -3.750
 17 FORCE Z -4.125
 18 FORCE Z -1.875

LOADING 4 wind load (side)

JOINT LOADS

11 FORCE X 4.125
 12 FORCE X 1.875
 2 FORCE X 4.125
 3 FORCE X 1.875

SOLVE

STRUCTURE Two story space frame

LOADING 1 - dead load

MEMBER FORCES

MEMB	JOINT	AXIAL FORCE	SHEAR FORCE Y	SHEAR FORCE Z	TORSION MOMENT	MOMENT Y	MOMENT Z
1	1	61.07	-4.62	1.30	0.00	-62.94	-218.33
1	2	-61.07	4.62	-1.30	0.00	-124.62	-446.48
2	2	30.07	-19.57	6.45	0.01	-349.72	-1064.70
2	3	-30.07	19.57	-6.45	-0.01	-424.42	-1284.10
3	4	137.86	0.00	2.60	0.00	-125.79	0.00
3	5	-137.86	0.00	-2.60	0.00	-249.08	0.00
4	5	69.86	0.00	12.88	0.00	-698.59	0.00
4	6	-69.86	0.00	-12.88	0.00	-847.41	0.00
5	7	61.07	4.62	1.30	0.00	-62.94	218.33
5	8	-61.07	-4.62	-1.30	0.00	-124.62	446.48
6	8	30.07	19.57	6.45	-0.01	-349.72	1064.70
6	9	-30.07	-19.57	-6.45	0.01	-424.42	1284.10

7	2	-14.96	22.00	0.00	-0.27	-0.09	1511.18
7	5	14.96	25.00	0.00	0.27	-0.48	-2050.78
8	3	19.57	21.07	0.00	-0.48	0.10	1284.10
8	6	-19.57	25.93	0.00	0.48	0.60	-2158.93
9	5	-14.96	25.00	0.00	0.27	0.48	2050.78
9	8	14.96	22.00	0.00	-0.27	0.09	-1511.18
10	6	19.57	25.93	0.00	0.48	-0.60	2158.93
10	9	-19.57	21.07	0.00	-0.48	-0.10	-1284.10
11	2	-5.15	9.00	0.00	0.00	0.07	474.07
11	11	5.15	9.00	0.00	0.00	-0.07	-474.07
12	3	6.45	9.00	0.00	0.00	-0.09	423.94
12	12	-6.45	9.00	0.00	0.00	0.09	-423.94
13	5	-10.28	18.00	0.00	0.00	0.00	948.21
13	14	10.28	18.00	0.00	0.00	0.00	-948.21
14	6	12.88	18.00	0.00	0.00	0.00	848.36
14	15	-12.88	18.00	0.00	0.00	0.00	-848.36
15	8	-5.15	9.00	0.00	0.00	-0.07	474.07
15	17	5.15	9.00	0.00	0.00	0.07	-474.07
16	9	6.45	9.00	0.00	0.00	0.09	423.94
16	18	-6.45	9.00	0.00	0.00	-0.09	-423.94
17	10	61.07	-4.62	-1.30	0.00	62.94	-218.33
17	11	-61.07	4.62	1.30	0.00	124.62	-446.48
18	11	30.07	-19.57	-6.45	-0.01	349.72	-1064.70
18	12	-30.07	19.57	6.45	0.01	424.42	-1284.10
19	13	137.86	0.00	-2.60	0.00	125.79	0.00
19	14	-137.86	0.00	2.60	0.00	249.08	0.00
20	14	69.86	0.00	-12.88	0.00	698.59	0.00
20	15	-69.86	0.00	12.88	0.00	847.41	0.00
21	16	61.07	4.62	-1.30	0.00	62.94	218.33
21	17	-61.07	-4.62	1.30	0.00	124.62	446.48
22	17	30.07	19.57	-6.45	0.01	349.72	1064.70
22	18	-30.07	-19.57	6.45	-0.01	424.42	1284.10
23	11	-14.96	22.00	0.00	0.27	0.09	1511.18
23	14	14.96	25.00	0.00	-0.27	0.48	-2050.78
24	12	19.57	21.07	0.00	0.48	-0.10	1284.10
24	15	-19.57	25.93	0.00	-0.48	-0.60	-2158.93
25	14	-14.96	25.00	0.00	-0.27	-0.48	2050.78
25	17	14.96	22.00	0.00	0.27	-0.09	-1511.18
26	15	19.57	25.93	0.00	-0.48	0.60	2158.93
26	18	-19.57	21.07	0.00	0.48	0.10	-1284.10

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z DISPL	X-ROTAT	Y-ROTAT	Z-ROTAT
2	-0.00725	-0.01723	-0.00271	0.00132	0.00001	-0.00166
3	0.00949	-0.02274	0.00339	0.00232	-0.00001	-0.00266
5	0.00000	-0.03889	-0.00541	0.00264	0.00000	0.00000
6	0.00000	-0.05169	0.00677	0.00464	0.00000	0.00000
8	0.00725	-0.01723	-0.00271	0.00132	-0.00001	0.00166
9	-0.00949	-0.02274	0.00339	0.00232	0.00001	0.00266
11	-0.00725	-0.01723	0.00271	-0.00132	-0.00001	-0.00166
12	0.00949	-0.02274	-0.00339	-0.00232	0.00001	-0.00266
14	0.00000	-0.03889	0.00541	-0.00264	0.00000	0.00000
15	0.00000	-0.05169	-0.00677	-0.00464	0.00000	0.00000
17	0.00725	-0.01723	0.00271	-0.00132	0.00001	0.00166
18	-0.00949	-0.02274	-0.00339	-0.00232	-0.00001	0.00266

STRUCTURE Two story space frame

LOADING 2 - live load

MEMBER FORCES

MEMB	JOINT	AXIAL FORCE	SHEAR FORCE Y	SHEAR FORCE Z	TORSION MOMENT	MOMENT Y	MOMENT Z
1	1	21.25	-1.48	0.54	0.00	-26.19	-69.82
1	2	-21.25	1.48	-0.54	0.00	-51.85	-142.77
2	2	10.48	-6.26	2.68	0.01	-145.52	-340.59
2	3	-10.48	6.26	-2.68	-0.01	-176.62	-410.94
3	4	50.48	0.00	1.30	0.00	-62.89	0.00
3	5	-50.48	0.00	-1.30	0.00	-124.53	0.00
4	5	25.54	0.00	6.44	0.00	-349.26	0.00
4	6	-25.54	0.00	-6.44	0.00	-423.64	0.00
5	7	21.25	1.48	0.54	0.00	-26.19	69.82
5	8	-21.25	-1.48	-0.54	0.00	-51.85	142.77
6	8	10.48	6.26	2.68	-0.01	-145.52	340.59
6	9	-10.48	-6.26	-2.68	0.01	-176.62	410.94
7	2	-4.79	7.03	0.00	-0.16	-0.05	483.35
7	5	4.79	7.97	0.00	0.16	-0.28	-653.32
8	3	6.26	6.73	0.00	-0.28	0.06	410.94
8	6	-6.26	8.27	0.00	0.28	0.35	-687.59
9	5	-4.79	7.97	0.00	0.16	0.28	653.32
9	8	4.79	7.03	0.00	-0.16	0.05	-483.35
10	6	6.26	8.27	0.00	0.28	-0.35	687.59
10	9	-6.26	6.73	0.00	-0.28	-0.06	-410.94
11	2	-2.14	3.74	0.00	0.00	0.04	197.21
11	11	2.14	3.74	0.00	0.00	-0.04	-197.21
12	3	2.69	3.74	0.00	0.00	-0.05	176.34
12	12	-2.69	3.74	0.00	0.00	0.05	-176.34
13	5	-5.14	9.00	0.00	0.00	0.00	474.11
13	14	5.14	9.00	0.00	0.00	0.00	-474.11
14	6	6.44	9.00	0.00	0.00	0.00	424.20
14	15	-6.44	9.00	0.00	0.00	0.00	-424.20
15	8	-2.14	3.74	0.00	0.00	-0.04	197.21
15	17	2.14	3.74	0.00	0.00	0.04	-197.21
16	9	2.69	3.74	0.00	0.00	0.05	176.34
16	18	-2.69	3.74	0.00	0.00	-0.05	-176.34
17	10	21.25	-1.48	-0.54	0.00	26.19	-69.82
17	11	-21.25	1.48	0.54	0.00	51.85	-142.77
18	11	10.48	-6.26	-2.68	-0.01	145.52	-340.59
18	12	-10.48	6.26	2.68	0.01	176.62	-410.94
19	13	50.48	0.00	-1.30	0.00	62.89	0.00
19	14	-50.48	0.00	1.30	0.00	124.53	0.00
20	14	25.54	0.00	-6.44	0.00	349.26	0.00
20	15	-25.54	0.00	6.44	0.00	423.64	0.00
21	16	21.25	1.48	-0.54	0.00	26.19	69.82
21	17	-21.25	-1.48	0.54	0.00	51.85	142.77
22	17	10.48	6.26	-2.68	0.01	145.52	340.59
22	18	-10.48	-6.26	2.68	-0.01	176.62	410.94
23	11	-4.79	7.03	0.00	0.16	0.05	483.35
23	14	4.79	7.97	0.00	-0.16	0.28	-653.32
24	12	6.26	6.73	0.00	0.28	-0.06	410.94
24	15	-6.26	8.27	0.00	-0.28	-0.35	-687.59
25	14	-4.79	7.97	0.00	-0.16	-0.28	653.32
25	17	4.79	7.03	0.00	0.16	-0.05	-483.35
26	15	6.26	8.27	0.00	-0.28	0.35	687.59
26	18	-6.26	6.73	0.00	0.28	0.06	-410.94

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z DISPL	X-ROTAT	Y-ROTAT	Z-ROTAT
2	-0.00232	-0.00599	-0.00113	0.00055	0.00001	-0.00053
3	0.00304	-0.00791	0.00141	0.00097	-0.00001	-0.00085
5	0.00000	-0.01424	-0.00270	0.00132	0.00000	0.00000
6	0.00000	-0.01892	0.00339	0.00232	0.00000	0.00000
8	0.00232	-0.00599	-0.00113	0.00055	-0.00001	0.00053
9	-0.00304	-0.00791	0.00141	0.00097	0.00001	0.00085
11	-0.00232	-0.00599	0.00113	-0.00055	-0.00001	-0.00053
12	0.00304	-0.00791	-0.00141	-0.00097	0.00001	-0.00085
14	0.00000	-0.01424	0.00270	-0.00132	0.00000	0.00000
15	0.00000	-0.01892	-0.00339	-0.00232	0.00000	0.00000
17	0.00232	-0.00599	0.00113	-0.00055	0.00001	0.00053
18	-0.00304	-0.00791	-0.00141	-0.00097	-0.00001	0.00085

STRUCTURE Two story space frame

LOADING 3 - wind load (front)

MEMBER FORCES

MEMB	JOINT	AXIAL FORCE	SHEAR FORCE Y	SHEAR FORCE Z	TORSION MOMENT	MOMENT Y	MOMENT Z
1	1	1.90	0.00	3.34	-0.16	-278.50	-0.10
1	2	-1.90	0.00	-3.34	0.16	-202.89	-0.11
2	2	0.57	0.00	1.15	-0.15	-35.24	-0.13
2	3	-0.57	0.00	-1.15	0.15	-103.08	-0.20
3	4	2.77	0.00	5.26	0.00	-435.52	0.00
3	5	-2.77	0.00	-5.26	0.00	-322.24	0.00
4	5	0.79	0.00	1.48	0.00	-35.16	0.00
4	6	-0.79	0.00	-1.48	0.00	-142.68	0.00
5	7	1.90	0.00	3.34	0.16	-278.50	0.10
5	8	-1.90	0.00	-3.34	-0.16	-202.89	0.11
6	8	0.57	0.00	1.15	0.15	-35.24	0.13
6	9	-0.57	0.00	-1.15	-0.15	-103.08	0.20
7	2	-0.08	0.00	0.15	0.17	-14.10	0.23
7	5	0.08	0.00	-0.15	-0.17	-39.70	0.24
8	3	-0.10	0.00	0.21	0.06	-19.44	0.20
8	6	0.10	0.00	-0.21	-0.06	-54.58	0.24
9	5	-0.08	0.00	-0.15	-0.17	39.70	-0.24
9	8	0.08	0.00	0.15	0.17	14.10	-0.23
10	6	-0.10	0.00	-0.21	-0.06	54.58	-0.24
10	9	0.10	0.00	0.21	0.06	19.44	-0.20
11	2	2.04	1.32	-0.08	0.00	14.09	238.30
11	11	-2.04	-1.32	0.08	0.00	14.14	238.44
12	3	0.95	0.57	-0.11	0.00	19.28	103.14
12	12	-0.95	-0.57	0.11	0.00	19.31	102.55
13	5	4.08	1.98	0.00	0.00	0.00	357.07
13	14	-4.08	-1.98	0.00	0.00	0.00	357.35
14	6	1.89	0.79	0.00	0.00	0.00	142.57
14	15	-1.89	-0.79	0.00	0.00	0.00	141.40
15	8	2.04	1.32	0.08	0.00	-14.09	238.30
15	17	-2.04	-1.32	-0.08	0.00	-14.14	238.44
16	9	0.95	0.57	0.11	0.00	-19.28	103.14
16	18	-0.95	-0.57	-0.11	0.00	-19.31	102.55
17	10	-1.90	0.00	3.37	-0.17	-280.43	0.10
17	11	1.90	0.00	-3.37	0.17	-204.69	0.11

18 11	-0.57	0.00	1.13	-0.15	-33.58	0.13
18 12	0.57	0.00	-1.13	0.15	-102.50	0.20
19 13	-2.77	0.00	5.31	0.00	-439.38	0.00
19 14	2.77	0.00	-5.31	0.00	-325.85	0.00
20 14	-0.79	0.00	1.44	0.00	-31.83	0.00
20 15	0.79	0.00	-1.44	0.00	-141.51	0.00
21 16	-1.90	0.00	3.37	0.17	-280.43	-0.10
21 17	1.90	0.00	-3.37	-0.17	-204.69	-0.11
22 17	-0.57	0.00	1.13	0.15	-33.58	-0.13
22 18	0.57	0.00	-1.13	-0.15	-102.50	-0.20
23 11	0.08	0.00	0.15	0.17	-14.16	-0.23
23 14	-0.08	0.00	-0.15	-0.17	-40.08	-0.24
24 12	0.10	0.00	0.21	0.05	-19.46	-0.20
24 15	-0.10	0.00	-0.21	-0.05	-54.75	-0.24
25 14	0.08	0.00	-0.15	-0.17	40.08	0.24
25 17	-0.08	0.00	0.15	0.17	14.16	0.23
26 15	0.10	0.00	-0.21	-0.05	54.75	0.24
26 18	-0.10	0.00	0.21	0.05	19.46	0.20

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z DISPL	X-ROTAT	Y-ROTAT	Z-ROTAT
2	-0.00004	-0.00054	-0.36743	-0.00162	0.00066	0.00000
3	-0.00005	-0.00064	-0.54493	-0.00071	0.00091	0.00000
5	0.00000	-0.00078	-0.56952	-0.00242	0.00000	0.00000
6	0.00000	-0.00093	-0.82260	-0.00098	0.00000	0.00000
8	0.00004	-0.00054	-0.36743	-0.00162	-0.00066	0.00000
9	0.00005	-0.00064	-0.54493	-0.00071	-0.00091	0.00000
11	0.00004	0.00054	-0.36958	-0.00162	0.00067	0.00000
12	0.00005	0.00064	-0.54593	-0.00070	0.00091	0.00000
14	0.00000	0.00078	-0.57381	-0.00243	0.00000	0.00000
15	0.00000	0.00093	-0.82460	-0.00096	0.00000	0.00000
17	-0.00004	0.00054	-0.36958	-0.00162	-0.00067	0.00000
18	-0.00005	0.00064	-0.54592	-0.00070	-0.00091	0.00000

STRUCTURE Two story space frame

LOADING 4 wind load (side)

MEMBER FORCES

MEMB	JOINT	AXIAL FORCE	SHEAR FORCE Y	SHEAR FORCE Z	TORSION MOMENT	MOMENT Y	MOMENT Z
1	1	-0.82	1.93	0.00	0.00	0.00	163.82
1	2	0.82	-1.93	0.00	0.00	0.00	114.63
2	2	-0.24	0.35	0.00	0.00	0.00	-3.07
2	3	0.24	-0.35	0.00	0.00	0.00	44.86
3	4	-0.01	2.20	0.00	0.00	0.00	174.77
3	5	0.01	-2.20	0.00	0.00	0.00	141.81
4	5	-0.01	1.13	0.00	0.00	0.00	52.90
4	6	0.01	-1.13	0.00	0.00	0.00	82.43
5	7	0.82	1.87	0.00	0.00	0.00	158.95
5	8	-0.82	-1.87	0.00	0.00	0.00	110.02
6	8	0.25	0.40	0.00	0.00	0.00	1.09
6	9	-0.25	-0.40	0.00	0.00	0.00	46.79
7	2	2.54	-0.58	0.00	0.00	0.00	-111.56
7	5	-2.54	0.58	0.00	0.00	0.00	-97.47
8	3	1.53	-0.24	0.00	0.00	0.00	-44.86

8	6	-1.53	0.24	0.00	0.00	0.00	-40.74
9	5	1.47	-0.58	0.00	0.00	0.00	-97.25
9	8	-1.47	0.58	0.00	0.00	0.00	-111.11
10	6	0.40	-0.25	0.00	0.00	0.00	-41.69
10	9	-0.40	0.25	0.00	0.00	0.00	-46.79
11	2	0.00	0.00	0.00	0.00	0.00	0.00
11	11	0.00	0.00	0.00	0.00	0.00	0.00
12	3	0.00	0.00	0.00	0.00	0.00	0.00
12	12	0.00	0.00	0.00	0.00	0.00	0.00
13	5	0.00	0.00	0.00	0.00	0.00	0.00
13	14	0.00	0.00	0.00	0.00	0.00	0.00
14	6	0.00	0.00	0.00	0.00	0.00	0.00
14	15	0.00	0.00	0.00	0.00	0.00	0.00
15	8	0.00	0.00	0.00	0.00	0.00	0.00
15	17	0.00	0.00	0.00	0.00	0.00	0.00
16	9	0.00	0.00	0.00	0.00	0.00	0.00
16	18	0.00	0.00	0.00	0.00	0.00	0.00
17	10	-0.82	1.93	0.00	0.00	0.00	163.82
17	11	0.82	-1.93	0.00	0.00	0.00	114.63
18	11	-0.24	0.35	0.00	0.00	0.00	-3.07
18	12	0.24	-0.35	0.00	0.00	0.00	44.86
19	13	-0.01	2.20	0.00	0.00	0.00	174.77
19	14	0.01	-2.20	0.00	0.00	0.00	141.81
20	14	-0.01	1.13	0.00	0.00	0.00	52.90
20	15	0.01	-1.13	0.00	0.00	0.00	82.43
21	16	0.82	1.87	0.00	0.00	0.00	158.95
21	17	-0.82	-1.87	0.00	0.00	0.00	110.02
22	17	0.25	0.40	0.00	0.00	0.00	1.09
22	18	-0.25	-0.40	0.00	0.00	0.00	46.79
23	11	2.54	-0.58	0.00	0.00	0.00	-111.56
23	14	-2.54	0.58	0.00	0.00	0.00	-97.47
24	12	1.53	-0.24	0.00	0.00	0.00	-44.86
24	15	-1.53	0.24	0.00	0.00	0.00	-40.74
25	14	1.47	-0.58	0.00	0.00	0.00	-97.25
25	17	-1.47	0.58	0.00	0.00	0.00	-111.11
26	15	0.40	-0.25	0.00	0.00	0.00	-41.69
26	18	-0.40	0.25	0.00	0.00	0.00	-46.79

FREE JOINT DISPLACEMENTS

JOINT	X DISPL	Y DISPL	Z DISPL	X-ROTAT	Y-ROTAT	Z-ROTAT
2	0.07892	0.00023	0.00000	0.00000	0.00000	-0.00036
3	0.11313	0.00027	0.00000	0.00000	0.00000	-0.00014
5	0.07768	0.00000	0.00000	0.00000	0.00000	-0.00024
6	0.11239	0.00000	0.00000	0.00000	0.00000	-0.00011
8	0.07697	-0.00023	0.00000	0.00000	0.00000	-0.00036
9	0.11220	-0.00028	0.00000	0.00000	0.00000	-0.00015
11	0.07892	0.00023	0.00000	0.00000	0.00000	-0.00036
12	0.11313	0.00027	0.00000	0.00000	0.00000	-0.00014
14	0.07768	0.00000	0.00000	0.00000	0.00000	-0.00024
15	0.11239	0.00000	0.00000	0.00000	0.00000	-0.00011
17	0.07697	-0.00023	0.00000	0.00000	0.00000	-0.00036
18	0.11220	-0.00028	0.00000	0.00000	0.00000	-0.00015

B 3.2 Output File for Two Story Space Frame

Member # 7	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x76	E = 29000.00 ksi	
Total weight of beam =	2.28 kips	Fy = 36.00 ksi	

Nominal Strength:			
Moment 0.9*Mn =	307.31 kip-ft	Shear 0.9*Vn = 433.51 kips	

Section Properties:	W18x76		
area= 22.30 in^2	Ix= 1330.0 in^4	Iy= 152.0 in^4	
d= 18.210 in	rx= 7.730 in	ry= 2.610 in	
bf= 11.035 in	sx= 146.00 in^3	sy= 27.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-215.6	38.9
1	2.0	-137.9	38.9
2	4.0	-60.2	38.9
3	6.0	17.5	38.9
4	8.0	95.3	38.9
5	10.0	173.0	-2.6
6	12.0	167.8	-2.6
7	14.0	162.7	-2.6
8	16.0	157.6	-2.6
9	18.0	152.5	-2.6
10	20.0	147.4	-44.0
11	22.0	59.5	-44.0
12	24.0	-28.4	-44.0
13	26.0	-116.3	-44.0
14	28.0	-204.3	-44.0
15	30.0	-292.2	-44.0
=====			

Member # 8	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment 0.9*Mn =	363.85 kip-ft	Shear 0.9*Vn = 491.83 kips	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-183.2	37.3
1	2.0	-108.7	37.3
2	4.0	-34.2	37.3
3	6.0	-40.4	37.3
4	8.0	114.9	37.3
5	10.0	189.4	-4.3
6	12.0	181.1	-4.3
7	14.0	172.8	-4.3
8	16.0	164.5	-4.3
9	18.0	156.2	-4.3
10	20.0	147.9	-45.7
11	22.0	56.8	-45.7
12	24.0	-34.2	-45.7
13	26.0	-125.6	-45.7
14	28.0	-216.5	-45.7
15	30.0	-307.5	-45.7

Member # 9	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x76	E = 29000.00 ksi	
Total weight of beam =	2.28 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment 0.9*Mn =	307.31 kip-ft	Shear 0.9*Vn = 433.51 kips	

Section Properties:	W18x76		
area= 22.30 in^2	Ix= 1330.0 in^4	Iy= 152.0 in^4	
d= 18.210 in	rx= 7.730 in	ry= 2.610 in	
bf= 11.035 in	sx= 146.00 in^3	sy= 27.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-292.2	44.0
1	2.0	-204.3	44.0
2	4.0	-116.3	44.0
3	6.0	-28.4	44.0
4	8.0	59.5	44.0
5	10.0	147.4	2.6
6	12.0	152.5	2.6
7	14.0	157.6	2.6
8	16.0	162.7	2.6
9	18.0	167.8	2.6
10	20.0	173.0	-38.9
11	22.0	95.3	-38.9
12	24.0	17.5	-38.9
13	26.0	-60.2	-38.9
14	28.0	-137.9	-38.9
15	30.0	-215.6	-38.9
=====			

Member # 10	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x86	E = 29000.00 ksi	
Total weight of beam =	2.58 kips	Fy = 36.00 ksi	

Nominal Strength:			
Moment 0.9*Mn =	363.85 kip-ft	Shear 0.9*Vn = 491.83 kips	

Section Properties:	W18x86		
area= 25.30 in^2	Ix= 1530.0 in^4	Iy= 175.0 in^4	
d= 18.390 in	rx= 7.770 in	ry= 2.630 in	
bf= 11.090 in	sx= 166.00 in^3	sy= 31.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-307.5	45.7
1	2.0	-216.5	45.7
2	4.0	-125.6	45.7
3	6.0	-34.2	45.7
4	8.0	56.8	45.7
5	10.0	147.9	45.7
6	12.0	156.2	4.3
7	14.0	164.5	4.3
8	16.0	172.8	4.3
9	18.0	181.1	4.3
10	20.0	189.4	-37.3
11	22.0	114.9	-37.3
12	24.0	40.4	-37.3
13	26.0	-34.2	-37.3
14	28.0	-108.7	-37.3
15	30.0	-183.2	-37.3
=====			

Member # 11	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W10x39	E = 29000.00 ksi	
Total weight of beam =	1.17 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment	0.9*Mn = 84.35 kip-ft	Shear 0.9*Vn = 223.56 kips	

Section Properties:	W10x39		
area= 11.50 in^2	Ix= 209.0 in^4	Iy= 45.0 in^4	
d= 9.920 in	rx= 4.270 in	ry= 1.980 in	
bf= 7.985 in	sx= 42.10 in^3	sy= 11.30 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-81.4	14.8
1	2.0	-53.6	13.0
2	4.0	-29.3	11.3
3	6.0	-8.5	9.6
4	8.0	8.9	7.8
5	10.0	22.8	6.1
6	12.0	33.2	4.3
7	14.0	40.1	2.6
8	16.0	43.6	0.9
9	18.0	43.5	-0.9
10	20.0	40.0	-2.6
11	22.0	33.0	-4.4
12	24.0	22.5	-6.1
13	26.0	8.6	-7.9
14	28.0	-8.9	-9.6
15	30.0	-29.8	-11.3

Member # 12	Type = Beam	Length = 30.00 ft
=====		
Design Results:	W8x35	E = 29000.00 ksi
Total weight of beam =	1.05 kips	Fy = 36.00 ksi
Nominal Strength:		
Moment 0.9*Mn =	67.11 kip-ft	Shear 0.9*Vn = 200.23 kips

Section Properties:	W8x35	
area= 10.30 in^2	Ix= 127.0 in^4	Iy= 42.6 in^4
d= 8.123 in	rx= 3.510 in	ry= 2.030 in
bf= 8.020 in	sx= 31.20 in^3	sy= 10.60 in^3

Brace Condition :	No Lateral Supports Between Two Ends	

Load Combination Case For Design:	1.2*D+1.6*L	

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)
		Design Shear (kips)
0	0.0	-65.9
1	2.0	-33.6
2	4.0	-6.0
3	6.0	17.1
4	8.0	35.6
5	10.0	49.4
6	12.0	58.6
7	14.0	63.2
8	16.0	63.2
9	18.0	58.6
10	20.0	49.4
11	22.0	35.6
12	24.0	17.1
13	26.0	-6.0
14	28.0	-33.6
15	30.0	-65.9

Member # 13	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W12x58	E = 29000.00 ksi	
Total weight of beam =	1.74 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment 0.9*Mn =	176.42 kip-ft	Shear 0.9*Vn = 330.48 kips	

Section Properties:	W12x58		
area= 17.00 in^2	Ix= 475.0 in^4	Iy= 107.0 in^4	
d= 12.19 in	rx= 5.280 in	ry= 2.510 in	
bf= 10.010 in	sx= 78.00 in^3	sy= 21.40 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-158.0	37.1
1	2.0	-88.8	32.1
2	4.0	-29.5	27.2
3	6.0	19.9	22.2
4	8.0	59.5	17.3
5	10.0	89.2	12.4
6	12.0	108.9	7.4
7	14.0	118.8	2.5
8	16.0	118.8	-2.5
9	18.0	108.9	-7.4
10	20.0	89.2	-12.4
11	22.0	59.5	-17.3
12	24.0	19.9	-22.2
13	26.0	-29.5	-27.2
14	28.0	-88.8	-32.1
15	30.0	-158.0	-37.1
=====			

Member # 14	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W12x53	E = 29000.00 ksi	
Total weight of beam =	1.59 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment 0.9*Mn =	154.29 kip-ft	Shear 0.9*Vn = 303.26 kips	

Section Properties:	W12x53		
area= 15.60 in^2	Ix= 425.0 in^4	Iy= 95.8 in^4	
d= 12.060 in	rx= 5.230 in	ry= 2.480 in	
bf= 9.995 in	sx= 70.60 in^3	sy= 19.20 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-141.4	37.1
1	2.0	-72.2	32.1
2	4.0	-12.9	27.2
3	6.0	36.6	22.2
4	8.0	76.1	17.3
5	10.0	105.8	12.4
6	12.0	125.6	7.4
7	14.0	135.5	2.5
8	16.0	135.5	-2.5
9	18.0	125.6	-7.4
10	20.0	105.8	-12.4
11	22.0	76.1	-17.3
12	24.0	36.6	-22.2
13	26.0	-12.9	-27.2
14	28.0	-72.2	-32.1
15	30.0	-141.4	-37.1
=====			

Member # 15 Type = Beam Length = 30.00 ft

Design Results: W10x39 E = 29000.00 ksi
Total weight of beam = 1.17 kips Fy = 36.00 ksi

Nominal Strength:
Moment $0.9 \cdot M_n = 84.35$ kip-ft Shear $0.9 \cdot V_n = 223.56$ kips

Section Properties: W10x39
area= 11.50 in² Ix= 209.0 in⁴ Iy= 45.0 in⁴
d= 9.920 in rx= 4.270 in ry= 1.980 in
bf= 7.985 in sx= 42.10 in³ sy= 11.30 in³

Brace Condition : No Lateral Supports Between Two Ends

Load Combination Case For Design: 1.2*D+1.6*L

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-81.4	14.8
1	2.0	-53.6	13.0
2	4.0	-29.3	11.3
3	6.0	-8.5	9.6
4	8.0	8.9	7.8
5	10.0	22.8	6.1
6	12.0	33.2	4.3
7	14.0	40.1	2.6
8	16.0	43.6	0.9
9	18.0	43.5	-0.9
10	20.0	40.0	-2.6
11	22.0	33.0	-4.4
12	24.0	22.5	-6.1
13	26.0	8.6	-7.9
14	28.0	-8.9	-9.6
15	30.0	-29.8	-11.3

Member # 16	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W8x35	E = 29000.00 ksi	
Total weight of beam =	1.05 kips	Fy = 36.00 ksi	

Nominal Strength:			
Moment	0.9*Mn = 67.11 kip-ft	Shear 0.9*Vn = 200.23 kips	

Section Properties:	W8x35		
area= 10.30 in^2	Ix= 127.0 in^4	Iy= 42.6 in^4	
d= 8.123 in	rx= 3.510 in	ry= 2.030 in	
bf= 8.020 in	sx= 31.20 in^3	sy= 10.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-65.9	17.3
1	2.0	-33.6	15.0
2	4.0	-6.0	12.7
3	6.0	17.1	10.4
4	8.0	35.6	8.1
5	10.0	49.4	5.8
6	12.0	58.6	3.5
7	14.0	63.2	1.2
8	16.0	63.2	-1.2
9	18.0	58.6	-3.5
10	20.0	49.4	-5.8
11	22.0	35.6	-8.1
12	24.0	17.1	-10.4
13	26.0	-6.0	-12.7
14	28.0	-33.6	-15.0
15	30.0	-65.9	-17.3

Member # 23 Type = Beam Length = 30.00 ft

Design Results: W18x76 E = 29000.00 ksi
Total weight of beam = 2.28 kips Fy = 36.00 ksi

Nominal Strength:
Moment $0.9 \cdot M_n = 307.31$ kip-ft Shear $0.9 \cdot V_n = 433.51$ kips

Section Properties: W18x76
area= 22.30 in² Ix= 1330.0 in⁴ Iy= 152.0 in⁴
d= 18.210 in rx= 7.730 in ry= 2.610 in
bf= 11.035 in sx= 146.00 in³ sy= 27.60 in³

Brace Condition : No Lateral Supports Between Two Ends

Load Combination Case For Design: 1.2*D+1.6*L

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-215.6	38.9
1	2.0	-137.9	38.9
2	4.0	-60.2	38.9
3	6.0	17.5	38.9
4	8.0	95.3	38.9
5	10.0	173.0	-2.6
6	12.0	167.8	-2.6
7	14.0	162.7	-2.6
8	16.0	157.6	-2.6
9	18.0	152.5	-2.6
10	20.0	147.4	-44.0
11	22.0	59.5	-44.0
12	24.0	-28.4	-44.0
13	26.0	-116.3	-44.0
14	28.0	-204.3	-44.0
15	30.0	-292.2	-44.0

Member # 24 Type = Beam Length = 30.00 ft

Design Results: W18x86 E = 29000.00 ksi
 Total weight of beam = 2.58 kips Fy = 36.00 ksi

Nominal Strength:
 Moment $0.9 \cdot M_n = 363.85$ kip-ft Shear $0.9 \cdot V_n = 491.83$ kips

Section Properties: W18x86
 area= 25.30 in² Ix= 1530.0 in⁴ Iy= 175.0 in⁴
 d= 18.390 in rx= 7.770 in ry= 2.630 in
 bf= 11.090 in sx= 166.00 in³ sy= 31.60 in³

Brace Condition : No Lateral Supports Between Two Ends

Load Combination Case For Design: 1.2*D+1.6*L

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-183.2	37.3
1	2.0	-108.7	37.3
2	4.0	-34.2	37.3
3	6.0	-40.4	37.3
4	8.0	114.9	37.3
5	10.0	189.4	-4.3
6	12.0	181.1	-4.3
7	14.0	172.8	-4.3
8	16.0	164.5	-4.3
9	18.0	156.2	-4.3
10	20.0	147.9	-45.7
11	22.0	56.8	-45.7
12	24.0	-34.2	-45.7
13	26.0	-125.6	-45.7
14	28.0	-216.5	-45.7
15	30.0	-307.5	-45.7

Member # 25	Type = Beam	Length = 30.00 ft	
=====			
Design Results:	W18x76	E = 29000.00 ksi	
Total weight of beam =	2.28 kips	Fy = 36.00 ksi	
Nominal Strength:			
Moment	0.9*Mn = 307.31 kip-ft	Shear 0.9*Vn = 433.51 kips	

Section Properties:	W18x76		
area= 22.30 in^2	Ix= 1330.0 in^4	Iy= 152.0 in^4	
d= 18.210 in	rx= 7.730 in	ry= 2.610 in	
bf= 11.035 in	sx= 146.00 in^3	sy= 27.60 in^3	

Brace Condition :	No Lateral Supports Between Two Ends		

Load Combination Case For Design:	1.2*D+1.6*L		

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-292.2	44.0
1	2.0	-204.3	44.0
2	4.0	-116.3	44.0
3	6.0	-28.4	44.0
4	8.0	59.5	44.0
5	10.0	147.4	2.6
6	12.0	152.5	2.6
7	14.0	157.6	2.6
8	16.0	162.7	2.6
9	18.0	167.8	2.6
10	20.0	173.0	-38.9
11	22.0	95.3	-38.9
12	24.0	17.5	-38.9
13	26.0	-60.2	-38.9
14	28.0	-137.9	-38.9
15	30.0	-215.6	-38.9

Member # 26 Type = Beam Length = 30.00 ft

Design Results: W18x86 E = 29000.00 ksi
Total weight of beam = 2.58 kips Fy = 36.00 ksi

Nominal Strength:
Moment 0.9*Mn = 363.85 kip-ft Shear 0.9*Vn = 491.83 kips

Section Properties: W18x86
area= 25.30 in² Ix= 1530.0 in⁴ Iy= 175.0 in⁴
d= 18.390 in rx= 7.770 in ry= 2.630 in
bf= 11.090 in sx= 166.00 in³ sy= 31.60 in³

Brace Condition : No Lateral Supports Between Two Ends

Load Combination Case For Design: 1.2*D+1.6*L

Section Number	Distance From Left End (ft)	Design Moment (kip-ft)	Design Shear (kips)
0	0.0	-307.5	45.7
1	2.0	-216.5	45.7
2	4.0	-125.6	45.7
3	6.0	-34.2	45.7
4	8.0	56.8	45.7
5	10.0	147.9	45.7
6	12.0	156.2	4.3
7	14.0	164.5	4.3
8	16.0	172.8	4.3
9	18.0	181.1	4.3
10	20.0	189.4	-37.3
11	22.0	114.9	-37.3
12	24.0	40.4	-37.3
13	26.0	-34.2	-37.3
14	28.0	-108.7	-37.3
15	30.0	-183.2	-37.3

Member # 1	Type = Column	Length = 12.00 ft
=====		
Design Results:	W10x49	E = 29000.00 ksi
Total weight of beam = 0.59 kips		Fy = 36.00 ksi

Section Properties:	W10x49	
area= 14.40 in ²	Ix= 272.0 in ⁴	Iy= 93.4 in ⁴
d= 9.980 in	rx= 4.350 in	ry= 2.540 in
bf= 10.000 in	sx= 54.60 in ³	sy= 18.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.23		Lx = 12.00 ft
Ky = 1.31		Ly = 12.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 107.28 kips		Pn = 387.36 kips
Smaller End Moment	Mx1 = 31.14 kip-ft	Mnx = 181.20 kip-ft
	My1 = 9.79 kip-ft	Mny = 84.90 kip-ft
Larger End Moment	Mx2 = 63.68 kip-ft	
	My2 = 19.38 kip-ft	
=====		

Member # 2	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x87	E = 29000.00 ksi
Total weight of beam = 0.87 kips		Fy = 36.00 ksi

Section Properties:	W12x87	
area= 25.60 in ²	Ix= 740.0 in ⁴	Iy= 241.0 in ⁴
d= 12.530 in	rx= 5.380 in	ry= 3.070 in
bf= 12.125 in	sx= 118.00 in ³	sy= 39.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.13		Lx = 10.00 ft
Ky = 1.31		Ly = 10.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 52.85 kips		Pn = 803.51 kips
Smaller End Moment	Mx1 = 151.88 kip-ft	Mnx = 396.00 kip-ft
	My1 = 54.37 kip-ft	Mny = 181.20 kip-ft
Larger End Moment	Mx2 = 183.20 kip-ft	
	My2 = 65.99 kip-ft	
=====		

Member # 3	Type = Column	Length = 12.00 ft
=====		
Design Results:	W12x65	E = 29000.00 ksi
Total weight of beam = 0.78 kips		Fy = 36.00 ksi

Section Properties:	W12x65	
area= 19.10 in ²	Ix= 533.0 in ⁴	Iy= 174.0 in ⁴
d= 12.120 in	rx= 5.280 in	ry= 3.020 in
bf= 12.000 in	sx= 87.90 in ³	sy= 29.10 in ³

Load Combination Case For Design:	1.2*D+1.3*W+0.5*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.20		Lx = 12.00 ft
Ky = 1.26		Ly = 12.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 194.27 kips		Pn = 568.42 kips
Smaller End Moment Mx1 = 0.00 kip-ft		Mnx = 290.40 kip-ft
	My1 = 62.38 kip-ft	Mny = 132.30 kip-ft
Larger End Moment Mx2 = 0.00 kip-ft		
	My2 = 65.01 kip-ft	
=====		

Member # 4	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x87	E = 29000.00 ksi
Total weight of beam = 0.87 kips		Fy = 36.00 ksi

Section Properties:	W12x87	
area= 25.60 in ²	Ix= 740.0 in ⁴	Iy= 241.0 in ⁴
d= 12.530 in	rx= 5.380 in	ry= 3.070 in
bf= 12.125 in	sx= 118.00 in ³	sy= 39.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.08		Lx = 10.00 ft
Ky = 1.19		Ly = 10.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 124.70 kips		Pn = 822.60 kips
Smaller End Moment Mx1 = 0.00 kip-ft		Mnx = 396.00 kip-ft
	My1 = 116.43 kip-ft	Mny = 181.20 kip-ft
Larger End Moment Mx2 = 0.00 kip-ft		
	My2 = 141.23 kip-ft	
=====		

Member # 5	Type = Column	Length = 12.00 ft
=====		
Design Results:	W10x49	E = 29000.00 ksi
Total weight of beam = 0.59 kips		Fy = 36.00 ksi

Section Properties:	W10x49	
area= 14.40 in ²	Ix= 272.0 in ⁴	Iy= 93.4 in ⁴
d= 9.980 in	rx= 4.350 in	ry= 2.540 in
bf= 10.000 in	sx= 54.60 in ³	sy= 18.70 in ³

Load Combination Case For Design:	1.2*D+1.3*W+0.5*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.20		Lx = 12.00 ft
Ky = 1.25		Ly = 12.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 86.38 kips		Pn = 397.64 kips
Smaller End Moment Mx1 = 24.75 kip-ft		Mnx = 181.20 kip-ft
	My1 = 36.60 kip-ft	Mny = 84.90 kip-ft
Larger End Moment Mx2 = 50.61 kip-ft		
	My2 = 37.56 kip-ft	
=====		

Member # 6	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x87	E = 29000.00 ksi
Total weight of beam = 0.87 kips		Fy = 36.00 ksi

Section Properties:	W12x87	
area= 25.60 in ²	Ix= 740.0 in ⁴	Iy= 241.0 in ⁴
d= 12.530 in	rx= 5.380 in	ry= 3.070 in
bf= 12.125 in	sx= 118.00 in ³	sy= 39.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.08		Lx = 10.00 ft
Ky = 1.19		Ly = 10.00 ft
Applied Forces:		Nominal Strength:
-----		-----
Axial Force Pu = 52.85 kips		Pn = 822.60 kips
Smaller End Moment Mx1 = 151.88 kip-ft		Mnx = 396.00 kip-ft
	My1 = 54.37 kip-ft	Mny = 181.20 kip-ft
Larger End Moment Mx2 = 183.20 kip-ft		
	My2 = 65.99 kip-ft	
=====		

Member # 17	Type = Column	Length = 12.00 ft
=====		
Design Results:	W14x48	E = 29000.00 ksi
Total weight of beam = 0.59 kips		Fy = 36.00 ksi

Section Properties:	W14x48	
area= 14.10 in ²	Ix= 485.0 in ⁴	Iy= 51.4 in ⁴
d= 13.790 in	rx= 5.580 in	ry= 1.910 in
bf= 8.030 in	sx= 70.30 in ³	sy= 12.80 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:	Unbraced Lengths:	
Kx = 1.20	Lx = 12.00 ft	
Ky = 1.26	Ly = 12.00 ft	
Applied Forces:	Nominal Strength:	

Axial Force Pu = 107.28 kips	Pn = 314.66 kips	
Smaller End Moment Mx1 = 31.14 kip-ft	Mnx = 235.20 kip-ft	
My1 = 9.79 kip-ft	Mny = 58.80 kip-ft	
Larger End Moment Mx2 = 63.68 kip-ft		
My2 = 19.38 kip-ft		
=====		

Member # 18	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x87	E = 29000.00 ksi
Total weight of beam = 0.87 kips		Fy = 36.00 ksi

Section Properties:	W12x87	
area= 25.60 in ²	Ix= 740.0 in ⁴	Iy= 241.0 in ⁴
d= 12.530 in	rx= 5.380 in	ry= 3.070 in
bf= 12.125 in	sx= 118.00 in ³	sy= 39.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:	Unbraced Lengths:	
Kx = 1.09	Lx = 10.00 ft	
Ky = 1.21	Ly = 10.00 ft	
Applied Forces:	Nominal Strength:	

Axial Force Pu = 52.85 kips	Pn = 819.08 kips	
Smaller End Moment Mx1 = 151.88 kip-ft	Mnx = 396.00 kip-ft	
My1 = 54.37 kip-ft	Mny = 181.20 kip-ft	
Larger End Moment Mx2 = 183.20 kip-ft		
My2 = 65.99 kip-ft		
=====		

Member # 19	Type = Column	Length = 12.00 ft
=====		
Design Results:	W12x65	E = 29000.00 ksi
Total weight of beam = 0.78 kips		Fy = 36.00 ksi

Section Properties:	W12x65	
area= 19.10 in ²	Ix= 533.0 in ⁴	Iy= 174.0 in ⁴
d= 12.120 in	rx= 5.280 in	ry= 3.020 in
bf= 12.000 in	sx= 87.90 in ³	sy= 29.10 in ³

Load Combination Case For Design:	1.2*D+1.3*W+0.5*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.20		Lx = 12.00 ft
Ky = 1.26		Ly = 12.00 ft
Applied Forces:		Nominal Strength:

Axial Force Pu = 190.66 kips		Pn = 568.42 kips
Smaller End Moment Mx1 = 15.36 kip-ft		Mnx = 290.40 kip-ft
	My1 = 15.20 kip-ft	Mny = 132.30 kip-ft
Larger End Moment Mx2 = 18.93 kip-ft		
	My2 = 30.10 kip-ft	
=====		

Member # 20	Type = Column	Length = 10.00 ft
=====		
Design Results:	W12x87	E = 29000.00 ksi
Total weight of beam = 0.87 kips		Fy = 36.00 ksi

Section Properties:	W12x87	
area= 25.60 in ²	Ix= 740.0 in ⁴	Iy= 241.0 in ⁴
d= 12.530 in	rx= 5.380 in	ry= 3.070 in
bf= 12.125 in	sx= 118.00 in ³	sy= 39.70 in ³

Load Combination Case For Design:	1.2*D+1.6*L	

Beta Angle = 0.0		
Effective Length Factors:		Unbraced Lengths:
Kx = 1.08		Lx = 10.00 ft
Ky = 1.19		Ly = 10.00 ft
Applied Forces:		Nominal Strength:

Axial Force Pu = 124.70 kips		Pn = 822.60 kips
Smaller End Moment Mx1 = 0.00 kip-ft		Mnx = 396.00 kip-ft
	My1 = 116.43 kip-ft	Mny = 181.20 kip-ft
Larger End Moment Mx2 = 0.00 kip-ft		
	My2 = 141.23 kip-ft	
=====		

Design Results:	W14x48	E = 29000.00 ksi
Total weight of beam = 0.59 kips		Fy = 36.00 ksi

Load Combination Case For Design: $1.2 \cdot D + 1.6 \cdot L$

Effective Length Factors:

$$K_y = 1.25$$
$$L_y = 12.00 \text{ ft}$$

Nominal Strength:

Pn = 316.85 kips
Mnx = 235.20 kip-ft
Mny = 58.80 kip-ft

Design Results: W12x87 E = 29000.00 ksi
Total weight of beam = 0.87 kips Fy = 36.00 ksi

Load Combination Case For Design: $1.2 \cdot D + 1.6 \cdot L$

Effective Length Factors:

$$K_y = 1.19$$
$$L_y = 10.00 \text{ ft}$$

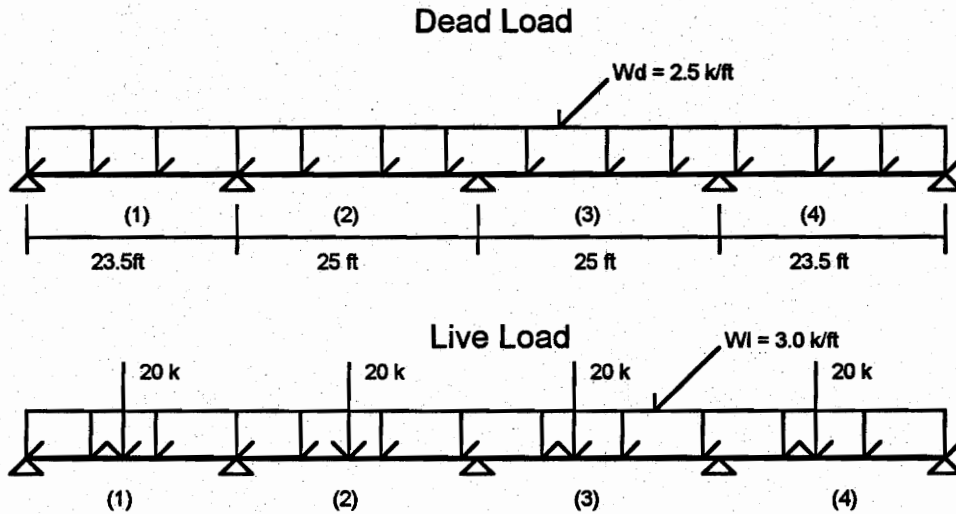
Nominal Strength:

```
Pn = 821.74 kips
Mnx = 396.00 kip-ft
Mny = 181.20 kip-ft
```

Appendix C

Hand Verification

I) Test Structure 1: Continuous Beam



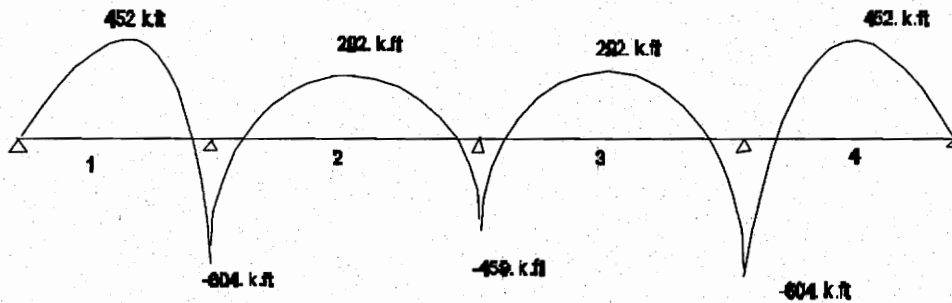
C.1 Loads On Continuous Beam

C.1 shows the structure and applied loads on the continuous beam. Two load combinations are to be considered: (1) $1.4 D$; (2) $1.2D + 1.6L$; By inspection, the second load combination controls. The moment diagram of the continuous beam for the $1.2D+1.6L$ load combination is shown in C.2. The beams are considered only braced at the supports. C_b is equal to 1 for the safer reason.

Only members 1 and 2 need to be designed because of the symmetric loading. Member 3 and 4 are the same as 2 and 1 respectively.

1) Design Member 1:

$$F_y = 36 \text{ ksi}, \quad E = 29000 \text{ ksi}, \quad C_b = 1.0$$



C. 2 Moment Diagram of the Continuous Beam Under 1.2D+1.6L

Unbraced length $L_b = 23.5$ ft, $M_u = 604$ kip-ft

From the Beam Design Moment charts in LRFD Steel Design Manual (Pg3-64),

Try: W 24x104

$b_f / 2t_f = 8.5$, $h_c / t_w = 43.1$, $L_p = 12.1$ ft, $L_r = 35.2$ ft,

$\phi M_p = 780$ kip-ft, $\phi M_r = 503$ kip-ft

Check:

A) Flange Local Buckling:

$$\lambda_p = \frac{65}{\sqrt{F_y}} = 10.8$$

$$\lambda = b_f / 2t_f = 8.5 < \lambda_p$$

Compact Section

B) Web Local Buckling:

$$\lambda_p = \frac{640}{\sqrt{F_y}} = \frac{640}{\sqrt{36}} = 106.7$$

$$\lambda = h_c / t_w = 43.1 < \lambda_p$$

Compact Section

C) Lateral-Torsional Buckling :

$$L_b = 23.5 \text{ ft} > L_p = 12.1 \text{ ft}$$

$$L_b < L_r = 35.2 \text{ ft}$$

$$\begin{aligned}\phi M_n &= C_b \{ \phi M_p - (\phi M_p - \phi M_r) (L_b - L_p) / (L_r - L_p) \} \\ &= (1.0) * \{ 780 - (780 - 503) (23.5 - 12.1) / (35.2 - 12.1) \} \\ &= 643 \text{ kip-ft} > M_u = 604 \text{ kip-ft} \quad \underline{\text{Ok}}\end{aligned}$$

Select W 24x104 A36 Steel for Member 1.

2) Design Member 2 :

Everything is the same as Member 1, except unbraced length $L_b = 25 \text{ ft}$.

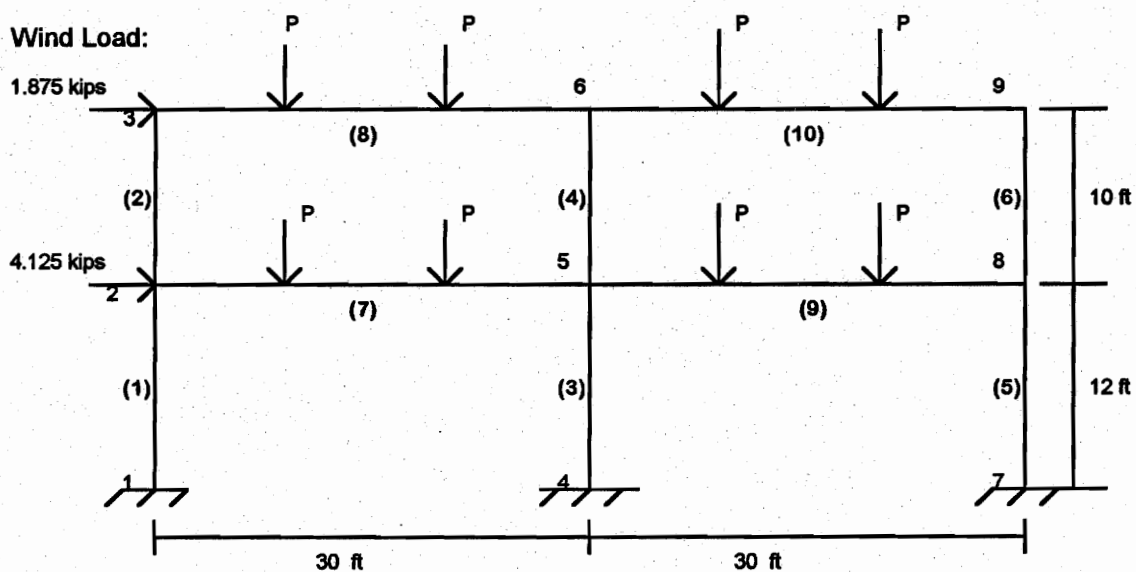
$$M_u = 604 \text{ kip-ft}$$

Try : W 24 x 104

$$\begin{aligned}\phi M_n &= C_b \{ \phi M_p - (\phi M_p - \phi M_r) (L_b - L_p) / (L_r - L_p) \} \\ &= (1.0) * \{ 780 - (780 - 503) (25.0 - 12.1) / (35.2 - 12.1) \} \\ &= 625 \text{ kip-ft} > M_u = 604 \text{ kip-ft} \quad \underline{\text{Ok}}\end{aligned}$$

Select W 24 x 104 A36 Steel for Member 2.

II) Test Structure 2 Two Story Unbraced Plane Frame



Where $P = 23.5 \text{ kips Dead Load} + 7.5 \text{ kips Live Load}$

C. 3 Two Story Plane Frame with Applied loads

In this calculation, three load combinations will be considered : (1) $1.4 D$; (2) $1.2 D + 1.6 L$; (3) $1.2 D + 1.3 W + 0.5 L$. The design results from the program are shown in Table C.1.

Table C.1 : Design Results from the Program

Member Number	Design Results	Member Number	Design Results
1	W 8x31	7	W 18x86
2	W 12x40	8	W 18x86
3	W 8x31	9	W 18x86
4	W 5x19	10	W 18x86
5	W 8x31		
6	W 12x40		

1) Design Columns:

A) Calculate K_x :

To calculate inplane effective length factor K_x for each column, the G value at each joint needs to be calculated first.

$$G = \frac{\sum \frac{I_c}{L_c}}{\sum \frac{I_g}{L_g}}$$

(where Σ indicates a summation of all the members rigidly connected to that joint, I_c is the moment of inertia, and L_c the unsupported length of a column, I_g is the moment of inertia, and L_g unbraced length of a girder). The computed G values are shown in Table C.2.

Table C.2 G Values for Calculating K_x

Joint Number	G	Joint Number	G
1	1.0	6	0.029
2	0.788	7	1.0
3	0.608	8	1.02
4	1.0	9	0.839
5	0.118		

Using these G values, K_x can be found from the alignment chart for effective length of columns in the LRFD Steel Design Manual (Pg. 6-153). The K_x values for each column are shown in Table C.3. K_y is out-of-plane and its default value is set to 1.

Table C.3 Effective Length Factor K_x

Member Number	K_x
1	1.29
2	1.23
3	1.19
4	1.02
5	1.31
6	1.30

B) Calculate Required Moment M_{ux} :

The required moment for beam-column design is obtained by considering the second order moment, which is calculated according to the LRFD manual section H1-2.

$$M_u = B_1 * M_{nt} + B_2 * M_{lt}$$

Table C.4 shows the member axial forces and end moments for dead load. Table C.5 shows the member axial forces and end moments for live load. Table C.6 shows the member axial forces and end moments for wind load.

Table C.4 Member End Forces for Dead Load

Member Number	Axial Force (kips)	I-End Moment (kip-ft)	J-End Moment (kip-ft)
1	39.46	-17.66	-34.69
2	19.14	-69.00	-78.94
3	-104.30	0.00	0.00
4	-53.39	0.00	0.00
5	-39.46	17.66	34.69
6	-19.14	69.00	78.94
7	10.43	103.69	-178.98
8	-14.80	78.94	190.99
9	10.43	178.98	-103.69
10	-14.79	190.99	-78.94

Table C.5 Member End Forces for Live Load

Member Number	Axial Force (kips)	I-End Moment (kip-ft)	J-End Moment (kip-ft)
1	-12.98	-5.72	-11.23
2	-6.29	-22.34	-25.56
3	-34.04	0.00	0.00
4	-17.42	0.00	0.00
5	-12.98	5.72	11.23
6	-6.29	22.34	25.56
7	3.38	33.57	-57.95
8	-4.79	25.56	-61.83
9	3.38	57.95	-33.57
10	-4.79	61.83	-25.56

Table C.6 Member End Forces for Wind Load

Member Number	Axial Force (kips)	I-End Moment (kip-ft)	J-End Moment (kip-ft)
1	0.85	12.84	10.01
2	0.20	0.82	3.24
3	0.01	14.07	12.67
4	0.01	4.63	5.70
5	-0.86	12.60	9.79
6	-0.21	1.00	3.36
7	-2.63	-10.83	-8.66
8	-1.47	-3.24	-2.82
9	-1.43	-8.64	-10.79
10	-0.44	-2.88	-3.36

$$M_{ux} = B_1 * M_{nt} + B_2 * M_{lt} \quad [\text{LRFD H1-2}]$$

The coefficient C_m can be calculated as:

$$C_m = 0.6 - 0.4 * (M_1 / M_2) \quad [\text{LRFD H1-4}]$$

$$B_1 = C_m / (1 - P_u / P_e) \geq 1 \quad [\text{LRFD H1-3}]$$

where $P_e = A_g * F_y / (\lambda_c)^2$

$$\lambda_c = \frac{Kl}{r\pi} \sqrt{\frac{F_y}{E}}$$

M_1 , M_2 are the factored smaller and larger moments at the ends of column in the plane of the bending.

The sway magnifier B_2 is given by the LRFD H1-5 as:

$$B_2 = \frac{1}{1 - \frac{\Sigma P_u}{\Sigma P_e}}$$

or

$$B_2 = \frac{1}{1 - \Sigma P_u \left(\frac{\Delta_{oh}}{\Sigma HL} \right)}$$

Table C.7 shows the calculation of the required flexural strength M_{ux} and factored axial force P_u for load combination 1.2 D+1.6 L. Table C.8 shows the calculation of the required flexural strength M_{ux} and factored axial force P_u for load combination 1.2 D +1.3 W + 0.5 L.

Table C.7 Required Flexural Moment M_{ux} and P_u for 1.2D+1.6L

Member Number	M_1	M_2	P_u	$M_{nt} = 1.2D+1.6L$	C_m	B_1	$M_{ux}=B_1*M_{nt}$ $M_{lt} = 0$
1	-30.65	-60.20	69.56	60.20	0.4	1.0	60.20
2	-119.76	-137.00	33.72	137.00	0.25	1.0	137.00
3	0.00	0.00	182.47	0.00	\	\	0.00
4	0.00	0.00	93.36	0.00	\	\	0.00
5	30.65	60.20	69.56	60.20	0.4	1.0	60.20
6	119.76	137.00	33.72	137.00	0.25	1.0	137.00

Table C.8 Required Flexural Moment M_{ux} and P_u for 1.2D+1.3W+0.5L

Member Number	P_u	B_1	$M_{nt} = 1.2D+0.5L$	B_2	$M_{lt} = 1.3W$	$M_{ux} = B_1M_{nt}+B_2M_{lt}$
1	54.18	1.0	47.24	1.20	16.69	67.27
2	26.54	1.0	107.50	1.07	4.21	112.00
3	145.00	\	0.00	1.20	18.29	21.95
4	74.20	\	0.00	1.07	7.41	7.93
5	56.40	1.0	47.24	1.20	16.38	66.90
6	27.10	1.0	107.50	1.07	4.37	112.20

To design the beam-column, the resistant strength of the column should meet the interaction formulas presented below:

For $P_u/\phi P_n \geq 0.2$

$$\frac{P_u}{\phi P_n} + \frac{8}{9} \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}} \right) \leq 1.0 \quad (H1-1a)$$

For $P_u/\phi P_n \leq 0.2$

$$\frac{P_u}{2\phi P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}} \right) \leq 1.0 \quad (H1-1b)$$

Table C.9 shows the results of the calculations for the columns using above formulas for load combination 1.2D+1.6L. Table C.10 shows the same calculations as Table C.9 but for 1.2D+1.3W+0.5L. The bending term for the y-axis is omitted. The nominal axial strength P_n of the column is calculated according to LRFD Section E2. The nominal flexural strength M_n is calculated according to LRFD Section F1.

Table C.9 Unity Checks for LRFD Beam-Column Interaction Formulas for 1.2D+1.6L

Memeber Number	P_u	ϕP_n	$P_u/\phi P_n$	M_{ux}	$\phi_b M_{nx}$	Ratio of H1-1a, b	Check
1	69.56	214	0.325	60.20	82.10	0.977	<1 ok
2	33.72	295	0.114	137.00	155.00	0.941	<1 ok
3	182.47	214	0.853	0.00	/	0.853	<1 ok
4	93.36	107	0.870	0.00	/	0.870	<1 ok
5	69.56	214	0.325	60.20	82.10	0.977	<1 ok
6	33.72	312	0.108	137.00	155.00	0.941	<1 ok

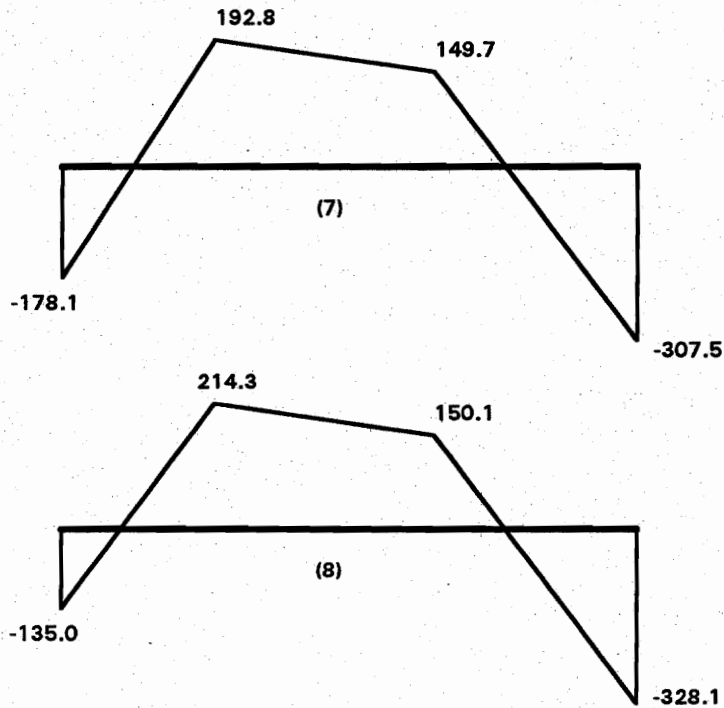
Table C.10 Unity Checks for LRFD Beam-Column Interaction Formulas

for 1.2D+1.3W+0.5L

Memeber Number	P_u	ϕP_n	$P_u/\phi P_n$	M_{ux}	$\phi_b M_{nx}$	Ratio of H1-1a, b	Check
1	54.18	214	0.253	67.27	82.10	0.982	<1 ok
2	26.54	295	0.09	112.00	155.00	0.768	<1 ok
3	145.0	214	0.678	21.95	82.10	0.916	<1 ok
4	74.2	107	0.693	7.93	29.20	0.965	<1 ok
5	56.4	214	0.264	66.90	82.10	0.988	<1 ok
6	27.1	295	0.09	112.20	155.00	0.768	<1 ok

2) Beam Design:

There are four beams in this frame. Because of the symmetry of the structure and gravity loadings. The design results for beams 7 and 8 will be the same as for beams 9 and 10. From the analysis results of the frame using Risa-2D and STAAD-III . The load combination 1.2D+1.6L is the critical case. The moment diagrams of members 7 and 8 for this load combination are shown in C.4. Beams 7 and 8 are verified in this calculation. C_b is equal to 1 for both members.



C. 4 Moment Diagrams for Members 7 and 8 for 1.2D+1.6L

Design Member 7:

$$F_y = 36 \text{ ksi}, \quad E = 29000 \text{ ksi}, \quad C_b = 1.0$$

$$\text{Unbraced Length } L_b = 30.0 \text{ ft}, \quad M_u = 307.5 \text{ kip-ft}$$

From the Beam Design Moment chart in LRFD Steel Design Manual (Pg3-69)

Try: W 18x86

$$b_f/2t_f = 7.2, \quad h_c/t_w = 33.4, \quad L_p = 11.0 \text{ ft}, \quad L_r = 35.5 \text{ ft},$$

$$\phi M_p = 502 \text{ kip-ft}, \quad \phi M_r = 324 \text{ kip-ft}$$

Check:

A) Flange Local Buckling:

$$\lambda_p = \frac{65}{\sqrt{F_y}} = 10.8$$

$$\lambda = b_f/2t_f = 7.2 < \lambda_p$$

Compact Section

B) Web Local Buckling:

$$\lambda_p = \frac{640}{\sqrt{F_y}} = \frac{640}{\sqrt{36}} = 106.7$$

$$\lambda = h_o/t_w = 33.4 < \lambda_p \quad \text{Compact Section}$$

C) Lateral-Torsional Buckling :

$$L_b = 30.0 \text{ ft} > L_p = 11.0 \text{ ft}$$

$$L_b < L_r = 35.5 \text{ ft}$$

$$\begin{aligned}\phi M_n &= C_b \{ \phi M_p - (\phi M_p - \phi M_r) (L_b - L_p) / (L_r - L_p) \} \\ &= (1.0) * \{ 502 - (502 - 324) (30.0 - 11.0) / (35.5 - 11.0) \} \\ &= 364 \text{ kip-ft} > M_u = 307.5 \text{ kip-ft} \quad \text{Ok}\end{aligned}$$

Select W 18x86 A36 Steel For Members 7 and 9.

Design Member 8:

$$F_y = 36 \text{ ksi}, \quad E = 29000 \text{ ksi}, \quad C_b = 1.0$$

$$\text{unbraced length } L_b = 30.0 \text{ ft}, \quad M_u = 328.1 \text{ kip-ft}$$

From the Beam Design Moment chart in LRFD Design Manual (Pg3-69),

Try: W 18x86

All the calculations are the same as member 7.

$$\phi M_n = 364 \text{ kip-ft} > M_u = 328.1 \text{ kip-ft}$$

Select W 18x86 A36 Steel For Members 8 and 10.