Structural Analysis of Plane Frames
in an Interactive XWindow Environment

by

Mehmet Erkan Erkek

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

Master of Science
in
Civil Engineering

APPROVED

Kamal B. Rojiani, Chairman
Siegfried M. Holzer
W. Samuel Easterling

April 1992
Blacksburg, Virginia
Structural Analysis of Plane Frames
in an Interactive XWindow Environment
by
Mehmet Erkan Erkek
Kamal B. Rojiani, Chairman
Civil Engineering
(ABSTRACT)

A study was performed to develop an interactive plane frame analysis program. The program was designed to run on Unix-based workstations and to be independent of the hardware platform. This was achieved by selecting the XWindow system as the underlying graphical user-interface. An event-driven, window-based program was developed using the XWindow Toolkit. The entire program was written in the C programming language.

The program consists of two parts: a preprocessor and a processor. The preprocessor employs pull-down menus and pop-up dialog boxes and provides a convenient way of creating and modifying structural models. The processor performs linear elastic analysis of plane frame structures based on the matrix displacement method. The processor computes joint displacements, support reactions and local member-end forces of the structure. It also handles internal hinges, joint loads, member loads, prescribed displacements, multiple load cases and load combinations. Other features include automatic mesh generation for orthogonal frames, automatic computation of self-weight and specification of output options.
To verify the accuracy of the program, analysis results from the program were compared with results obtained from a commercial structural analysis program. A comparison of joint displacements, support reactions and member-end forces indicate that the difference in the values obtained from the two programs is less than two percent and that the program provides accurate analysis results.
ACKNOWLEDGEMENTS

I would like to thank Dr. Kamal Rojiani for his continuous guidance and support throughout my education at Virginia Tech. He not only significantly contributed to the quality of the thesis, but also broadened my view by offering me the assistantship at the CAE Lab and introducing me to new technologies.

I would like to thank Dr. Sam Easterling for serving on my committee and providing guidance since my first day at Virginia Tech. I would also like to thank Dr. Siegfried Holzer for his constructive comments and for serving on my committee. Appreciation is extended to Dr. Thomas Murray for offering me a teaching assistantship for my first semester at Tech which made everything possible.

There are many friends who provided encouragement and laughs in the last two years, especially the Turkish Gang, Taylan, Tunc, Suha, Evren and Kagan.Thanks, Guys.

I would like to thank my sister Tijen for being so good and understanding to me all these years. Special thanks go to Pam Hopkins for her inspiration and support during last year. Finally, I would like to dedicate this thesis to my parents Meral and Cevat Erkek for their love and encouragement throughout my life. I sincerely wish to be as good a parent to my kids as they have been to me.
# TABLE OF CONTENTS

1. Introduction
   1.1 Background 1
   1.2 Objectives 4
   1.3 Scope & Limitations 6
   1.4 Organization 7

2. Overview Of Computing Environment
   2.1 Introduction 8
   2.2 Engineering Workstations 8
   2.3 Unix Operating System 12
   2.4 The C Programming Language 14
   2.5 The XWindow System 15
      2.5.1 The Server and Client 16
      2.5.2 Event-driven Programming 17
      2.5.3 Input Devices 17
      2.5.4 X Toolkit 17
      2.5.5 Toolkit Application 18
      2.5.6 Callback Procedures 20
      2.5.7 Widget Classes 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.8 Resources</td>
<td>26</td>
</tr>
<tr>
<td>3. Preprocessor</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Characteristics</td>
<td>27</td>
</tr>
<tr>
<td>3.3 Capabilities</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Modification Of Existing Data</td>
<td>30</td>
</tr>
<tr>
<td>3.5 User-Interface</td>
<td>31</td>
</tr>
<tr>
<td>3.5.1 Event-Driven Structure</td>
<td>31</td>
</tr>
<tr>
<td>3.5.2 Input Techniques</td>
<td>32</td>
</tr>
<tr>
<td>3.5.3 Main Window</td>
<td>32</td>
</tr>
<tr>
<td>3.5.4 Pop-up Windows</td>
<td>34</td>
</tr>
<tr>
<td>3.6 Functions Of The Preprocessor</td>
<td>36</td>
</tr>
<tr>
<td>3.6.1 The File Option</td>
<td>36</td>
</tr>
<tr>
<td>3.6.2 Control Variables</td>
<td>39</td>
</tr>
<tr>
<td>3.6.3 Joint Coordinates</td>
<td>40</td>
</tr>
<tr>
<td>3.6.4 Joint Constraints</td>
<td>41</td>
</tr>
<tr>
<td>3.6.5 Member Incidences</td>
<td>42</td>
</tr>
<tr>
<td>3.6.6 Internal Hinges</td>
<td>42</td>
</tr>
<tr>
<td>3.6.7 Material Properties</td>
<td>43</td>
</tr>
<tr>
<td>3.6.8 Member Properties</td>
<td>45</td>
</tr>
<tr>
<td>3.6.9 Loads</td>
<td>47</td>
</tr>
<tr>
<td>3.6.10 Load Descriptions</td>
<td>50</td>
</tr>
<tr>
<td>3.6.11 Load Combinations</td>
<td>51</td>
</tr>
<tr>
<td>3.6.12 Output Options</td>
<td>51</td>
</tr>
<tr>
<td>3.6.13 Quit</td>
<td>52</td>
</tr>
</tbody>
</table>
4. Processor

4.1 Overview

4.2 Plane Frames

4.3 Matrix Displacement Method

4.4 Structure And Features Of The Processor

4.5 Modifications To Holzer's Program

5. Program Verification

5.1 Overview

5.2 Example Structure 1

5.2.1 Description

5.2.2 Input Data

5.2.3 Comparison Of Results

5.3 Example Structure 2

5.3.1 Description

5.3.2 Input Data

5.3.3 Comparison Of Results

5.4 Example Structure 3

5.4.1 Description

5.4.2 Input Data

5.4.3 Comparison Of Results

5.5 Example Structure 4

5.5.1 Description

5.5.2 Input Data
5.5.3 Comparison Of Results 94
5.6 Example Structure 5 97
  5.6.1 Description 97
  5.6.2 Input Data 97
  5.6.3 Comparison Of Results 101
5.7 Conclusions 104

6. Summary And Future Extensions 105
  6.1 Summary 105
  6.2 Future Extensions 106

References Cited 108

References Consulted 111

Appendix A 114

Vita 125
LIST OF FIGURES

Figure 2.1 Layers of an X Toolkit application 19
Figure 2.2 Class hierarchy of Athena widgets used in the preprocessor 21
Figure 2.3 A sample application's interface and widget tree 25
Figure 3.1 Main window of the preprocessor 33
Figure 3.2 Pop-up window for entering joint coordinates 35
Figure 3.3 Automatic mesh generation 38
Figure 3.4 Member load types 49
Figure 4.1 Plane frame member 58
Figure 4.2 Local coordinate system 61
Figure 4.3 Global coordinate system 62
Figure 4.4 Structure of the processor 64
Figure 5.1 Example structure 1 71
Figure 5.2 Example structure 2 76
Figure 5.3 Example structure 3 84
Figure 5.4 Example structure 4 91
Figure 5.5 Example structure 5 98
LIST OF TABLES

Table 3.1 List of the types of joint constraints ........................................... 41
Table 5.1 Support reactions for example structure 1 .................................... 74
Table 5.2 Joint displacements for example structure 1 ................................ 74
Table 5.3 Local member end forces for example structure 1 .......................... 75
Table 5.4 Support reactions for example structure 2 .................................... 81
Table 5.5 Joint displacements for example structure 2 ................................ 82
Table 5.6 Local member end forces for example structure 2 ......................... 83
Table 5.7 Support reactions for example structure 3 .................................... 88
Table 5.8 Joint displacements for example structure 3 ................................ 89
Table 5.9 Local member end forces for example structure 3 ......................... 90
Table 5.10 Support reactions for example structure 4 .................................. 95
Table 5.11 Joint displacements for example structure 4 ............................... 95
Table 5.12 Local member end forces for example structure 4 ....................... 96
Table 5.13 Support reactions for example structure 5 .................................. 102
Table 5.14 Joint displacements for example structure 5 ............................... 102
Table 5.15 Local member end forces for example structure 5 ....................... 103
CHAPTER 1

INTRODUCTION

1.1 Background

The purpose of structural analysis is to determine the behavior of a structure under design loads in order to judge the adequacy of the design (Jenkins 1990). The magnitude of the forces, stresses and displacements in the structure are obtained from the analysis, and these can be examined for acceptability.

Computers have been used in structural analysis since the mid-1950s. Since that time, there have been tremendous developments in computer hardware, programming languages and the methods used for structural analysis. The widespread availability of microcomputers and workstations and the development of matrix methods for structural analysis have allowed structural engineers to perform complex analyses on computers in an efficient and cost effective manner.
When performing structural analysis on computers, a large amount of information has to be entered into the computer in order to describe the geometry of the structure, the material properties and the applied loads. This is usually a fairly time-consuming task. It is also one that is susceptible to user errors. For this reason, there is considerable interest in better and improved methods for the entry of structural data and the modification of previously entered data to a structural analysis program. Most structural analysis programs have a character-based user interface. In this environment, input to the program is through a series of text commands that are typically entered on a data file. Modifications to the input data are performed by manually editing this data file. Besides being cumbersome, character-based user-interfaces have limited usefulness for structural analysis and design since they provide little or no visual feedback to the user thus making it extremely difficult to verify the accuracy of the structural data entered in the program. Hence, there is a great need for a user-friendly, interactive and flexible program with a graphical interface and automatic mesh generation capabilities.

The significant advances in the computational and graphical capabilities of workstations have made them suitable for use in structural analysis and design. These capabilities have made the development of interactive and user-friendly software possible. Graphical windowing interfaces such as XWindow have now become the accepted way of interacting with the user on workstations (Pountain 1989). An important characteristic of such interfaces is that they are event-driven rather than modal.

A modal interface consists of modes that last for a period of time and have no role other than to place an interpretation on user input (Tesler 1981). In a modal
interface a given action can have very different effects depending on the mode of the program.

One problem with the traditional modal user-interface developed through procedural programming is that it is the application that is in control rather than the user. The user interactions and the order in which the work progresses are for the most part, dictated by the application. Typically, the application performs some action based on user input and then requests additional input. Also, in this type of environment, communication between the user and the application is primarily through the keyboard.

Event-driven applications are radically different. In an event-driven model, the user is in control of the application. The application is developed so that it is always ready to respond to events invoked by the user. Different functions are called within the application to respond to various events generated by the user.

Event-driven programming allows flexibility and power. The various tasks performed by the program can be isolated since events work independently of each other. This results in a program that is highly structured and simplifies the maintenance and modification of programs.

Most event-driven applications have a window-based user-interface. Modes are eliminated by using separate windows for different aspects of the program. Also, the user can use a pointing device such as a mouse to select from a menu or command buttons and scroll bars.
Window-based applications are simple to use but very complex to develop. The main reason for their complexity is that the application has to be able to respond to the events invoked by the user at any time. Young states "some experts estimate that as much as 90 percent of the total effort required to develop a typical window and mouse-based application goes into the user-interface" (Young 1989).

A dominant windowing environment on workstations is the XWindow system developed at Massachusetts Institute of Technology (MIT) in 1985. XWindow is a network-transparent, hardware-independent, multitasking windowing and graphics system. XWindow is the industry standard for graphics software because of its device-independent architecture.

The XWindow Toolkit allows programmers to develop event-driven window-based graphics applications that run under XWindow. The X Toolkit simplifies the development of applications and provides consistent user-interfaces. Asente states that "the user interface code for a toolkit application is typically about one-fifth as large as it would be in an equivalent program that used the lower-level X library directly" (Asente and Swick 1990).

1.2 Objectives

The main objective of this study is to develop a user-friendly plane frame analysis program on workstations using the XWindow system. The program consists of two parts; a preprocessor and a processor. The preprocessor accepts user input

Introduction 4
and commands and present a convenient way to input information for a structural model and to save this in a data file. The preprocessor should also have the capability to modify data for existing structural models. The processor should be capable of reading the structural model created by the preprocessor and performing linear elastic analysis on the structure.

The preprocessor should have an interactive, user-friendly interface employing pull-down menus and pop-up dialog boxes and should have an event-driven structure. The preprocessor should be able to accept input from a pointing device such as a mouse, or the keyboard. It should also have an error checking mechanism and on-line help to create error-free data files.

Another objective of this study is to provide a teaching tool to be used in structural analysis and design courses. The program would be a useful tool for introducing students to structural analysis and design in the workstation environment.

To achieve these objectives, the XWindow Toolkit is used to develop the user-interface for the preprocessor. Since the XWindow System is written in the C programming language, C is the language of choice for developing XWindow applications. Thus, both the preprocessor and the processor are written in C.
1.3 Scope & Limitations

The program is developed for the analysis of plane frames. Plane frames are two-dimensional structures consisting of straight members connected at the joints. Wang states "the word plane means that the applied loads and the structure itself all lie in the same plane" (Wang 1986).

The program analyzes structures with prismatic members only. Different member and material properties can be assigned for each member of the structure. Internal hinges can be specified on the structures. Only linear static analysis is considered. The program permits joint loads, member loads, prescribed joint displacements, multiple load cases and load combinations. Analysis for the self-weight of the structures can be incorporated automatically. The program does not have any postprocessing capabilities.

A limitation of the program's preprocessor is that it has no graphical capabilities. Automatic mesh generation is available for orthogonal plane frames. Also, the preprocessor allows for the modification of existing data and the specification of output options.

The processor is based on a frame analysis program written in FORTRAN (Holzer 1985). The entire program is rewritten in C and some features are added. These will be discussed in more detail in Section 4.5. Also a number of modifications are made to the program so that it would accept the input data created by the preprocessor.
1.4 Organization

Chapter 2 contains a description of the computing environment for which the preprocessor was developed. It presents background information on workstations, the Unix operating system, the C programming language and the XWindow system.

Chapter 3 describes the preprocessor’s characteristics and capabilities, as well as user-interface techniques and features.

Chapter 4 presents the features and limitations of the processor. The processor is modeled after a frame analysis program written in FORTRAN (Holzer 1985). The modifications made to this program are also described.

Chapter 5 describes the verification study performed to verify the accuracy of the program. The input information and results of the analyses for five example structures are presented.

Chapter 6 presents the conclusions and describes possible extensions to the program.

Appendix A presents the results obtained by hand calculations for example structures 1 through 4.
CHAPTER 2

OVERVIEW OF COMPUTING ENVIRONMENT

2.1 Introduction

This chapter describes the computing environment for which the preprocessor and processor were developed. It provides details of the computer hardware, software, programming language and operating system. More specifically, it presents background material on workstations, the Unix operating system, the C programming language and the XWindow system. It also highlights some of their advantages.

2.2 Engineering Workstations

A workstation can be defined as an interactive graphics system intended to be used by scientists and engineers in the analysis, design and manufacturing of products (Harlow 1986). The main characteristics of workstations are: friendly
user-interface, high resolution graphics display on a large screen, networking capability, fast processing without observable response time and large memory and external storage capacity.

The workstation environment has three essential components: the user, the machine and the interaction between the two (Bowerman and Fertig 1988). This is in contrast to earlier developments which focused only on the machine, with little or no consideration to the user and the process of interaction (Bowerman and Fertig 1988). In the early days, computers were used almost exclusively for data processing, which for the most part meant just alphanumeric data. Today's workstations, however, can handle a much more varied range of tasks which require diverse graphics. Thus, a high-resolution graphics display is fundamental for an engineering workstation. High resolution is considered as meaning close to one million picture elements (pixels) on the screen. Because today’s workstation supports a wider range of symbols than in the past, its user-interface is capable of supporting a wider range of applications.

Workstations have friendly graphical user-interfaces that employ windows, icons, pop-up menus and a pointing device such as a mouse. Windows have been a standard feature on workstations since 1980 (Bowerman and Fertig 1988). Powerful windowing systems permit the user to work on various independent tasks simultaneously. These tasks include analysis, design, computation, communication, report writing, data analysis and management and cost analysis (Wright 1988). A more efficient way of moving between activities is achieved by the use of this modern windowing technology. Furthermore, a pointing device such as a mouse allows the keyboard to be used only for text entry. All of these
features can increase productivity.

The need to drive the sophisticated user-interface, to handle a symbolic display and to respond quickly all demand a high-speed central processing unit (CPU) (Bowerman and Fertig 1988). Also, a high-speed parallel bus connection between the CPU and the display is needed. A second CPU called a graphics processor is also utilized to handle the display.

Networking is another significant feature of workstations. Networking enables engineers to work on subdivisions of a project by interacting with each other. Networking allows the sharing and exchanging of data and programs. Electronic mail and telecommunications facilities allow users of a network to work together regardless of location or time (Harlow 1986).

A study by IBM Zurich in 1983 evaluated the advantage of faster response times. The study showed that engineering productivity can be raised five or six times just by speeding up the response time three or four times. Furthermore, it was shown that an average engineer using a machine with a 1/4 second response time could perform at the same speed as an expert engineer using a machine with a response time of one second (Bowerman and Fertig 1988). The conclusion reached from this study is that a faster response time can raise productivity. In this study, workstations proved to be more productive than host-based terminals.

Workstations frequently have a response time of below 100 milliseconds, while even the best mainframe terminals can seldom do better than a one-second response time (Bowerman and Fertig 1988). Frequently, the response time can
slip to minutes as more users log. Removing delays and interruptions decreases the error rate significantly, as the computer reacts to the user's every move almost simultaneously.

Another advantage of workstations over host-based terminals is that they require no unusual environmental controls. Thus, they can be situated on the user's desk in a comfortable office setting. Furthermore, workstations are easier to use than host-based terminals because of their extensive graphics capabilities which allow interactive programming.

Personal computers lack the graphics and networking capabilities of workstations. Personal computers do not have the memory and storage capacity, graphics processing, graphics resolution, screen size or networking capabilities of workstations. Workstations are also faster and more accurate than personal computers because of the difference in their architecture. Bowerman and Fertig (1988) state that "accuracy and execution speeds for finite element analysis programs (FEA) on the PC are over an order of magnitude below that of the standard FEA on workstations." However, Wright (1987) states that "as the performance of personal computers (PCs) increases and the cost of professional workstations decreases, the distinction between the two classes of computers becomes blurred".

The power and user-friendliness of workstations have increased the burden on developers of application software. Users are demanding interactive software that takes full advantage of the graphical capabilities of workstations. However, these developers are still faced with the problem and expense of developing a different
graphical interface for each workstation hardware. The XWindow system grew out of this need to develop a device-independent graphics system.

2.3 Unix Operating System

Kochan and Wood (1990) state that "an operating system is a collection of programs that coordinates the operation of computer hardware and software". Unix, developed by Ken Thompson at Bell Telephone Laboratories in 1969, is the dominant operating system on workstations. The primary goal in the design of Unix was to create a small and memory-efficient operating system that supports efficient program development (Kochan and Wood 1990).

Unix is a multiuser, multitasking operating system. Multiuser means that several users can work on the computer at the same time; multitasking means that even a single user can work on more than one task or program at the same time (Prata 1985).

The portability of Unix was achieved by developing the operating system without making too many assumptions about specific computer architecture and also by writing the operating system in the high-level programming language C instead of assembly language (Kochan and Wood 1990). Thus, programs can run without modification on any computer running Unix.

C is the programming language of choice for a Unix-based operating system, for two reasons. First, Unix is written in C. Second, the Unix system calls and
library functions can be used as C functions in C programs (Prata 1985). Thus, the interface between C and Unix is straightforward and natural.

The Unix operating system provides the following tools (Kochan and Wood 1990):
1. **Input/output**: allows data storage and retrieval; interacts with terminals; prints output on paper.
2. **Command interpreter**: (the interface between the user and system); reads commands entered by the user and changes them into instructions the computer can understand.
3. **Data management**: allows the user to organize data into files. Unix has a structured file system with multiple levels.
4. **Program development tools**: supports development and maintenance of programs.
5. **Time-sharing**: allows many users to work from the same computer at the same time.
6. **Security**: keeps unauthorized people from gaining access to a computer system; protects the users from each other and the operating system from all users.
7. **Communication**: enables a computer to send mail, programs or data to other computers.
8. **Accounting**: charges the users for the resources used.

These characteristics make Unix very suitable for engineering purposes and useful in an office environment.
2.4 The C Programming Language

The C programming language was developed around 1972 by Dennis Ritchie, a systems engineer at Bell Telephone Laboratories. C was originally designed as a systems programming language to be used with the Unix operating system. However, it has become a powerful general purpose programming language (Rojiani 1992). C combines the features of high-level languages such as FORTRAN, BASIC and Pascal with the functionalism of assembly language (Schildt 1990). Thus, it is often called a middle-level computer language.

There are several advantages in using C as opposed to other languages. The first advantage of programming in C is its efficiency. Efficiency refers to the speed of execution and the use of system resources. Programs written in C run faster and take less storage space and memory than other high-level languages. Second, C is independent of any particular computer architecture, so it is easy to write portable programs with C (Kernighan and Ritchie 1978). Third, C is a flexible language. C allows manipulation of bits, bytes and addresses. This makes C well-suited for system-level programming (Schildt 1990). C is not a strongly-typed language like Pascal or Algol68. It is relatively permissive about data conversions. Fourth, C is a structured language. Structured programming is a discipline intended to make programs easier to read, write and maintain. C provides the fundamental flow-control structures required for structured programming. Fifth and lastly, C is a modular language. C's main structural component is the function. A C function is a self-contained block of code that performs a coherent task independent of other parts of the program (Hancock and Krüger 1986). Functions can be written and tested separately allowing C programs to be
modular.

The C language has a few disadvantages. Because C performs no run-time error checking, the programmer is responsible for these types of checks. C allows direct access to memory by using pointers; however this flexibility can cause errors in programs if used incorrectly (Rojiani 1992).

2.5 The XWindow System

The XWindow system (referred hereafter as X) was developed in 1984 as a result of an MIT project called Athena, which examined the use of networked workstations as a teaching aid for students (Pountain 1989). X is adopted by the computer industry as a standard for graphics software. The purpose of X is to provide a network-transparent, hardware-independent, multitasking windowing and graphics system.

X is the industry standard for user-interfaces because of its device-independent architecture. This portability of X allows graphics applications to run on various environments like workstations, mainframes and personal computers. X provides communication between applications and the sharing and exchanging of data. These features make X an ideal system for networking environments. X is a multitasking system, as it allows various applications to run at the same time.

These capabilities of X require a tremendous amount of hard disk space and random access memory. Even the smallest program uses over half a megabyte of
disk space. The philosophy behind X is that the hardware will respond to these demands in a short time (Brain 1991).

Both Xlib, the standard interface to X, and the higher level Xt Intrinsics and widget sets are written in C. Furthermore, event types in X are defined as C structures. Thus, C is the expected language for XWindow applications.

2.5.1 The Server and Client

X allows applications to run on any hardware that supports the X protocol without modifying, recompiling, or relinking the application (Young 1989). This device-independent architecture is based on a client-server model.

In the client-server model, the server controls the user-interface. The server is responsible for creating and manipulating windows, drawing text and graphics and handling input devices such as a keyboard and mouse. Thus, the server provides a portable layer between the application and the hardware (Young 1989).

The client is an application program that requests the facilities provided by the X server. The client communicates with the server through a network connection. Multiple clients can connect to a single server at the same time. A single client can also connect to multiple servers.

The client/server architecture hides the details of the workstation's display hardware from the client and allows the server and client to run on different
machines in the network (Young 1989).

2.5.2 Event-driven Programming

XWindow applications are event-driven. The server communicates with the client application by sending events. The application waits until an event occurs, responds to the event and waits for the next event. Pointer motion, pointer button press and release, key press, window entry and exit, and exposure of previously covered windows are some typical event types. This event-driven model supports the development of interactive applications.

2.5.3 Input Devices

X takes user input from both the pointer and the keyboard. The keyboard is used for textual input. The pointer allows the user to point at and select objects drawn on the screen like menus and command buttons.

2.5.4 X Toolkit

To simplify and standardize the design of a graphical user-interface, the user-interface elements of an application such as command buttons, labels, dialog boxes, scrollbars should be available (Nye and O'Reilly 1990). The purpose of the X Toolkit is to simplify the development of user-interfaces so that the programmer can spend more time on the application itself and less time on the user-interface. The user-interface code for a toolkit application is about one-fifth of an equivalent program that uses the lower level X library directly (Asente and
Swick 1990).

The X Toolkit consists of two parts. The first part is a layer known as the Xt Intrinsics. The second part is a set of user-interface components called widgets. The Xt Intrinsics supports many different widget sets. The Athena widget set developed by MIT’s Project Athena is used in this application.

The Athena widget set implements user-interface components including command buttons, dialog boxes, menus, boxes, viewports, while the Xt Intrinsics creates and destroys the widgets, assembles widgets into a user-interface and sends input events to the correct widgets. The widgets in turn take these input events and call the appropriate callback procedures which cause some action to take place in the application.

2.5.5 Toolkit Application

A typical X Toolkit application consists of four layers as shown in Figure 2.1. At the lowest level is the X library which is directly connected to the X server. The next layer is the X Toolkit Intrinsics library which supports widgets and callback procedures. Callback procedures notify the application that the user has interacted with a widget (Asente and Swick 1990). The third layer consists of widgets. The top layer is the application code itself. Widgets are reusable, configurable pieces of code that combine a window with a set of procedures that perform actions on that window.
There are six steps in developing a widget-based application:

1. **Choose the right classes for the widgets.** The widget classes are arranged into a hierarchical tree structure and all widget classes inherit something from their parents. Thus, it is important to choose the right class.

2. **Initialize the Intrinsics.** This step sets up a connection to the X server and creates a top-level widget which becomes the parent of all other widgets in the application (Johnson and Reichard 1989).

3. **Create subwidgets.** Each widget should have a unique name, a class, a parent and some resources. The widgets construct the application’s user-interface.

4. **Register callbacks and event-handlers.** Callbacks and event-handlers respond to user actions within each window.

5. **Realize all widgets.** Realizing a widget causes its window to be mapped to the screen.
6. **Enter the event loop.** The application loops indefinitely responding to the events. The event loop retrieves the events and invokes the callback procedures which connect the application functions with the user-interface (Young 1989).

2.5.6 Callback Procedures

Callback procedures connect the user-interface with the application functions, for example, by selecting a command button or moving a scrollbar. All callback procedures take three parameters: the widget that initiated the callback, the function the widget is to call and any data to be passed to the callback function (Asente and Swick 1990). Callback procedures are single-function, independent routines that support modular programming.

2.5.7 Widget Classes

The widget classes are arranged in a hierarchical tree structure called the class hierarchy (Asente and Swick 1990). The class hierarchy defines the relationships between widget classes. A widget that belongs to one class belongs to all classes higher up in the hierarchy. Arranging classes into a hierarchy makes it easy to create widget classes since a widget class inherits all the attributes of its superclasses.

The class hierarchy is a static property of the widget set and is different from the instance tree which shows the organization of the widget instances in a particular application (Asente and Swick 1990). Figure 2.2 shows the class hierarchy of the Athena widgets used in the user-interface of the preprocessor.
Figure 2.2 Class hierarchy of Athena widgets used in the preprocessor
The core widget is the fundamental class in the Toolkit. All other widgets are subclasses of the core. Composite widgets usually do not display any information themselves but instead serve as a container for a group of widgets. Thus the composite class adds geometry-management capabilities to the basic characteristics of the core widget class (Nye and O'Reilly 1990).

The box widget is from the composite class and provides geometry management of its children in a box of a specified dimension. The box widget can be resized, but it does not resize its children. The box widget always tries to pack its children as closely as possible within the geometry allowed by its parent. Box widgets are generally used to manage a related set of similarly-sized command widgets (O'Reilly 1990).

The constraint widget class allows the application or the user to supply information about the size and position of each child (Nye and O'Reilly 1990). Thus, a constraint widget has more information about its children than a composite widget.

The viewport widget consists of a framed window, one or two scrollbars and an inner window (O'Reilly 1990). The size of the frame window is determined by the dimensions of the viewport. The inner window is the full size of the data and is cut by the frame window. The viewport widget manages the scrolling of data directly so no callback procedures are invoked for scrolling. The viewport widget automatically removes the scrollbars if the data is small enough to fit into the frame window. The scrollbar widget is a rectangular area that consists of a slide region and a thumb. The thumb shows the position of the visible data within a
large data buffer and allows the user to change the current position in the buffer by dragging the thumb with the mouse (O'Reilly 1990).

The form widget provides geometry management for its children. When the form widget is resized, it rearranges the positions and sizes of its children. The difference between the form widget and the box is that the form widget can specify the initial position of its children relative to the positions of other children.

The dialog widget is commonly used as a prompt for input from the user. A typical dialog widget contains three areas. The first line contains a message. The second line contains an area for data entry. The third line contains command buttons that allow the user to confirm or cancel the action of the dialog box. Any of these areas may be omitted by the application.

Shell widgets encapsulate other widgets. Thus, shell widgets are invisible to the user. The shell widget has the same size as the child widget that it displays. Each independent window of an application requires a shell widget.

The toplevel shell is used for the main window of the application. The toplevel shell widget reacts to the user's requests like moving or resizing the main window of the application. The main window of an application can be iconified while the other windows cannot be iconified separately.

Many applications create temporary windows like dialog boxes, confirmation boxes, pop-up menus and warning messages. These pop-up windows use the transientshell widget class. Pop-ups are usually invisible when the application
starts up. The pop-up suspends the operation of the rest of the application while it is on the screen by making the other widgets insensitive to user input. Pop-down callbacks reset the changes to the user-interface done by the pop-up callbacks.

Simple widgets have no children. Commands, labels and lists are typical simple widgets. The label widget displays a text string on the screen that may be more than one line long. A label widget is for output only; it cannot be directly edited or selected by the user.

The command widget invokes a callback procedure when activated. The command widget is a rectangle that contains text. When the pointer cursor is on the command button, the button border is highlighted to show that the button is ready for selection. When the left mouse button is pressed, the command button indicates that it has been selected by reversing its foreground and background colors. When the left mouse button is released, the functions on the callback list are invoked.

The list widget is a rectangle that contains a list of strings formatted into rows and columns (O’Reilly 1990). When a string is selected, it is highlighted and a callback procedure is activated. Strings are selected one at a time and the selected string can be passed to an application function. Figure 2.3 shows a sample application’s interface and its widget instance tree.
Figure 2.3 A sample application's interface and widget tree
2.5.8 Resources

Widgets should be highly configurable to serve the purpose of reusable user-interface components (Nye and O'Reilly 1990). Attributes of a widget such as position and size of its window, label, font and color should be user-defined. To support this configurability, widget classes can declare some specific features as resources of the widget. A widget class defines its own resources and also inherits the resources of its superclasses.

The initial resources of a user-interface can be hardcoded in the application program or specified in an application defaults file.

The user can easily change the resources stored in the applications defaults file. This provides users some flexibility. For example, an application can be translated into another language by changing only the text in the applications defaults file without modifying or recompiling the source code.

If the resource of a widget is not specified or incorrectly specified in the application program or the applications default file, the toolkit takes the default value of the resource.
CHAPTER 3

PREPROCESSOR

3.1 Overview

In this chapter, a description of the preprocessor is presented. First the characteristics and capabilities of the preprocessor are discussed. Then, the details of the user-interface and the features for modification of existing data are presented. Finally, the different parts of the preprocessor and the various input parameters required for creating a computer model of a structure are described.

3.2 Characteristics

The preprocessor is an interactive menu and mouse-driven program with pop-up dialog boxes and on-line help. The preprocessor provides a user-friendly method of generating structural models and modifying data contained in existing input files. The input files created by the preprocessor are used by the frame analysis
program for performing the structural analysis.

The preprocessor has the following characteristics:

1. **Flexibility**: the user can follow any logical path during the generation of the computer model of the structure.

2. **User-friendly data entry**: this is achieved by employing a menu and mouse-driven user-interface with pop-up dialog boxes.

3. **On-line help**: on-line help is available in the form of pop-up windows which explain the different options that are available and the required input parameters.

4. **Error checking mechanism**: pop-up windows containing error messages are used to inform the user about errors that occurred during the definition of the structure. Error recovery is permitted at this point.

5. **Portability**: the preprocessor can be used on different hardware platforms due to the machine and operating system independent characteristics of the XWindow system.

6. **Expandability**: the preprocessor can be modified and new options and features can be added without any difficulty. The program is written in C and consists of a series of modules that can easily be modified or even replaced.

### 3.3 Capabilities

The capabilities of the preprocessor are as follows:

1. **Modification of existing data**: the preprocessor is designed to allow the modification of existing data very efficiently. This will be discussed in more detail in Section 3.4.
2. **Automatic mesh generation**: automatic mesh generation can be utilized for orthogonal plane frames. This will be discussed in more detail in Section 3.6.1.

3. **Command duplication**: the preprocessor has the capability to assign the same material and member properties to multiple members with a minimal amount of effort on the part of the user.

4. **Material & section data files**: data files containing material data and section properties of standard steel sections taken from the AISC Steel Manual (AISC 1986) are used for selecting the type of material and steel sections for members of the structure.

5. **Multiple load cases and load combinations**: the preprocessor provides support for entering several load cases and for combining these load cases with different multipliers to generate loading combinations. These will be discussed in more detail in Sections 3.6.9 and 3.6.11.

6. **Automatic computation of self-weight**: if the self-weight of the structure is to be considered as a load case, the preprocessor computes the member loads due to self-weight automatically.

7. **Different load types**: loads can be specified as joint loads, member loads and prescribed displacements. Member loads include concentrated loads, axial loads, uniform loads, uniform axial loads, linearly varying loads and uniform temperature change.

8. **Output options**: output options for the output file created by the frame analysis program can be specified in the preprocessor.

An important limitation of the preprocessor is that it does not have any graphical capabilities. Plotting the structure and labeling the joint and member numbers would have been very helpful in creating error-free structural models. This could
not be achieved because of compatibility problems between Xlib and X Toolkit. X Toolkit does not provide any graphics functions and the Athena widget set does not have a widget that accepts Xlib drawing functions directly. The Xlib functions are not incorporated into the program because of the difference in the attributes of the widgets and the Xlib functions.

3.4 Modification Of Existing Data

The preprocessor is capable of editing input parameters without having to enter a separate editor mode. Thus, input correction is permitted at any point in the program.

Modification of input parameters is performed in two steps. First, the existing input parameters are deleted when the delete option is selected. Then, the new parameters are entered. This procedure is more efficient and easier to use than changing to an editing mode to modify existing input data.

This simplicity is achieved by storing all data in memory as arrays instead of writing the data directly to the data file. The preprocessor sorts through the arrays to delete and modify existing data as needed prior to actually writing the data to the file. The data stored in the arrays is written to the input file when the quit option from the main menu is selected. The program also performs error checking prior to actually writing the data to the file. Error checking is discussed in more detail in Section 3.6.13.
3.5 User-Interface

3.5.1 Event-Driven Structure

The user has complete control over the program, and the preprocessor does not dictate the sequence of user actions or the order in which input information is entered. This is achieved by the event-driven feature of the XWindow system. The application is written so that it can respond to any of the many different events at any time by calling the functions that are associated with these events. Thus, functions in the application program are activated only by user actions.

The only restriction imposed by the preprocessor is that before entering data, the user has to first select the file option from the main menu to choose between automatic and manual mesh generation, enter the name of the input file and choose between creating a new file or modifying an existing file. The preprocessor has to know the type of mesh generation since some of the input parameters are different for automatic and direct mesh generation. The preprocessor also has to know if the data file is a new or existing file since in the case of an existing file, the existing data must be read into arrays for later modification.

After this step, the user is free to enter input parameters in any order. Thus, it is possible to enter a few joint coordinates, then member incidences, then joint coordinates for the remaining joints and then some other data. Joint coordinates and member incidences are stored in different arrays. When the quit option is selected, the preprocessor sorts through the member incidences array and writes the member incidences to the data file starting with the first member. An error
message is displayed on the screen if an error like a missing or out of range member incidence is detected. The same procedure is repeated for joint coordinates and the other input parameters.

3.5.2 Input Techniques

The preprocessor accepts user input via both the mouse pointer and the keyboard. The left mouse button is used to select an option from a menu or to click a command button to invoke an event. Keyboard input is used to enter data in dialog and input boxes.

3.5.3 Main Window

The user-interface consists of a main window and pop-up windows. The main window consists of 16 command buttons which represent the different options offered in the main menu. Figure 3.1 shows the main window and the various command buttons.

The user is free to move, resize or iconify the main window. When the program is first activated, the main window is placed so that the top left corner of the window is placed on the location of the pointer cursor. Initially, the main window covers a small portion of the screen since the user may be using some other programs at the same time due to the multitasking capabilities of XWindow and Unix. It is, however, possible to resize the main window so that it covers the whole screen.
Figure 3.1 Main window of the preprocessor
3.5.4 Pop-up Windows

Pop-up windows are used to display information and interact with the user. Pop-up windows appear on the screen as soon as the user selects one of the options from the main menu. Pop-up windows are placed on the screen with respect to their parents. It is not possible to move, resize or minimize pop-up windows. Pop-up windows are also modal which means that they have to be dismissed before any other window can be selected. The main menu is disabled and does not accept user input when a pop-up window appears on the screen. This procedure makes the pop-up window the only active window and prevents possible input errors.

Each pop-up window is arranged in three logical parts. The first part is the label widget which shows the function of the window. The second part is the section where the user enters input by selecting command buttons and entering text in dialog boxes. The third part is a box widget which contains a group of command buttons. Figure 3.2 shows the pop-up window for entering joint coordinates.

In some menu options, multiple pop-up windows are used. In this case, the window that is on top is the active window and the other windows do not accept user input.

The same group of command buttons are used in all the pop-up windows to develop a consistent and user-friendly interface. These command buttons are labeled "Next", "Delete", "Done", "Cancel" and "Help".
Figure 3.2 Pop-up window for entering joint coordinates
Selecting the Next button allows the input parameters entered in the window to be written to the arrays and clears the dialog boxes for new input entry.

The Delete button is used to erase existing data. Selecting the Delete button deletes the input parameters entered in the input fields in the window from the arrays which store the existing data. When the Delete button is selected, an application function is called by the callback routine and the input parameters entered in the window are compared with the existing values in the arrays. If the two set of values match, the parameters are deleted from the array.

The Done button saves input parameters into the arrays and dismisses the window. The Cancel button dismisses the window without saving the current input parameters.

Selecting the Help button brings up a help window. The help window provides additional information about the option selected, the required input parameters and the use of command buttons.

3.6 Functions Of The Preprocessor

3.6.1 The File Option

When this option is selected a pop-up window appears. The first task to be performed is the selection of the type of mesh generation, manual or automatic.
In the manual mesh generation mode, the joint numbers and the member numbers are specified by the user. Since the program does not perform bandwidth minimization, it is the user's responsibility to use a numbering scheme that would produce the minimum band width of the system stiffness matrix. This is very significant since the band width of the system stiffness matrix has an important effect on both computation time and accuracy. Tezcan (1966) states that "to obtain a relatively narrow band matrix, the joints of a structure should be numbered consecutively along the longest dimension, covering all the joints across the width at each stage." Holzer (1985) states "A general guideline for achieving a narrow band is to number the joints in sequence across the smaller dimension of the structure."

Automatic mesh generation is the process of generating joint numbers and member numbers for a series of equally spaced joints by specifying the number of bays, the width of each bay, the number of stories and height of each story of the structure. All bays should have the same width and all stories should have the same height.

In automatic mesh generation, the preprocessor creates the necessary data for the joint coordinates and member incidences. Thus, the user does not need to enter these input parameters. The number of joints and members are computed and the joints and members are numbered one story at a time starting with the joint at the bottom left corner of the structure. Figure 3.3 shows a structure created using the automatic mesh generation option.
Figure 3.3 Automatic mesh generation

In the File option, the name of the input file that will be created by the preprocessor is also specified. The name of the file should consist of a maximum of eight alphanumeric characters. Special characters such as a period, comma, colon or semicolon are not accepted. An error message appears if this requirement is not met.

Within the File option it is possible to create a new data file or modify an existing data file. A warning message appears if the new file option is selected and a file with the same name already exists. At this point, it is possible to either overwrite the existing file or go back to the file menu and select another file name.
If the existing file option is selected, then data is read from the input file and stored in arrays in memory. It is now possible to delete, modify or make additions to any part of the data as long as the changes are consistent with the rest of the existing data. For example, if the units are changed in the Control variables option, the program does not automatically change the values for the geometric and loading data. Thus, it is the user's responsibility to modify all the data that is unit dependent.

3.6.2 Control Variables

In the direct method of mesh generation, the control variables are the number of members and the number of joints. For automatic mesh generation the control variables are the number of stories, the height of each story, the number of bays and the width of each bay. The number of load cases and the system of units are also required. The user must also indicate whether or not the self-weight of the structure is to be included.

If self-weight of the structure is to be considered, it should be included as a separate load case. However, the self-weight of the members should not be entered manually, since the preprocessor will automatically compute member loads resulting from the self-weight of the members. Member loads due to self-weight are treated as uniformly distributed loads. The self-weight is considered as the last load case. For example, if only dead and live loads are acting on the beams of a plane frame and self-weight of the structure is to be considered, the number of load cases should be entered as 3. The load data for the dead load should be entered as load case 1, and the load data for the live load as load case2.
No load data has to be entered for load case 3 which is the self-weight of the structure. This is entered automatically by the preprocessor.

Input data can be entered in English or metric units. The default is English units. English units are inch, kip, radian and Fahrenheit. Metric units are meter, kiloNewton, radian and Celsius. The units should be consistent throughout the input procedure. All the input windows include a label showing the expected units for each input parameters.

3.6.3 Joint Coordinates

The geometry of the structure is specified by the joint coordinates. All joints are assigned a number and their location is specified by x and y coordinates that are associated with a global coordinate system.

The necessary joint input data consists of the joint number, and the x and y coordinates of each joint.

In automatic mesh generation, joint coordinates are computed by the preprocessor from the control variables and so they do not have to be entered manually. If the joint coordinates option is selected from the main menu while the automatic mesh generation option is on, a pop-up window appears on the screen with a message indicating that it is not necessary to enter joint coordinates.

A joint can be deleted by specifying the joint number and then selecting the delete button.
3.6.4 Joint Constraints

Supports are modeled using horizontal, vertical and rotational constraints. The joint constraints option allows the user to specify these constraints for a particular joint. The necessary input information consists of the joint number and the type of constraint. A menu listing the various types of constraints is provided. The menu options are fixed support, pinned support, horizontal roller, vertical roller, horizontal shear release, vertical shear release and a user defined support. Table 3.1 shows the constrained directions for the different types of supports.

Table 3.1 List of the types of joint constraints

<table>
<thead>
<tr>
<th>Type Of Support</th>
<th>Constrained Directions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Pinned</td>
<td>x, y</td>
</tr>
<tr>
<td>Horizontal roller</td>
<td>y</td>
</tr>
<tr>
<td>Vertical roller</td>
<td>x</td>
</tr>
<tr>
<td>Horizontal shear release</td>
<td>y, z</td>
</tr>
<tr>
<td>Vertical shear release</td>
<td>x, z</td>
</tr>
</tbody>
</table>

If the user defined support option is chosen, another window pops-up. In this window, the user selects whether the joint is constrained for x-displacement, y-displacement or z-rotation.
All the previously entered joint constraints of a joint can be deleted by entering the joint number and selecting the delete button.

3.6.5 Member Incidences

The joints of a structure are connected by members. In direct mesh generation, it is necessary to enter the member number for the member, the joint number corresponding to the start of the member and the joint number corresponding to the end of the member. This information is required for each member of the structure.

In automatic mesh generation, the member incidences are computed by the preprocessor, so it is not necessary to enter the above mentioned information. If the member incidences button is selected in the main menu when the automatic mesh generation option is active, a pop-up window appears on the screen with a message indicating that it is not necessary to enter member incidences.

A member can be deleted by entering the member number and selecting the delete button.

3.6.6 Internal Hinges

For structures in which the beams are connected to columns by simple connections, no moments are generated at the ends of the members. For such cases, the moments at either end of the member can be set to zero by specifying internal hinges. Thus, two members fastened by an internal hinge transfer only
axial and shear forces.

To specify an internal hinge, it is necessary to enter the member number of the member containing the hinge, and also the location of the hinge (i.e. whether the hinge is located at the start or end joint).

3.6.7 Material Properties

The preprocessor requires that material property sets be developed for each type of material. Each material property set contains a list of material properties. The number of material property sets depends on the number of different materials used in the structure. Each material property set is assigned a unique number. Each member of the structure is assigned a material property set. However, it is necessary to create a material property set prior to assigning it to a member.

The required material properties are the modulus of elasticity and the coefficient of thermal expansion. The modulus of elasticity is used to compute the stiffness coefficients. The coefficient of thermal expansion is used to calculate the expansion of members when a uniform temperature change is applied.

The material type can be selected from the material menu. For English units, the menu options are steel, 4ksi concrete, 5ksi concrete, 6ksi concrete and a user-defined material. For metric units, the menu options are steel, 35Mpa concrete, 40MPa concrete, 45MPa concrete and a user-defined material.
After a material type has been selected, the preprocessor searches the material database and reads in the material properties for the selected material. The material database should reside in the same directory as the preprocessor.

If the user defined material option is selected, another window pops up. This pop-up window displays the units for the input parameters and displays input fields for entering the modulus of elasticity and coefficient of thermal expansion.

A material set can be assigned to all members, to a series of members or to individual members. If all of the members of the structure have the same material properties then only one material set is defined and this is assigned to all members. If more than one material is used in the structure, then material sets can be assigned to individual members or to a series of members.

If the user chooses to assign the material set to members in a series, another window pops-up and the user is expected to enter the first and last members of the series and the increment number. The default value for the increment number is 1.

If the material set is to be assigned to individual members, a new window pops-up. This window contains a dialog box to accept user input for individual members. Selecting the Next button in the window assigns the material set to the member that is just entered and clears the dialog box to allow for the entry of a new member. Selecting the Done button assigns the material set to the number just entered and pops down the window.
All members should be assigned with material properties. A zero value for the modulus of elasticity for a member will result in a division by zero error during the analysis phase. When the quit button is selected to exit the preprocessor, the program performs a check to make sure that material properties are assigned to all members of the structure. A warning is displayed if an error is detected. It is sufficient to enter the material set number to delete an existing material set.

3.6.8 Member Properties

Entering member properties is similar to entering material properties. Member sets that contain the values of the member properties have to be defined. The number of member sets depends on the number of different member properties within the structure. Each member set is assigned a unique number.

Member properties consist of area of cross-section, weight per unit length and moment of inertia about the z axis. The area of cross section and moment of inertia are used to compute stiffness coefficients and weight per unit length is used to compute member loads due to self-weight if self-weight of the structure is to be considered.

Member properties can be entered by selecting a section from the AISC steel sections database or by entering the values directly.

If the user-defined member option is selected, the area, the weight per unit length and moment of inertia are entered in the input fields that are displayed in the member properties pop-up dialog window.
If standard steel shapes from the AISC steel manual are to be used, the program displays a list box containing the AISC sections. Only a small portion of the available sections are displayed, but the user can scroll through the entire database using the scrollbar on the left side of the list box. The sections contained in the database include wide-flange shapes (W), I-beams (S), standard channels (C), structural tees (WT, ST), miscellaneous shapes (M, MC, MT) and H-pile shapes (HP). A section is selected by pressing the left mouse button when the cursor is over the section. Once a section is chosen, the member properties are read from the database of AISC steel sections. The AISC database should reside in the same sub-directory as the preprocessor. The orientation of the section (strong or weak axis) can also be specified. The member properties of the sections in the AISC database are in English units. If the structure is defined in metric units and AISC database is used to designate member properties, the values for the member properties that are read from the AISC database are converted into metric units.

A member set can be assigned to all members of the structure, to a series of members, or to individual members. The rules that govern the assignment of material properties also apply to the assignment of member properties.

Each member of the structure should be assigned some member properties. When the quit option is selected to exit the preprocessor, the program performs a check to make sure that all members have been assigned member properties prior to writing the input data file. An error message is displayed if a member does not have member properties assigned to it.
Member sets can be deleted by simply entering the member set number corresponding to that member set and then selecting the delete button.

3.6.9 Loads

All loads acting on a structure are associated with a load case. Load cases are independent loadings for which a structure is analyzed. A load case can consist of any number of loads.

Loads on a structure can be specified as joint loads, member loads, and prescribed displacements. The processor can also compute the self-weights of the members and treat these as uniformly distributed member loads in the analysis. Member loads due to self-weight are considered as a separate load case.

Loads can be entered in any order. Since every load is associated with a load case, the program sorts through the load arrays and writes the loads to the input data file starting with the loads for the first load case.

Concentrated forces or moments may be applied to any free joint of the structure. These loads act in the global coordinate system. Thus, positive loads act in the direction of the positive coordinates.

The required input for joint loads is the joint number at which the load acts, the magnitude of the load and the type of joint load. The types of joint loads that can be applied include a force in the x direction, a force in the y direction and a moment in the z direction. It is also necessary to enter the load case that the
joint load is associated with.

The loads that can be applied to members consist of concentrated loads, axial loads, uniform loads, uniform axial loads, linearly varying loads and uniform temperature changes.

A list of the various types of member loads appears in the member load window. When a member load is selected from the list, a second window containing the appropriate input fields is displayed. Since input parameters change with the type of member load, a different window is used to obtain input parameters for each load type. All member loads are entered in the local member coordinate system. Figure 3.4 shows the different types of member loads and the input parameters needed to define each type. Forces are shown in the positive direction. The names in parentheses represent the abbreviations used in the output file to define the type of member load.

The input parameters for concentrated load and axial load are the member number of the member on which the load acts, magnitude of the load, distance from start of member to the load and the load case.

Uniform distributed loads and uniform axial loads act over the full length of a member. Thus, the input parameters are the member number, the magnitude of the load and the load case number.
Figure 3.4 Member load types

Preprocessor
Linearly varying loads may act over the full or partial length of a member. The input parameters for this load type are the member number, magnitude of the load at the starting point, distance from the start of the member to the start of the load, magnitude of the load at the end point, distance from the start of the member to the end of the load and the load case number.

For uniform temperature changes, the input parameters are the member number, the value of the temperature change and the load case number. An increase in temperature is considered positive.

If the self-weight of the structure is to be considered, the member loads due to self-weight need not be entered manually. The load due to member weight is applied as a uniformly distributed member load.

Prescribed translational or rotational displacements can be specified on the joints of a structure. Prescribed displacements act in the global coordinate system. The required input consists of the joint number for the prescribed displacement, magnitude of the displacement and the type of displacement. The type of displacement can be selected from a list of menu options and consist of displacement in the x direction, displacement in the y direction, and rotation in the z direction. The load case that the prescribed displacement is associated with must also be entered.

3.6.10 Load Descriptions

A description can be given for each load case. A maximum of forty-five
characters are allowed for each description. The load descriptions are printed on
the output file created by the program. To add a description to a load case it is
necessary to enter the load case number and the load case description in the load
description window. An existing load case description can be deleted by simply
entering the load case number.

3.6.11 Load Combinations

Load combinations are loadings formed by linear combinations of independent
load cases. The required input consists of the load combination number, the load
case numbers for the load cases to be combined and the corresponding load factors
to be applied to each load case. A maximum of five load cases can be combined
in a given load combination. Five dialog boxes are placed on the window to enter
the load cases and load factors. If there are less than five load cases for a load
combination, the remaining dialog boxes can be left empty. A load combination
can be deleted by simply entering the load combination number for an existing
load combination.

3.6.12 Output Options

The input information and the results of the analysis are written to an output file
which can be printed to obtain a hard copy. The output options window lets the
user specify whether the input information and results for individual load cases
and load combinations are to be written to an output file. The user has the
option to select which load cases are to be printed.
The default for the output options is to write the input information and all load cases to the output file.

3.6.13 Quit

When the quit button is selected from the main menu, a window pops up requesting confirmation to exit the program. This window also has an option for canceling the request to exit the program.

An error message appears if an attempt is made to exit the program before entering the name of the file on which input data should be saved. It is necessary to select the file option and enter a file name prior to exiting the program.

A number of error checks are performed prior to writing input data to a file. An error message is displayed if an error is found in the input data and the user is provided with the opportunity to correct the error or simply exit the program without creating the input data file.

The following error messages appear if an error is found during the error checks:
1. The value for the number of members is not acceptable, e.g., a negative or zero value.
2. The value for the number of joints is not acceptable, e.g., a negative or zero value.
3. The value for the number of load cases is not acceptable, e.g., a negative or zero value.
4. In automatic mesh generation, a negative value is entered for the number of stories or bays.

5. In automatic mesh generation, a zero or negative value is entered for the height of a story or the width of a bay.

6. The value of a member incidence is not acceptable, e.g., out of range, zero or negative.

7. Member incidences of a member are missing.

8. The number of a constrained joint is not acceptable, e.g., out of range, zero or negative.

9. The member number for an internal hinge is not acceptable, e.g., out of range, zero or negative.

10. Two joints of a member have the same coordinates. This is not acceptable because the member has zero length and infinite stiffness which would cause an overflow error in the processor.

11. Coordinates of a joint are missing.

12. A negative or zero value is entered for modulus of elasticity.

13. A member number is not acceptable for assigning material properties, e.g., out of range, zero, or negative.

14. Material properties are not assigned to all members.

15. A negative or zero value is entered for moment of inertia.

16. A member number is not acceptable for assigning member properties, e.g., out of range, zero or negative.

17. Member properties are not assigned to all members.

18. A joint number for a joint load is not acceptable, e.g., out of range, zero or negative.
19. A member number for a member load is not acceptable, e.g., out of range, zero or negative.

20. The distances from the start of the member to the start of load and end of load cannot be equal for linearly varying member loads.

21. A negative distance is not acceptable for member loads.

22. A joint number for a joint displacement is not acceptable, e.g., out of range, zero, or negative.

23. A load case number is not acceptable for a load combination, e.g., out of range, zero or negative.

3.6.14 About

This option gives brief information about the preprocessor, the current version number and the author.
CHAPTER 4

PROCESSOR

4.1 Overview

In this chapter, a description of the frame analysis program is presented. The features and limitations of the processor are discussed and the program structure is presented along with a description of the main components of the program. In Section 4.2, background information on the analysis of plane frames is presented. Section 4.3 describes the characteristics and features of the matrix displacement method. In Section 4.4, the design, features and computation procedures of the processor are presented. In Section 4.5, the modifications made to Hoizer's frame analysis program (1987) are discussed.
4.2 Plane Frames

Plane frame structures consist of members that are long in comparison to their cross-sectional dimensions. The joints are points of intersection of the members. The location of a joint is specified by x and y coordinates. The following factors should be considered when locating joints on a structure:

1. The number of joints should be sufficient to describe the geometry of the structure.
2. Joints need to be located at points of discontinuity; e.g. at changes in material properties, member properties.
3. Joints should be located at all support points.

The joints may be constrained by any combination of translational or rotational constraints to represent fixed, pinned or roller supports. The joint constraints must be of sufficient number and type to ensure that the structure is stable under any system of applied loads or prescribed displacements. Loads on the frame may be concentrated forces acting on the joints or concentrated loads, distributed loads or temperature change acting on the members. Prescribed displacements of joints are considered as loads.

A plane frame consists of members lying in a single plane with axis of symmetry in that plane. The forces and displacements are in the plane of the structure. Moments must act about an axis perpendicular to the plane of the structure. The internal stress resultants acting at a cross-section of a member consist of an axial force, a shearing force and a bending moment. The frame has prismatic members with a straight axis and constant cross-section throughout its length.
4.3 Matrix Displacement Method

The application of matrix methods to the field of structural analysis has improved the efficiency of the analysis of structural systems. The matrix approach is easily programmed for digital computers. Use of matrices enables large groups of numbers to be handled in an effective way. Thus, the entire computation process can be carried out by a series of simple matrix operations. The matrix displacement method enables the analysis of a wide variety of structural types in contrast to traditional methods which often differ for each type of structure (Weaver and Gere 1990).

The direct stiffness method is the best and most general approach for the analysis of structures on digital computers. While the basic principles of the stiffness method is the same for all framed structures, the details of the analysis are different for beams, plane frames, plane trusses, space trusses, grids and space frames.

In the stiffness method of analysis, the components of joint displacements that are free to move are called degrees of freedom. The total number of degrees of freedom represent the number of unknowns in the analysis. In a plane frame, the number of degrees of freedom of a free joint are equal to 3. Thus, a plane frame member has six degrees of freedom. Figure 4.1 shows a plane frame member with local end displacements \( d_i \) and local end forces \( f_i \).
In the matrix displacement method, the structure is idealized as an assembly of discrete structural members interconnected at joints. A correct solution for a structural analysis must satisfy all conditions of equilibrium for the entire structure as well as any part of the structure taken as a free body. In addition to the static equilibrium conditions, all conditions of compatibility should be satisfied. Conditions of compatibility refer to the continuity of displacements between members and joints. Matrix analysis gives the exact solution of a structure since the solution simultaneously satisfies equilibrium, compatibility and boundary conditions.

In the matrix displacement method, members are represented by stiffness matrices that relate the member-end displacements to member-end forces. The member models are assembled into a system model by imposing conditions of compatibility and equilibrium. The constitutive laws which model the behavior of materials are already contained in the member models. Thus, the system model relates the joint displacements to applied joint forces through the system stiffness matrix. The member code matrix permits the assembly of member models into the system model without visual reference to the structure.
In the matrix displacement method, member loads on the structure are transformed into discrete concentrated forces acting at the joints of the structure through the use of fixed-end actions. The stiffness matrix relates these loads to the displacements of the joints. The principles of superposition may be used since a linear relationship exists between forces and displacements.

The stiffness method allows the incorporation of the effects of temperature changes and prescribed joint displacements into the analysis.

The following analysis procedure is followed in the matrix displacement method:
1. Determine the unknown displacements.
2. Form the member models.
\[ \hat{\mathbf{F}}^i = k^i \mathbf{d}^i + \hat{\mathbf{f}}, \quad i=1,2,\ldots \]
where \( \hat{\mathbf{F}} \) = local member force vector, \( k^i \) = stiffness matrix, \( \mathbf{d}^i \) = local member displacement vector, \( \hat{\mathbf{f}} \) = local fixed-end vector (incorporates member loads).
3. Assemble the member models into the system model by imposing conditions of compatibility and equilibrium. Use the member code matrix for the assembly.

System model: \( Kq = Q \)
where \( K \) = system stiffness matrix, \( q \) = unknown joint displacements, \( Q \) = applied joint force vector.
4. Solve for the unknown displacements, \( q^i \).
5. Compute the member forces(\( \hat{\mathbf{F}}^i \)) from the displacements, \( \hat{\mathbf{F}}^i = f^i + \hat{\mathbf{f}} \).
6. Compute the joint forces from the member forces by condition of joint equilibrium.
\[ \mathbf{F}_j = \sum_{i=1}^{NM} \mathbf{F}^i_j \]
where \( \mathbf{F}_j \) = joint forces, \( NM \) = number of members, \( \mathbf{F}^i_j \) = global member forces.
The bandwidth is defined as the numerical difference of the incidences of a member. Reducing the bandwidth size results in a faster solution time. Numbering the joints in sequence across the smaller dimension of the structure results in a narrow band matrix.

Coordinate transformations are required in the assembly of member models. This is because member models are expressed in local coordinates and assembled in global coordinates. Thus, it is important to know the relationship between the local and global coordinate systems.

Member models are expressed in the local coordinate system. All member loads are referenced to the local coordinate axes. The local coordinate axes are defined as follows: the local x-axis extends from the start joint to the end joint to which the member is incident; the local y-axis makes the local frame of reference right-handed, the local z-axis is oriented in the direction of the global z-axis. Figure 4.2 shows the local coordinate system and the local member displacements and forces.

The local member displacements and forces are denoted by lowercase letters $d_j, f_j$. They are numbered in the sequence of the local coordinate axes from the start to the end of the member.
Figure 4.2 Local coordinate system

The member models are assembled into the system model in global coordinates. Global coordinate system follows the orthogonal right-handed rule. Joint locations, loading directions, displacements and forces are all defined in the global coordinate system. The global member displacements and forces are denoted by the uppercase letters $D_j$, $F_j$. They are numbered in the sequence of the global coordinate axes from the start to the end of the member. Figure 4.3 shows the global coordinate system and the global displacements and forces.
Figure 4.3 Global coordinate system

The sign of a displacement or force is determined by the direction of the corresponding coordinate axes. The rotations and moments about the z-axis are positive in the counterclockwise direction. All displacements and forces shown in Figures 4.2 and 4.3 are positive.

4.4 Structure and Features of the Processor

The processor retrieves the input data from the input file created by the preprocessor and performs matrix displacement analysis of the structure. The
results of the analysis are written to an output file.

The processor is based on a plane frame analysis program written by Holzer (1985). Figure 4.4 shows the structure of the processor. The program structure and the functions, Nassi-Schneiderman diagrams and input and output arguments of all the subprograms are presented in Holzer (1985). The original program was written in FORTRAN. The program is completely rewritten using the C programming language. Dynamic memory allocation is utilized for data and solution arrays. This enables efficient use of memory.

The processor performs matrix displacement analysis in the following sequence:

1. Identification of structural data: this is performed by the STRUCT module of the program. Information about the structure is assembled and written to the output file. This information includes the number of members, the number of joints, the number of load cases, the member incidences and the material properties. The locations of the joints are specified by the geometric coordinates. The section properties of each member are also processed. Furthermore, the joint constraints are identified. The length and direction cosines of each member are computed. The joint code and member code matrices are generated according to the degrees of freedom.

2. Identification of load data: this is performed by the LOAD module. All loads acting on the structure are specified. These loads include joints loads and member loads.

3. Construction of load vector: this is done by the LOAD module. The fixed-end forces due to member loads are transformed to equivalent joint loads and these equivalent joint loads are added to the actual joint loads.
Figure 4.4 Structure of the processor
4. Construction of stiffness matrix: this is performed by the SYSTEM1 module. First, the global stiffness coefficients are computed for each member. Then, the stiffness coefficients of each member are assigned to the system stiffness matrix. Prescribed joint displacements are imposed at the system level by modifying the system stiffness matrix and joint load vectors according to the penalty method. In the penalty method, a spring of large stiffness is added at the degree of freedom where the displacement is prescribed. The spring stiffness is chosen to be 10,000 times the maximum stiffness in the structure.

5. Solution of system equations: this is done by the SYSTEM1 module. The system model $Kq = Q$ is solved for the unknown joint displacements ($q$). The system equations are solved by compact Gaussian elimination. If $K$ is singular, the structure is unstable and rigid-body motions are possible. In this case, the program stops and an error message appears.

6. Computation of member forces and joint loads: this is performed by the RESULT module. The global member end-displacements are determined from the joint displacements and transformed to local member end-displacements. The local forces of each member are computed from the local member end-displacements by using the member models. To compute the joint forces, the local forces of each member are transformed to global forces and assigned to the joint force matrix.

7. Computation of load combinations: this is done by the COMBINE module. The joint displacements, joint forces and local member end forces are multiplied by the load factor for each load case and these values are combined to get the results for the load combination.
The processor has the following features:

1. Each member may have distinct material and member properties.
2. Internal hinges can be specified on the structure. Members that are connected by an internal hinge transfer only axial and shear forces.
3. The program is capable of handling multiple load cases and load combinations.
4. The processor accepts the following load types:
   a. Joint loads
   b. Member loads
      1. Concentrated loads
      2. Uniform loads
      3. Axial loads
      4. Uniform axial loads
      5. Linearly varying loads
      6. Uniform temperature changes
5. Prescribed joint displacements can be specified.
6. Self-weight of the structure can be considered as a separate case.

The limitations of the processor are as follows:

1. Analysis is confined to prismatic members. Prismatic members have the same cross-sectional properties from one end to the other (Weaver and Gere 1990).
2. The structure is subjected to static loads and only linear analysis is considered.
3. Loads and responses must lie in the plane of the structure.
4.5 Modifications to Holzer’s Program

The program is based on a modified version of Holzer’s frame analysis program. Some features have been added to the original frame analysis program by Holzer himself. These extensions include the use of a Skyline equation solver and utilization of internal hinges and prescribed joint displacements.

In this study, the entire program is rewritten in C and the following features are added:

1. Concentrated axial load, uniform axial load and trapezoidal load are added to the existing member load types.

2. Modifications are made to account for the self-weight of the structure as a separate load case. If self-weight is to be considered, the member loads due to self-weight are computed automatically. This is achieved by multiplying the weight per unit length for each member by the direction cosines to compute the uniform and/or uniform axial loads due to self-weight.

3. Modifications are made in the processor to handle load combinations. Since load combinations are formed by the linear combination of independent load cases, the joint displacements, joint forces and local member-end forces are multiplied by the load factor for each load case and these values are combined to get the joint displacements, joint loads and local member-end forces for the load combination.

4. Modifications are made in the processor to assign a description for each load case and to specify output options for the output file.

5. Changes are made to assign material and member property sets to all members, members in a series or individual members.
6. Some other minor changes are utilized to accept data from the input data file created by the preprocessor.
CHAPTER 5

PROGRAM VERIFICATION

5.1 Overview

In this chapter, five structures considered in the verification and evaluation of the frame analysis program are presented. The example structures demonstrate the capabilities of the frame analysis program. For each structure, a short description and a list of input parameters and options activated during the creation of the input data file by the preprocessor are given. To verify the accuracy of the frame analysis program, the results of the analyses are compared with the results obtained from STAAD-III (Research Engineers, Inc. 1989). The results that were compared include support reactions, joint displacements and local member end forces. STAAD-III is a fully integrated structural engineering software package capable of structural analysis, design and drafting. The accuracy of the results for example structures 1 through 4 are also verified by hand calculations. The results obtained by hand calculations are presented in Appendix A.
5.2 Example Structure 1

5.2.1 Description

The first example structure is a test problem given in Holzer (1985). The structure is a plane frame and consists of six joints and six members. The frame is subjected to two load cases. The first load case consists of uniformly distributed loads acting on the beams. The second load case consists of joint loads. Automatic mesh generation is utilized in the creation of the input data file since the structure is an orthogonal plane frame with equally spaced joints. The structure is shown in Figure 5.1.

5.2.2 Input Data

1. **Name of input file**: ex1
2. **Units**: English (kip, inch, radian, Fahrenheit).
3. **Type of mesh generation**: automatic
4. **Control variables**:
   Number of stories: 2
   Story height: 120 in.
   Number of bays: 1
   Bay width: 120 in.
   Number of load cases: 2
   Self-weight of the structure is not considered.
5. **Output options**: print input data, print output for all load cases.
Figure 5.1 Example structure 1
6. **Joint coordinates:** since automatic mesh generation option is on, joint coordinates are computed directly by the preprocessor.

7. **Joint constraints:** joints 1 and 2 are fixed.

8. **Member incidences:** since automatic mesh generation option is on, member incidences are computed directly by the preprocessor.

9. **Internal hinges:** no internal hinges.

10. **Material properties:**

    Material set number: 1

    Material type: user-defined

    Modulus of elasticity = 30000 ksi

    Coefficient of thermal expansion = 6.5 x 10^6

    Assign material property set 1 to all members.

11. **Member properties:**

    Member set number: 1

    Member type: user-defined

    Area = 5 sq. in.

    Moment of inertia = 50 in^4

    Weight per length = 0.001415 k/in

    Assign member property set 1 to all members.

12. **Joint Loads:**

    Load case: 2

<table>
<thead>
<tr>
<th>Joint</th>
<th>Direction</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>x</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>1.5</td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td>1.5</td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td>1.5</td>
</tr>
</tbody>
</table>
13. **Member loads:**

Load case: 1

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Load1</th>
<th>Load2</th>
<th>Distance1</th>
<th>Distance2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>uniform</td>
<td>-0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>uniform</td>
<td>-0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. **Prescribed displacements:** no prescribed displacements.

15. **Load descriptions:**

<table>
<thead>
<tr>
<th>Load case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dead+live</td>
</tr>
<tr>
<td>2</td>
<td>wind</td>
</tr>
</tbody>
</table>

16. **Load combinations:** no load combinations.

### 5.2.3 Comparison Of Results

Tables 5.1, 5.2 and 5.3 show the support reactions, joint displacements and local member end forces obtained from the frame analysis program and STAAD III and the percentage difference between the two. The results from the program agree with those from STAAD III within 1% for example 1.
Table 5.1 Support reactions for example structure 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek Fx (kips)</th>
<th>Erkek Fy (kips)</th>
<th>Erkek Mz (kip-in.)</th>
<th>STAAD III Fx (kips)</th>
<th>STAAD III Fy (kips)</th>
<th>STAAD III Mz (kip-in.)</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.87</td>
<td>24.00</td>
<td>-34.77</td>
<td>0.87</td>
<td>24.00</td>
<td>-34.77</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>2</td>
<td>-0.87</td>
<td>24.00</td>
<td>34.77</td>
<td>-0.87</td>
<td>24.00</td>
<td>34.77</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Table 5.2 Joint displacements for example structure 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-0.00068</td>
<td>-0.001920</td>
<td>-0.00137</td>
<td>-0.00068</td>
<td>-0.001920</td>
<td>-0.00137</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>4</td>
<td>0.00068</td>
<td>-0.001920</td>
<td>0.00137</td>
<td>0.00068</td>
<td>-0.001920</td>
<td>0.00137</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>5</td>
<td>0.000103</td>
<td>-0.002880</td>
<td>-0.00276</td>
<td>0.000103</td>
<td>-0.002880</td>
<td>-0.00276</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>6</td>
<td>-0.000103</td>
<td>-0.002880</td>
<td>0.00276</td>
<td>-0.000103</td>
<td>-0.002880</td>
<td>0.00276</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Program Verification 74
Table 5.3 Local member end forces for example structure 1

### Load Case 1

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kips</td>
<td>kips</td>
<td>kip-in.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>24.00</td>
<td>-0.87</td>
<td>-34.77</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-24.00</td>
<td>0.87</td>
<td>-69.12</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>24.00</td>
<td>0.87</td>
<td>34.77</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-24.00</td>
<td>-0.87</td>
<td>69.12</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-1.70</td>
<td>12.00</td>
<td>205.70</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.70</td>
<td>12.00</td>
<td>-205.70</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>12.00</td>
<td>-2.56</td>
<td>-136.50</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-12.00</td>
<td>2.56</td>
<td>-171.10</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>12.00</td>
<td>2.56</td>
<td>136.50</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-12.00</td>
<td>-2.56</td>
<td>171.10</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2.56</td>
<td>12.00</td>
<td>171.10</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-2.56</td>
<td>12.00</td>
<td>-171.10</td>
</tr>
</tbody>
</table>

### Load Case 2

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kips</td>
<td>kips</td>
<td>kip-in.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-5.39</td>
<td>3.00</td>
<td>216.80</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5.39</td>
<td>-3.00</td>
<td>143.20</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5.39</td>
<td>3.00</td>
<td>216.80</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-5.39</td>
<td>-3.00</td>
<td>143.20</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.00</td>
<td>-3.59</td>
<td>-215.50</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.00</td>
<td>3.59</td>
<td>-215.50</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>-1.80</td>
<td>1.50</td>
<td>72.33</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.80</td>
<td>-1.50</td>
<td>107.70</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1.80</td>
<td>1.50</td>
<td>72.33</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-1.80</td>
<td>-1.50</td>
<td>107.70</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>0.00</td>
<td>-1.80</td>
<td>-107.70</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.00</td>
<td>1.80</td>
<td>-107.70</td>
</tr>
</tbody>
</table>

Program Verification
5.3 Example Structure 2

5.3.1 Description

The second example structure is a plane frame that consists of four joints and three members. Standard steel shapes from the AISC steel manual are used for the members of the structure. The frame is subjected to three load cases. The first load case consists of uniformly distributed load acting on the second member of the structure. The second load case consists of a joint load. The third load case consists of the self-weight of the members which are computed automatically by the program. The first and second load cases are combined in load combination 1. The structure is shown in Figure 5.2.

Figure 5.2 Example structure 2
5.3.2 Input Data

1. Name of input file: ex2
2. Units: English (kip, inch, radian, Fahrenheit).
3. Type of mesh generation: direct
4. Control variables:
   Number of members: 3
   Number of joints: 4
   Number of load cases: 3
   Self-weight of the structure is considered as load case 3.
5. Output options: print input data, print output for all load cases, print output for load combinations.
6. Joint coordinates:

<table>
<thead>
<tr>
<th>Joint number</th>
<th>X-direction</th>
<th>Y-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>180.00</td>
</tr>
<tr>
<td>3</td>
<td>240.00</td>
<td>180.00</td>
</tr>
<tr>
<td>4</td>
<td>240.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
7. Joint constraints: joint 1 is fixed and joint 2 is pinned.
8. Member incidences:

<table>
<thead>
<tr>
<th>Member</th>
<th>Start joint</th>
<th>End joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
9. Internal hinges: no internal hinges.
10. **Material properties:**

Material set number: 1

Material type: steel

Assign material property set 1 to all members.

11. **Member properties:**

Member set number: 1

Member type: AISC section W8×18

Assign member property set 1 to a series of members

First member: 1

Last member: 3

Increment: 2

Member set number: 2

Member type: AISC section W12×26

Assign member set 2 to individual members

Member number: 2

12. **Joint Loads:**

Load case: 2

<table>
<thead>
<tr>
<th>Joint</th>
<th>Direction</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>x</td>
<td>10</td>
</tr>
</tbody>
</table>

13. **Member loads:**

Load case: 1

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Load1</th>
<th>Load2</th>
<th>Distancel</th>
<th>Distance2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. **Prescribed displacements:** no prescribed displacements.
15. **Load descriptions:**

<table>
<thead>
<tr>
<th>Load case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dead+live</td>
</tr>
<tr>
<td>2</td>
<td>wind from left</td>
</tr>
<tr>
<td>3</td>
<td>self weight</td>
</tr>
</tbody>
</table>

16. **Load combinations:**

Load combination number: 1

<table>
<thead>
<tr>
<th>Load case</th>
<th>Load factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
</tr>
</tbody>
</table>
5.3.3 Comparison Of Results

Tables 5.4, 5.5 and 5.6 show the support reactions, joint displacements and local member end forces obtained from the frame analysis program and STAAD III and the percentage difference between the two. Except for a few isolated cases, the results agree within 1%. The reason for the 4.17% difference in the rotation of joint 3 for load case 3 is because of round-off errors since the magnitude of the compared results is very small. The reason for the 1.92% difference in the local end force for member 1 and the support reaction in y-direction for joint 1 for load case 3 is because of differences in the calculation of member loads due to self-weight between STAAD-III and the frame analysis program. STAAD-III calculates member loads due to self-weight by multiplying the area of the cross section by the density of the material. The frame analysis program computes the self-weight by taking the weight per foot value from the AISC steel sections database and converting it to weight per inch if the member is made up of a standard steel section, or asks the user to input the value for weight per unit length. STAAD-III computes the weight per inch value for member 1 as 0.0014886 while it is taken as 0.0015 in the frame analysis program. This difference in weight causes the 1.92% difference in the local member end forces and the support reactions.
Table 5.4 Support reactions for example structure 2

### Load Case 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fx</td>
<td>Fy</td>
<td>Mz</td>
</tr>
<tr>
<td>1</td>
<td>1.17</td>
<td>11.81</td>
<td>-44.70</td>
</tr>
<tr>
<td>4</td>
<td>-1.17</td>
<td>12.19</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Case 2

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fx</td>
<td>Fy</td>
<td>Mz</td>
</tr>
<tr>
<td>1</td>
<td>-7.76</td>
<td>-4.34</td>
<td>758.00</td>
</tr>
<tr>
<td>4</td>
<td>-2.24</td>
<td>4.34</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Case 3

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fx</td>
<td>Fy</td>
<td>Mz</td>
</tr>
<tr>
<td>1</td>
<td>0.03</td>
<td>0.53</td>
<td>-0.97</td>
</tr>
<tr>
<td>4</td>
<td>-0.03</td>
<td>0.53</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Combination 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fx</td>
<td>Fy</td>
<td>Mz</td>
</tr>
<tr>
<td>1</td>
<td>-4.94</td>
<td>5.60</td>
<td>534.90</td>
</tr>
<tr>
<td>4</td>
<td>-2.56</td>
<td>12.40</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Program Verification
### Table 5.5 Joint displacements for example structure 2

#### Load Case 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-Trans</td>
<td>Y-Trans</td>
<td>Z-Rot</td>
</tr>
<tr>
<td>2</td>
<td>0.22990</td>
<td>-0.01394</td>
<td>-0.00607</td>
</tr>
<tr>
<td>3</td>
<td>0.22860</td>
<td>-0.01438</td>
<td>0.00577</td>
</tr>
<tr>
<td>4</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00479</td>
</tr>
</tbody>
</table>

#### Load Case 2

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-Trans</td>
<td>Y-Trans</td>
<td>Z-Rot</td>
</tr>
<tr>
<td>2</td>
<td>2.63800</td>
<td>0.00512</td>
<td>-0.00596</td>
</tr>
<tr>
<td>3</td>
<td>2.63500</td>
<td>-0.00512</td>
<td>-0.00117</td>
</tr>
<tr>
<td>4</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.02137</td>
</tr>
</tbody>
</table>

#### Load Case 3

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-Trans</td>
<td>Y-Trans</td>
<td>Z-Rot</td>
</tr>
<tr>
<td>2</td>
<td>0.00498</td>
<td>-0.00046</td>
<td>-0.00013</td>
</tr>
<tr>
<td>3</td>
<td>0.00495</td>
<td>-0.00047</td>
<td>0.00013</td>
</tr>
<tr>
<td>4</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00010</td>
</tr>
</tbody>
</table>

#### Load Combination 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-Trans</td>
<td>Y-Trans</td>
<td>Z-Rot</td>
</tr>
<tr>
<td>2</td>
<td>2.15100</td>
<td>-0.00661</td>
<td>-0.00902</td>
</tr>
<tr>
<td>3</td>
<td>2.14800</td>
<td>-0.01463</td>
<td>0.00345</td>
</tr>
<tr>
<td>4</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.01962</td>
</tr>
</tbody>
</table>
Table 5.6 Local member end forces for example structure 2

### Load Case 1

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11.81</td>
<td>-1.17</td>
<td>-44.70</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-11.81</td>
<td>1.17</td>
<td>165.80</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.17</td>
<td>11.81</td>
<td>165.80</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-1.17</td>
<td>12.19</td>
<td>-210.50</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>12.19</td>
<td>1.17</td>
<td>210.50</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-12.19</td>
<td>-1.17</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Case 2

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-4.34</td>
<td>7.76</td>
<td>758.00</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4.34</td>
<td>-7.76</td>
<td>639.10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.24</td>
<td>-4.34</td>
<td>-639.10</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-2.24</td>
<td>4.34</td>
<td>403.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4.34</td>
<td>2.24</td>
<td>403.00</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-4.34</td>
<td>-2.24</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Case 3

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.53</td>
<td>-0.03</td>
<td>-0.97</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-0.26</td>
<td>0.03</td>
<td>-3.59</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.03</td>
<td>0.26</td>
<td>3.59</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-0.03</td>
<td>0.26</td>
<td>-4.56</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.26</td>
<td>0.03</td>
<td>4.56</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-0.53</td>
<td>-0.03</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Load Combination 1

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial</td>
<td>Shear</td>
<td>Moment</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5.60</td>
<td>4.94</td>
<td>534.90</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-5.60</td>
<td>-4.94</td>
<td>354.90</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.56</td>
<td>5.60</td>
<td>-354.90</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-2.56</td>
<td>12.40</td>
<td>-460.10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>12.40</td>
<td>2.56</td>
<td>460.10</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-12.40</td>
<td>-2.56</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Program Verification 83
5.4 Example Structure 3

5.4.1 Description

The third example structure is a plane frame that consists of two members and three joints. The input parameters are defined in metric units. The structure is made of steel members, so material properties are read from the material database. The frame is subjected to two load cases. The first load case consists of a prescribed joint rotation. The second load case consists of the self-weight of the members which are computed automatically by the program. The two load cases are combined in load combination 1. The structure is shown in Figure 5.3.

Figure 5.3 Example structure 3
5.4.2 Input Data

1. **Name of input file**: ex3

2. **Units**: metric (kN, meters, radians, degrees Celsius).

3. **Type of mesh generation**: direct

4. **Control variables**:

   Number of members: 2

   Number of joints: 3

   Number of load cases: 2

   Self-weight of the structure is considered as load case 2.

5. **Output options**: print input data, print output for all load cases, print output for load combinations.

6. **Joint coordinates**:

<table>
<thead>
<tr>
<th>Joint number</th>
<th>X-direction</th>
<th>Y-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>4.00</td>
<td>3.00</td>
</tr>
<tr>
<td>3</td>
<td>4.00</td>
<td>3.00</td>
</tr>
</tbody>
</table>

7. **Joint constraints**: joint 1 is fixed, joints 2 and 3 are pinned.

8. **Member incidences**:

<table>
<thead>
<tr>
<th>Member</th>
<th>Start joint</th>
<th>End joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

9. **Internal hinges**: no internal hinges.

10. **Material properties**:

    Material set number: 1

    Material type: steel
Assign material property set 1 to all members.

11. **Member properties:**

Member set number: 1

Member type: user-defined

Area = 0.003 sq. m

Moment of inertia = 2 × 10⁻⁵ m⁴

Weight per length = 0.23 kN/m

Assign member property set 1 to all members.

12. **Joint Loads:** no joint loads

13. **Member loads:** no member loads

14. **Prescribed displacements:**

Load case: 1

<table>
<thead>
<tr>
<th>Joint</th>
<th>Direction</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>z</td>
<td>0.002</td>
</tr>
</tbody>
</table>

15. **Load descriptions:**

Load case Description

1 joint rotation

2 self weight

16. **Load combinations:**

Load combination number: 1

<table>
<thead>
<tr>
<th>Load case</th>
<th>Load factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1.4</td>
</tr>
</tbody>
</table>
5.4.3 Comparison Of Results

Tables 5.7, 5.8 and 5.9 show the support reactions, joint displacements and local member end forces obtained from the frame analysis program and STAAD III and the percentage difference between the two. The results obtained from the program for example 3 are identical to those obtained from STAAD III.
Table 5.7 Support reactions for example structure 3

### Load Case 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek (Fx, Fy, Mz)</th>
<th>STAAD III (Fx, Fy, Mz)</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kN, kN, kN-m</td>
<td>kN, kN, kN-m</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>0.00, -0.86, -1.14</td>
<td>0.00, -0.86, -1.14</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>2</td>
<td>-3.81, 0.86, 0.00</td>
<td>-3.81, 0.86, 0.00</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3.81, 0.00, 9.14</td>
<td>3.81, 0.00, 9.14</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
</tbody>
</table>

### Load Case 2

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek (Fx, Fy, Mz)</th>
<th>STAAD III (Fx, Fy, Mz)</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kN, kN, kN-m</td>
<td>kN, kN, kN-m</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>0.00, 0.52, 0.38</td>
<td>0.00, 0.52, 0.38</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>2</td>
<td>-0.05, 0.75, 0.00</td>
<td>-0.05, 0.75, 0.00</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>0.05, 0.35, 0.00</td>
<td>0.05, 0.35, 0.00</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
</tbody>
</table>

### Load Combination 1

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek (Fx, Fy, Mz)</th>
<th>STAAD III (Fx, Fy, Mz)</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kN, kN, kN-m</td>
<td>kN, kN, kN-m</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>0.00, -0.14, -0.62</td>
<td>0.00, -0.14, -0.62</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>2</td>
<td>-3.88, 1.90, 0.00</td>
<td>-3.88, 1.90, 0.00</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3.88, 0.48, 9.14</td>
<td>3.88, 0.48, 9.14</td>
<td>0.00%, 0.00%, 0.00%</td>
</tr>
</tbody>
</table>

Program Verification 88
Table 5.8 Joint displacements for example structure 3

Load Case 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>%</td>
</tr>
<tr>
<td>2</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00057</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00057</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00200</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00200</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Load Case 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>%</td>
</tr>
<tr>
<td>2</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00004</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00004</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00002</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00002</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Load Combination 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>meter</td>
<td>meter</td>
<td>radian</td>
<td>%</td>
</tr>
<tr>
<td>2</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00052</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-0.00052</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00197</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00197</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Program Verification
Table 5.9 Local member end forces for example structure 3

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek Axial (kN)</th>
<th>Erkek Shear (kN)</th>
<th>Erkek Moment (kN-m)</th>
<th>STAAD III Axial (kN)</th>
<th>STAAD III Shear (kN)</th>
<th>STAAD III Moment (kN-m)</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>-0.86</td>
<td>-1.14</td>
<td>0.00</td>
<td>-0.96</td>
<td>-1.14</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>0.86</td>
<td>-2.29</td>
<td>0.00</td>
<td>0.86</td>
<td>-2.29</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.00</td>
<td>3.81</td>
<td>2.29</td>
<td>0.00</td>
<td>3.81</td>
<td>2.29</td>
<td>0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.00</td>
<td>-3.81</td>
<td>9.14</td>
<td>0.00</td>
<td>-3.81</td>
<td>9.14</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Load Case 2

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek Axial (kN)</th>
<th>Erkek Shear (kN)</th>
<th>Erkek Moment (kN-m)</th>
<th>STAAD III Axial (kN)</th>
<th>STAAD III Shear (kN)</th>
<th>STAAD III Moment (kN-m)</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>0.52</td>
<td>0.38</td>
<td>0.00</td>
<td>0.52</td>
<td>0.38</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>0.40</td>
<td>-0.15</td>
<td>0.00</td>
<td>0.40</td>
<td>-0.15</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.35</td>
<td>0.05</td>
<td>0.15</td>
<td>0.35</td>
<td>0.05</td>
<td>0.15</td>
<td>0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.35</td>
<td>-0.05</td>
<td>0.00</td>
<td>0.35</td>
<td>-0.05</td>
<td>0.00</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Load Combination 1

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek Axial (kN)</th>
<th>Erkek Shear (kN)</th>
<th>Erkek Moment (kN-m)</th>
<th>STAAD III Axial (kN)</th>
<th>STAAD III Shear (kN)</th>
<th>STAAD III Moment (kN-m)</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>-0.13</td>
<td>-0.61</td>
<td>0.00</td>
<td>-0.13</td>
<td>-0.61</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>1.42</td>
<td>-2.50</td>
<td>0.00</td>
<td>1.42</td>
<td>-2.50</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.48</td>
<td>3.88</td>
<td>2.50</td>
<td>0.48</td>
<td>3.88</td>
<td>2.50</td>
<td>0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.48</td>
<td>-3.88</td>
<td>9.14</td>
<td>0.48</td>
<td>-3.88</td>
<td>9.14</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
5.5 Example Structure 4

5.5.1 Description

The fourth example structure is a plane frame that consists of three members and four joints. Member 3 is connected to the other members by an internal hinge. The frame is subjected to one load case. This load case consists of a linearly varying member load acting on member 1 and a uniform temperature change acting on member 2. The structure is shown in Figure 5.4.

Figure 5.4 Example structure 4
5.5.2 Input Data

1. **Name of input file:** ex4
2. **Units:** English (kip, inch, radian, Fahrenheit).
3. **Type of mesh generation:** direct
4. **Control variables:**
   Number of members: 3
   Number of joints: 4
   Number of load cases: 1
   Self-weight of the structure is not considered.
5. **Output options:** print input data, print output for all load cases.
6. **Joint coordinates:**

<table>
<thead>
<tr>
<th>Joint number</th>
<th>X-direction</th>
<th>Y-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>-180.00</td>
<td>120.00</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>120.00</td>
</tr>
<tr>
<td>4</td>
<td>120.00</td>
<td>264.00</td>
</tr>
</tbody>
</table>
7. **Joint constraints:** joints 1, 2 and 4 are fixed.
8. **Member incidences:**

<table>
<thead>
<tr>
<th>Member</th>
<th>Start joint</th>
<th>End joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
9. **Internal hinges:**

<table>
<thead>
<tr>
<th>Member</th>
<th>Hinge location</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>start joint</td>
</tr>
</tbody>
</table>

*Program Verification*
10. Material properties:

Material set number: 1

Material type: steel

Assign material property set 1 to all members.

11. Member properties:

Member set number: 1

Member type: AISC section W14×90

Assign member property set 1 to individual members

Member number: 1

Member set number: 2

Member type: AISC section W21×50

Assign member set 1 to a series of members

First member: 2

Last member: 3

Increment: 1

12. Joint Loads: no joint loads

13. Member loads:

Load case: 1

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Load1</th>
<th>Load2</th>
<th>Distance1</th>
<th>Distance2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>trap</td>
<td>0.2</td>
<td>0.3</td>
<td>0.00</td>
<td>120.00</td>
</tr>
<tr>
<td>2</td>
<td>temp</td>
<td>20.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


15. Load descriptions: no load descriptions.

16. Load combinations: no load combinations.
5.5.3 Comparison Of Results

Tables 5.10, 5.11 and 5.12 show the support reactions, joint displacements and local member end forces obtained from the frame analysis program and STAAD III and the percentage difference between the two. The results from the two programs agree within 1.5% for example 4.
Table 5.10 Support reactions for example structure 4

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek Fx (kips)</th>
<th>Erkek Fy (kips)</th>
<th>Erkek Mz (kip-in.)</th>
<th>STAAD III Fx (kips)</th>
<th>STAAD III Fy (kips)</th>
<th>STAAD III Mz (kip-in.)</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.82</td>
<td>13.01</td>
<td>-277.10</td>
<td>14.81</td>
<td>13.03</td>
<td>-278.89</td>
<td>0.07% 0.15% 0.64%</td>
</tr>
<tr>
<td>2</td>
<td>24.99</td>
<td>-1.46</td>
<td>-83.96</td>
<td>25.00</td>
<td>-1.48</td>
<td>-85.24</td>
<td>0.04% 1.35% 1.50%</td>
</tr>
<tr>
<td>4</td>
<td>-9.81</td>
<td>-11.55</td>
<td>-27.21</td>
<td>-9.81</td>
<td>-11.55</td>
<td>-27.22</td>
<td>0.00% 0.00% 0.04%</td>
</tr>
</tbody>
</table>

Table 5.11 Joint displacements for example structure 4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.01285</td>
<td>-0.00203</td>
<td>-0.00030</td>
<td>0.01285</td>
<td>-0.00203</td>
<td>-0.00030</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
</tbody>
</table>

Program Verification
Table 5.12 Local member end forces for example structure 4

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek Axial (kips)</th>
<th>Erkek Shear (kips)</th>
<th>Erkek Moment (kip-in.)</th>
<th>STAAD III Axial (kips)</th>
<th>STAAD III Shear (kips)</th>
<th>STAAD III Moment (kip-in.)</th>
<th>% Difference Axial</th>
<th>% Difference Shear</th>
<th>% Difference Moment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>13.01</td>
<td>-14.82</td>
<td>-277.10</td>
<td>13.03</td>
<td>-14.81</td>
<td>-278.89</td>
<td>0.15%</td>
<td>0.07%</td>
<td>0.64%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-13.01</td>
<td>-15.18</td>
<td>178.70</td>
<td>-13.03</td>
<td>-15.19</td>
<td>181.22</td>
<td>0.15%</td>
<td>0.07%</td>
<td>1.39%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>24.99</td>
<td>-1.46</td>
<td>-83.96</td>
<td>25.00</td>
<td>-1.48</td>
<td>-85.24</td>
<td>0.04%</td>
<td>1.35%</td>
<td>1.50%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-24.99</td>
<td>1.46</td>
<td>-178.70</td>
<td>-25.00</td>
<td>1.48</td>
<td>-181.22</td>
<td>0.04%</td>
<td>1.35%</td>
<td>1.39%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>15.16</td>
<td>-0.15</td>
<td>0.00</td>
<td>15.15</td>
<td>-0.15</td>
<td>0.00</td>
<td>0.07%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-15.16</td>
<td>0.15</td>
<td>-27.21</td>
<td>-15.15</td>
<td>0.15</td>
<td>-27.22</td>
<td>0.07%</td>
<td>0.00%</td>
<td>0.04%</td>
</tr>
</tbody>
</table>

Program Verification
5.6 Example Structure 5

5.6.1 Description

The fifth example structure is a plane frame consisting of 20 joints and 28 members. The frame is subjected to two load cases. The first load case consists of uniformly distributed loads acting on the beams. The second load case consists of joint loads. Automatic mesh generation is utilized in the creation of the input data file since the structure is an orthogonal plane frame with equally spaced joints. The structure is shown in Figure 5.5.

5.6.2 Input Data

1. Name of input file: ex5
2. Units: English (kip, inch, radian, Fahrenheit).
3. Type of mesh generation: automatic
4. Control variables:
   Number of stories: 4
   Story height: 180 in.
   Number of bays: 3
   Bay width: 240 in.
   Number of load cases: 2
   Self-weight of the structure is not considered.
5. Output options: print input data, print output for all load cases.
6. Joint coordinates: since automatic mesh generation option is on, joint coordinates are computed directly by the preprocessor.
Figure 5.5 Example structure 5

Program Verification
7. **Joint constraints**: joints 1, 2, 3 and 4 are fixed.

8. **Member incidences**: since automatic mesh generation option is on, member incidences are computed directly by the preprocessor.

9. **Internal hinges**: no internal hinges.

10. **Material properties**:

    Material set number: 1

    Material type: steel

    Assign material property set 1 to all members.

11. **Member properties**:

    Member set number: 1

    Member type: AISC section W12×65

    Assign member property set 1 to individual members

    Member numbers: 1, 4, 8, 11, 15, 18, 22, 25

    Member set number: 2

    Member type: AISC section W14×109

    Assign member property set 2 to individual members

    Member numbers: 2, 3, 9, 10, 16, 17, 23, 24

    Member set number: 3

    Member type: AISC section W10×26

    Assign member property set 3 to individual members

    Member numbers: 5, 6, 7, 12, 13, 14, 19, 20, 21

    Member set number: 4

    Member type: AISC section W10×22

    Assign member property set 4 to individual members

    Member numbers: 26, 27, 28
12. **Joint Loads:**

Load case: 2

<table>
<thead>
<tr>
<th>Joint</th>
<th>Direction</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>x</td>
<td>2</td>
</tr>
</tbody>
</table>

13. **Member loads:**

Load case: 1

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Load1</th>
<th>Load2</th>
<th>Distance1</th>
<th>Distance2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>uniform</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. **Prescribed displacements:** no prescribed displacements.
15. **Load descriptions:**

<table>
<thead>
<tr>
<th>Load case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dead+live</td>
</tr>
<tr>
<td>2</td>
<td>wind from left</td>
</tr>
</tbody>
</table>

16. **Load combinations:** no load combinations.

### 5.6.3 Comparison Of Results

Tables 5.13, 5.14 and 5.15 show the support reactions, joint displacements and local member end forces obtained from the frame analysis program and STAAD III and the percentage difference between the two. The results from the program agree with those from STAAD III within 0.5% for example 5.
Table 5.13 Support reactions for example structure 5

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fx  kips</td>
<td>Fy  kips</td>
<td>Mz  kip-in.</td>
</tr>
<tr>
<td>1</td>
<td>1.60 47.13</td>
<td>-97.54</td>
<td>-1.60 47.13 -97.54</td>
</tr>
<tr>
<td>2</td>
<td>-0.01 96.87</td>
<td>-0.59</td>
<td>-0.01 96.87 -0.59</td>
</tr>
<tr>
<td>3</td>
<td>0.01 96.87</td>
<td>0.59</td>
<td>0.01 96.87 0.59</td>
</tr>
<tr>
<td>4</td>
<td>-1.60 47.13</td>
<td>97.54</td>
<td>-1.60 47.13 97.54</td>
</tr>
</tbody>
</table>

Table 5.14 Joint displacements for example structure 5

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-Trans inch</td>
<td>Y-Trans inch</td>
<td>Z-Rot radian</td>
</tr>
<tr>
<td>5</td>
<td>-0.00178</td>
<td>-0.01532</td>
<td>-0.00054</td>
</tr>
<tr>
<td>10</td>
<td>0.00015</td>
<td>-0.03288</td>
<td>0.00001</td>
</tr>
<tr>
<td>15</td>
<td>0.00069</td>
<td>-0.04229</td>
<td>0.00001</td>
</tr>
<tr>
<td>20</td>
<td>-0.00718</td>
<td>-0.03822</td>
<td>0.00115</td>
</tr>
</tbody>
</table>

Program Verification 102
Table 5.15 Local member end forces for example structure 5

### Load Case 1

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial kips</td>
<td>Shear kips</td>
<td>Moment kip-in.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>47.13</td>
<td>-1.60</td>
<td>-97.54</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-47.13</td>
<td>1.60</td>
<td>-190.00</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>-0.99</td>
<td>12.00</td>
<td>480.30</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0.99</td>
<td>12.00</td>
<td>-480.30</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0.50</td>
<td>11.83</td>
<td>452.00</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-0.50</td>
<td>12.17</td>
<td>-492.20</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>48.48</td>
<td>0.07</td>
<td>2.58</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>-48.48</td>
<td>-0.07</td>
<td>10.00</td>
</tr>
<tr>
<td>25</td>
<td>16</td>
<td>11.62</td>
<td>3.84</td>
<td>273.00</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>-11.62</td>
<td>-3.84</td>
<td>418.20</td>
</tr>
<tr>
<td>28</td>
<td>19</td>
<td>3.84</td>
<td>12.38</td>
<td>508.70</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>-3.84</td>
<td>11.62</td>
<td>-418.20</td>
</tr>
</tbody>
</table>

### Load Case 2

<table>
<thead>
<tr>
<th>Member No.</th>
<th>Joint No.</th>
<th>Erkek</th>
<th>STAAD III</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Axial kips</td>
<td>Shear kips</td>
<td>Moment kip-in.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-3.13</td>
<td>1.28</td>
<td>207.80</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>3.13</td>
<td>-1.28</td>
<td>22.41</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.93</td>
<td>-0.95</td>
<td>-113.60</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>-0.93</td>
<td>0.95</td>
<td>-113.50</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>1.70</td>
<td>-0.99</td>
<td>-118.10</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-1.70</td>
<td>0.99</td>
<td>-118.70</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>0.02</td>
<td>1.35</td>
<td>67.99</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>-0.02</td>
<td>1.35</td>
<td>174.30</td>
</tr>
<tr>
<td>25</td>
<td>16</td>
<td>0.44</td>
<td>0.35</td>
<td>9.60</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>-0.44</td>
<td>-0.35</td>
<td>-52.88</td>
</tr>
<tr>
<td>28</td>
<td>19</td>
<td>0.35</td>
<td>0.44</td>
<td>-53.31</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>-0.35</td>
<td>0.44</td>
<td>-52.88</td>
</tr>
</tbody>
</table>
5.7 Conclusions

The results obtained from the five example structures indicate that the frame analysis program provides accurate analysis results. The only difference in the analysis results between STAAD-III and the frame analysis program is because of the difference in the computation of member loads due to the self-weight of the structures. However, this difference is very small and is not of any real importance.
CHAPTER 6

SUMMARY AND FUTURE EXTENSIONS

6.1 Summary

The main objective of this study was to develop an interactive and user-friendly computer program for the analysis of plane frame structures that could be used on a wide range of workstations. This was achieved by using the XWindow graphical system. An event-driven, window-based program was developed using the XWindow Toolkit. Since the program runs under XWindow it is possible to run the program on a number of different hardware platforms and also across networks.

The program consists of two parts: a preprocessor and a processor. The preprocessor has an interactive and user-friendly interface with pull-down menus and pop-up dialog boxes. The preprocessor provides a convenient way to input and modify data for a structural model.
The processor analyzes the structural model based on the matrix displacement. The processor performs matrix analysis on the structure and computes joint displacements, local member-end forces and support reactions.

Five structures were considered in the verification and evaluation of the frame analysis program. The results of the analysis were compared with the results obtained from a commercial program (STAAD-III) to verify the accuracy of the program. The results indicate that the frame analysis program provides accurate analysis results.

6.2 Future Extensions

A possible modification to the program is to extend the program so that it can analyze space frames and trusses. Another extension would be to incorporate elastic supports and shear and axial force releases in the analysis.

Developing the preprocessor with the Motif user-interface of Open Software Foundation, Inc. would provide a more professional interface.

Another useful extension would be to add graphical capabilities to the program for plotting the structure and labeling the joint and member numbers. This would be very helpful during the creation of the structural model. Postprocessing capabilities like displaying deflection, shear, moment and stress diagrams for the whole structure and single elements would also be useful. Additional postprocessing capabilities that can be incorporated into the program would be to
compute member stresses at specified intermediate sections as well as at the start and end joints. Since the program has been developed in a modular manner, it would not be a difficult task to make the modifications mentioned above.
REFERENCES CITED


REFERENCES CONSULTED


APPENDIX A

VERIFICATION BY HAND CALCULATIONS

In this appendix, the results for example structures 1 through 4 are calculated by hand to verify the accuracy of the results obtained by the frame analysis program. The steps followed during the hand calculations are also presented.

A.1 Example Structure 1

The structure is shown in Figure 5.1. The analysis is done in the following order:

1. Unknowns
   \[ q_k, k=1, 2, ..., 12 \]

2. Member Models
   \[ \vec{f} = kd \hat{d} + \vec{f}, \quad i=1, 2, ..., 6 \]
   \[ \vec{f} = \text{local member force vector} \]
   \[ k = \text{member stiffness matrix} \]
   \[ \hat{d} = \text{local member displacement vector} \]
   \[ \vec{f} = \text{local fixed-end force vector} \]
\[
M = \begin{bmatrix}
0 & 0 & 1 & 1 & 4 & 7 \\
0 & 0 & 2 & 2 & 5 & 8 \\
0 & 0 & 3 & 3 & 6 & 9 \\
1 & 4 & 4 & 7 & 10 & 10 \\
2 & 5 & 5 & 8 & 11 & 11 \\
3 & 6 & 6 & 9 & 12 & 12
\end{bmatrix}; \quad M=\text{member code matrix}
\]

\[\hat{\mathbf{f}}^i=\lambda^iT \hat{\mathbf{p}}^i=\text{global fixed-end force vector}\]

\[\hat{\mathbf{F}}^{(i)}=\text{generalized fixed-end force vector (obtained from } \hat{\mathbf{f}}^i \text{ using } M)\]

Load case 1:

\[\hat{\mathbf{F}}^3 = \hat{\mathbf{F}}^6 = \begin{bmatrix} 0 & 12 & 240 & 0 & 12 & -240 \\ 0 & 12 & 240 & 0 & 12 & -240 \end{bmatrix}\]

3. System Model

\[\hat{\mathbf{Q}} = \sum_{i=1}^{NE} \hat{\mathbf{F}}^{(i)}\]

\[\hat{\mathbf{Q}} = \overline{\hat{\mathbf{Q}}} \cdot \hat{\mathbf{Q}}\]

\[\hat{\mathbf{Q}} = \text{equivalent joint load vector; } \overline{\hat{\mathbf{Q}}} = \text{actual joint load vector; } \hat{\mathbf{Q}} = \text{fixed-joint load vector.}\]

Load case 1:

\[\hat{\mathbf{Q}} = \begin{bmatrix} 0 & 12 & 240 & 0 & 12 & -240 \\ 0 & 12 & 240 & 0 & 12 & -240 \end{bmatrix}\]

\[\overline{\hat{\mathbf{Q}}} = 0\]

\[\hat{\mathbf{Q}} = \begin{bmatrix} 0 & -12 & -240 & 0 & -12 & 240 \\ 0 & -12 & -240 & 0 & -12 & 240 \end{bmatrix}\]

Load case 2:

\[\overline{\hat{\mathbf{Q}}} = \begin{bmatrix} 1.5 & 0 & 0 & 1.5 & 0 & 0 \\ 1.5 & 0 & 0 & 1.5 & 0 & 0 \end{bmatrix}\]

\[\hat{\mathbf{Q}} = 0\]

\[\hat{\mathbf{Q}} = \begin{bmatrix} 1.5 & 0 & 0 & 1.5 & 0 & 0 \\ 1.5 & 0 & 0 & 1.5 & 0 & 0 \end{bmatrix}\]
System stiffness matrix:

\[
K = \begin{bmatrix}
1464 & 0 & 0 & -1440 & 0 & 0 & -12 & 0 & -720 & 0 & 0 & 0 \\
2892 & 720 & 0 & -12 & 720 & 0 & -1440 & 0 & 0 & 0 & 0 & 0 \\
172800 & 0 & -720 & 28800 & 720 & 0 & 28800 & 0 & 0 & 0 & 0 & 0 \\
1464 & 0 & 0 & 0 & 0 & 0 & -12 & 0 & -720 & 0 & 0 & 0 \\
2892 & -720 & 0 & 0 & 0 & 0 & -1440 & 0 & 0 & 0 & 0 & 0 \\
172800 & 0 & 0 & 0 & 720 & 0 & 28800 & 0 & 0 & 0 & 0 & 0 \\
1452 & 0 & 720 & -1440 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1452 & 720 & 0 & -12 & 720 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
115200 & 0 & -720 & 28800 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1452 & 0 & 720 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
115200 & \\
\end{bmatrix}
\]

where \( \alpha = 0.86806 \)

System model:

\[ Kq = Q \]

4. Solution

Load case 1:

\[
q[1] = -0.00068; \quad q[2] = -0.01920; \quad q[3] = -0.00137; \quad q[4] = 0.00068; \quad q[5] = -0.01920; \\
q[6] = -0.00137; \quad q[7] = 0.00103; \quad q[8] = -0.02880; \quad q[9] = -0.00276; \quad q[10] = -0.00103; \\
q[11] = -0.02880; \quad q[12] = 0.00276
\]

Load case 2:

\[
q[1] = 0.46470; \quad q[2] = 0.00431; \quad q[3] = -0.00295; \quad q[4] = 0.46470; \quad q[5] = -0.00431; \\
q[6] = -0.00295; \quad q[7] = 0.87730; \quad q[8] = 0.00575; \quad q[9] = -0.00153; \quad q[10] = 0.87730; \\
q[11] = -0.00575; \quad q[12] = -0.00153
\]
The joint displacements are identical to the joint displacements obtained from the frame analysis program. Local member-end forces and joint forces are computed from the joint displacements. Since joint displacements are the same for hand calculations and the program, the local member-end forces and joint forces are also identical.

### A.2 Example Structure 2

The structure is shown in Figure 5.2. The analysis is done in the following order:

1. **Unknowns**
   
   \[ q_k, k=1, 2, \ldots, 7 \]

2. **Member Models**
   
   \[ \tilde{\mathbf{f}}^i = k \mathbf{d}^i + \tilde{\mathbf{f}}^i, \ i=1, 2, \ldots, 3 \]

   \[ \mathbf{d}^i = \text{local member displacement vector} \]

   \[ k = \text{member stiffness matrix} \]

   \[ \tilde{\mathbf{f}}^i = \text{local fixed-end force vector} \]

   \[ M = \begin{bmatrix} 0 & 1 & 4 \\ 0 & 2 & 5 \\ 0 & 3 & 6 \end{bmatrix} \]

   ; \ M = \text{member code matrix}
\( \hat{f}^i \) = local fixed-end force vector

\( \hat{F}^i = \lambda^i T \hat{f}^i \) = global fixed-end force vector

\( \hat{F}^{(i)} \) = generalized fixed-end force vector (obtained from \( \hat{F}^i \) using M)

Load case 2:

\[
\hat{f}^2 = \hat{F}^2 = \begin{bmatrix}
0 & 12 & 480 & 0 & 12 & -480
\end{bmatrix}
\]

Load case 3:

\[
\hat{f}^1 = \begin{bmatrix}
0.135 & 0 & 0 & 0.135 & 0 & 0
\end{bmatrix}, \quad \hat{F}^1 = \begin{bmatrix}
0 & 0.135 & 0 & 0 & 0.135 & 0
\end{bmatrix}
\]

\[
\hat{f}^2 = \hat{F}^2 = \begin{bmatrix}
0 & 0.26 & 10.40 & 0 & 0.26 & -10.40
\end{bmatrix}
\]

\[
\hat{f}^3 = \begin{bmatrix}
-0.135 & 0 & 0 & -0.135 & 0 & 0
\end{bmatrix}, \quad \hat{F}^3 = \begin{bmatrix}
0 & 0.135 & 0 & 0 & 0.135 & 0
\end{bmatrix}
\]

3. System Model

\[
\hat{Q} = \sum_{i=1}^{\text{NE}} \hat{F}^{(i)}
\]

\[
Q = \overline{Q} \cdot \hat{Q}
\]

Q = equivalent joint load vector; \( \overline{Q} \) = actual joint load vector; \( \hat{Q} \) = fixed-joint load vector.

Load case 1:

\[
\hat{Q} = \begin{bmatrix}
0 & 12 & 480 & 0 & 12 & -480 & 0
\end{bmatrix}
\]

\( \overline{Q} = 0 \)

\[
Q = \begin{bmatrix}
0 & -12 & -480 & 0 & -12 & 480 & 0
\end{bmatrix}
\]

Load case 2:

\[
\overline{Q} = \begin{bmatrix}
10 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\( \hat{Q} = 0 \)

\[
Q = \begin{bmatrix}
10 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Load case 3:

\[
\hat{Q} = \begin{bmatrix}
0 & 0.395 & 10.416 & 0 & 0.395 & -10.416 & 0
\end{bmatrix}
\]

\( \overline{Q} = 0 \)
\[ Q = \begin{bmatrix} 0 & -0.395 & -10.416 & 0 & -0.395 & 10.416 & 0 \end{bmatrix} \]

System stiffness matrix:

\[
K = \begin{bmatrix}
928.0686 & 0 & 332.4259 & -924.3750 & 0 & 0 & 0 \\
852.5798 & 616.2500 & 0 & -5.1354 & 616.2500 & 0 & 0 \\
138491.1111 & 0 & -616.2500 & 49300 & 0 & 39891.1111
\end{bmatrix}
\]

System model:

\[ Kq = Q \]

4. Solution

Load case 1:

\[ q[1] = 0.22990; \quad q[2] = -0.01394; \quad q[3] = -0.00607; \quad q[4] = 0.22860; \quad q[5] = -0.01438; \]

\[ q[6] = 0.00577; \quad q[7] = -0.00479 \]

Load case 2:

\[ q[1] = 2.63800; \quad q[2] = 0.00512; \quad q[3] = -0.00596; \quad q[4] = 2.63500; \quad q[5] = -0.00512; \]

\[ q[6] = -0.00117; \quad q[7] = -0.02137; \]

Load case 3:

\[ q[1] = 0.00498; \quad q[2] = -0.00046; \quad q[3] = -0.00013; \quad q[4] = 0.00495; \quad q[5] = -0.00047; \]

\[ q[6] = 0.00013; \quad q[7] = -0.00010; \]
The joint displacements are identical to the joint displacements obtained from the frame analysis program. Local member-end forces and joint forces are computed from the joint displacements. Since joint displacements are the same for hand calculations and the program, the local member-end forces and joint forces are also identical.

A.3 Example Structure 3

The structure is shown in Figure 5.3. The analysis is done in the following order:

1. Unknowns
   \[ q_1, q_2 \]

2. Member Models
   \[ \mathbf{\tilde{f}^i} = k \mathbf{d}^i + \mathbf{\tilde{f}}, \quad i=1, 2 \]

\[ \mathbf{\tilde{f}} = \text{local member force vector} \]

\[ k = \text{member stiffness matrix} \]

\[ \mathbf{d}^i = \text{local member displacement vector} \]

\[ \mathbf{\tilde{f}} = \text{local fixed-end force vector} \]

\[
\begin{bmatrix}
0 & 0 \\
0 & 0 \\
0 & 0 \\
1 & 2
\end{bmatrix}
\]

\[ M = \begin{bmatrix}
0 & 1 \\
0 & 0 \\
0 & 0 \\
1 & 2
\end{bmatrix} \quad ; \quad M = \text{member code matrix} \]
\( \hat{\mathbf{f}} \) = local fixed-end force vector

\( \hat{\mathbf{F}}^i = \mathbf{i}^T \hat{\mathbf{f}} \) = global fixed-end force vector

\( \hat{\mathbf{F}}^{(i)} \) = generalized fixed-end force vector (obtained from \( \hat{\mathbf{F}}^i \) using M)

Load case 2:

\[ \begin{align*}
\hat{\mathbf{f}}^1 &= \hat{\mathbf{F}}^1 = \begin{bmatrix} 0 & 0.1150 & 0.3067 & 0 & 0.1150 & -0.3067 \end{bmatrix} \\
\hat{\mathbf{f}}^2 &= \begin{bmatrix} 0.3450 & 0 & 0 & 0.3450 & 0 & 0 \\ 0 & -0.3450 & 0 & 0 & 0 & -0.3450 \end{bmatrix}
\end{align*} \]

3. System Model

\( \hat{\mathbf{Q}} = \sum_{i=1}^{\text{NE}} \hat{\mathbf{F}}^{(i)} \)

\( \mathbf{Q} = \overline{\mathbf{Q}} \cdot \hat{\mathbf{Q}} \)

\( \mathbf{Q} \) = equivalent joint load vector; \( \overline{\mathbf{Q}} \) = actual joint load vector; \( \hat{\mathbf{Q}} \) = fixed-joint load vector.

Load case 1:

\( \overline{\mathbf{Q}} = \begin{bmatrix} 0 & Q_2 \end{bmatrix} \)

\( \hat{\mathbf{Q}} = 0 \)

\( \mathbf{Q} = \begin{bmatrix} 0 & Q_2 \end{bmatrix} \)

Load case 2:

\( \hat{\mathbf{Q}} = \begin{bmatrix} -0.3067 & 0 \end{bmatrix} \)

\( \overline{\mathbf{Q}} = 0 \)

\( \mathbf{Q} = \begin{bmatrix} 0.3067 & 0 \end{bmatrix} \)

System stiffness matrix:

\[ \mathbf{K} = \begin{bmatrix} 9333.3333 & 2666.6667 \\ 2666.6667 & 5333.3333 \end{bmatrix} \]

System model:

\[ \mathbf{K} \mathbf{q} = \mathbf{Q} \]

Appendix A: Verification By Hand Calculations 121
4. Solution

Load case 1:
\[ q[1] = -0.00057; \quad q[2] = 0.00200; \]

Load case 2:
\[ q[1] = 0.00004; \quad q[2] = -0.00002; \]

The joint displacements are identical to the joint displacements obtained from the frame analysis program. Local member-end forces and joint forces are computed from the joint displacements. Since joint displacements are the same for hand calculations and the program, the local member-end forces and joint forces are also identical.

**A.4 Example Structure 4**

The structure is shown in Figure 5.4. The analysis is done in the following order:

1. Unknowns
   \[ q_k, \quad k=1, 2, ..., 4 \]

2. Member Models
   \[ \ddot{t}_i = k d^i + \ddot{t}_i, \quad i=1, 2, ..., 3 \]
   \[ \ddot{t}_i = \text{local member force vector} \]
   \[ k = \text{member stiffness matrix} \]
   \[ d^i = \text{local member displacement vector} \]
   \[ \ddot{t}_i = \text{local fixed-end force vector} \]
\[
M = \begin{bmatrix}
0 & 0 & 1 \\
0 & 0 & 2 \\
0 & 0 & 4 \\
1 & 1 & 0 \\
2 & 2 & 0 \\
3 & 3 & 0 \\
\end{bmatrix}; \; M = \text{member code matrix}
\]

$\hat{\mathbf{f}}$ = local fixed-end force vector
$\hat{\mathbf{p}}^i = \hat{\alpha}^T \mathbf{f}$ = global fixed-end force vector
$\hat{\mathbf{F}}^{(i)}$ = generalized fixed-end force vector (obtained from $\hat{\mathbf{p}}^i$ using $M$)

Load case 1:
$\hat{\mathbf{f}}^1 = \begin{bmatrix} 0 & -13.80 & -288 & 0 & -16.20 & 312 \end{bmatrix}^T$
$\hat{\mathbf{F}}^1 = \begin{bmatrix} 13.80 & 0 & -288 & 16.2 & 0 & 312 \end{bmatrix}$
$\hat{\mathbf{f}}^2 = \hat{\mathbf{F}}^2 = \begin{bmatrix} 55.419 & 0 & 0 & -55.419 & 0 & 0 \end{bmatrix}^T$

3. System Model
$\hat{\mathbf{Q}} = \sum_{i=1}^{NE} \hat{\mathbf{F}}^{(i)}$

$\mathbf{Q} = \hat{\mathbf{Q}} \cdot \hat{\mathbf{Q}}$

$\mathbf{Q}$ = equivalent joint load vector; $\hat{\mathbf{Q}}$ = actual joint load vector; $\hat{\mathbf{Q}}$ = fixed-joint load vector.

Load case 1:
$\hat{\mathbf{Q}} = 0$
$\hat{\mathbf{Q}} = \begin{bmatrix} -39.2190 & 0 & 312.0000 & 0 \end{bmatrix}$
$\mathbf{Q} = \begin{bmatrix} 39.2190 & 0 & -312.0000 & 0 \end{bmatrix}$
System stiffness matrix:
\[
K = \begin{bmatrix}
3532.2770 & 1092.9157 & 12071.2500 & -3743.5045 \\
7826.3747 & -5284.4444 & 3119.5871 \\
1599833.3333 & 0 \\
\text{symmetry} & 608943.4010
\end{bmatrix}
\]

System model:

\[Kq = Q\]

4. Solution

Load case 1:

\[q[1] = 0.01285; \quad q[2] = -0.00203; \quad q[3] = -0.00030; \quad q[4] = 0.00009\]

The joint displacements are identical to the joint displacements obtained from the frame analysis program. Local member-end forces and joint forces are computed from the joint displacements. Since joint displacements are the same for hand calculations and the program, the local member-end forces and joint forces are also identical.
Erkan Erkek was born on December 1, 1967 in Konya-Eregli, Turkey. He graduated from Istanbul American Robert High School in 1985. He completed his undergraduate degree in Civil Engineering at Istanbul Technical University in July 1989. In January 1990, he enrolled in the graduate program in the Structures Division of the Civil Engineering Department at Virginia Tech where he earned an M.S. Degree in Civil Engineering in April 1992.