BEHAVIOR OF REINFORCED CONCRETE COLUMNS SUBJECTED TO AXIAL AND LATERAL LOAD REVERSALS

APPROVED:

Michael E. Kreger

James O. Jirsa

To my parents, who are the finest people I have ever met,

and to God, for blessing me with them.

BEHAVIOR OF REINFORCED CONCRETE COLUMNS SUBJECTED TO CYCLIC AXIAL AND LATERAL LOAD REVERSALS

by

LEO EDWARD LINBECK, III, B.A., B.S.C.E.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 1987

ACKNOWLEDGMENTS

The experimental tests performed in this study was conducted at the Phil M. Ferguson Structural Engineering Laboratory at the Balcones Research Center of the University of Texas at Austin. Financial support of the project was provided by the National Science Foundation, Grant No. CEE-8404578.

This project was conducted under the direction of Dr. Michael E. Kreger. I would like to express his thanks to Dr. Kreger for his valuable support and guidance through every phase of the program. At times, when I felt that my wheels were spinning, he would get me going again, always in the right direction. His patient tenacity helped precipitate my transformation from a sloppy student to a thourough engineer, and his friendship made the transformation bearable.

I would also like to thank some other members of the Structures faculty at the University of Texas, especially: Dr. James O. Jirsa, who is living proof that being the best at what you do is not incompatible with being a gentleman, for his invaluable comments and suggestions about this thesis; Dr. John E. Breen, whose open door was often filled with my lurking presence, for showing me that engineering is more than an academic discipline; Dr. Karl H. Frank, whose clarity of thought is matched only by his breadth of knowledge, for his assistance

in many facets of the experimental program; Dr. Jose Roesset, for demonstrating that good engineering is simple and elegant; and Dr. Joseph A. Yura, the consummate teacher, for sharing with me his profound understanding of engineering, people, and life.

In my rather short life, I have had the good fortune of being around many exemplary people, but none better than the graduate students here at the University of Texas. Although there are many of them, I would like to especially mention the following: Marc Badoux, Jack Burgess, Tommy Bush, Lisa Carter, Lex Collins, Dominic Kelly, Bob MacGregor, Troy Madeley, Vince Oue, David Powell, David Sanders, Tauqir Sheikh, Mike Stallings, Dan Stoppenhagen, Paul Tikalsky, Robert Treece, and Akbar Vasseghi. They have provided me with friendship, assistance, and enthusiastic support when I needed it most. The graduate students and faculty in the Structures section is a very special group of people. If I am a better person now than when I arrived at Texas, the credit is all theirs; if not, the failure is all mine.

Work performed at the Ferguson Lab would not have been possible were it not for the work of a very dedicated and competant staff: Pat Ball, Sharon Cunningham, Maxine DeButts, Jean Gehrke, Laurie Golding, Gorham Hinckley, Cecil Jones, Dick Marshall, and Blake Stassney. Special thanks goes to Alex Tahmassebi, for his computer expertise and continuing friendship.

Finally, my deepest thanks goes to my family: my parents Leo and Connie Linbeck; my sisters, Tammy, Betsy, and Mary Clark; and my brothers, John, Andrew, and Patrick. Their constant support was a tremendous source of strength; they will always have my love.

L. E. L., III

Austin, Texas

owing to the path dependency of concrete which was not accounted for in the analytical model.

Lateral column strength was found to be reasonable predicted using traditional design methods, but lateral stiffness was overestimated by commonly used methods of calculating stiffness. Based on the limited number of tests performed in this study, a more reasonable value of $0.3E_cI_g$ was recommended. Potential topics for future research were discussed.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1

OVERVIEW

Reinforced concrete columns which form part of a lateral force resisting system for use in a region of high seismic risk may experience complex variations in shear as well as axial load during a seismic event. While several experimental studies have been performed to determine the behavior of reinforced concrete columns subjected to shear reversals under constant axial load, very little research has been conducted on the behavior of columns subjected to cyclic shear reversals accompanied by cyclic variations in axial load. The objective of the current investigation is to study the response of reinforced concrete columns subjected to cyclic changes in shear and axial load.

There are three principal components to this investigation. The first, which is developed in Chapter 2, is to discuss representative load histories for exterior columns in selected reinforced concrete structures subjected to substantial lateral forces.

The second component of this investigation is an experimental program, which is described in Chapters 3 and 4. Four approximately half-scale reinforced concrete column specimens were fabricated and tested at the Phil M. Ferguson

Structural Engineering Laboratory in the Balcones Research Center of the University of Texas at Austin. One of three load histories, representative of column loads in three different types of structures, was imposed on each specimen, and the response was recorded and analyzed. In Chapter 3, the development and execution of the experimental programis described, and Chapter 4 contains a presentation of the results of that test program.

The third and final component of this investigation was the development of a simple computer-based analytical model for the monotonic response of a reinforced concrete column subjected to varying axial and lateral loads. Constitutive relationships for steel and concrete behavior are used to generate a model for behavior of a reinforced concrete section, which in turn is used to determine the response of an entire column. The theoretical development of these models is treated in Chapter 5, and results are compared with experimental data and evaluated in Chapter 6.

In Chapter 7, important findings of each part of the investigation are summarized, and implications of those findings are described. The appendix contains a listing of the computer program written to implement the analytical model.

CHAPTER 2

BACKGROUND AND OBJECTIVES

2.1 Overview

This chapter is composed of three parts: a discussion of the behavior of reinforced concrete structural systems and the loads experienced by first-story columns, a review of previous related research on reinforced concrete columns, and an outline of the experimental program presented in later chapters.

2.2 Discussion of Reinforced Concrete Structural Systems

In this section, a number of reinforced concrete structural systems are investigated, with emphasis on behavior under lateral loads resulting from an earthquake. The focus of the investigation is to qualitatively examine loads experienced by base columns when the system is subjected to strong ground motion. These loads will then be used as input in the development of loading programs for the experimental program.

Two structural systems are examined: a typical plane frame and a staggered shear wall-frame system. The plane frame was chosen because it is the most common two-dimensional structural system used. The staggered shear wall-frame system was chosen because of the severe loads to which some columns may be exposed as a result of the complex, three-dimensional structure.

The interaction of the two orthogonal structural systems which comprise this complex system has a profound effect on column loads.

2.2.1. Plane frame. Two typical plane frames are shown in Fig. 2.1. The difference between the frames is frame 2.1a has equal-width bays while frame 2.1b has bay widths which vary. The effect this difference has on the response of the frame will be examined later. For now the discussion will focus on frames with equal-width bays.

Forces imposed on frames by strong ground motion are generally represented by lateral loads of the type shown in Fig. 2.2. The lateral load on the frame increases linearly from the base to the top, and the total lateral load and overturning moment are resisted at the base. For analysis purposes, the lateral load is treated as a concentrated load acting on each floor. When a frame is subjected to this type of lateral load, it deforms as shown in Fig. 2.3 (the deformation is exaggerated for clarity). Beam-column joints are assumed to experience rigid body rotation and both beams and columns deform in double curvature.

In a frame which deforms as in Fig. 2.3, interior and exterior columns are not subjected to the same loads, even though they deform in the same manner. Figure 2.4 shows a typical beam from the frame in its initial and deformed states. When the beam

TYPICAL PLANE FRAMES

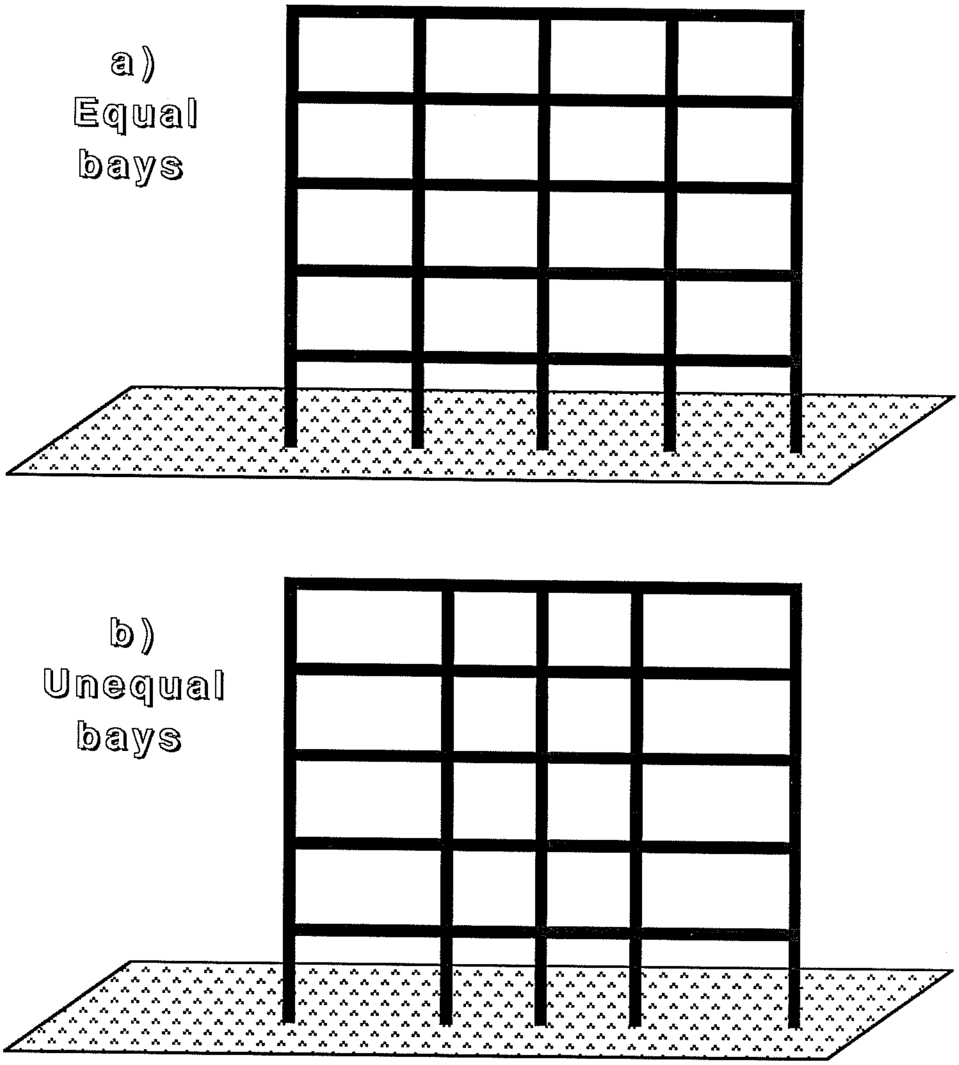


Figure 2.1

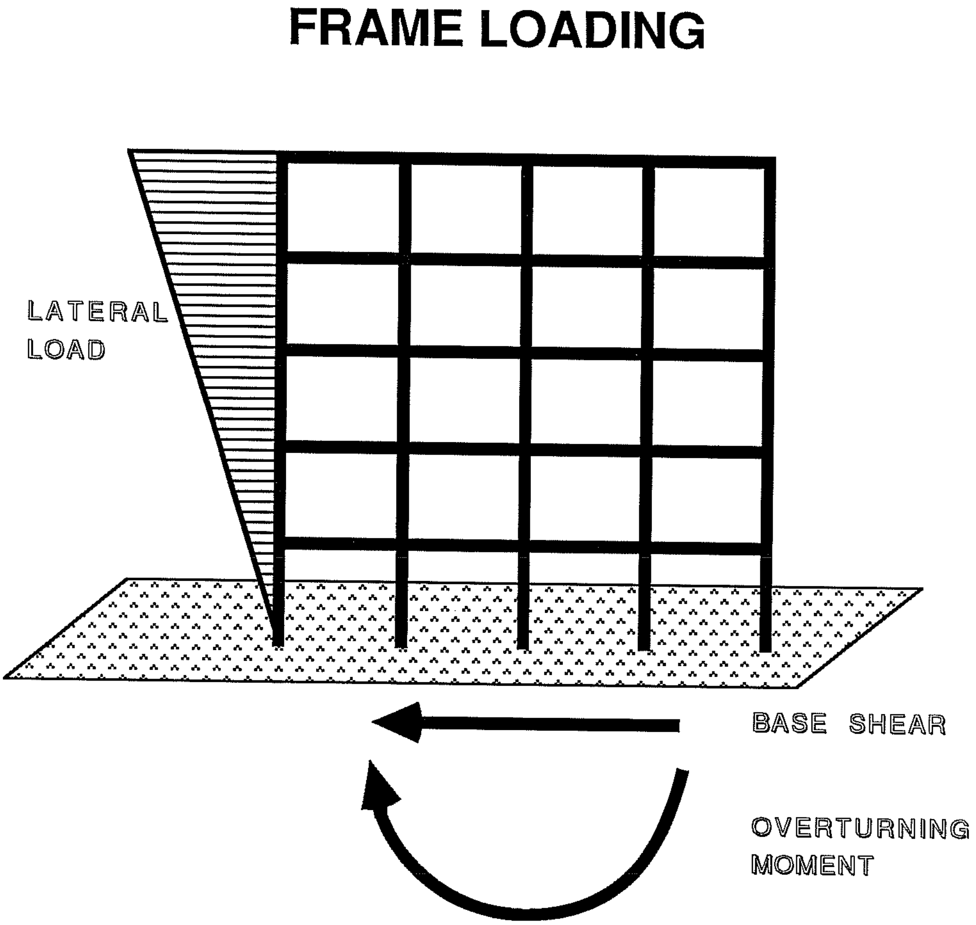


Figure 2.2

PLANE FRAME -
DEFORMED SHAPE

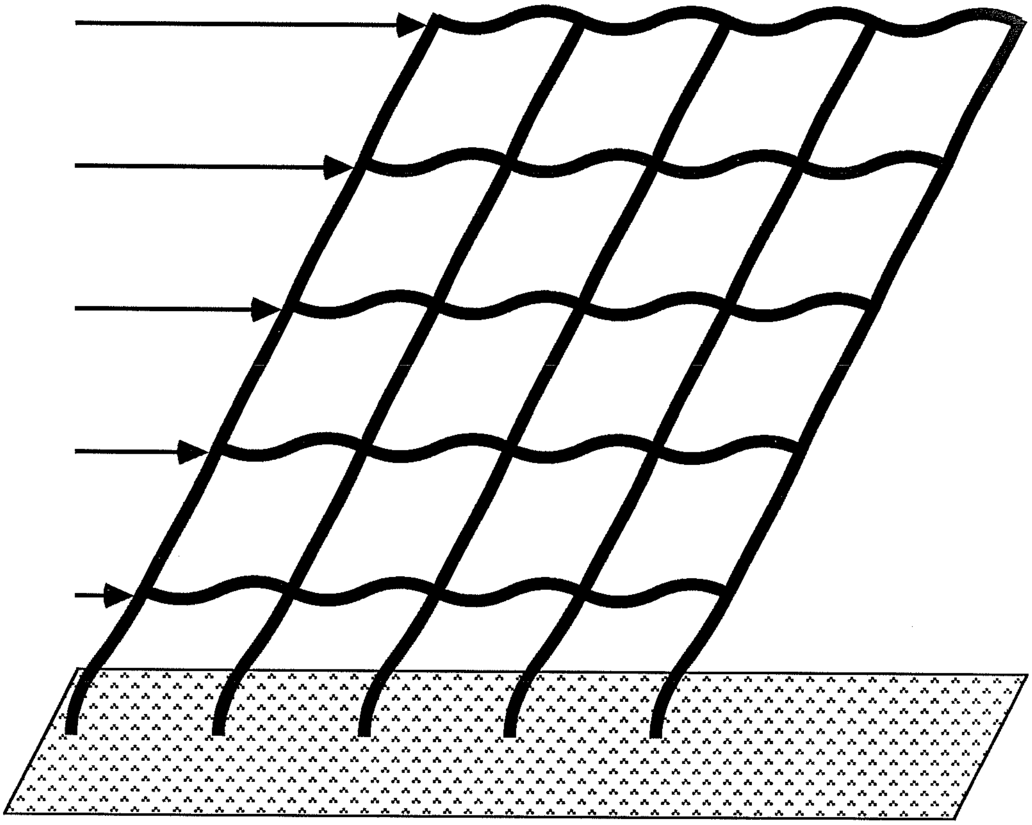
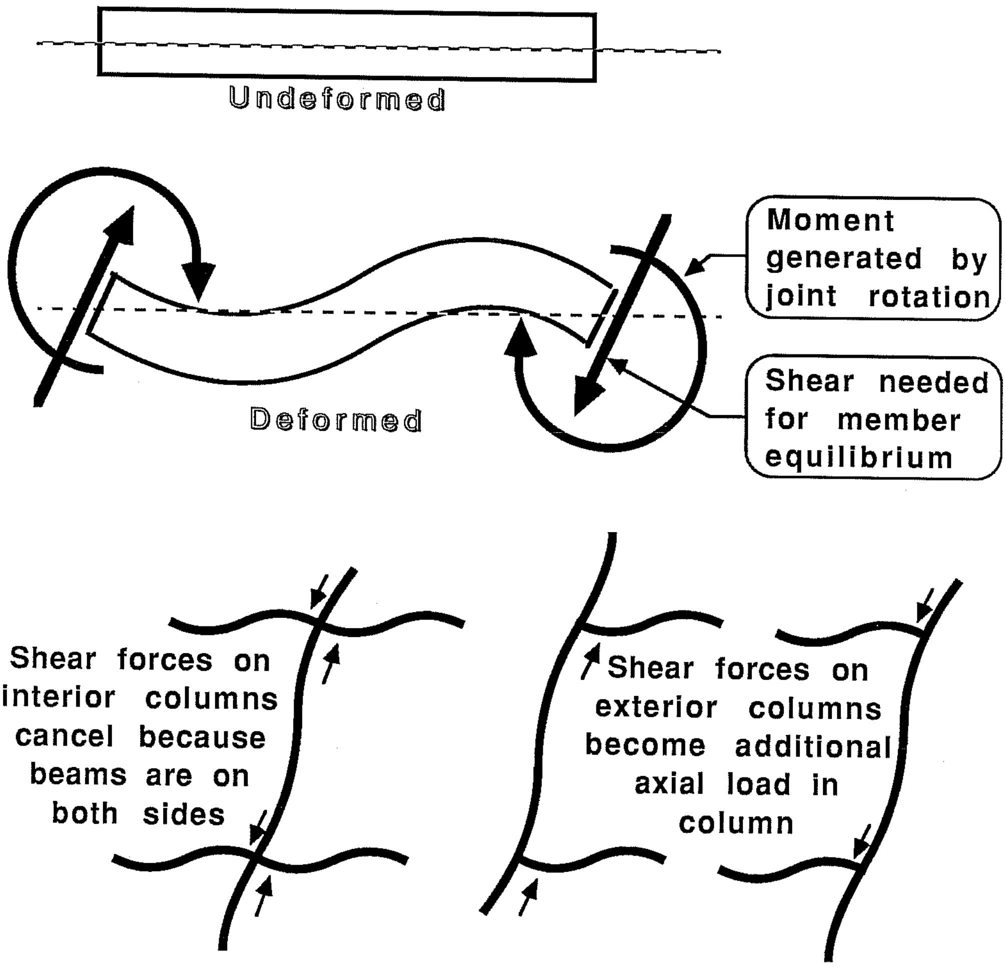


Figure 2.3

BEAM DEFORMATION AND ITS EFFECT ON COLUMN AXIAL LOAD



Moment generated by joint rotation

Shear needed for member equilibrium

Shear forces on interior columns cancel because beams are on both sides

Shear forces on exterior columns become additional axial load in column

Figure 2.4

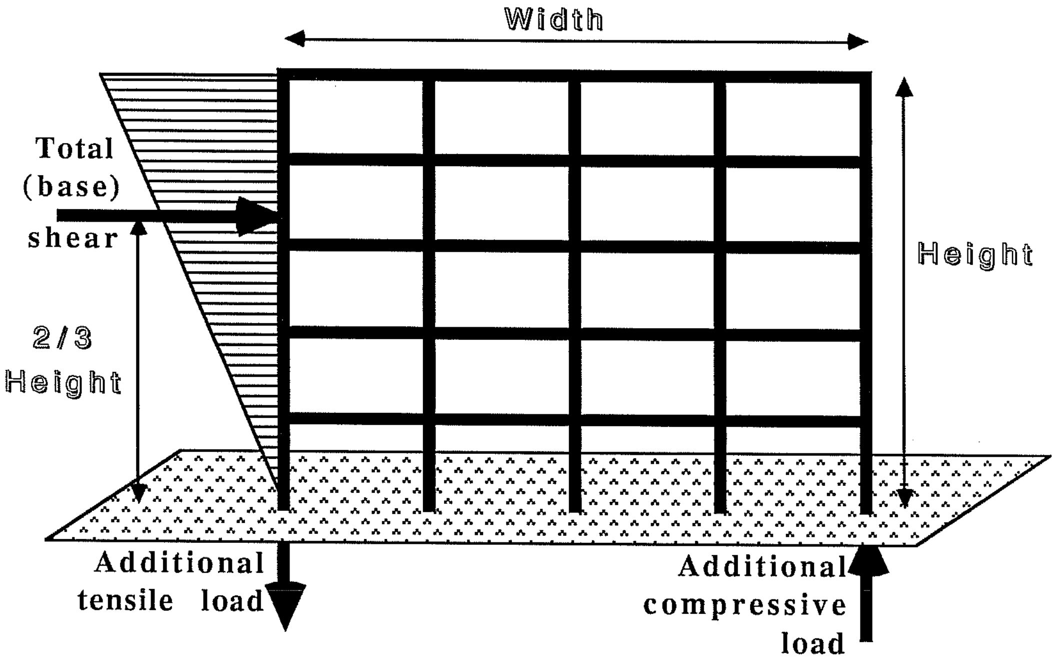
deforms in double curvature, due to joint rotation at each end, shear forces develop because of vertical restraint by the column. At interior joints, these forces tend to cancel each other because there is a beam on each side of the joint. At exterior joints, however, there is only one beam framing into the joint, therefore the shear force must be equilibrated by an axial force in the column. Consequently exterior columns not only resist an axial force due to gravity loads but also an axial force due to lateral loads. This additional axial load is the sum of the shear forces in all beams above the column.

Additional axial load due to unbalanced beam shears also occurs at interior joints where the beams on each side of the joint are not equally stiff -- for instance, a frame with unequal bay widths (see Fig. 2.1b). The change in axial load in these interior columns, however, would not be as significant as the change in exterior columns for two reasons. First, there is some cancelling of shear forces because there are beams framing into both sides of the column. Second, interior columns support gravity load from two bays, whereas exterior columns support load from only one. Therefore, any change in axial load in an interior column is less significant relative to the gravity load than an axial load change in an exterior column.

In an exterior column, this additional axial load can be quite significant. Idealistically, for an exterior column at the base of a frame, the axial load due to lateral loading can be as large as the total overturning moment on the frame divided by the width of the frame. Assuming the lateral load has a triangular distribution, the overturning moment on the frame is the base shear times two-thirds the frame height. Therefore, the additional axial load on the exterior columns of a frame is the total base shear on the frame times two-thirds its aspect ratio (height divided by width). This is illustrated in Fig. 2.5. The additional axial load is a tensile load on one exterior column and a compressive load on the other exterior column. When superimposed on the axial load due to gravity loading, the additional tensile load may cause net tension in the column, and the additional compressive load may push the total axial compression above the axial load at the balanced failure condition (when steel yields and concrete reaches its maximum usable strain simultaneously; also called the balance point).

Each of these loading cases can result in conditions which are not explicitly considered by the designer. If the net axial load on the column due to gravity and lateral loading is quite low, the column becomes very flexible because concrete contribution to the stiffness is reduced. A column in tension will attract less moment than a column under moderate levels of

DETERMINATION OF ADDITIONAL AXIAL LOAD



Aspect ratio = Height / Width

Base shear = sum of Lateral loads

Overturning moment = Base shear * 2/3 * Height

Additional axial load on bottom column = Overturning moment / Width

= Base shear * 2/3 * Aspect ratio

Figure 2.5

compression. In addition, a column in tension has less moment capacity than a column in compression, so less moment may be required for failure of an exterior column. On the other hand, if axial compression on the column reaches levels above the axial load at the balanced failure condition, the column will fail in a brittle mode. Columns which fail at axial loads below the balance point are generally capable of maintaining much of their load capacity at higher deformations, while columns which fail at axial loads above the balance point lose most of their capacity because a significant portion of the column section is crushed when the section fails. Sudden loss of column capacity because of brittle failure may result in a local collapse and threaten the integrity of the entire structure.

Load histories which would be expected for columns in a typical frame are discussed below and depend on two factors: the location of the column in the frame and the failure mechanism for the frame, which is related to the relative flexural strengths of beams and columns. Interior columns would probably have load histories like that shown in Fig. 2.6 because beam shears do not contribute significant axial load relative to gravity load. Axial load would be maintained at or near the gravity load level while shear and moment would vary with lateral loading. Therefore, axial load-vs-lateral load and axial load-vs-moment

LOAD HISTORY
INTERIOR COLUMN

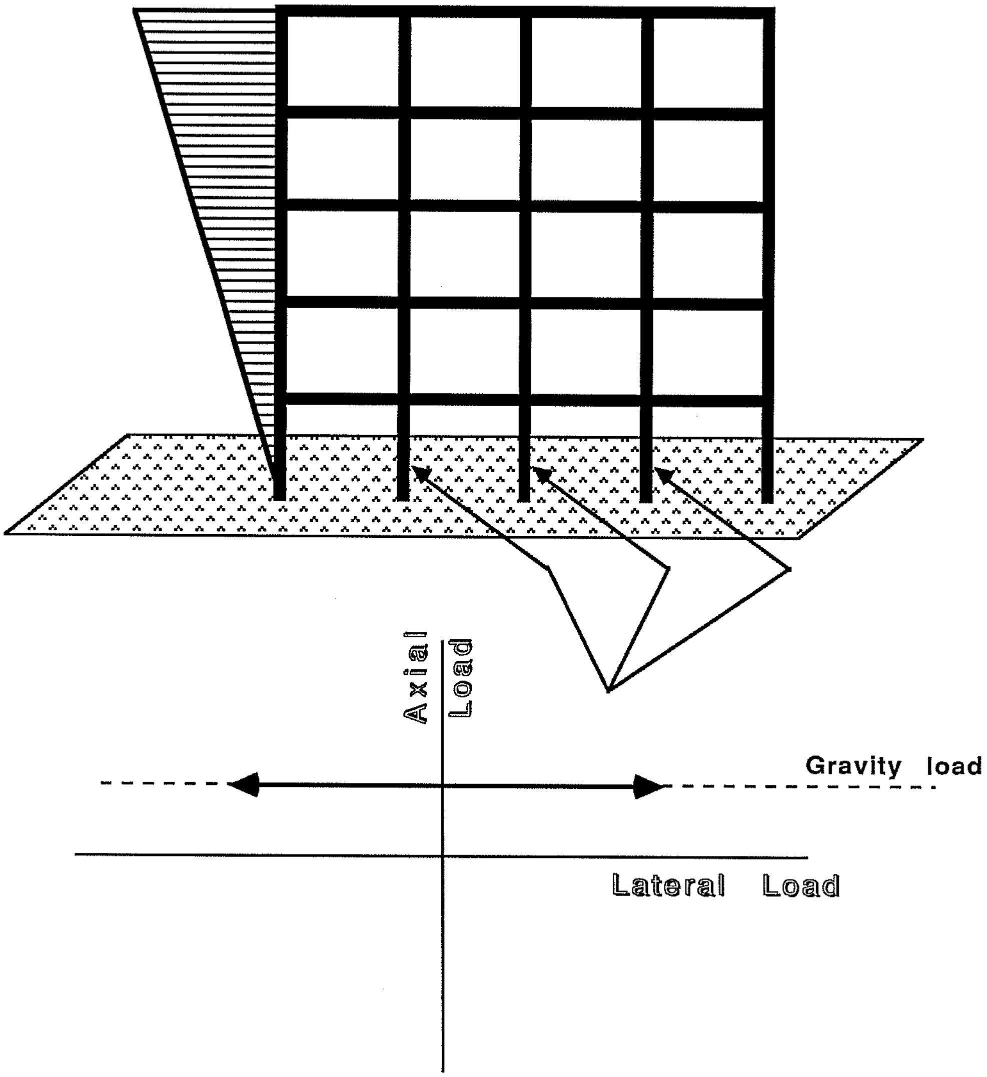


Figure 2.6

load paths would be horizontal lines. These load paths would be valid for both elastic and inelastic column and beam behavior. Even for frames with unequal bays, where shear forces from beams on each side of interior columns do not completely cancel, changes in axial load would generally be small relative to gravity load. Consequently, the load path would be nearly horizontal so long as beams framing into an interior joint do not have radically different stiffnesses.

Load paths for exterior columns are not as simple as for interior columns because of the additional axial load imposed on exterior columns from lateral loading on the frame. The column load path is not only a function of applied loads, but also depends on the behavior of the entire frame. When an exterior column and all beams above it behave elastically, the load path imposed on the column is like the one shown in Fig. 2.7. As lateral loads change, both the moment and axial load on the column change. The change in moment divided by the change in axial load can be considered an eccentricity relative to the gravity load on the column. This relative eccentricity is a function of the total stiffness of all beams above the column and the bending stiffness of the column itself. As long as the beams and columns are elastic, this relative eccentricity is constant and the load path is defined by a straight line.

LOAD HISTORY
EXTERIOR COLUMN
ELASTIC BEHAVIOR

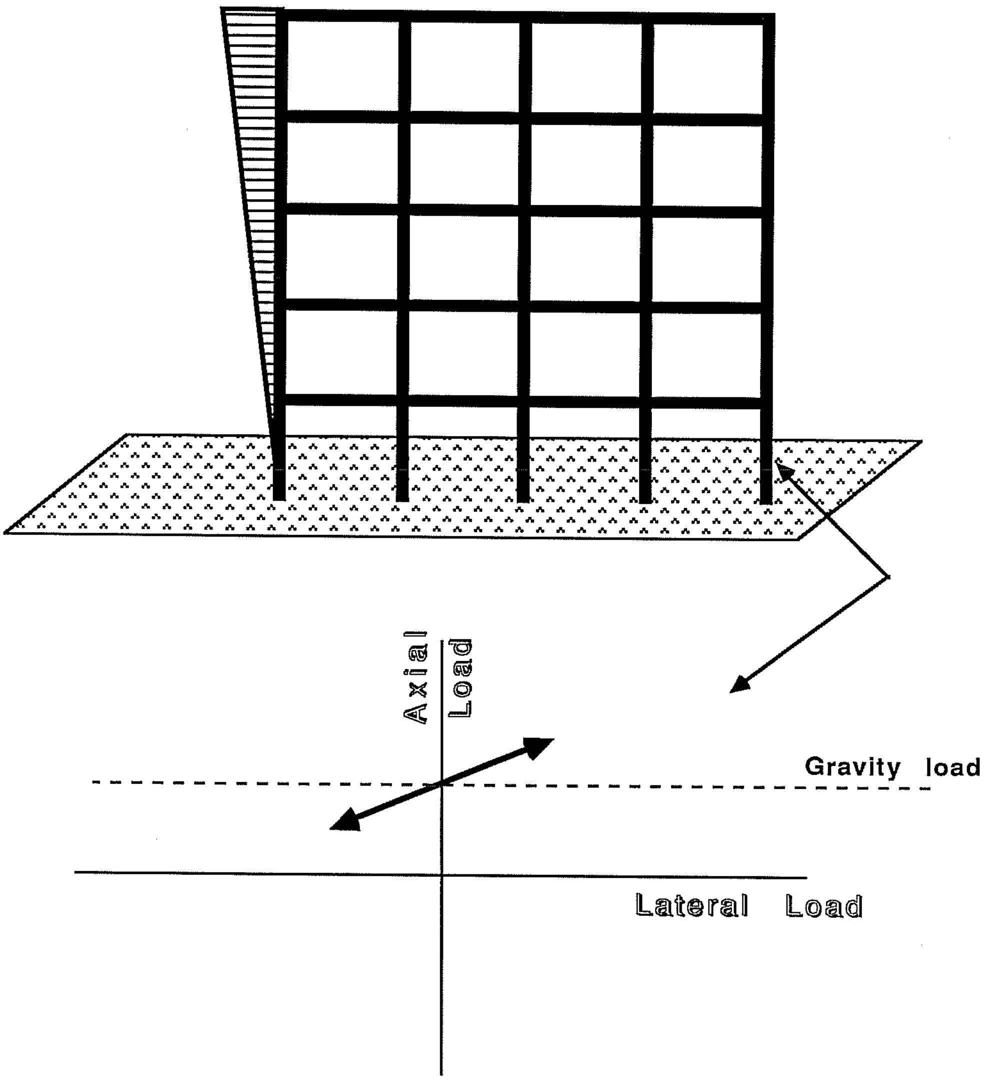


Figure 2.7

Under strong ground motion, however, it is almost certain that some members of the frame will reach deformation levels at which tensile strain in the reinforcement exceeds yield strain. Typically, when this occurs a hinge forms in the member and additional joint rotation produces little or no increase in moment. When designing frames to resist strong ground motion, commom practice is to proportion the beams and columns at a joint so that hinges form in beams before they form in the columns.

If some beams do indeed hinge first, then there is little increase in the additional axial load from those beams, even as lateral loads increase. Therefore, the relative eccentricity of the loading increases and the slope of the axial load-moment path decreases. Figure 2.8 shows a frame in which two hinges above an exterior first-story column have formed, and the corresponding axial load-moment path experienced by the column.

Formation of additional flexural hinges may occur as shown in Fig. 2.9. After hinges have formed in three levels of beams, hinges could form at the base of first-story columns. As beam hinges form, the relative eccentricity increases and the slope of the axial load-moment path decreases (the change in slope is exaggerated for emphasis). Once column hinges develop at the base of the frame, the moment at the base of the exterior columns will not increase, although moment will continue to

LOAD HISTORY EXTERIOR COLUMN - SOME BEAMS HINGED

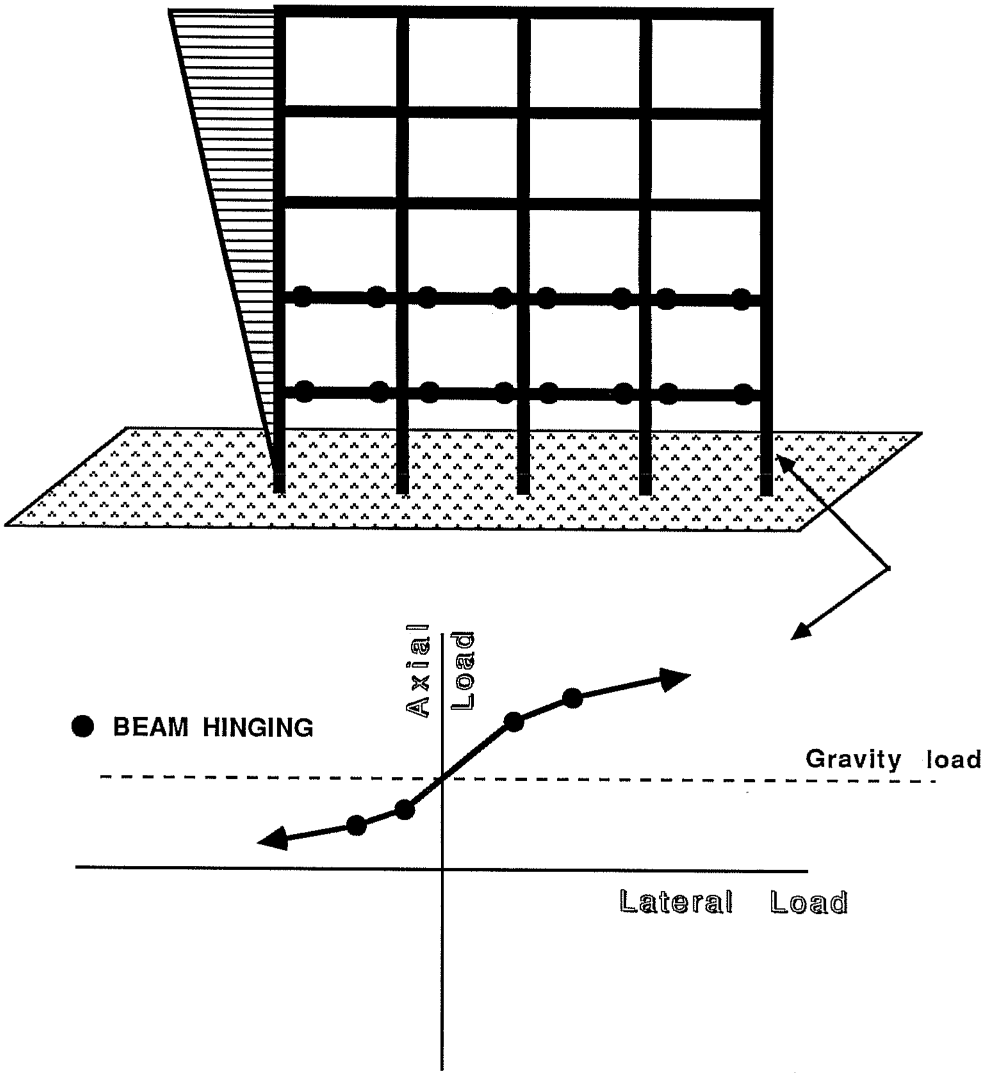


Figure 2.8

LOAD HISTORY EXTERIOR COLUMN - SOME BEAMS AND ALL COLUMNS HINGED AT BASE

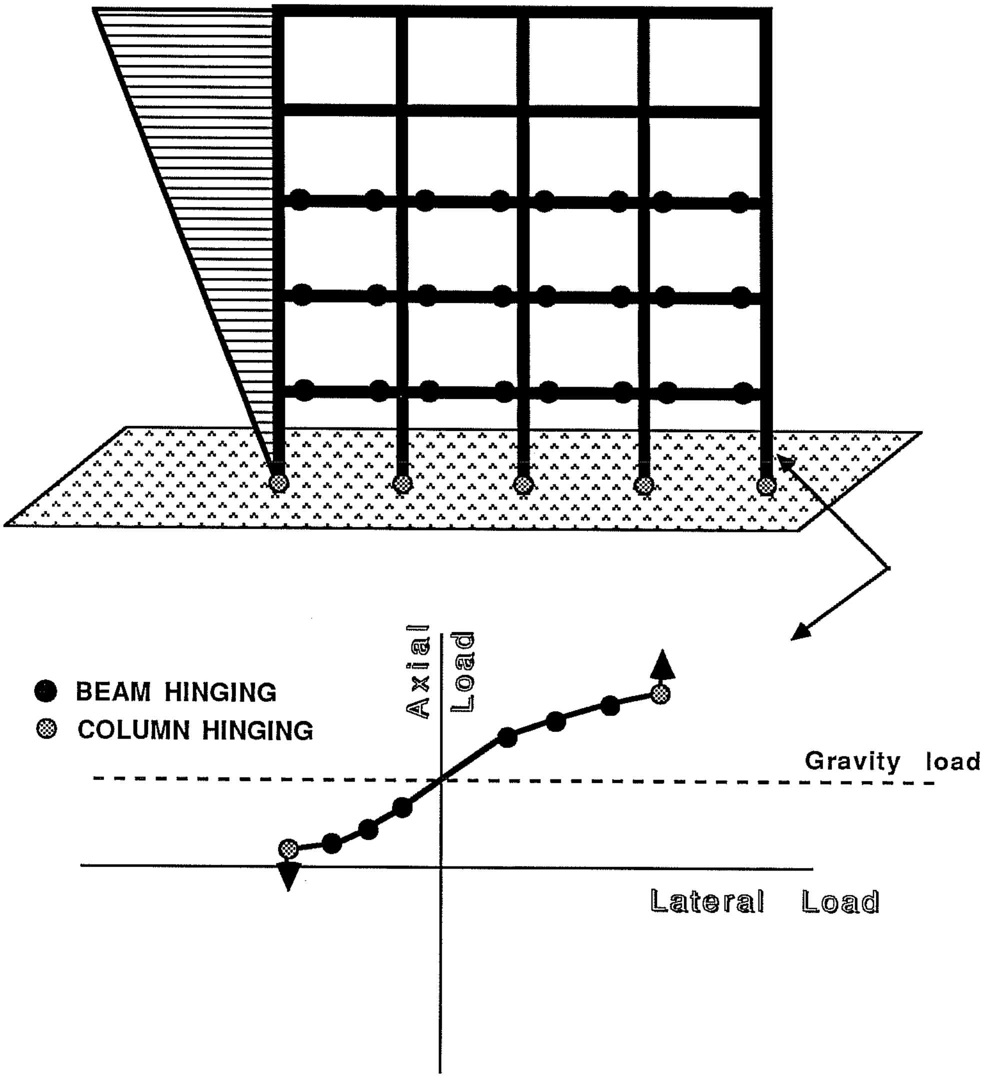


Figure 2.9

increase for the top of the column, and axial load will increase because beams that have not hinged above the column can resist additional shear. One of two possibilities will occur after hinging at the base of first-story columns: remaining beams in the frame will hinge and the mechanism shown in Fig. 2.10 will form, or another hinge will develop in the columns, resulting in a failure mechanism like the one shown in Fig. 2.11. While both of these failure mechanisms are possible, it is far more likely that a frame designed in accordance with modern design codes and recommendations will develop the first mechanism, shown in Fig. 2.10.

In some frames, such as a well designed low-rise frame, it is possible that all beams will hinge before columns hinge at the base. When this occurs, any subsequent lateral deformation in the frame causes no additional axial load on an exterior column because no additional shear can be resisted by the beams. Therefore, axial load remains constant after all beams above the column have hinged, as shown in Fig. 2.12. Eventually, after all beams have hinged and hinges have developed at the base of columns, a mechanism is formed. The resulting axial load-moment path would be similar to the one shown in Fig. 2.10.

Although each of the external column axial load-moment paths described above is somewhat different, they all follow the

LOAD HISTORY EXTERIOR COLUMN - ALL BEAMS AND COLUMNS HINGED

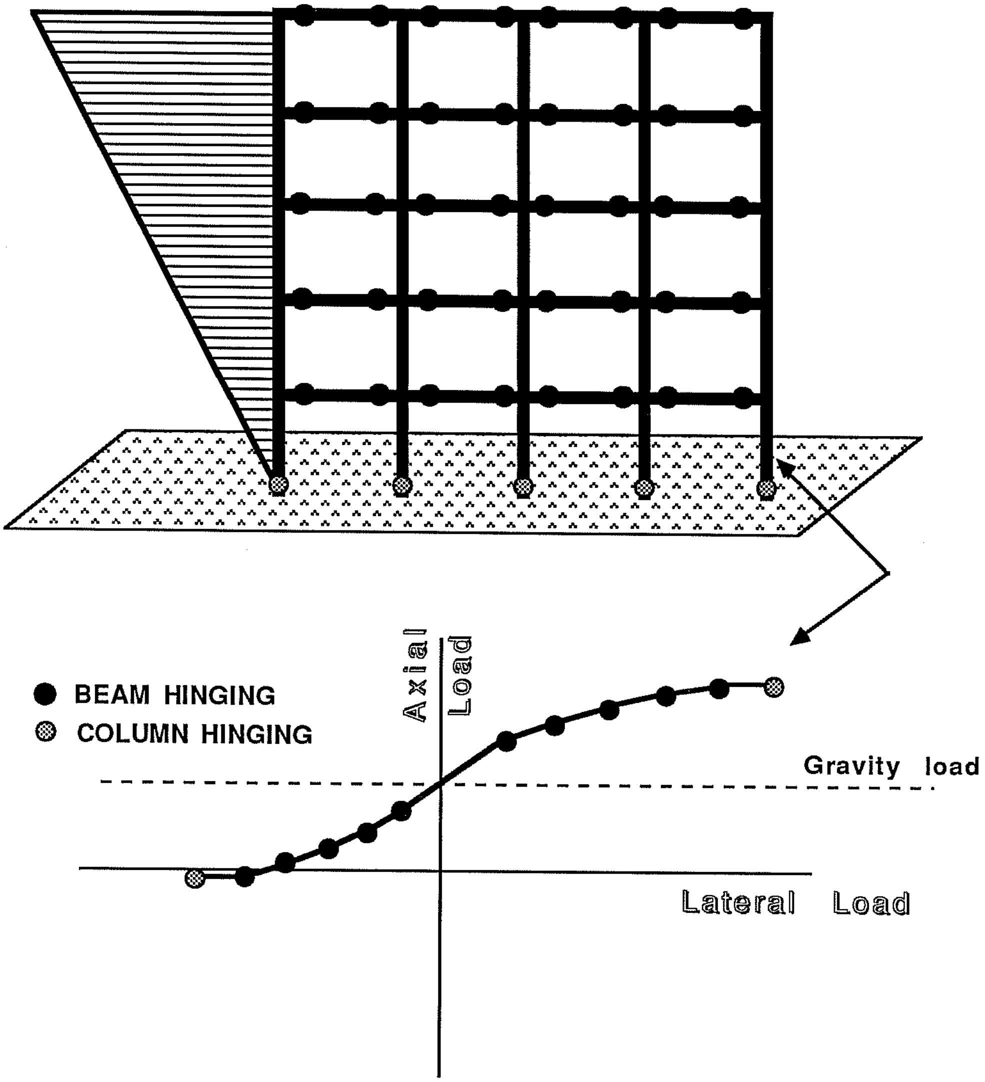


Figure 2.10

LOAD HISTORY EXTERIOR COLUMN - SOME BEAMS AND ALL COLUMNS HINGED

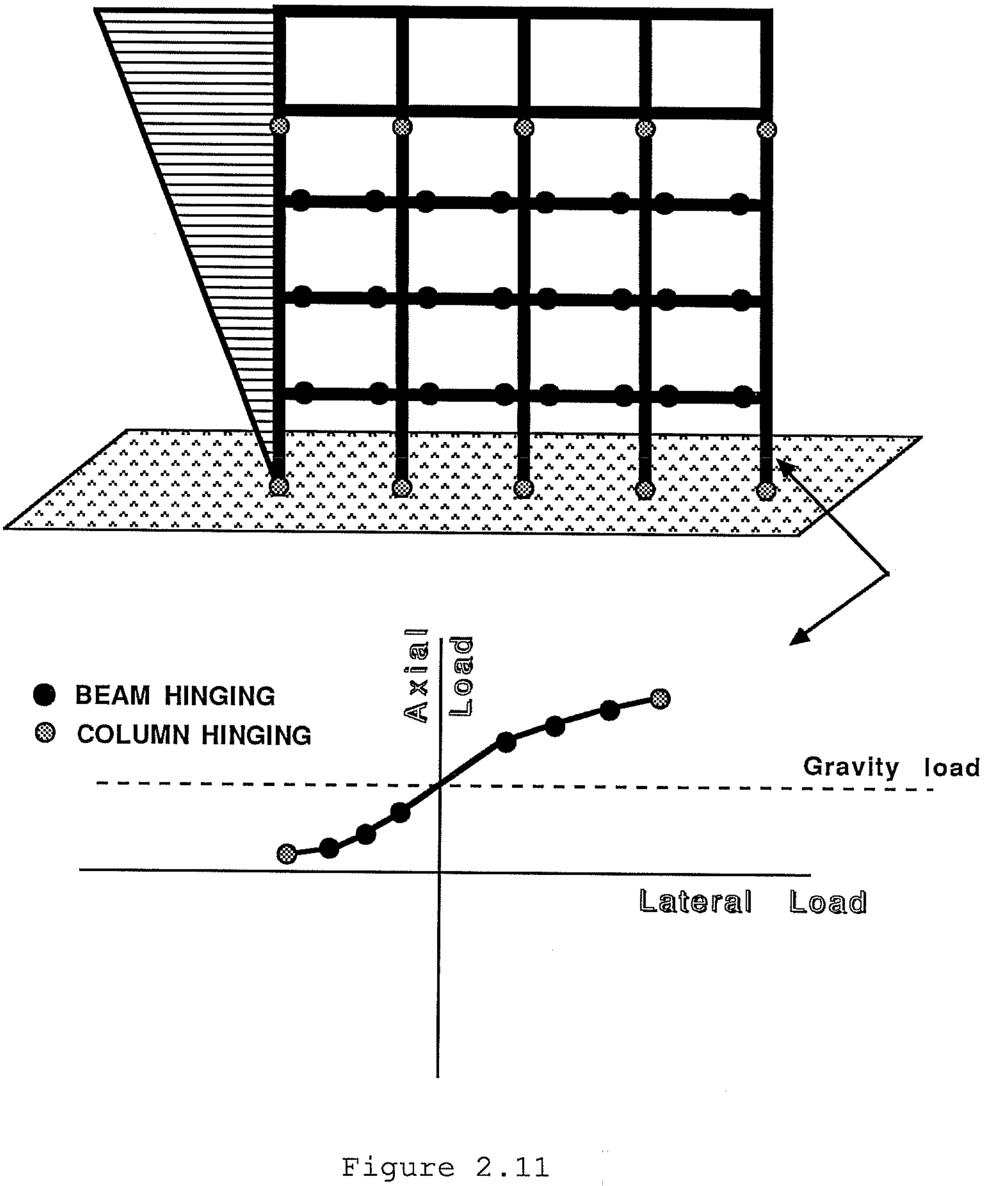


Figure 2.11

LOAD HISTORY
EXTERIOR COLUMN -
ALL BEAMS HINGED

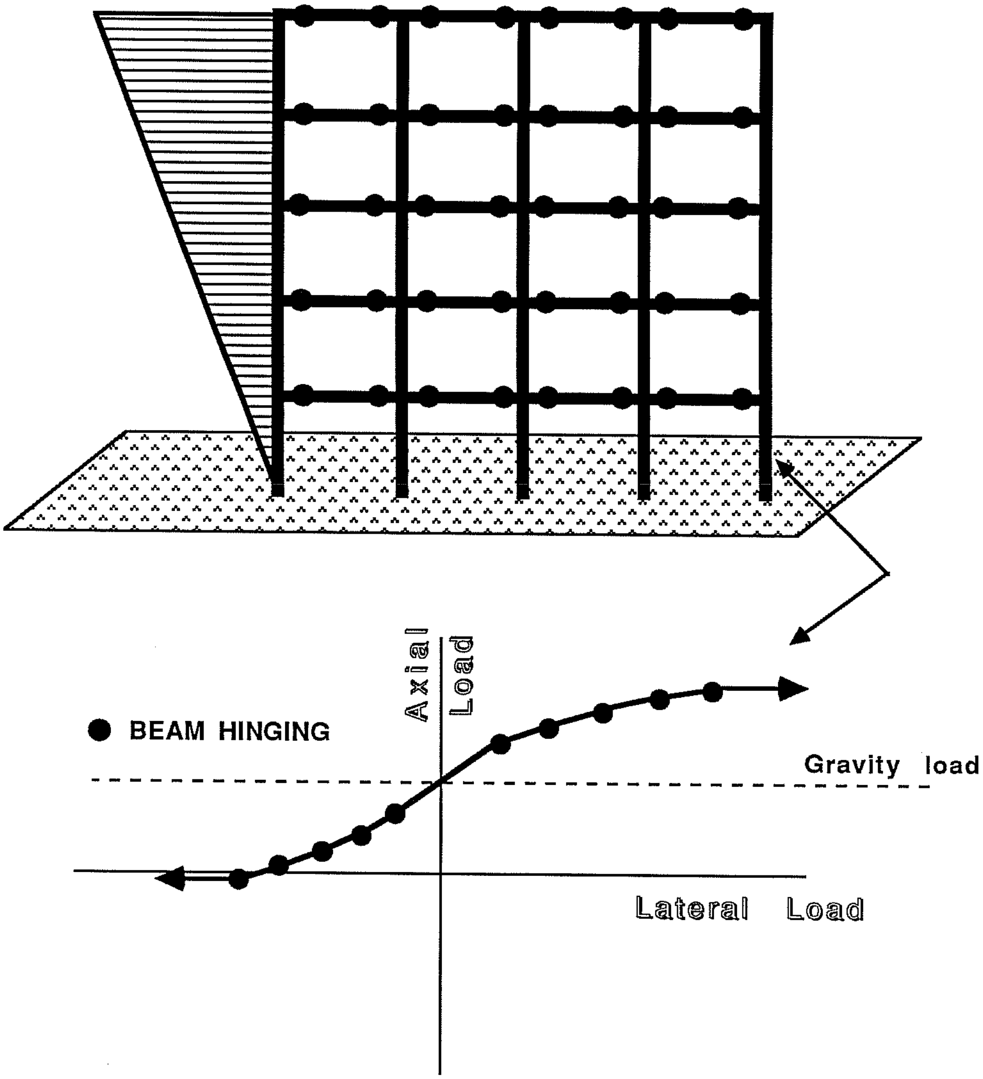


Figure 2.12

same pattern. Axial load and moment increase linearly from the initial state of zero moment, and axial load due to gravity. While the frame remains elastic, the ratio of change in moment to change in axial load remains constant. The relative eccentricity of the axial load with respect to gravity slightly increases each time a beam above the column hinges. Eventually, a mechanism forms and no additional load can be resisted by the frame.

Although none of the axial load-moment paths discussed above are precisely described by the equation of a line once members in a frame begin yielding, the deviation from a linear relationship is generally not large. To reduce complication in the experimental program, it was reasonable to represent the axial load-moment path for an exterior column in a plane frame with a constant relative eccentricity relationship.

2.2.2. _Staggered shear wall-frame system._ While determination of column loads is conceptually simple for a two-dimensional structure like a plane frame, similar analysis for more complex three-dimensional structures is not as straightforward. This is especially true of structures with very different lateral load-resisting systems which are orthogonal to one another.

An example of this type of structure is the staggered shear wall-frame system shown in Figs. 2.13 through 2.16. In the north-south direction, lateral loads are resisted by a moment-

STAGGERED SHEAR WALL - FRAME SYSTEM
EXTERIOR VIEW

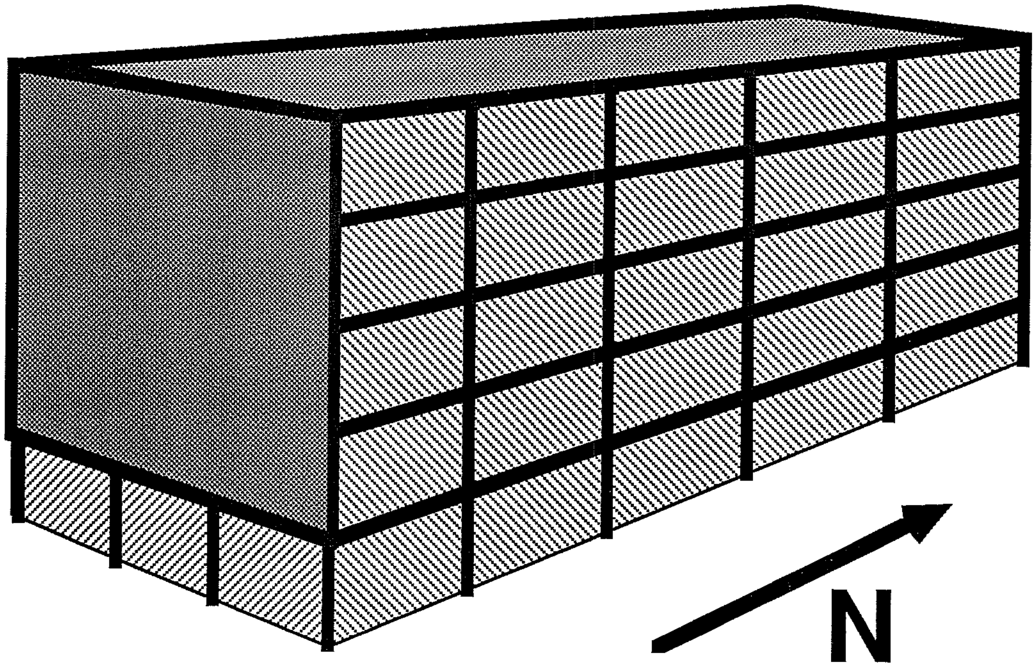


Figure 2.13

STAGGERED SHEAR WALL - FRAME SYSTEM INTERIOR VIEW

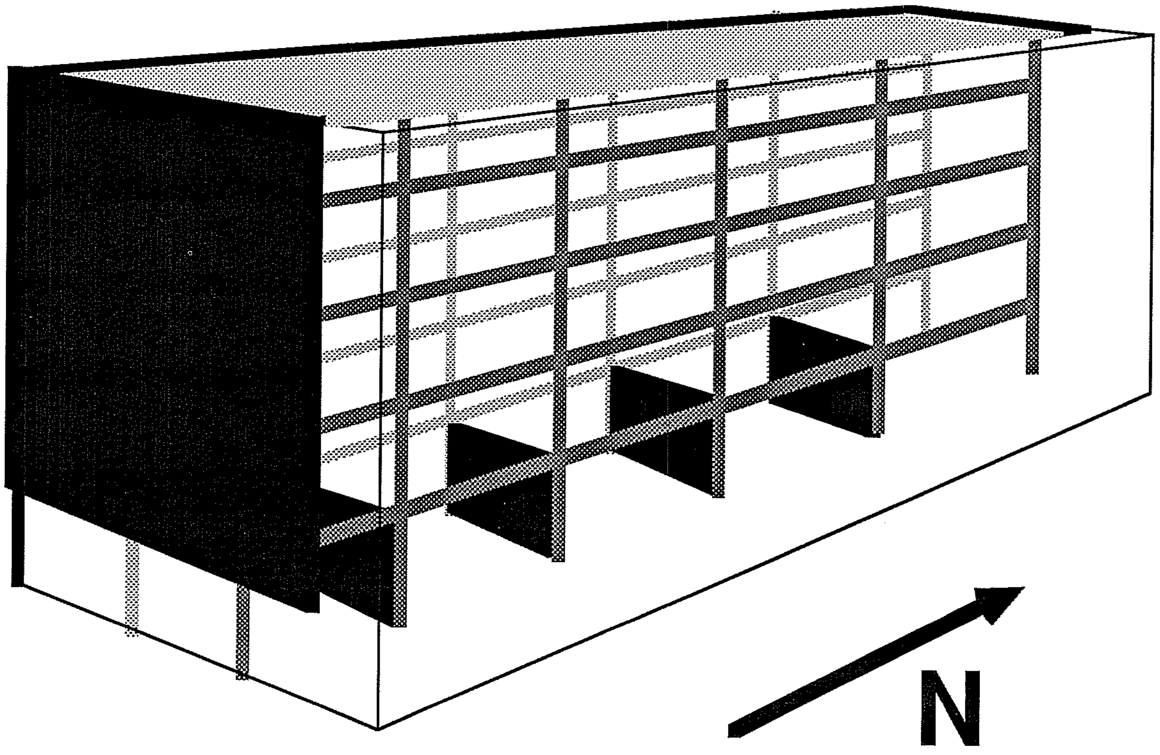


Figure 2.14

STAGGERED SHEAR WALL - FRAME SYSTEM
NORTH - SOUTH SECTION

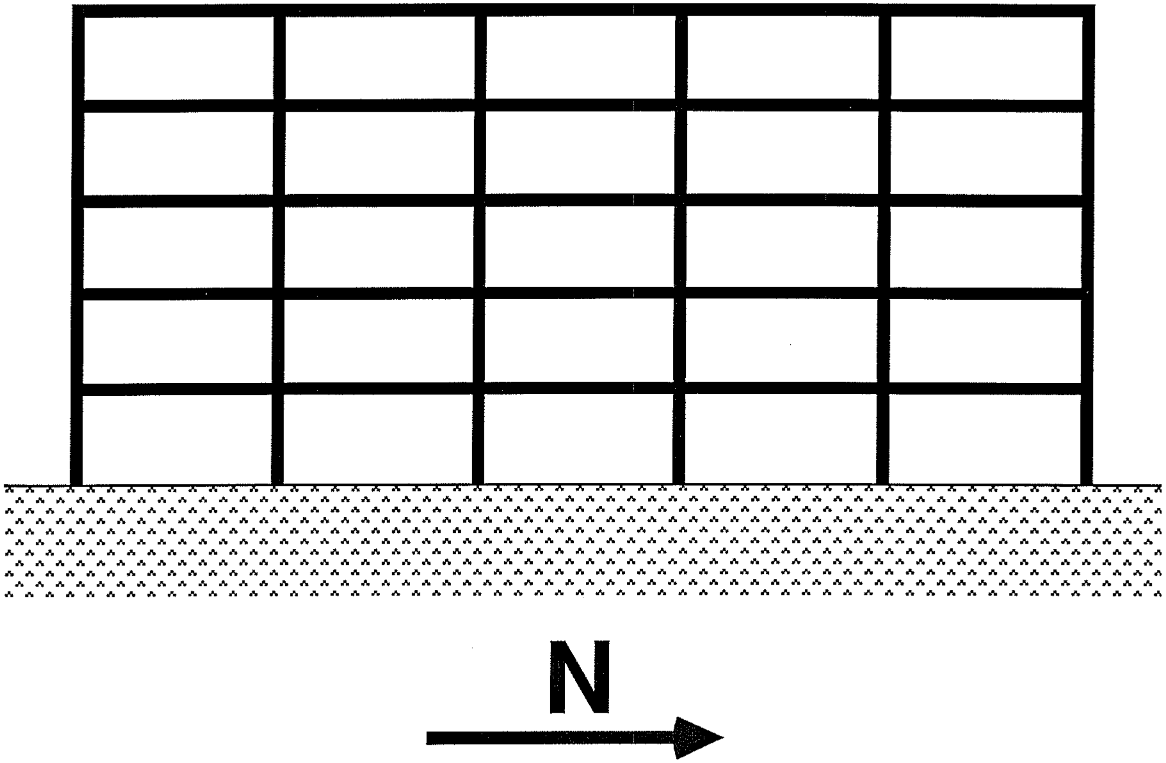


Figure 2.15

STAGGERED SHEAR WALL - FRAME
EAST - WEST SECTION

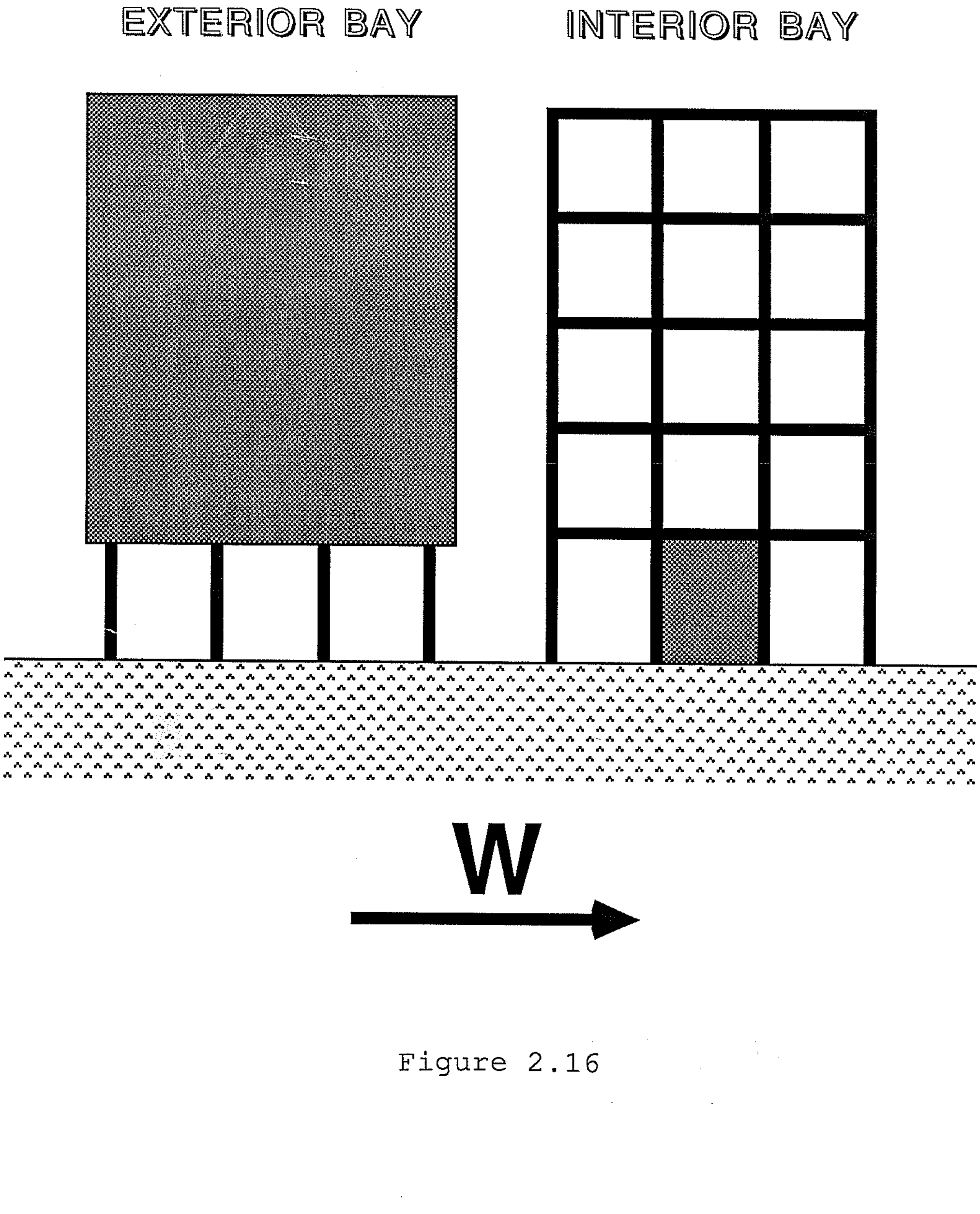


Figure 2.16

resisting frame system similar to those discussed in section 2.2.1 (see Fig. 2.15). Lateral force resistance comes from the flexural stiffness of beams and columns. In the east-west direction, a staggered shear wall system is used (see Fig. 2.16). The shear wall system has very different characteristics from the frame system used in the orthogonal direction. Lateral forces in the east-west direction are mainly resisted by the massive exterior shear wall from the second story to the top of the structure and separate one-story interior walls located in the first story. The massive shear wall does not carry forces directly to the foundation. At the first level (first floor above ground), the shear at the base of the wall is transmitted to interior shear walls by the floor system, which acts as a diaphragm. The overturning moment at the base of the massive wall, however, is not transmitted to the interior walls because the floor slab has very little torsional stiffness. Instead, overturning moment is resisted by large axial forces developed in first-story columns (see Fig. 2.17). Little or no shear force is resisted by the columns in the east-west direction because shear at the base of the massive wall is transmitted to the interior shear walls.

The very different nature of these two orthogonal structural systems results in unusual loads in corner columns.

OVERTURNING EFFECT - STAGGERED SHEAR WALL

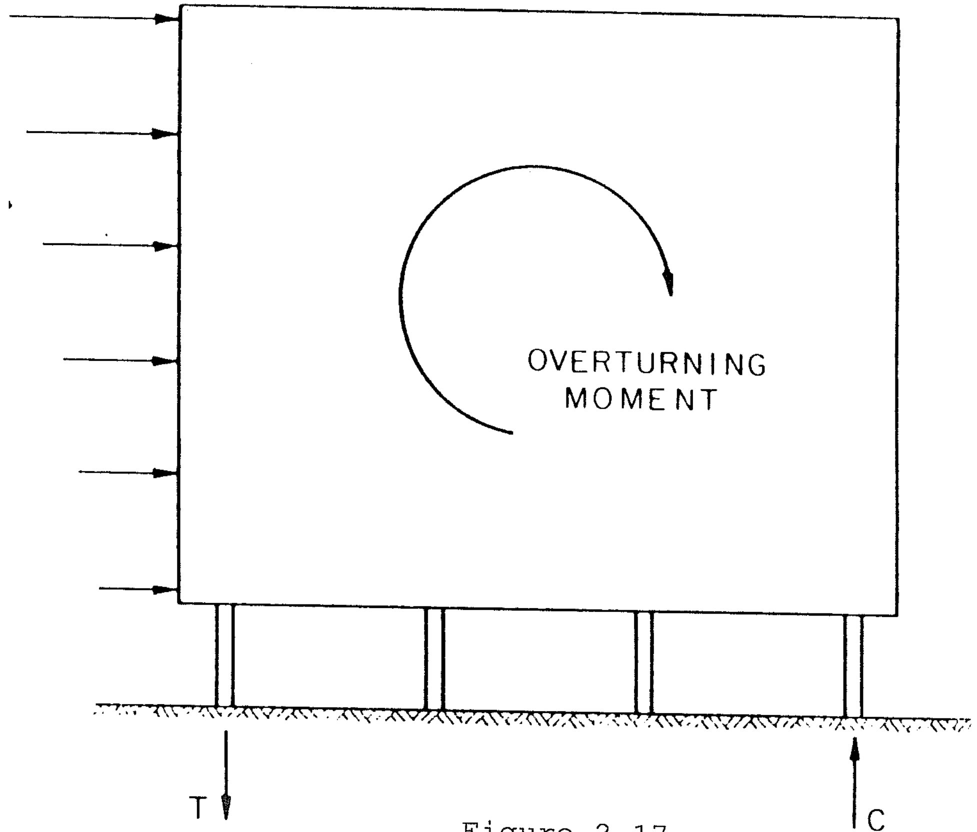
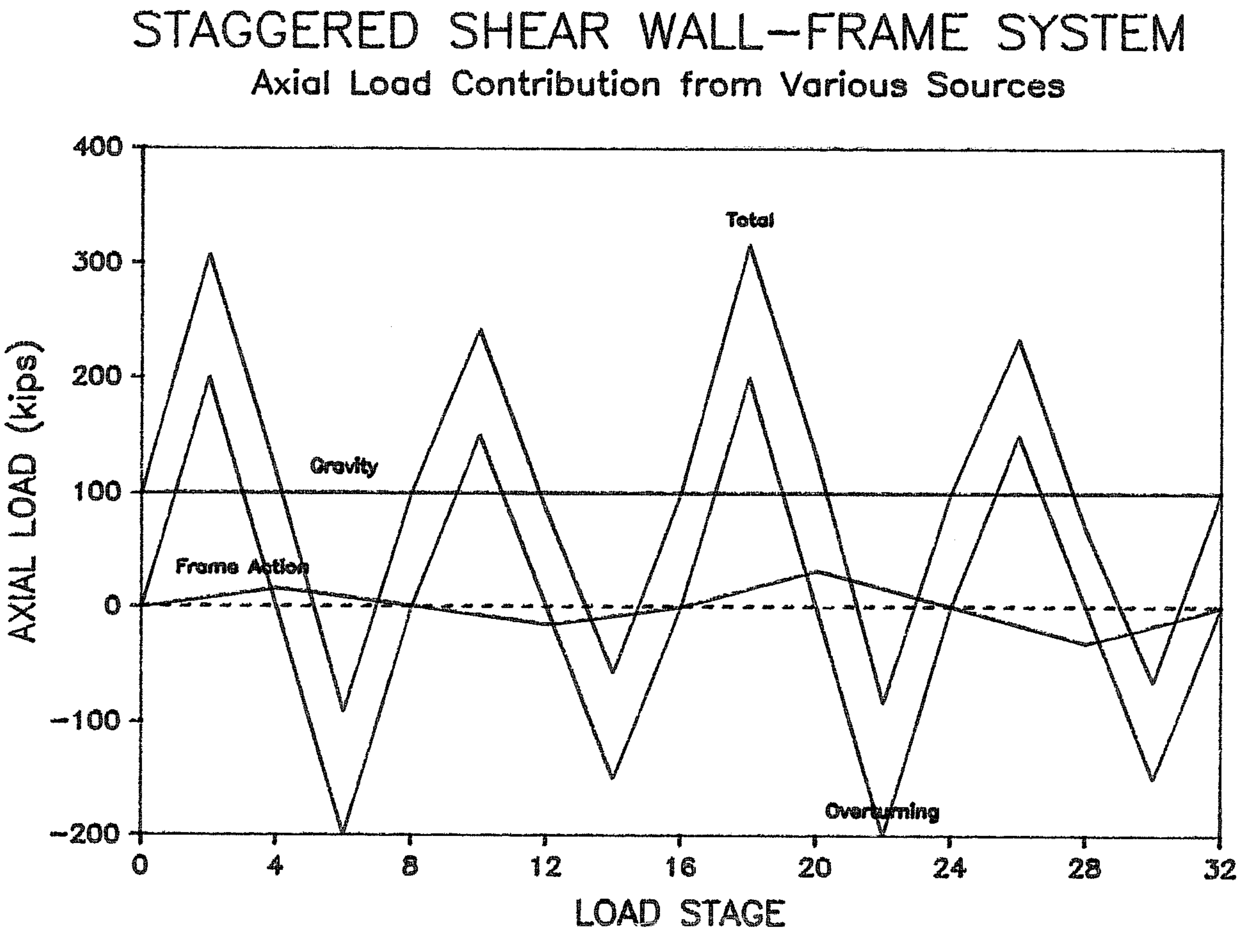


Figure 2.17

Corner columns, because they act as exterior columns in the N-S frame system, are exposed to varying axial load and moment as discussed in Section 2.2.1. These same columns, which are involved in resisting the overturning moment developed by the end E-W shear wall, are also subjected to additional variation in axial load. Therefore, corner columns are subjected to moment due to lateral deformation or loading of the frame system, and axial load from both the frame and staggered wall systems.

Corner column loads are further complicated by the fact that the frequency of response of the frame system is very different from that of the staggered shear wall system. Because the shear wall in the staggered wall system is much stiffer than the frame system, the fundamental frequency of the shear wall system is much higher than the fundamental frequency of the frame. This means that under a strong ground motion which excites both systems, axial load contributed to corner columns by the staggered wall system will undergo more cycles than moment and axial load resulting from response of the frame. Consequently, the direct relationship between moment and axial load in external columns of plane frames does not hold for the staggered shear wall-frame system. In effect, axial load and moment are independent.

The consequences of uncoupling axial load and moment becomes apparent when an idealized load path is examined. For purposes of discussion, it is assumed that the loading described here is for a staggered shear wall-frame structure in which the stiffness of the shear wall system is assumed to be four times the stiffness of the frame (which implies the frequency of the shear wall system is twice that of the frame). There are three sources of axial load on a corner column in such a system: gravity, overturning of the shear wall system, and unbalanced beam shears coming from the frame system. These loads are shown in Fig. 2.18. As can be seen, the additional axial load due to unbalanced beam shears is much smaller than the other axial loads. When these loads are superimposed it results in the axial load path shown in Fig. 2.19 together with the lateral load due to the response of the frame system. When these load paths are plotted on an axial load-vs-moment diagram, the result would be the diagram shown in Fig. 2.20. When this diagram is compared to those shown in Figs. 2.7 through 2.12, it is clear there is a substantial difference between loads experienced by exterior columns in a plane frame and loads experienced by corner columns in the staggered shear wall-frame system. Because the actual load path experienced by a corner column is dependent on many factors, such as stiffness of members in each system, frequency content of the ground motion, and relative stiffness of each independent system, a generalized model for this type of loading



STAGGERED SHEAR WALL–FRAME SYSTEM
Axial Load Contribution from Various Sources

Figure 2.18

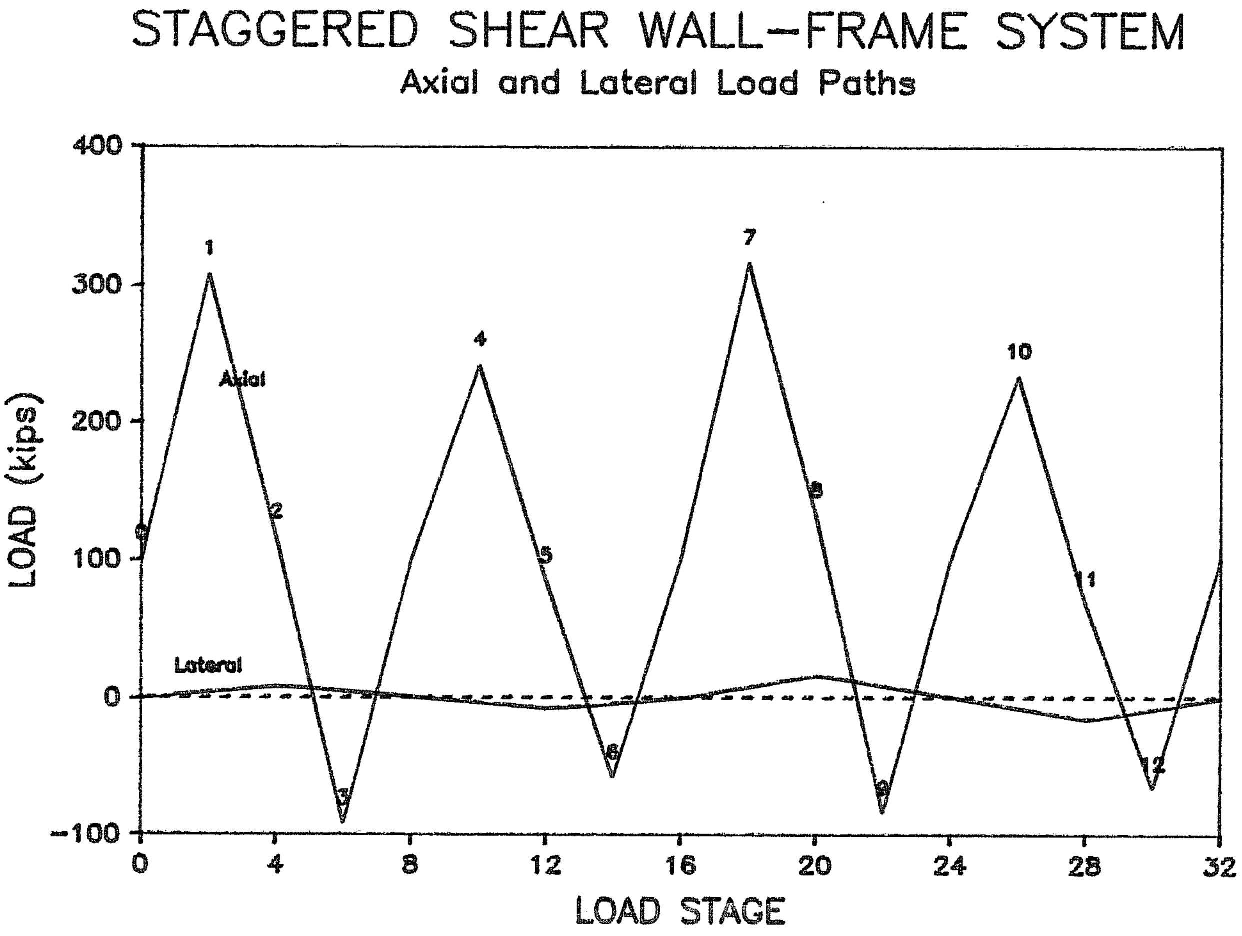

STAGGERED SHEAR WALL-FRAME SYSTEM
Axial and Lateral Load Paths
LOAD (kips)
400
300
200
100
0
-100
LOAD STAGE
0
4
8
12
16
20
24
28
32
Axial
Lateral
0
1
2
3
4
5
6
7
8
9
10
11
12

Figure 2.19

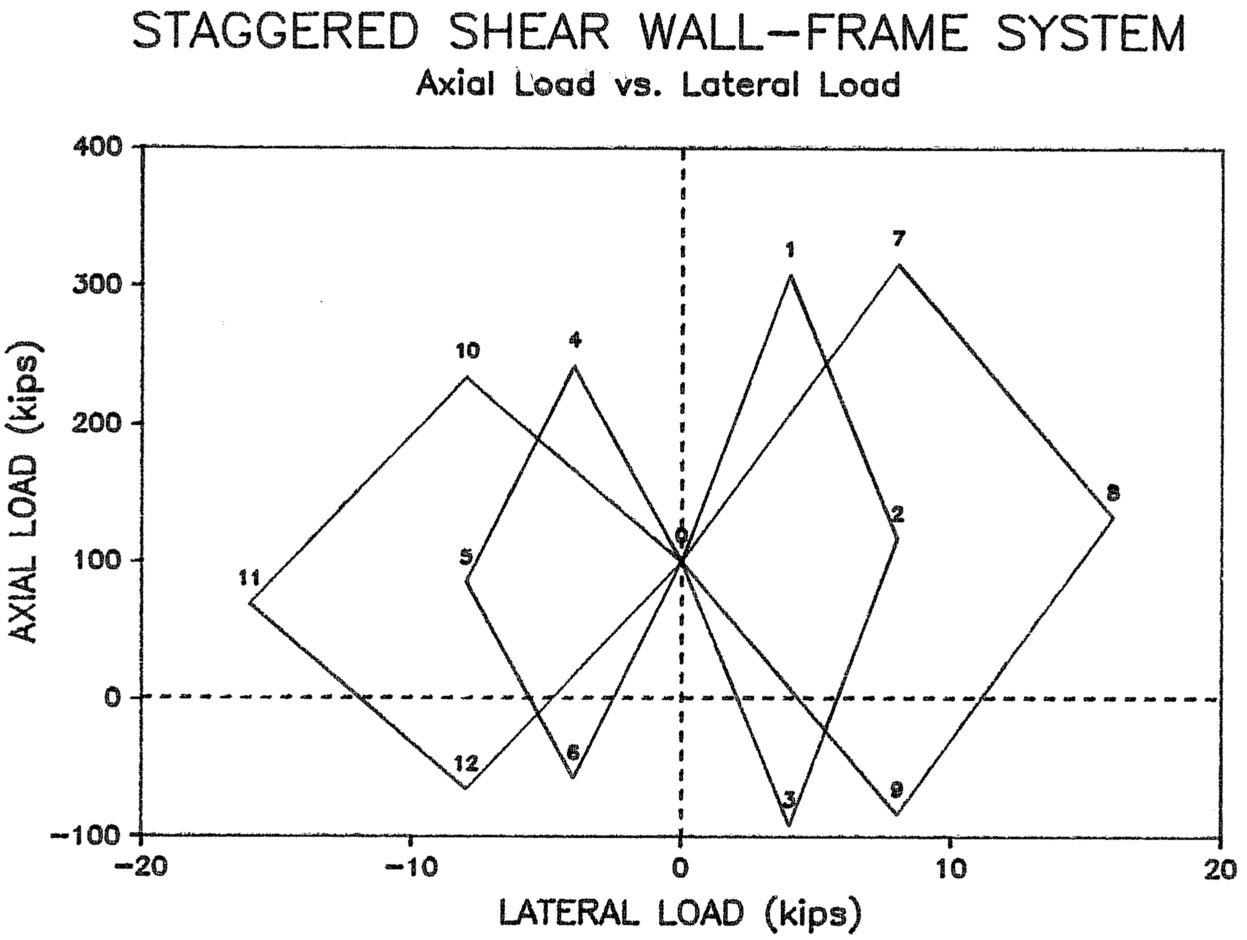

STAGGERED SHEAR WALL–FRAME SYSTEM
Axial Load vs. Lateral Load
AXIAL LOAD (kips)
400
300
200
100
0
−100
LATERAL LOAD (kips)
−20
−10
0
10
20
0
1
2
3
4
5
6
7
8
9
10
11
12

Figure 2.20

(such as the constant relative-eccentricity model of an exterior column described above) is not practical.

2.2.3. Summary. Consideration of two structural systems, a plane frame and a staggered shear wall-frame, revealed three column load paths for structures subjected to strong earthquake motions. The first of these was associated with interior columns of plane frames. Columns subjected to this type of loading experienced reversed cyclic moments due to lateral loads, and constant axial load due to gravity loads. A second type of loading was found for exterior columns in plane frames. Like interior columns, reversed cyclic moments due to lateral loads and an axial load due to gravity were experienced by exterior columns, but an additional axial load due to lateral loads on the frame was also experienced. A constant axial load-moment relationship relative to a gravity load provided a good approximation for load envelope paths. The third type of loading was associated with corner columns in a staggered shear wall-frame system. The distinguishing characteristic of this type of loading was that axial load and moment were imposed on the column by one lateral-force-resisting system and an additional axial load was contributed by a second lateral-force-resisting system orthogonal to the first system. Consequently, axial load and moment were no longer coupled as they were for the plane frame. No general model for this loading was proposed because the load path was dependent on many factors, including stiffness of the elements in each system, relative stiffnesses of the two systems, and frequency content of the ground motion.

2.3. Previous Research

In this section, previous investigations of reinforced concrete column behavior are reviewed. Particular attention is paid to those investigations in which one or more of the three cases dicussed above were studied.

2.3.1. Concentrically and slightly eccentrically loaded columns. Many early studies were performed on columns which were subjected to axial load with little or no eccentricity [1-9]. The focus of these studies was the effect of transverse reinforcement on the ultimate and post-ultimate strength of reinforced concrete columns. Loading was applied monotonically until failure, and only moment due to eccentricity of the axial load was imposed on the column. Neither the concentric nor slightly eccentric axial load represents a realistic load path for a column in the lower stories of a structure subjected to strong ground motion. Rather, these studies were developed for static loads. Models from these studies, however, were a good step toward a better understanding of column behavior under extreme loading conditions.

2.3.2. Columns subjected to reversed cyclic lateral loads under constant axial load. A limited number of investigations of reinforced concrete column behavior under reversed cyclic loads have been undertaken, and in practically all of them columns under constant axial load were studied [10-16]. Interior columns in frames are subjected to this type of loading, but exterior columns in frames and columns in other structural systems are not necessarily well represented by this loading.

Three important trends were observed in these investigations. First, it was found that the hysteresis loops for columns under constant axial load were symmetric about the origin of the moment-drift diagram. This type of symmetry is also a characteristic of beam hysteresis loops. Second, as the level of axial load increased, the capacity of the column under large deformations decreased. This was principally the result of decrease in the effective size of the concrete section due to spalling of cover concrete. Third, as axial compression increased, the amount of energy which the column was able to dissipate decreased.

2.3.3. Columns subjected to reversed cyclic axial and lateral loads with constant relative eccentricity. At least three experimental investigations in which columns were subjected to reversed cyclic axial and lateral loads have been conducted.

The first investigation [16] studied the response of short columns to reversed cyclic lateral loads with both constant and varying axial load. The emphasis of this investigation was on the effect of axial load on the shear behavior of short columns, while later research concentrated on flexural behavior of intermediate (neither short nor long) columns. The second set of tests [17], which was performed on small-scale specimens, involved load histories with constant relative eccentricity throughout the test. Axial load-moment paths were chosen such that combinations above the balanced point were not reached. The third investigation [18-20] involved load histories which had a constant relative eccentricity up to a given axial load, at which point axial load was held constant as moment or drift was increased. This loading was intended to model the load path which would be expected in an exterior column of a frame if all beams in the frame hinged simultaneously before hinges developed at the base of the structure. Again, axial load-moment combinations above the balance point were not reached in this investigation.

In these studies, many of the same general response characteristics of columns subjected to a constant-relative-eccentricity load path were found. First, researchers reported that load-deformation hysteresis relationships were not symmetrical with respect to the origin, as was the case for constant axial load tests. This was principally due to the effect of axial load on the stiffness of the column. Also, the later two investigations [17-20] found that response of intermediate-length columns was strongly related to the bond and anchorage characteristics of the reinforcement.

2.4. Development of Research Program

To further the understanding of the behavior of reinforced concrete columns subjected to varying axial and lateral loads, an experimental and analytical research program was developed.

2.4.1. Experimental program. The behavior of a reinforced concrete column involves the subtle interaction of many factors, making it difficult to predict the response without physical testing. Consequently, an experimental investigation was performed to observe and record the response of approximately half-scale columns subjected to different complex load paths. From the results of the experimental program, general trends of behavior can be identified and the implication of these trends in column design and analysis can be drawn. The experimental program is discussed in detail in Chapter 3.

2.4.2. Analytical program. Although results of an experimental study provide the most direct and comprehensive information about column behavior, an analytical study can also be helpful as an aid for understanding column response. While experimental data is practically indisputable for the specific column and loading tested, it can sometimes be difficult to draw general conclusions. On the other hand, an analytical model based on rational principles of mechanics can be more easily generalized to columns and loadings which were not studied experimentally, especially when results from the analytical model closely agree with results from the experimental program.

Consequently, the second phase of this work involves development and implementation of an analytical model for a reinforced concrete column subjected to general combinations of axial and lateral loads. Development of this analytical model and discussion of computed column response are presented in Chapters 5 and 6.

CHAPTER 3

EXPERIMENTAL PROGRAM

3.1 Overview

An experimental program was developed to investigate the behavior of reinforced concrete columns subjected to loading conditions which occur during an earthquake, such as those discussed in Chapter 2. In this chapter, the development and implementation of that program, in which four approximately half-scale reinforced concrete columns were tested, is described. Three different loading histories were applied to these columns, and their response was recorded and analyzed.

Development of the experimental program involved the design of column specimens, determination of the loading programs, and design of the loading and instrumentation systems. Implementation of the experimental program involved construction and testing of four column specimens, and measurement of their response. The experimental work was conducted between May, 1985 and May, 1986 in the Phil M. Ferguson Structural Engineering Laboratory at the Balcones Research Center of the University of Texas at Austin.

3.2 Design of Column Specimens

Designing the column specimens was a two step process. First, a full-scale prototype column was chosen. Then, based on the limitations of the test setup, a scale factor was determined for the model column and the design finalized.

3.2.1 Prototype column. The principal factor in choosing a prototype column was the availability of data for determining realistic loadings. This was of particular concern for the staggered shear wall-frame system because there is considerably less loading data for this system compared to a typical framed structure, and loading for this system would be much more difficult to develop conceptually. Therefore, the prototype column chosen was from the first story of the Imperial County Services Building, a staggered shear wall-frame system damaged in the 1979 Imperial Valley Earthquake. This building was the subject of an extensive experimental and analytical study [21] from which column loads could be readily determined.

The prototype column was two feet square and longitudinally reinforced with ten No. 11 bars for a reinforcement ratio of 2.7 percent. The height of the column was twelve feet, and No. 3 transverse reinforcement was tied at one foot intervals. The longitudinal reinforcement pattern was the same as that used for the model column, shown in Fig. 3.1.

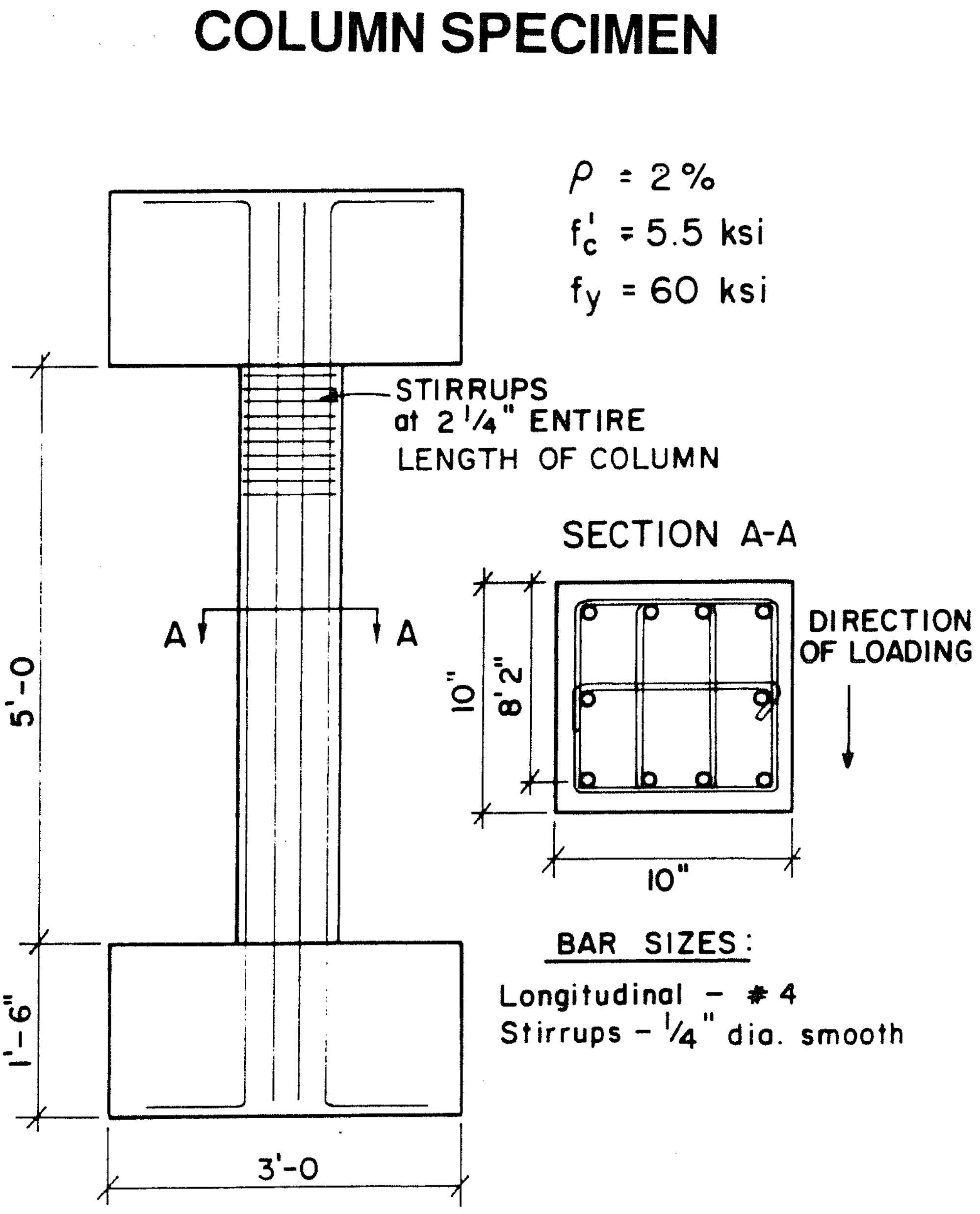

COLUMN SPECIMEN
$\rho = 2\%$
$f'_c = 5.5$ ksi
$f_y = 60$ ksi
STIRRUPS
at 2 1/4" ENTIRE
LENGTH OF COLUMN
SECTION A-A
DIRECTION
OF LOADING
A
A
5'-0
1'-6"
3'-0
10"
8'2"
10"
BAR SIZES:
Longitudinal – # 4
Stirrups – 1/4" dia. smooth

Figure 3.1

3.2.2 Specimen dimensions. The basic philosophy for determining column specimen dimensions was to make the specimen as large as possible without making it so large that it could not be loaded to capacity with equipment available in the laboratory. A large specimen was desired so that details which are used in actual buildings could be accurately reproduced, and so scaling considerations would not appreciably influence test results.

As mentioned above, a prototype column with a two foot square cross section, 2.7 percent reinforcement ratio, and twelve feet tall was selected for this program. A scale factor was determined based on the available load capacity of the testing system. The lateral ram and the axial ram had a capacity of 150 and 300 kips, and the total height of the column and endblocks could not exceed nine feet. In order to determine the scale factor, the following preliminary assumptions about the section were made:

concrete compressive strength = 6 ksi

longitudinal reinforcement ratio = 3.0%

yield stress of reinforcement = 60 ksi

Also, the maximum axial load required during the test was assumed to be 60 percent of the pure axial capacity of the column. On the basis of these assumptions, a scale factor of 2.4 was chosen, giving a column ten inches square and five feet tall. This scale factor satisfied all loading capacity requirements while allowing for realistic detailing of the column.

3.2.3 Specimen reinforcement. Reinforcing steel for each specimen was designed in accordance with ACI 318-83, Appendix A, Special Provisions for Seismic Design [22,23]. Longitudinal steel in the prototype section was 10 No.11 reinforcing bars, four on each face and two at mid-depth. The model utilized the same bar layout as the prototype (see Fig. 3.1) but with No. 4 bars for a reinforcement ratio of 2.0 percent as compared with 2.7 percent for the prototype. Anchorage of the longitudinal reinforcement was developed in the endblocks. Anchorage provided was 30 percent more than required by Appendix A, and was further enhanced by externally prestressing the endblocks. This insured that there would be no anchorage failure of longitudinal reinforcement.

Transverse steel in test specimens was designed in accordance with ACI 318-83 A.4.4, which requires closely spaced transverse reinforcement where severe inelasic deformations are expected in the column. Generally, these hinging regions are at each end of the column, in or near the beam-column joint, but due to the uncertainty of the column response, specimens in this experimental program were designed in accordance with A.4.4.5. This provision calls for closely spaced transverse reinforcement over the entire height of columns which support reactions from a building with discontinuous stiffness. This assures that adequate confinement will be provided no matter where hinges form in the column.

Undeformed Grade 60 bar, one-quarter inch in diameter (the equivalent of a No.2 bar), was used as transverse reinforcement in test specimens. The undeformed bar was used because No.2 deformed reinforcement with properties similar to standard deformed bars was unavailable. The stirrups, which are shown in Fig. 3.2, were placed at two and one-quarter inches, on center, the entire length of the column. This resulted in a heavily reinforced cage which was difficult to fabricate. Figure 3.3 is a photograph of a completed cage, and Fig. 3.1 shows the layout of reinforcement in the cross-section.

3.2.4 Concrete mix proportioning. The concrete in the specimen was designed for a compressive strength of 4500 psi, and the target range for cylinder strength was 5000 to 6000 psi. The mix called for five and three-quarters sacks of cement, three-eighths inch maximum size aggregate, and a slump of six inches. Three-eighths inch maximum size aggregate allowed ACI mix proportioning procedures to be followed and made the use of microconcrete unnecessary. The six inch slump was needed to assure proper consolidation of concrete during casting. As mentioned earlier, large amounts of transverse reinforcement were called

TRANSVERSE REINFORCEMENT

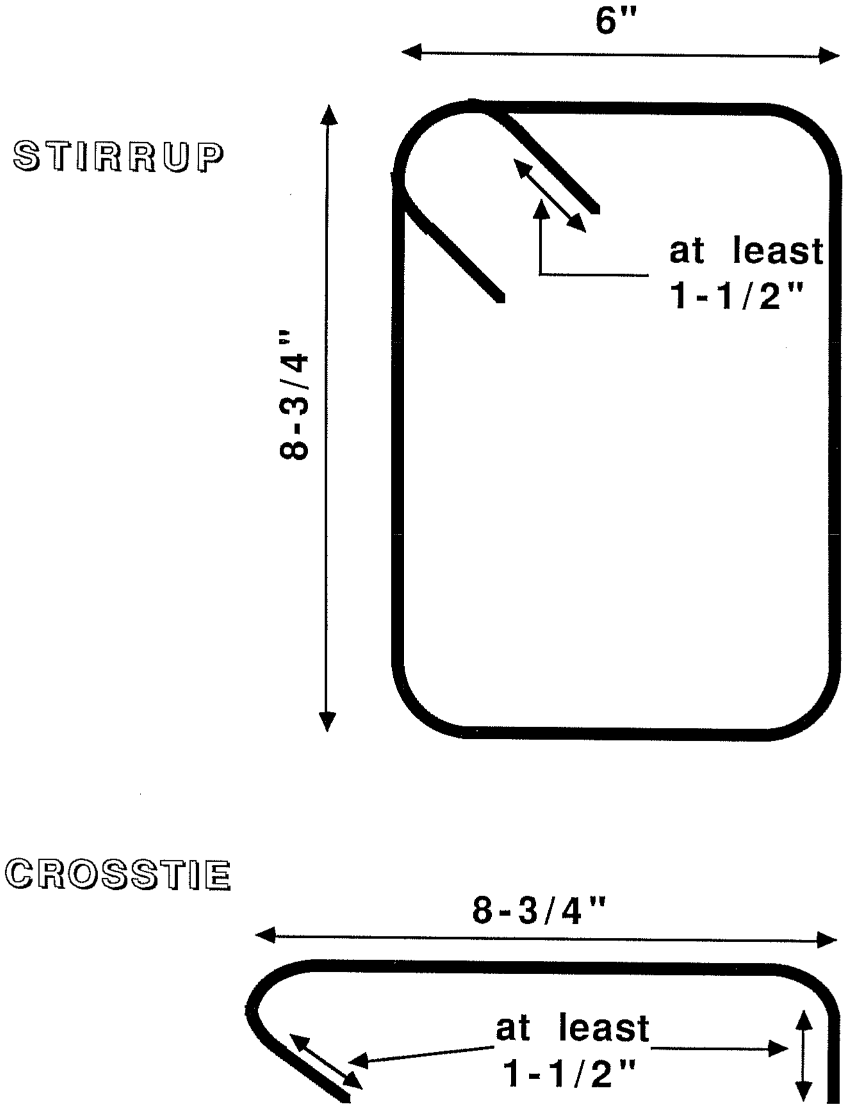


Figure 3.2

REINFORCING
CAGE

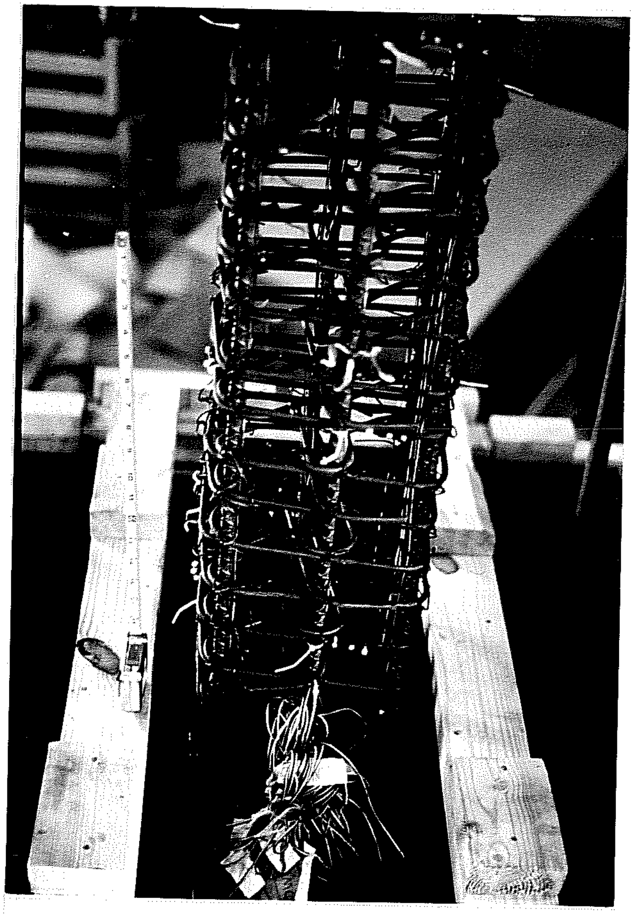


Figure 3.3

for in the design, resulting in severe congestion of the specimens. A retarder was also used in the mix because most specimens were cast during the summer in Austin when temperatures in excess of 100 degrees were experienced.

3.3 Construction of Column Specimens

3.3.1 Formwork. Design of the formwork was a complex task, given the shape and casting requirements of the specimen. The first decision to be made in designing the formwork was how the specimen should be cast. Alternatives included casting the entire specimen (column plus top and bottom enblocks) monolithically, casting in two stages with a cold joint at mid-height of the column, and casting in three stages with a cold joint at each connection between column and endblock. Because columns in structures located in seismic regions are generally cast monolithically with beams and slabs, either a monolithic or a two-phase specimen was necessary. For purposes of analytical modelling, a monolithic column was the best alternative due to the absence of any cold joint. Finally, a two-phase specimen seemed to be no easier to fabricate than a monolithic specimen, so a monolithic specimen was chosen.

Fabrication of a monolithic specimen presented some difficult problems. Casting a specimen monolithically meant that concrete in the bottom endblock would still be fresh while concrete in the top endblock was being placed. This would result in a hydrostatic pressure large enough to force concrete out of the bottom endblock unless a closing form was provided over the top of the bottom endblock. A closing form, however, would make it difficult to place the concrete in the bottom endblock, and would make stripping of the forms even more difficult. After two trials, a system was developed which allowed the specimen to be cast monolithically with relative ease, but still allowed for efficient removal of the forms the next day. A photograph of this system is shown in Fig. 3.4 and design drawings are shown in Figs 3.5 through 3.8.

Tolerances were a major concern in the development of the formwork system. Seemingly small errors in the fabrication of the model specimen could be quite significant. For instance, an extra quarter-inch on each side of the column would have provided an additional 30 kips of axial capacity. Therefore, extra measures were taken to control the dimensional tolerances of the specimen. The column form, which was designed to minimize bulging, worked quite well, as all column sections ended up ten inches square, plus or minus a sixteenth of an inch (plus or minus 1% in terms of area), over the entire column height.

3.3.2 Assembly of reinforcement. Construction of reinforcing cages involved considerable time and effort because of the closely spaced transverse reinforcement and the

FORMWORK

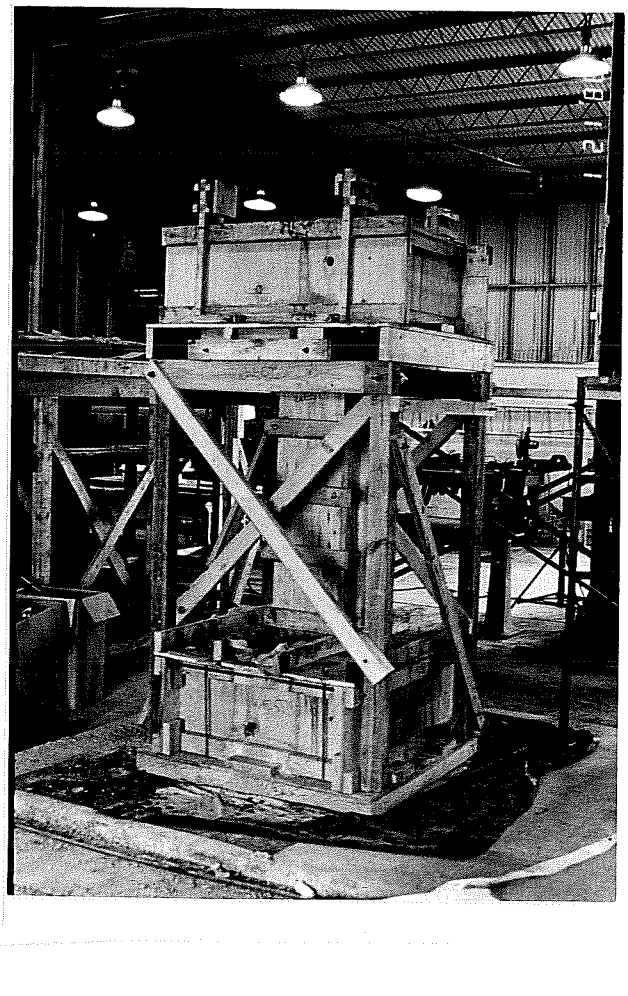


Figure 3.4

FORMWORK ELEVATION 1
Bracing Removed for Clarity

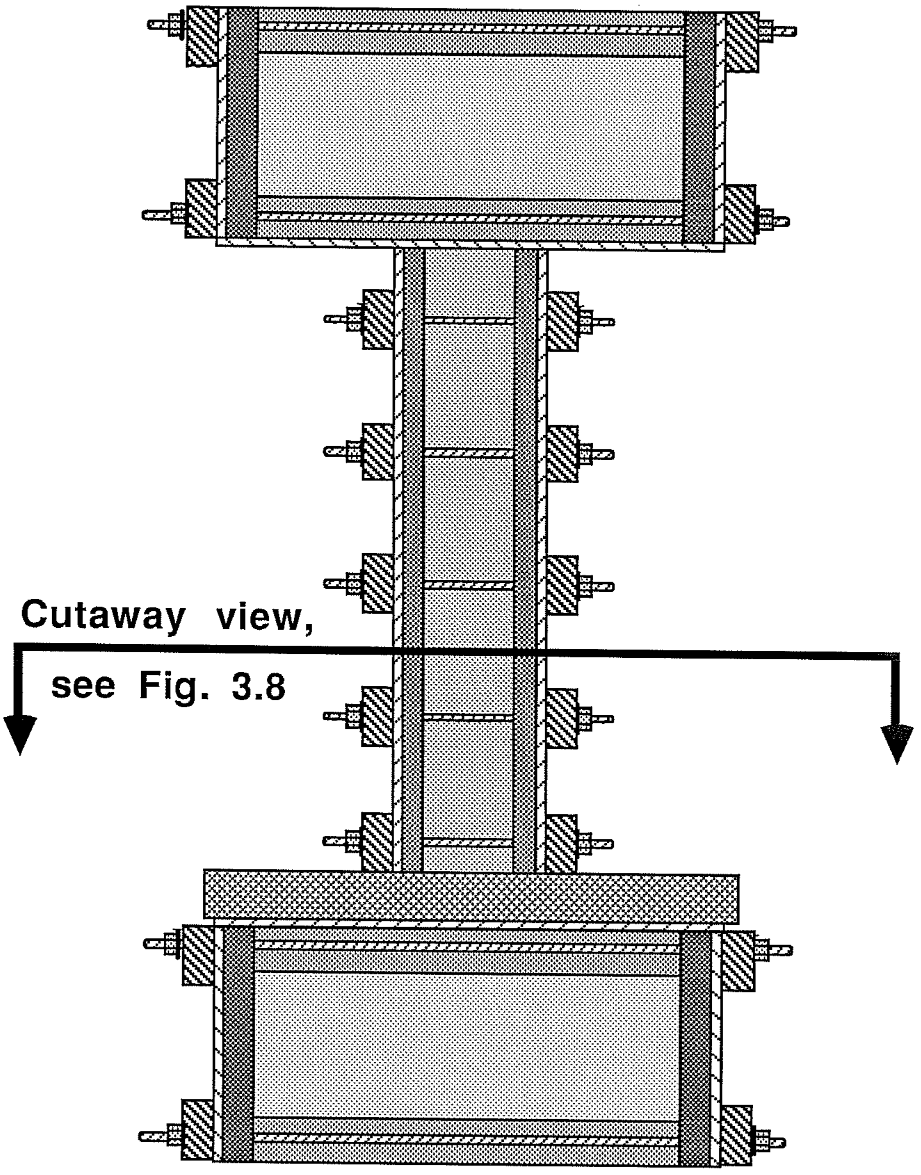


Figure 3.5

FORMWORK ELEVATION 2 w/ Bracing System

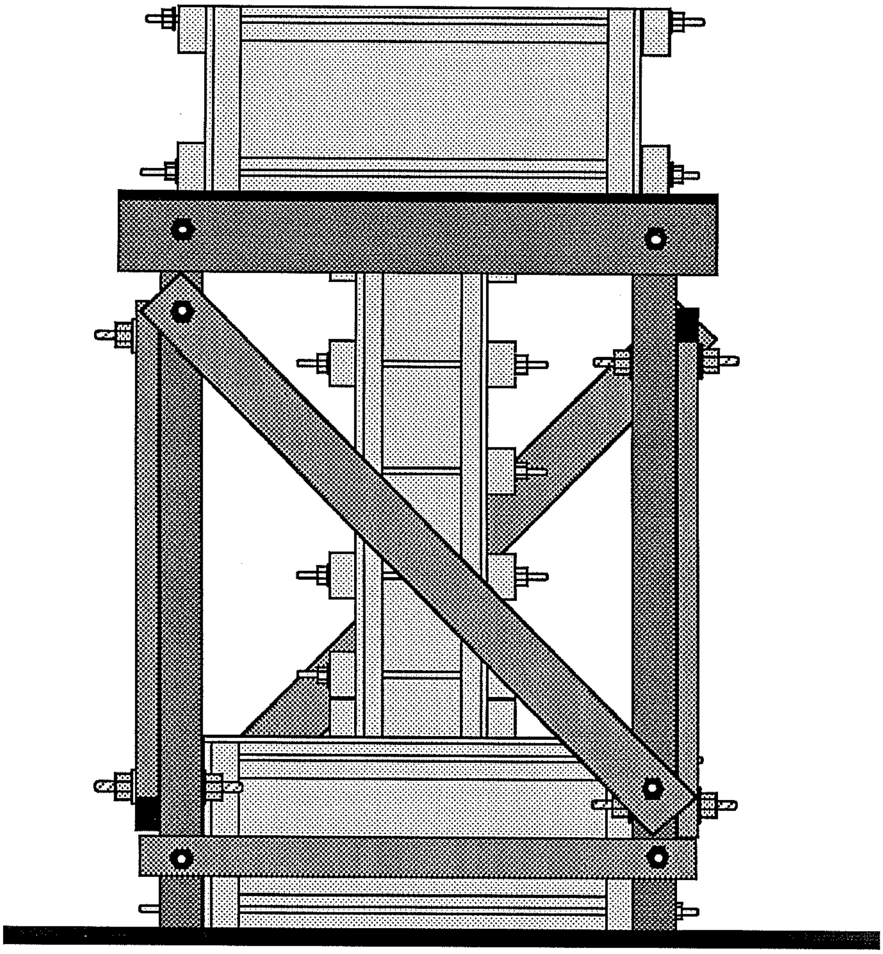


Figure 3.6

FORMWORK
Top View

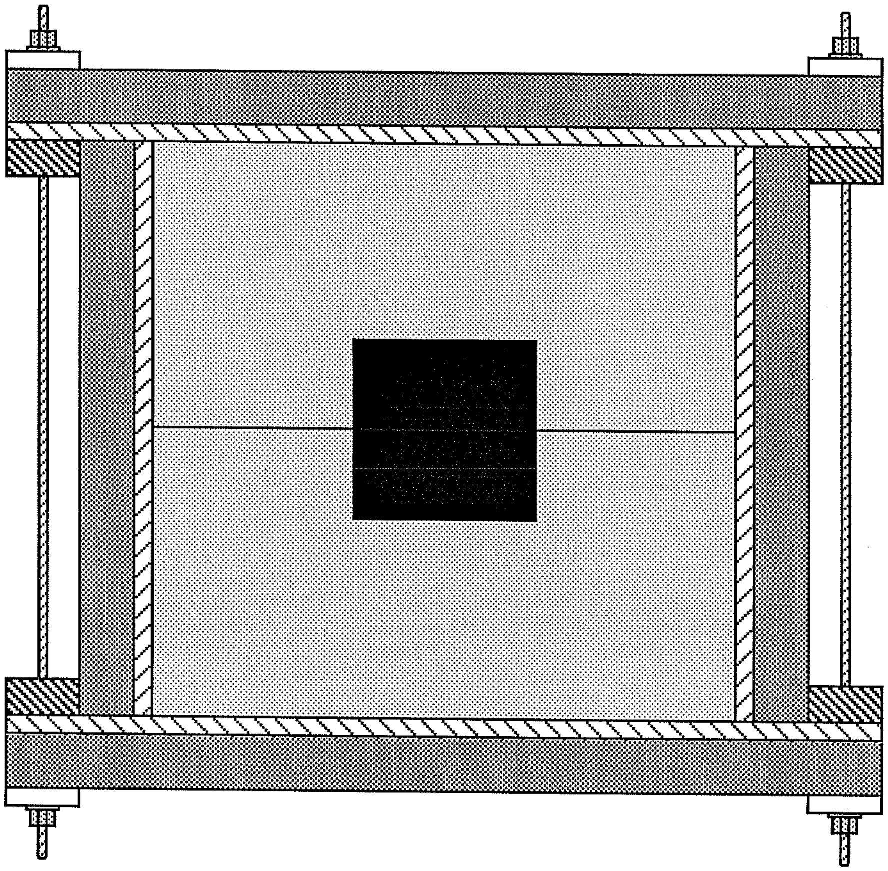


Figure 3.7

FORMWORK
Cutaway View

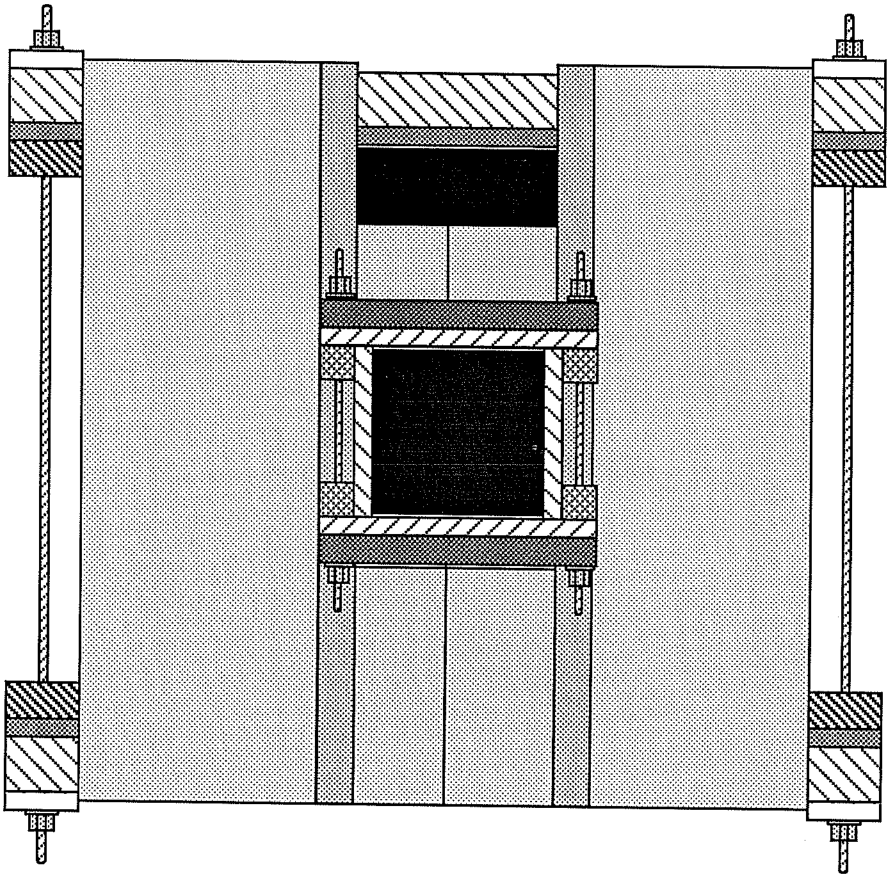


Figure 3.8

extraordinary care required when tying steel which had strain gages attached. The process of building a cage involved three phases: cutting and bending of reinforcing steel, strain gaging, and tying the elements into a single cage.

Transverse steel was bent by a fabricator, Alamo Steel Co., to specifications in accordance with ACI 318-83. Each stirrup was made of three pieces: two rectangular ties and one crosstie (Figs. 3.1 and 3.2). Longitudinal steel was ordered in stock lengths, cut and, where required, bent to size. Longitudinal steel for all specimens came from the same heat, to minimize variations in material properties from one specimen to the next.

Strain gages were attached at locations within the column to both longitudinal and transverse bars, as shown in Fig. 3.9. Two types of gages were used, FLA-5-11 and FLA-2-11, manufactured by Tokyo Sokki Kenkyujo Co. Bars were prepared for gaging by grinding away deformations on the longitudinal bars in the region of a gage and then fine-grinding all of the strain gage locations in order to assure good contact between gage and bar. Gage locations were cleaned with acetone, and gages were attached with M-bond, a high-strength adhesive. Leads were then soldered to the gages and the entire location was waterproofed,

INSTRUMENTATION

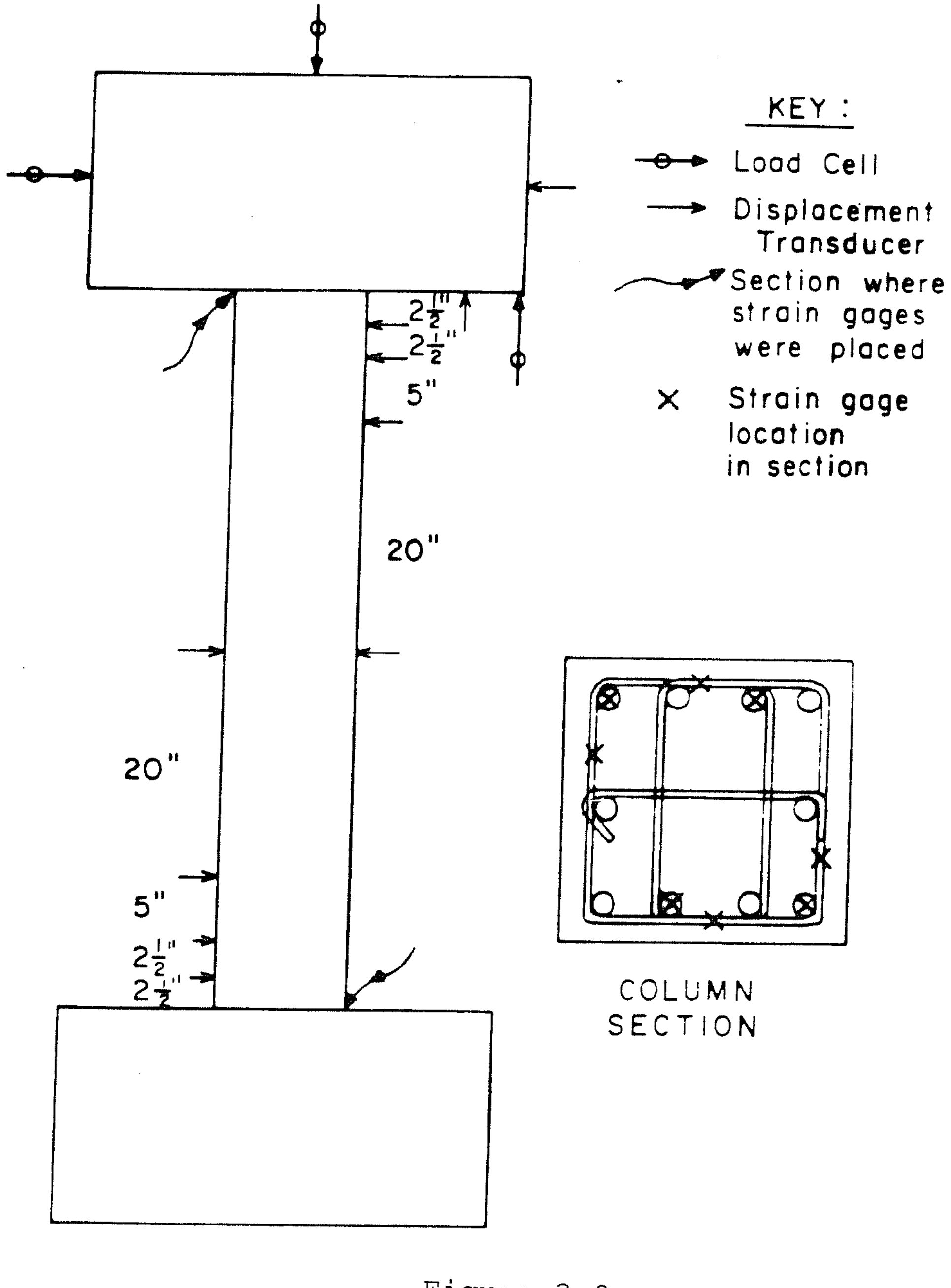


Figure 3.9

first with a white polymer paint, then with a black plastic patch, Barrier E.

Once reinforcing steel was prepared, the elements were assembled. Stirrups were placed at two and one-quarter inches on center, and a total of 32 stirrups were used in each specimen. Once stirrups were tied, lead wires from the strain gages were bundled together and directed out of the path where concrete would be flowing in order to keep them from being damaged. Once the tying was completed, the cage was ready for placement in the forms.

3.3.3 Casting. Once forms were assembled and oiled, a reinforcing cage was inserted and positioned in the column forms. Typically, three cubic yards of concrete was ordered from Texas Readymix Co. for casting of each column specimen and material samples. Upon arrival, a slump test was performed to check the workability of the concrete. On two occasions, water was added to bring the slump to the desired six inches. Once the six inch slump was obtained, a wheelbarrow was filled with concrete for use in casting six-by-twelve inch cylinder specimens. A one cubic yard bucket was used to fill the column forms. First the bottom endblock was cast, and the door on the endblock form sealed. Next the column was cast and thoroughly vibrated with a one and one-half inch diameter stick vibrator to assure consolidation of the concrete, and finally the top endblock was cast.

After the casting process was complete, the top endblock, which was the only area where fresh concrete was exposed, was finished with a hand trowel and covered with plastic to help minimize water loss and facilitate curing. The forms were left on the specimen for approximately 24 hours before being removed. The specimen was then moved to a storage area, and the forms were cleaned, oiled, and reassembled for casting the next specimen.

3.4 Material and Specimen Properties

In this section, the as-built dimensions of the specimens are presented, along with the properties of the concrete and reinforcing steel.

3.4.1 Concrete. Concrete properties were determined from uniaxial compression tests on plain concrete cylinders. Results of these tests are shown in Table 3.1. Each series involved testing of at least three cylinders, with more tested in a given series if the scatter was large in the first three tests. With the results of these series of tests, a 28 day compressive strength could be estimated using a maturity curve [24]. Table 3.1 also shows the compressive strength of the concrete cylinders on the days when each specimen was tested.

CONCRETE CYLINDER COMPRESSIVE STRENGTH

COLUMN	NO. TESTS	TEST VALUES		AVERAGE STRESS @ 28 DAYS
		DAYS	AVG STRESS (psi)	
C - HA tested at 114 days	9	6 14 115	3816 4062 4843	**4300**
C - LA - 1 tested at 63 days	10	32 52 64	5836 5887 6487	**5800**
C - LA - 2 tested at 56 days	7	38 58	5386 5704	**5300**
I - HA tested at 30 days	5	31	5910	**5900**

Table 3.1

3.4.2 Reinforcement. Reinforcing steel was tested in uniaxial tension on a 600 kip Universal Testing Machine at the Ferguson Structural Engineering Laboratory. A total of four bars were tested, with lengths between eight and twelve inches. Values were recorded for yield and ultimate stress, and test values for each were averaged to obtain Fy and Fu, respectively. The yield stress was found to be 75.3 ksi, and the ultimate stress was found to be 117 ksi. All reinforcing used in column specimens came from the same heat.

3.4.3 As-built specimen dimensions. Although the specimens were built as close to specifications as possible, there was nevertheless a variation in dimensions from specimen to specimen. The as-built dimensions of each specimen are shown in Table 3.2.

3.5 Description of the Test Apparatus

3.5.1 Test frame. Specimens were tested in a frame which was adapted from prior column tests performed at the Phil M. Ferguson Structural Engineering Laboratory [13,14,16,25,26]. The frame consists of four wide-flange steel columns supporting an H-shaped crosshead. An axial ram is attached to the center of this crosshead by means of a pin and swivel plate. The flanges of the columns have bolt holes every three inches to allow for adjustment of the crosshead for specimens of different heights.

AS-BUILT DIMENSIONS

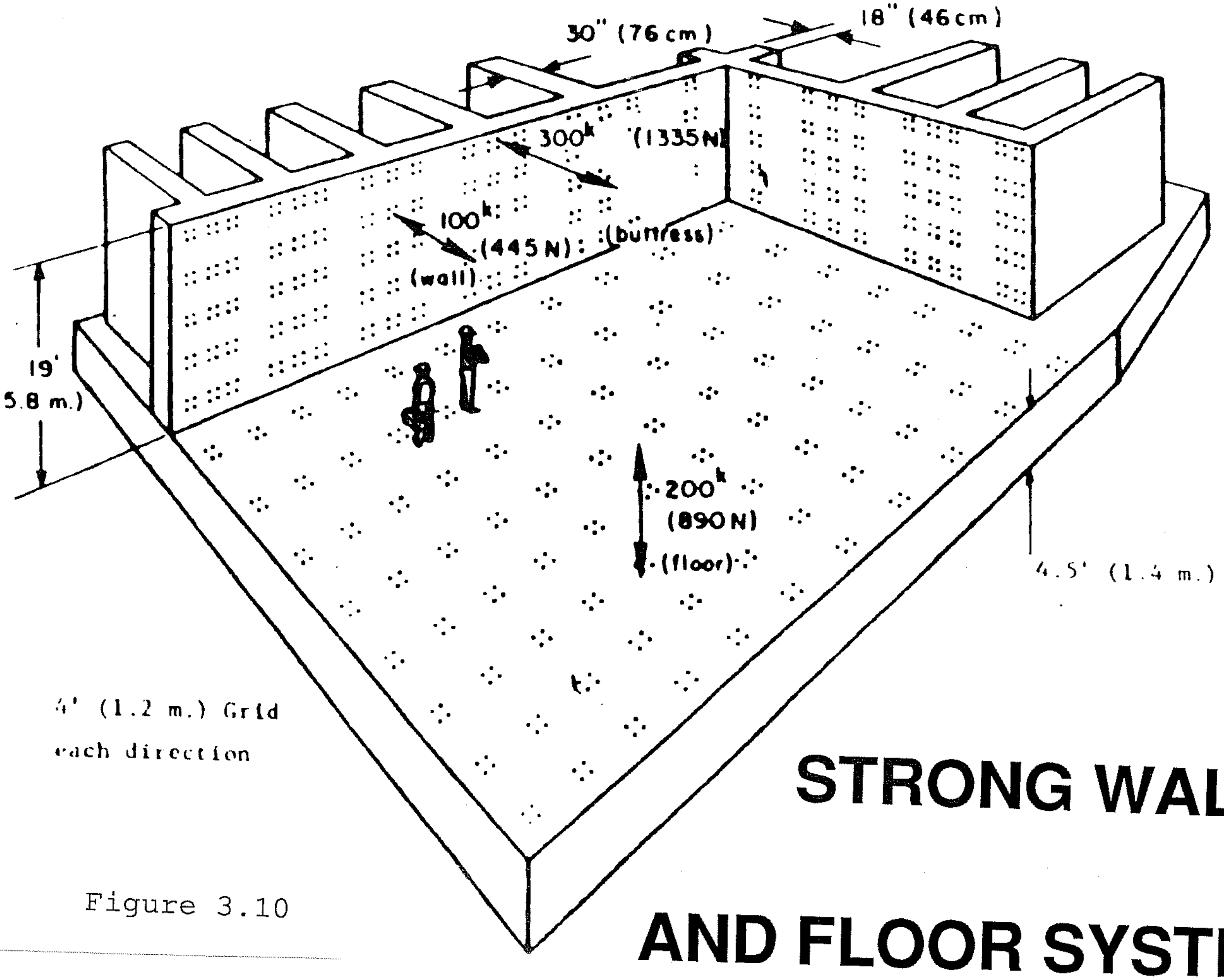
COLUMN	LENGTH (IN.)	LOCATION	WIDTH (IN.)	DEPTH (IN.)	COVER (IN.)
C-HA	60.0	TOP	10.0	10.0	1.0
		BOTTOM	10.0	10.0	1.0
C-LA-1	59.8	TOP	10.0	10.0	1.2
		BOTTOM	10.0	10.0	1.0
C-LA-2	60.0	TOP	10.0	10.0	1.0
		BOTTOM	10.0	10.0	1.0
I-HA	59.9	TOP	10.0	10.0	1.0
		BOTTOM	9.9	10.0	1.4

Table 3.2

The columns are bolted to the "strong" floor of the laboratory and the frame is braced against the "strong" wall in both the north-south and east-west directions. A schematic of the frame and the "strong" wall and floor system is shown in Fig. 3.10.

Loads were transferred from the hydraulic rams to the specimen through a cross-shaped steel cap. This cap was attached to the top endblock of the specimen using four high-strength steel rods. Each bolt was tensioned to approximately 14 kips to assure that the cap remained firmly attached during load application, and hydrostone (gypsum cement) was used between the cap and the endblock to assure a uniform transfer of force. The bottom endblock was attached to a similar cap which was firmly anchored to the floor. To assure that these caps remained level with respect to one another, four rams were placed between them, one at each tip. Also, to prevent relative twisting of the endblocks, two rams were placed between the top cap and the two south columns of the load frame. This leveling system is described in greater detail below. A photograph of the column in the testing apparatus is shown in Fig. 3.11.

3.5.2 Primary loading system. Load was applied to the specimens using a series of hydraulic actuators (piston-cylinder rams). These rams operate by forcing oil into the ram cylinder, which applies a force to the piston equal to the cylinder pressure times the area of the piston. In this way, the force



STRONG WALL AND FLOOR SYSTEM

Figure 3.10

TEST SETUP

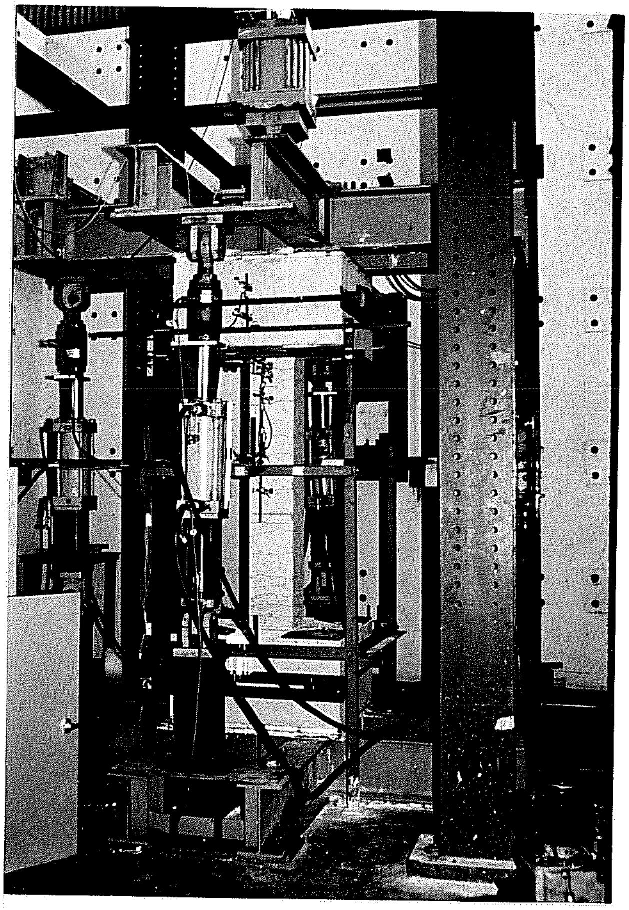


Figure 3.11

applied to the piston (and to the specimen) can be controlled by controlling the pressure of the oil in the cylinder.

The primary loading system consists of two main loading rams. The axial ram is made by Shore Western and has a 160 ton (320 kip) capacity and a maximum operating pressure of 3000 psi. The lateral ram is also a Shore Western ram, and it has a 75 ton (150 kip) capacity and a maximum operating pressure of 3000 psi. Both of these rams were supplied oil from the same pump at a pressure of 3000 psi, and a multi-stage filter and accumulator was used at the splitting junction to remove any hydraulic pulses or grit in the oil lines. The pump, the accumulator, and the main rams were all monitored and controlled electronically in a closed loop configuration as described below.

3.5.3 Leveling system. The leveling system was used to keep the endblocks of the specimen from rotating or twisting relative to one another during the test. This assured that the column deformed in double curvature as shown in Fig. 3.12. The principle behind the leveling system is illustrated in Fig. 3.13. The leveling system consisted of six hydraulic piston rams, three pairs of 50 ton (100 kip) Shore Western rams, each pair restricting rotation about an axis orthogonal to the other two. Each pair of rams was cross-linked; the pressure chamber on each ram was directly connected to the return chamber of the other

COLUMN DEFORMATION

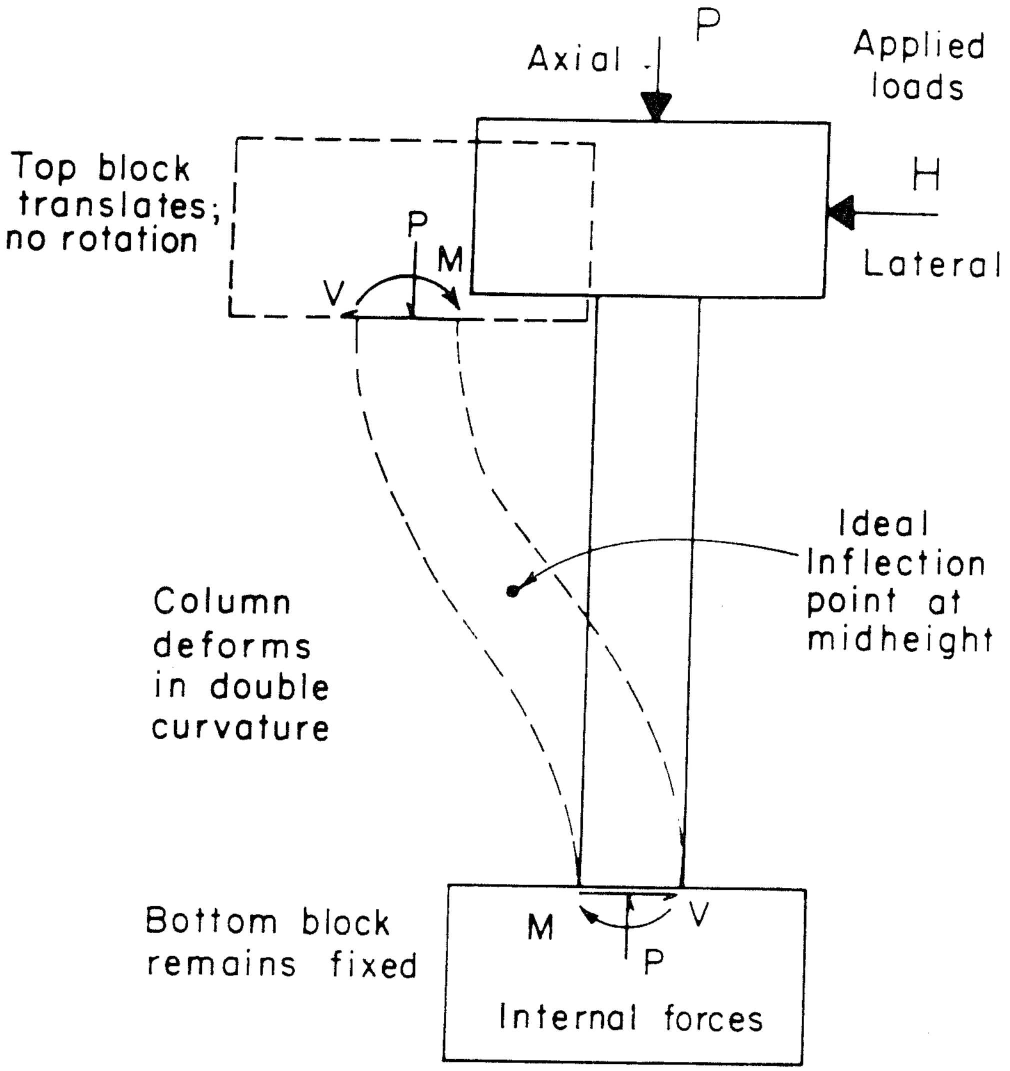

Axial
P
Applied loads
Top block translates; no rotation
P
M
V
H
Lateral
Ideal Inflection point at midheight
Column deforms in double curvature
Bottom block remains fixed
M
V
P
Internal forces

Figure 3.12

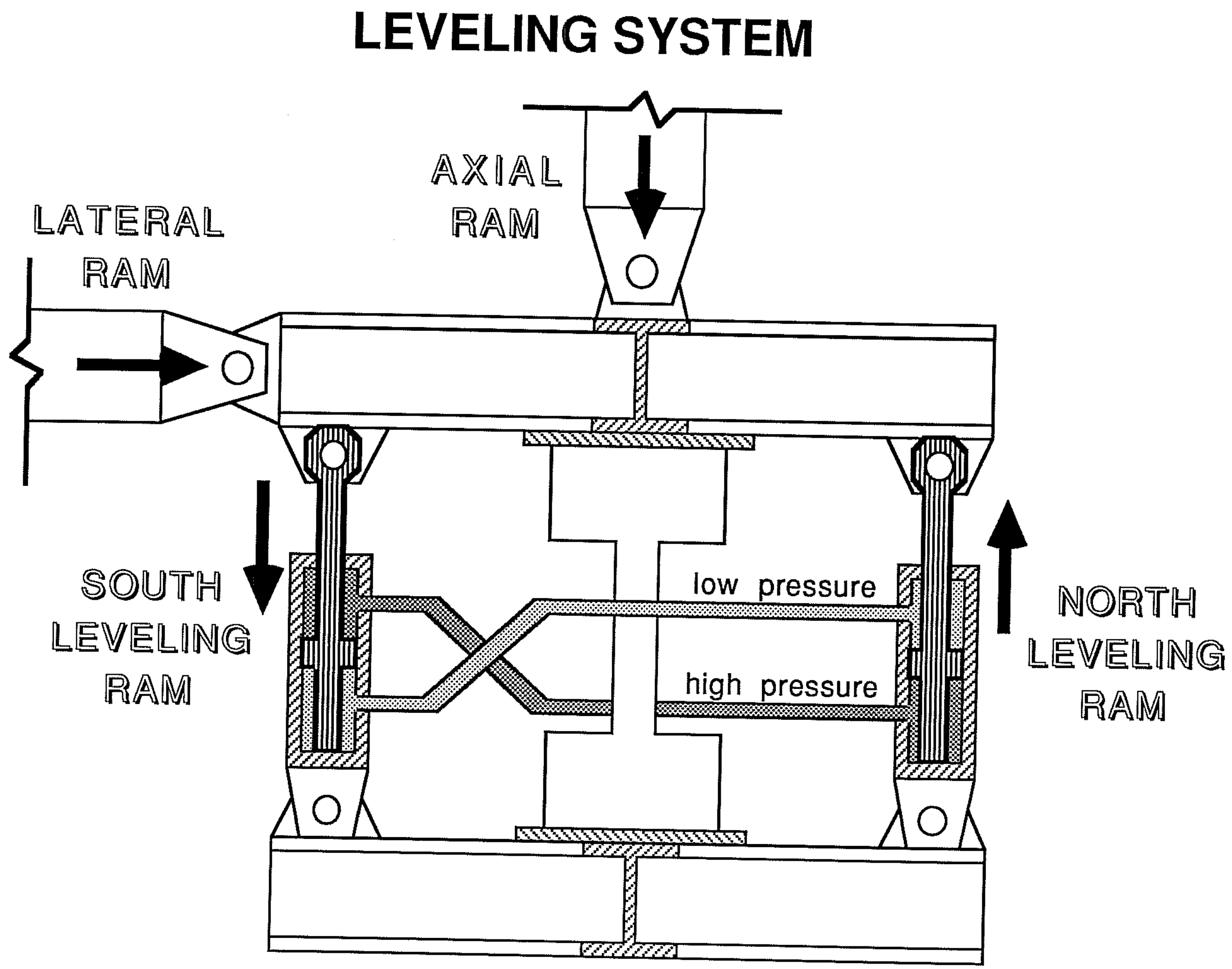

LEVELING SYSTEM
AXIAL RAM
LATERAL RAM
SOUTH LEVELING RAM
NORTH LEVELING RAM
low pressure
high pressure

Figure 3.13

ram. A pressure of approximately 1200 psi was applied to both lines. If the endblocks attempt to rotate relative to one another, one ram has a tendency to shorten while the other extends. This differential movement causes oil from the pressure side of the cylinders to move to the return sides of the coupled ram, causing an equal and opposite pair of forces which prevent the endblocks from rotating. Unlike the rams used to load the columns, the rams in the leveling system were not controlled and operated independently of the main loading system. Earlier tests [13,14,16,24,25] performed using this leveling system showed its utility, and results of this experimental program further confirm this.

3.5.4 Electronics. As mentioned above, the main loading rams were controlled electronically in a closed loop configuration. Figure 3.14 is a schematic of a closed-loop system. In this type of system, each ram is fitted with a servovalve, which is simply a device which electronically controls the amount of oil allowed to flow into the ram. It is nothing more than an electromagnet and a spring-loaded pin valve, so that the more current which flows to the servovalve, the stronger the attraction between the pin and the magnet and the wider the valve is opened. The valve current is controlled by a servocontroller, which monitors the state of the system and adjusts the servovalve as needed. The servocontroller uses the

CLOSED-LOOP CONTROL SYSTEM

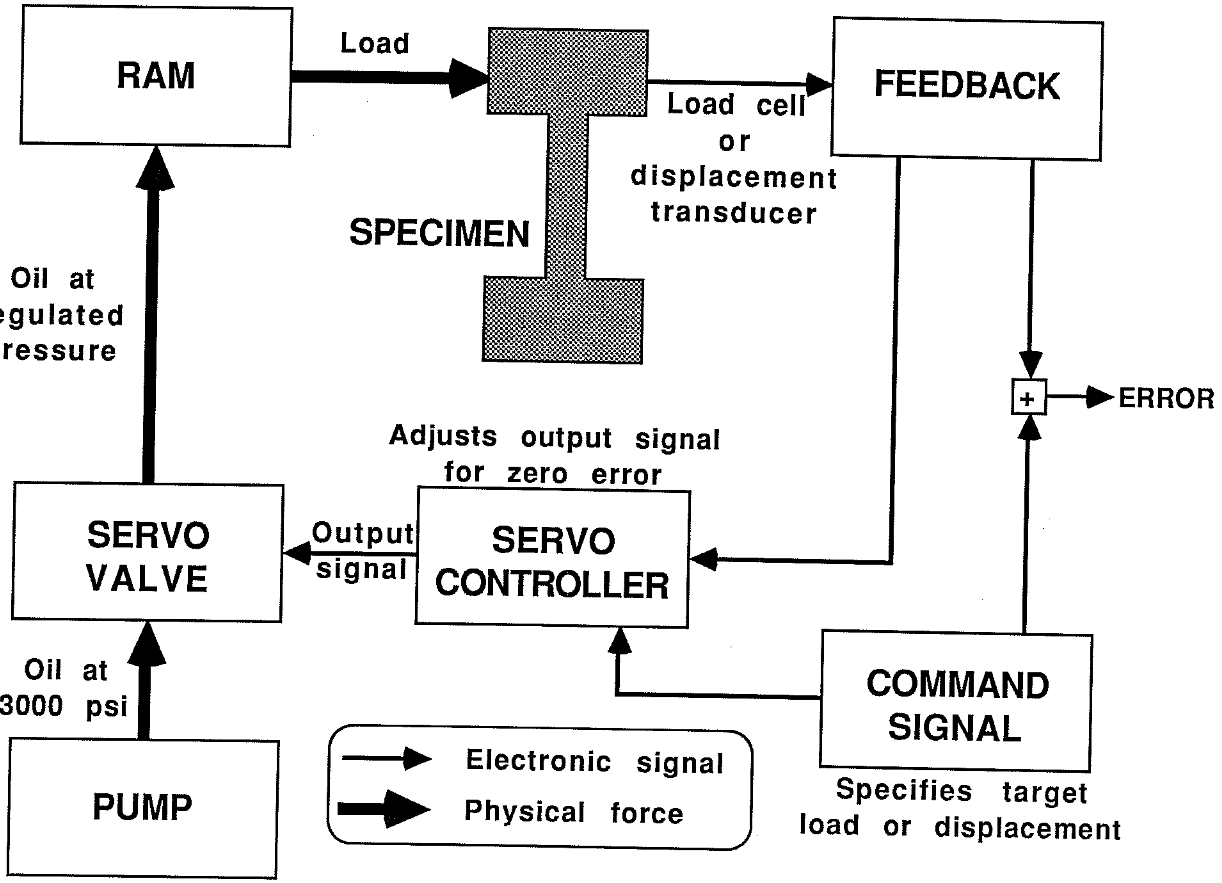

RAM
Load
SPECIMEN
Load cell or displacement transducer
FEEDBACK
Oil at regulated pressure
+
ERROR
Adjusts output signal for zero error
SERVO VALVE
Output signal
SERVO CONTROLLER
Oil at 3000 psi
PUMP
Electronic signal
Physical force
COMMAND SIGNAL
Specifies target load or displacement

Figure 3.14

feedback from the instrumentation of the specimen, either a load cell or a displacement transducer, to determine how much to open or close the servovalve. This is done by comparing the feedback signal with the command signal. Any difference between these two signals is called an error, and the current sent to the servovalve is proportional to the error. Therefore, if there is an error in the system, the servovalve is adjusted until the error is zero. Any change in the command signal will cause a temporary error which will cause the valve to be adjusted to eliminate the error. This is the basic principle behind a closed loop system. Pegasus servocontrollers and Moog servovalves were used in this experimental program.

3.5.5 Loading configuration. There are as many ways to configure a closed loop system as there are feedback signals to monitor. In this experimental program, two different configurations were used -- load control and displacement control. Load control uses a feedback signal from a load cell to control the state of the system. This means that the command signal which is sent to the servovalve is always related to a load. This is, in general, a dangerous and undesirable way to perform closed-loop tests which approach ultimate strength of the specimen because a slight increase in the command signal near ultimate load can cause complete destruction of the specimen.

Figure 3.15 is a typical load-displacement curve for a structural member. If the system is operating under load control, the y-coordinate (load) of the graph is specified and the specimen responds with an x-coordinate (displacement). As the load increases, the displacement increases. When the ultimate load is reached, the response changes: the graph turns and the load decreases as the displacement increases. Under load control, if the servocontroller is commanded to find a load higher than ultimate, it would keep opening the valve wider, but since there is no displacement corresponding to that load, the error would never reach zero. As long as an error was sensed by the servocontroller, the ram would continue to extend until the specimen was destroyed. Furthermore, if the load were decreased in an attempt to obtain the post-ultimate portion of the curve, the deflection would decrease as well because the lower displacement is a lower strain energy state. Therefore, under load control, the post-ultimate portion of a load-displacement curve is inaccessible. This problem is avoided by operating under displacement control.

Under displacement control, the feedback signal originates from a linear potentiometer displacement transducer mounted on the ram piston. When the system is configured in this way, the command signal from the servocontroller is related to the displacement of the ram piston. Unlike load control, all

TYPICAL LOAD-DEFORMATION CURVE

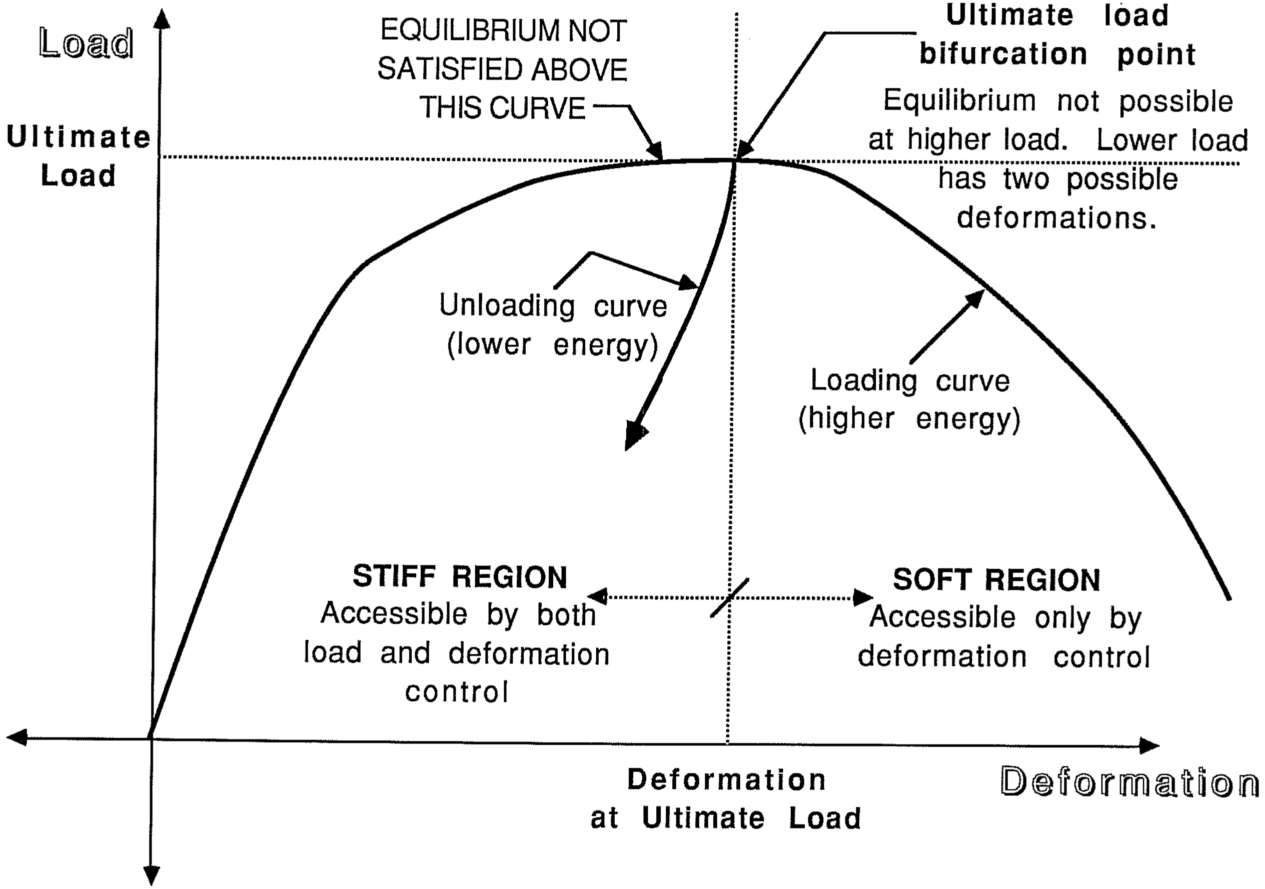

Load
Ultimate Load
EQUILIBRIUM NOT SATISFIED ABOVE THIS CURVE
Ultimate load bifurcation point
Equilibrium not possible at higher load. Lower load has two possible deformations.
Unloading curve (lower energy)
Loading curve (higher energy)
STIFF REGION
Accessible by both load and deformation control
SOFT REGION
Accessible only by deformation control
Deformation at Ultimate Load
Deformation

Figure 3.15

portions of the specimen response, including pre- and post-ultimate behavior, can be obtained. Electronically, however, there is virtually no difference between the two configurations.

In this experimental program, displacement control was the primary configuration used, and load control was not used unless it was coupled to a system under displacement control. For some tests, the axial load was required to be proportional to the lateral load. To accomplish this, the axial and lateral systems were coupled electronically. In this configuration, the lateral system operated under displacement control and the axial system under load control. The lateral ram was also equipped with a load cell, and the signal from this load cell was conditioned, amplified, and used as the command signal for the axial system (see Fig. 3.14 and 3.16). In this way, the axial system, in a sense, operated under lateral displacement control. The loading program was applied by commanding a lateral displacement, which produced a lateral load. The lateral load produced a signal from the lateral load cell that was amplified and sent as a command to the axial servocontroller, which adjusted the axial load appropriately. The ratio of the axial load to the lateral load can be controlled by the amplification of the lateral load cell signal. A schematic of this coupling system is shown in Fig. 3.16.

COUPLING SYSTEM

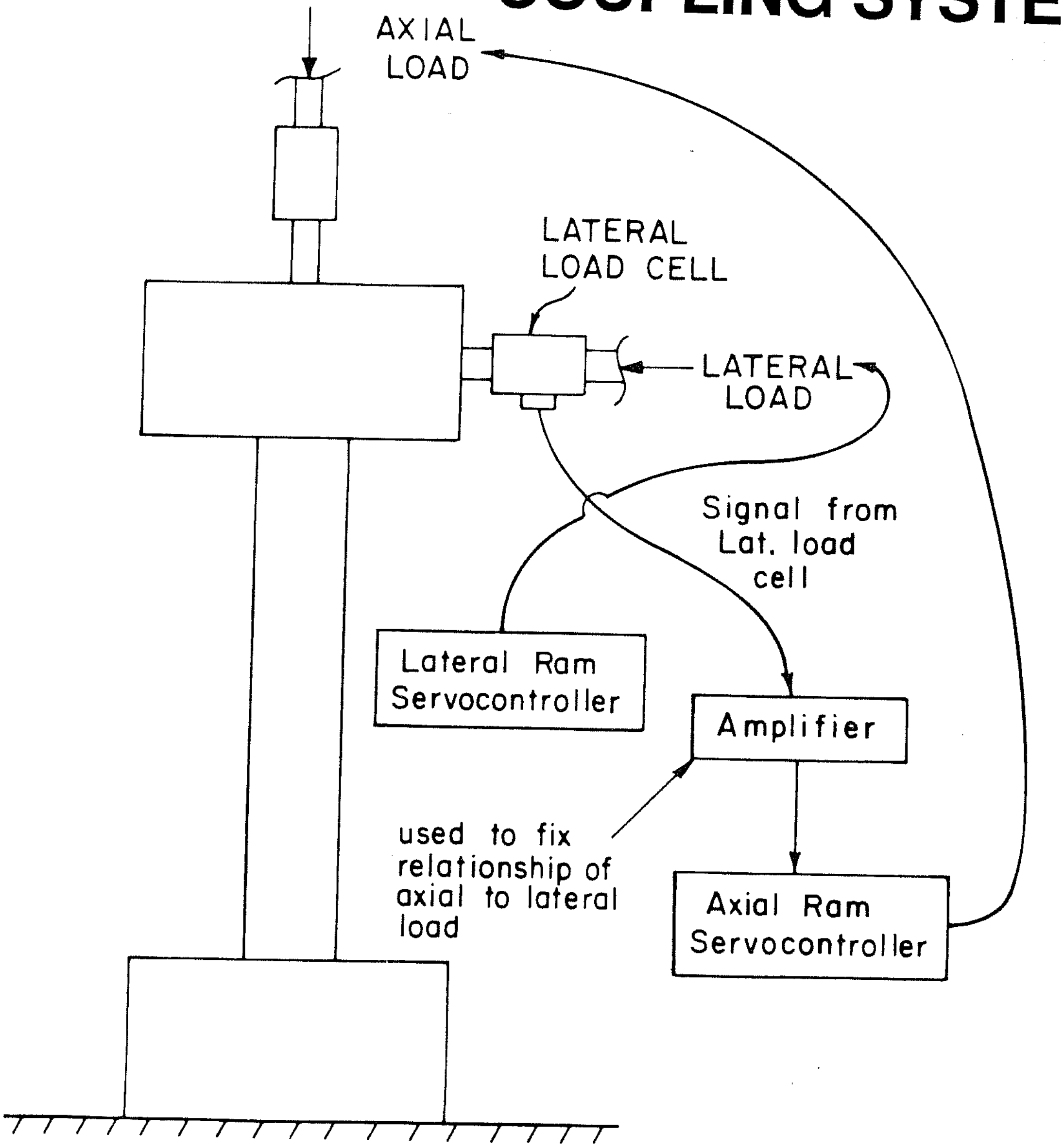

AXIAL LOAD
LATERAL LOAD CELL
LATERAL LOAD
Signal from Lat. load cell
Lateral Ram Servocontroller
Amplifier
used to fix relationship of axial to lateral load
Axial Ram Servocontroller

Figure 3.16

3.5.6 Instrumentation and data acquisition. Data was collected during testing from a number of different sources. Load cells, displacement transducers, strain gages, dial gages, and real-time plots were used to monitor the response of each column specimen during loading. Each of these data sources is discussed in detail below. Figure 3.9 shows the location of the instrumentation on the specimen.

The object of the data collection was to obtain enough information during the test to be able to interpret the behavior of the specimen. In three tests, lateral load and lateral crosshead displacement were monitored to determine when data should be taken so that real-time plots could be accurately represented by connecting data points with a series of straight lines. In the fourth test, the load path was very complex, so load steps were determined before the test based on the load steps of the other three tests.

During the test, selected information was plotted using a Houston Instruments HI-2000 plotter. This information was used to continuously track selected data sources throughout each test. These plots are referred to as real-time plots. Depending on the test, plots were lateral load vs lateral crosshead displacement, axial load vs lateral crosshead displacement, or axial load vs lateral load. These plots were also used to reconstruct data which was inadvertently lost due to system malfunction. These are described in greater detail in Chapter 4.

Five load cells were used to monitor loads applied to the specimen. Axial and lateral rams were mounted with 300 and 150 kip Lebow load cells, respectively. Load cells were also placed on the north, east, and west leveling rams to monitor the moment applied to the endblocks.

Ten displacement transducers were placed on the column and top endblock. Linear potentiometers were used to measure displacements because of their accuracy and simplicity. These transducers were used to obtain a displacement profile of the column throughout each test. Dial gages were also used in the initial test to verify the electronic instrumentation, to check the fixity of the bottom endblock, and to monitor the rotation of the top endblock. Because none of these presented any problems, the dial gages were not used in later tests.

Strain gages were placed on the reinforcing steel at critical sections of the column as shown in Fig. 3.9. These gages were used to monitor the strain of reinforcing steel during each test. Strain data can also be used to verify existing models which use strain information to determine overall column behavior.

To support displacement transducers and lead wires, a small instrumentation frame was built. This frame, made up of light angles and tubes, held the linear potentiometers in place on the specimen independently of the load frame. This allowed for potential problems to be diagnosed more easily because the load and displacement readings were uncoupled. Figure 3.11 shows a specimen installed in the loading and instrumentation frames.

Data was collected using a Hewlett-Packard HP-3497A Data Acquisition System with the aid of an HP-86 microcomputer. The data sources (load cells, strain gages, etc.) were connected to the HP system through a front-end unit built by the technicians at the Ferguson Laboratory. The system has a capacity of thirty full-bridge channels for load cells and displacement transducers in addition to thirty quarter-bridge channels for strain gages. Data was collected by scanning each channel with the HP-3497A and sending the output to the HP-86. Using software written at Ferguson Laboratory, the HP-86 converted the scanned data, which was transmitted as voltages, to engineering units appropriate for the given data source. Voltages and engineering units were then printed by a Texas Instruments Omni 800 printer and stored on a floppy disk. Once the floppy disk was filled (about 200 data points), a new data disk was inserted into the disk drive and the test resumed. As a result, there was virtually no limit to the amount of data which could be collected during a given test.

Once the test was completed, the data was processed and converted to a format compatible with an IBM PC. SuperCalc4 [27], a spreadsheet program, was then used to manipulate and plot the data. Most of the graphs in this thesis were generated with SuperCalc4. Test data is presented and discussed in Chapter 4.

3.6 Loading Programs

Three loading programs were used in this study to investigate the response of reinforced concrete columns subjected to variations in lateral and axial load. While columns with constant axial load and varying lateral load have been extensively studied in the past, the effect of varying axial load in combination with varying lateral load on reinforced concrete columns is not well understood.

The first specimen, C - HA, was loaded with the program shown in Fig. 3.17. This load program is representative of an exterior column in a tall moment-resisting frame. In moment-resisting frames of the type shown in Fig. 3.18, the axial and lateral forces vary at approximately the same frequency, provided the column is not pushed far into the inelastic region. This response results in a loading program in which the axial and lateral forces can be expressed in a linear equation, $P = A * H + B$, where P is axial load and H is lateral load. In this form, the value of A would correspond to the variation in axial load

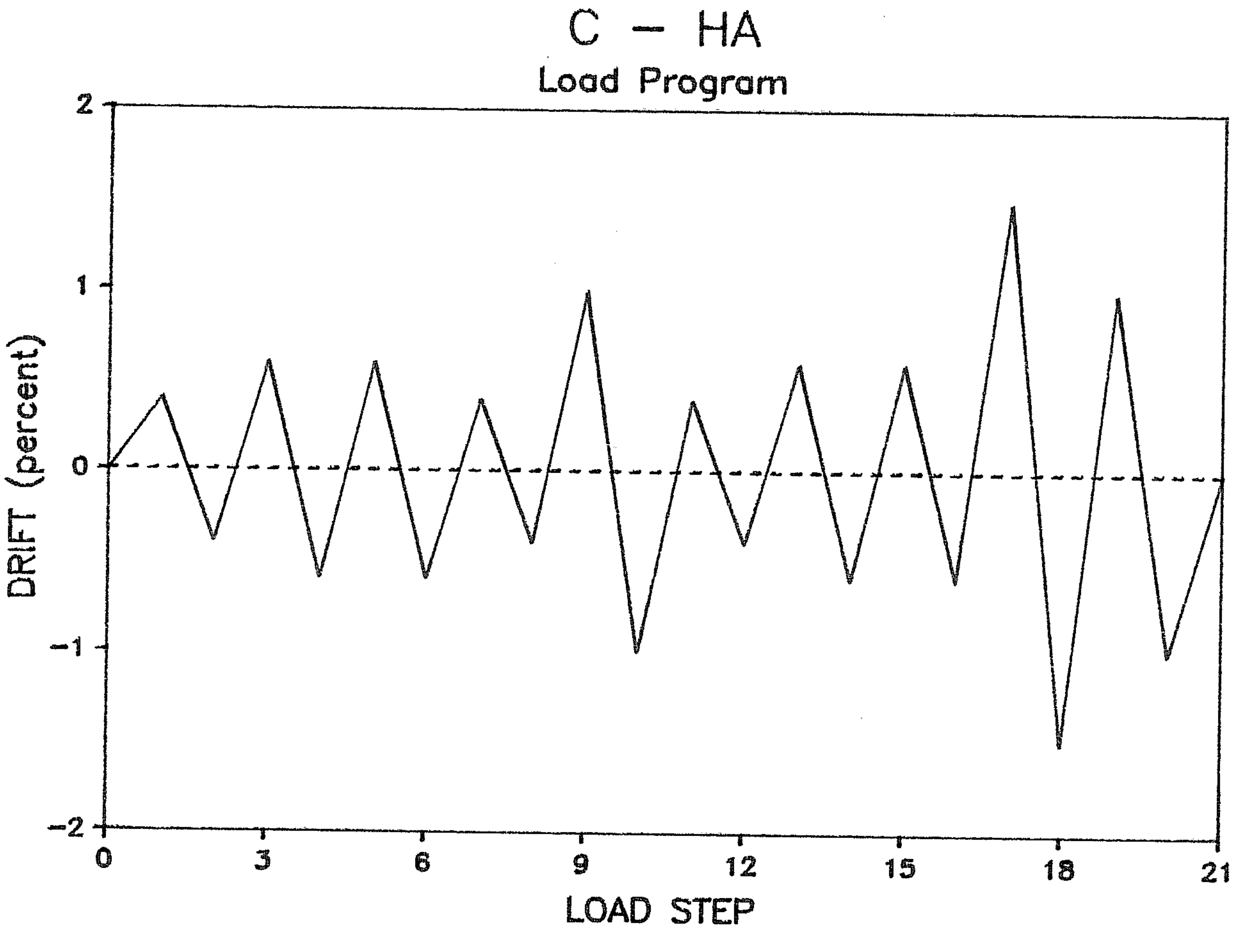

C – HA
Load Program
DRIFT (percent)
2
1
0
−1
−2
0
3
6
9
12
15
18
21
LOAD STEP

Figure 3.17

TYPICAL PLANE FRAME

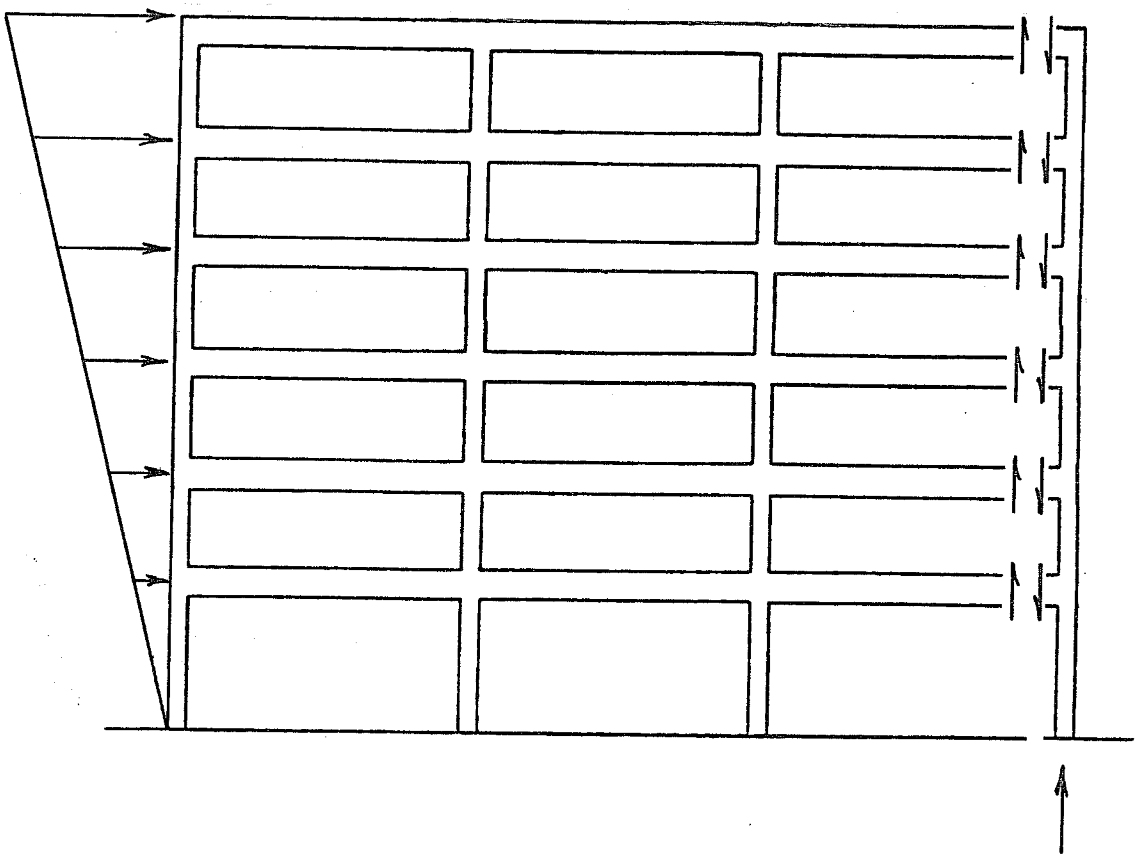


Figure 3.18

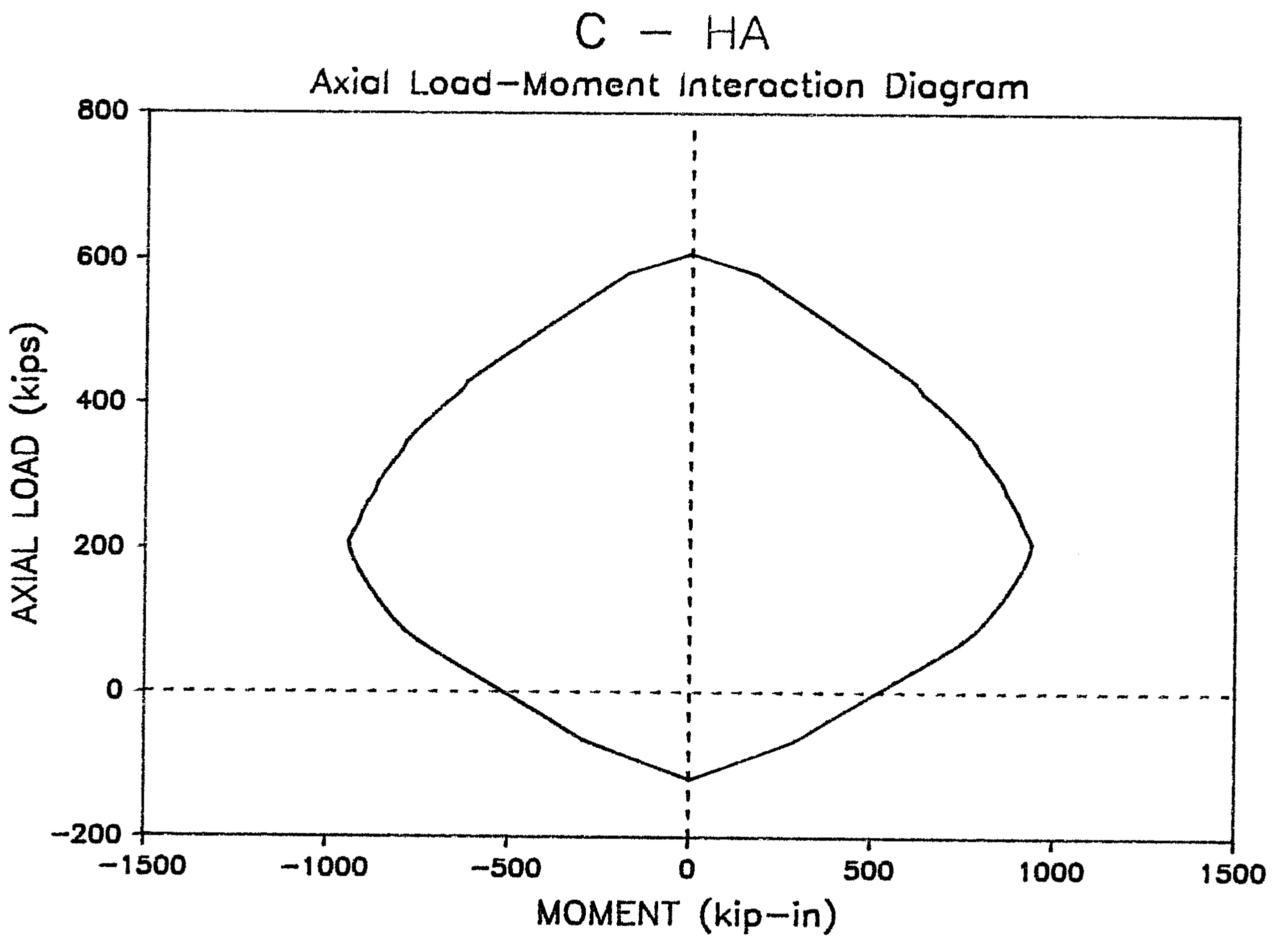

C – HA
Axial Load–Moment Interaction Diagram
800
600
400
200
0
–200
AXIAL LOAD (kips)
–1500
–1000
–500
0
500
1000
1500
MOMENT (kip–in)

Figure 3.19

expected to occur at a load above the balanced load condition: concrete was expected to crush before the reinforcing steel yielded.

The loading program used in the second and fourth tests, C - LA - 1 and C - LA - 2, was of the same form as that used for C - HA. The principal difference between the C - HA program and the C - LA programs was in the amount of variation in the axial load, which is expressed by the value of A. The coefficient A is significantly smaller in the C - LA programs than in the C - HA program (3.31 and 4.08 for C - LA - 1 and C - LA - 2, respectively), which means that C - LA is representative of a shorter moment-resisting frame. So, while both programs are representative of exterior columns in moment-resisting frames, they differ in the amount of variation in the axial load. This difference was expected to make a qualitative difference in the response of the column because the failure of C - HA was expected to be above the balance point (supposedly a brittle failure mode), while the failure of C - LA was expected to be in the ductile region below the balance point. The load program used for tests C - LA - 1 and C -LA - 2 is shown in Figs. 3.20 and 3.21. Notice that the C - LA - 1 program is slightly different than the C - LA - 2 program in the number of cycles to each drift level.

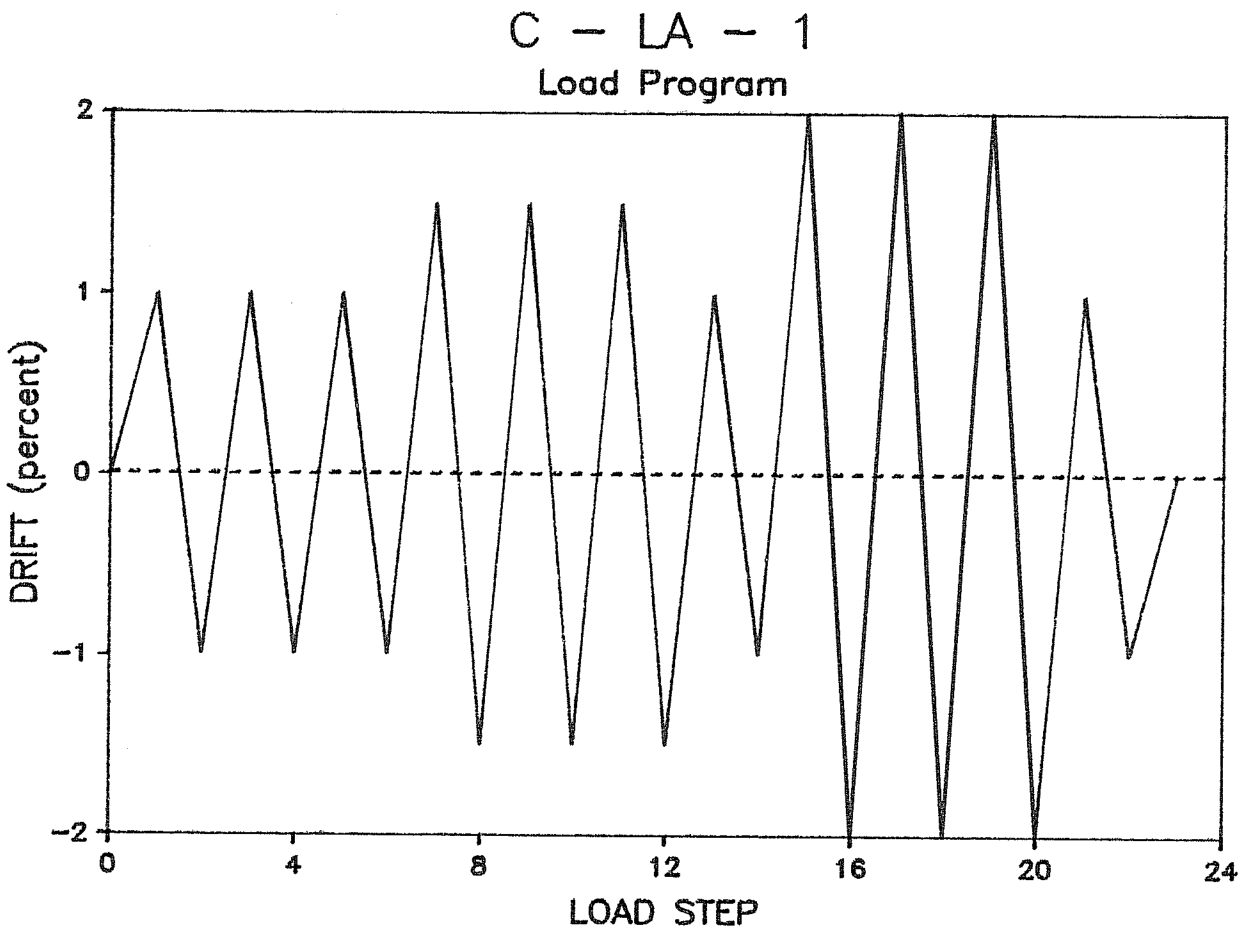

C – LA – 1
Load Program
DRIFT (percent)
2
1
0
-1
-2
0
4
8
12
16
20
24
LOAD STEP

Figure 3.20

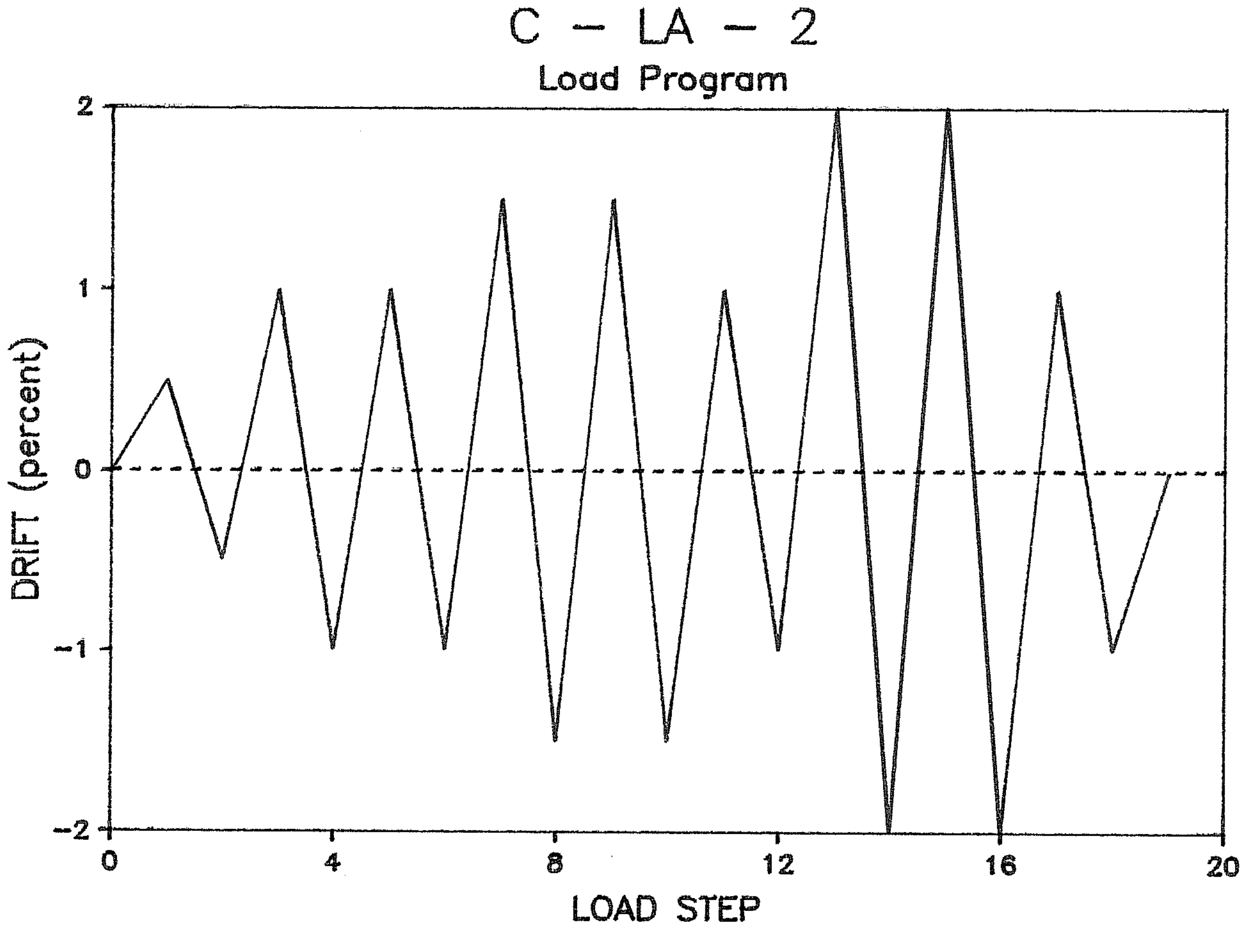


Figure 3.21

The loading program for the third column was radically different from the loading program of both C - HA and C - LA. The loading program of this specimen, called I - HA, was based on an analytical investigation of the Imperial County Services Building. This structure had two very different lateral force resisting systems in orthogonal directions -- a flexible moment-resisting frame and a stiff shear wall system -- so that the corner columns experienced variations in axial and lateral loads which were not linearly related. The reason this occurred was that both systems contributed axial forces to the corner columns while only one system -- the moment-resisting frame -- contributed significant lateral forces. The load program used for I - HA, which is based on the investigation in Ref. 21, is shown in Fig. 3.22 and Table 3.3.

3.7 Testing Procedure

The test frame required some modification prior to the initial test. The previous series of column tests using this frame had been performed on columns which were three feet tall, while the specimens in this experimental program were five feet tall. Therefore, the loading crosshead was moved up two feet in order to accomodate the taller specimen. Also, the north leveling ram was removed to facilitate the placement of the specimen.

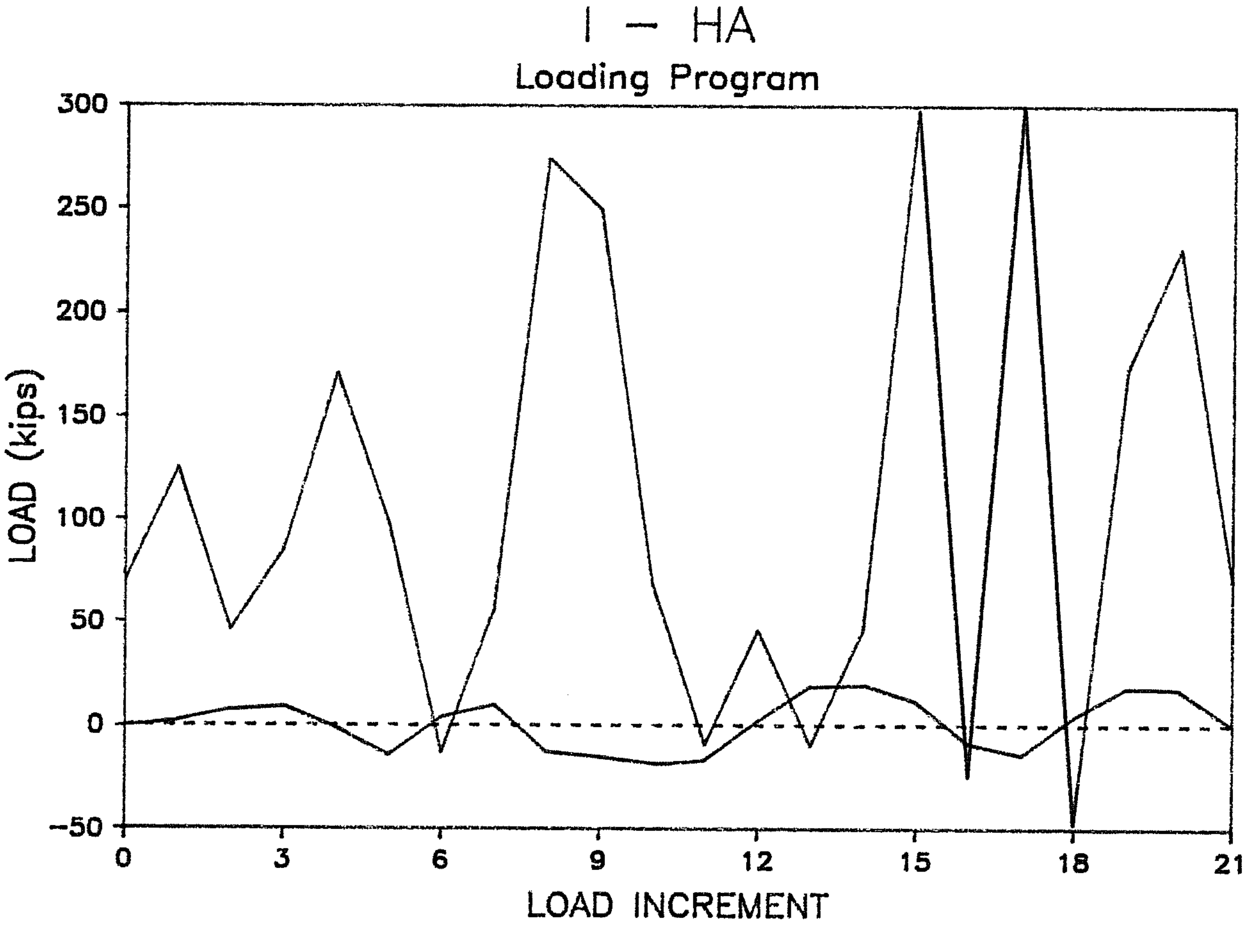

I – HA
Loading Program
LOAD (kips)
300
250
200
150
100
50
0
−50
0
3
6
9
12
15
18
21
LOAD INCREMENT

Figure 3.22

LOADING PROGRAM
SPECIMEN I - HA

LOAD STAGE	AXIAL (kips)	LATERAL (kips)
0	70	0
1	125	2.34
2	45.9	7.69
3	84.7	9.2
4	171	-1.3
5	97.7	-14.4
6	-13.1	4.01
7	55.9	9.83
8	274	-13.0
9	249	-15.4
10	67.9	-18.5
11	-9.2	-16.3
12	45.9	2.28
13	-9.7	19.0
14	46.2	19.7
15	298	11.4
16	-24.4	-8.72
17	300	-14.1
18	-47.5	4.44
19	173	18.4
20	231	17.1
21	70	0

Table 3.3

Once modifications to the loading frame were completed, the first specimen was prepared for testing. Each endblock was fitted with a device which provided external prestressing in order to keep the endblock from cracking and the longitudinal reinforcement from experiencing anchorage failure. Also, displacement transducer contact points on the face of the specimen were covered with a small square of plexiglass to provide a smooth surface for displacement readings.

The specimen was then moved into the empty load frame using a forklift. The load cap was then maneuvered into position over the top endblock using the loading and leveling rams. Four seven-eighths inch diameter high-strength steel rods were used to connect each endblock to the load cap. The specimen was lifted and moved with the loading rams so that the bottom enblock was aligned with the base of the test frame. Once the specimen was properly aligned, a one-half inch layer of Hydrostone (gypsum cement) was used to fill the gap between each endblock and the test frame. The Hydrostone assured a smooth transfer of force from test frame to specimen. The eight rods were then tensioned to 80 percent of their ultimate capacity to provide clamping between each endblock and load cap.

The north leveling ram was then returned to its position in the test frame, and a pressure of 1200 psi was applied to the hydraulic lines in the leveling system. The instrumentation frame, which was without its north side in order to allow the specimen to be inserted, was completed and fastened together. Displacement transducers were then attached at the appropriate places on the instrumentation frame, as shown in Fig. 3.9. Lead wires for displacement transducers and strain gages were connected to the data acquisition system, and software for the data acquisiton system was then initialized.

The test was begun by applying pressure to the loading system; it generally took two to three seconds for the system to come to equilibrium. Zero readings for all data channels were read by the data acquisition system after which the axial load surcharge was applied. Testing was conducted according to the following procedure:

1. Increment load or displacement
2. Scan all data channels
3. Check for cracking or spalling
4. Take photographs when necessary
5. Make comments on tape if appropriate

For tests C - HA and C - LA - 1, the lateral and axial rams were coupled together electronically (as described above) so step 1 was accomplished by adding a small increment of lateral displacement. For C - LA - 2 and I - HA, the axial and lateral rams were both under displacement control and electronically independent, so step 1 involved adding a small increment of axial displacement and then adding enough lateral displacement to bring the loads to their required level. For C - LA - 2, the axial and lateral displacements were added simultaneously, and the axial to lateral load ratio was controlled by observing a real-time plot of lateral load vs axial load and tracing a line on the graph paper with a slope equivalent to the ratio of axial-to-lateral load.

CHAPTER 4

PRESENTATION OF EXPERIMENTAL DATA

4.1 Overview

In this chapter, results of each test performed in the experimental program are presented. Lateral load, axial load, moment, lateral displacement, curvature, and reinforcing steel strain data are examined. Hysteresis relationships for some of these variables are presented to aid in understanding the response of each specimen. Cracking under load is examined for qualitative information about specimen deformation. Comparisons are made between specimens, and general trends of behavior are noted.

4.2 Presentation of Data

Data for each test is presented in the form of X-Y plots and tables. Plots shown include:

- lateral load vs drift
- lateral load vs axial load
- moment vs curvature
- moment vs drift
- moment vs axial load
- load histories
- drift history

- strain gage histories
- moment-axial loadinteraction (from analysis)

A table of secant stiffnesses is also shown for each specimen, as are drawings of crack patterns at selected load stages.

Axial loads were corrected to include test setup characteristics. Loads recorded by a load cell mounted on the axial ram did not include the weight of the ram or loading crosshead. Additional axial load carried by the column due to these effects was 1500 pounds compression. This value was added to all axial loads.

End moments were calculated from static equilibrium using values obtained from load cell readings and dimensions of the test setup. Two different formulas were used to calculate moments at the top and bottom of test specimens. Top end moment was calculated from leveling ram data (see Fig. 3.8) because it provided the most direct moment information. The relationship used for top end moment determination was

Top End Moment = Leveling Load

* Distance Between Leveling Rams

where

Distance Between Leveling Rams = 96 inches

Calculation of bottom end moment could not be performed as simply as top end moment because no direct measure of the bottom endblock reaction was made. However, top end moment, axial load, lateral load, and drift were known. Therefore, bottom end moment could be calculated from global static equilibrium. The equation used for calculating bottom end moment was

Bottom End Moment = Axial Load * Top Displacement
+ Lateral Load * Height of Load
- Top End Moment

where

Height of Load = 69 inches

Top End Moment = Value determined above

Curvatures at critical sections, at the top and bottom of the specimen, were determined from strain gage data. Strain gages were located on both faces of the column, as shown in Fig. 3.5. Curvatures were estimated by averaging strain readings from bars on each face then dividing the difference between these average strains by the distance between steel layers. The equation used to calculate curvature at critical sections was

Curvature = (Strain in One Steel Layer
- Strain in Opposite Layer)
/ Distance Between Layers

where

Distance Between Layers = 8 inches

Column specimens are identified according to the type and magnitude of loading to which they were subjected. The first letter of their designation indicates whether axial and lateral loads were coupled during the test: C means the axial and lateral loads were coupled (constant eccentricity relative to the initial axial load), and I means the axial and lateral loads were applied independently of one another. The second and third letters indicate the relative magnitude of the range of axial loads to which the column specimen was subjected : HA denotes high axial loads and LA denotes low axial loads. The final character in the identifier is used only to distinguish between the different C - LA specimens : 1 and 2 denote the two principal tests, and S denotes the supplementary test performed on specimen I - HA after it was damaged.

Strain gages are identified by a three- or four-letter designation using the nomenclature shown in Fig. 4.1. The first letter specifies the gage location along the column length: B for bottom, T for top, M for midheight. The second letter designates the type of bar that was instrumented: T for transverse, L for longitudinal. The third letter specifies the column face on which the gage was mounted: N for north, S for south, E for east, W for west. The final letter, used only for longitudinal bars,

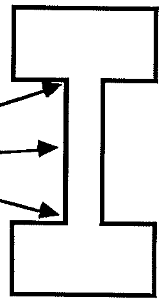
STRAIN GAGE NOMENCLATURE

FIRST LETTER: **Column Location**

T : Top

M : Midheight

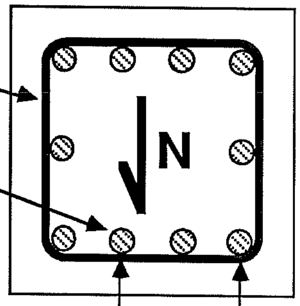
B : Bottom



SECOND LETTER: **Bar Type**

T : Transverse

L : Longitudinal



THIRD LETTER: **Section Location**

N : North S : South

E : East W : West

FOURTH LETTER: **Location in Steel Layer (longitudinal bar only)**

F : Face (non-corner)

C : Corner

Figure 4.1

indicates whether the gage was attached to a corner bar: F for face (non-corner), C for corner. Longitudinal bars are identified with four-letter mnemonics and transverse bars with three-letter mnemonics.

Lateral secant stiffness was calculated from lateral load-drift relationships. Two secant stiffnesses are calculated for each load cycle, one in the increasing compression direction (first quadrant of lateral load-drift plot) and one in the decreasing compression direction (third quadrant of lateral load-drift plot). Secant stiffnesses were calculated by dividing the maximum lateral load in each direction of a load cycle by the corresponding lateral displacement. Secant stiffnesses were used as a general measure of deterioration in lateral stiffness during testing.

4.3 Specimen C - HA

Specimen C - HA was subjected to reversed cyclic axial and lateral loads. Changes in axial and lateral loads were kept proportional during the test, effectively maintaining a constant eccentricity relative to the initial axial load placed on the column. Because the ratio of change in axial load to change in lateral load was large, the specimen was subjected to both axial tension and high axial compression during the test. The target loading program for this test was discussed in Section 3.6 and is shown in Fig. 3.17.

4.3.1. Load and drift histories. Lateral load, axial load, moment, and drift histories (Figs. 4.2, 4.6, 4.10, and 4.14) show that the loads were in phase with each other during the test. They also illustrate modifications which were made in the planned loading shown in Fig. 3.17. The modifications became necessary during the test program because pressure in the closed-loop hydraulic system was lost twice during testing due to an electronic equipment failure. The specimen was put through additional load cycles to determine if any damage was sustained when the malfunction occurred and to ascertain whether the hysteresis was stable.

4.3.2. Moment-axial load interaction. Figures 4.18 and 4.19 are the moment-axial load interaction diagram along with the moment-axial load path for the sections at the top and bottom sections of specimen C - HA. As they show, load combinations were reached which were above the balance point. Combinations above balance point are normally associated with failure in a brittle mode. Note that column response extends well outside the calculated failure envelope in the increasing compression direction, but barely exceeds calculated failure in the decreasing compression direction.

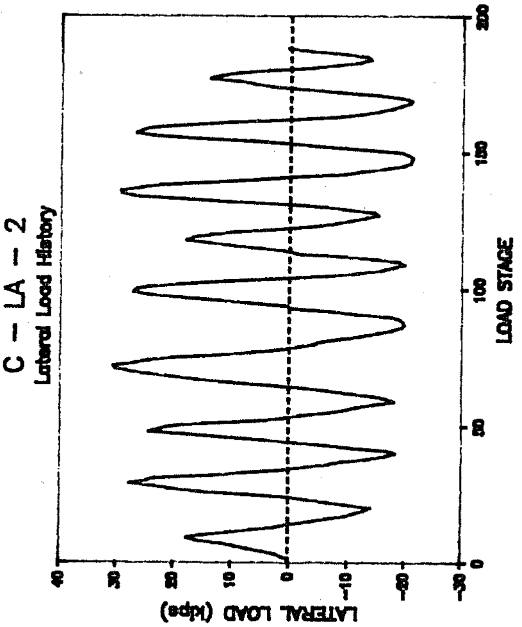


Figure 4.4

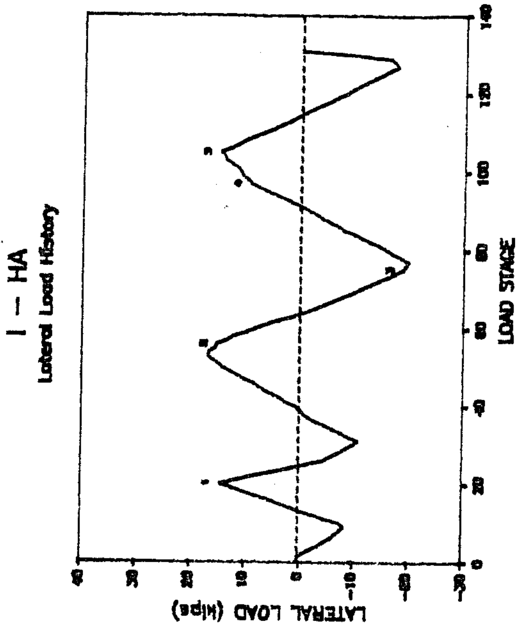


Figure 4.5

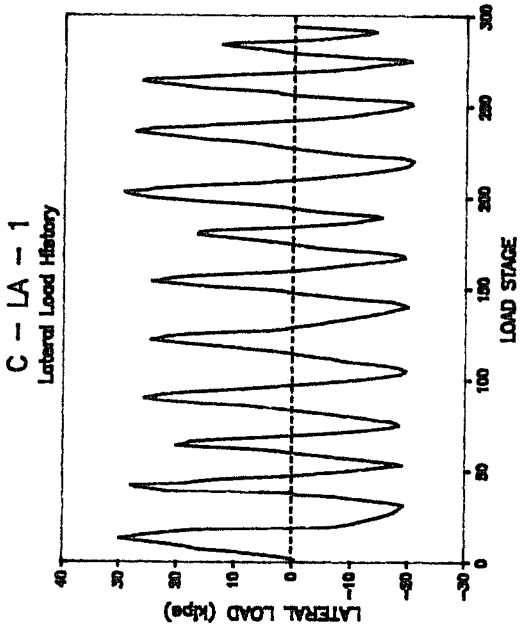


Figure 4.3

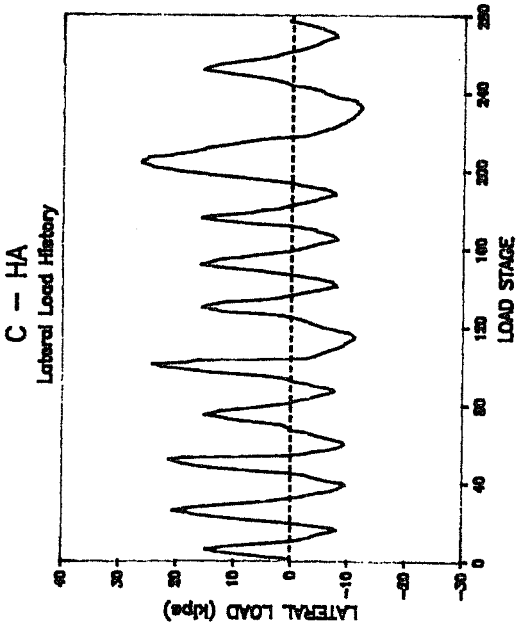


Figure 4.2

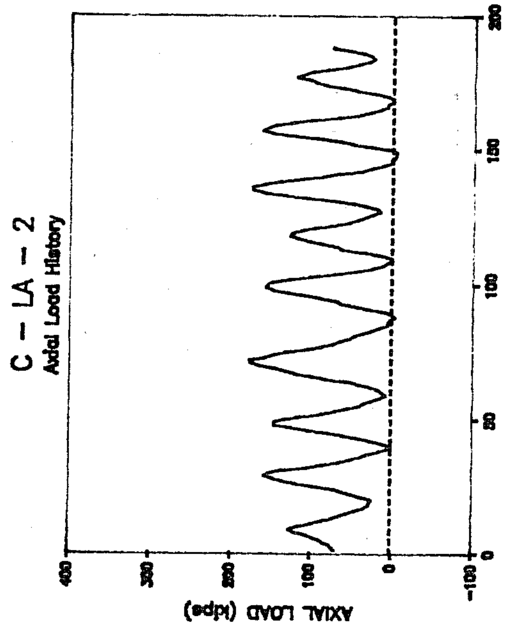


Figure 4.8

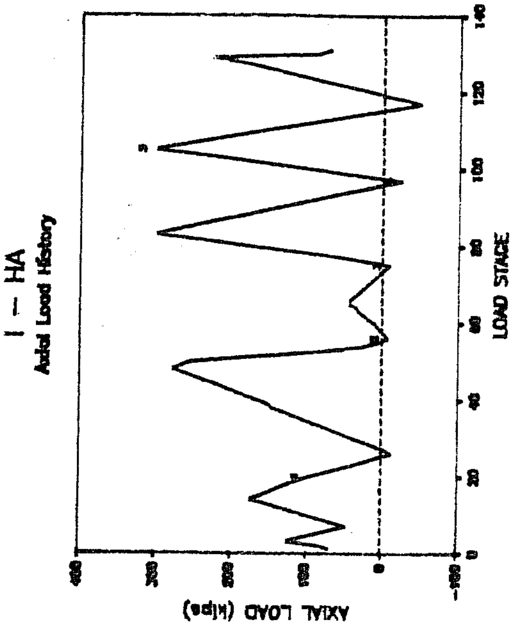


Figure 4.9

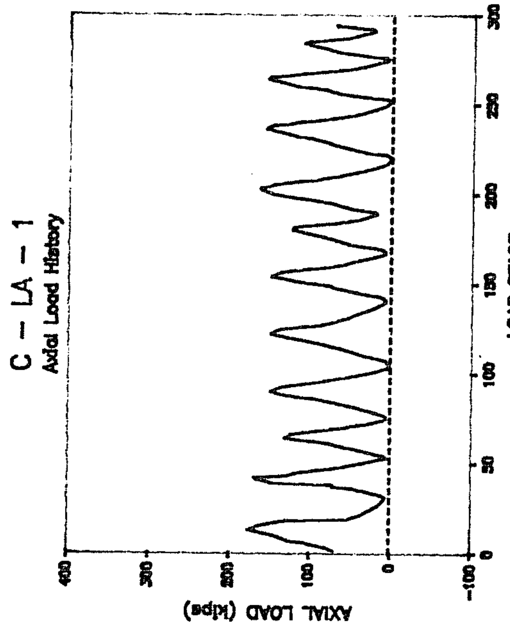


Figure 4.7

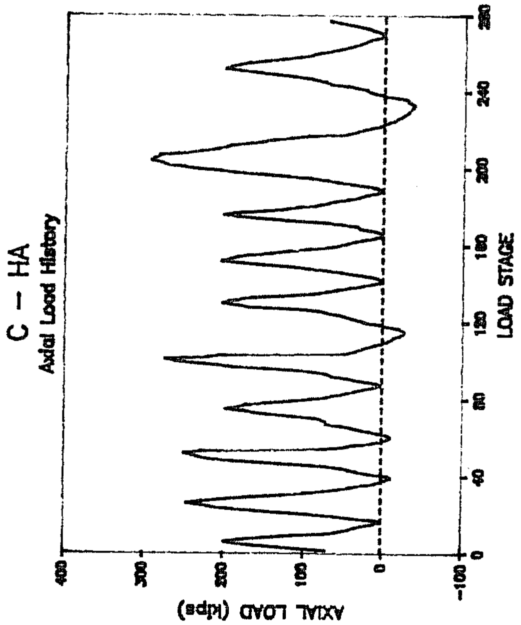


Figure 4.6

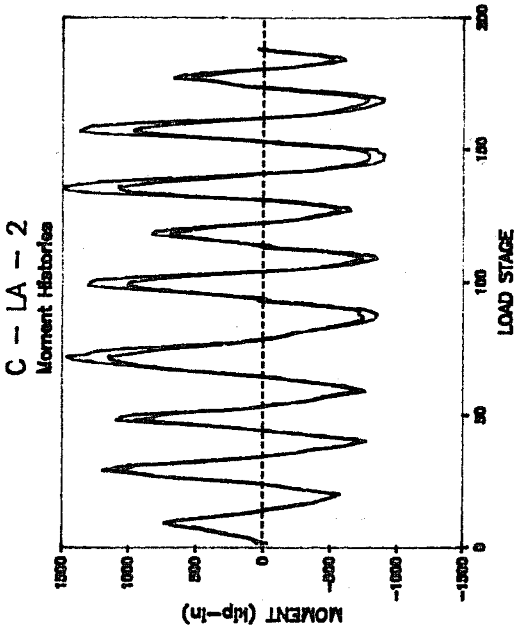


Figure 4.12

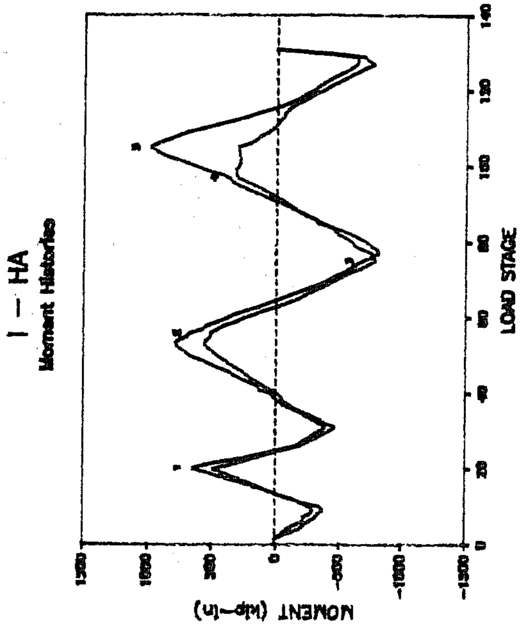


Figure 4.13

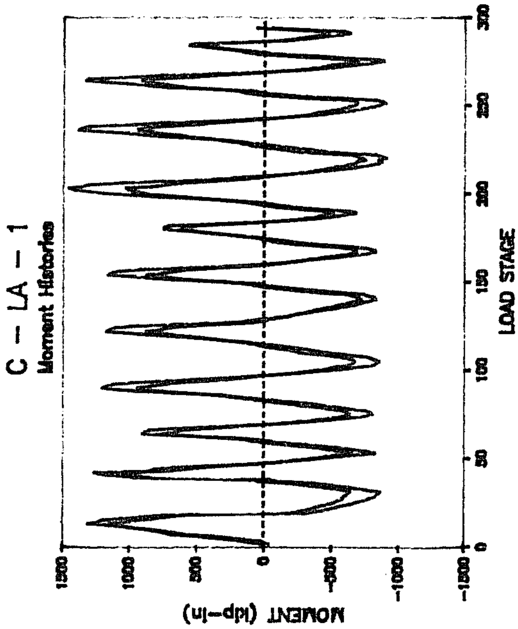


Figure 4.11

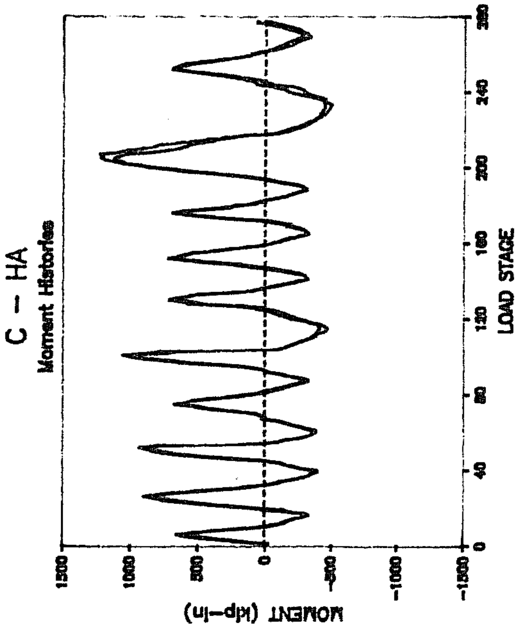


Figure 4.10

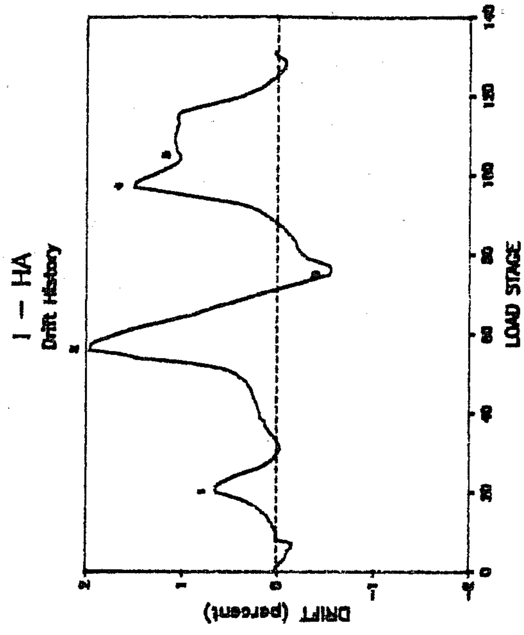


Figure 4.17

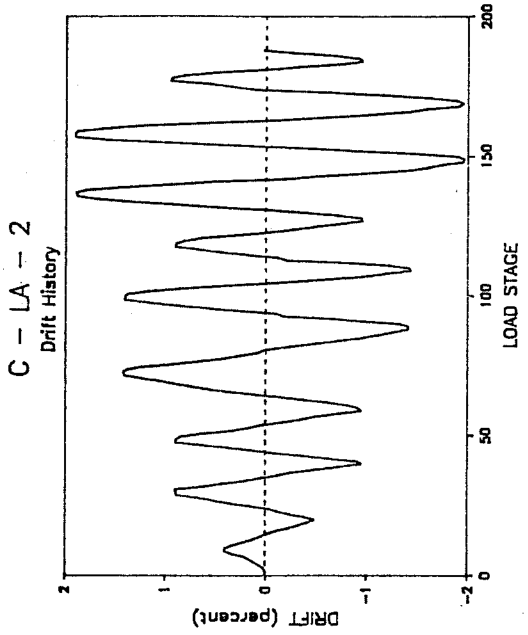


Figure 4.16

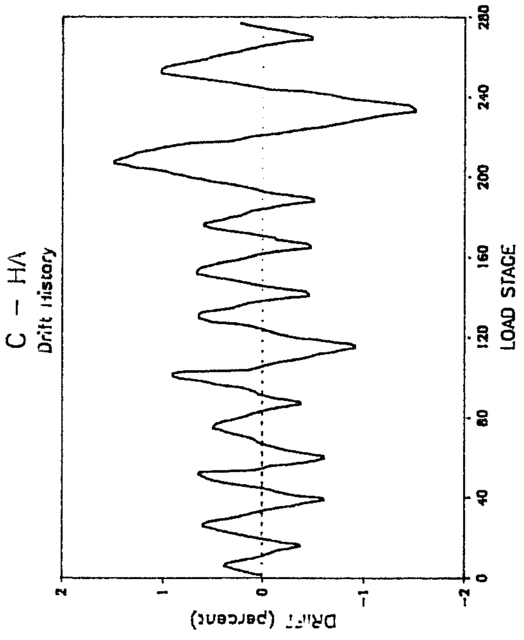


Figure 4.14

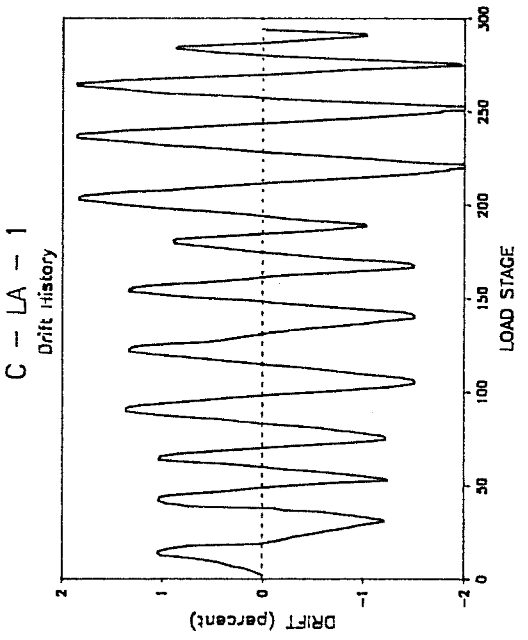


Figure 4.15

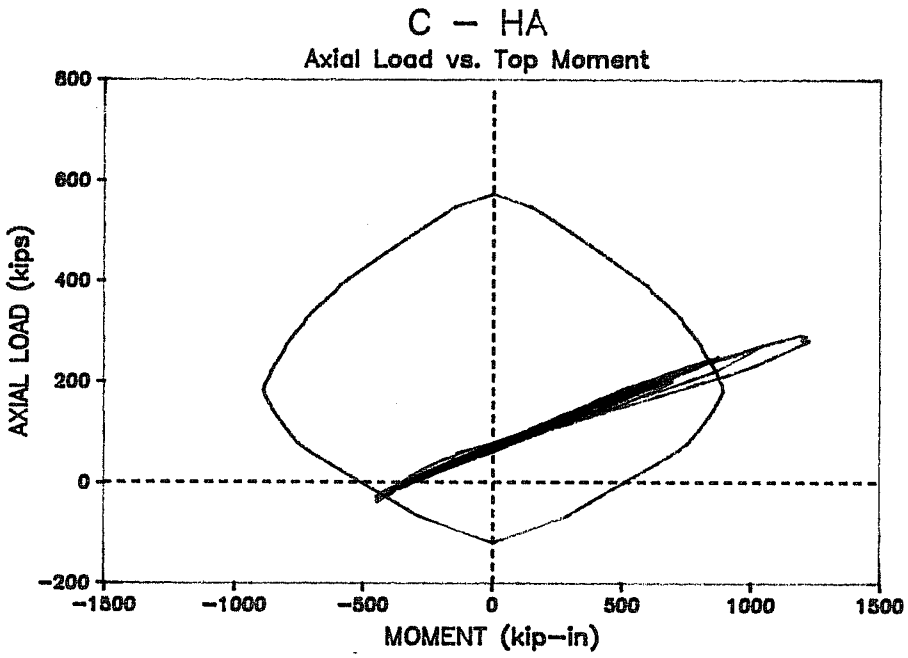


Figure 4.18

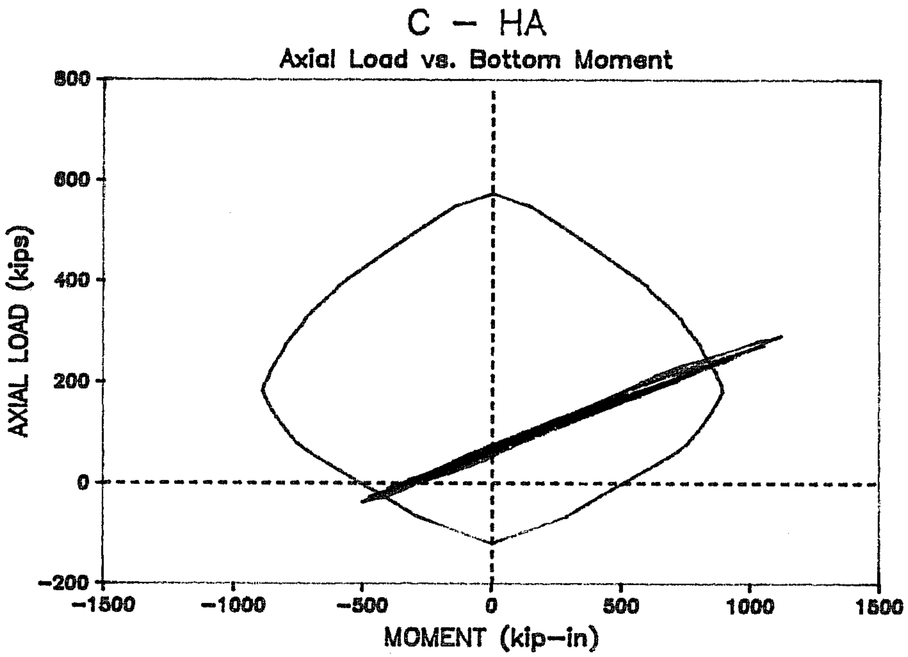


Figure 4.19

4.3.3. Lateral load-drift relationship. The lateral load-drift hysteresis relationship, shown in Fig. 4.20, is not symmetrical with respect to the origin. This dissymmetry contrasts with the hysteresis relationship of columns subjected to cyclic lateral loads under constant axial load [10-16]. Regardless of axial load level, lateral load-drift hysteresis relationships for columns under constant axial load exhibit symmetry with respect to the origin. However, specimen C - HA was not symmetrical because the changing axial load effected a change in lateral stiffness. The lateral secant stiffness, shown in Table 4.1, was generally greater under high axial compression than under low axial load for comparable levels of drift. As greater loads were experienced by the specimen, however, the difference between the two secant stiffnesses decreased. In fact, by the final cycle, the stiffnesses were nearly equal. Note, however, that these secant stiffnesses were computed at force-displacement levels much lower than previously experienced. If loading had continued to higher levels, the two secant stiffnesses would most likely not be equal although the difference between them would almost certainly be less than earlier cycles. It is important to note that despite having equal secant stiffnesses, the final cycle still exhibited some of the dissymmetry associated with varying axial load seen in earlier cycles.

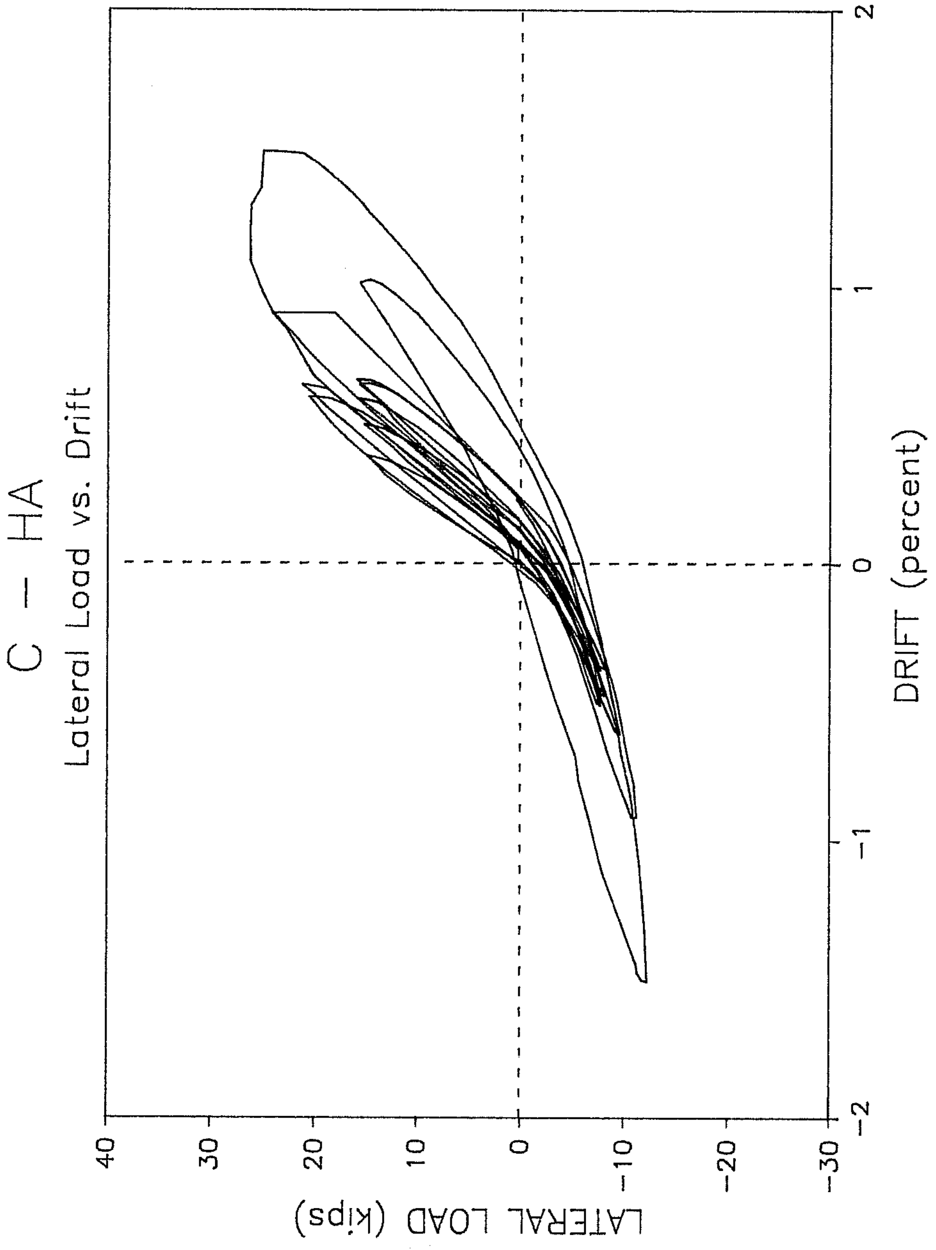

C - HA
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
-10
-20
-30
DRIFT (percent)
-2
-1
0
1
2

Figure 4.20

SECANT STIFFNESSES
SPECIMEN C - HA

CYCLE NO.	DIRECTION	LATERAL LOAD (kips)	DRIFT (in.)	SECANT STIFFNESS (kip/in)
1	+	14.9	0.23	64.7
	-	-8.2	-0.23	35.2
2	+	20.6	0.36	57.6
	-	-9.7	-0.38	25.9
3	+	21.3	0.38	55.4
	-	-9.5	-0.38	25.4
4	+	15.3	0.30	51.2
	-	-7.8	-0.24	33.0
5	+	24.2	0.54	44.9
	-	-11.3	-0.56	20.4
6	+	15.7	0.38	40.9
	-	-8.1	-0.28	28.5
7	+	16.0	0.40	40.5
	-	-8.2	-0.29	28.0
8	+	15.8	0.35	44.6
	-	-7.8	-0.31	24.9
9	+	26.3	0.65	40.3
	-	-12.4	-0.91	13.6
10	+	15.7	0.60	26.0
	-	-7.8	-0.30	26.1

+ direction is increasing compression

- direction is decreasing compression

Table 4.1

The decrease in the difference between positive and negative secant stiffness occurred primarily due to a decrease in the positive secant stiffness. The negative stiffness, which is controlled by the behavior of the steel, does not change very much between the initial and final cycles. The positive stiffness, on the other hand, decreased significantly during the course of testing. This fact is important because it indicates that the concrete is primarily responsible for the dissymmetry of the hysteresis loops.

By dividing the section response into its steel and concrete components, the influence of the concrete on the dissymmetry of the hysteresis loops becomes more evident. The stiffness contribution of the steel remained consistent during the test because the area and stiffness of the steel was not significantly affected by the load level reached in previous cycles. The stiffness contributed by the concrete, on the other hand, varied considerably between load levels and was strongly affected by previous load levels. At low axial loads, the stiffness of the specimen was dominated by steel because the compression zone, and therefore the concrete contribution, was relatively small. At high axial loads, however, the specimen stiffness was dominated by concrete because the compression zone was large. Consequently, deterioration in the concrete due to spalling and crushing more profoundly affected the positive secant stiffness. As more concrete was spalled off, the dissymmetry of the hysteresis loops became less pronounced. This indicates that the degree of dissymmetry in the hysteresis loops was primarily due to the concrete contribution to the column response.

This assertion is also supported by the material properties of steel and concrete. Steel behaves in a symmetrical fashion, that is the stress-strain relationship is the same in tension as it is in compression. Concrete, on the other hand, is highly unsymmetric because it has great compressive strength and virtually no tensile strength. Therefore, the stiffness of a reinforced concrete section is very sensitive to the amount of the section that is in compression: the greater the section in compression, the more concrete is effective, and so the higher the stiffness. At high load levels, another difference between steel and concrete has an effect on section behavior. Steel is ductile, therefore even after much inelastic action its strength is not significantly decreased. The same cannot be said for concrete, however. After loading beyond ultimate stress, the strength and stiffness of concrete decreases rapidly. This would suggest that under loads controlled by concrete behavior, such as high axial compression, concrete sections would lose their stiffness after some of the concrete reached ultimate stress. This is supported by the observations made in this test.

After an initial excursion to a given drift, subsequent excursions to the same drift resulted in a slightly lower load. This lower load level was approximately maintained for each subsequent excursion to the same drift level. New maximum load levels resulted in reduced stiffness in successive cycles, as shown in Fig. 4.20. The stiffness reduction was much more pronounced in the increasing compression direction (first quadrant of Fig. 4.20) than in the decreasing compression direction (third quadrant of Fig. 4.20). This stiffness reduction may be attributed to post-yield properties of the reinforcement and the opening and closing of cracks, but the principal reason for loss of stiffness is the reduction in the effective size of the concrete section because of cracking, spalling, and crushing of the cover concrete.

The increasing compression loop of the lateral load-drift hysteresis curve encompassed more area than the decreasing compression loop. This implies more energy was dissipated in the increasing compression direction than in the decreasing compression direction. This is interesting in that more energy was dissipated in the direction commonly associated with brittle behavior than the direction associated with ductile behavior. One possible explanation for this can be found by examining the moment-axial load path and interaction diagrams shown in Figs. 4.18 and 4.19. Loading in the increasing compression direction was well outside the calculated failure envelope, while loading in the decreasing compression direction barely exceeded the envelope. Since loop size is related to energy dissipation, which in turn is related to inelastic deformation of the column, it is understandable that the direction which exhibited greater inelastic action also had a larger hysteresis loop.

4.3.4. Strain histories. The strain histories shown in Figs. 4.21 and 4.22 were obtained from gages on longitudinal bars at sections through the top and bottom of the column specimen, respectively. Strains measured in longitudinal bars on the same face were practically identical. Also, longitudinal strains cycled in phase with applied load and in opposite directions when on opposite faces of the column. The maximum strain is shown to be well above yield strain for one face of the column. Bars on the other face, interestingly, had very low strains, indicating that the neutral axis remained relatively close to that layer of steel throughout loading.

The strain history for transverse gage TTS, which was mounted on a transverse bar, is shown in Fig. 4.23. As can be seen, this location on the bar never approached yield strain, which is not surprising because this location was not on a highly

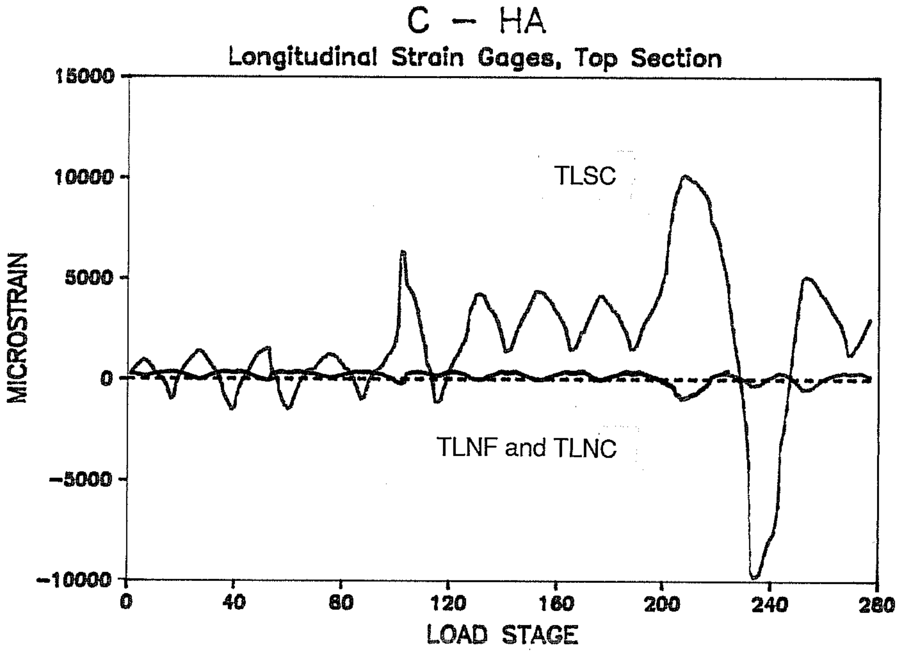


Figure 4.21

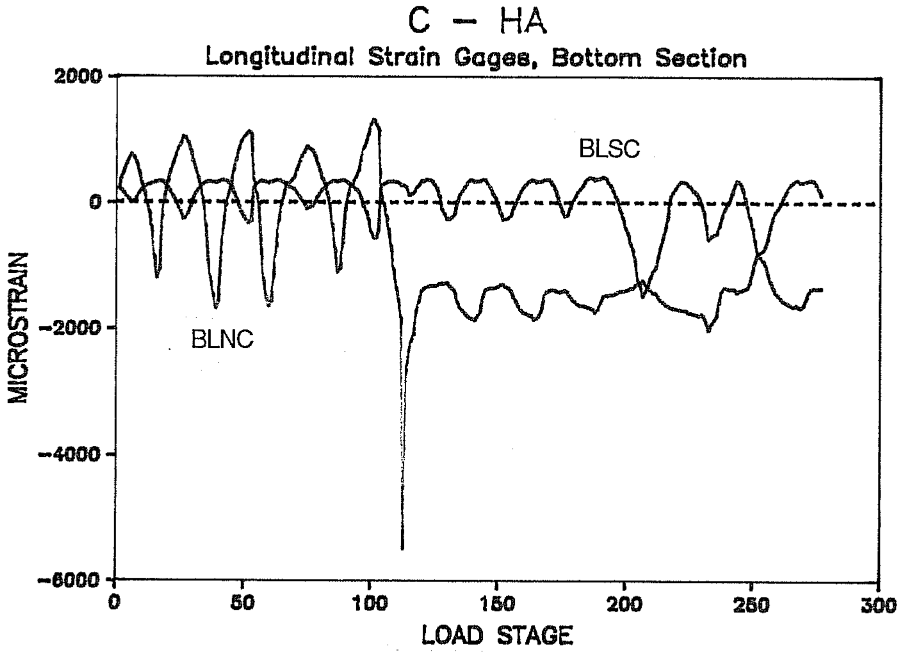


Figure 4.22

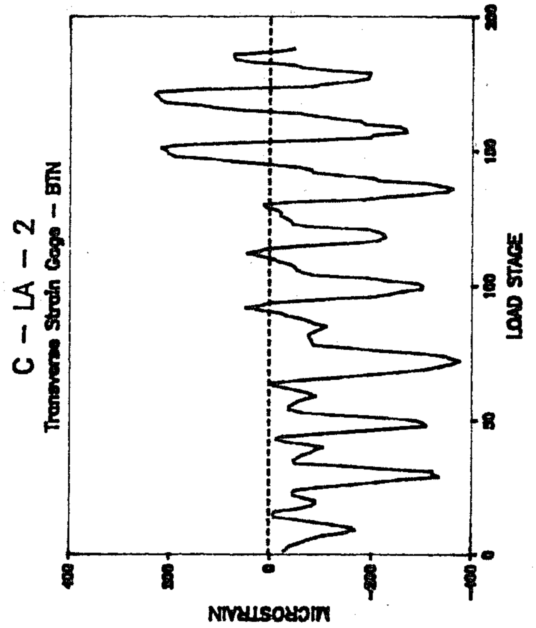


Figure 4.25

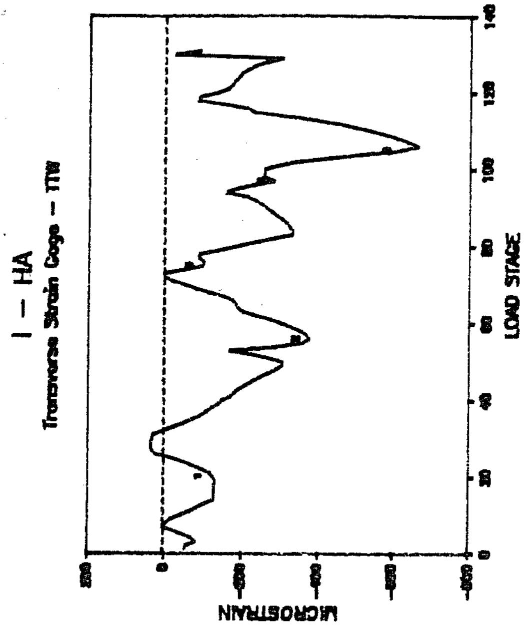


Figure 4.26

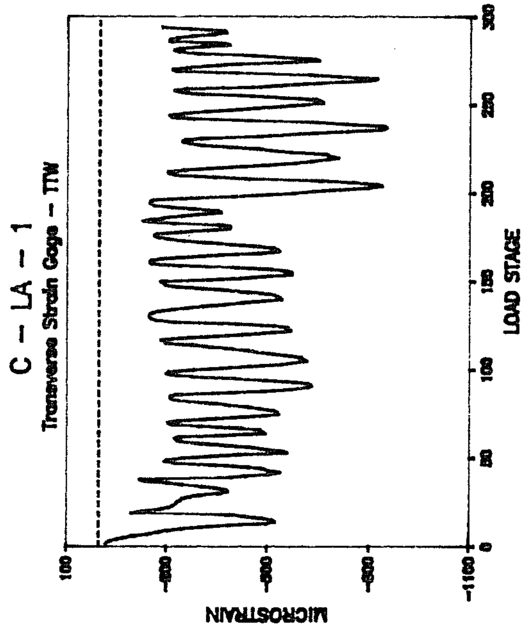


Figure 4.24

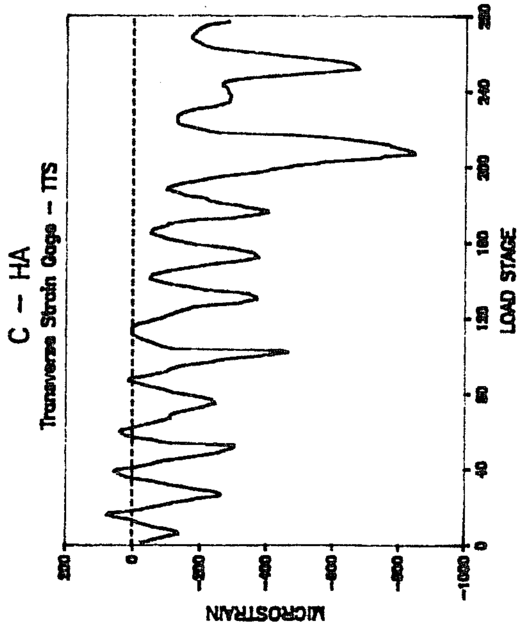


Figure 4.23

stressed face of the column and the section had closely spaced transverse reinforcement. Also, the strain of this gage changed at the same frequency as the lateral load.

4.3.5. Moment-curvature relationships. Moment-curvature hysteresis relationships (shown in Figs. 4.27 and 4.28) relationships (Figs. 4.29 and 4.30), they do show some similar characteristics. Both show the same dissymmetry due to varying axial loads and later cycles to a higher load or drift occur with a lower stiffness. Also, the loops in the increasing compression direction encompass more area than the loops in the decreasing compression direction. One difference between moment-curvature and moment-drift relationships is the offset in moment-curvature loops seen after the first excursion to a high load. In cycles subsequent to this initial excursion, the curvatures corresponding to zero moment are several times larger than for previous excursions. This offset seems to indicate permanent deformation in the top section. This permanent set was also exhibited in longitudinal steel strain histories (Fig. 4.21). However, an analogous offset is not seen in moment-drift relationships.

The difference between top and bottom moment-curvature relationships is profound, and deserving of close scrutiny. There would appear to be two possible explanations for the difference. First, strain gages used to calculate curvature may

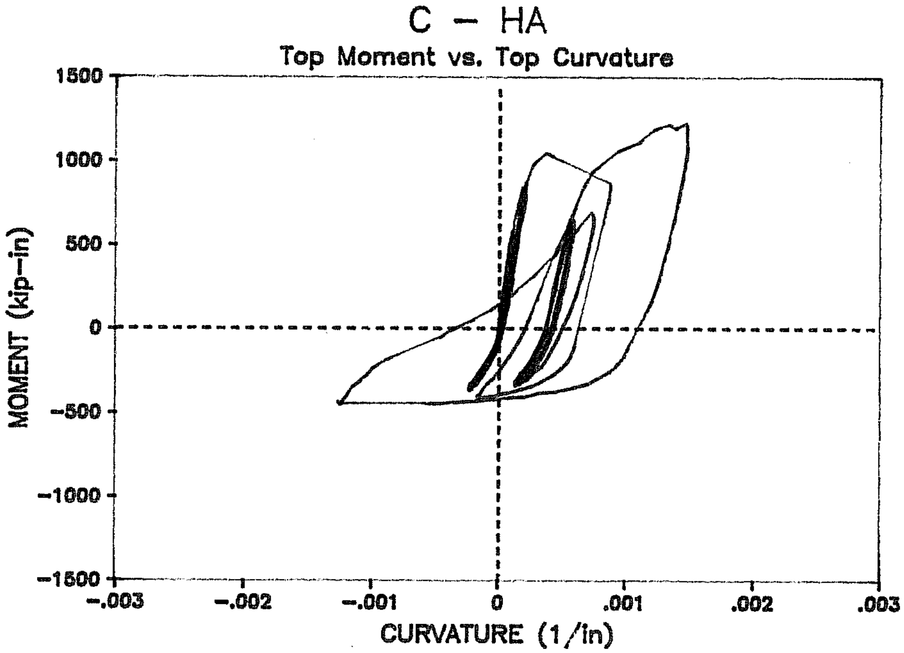


Figure 4.27

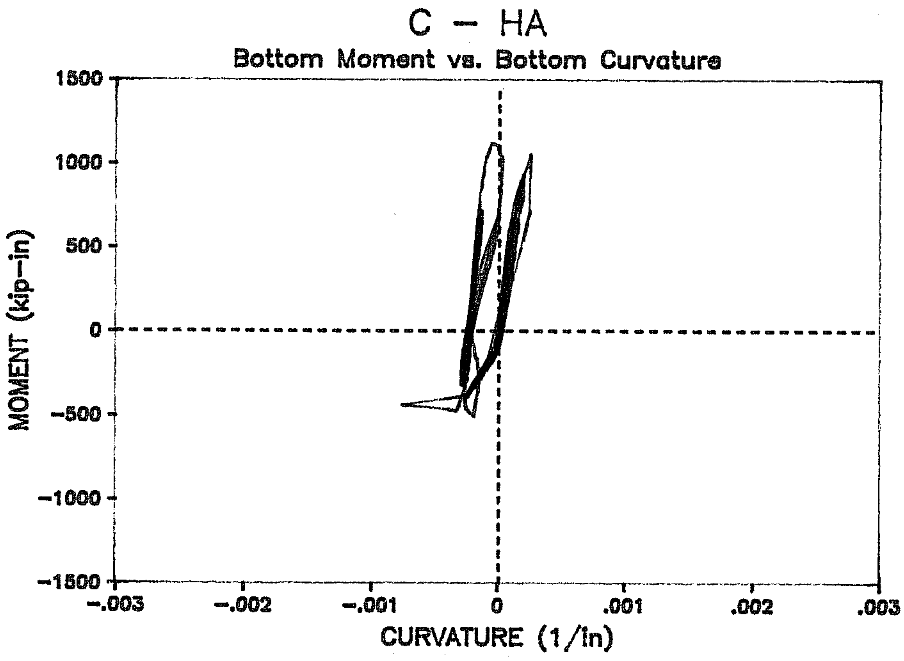


Figure 4.28

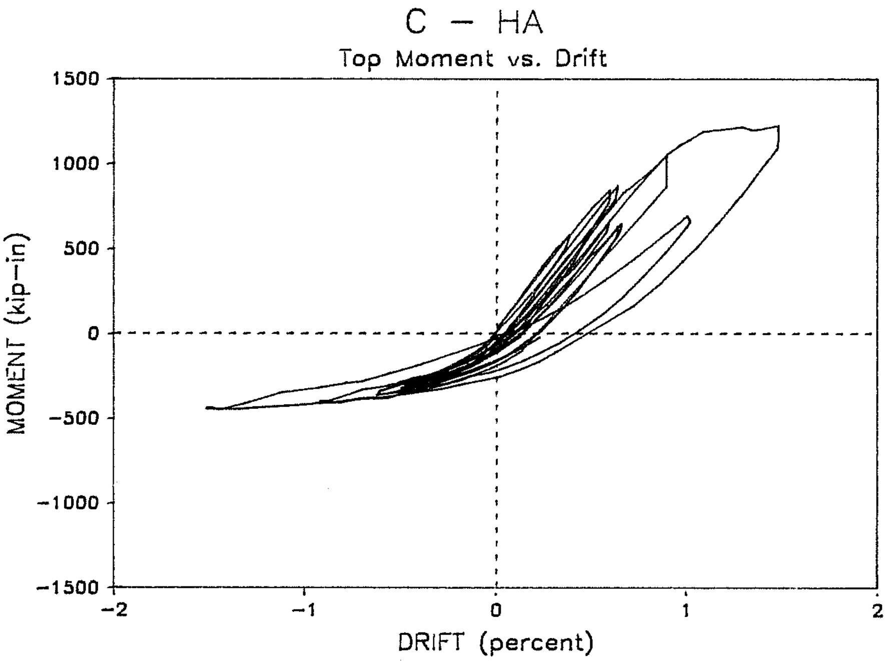


Figure 4.29

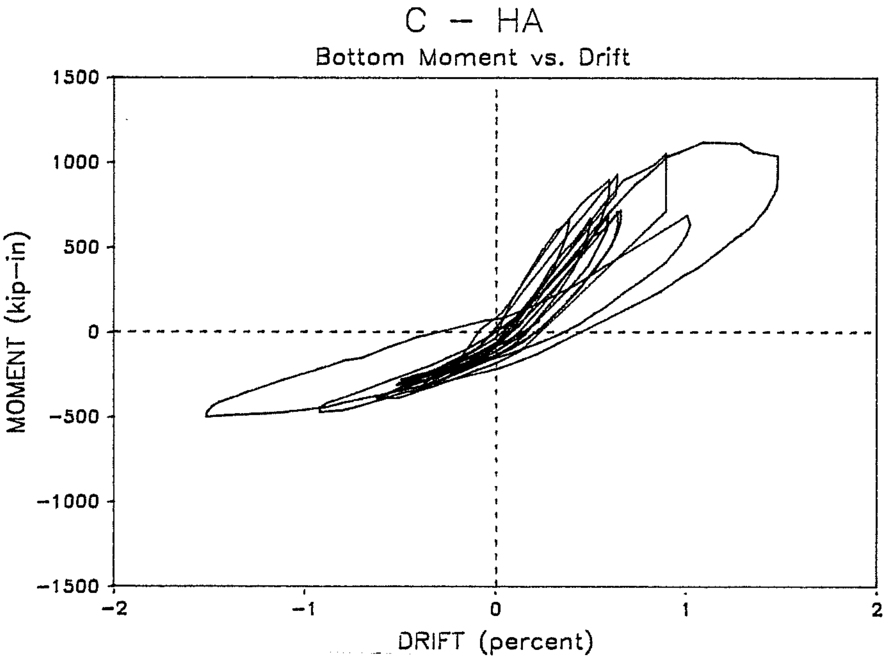


Figure 4.30

be more reliable for one section than the other. There does not appear to be much reason for asserting this, however. The second explanation could be that only the top section experienced significant inelastic behavior, while the bottom section remained almost elastic in the direction of positive moment. Why this would be so is not exactly clear, but it would seem to be the best supported assertion.

4.3.6. Cracking. As load was applied to the specimen, cracks formed and propagated. These crack patterns give a qualitative indication of how much shear and flexure influence specimen behavior. When specimen behavior is dominated by flexure, cracks propagate perpendicular to the longitudinal axis of the column. As shear begins to influence the behavior, cracks begin to extend at an incline. Theoretically, under pure shear, the angle of inclination would be 45 degrees. As long as shear contributed little to specimen behavior, it can be presumed that plane sections of the column remained plane during loading.

For load levels below ultimate, cracking in specimen C - HA was perpendicular to the longitudinal axis of the column, indicating that flexure controlled the column behavior. At very high and very low axial load levels, cracks began to turn, which suggested an increasingly important presence of shear in the column. At high load levels inclined cracking occurred because large shear forces carried by the column exceeded the capacity of the concrete. Concrete shear capacity was exceeded despite the fact that concrete shear strength was increasing as axial load was added. At low load levels, inclined cracking resulted from loss of concrete shear capacity due to decreasing axial load. Crack patterns at an early stage of the test are shown in Fig. 4.31, at a higher drift in Fig. 4.32, and after spalling of cover concrete in Fig. 4.33.

There was a significant difference in the number of cracks which formed under increasing compression compared to decreasing compression, especially at low load levels. Many more cracks formed under decreasing compression, which is not surprising since high compression has a tendency to discourage cracking. At very high load levels, this difference was less noticeable due to increased cracking in the increasing compression direction.

Cracking and spalling of cover concrete indicated the length of the hinge region at each end of the column. From measurements made after completion of the test, it appeared that the hinge length of specimen C - HA was approximately ten inches.

4.4 Specimen C - LA - 1

Specimen C - LA - 1 was subjected to reversed cyclic axial and lateral loads. Changes in axial and lateral loads

CYCLE 3 CRACK PATTERN
C - HA

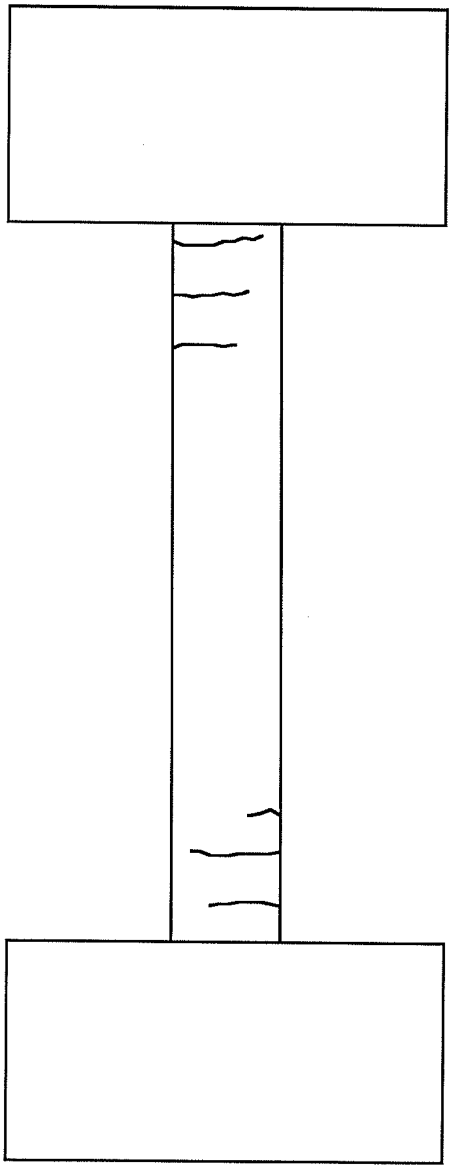


Figure 4.31

CYCLE 6 CRACK PATTERN
C - HA

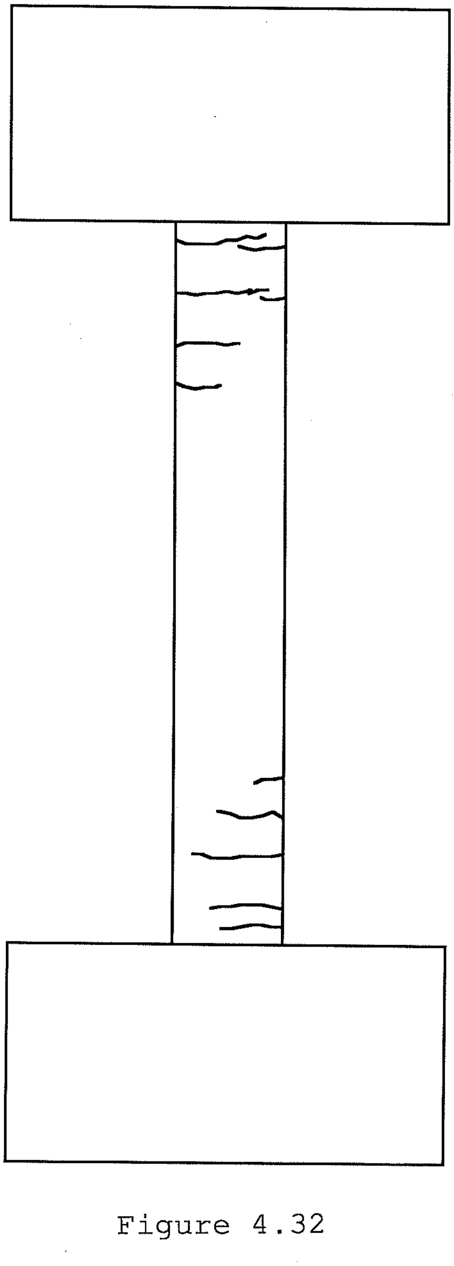


Figure 4.32

FINAL CRACK PATTERN
C - HA

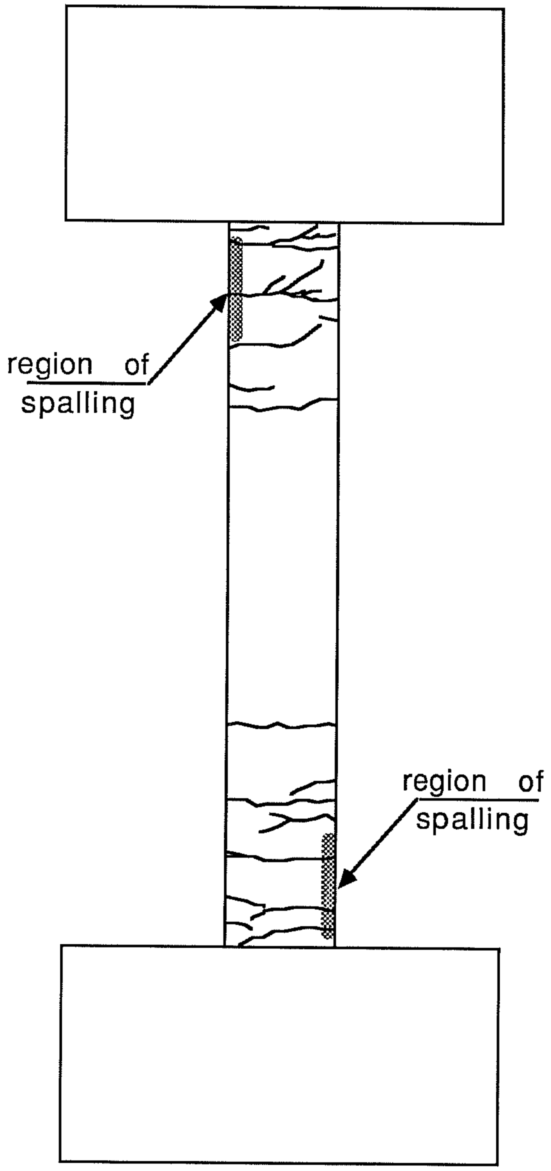


Figure 4.33

were, like specimen C - HA, held proportional, thus maintaining constant eccentricity relative to the initial axial load on the specimen. However, the ratio of change in axial load to change in lateral load was much smaller for C - LA - 1 than for C - HA (see Figs. 4.34 and 4.35). As a result, specimen C - LA - 1 experienced neither axial tension nor axial loads above the balance point as did specimen C - HA. The loading program for this column was discussed in depth in section 3.6 and shown in Fig. 3.15.

Before a full discussion of this test, it must be noted that, due to a malfunction of the electronic controls in the loading system, the initial excursion past one percent drift was lost. Test C - LA - 2, however, had almost the same loading as C - LA - 1, so a complete response to this loading was still obtained in the experimental program.

4.4.1. Load and drift histories. Lateral load, moment, axial load, and drift histories (Figs. 4.3, 4.7, 4.11, and 4.15) show the in-phase nature of the loading of specimen C - LA - 1.

4.4.2. Moment-axial load interaction. The calculated moment-axial load interaction diagram for specimen C - LA - 1 is shown with moment-axial load paths for the top and bottom column sections in Figs. 4.38 and 4.39. As these plots show, load combinations on specimen C - LA - 1 never exceeded the balance point, therefore a ductile mode of failure was expected. Also,

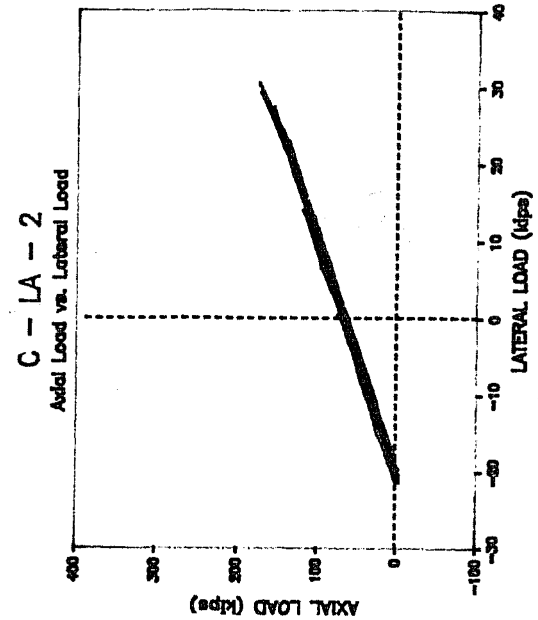


Figure 4.36

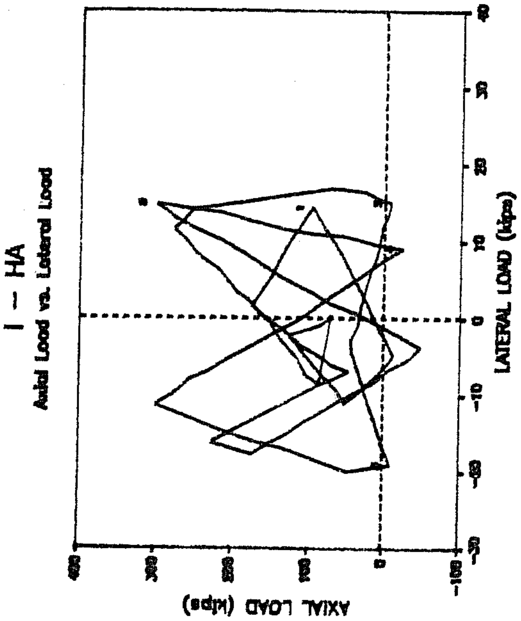


Figure 4.37

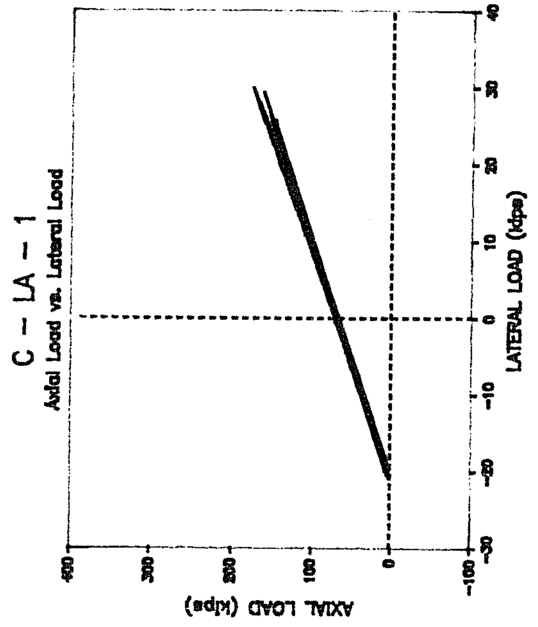


Figure 4.35

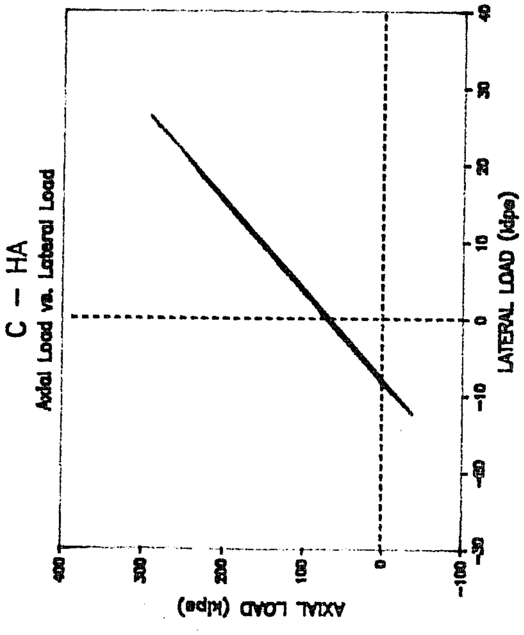


Figure 4.34

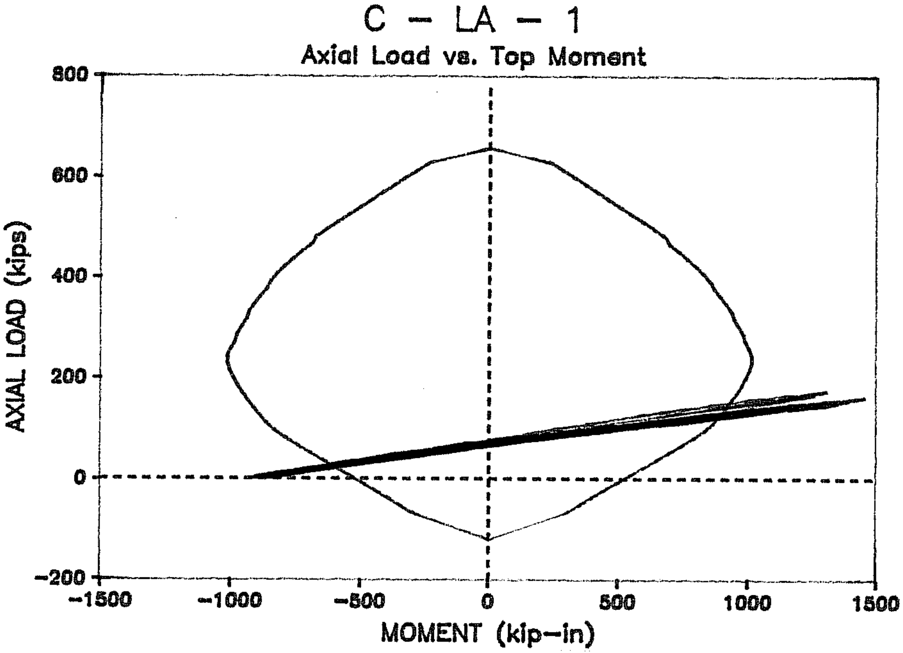


Figure 4.38

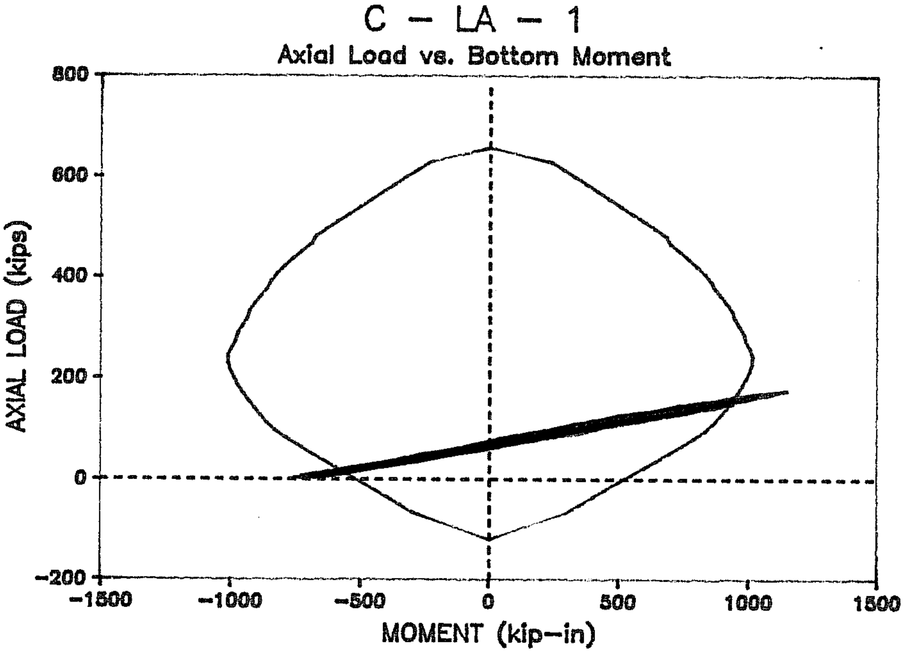


Figure 4.39

excursions beyond the calculated failure envelope in the increasing compression direction were approximately the same as in the decreasing compression direction. This is different from specimen C - HA, where more inelastic action was experienced in the increasing compression direction.

4.4.3. Lateral load-drift relationship. While the lateral load-drift hysteresis of specimen C - LA - 1 (Fig. 4.40) shows the same dissymmetry about the origin that the corresponding hysteresis of specimen C - HA did, this dissymmetry is not as pronounced. The loops of the hysteresis curve are almost the same size and shape, with the increasing compression loop slightly narrower and stiffer than the decreasing compression loop. This indicates that, unlike specimen C - HA, more energy is dissipated while axial compression is decreasing than while axial compression is increasing. This would tend to support the statement made earlier for speciemen C - HA that size of the hysteresis loops was related to how much the axial load-moment paths travelled outside the calculated failure envelope.

Loss of stiffness after initial excursions, as observed for specimen C - HA, was found in the response of specimen C - LA - 1, as can be seen in Table 4.2. The stiffness reduction is more uniform for C - LA - 1. Reductions in the increasing compression direction are approximately the same as reductions in

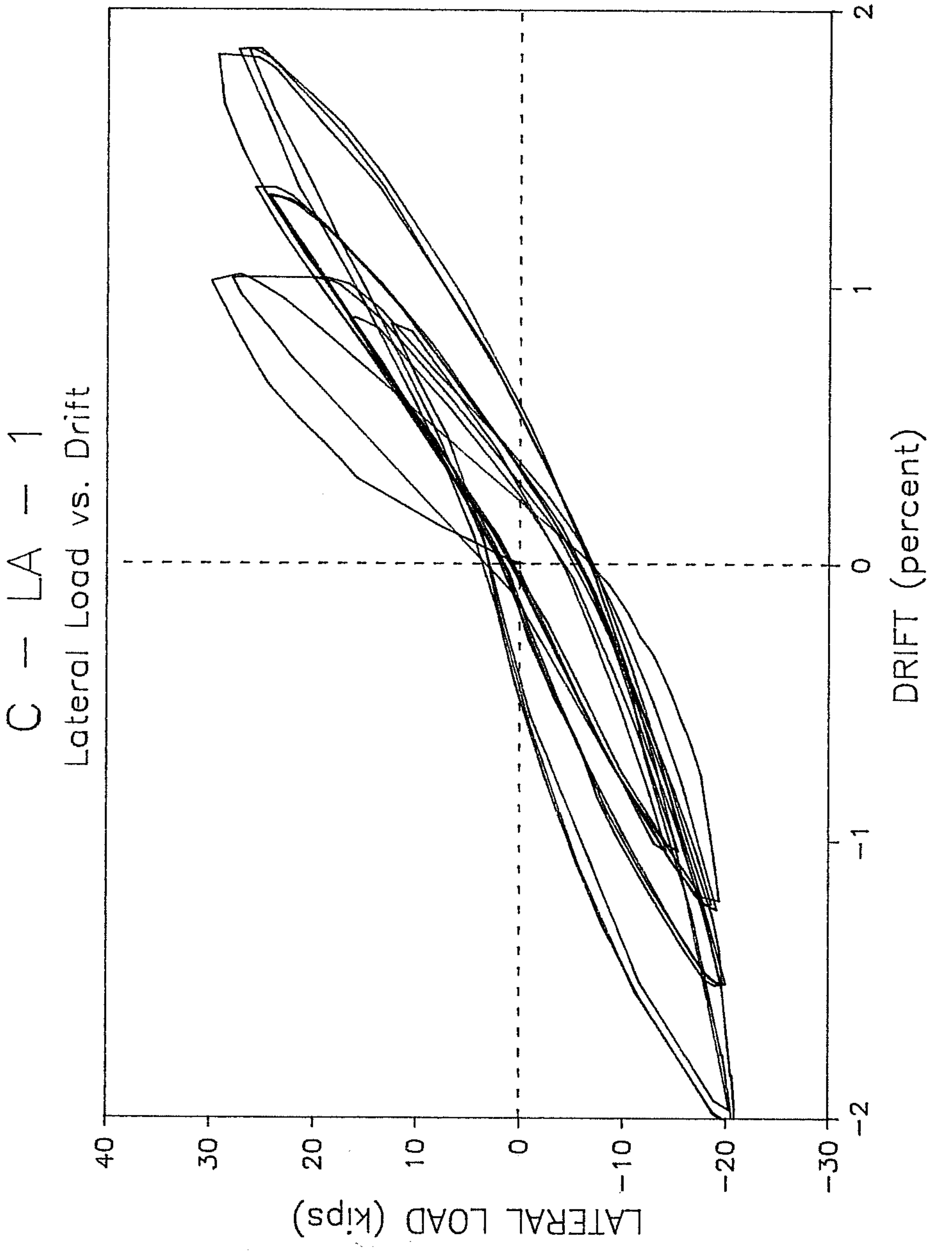

C – LA – 1
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
–10
–20
–30
DRIFT (percent)
–2
–1
0
1
2

Figure 4.40

SECANT STIFFNESSES
SPECIMEN C - LA - 1

CYCLE NO.	DIRECTION	LATERAL LOAD (kips)	DRIFT (in.)	SECANT STIFFNESS (kip/in)
1	+	29.9	0.61	**48.9**
	-	-19.5	-0.73	**26.6**
2	+	28.0	0.62	**45.1**
	-	-19.2	-0.75	**25.6**
3	+	20.1	0.62	**32.5**
	-	-18.6	-0.74	**25.3**
4	+	25.8	0.82	**31.6**
	-	-19.8	-0.91	**21.8**
5	+	24.6	0.80	**30.8**
	-	-20.1	-0.91	**22.1**
6	+	22.3	0.79	**28.2**
	-	-19.6	-0.91	**21.6**
7	+	16.6	0.53	**31.3**
	-	-15.4	-0.62	**31.7**
8	+	29.4	1.10	**26.6**
	-	-20.9	-1.20	**17.4**
9	+	27.4	1.11	**24.6**
	-	-20.6	-1.21	**17.1**
10	+	26.4	1.11	**23.7**
	-	-20.4	-1.18	**17.3**
11	+	12.6	0.52	**24.1**
	-	-14.3	-0.62	**22.9**

+ direction is increasing compression
- direction is decreasing compression

Table 4.2

the decreasing compression direction. Secant stiffnesses were initially lower for C - LA - 1 than for C - HA, but the difference between the two specimen stiffnesses decreased as higher levels of drift were experienced.

4.4.4. Strain histories. Figures 4.41, 4.42, and 4.24 show strain gage histories for top longitudinal, bottom longitudinal, and transverse bars, respectively. Longitudinal strain exceeded yield strain many times during testing. Note that strains measured on both faces reached or approached yield strain during many of the load cycles. This was contrary to the behavior measured for specimen C - HA. This information indicates the neutral axis did not remain positioned near one face of longitudinal steel as it did during the testing of specimen C - HA.

Transverse bars, however, never approached yield strain, indicating that an abundance of transverse reinforcement was provided for confinement of core concrete. Also, this strain cycled at twice the frequency of the lateral load, unlike the longitudinal strains, which were in phase with load cycles. This occurred because longitudinal bar strains are related to bending moment, which is directly related to the lateral load. Transverse bar strains, however, are related to concrete compression; when concrete is highly compressed, it causes a tensile strain in confining reinforcement. Because compression

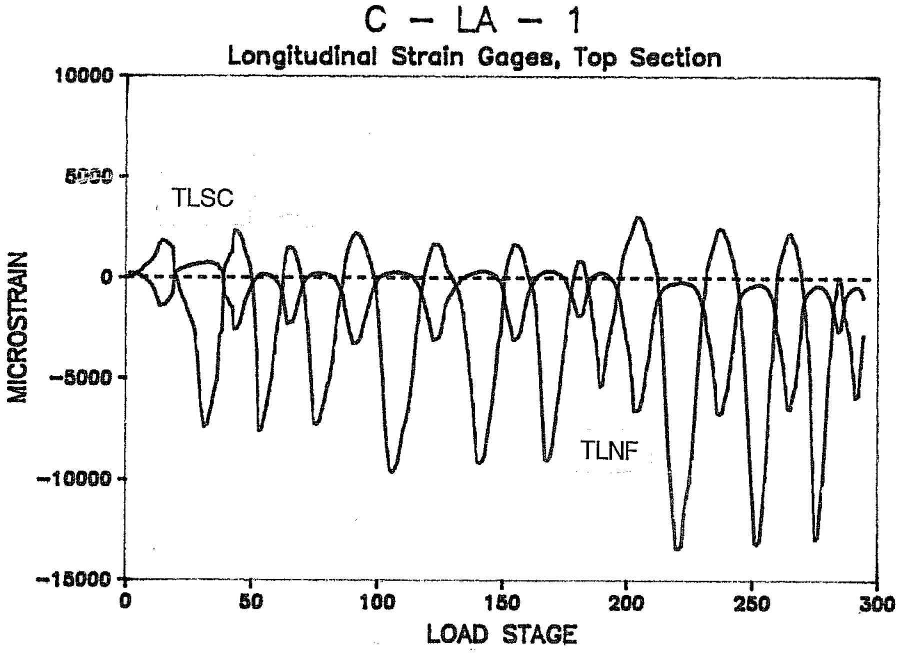


Figure 4.41

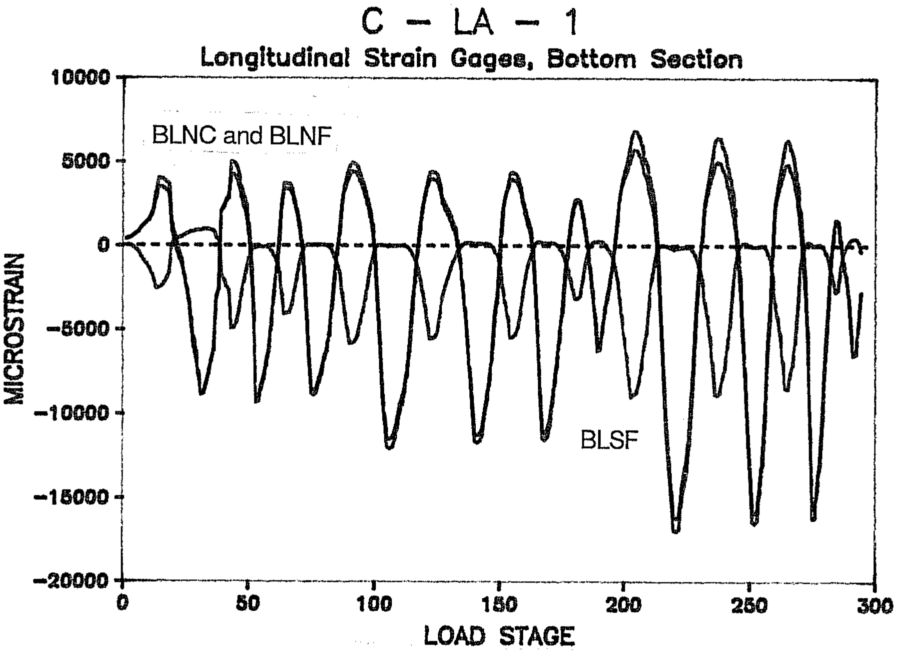


Figure 4.42

developed in both loading directions and because gage TTW was mounted on the side of the column, not a primary face, tension was developed at that point on the stirrups in both directions of loading. Therefore, the strain history of gage TTW cycled at twice the frequency of the lateral load.

4.4.5. Moment-curvature relationships. The moment-curvature relationships for specimen C - LA - 1 are shown in Figs. 4.43 and 4.44 and the moment-drift relationships are shown in Figs. 4.45 and 4.46. These relationships differ in a subtle way. After an initial excursion to a higher load-drift combination, both moment and curvature are smaller on subsequent excursions. This indicates that the hinges which form at each end become slightly larger with each subsequent excursion, so the same drift can be reached with lower curvatures at the end sections because hinge lengths are increasing.

In addition, very little permanent deformation was observed in the hinging region for specimen C - LA - 1, which contrasts with the behavior observed for specimen C - HA. This may in large part be due to the fact that specimen C - HA was controlled by concrete behavior and specimen C - LA - 1 was controlled by steel behavior.

4.4.6. Cracking. Cracks in C - LA - 1 tended to turn at a lower lateral load than cracks in C - HA. This would

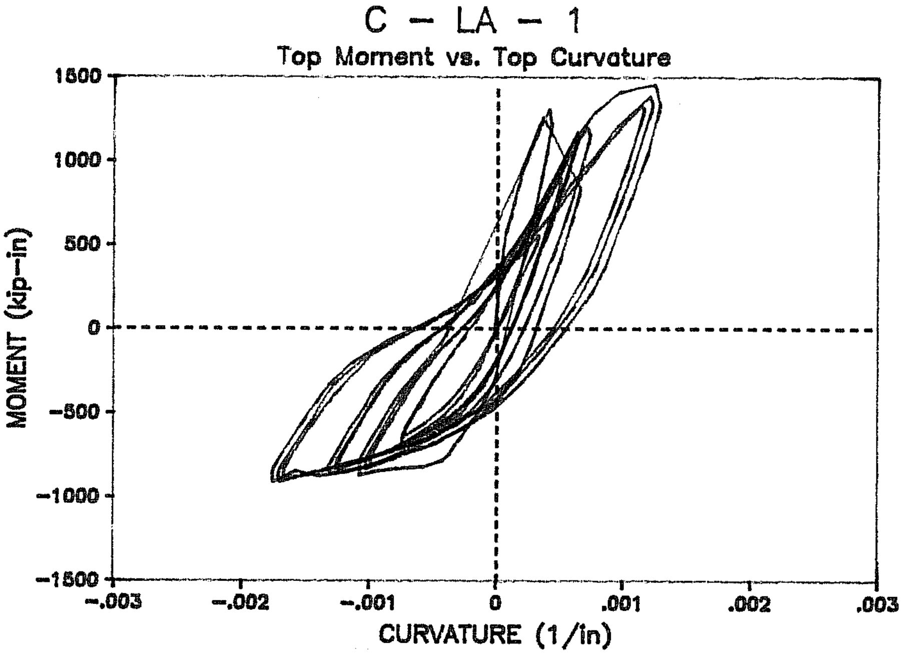


Figure 4.43

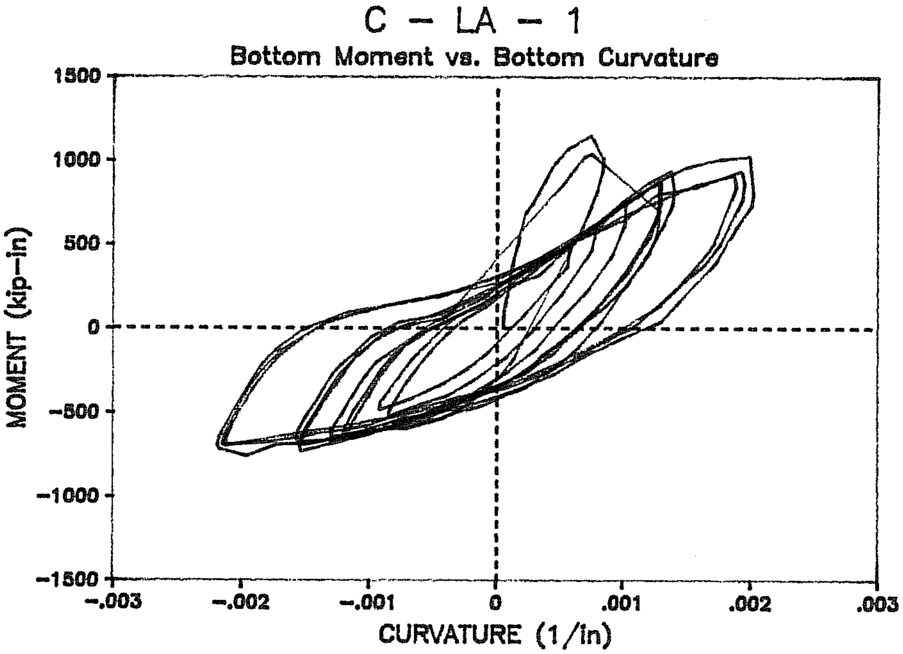


Figure 4.44

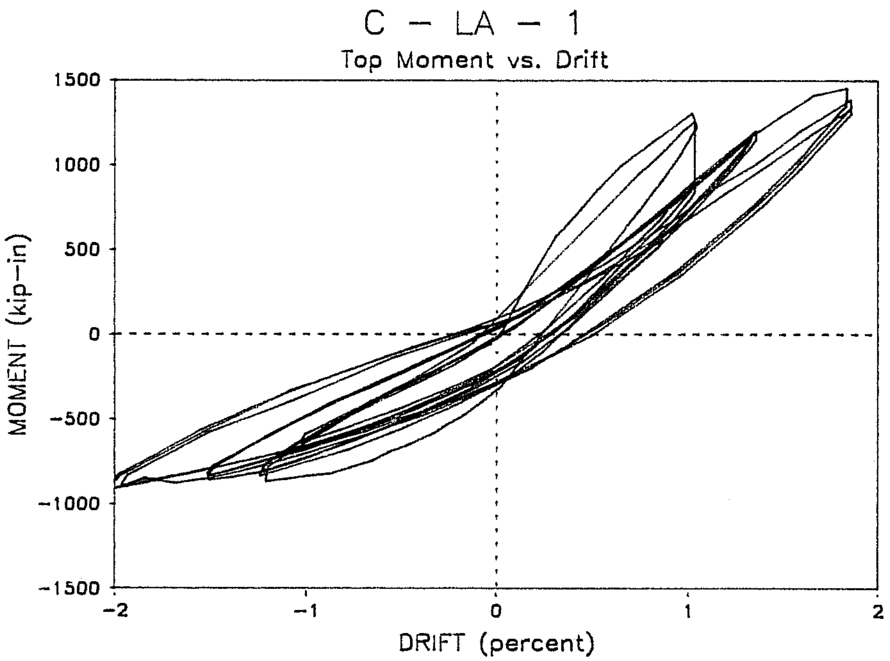


Figure 4.45

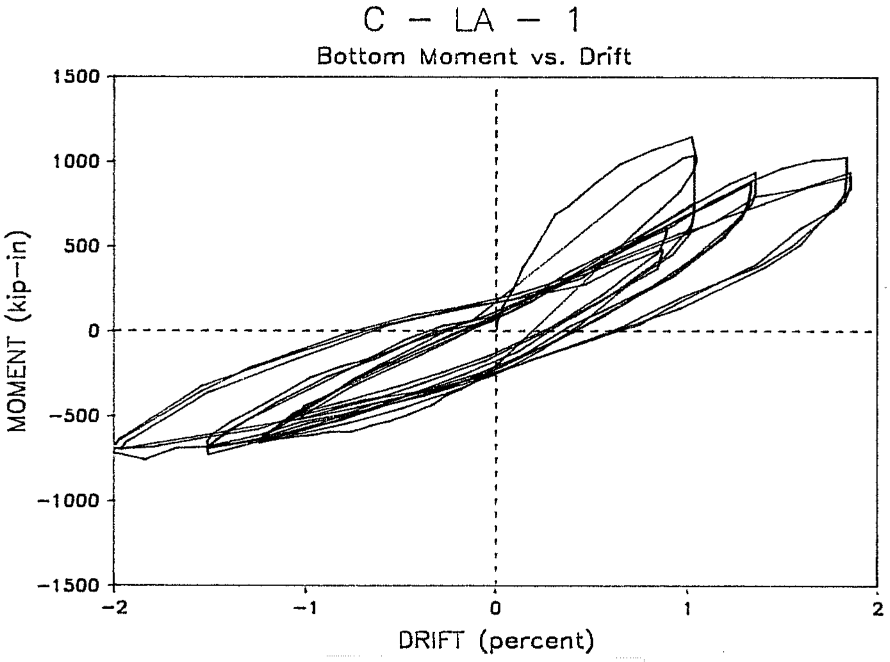


Figure 4.46

suggest that shear had a greater influence in the behavior of C - LA - 1. At low loads, however, cracks appeared to be perpendicular to the longitudinal axis of the column, suggesting that at these loads flexure dominated the column behavior. At late stages of the test, diagonal cracks, normally associated with pure shear, began to appear. Cracking at an intermediate load is shown in Fig. 4.47 and at a late stage in Fig. 4.48.

Based on spalling of cover concrete, it appeared that hinging in specimen C - LA - 1 occurred over a length of approximately ten inches. This is the same hinge length estimated for specimen C - HA.

4.5. Specimen C - LA - 2

Specimen C - LA - 2 was subjected to varying axial and lateral loads, and a constant eccentricity was maintained with respect to the initial axial load. The loading used was very similar to that used for specimen C - LA - 1 (see Figs. 4.35 and 4.36) in that the ratio of change in axial load to change in lateral load assured that moment-axial load combinations attained during the test were below the balance point.

4.5.1. Load and drift histories. Lateral load, moment, axial load, and drift histories (Figs. 4.4, 4.8, 4.12, and 4.16) show results similar to specimen C - LA - 1. Because this test was displacement controlled, lateral displacement levels remained

CYCLE 5 CRACK PATTERN
C - LA - 1

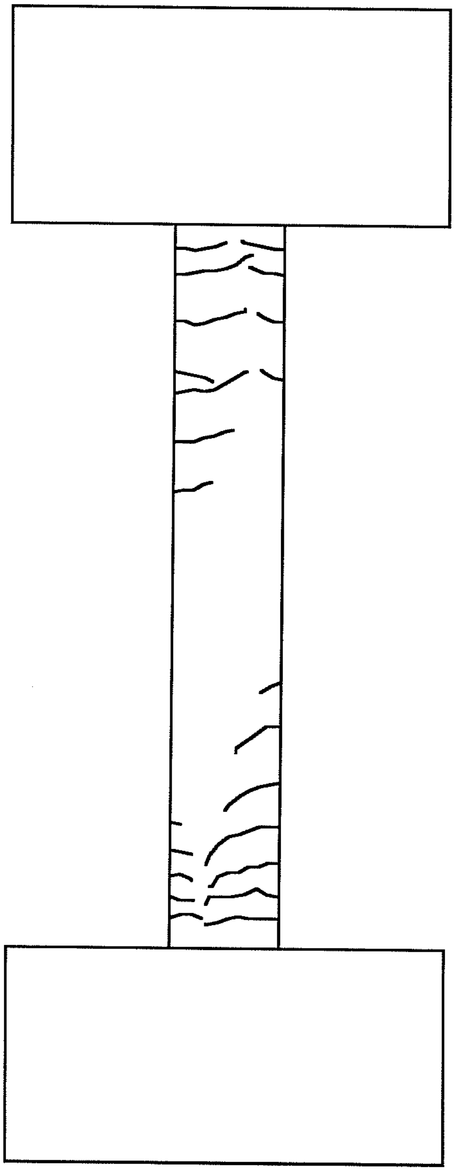


Figure 4.47

FINAL CRACK PATTERN
C - LA - 1

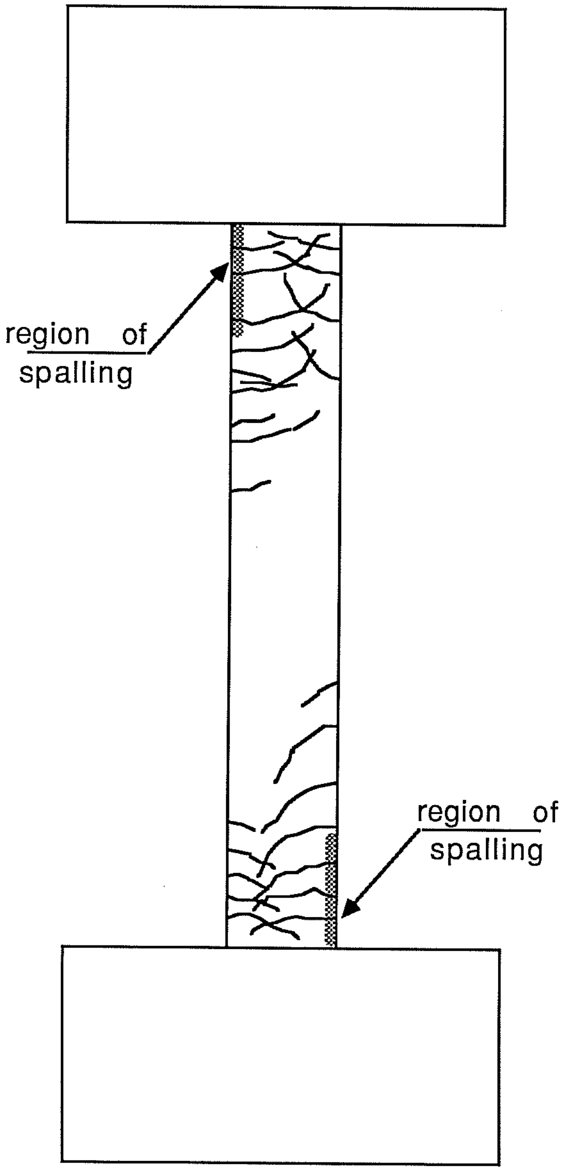


Figure 4.48

regular throughout testing. Moment, lateral load, and axial load, however, show a decrease after initial excursions to higher displacement levels. This deterioration in stiffness is similar to that found for both specimens C - HA and C - LA - 1.

4.5.2. Moment-axial load interaction. Figures 4.49 and 4.50 show calculated moment-axial load interaction relationships along with the moment-axial load path imposed on the top and bottom critical sections of the specimen. This failure envelope was exceeded in both the increasing and decreasing compression directions, indicating a substantial amount of inelastic action occurred in the specimen. Like specimen C - LA - 1, and unlike specimen C - HA, there was as much inelastic action in the decreasing compression direction as in the increasing compression direction.

4.5.3. Lateral load-drift relationship. The lateral load-drift relationship for C - LA - 2 is shown in Fig. 4.51, and secant stiffnesses at peak drift levels for each cycle are shown in Table 4.3. Like specimen C - LA - 1, the hysteresis is unsymmetrical with respect to the origin, and the loops in both directions are approximately equal in size, with the decreasing compression loop being perhaps slightly larger. After an initial excursion to a larger drift in the increasing compression direction, subsequent excursions to that same drift level occurred at a significantly lower lateral load. Similar

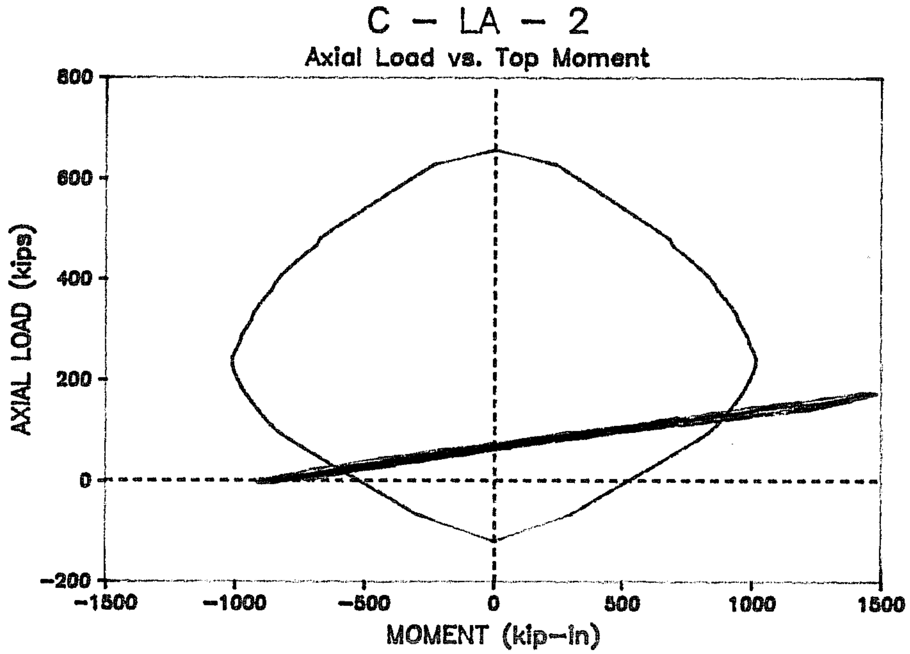


Figure 4.49

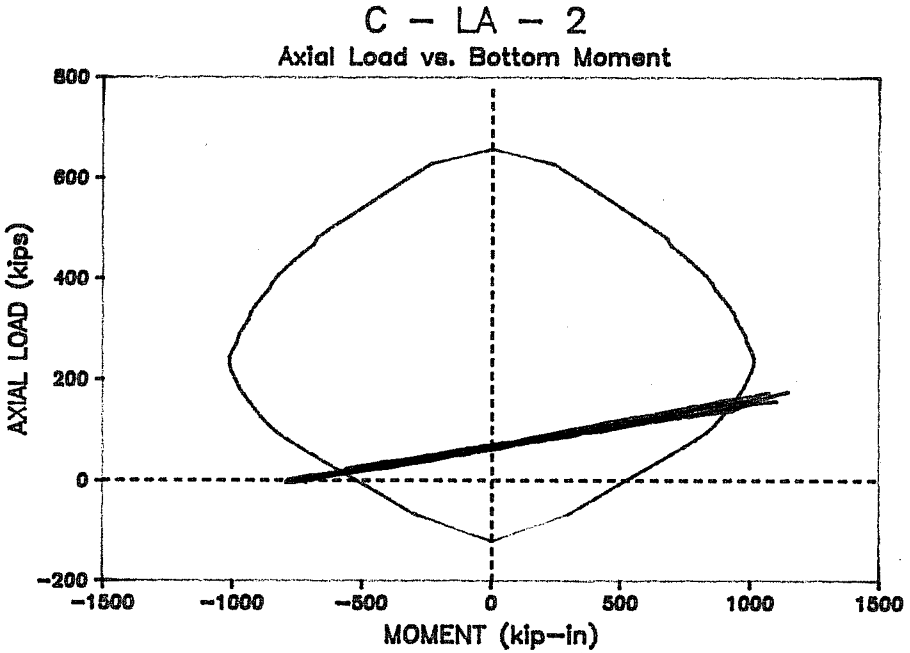


Figure 4.50

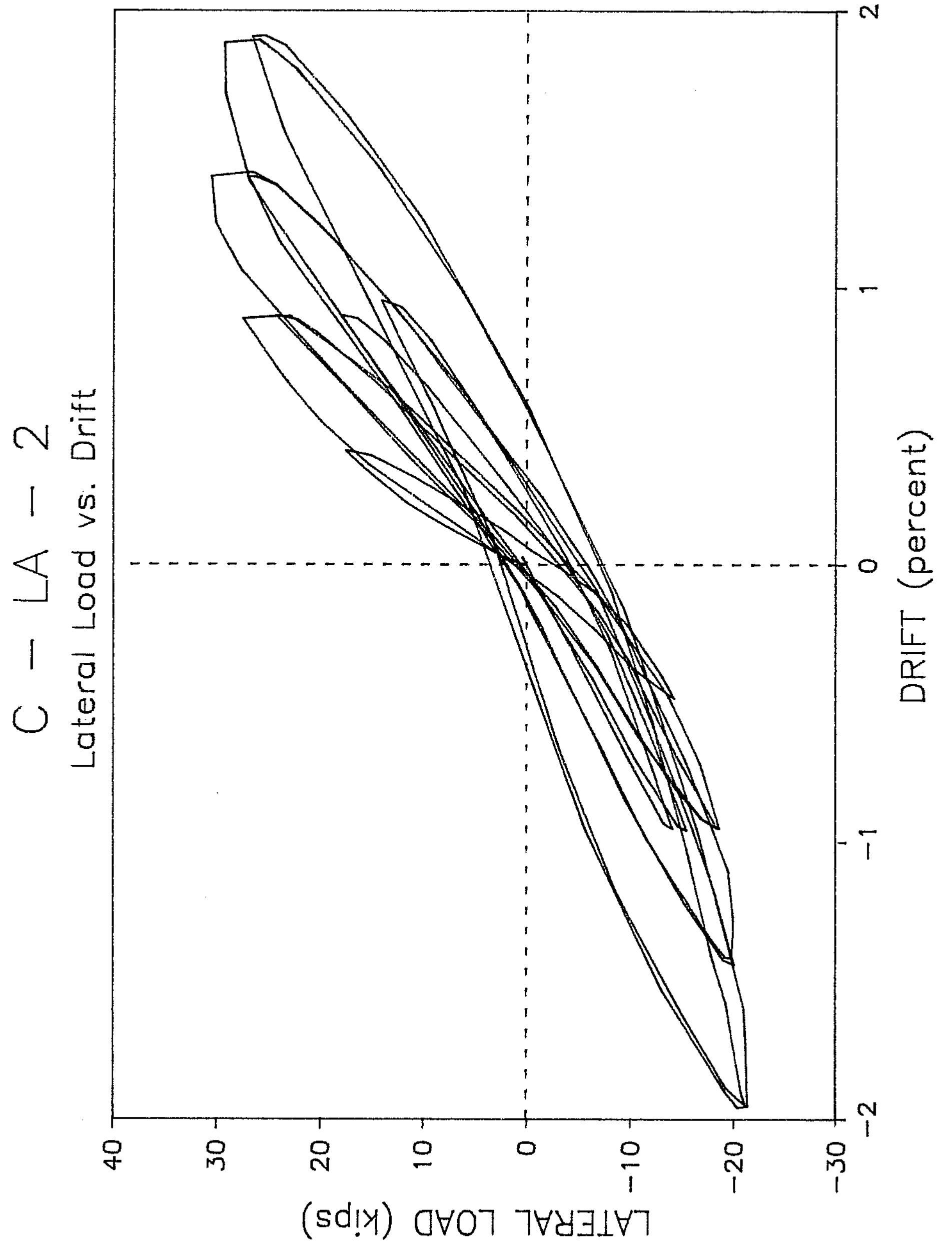

C - LA - 2
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
-10
-20
-30
DRIFT (percent)
-2
-1
0
1
2

Figure 4.51

SECANT STIFFNESSES SPECIMEN C - LA - 2

CYCLE NO.	DIRECTION	LATERAL LOAD (kips)	DRIFT (in.)	SECANT STIFFNESS (kip/in)
1	+	17.6	0.25	**71.1**
	-	-14.3	-0.29	**49.1**
2	+	27.6	0.40	**51.8**
	-	-18.7	-0.57	**32.6**
3	+	24.4	0.54	**45.1**
	-	-18.4	-0.57	**32.2**
4	+	30.6	0.84	**36.3**
	-	-20.0	-0.76	**26.2**
5	+	27.1	0.84	**32.2**
	-	-20.1	-0.87	**23.2**
6	+	18.0	0.54	**33.3**
	-	-15.5	-0.58	**26.8**
7	+	29.4	1.13	**26.0**
	-	-21.4	-1.17	**18.3**
8	+	26.8	1.14	**23.4**
	-	-21.2	-1.17	**18.1**
9	+	14.1	0.57	**24.6**
	-	-14.1	-0.57	**24.6**

+ direction is increasing compression

- direction is decreasing compression

Table 4.3

excursions with decreasing axial compression reached loads approximately the same as experienced previously. This appears to occur because behavior in the increasing compression direction is dominated by concrete behavior while column behavior in the decreasing compression direction is dominated by steel behavior. As cover concrete is lost, lateral load capacity is reduced. However, because cover is lost on only one face of the column, lateral load capacity in the other direction is not affected. The degree of load reduction is related to how much concrete is lost on the initial excursion. Therefore, the rate of reduction, after an initial excursion, decreases for excursions to very large drifts because most cover has already been lost and remaining concrete is well-confined.

4.5.4. Strain histories. Longitudinal strain histories for top and bottom critical sections are shown in Figs. 4.52 and 4.53. As can be seen, longitudinal bar strain exceeded yield strain on several occasions. Unlike the strain response observed for specimen C - HA, strains measured on opposite faces showed significant variations during the test. At points of peak response in the decreasing compression direction, strains measured on one face were nearly zero. This resulted from movement of the neutral axis to approximately the locations of the layer of longitudinal steel.

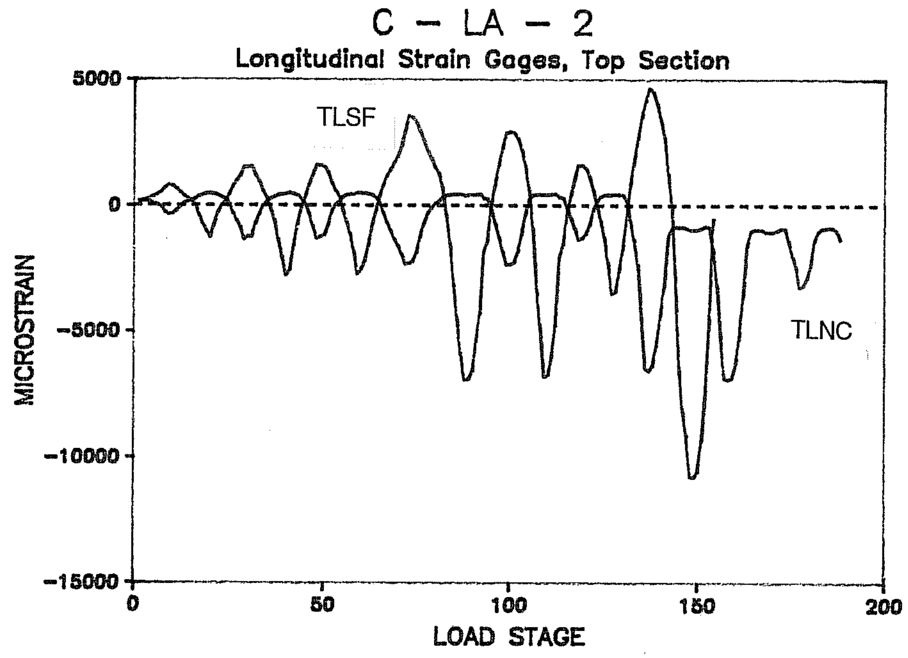


Figure 4.52

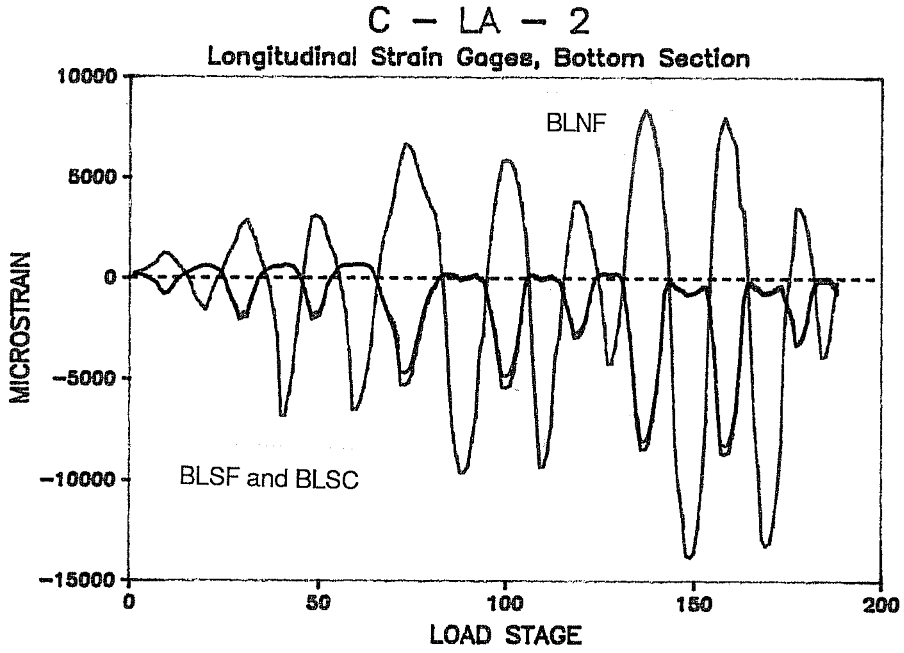


Figure 4.53

The strain history typical of the transverse reinforcement is shown in Fig. 4.25. All transverse steel gaged in this specimen remained well within the elastic range. Again, as in C - HA and C - LA - 1, it appears the abundance of stirrups in the hinge region kept the strain in each stirrup below yield strain.

4.5.5. Moment-curvature relationships. Moment-curvature relationships for critical sections at the top and bottom of specimen C - LA - 2 (Figs. 4.54 and 4.55) and corresponding moment-drift relationships (Figs. 4.56 and 4.57) exhibit the same dissymmetry due to variations in axial load seen in previous specimens. This dissymmetry decreases as excursions to higher curvatures occur. As was the case for the lateral load-drift relationship, following an initial excursion to a higher drift level, subsequent excursions to the same drift required less moment at the end sections of the column. However, as was also the case for specimen C - LA - 1, the subsequent excursions returned to a lower curvature as well as a lower moment, suggesting an increase in the hinge length at the ends of the column.

4.5.6. Cracking. Specimen C - LA - 2 exhibited nearly identical cracking behavior as specimen C - LA - 1. Early cracking was perpendicular to the longitudinal axis of the column, and cracks tended to turn at an earlier stage than

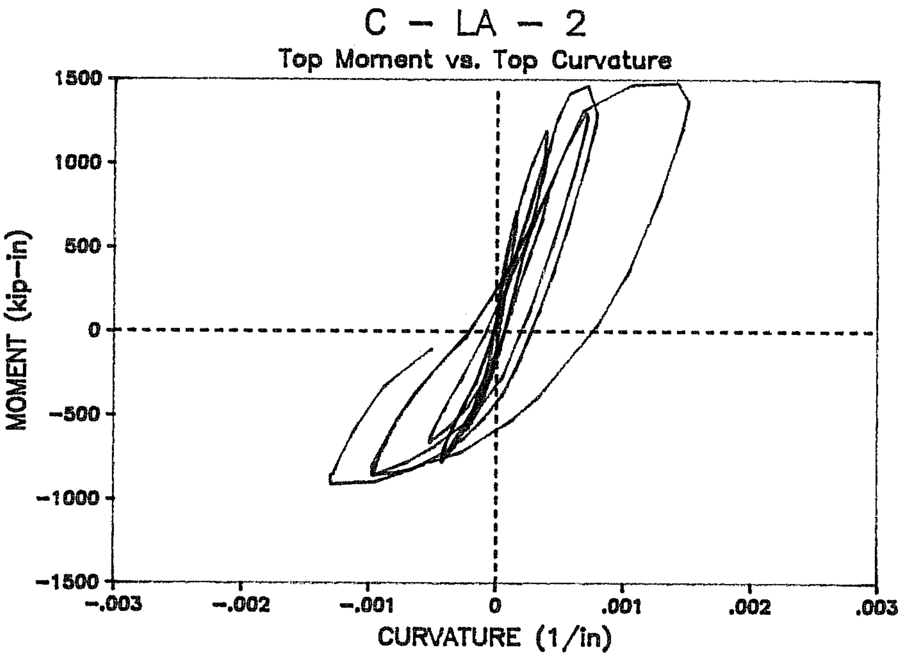


Figure 4.54

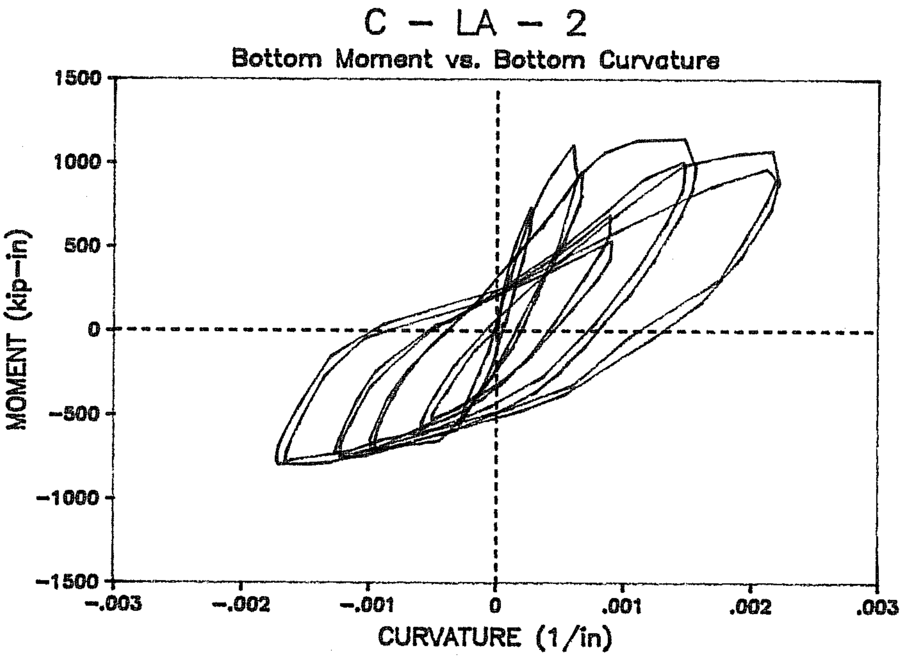


Figure 4.55

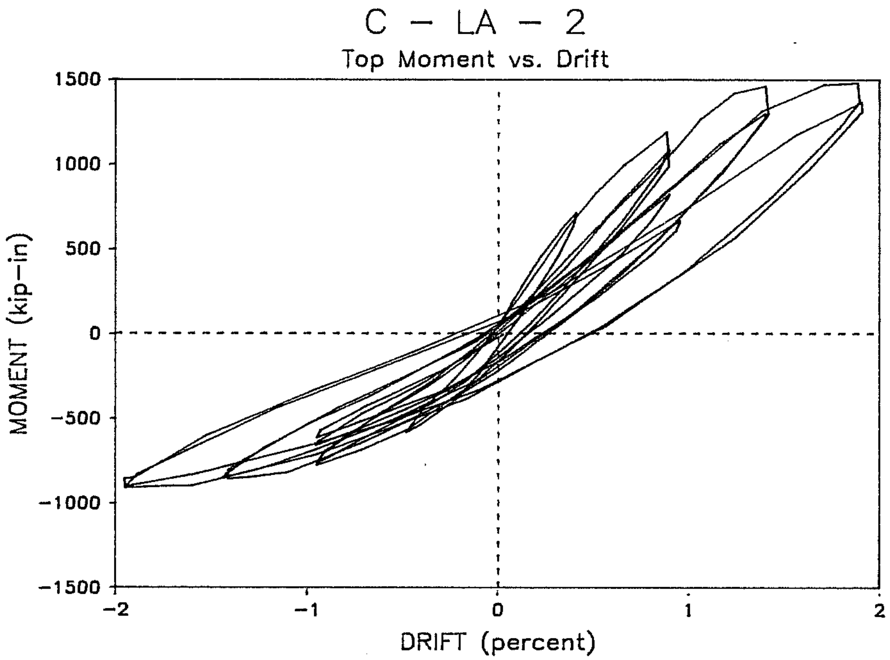


Figure 4.56

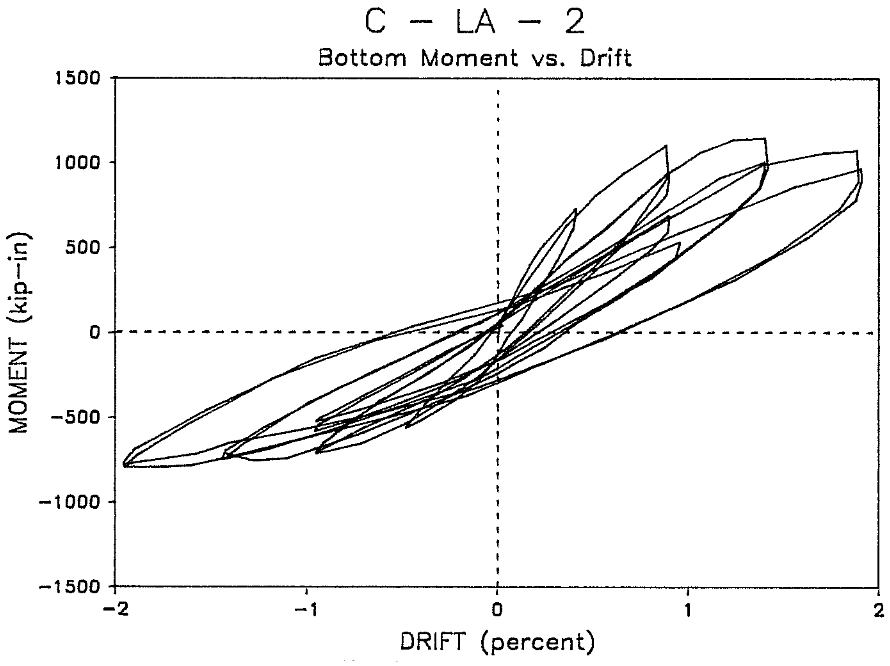


Figure 4.57

specimen C - HA. Figure 4.58 is a drawing of cracking at an intermediate load stage and Fig. 4.59 is a drawing of cracking at a late load stage. Like specimens C - HA and C - LA - 1, spalling and cracking indicated a hinge length of approximately ten inches.

4.6 Specimen I - HA

Specimen I - HA was subjected to two very different programs of varying axial and lateral loads. The first loading, unlike other loadings in this experimental program, applied axial and lateral loads which did not change proportionately. Eccentricity relative to the initial axial load changed during testing. While both axial and lateral loads were reversed cyclic loads, they were not in phase and consequently the response was not as well-behaved as for other specimens.

The second loading, applied after the first was completed, maintained constant eccentricity with respect to the initial axial load. The ratio of change in axial load to change in lateral load was relatively low, making it much like the loading applied to specimen C - LA - 1. For purposes of discussion, the specimen I - HA will be treated as two specimens, I - HA and C - LA - S, which were subjected to the first and second loadings, respectively.

CYCLE 4 CRACK PATTERN
C - LA - 2

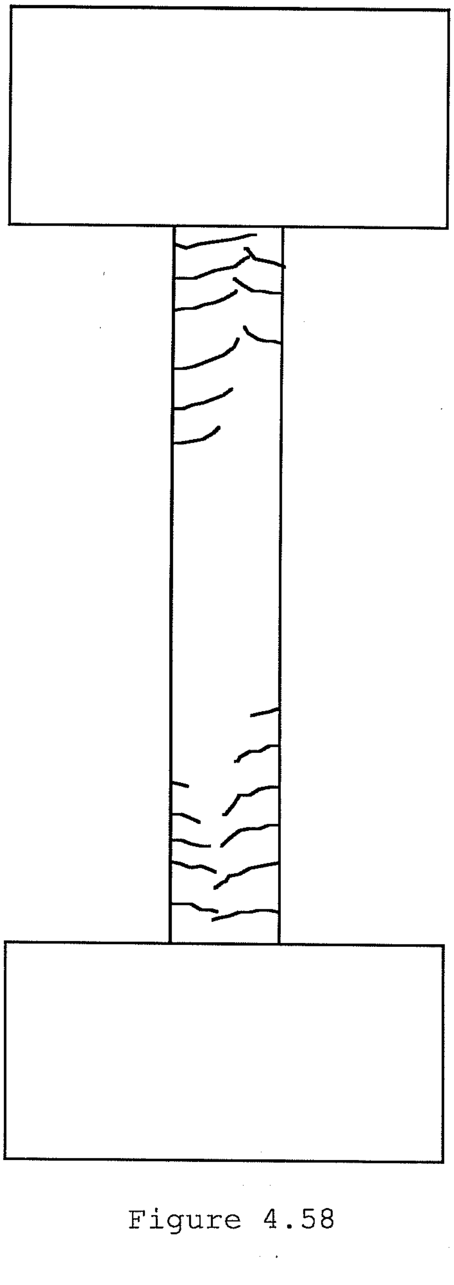


Figure 4.58

FINAL CRACK PATTERN C - LA - 2

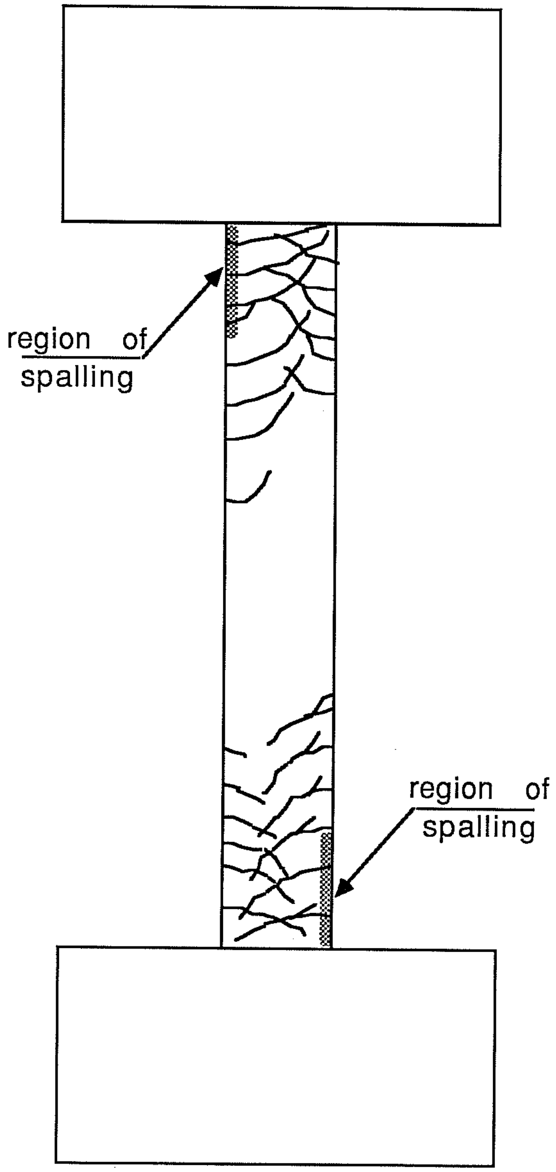


Figure 4.59

Data from specimen I - HA is included with the constant eccentricity tests discussed above, as it is one of the main tests of the experimental program. Specimen C - LA - S, however, being damaged before testing, is presented seperately from other tests.

4.6.1. *Results of first load program.* This specimen was loaded with uncoupled lateral and axial loads as described in section 3.6 and shown in Fig. 3.16. As mentioned above, eccentricity relative to the initial axial load varied during the test, resulting in the axial load vs lateral load diagram shown in Fig. 4.37. Five reference points are provided on each plot at important load stages.

4.6.1.1. Load and drift histories. The load and drift histories (Figs. 4.5, 4.9, 4.13, and 4.17) demonstrate that lateral and axial loads were applied out of phase. It is also important to note how quickly axial loads changed relative to lateral loads. This loading was intended to represent forces experienced by columns which were part of a complex lateral load resisting system composed of a staggered structural wall and a moment-resisting frame orthogonal to the strong axis of the wall system. This loading was discussed in depth in section 3.6.

4.6.1.2. Moment-axial load interaction. Figures 4.60 and 4.61 show the moment-axial load history for the top and bottom of the column superimposed on the calculated moment-axial

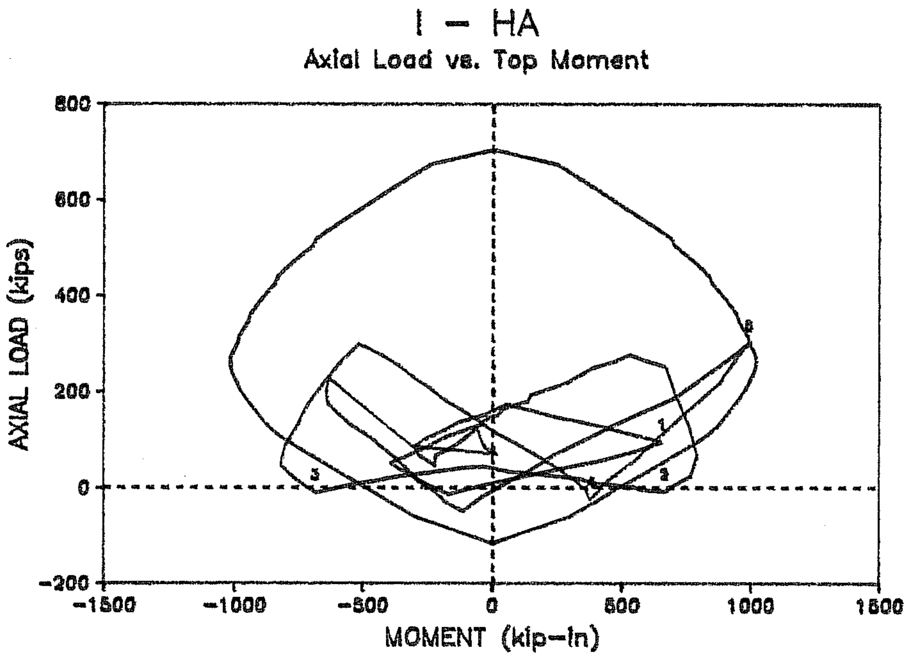


Figure 4.60

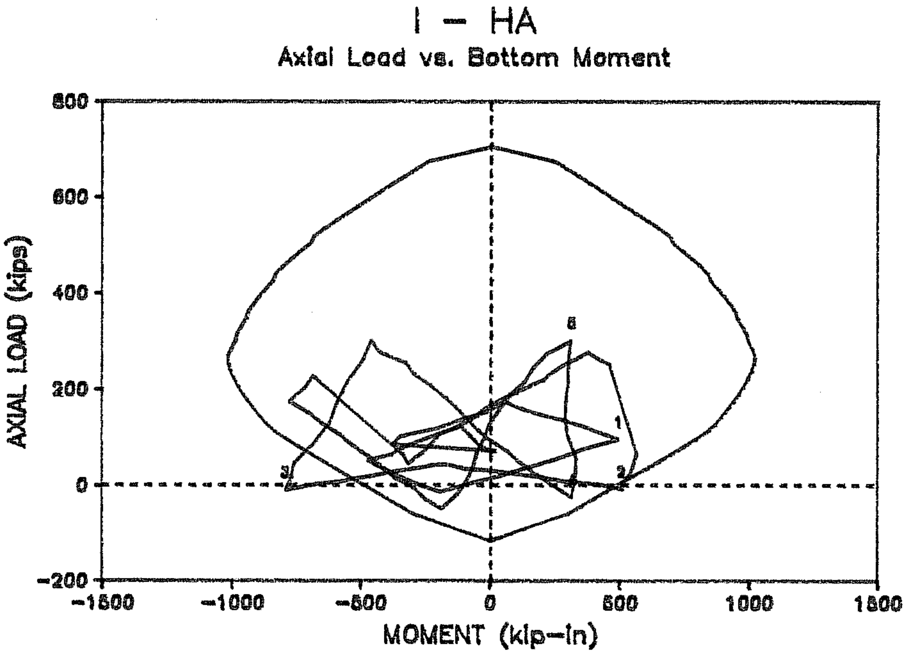


Figure 4.61

load interaction diagram. These figures indicate the wide range of load combinations to which I - HA was exposed and illustrate how severe those load combinations were relative to calculated failure conditions. On three occasions the top critical section reached moment-axial load combinations on or beyond the calculated failure envelope (reference points 2, 3, and 5). One of those combinations (reference point 5) was above the balance point. The bottom critical section reached the envelope twice (reference points 2 and 3).

4.6.1.3. Lateral load-drift relationship. The lateral load-drift hysteresis relationship for I - HA is shown in Fig. 4.62, and lateral secant stiffness at the peak drift of each lateral load cycle is shown in Table 4.4. Before discussing this response, it is important to emphasize that this is only a two dimensional graph (lateral load vs drift) of a function with three independent variables (lateral load, axial load, and drift). Consequently, this graph only shows a projection of the entire response on a plane.

In looking at Fig. 4.62, one area that stands out is the loop which is formed when the path crosses over itself. At the top of this loop is a lateral load-drift path in which the load increases while the drift decreases (the part of the response between reference points 4 and 5). This apparent

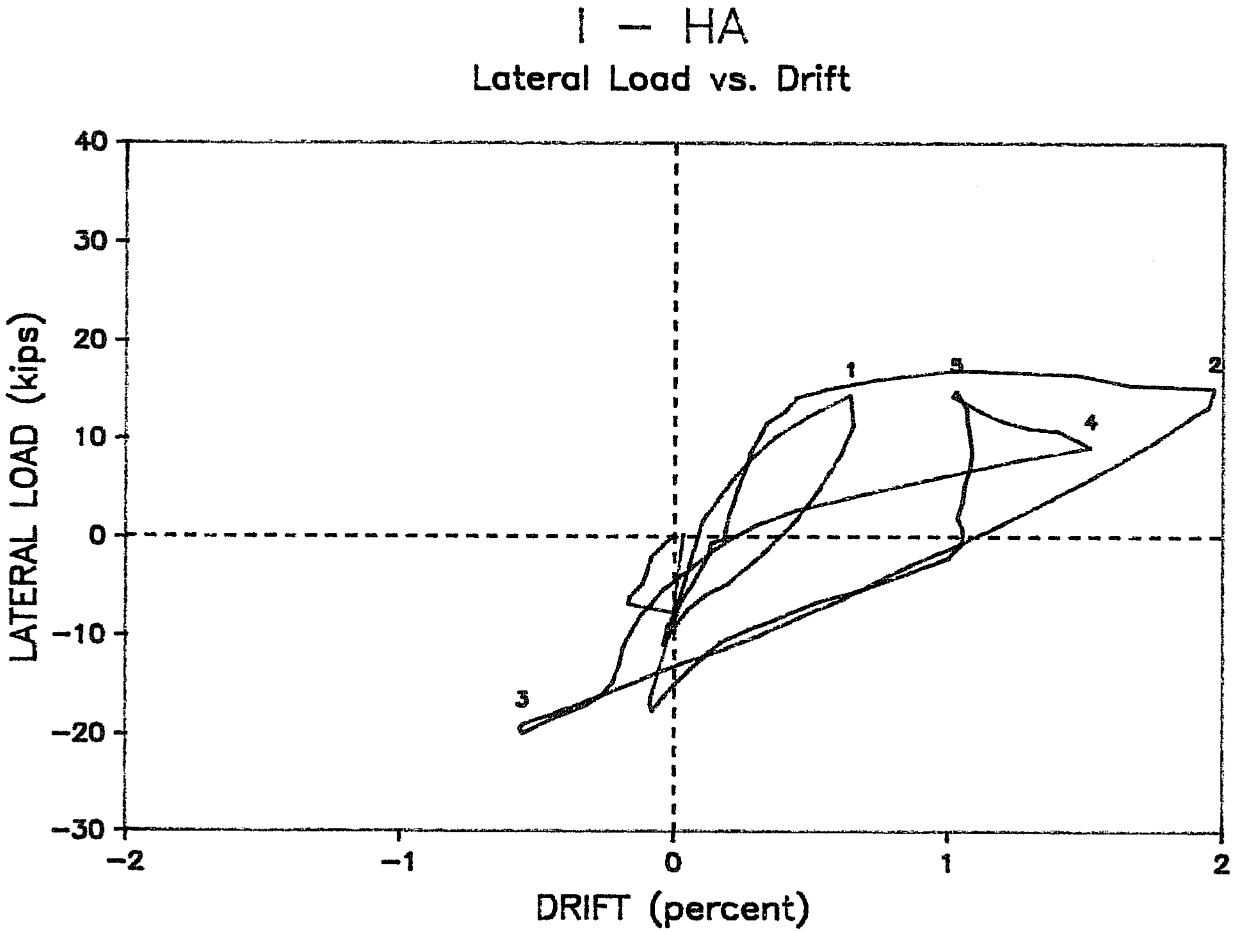

I — HA
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
DRIFT (percent)
−2
−1
0
1
2
1
5
2
4
3

Figure 4.62

SECANT STIFFNESSES
SPECIMEN I - HA

CYCLE NO.	DIRECTION	AXIAL LOAD (kips)	LATERAL LOAD (kips)	DRIFT (in.)	SECANT STIFFNESS (kip/in)
1	+	N.A.	N.A.	N.A.	N.A.
	-	85.0	-8.6	-0.18	**48.0**
2	+	95.5	14.4	0.23	**63.0**
	-	52.1	-11.1	-0.21	**52.3**
3	+	68.9	17.0	0.46	**37.1**
	-	46.7	-20.0	-0.53	**37.5**
4	+	300.6	14.9	0.46	**32.2**
	-	174.1	-17.8	-0.24	**74.7**

+ direction is increasing compression
- direction is decreasing compression

Table 4.4

anomaly is explained by the change in axial load which occurs along this path, a change which is not shown in this plot. Axial load changes from tension at reference point 4 to very high compression (about half of ultimate) at reference point 5. Under a large compression force, the lateral stiffness of the column is much higher than it is when the column experiences a net tension force. As a result, drift decreases while lateral load increases because lateral stiffness increases at a much higher rate than does the lateral load. The increase in lateral stiffness is due to the rapidly increasing axial load. More load is therefore required to maintain the column at a lower level of drift. This clearly illustrates how sensitive lateral stiffness is to changes in axial load.

It is difficult to say much more about the hysteresis relationships for I - HA. Complex loadings with many independent components often defy understanding when they are graphed on only two axes. However, if two-dimensional observations about energy dissipation for a three-dimensional phenomenon are valid, it appears more energy was dissipated in the third lateral load cycle (the large loop bounded by reference points 2 and 3) than in all other cycles combined. Much of the energy was dissipated in the positive direction, while axial load was relatively low. When axial load increased, width of the hysteresis decreased dramatically, thus decreasing energy dissipation.

4.6.1.4. Strain histories. Figures 4.63, 4.64, and 4.26 show top longitudinal, bottom longitudinal, and transverse strain gage histories, respectively. Once again, longitudinal bars strained beyond yield, while transverse bars remain below yield during the entire test. Also, notice that neither longitudinal bars nor transverse bars are strained at a frequency that corresponds with one of the loadings. Rather, they cycle in much the same way as the drift history, which is the result of a combination of axial and lateral load.

4.6.1.5. Moment-curvature relationships. In the moment-curvature relationships (Figs. 4.65 and 4.66), some of the same characteristics appear as were observed in moment-drift hysteresis relationships (Figs. 4.67 and 4.68). The moment-curvature path exhibits fewer instances of crossing over itself than the moment-drift hystereses. In fact, only the moment-curvature hysteresis for the top of the column shows any "reverse loops." Another difference which can be noted is that narrowing of the hysteresis loops due to increasing lateral load is not as dramatic for the moment-curvature relationships as it was for the moment-drift relationships. Both of these differences suggest that changes in axial load more profoundly affect global column stiffness than stiffness of an individual section.

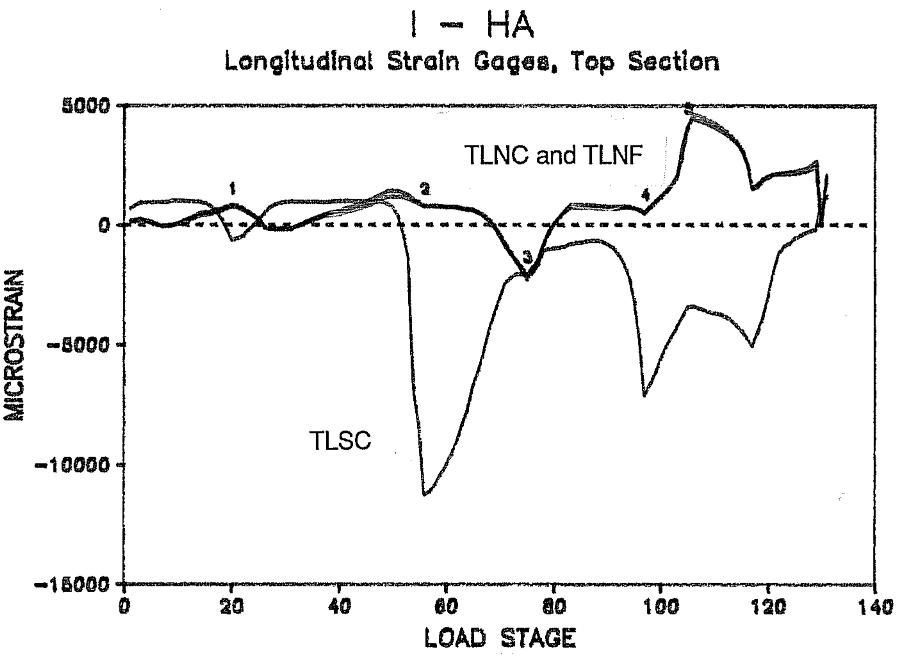


Figure 4.63

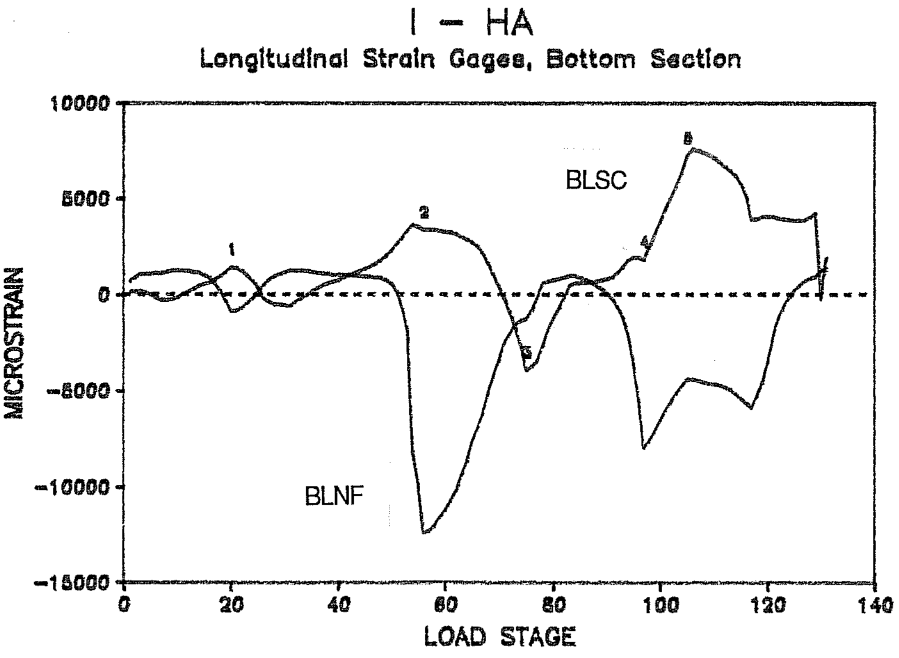


Figure 4.64

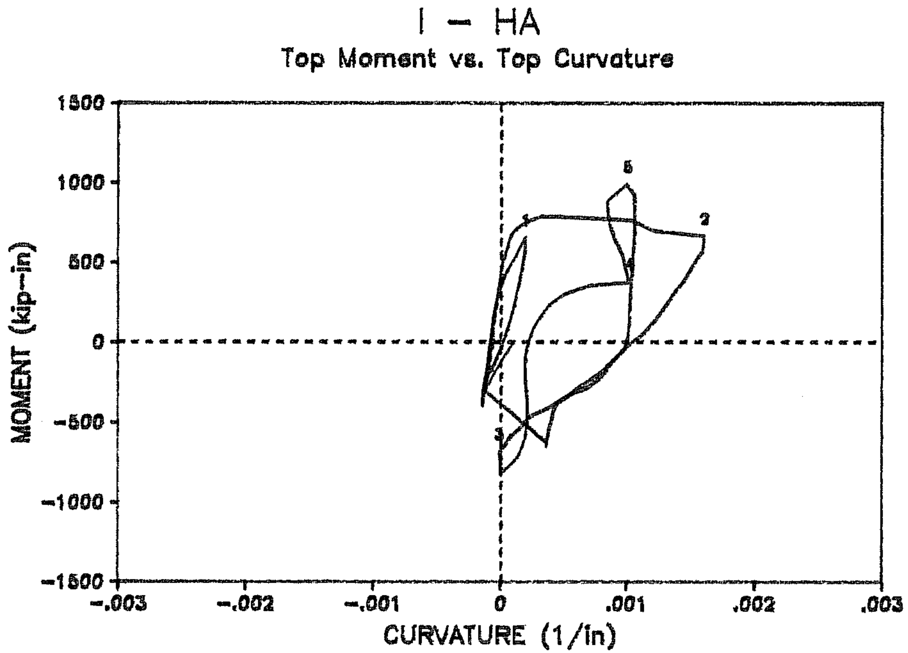


Figure 4.65

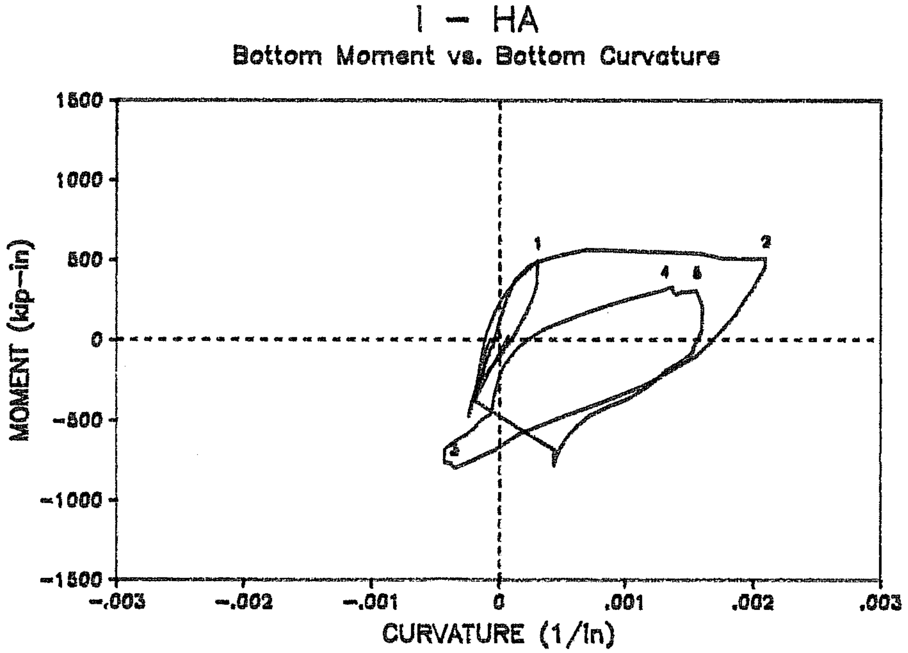


Figure 4.66

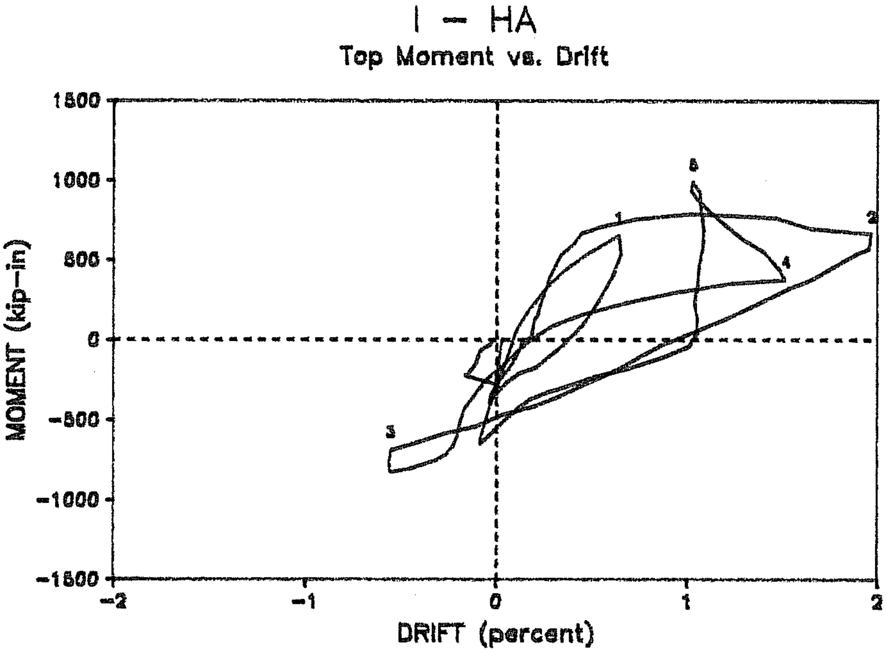


Figure 4.67

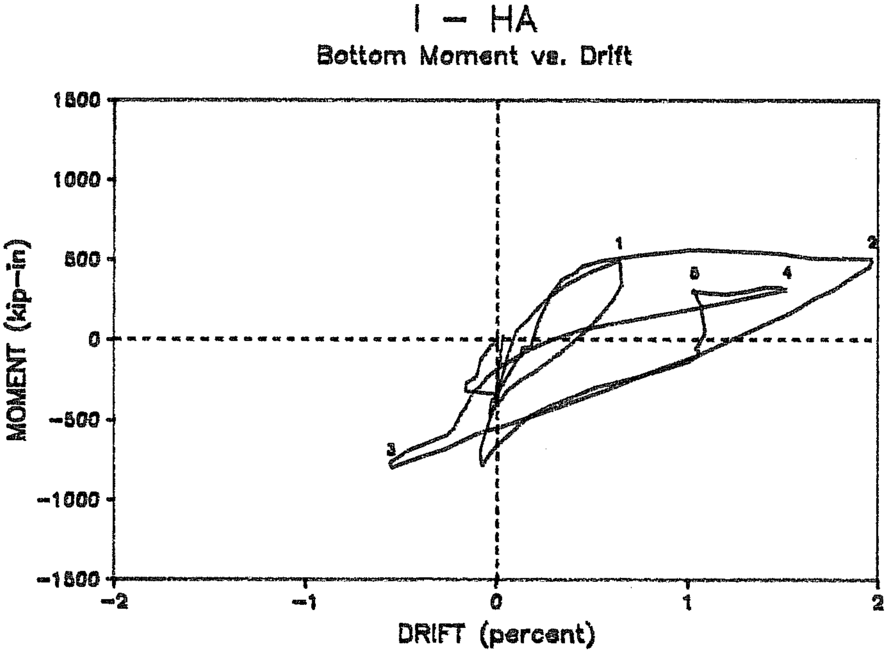


Figure 4.68

4.5.1.6. Cracking. Cracks in I - HA tended to be inclined at both early and late load stages, as shown in Figs. 4.69 and 4.70. This indicated a strong influence of shear in the behavior of the column. This influence is due to the fact that axial load significantly affects shear strength of the concrete, and unlike earlier tests, high levels of lateral load were experienced by specimen I - HA when axial load (and consequently concrete shear strength) was low.

Determination of hinge length of specimen I - HA was much more subjective than it was for the other three specimens because spalling of cover concrete did not occur. It did appear, however, that most of the major cracks during the test occurred within ten inches of the endblock, so the hinge length of ten inches used for other columns appears reasonable.

4.6.2. Loading C - LA - S. After specimen I - HA was subjected to the loading described in section 4.6.1, a loading was applied which was similar to that applied to specimen C - LA - 1 (see Figs. 4.35 and 4.71). Changes in lateral load and axial load were held proportional, and the eccentricity was kept low so that moment-axial load combinations above the balance point were not achieved.

4.6.2.1. Load and drift histories. Lateral load, axial load, moment, and drift histories (Figs. 4.73 through 4.76) show

CYCLE 2 CRACK PATTERN
I - HA

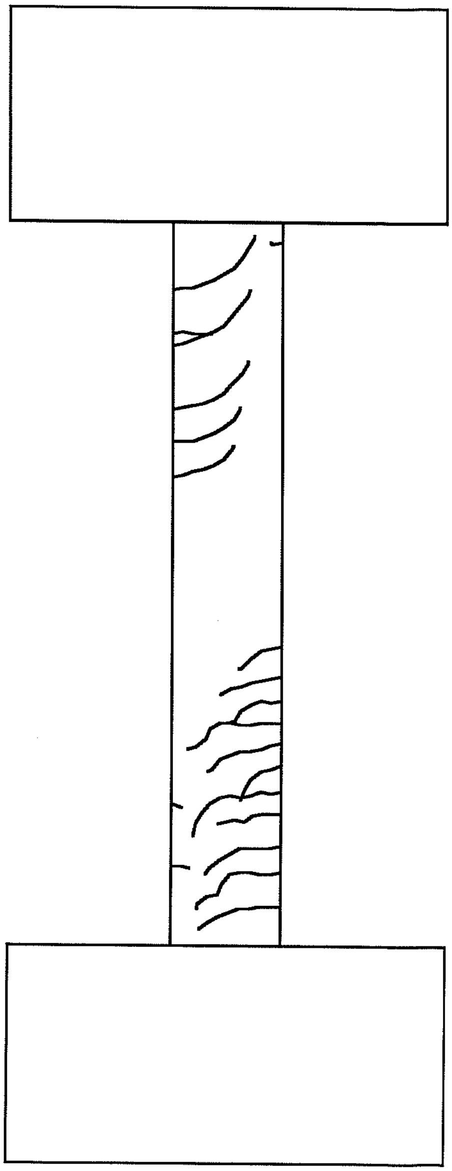


Figure 4.69

FINAL CRACK PATTERN
I - HA

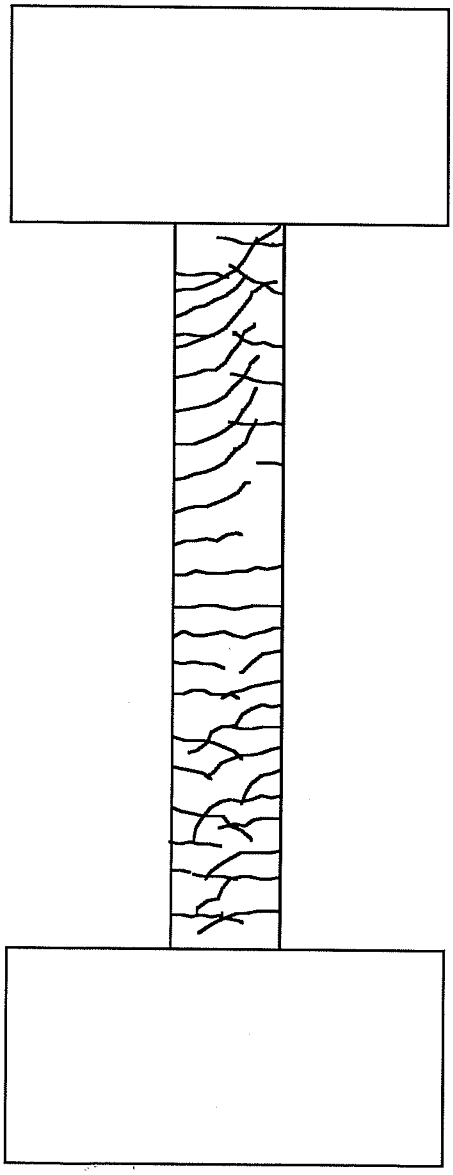


Figure 4.70

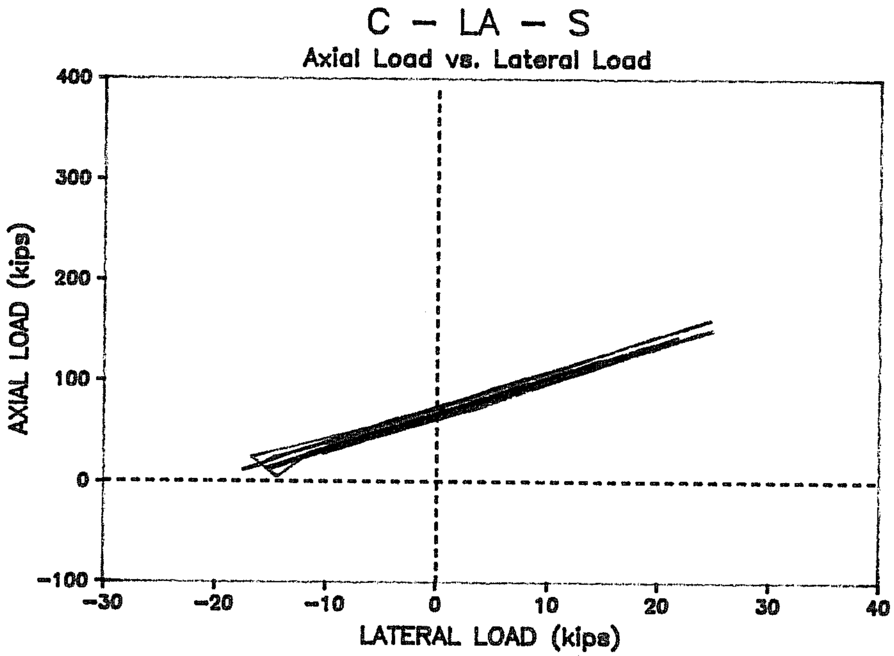


Figure 4.71

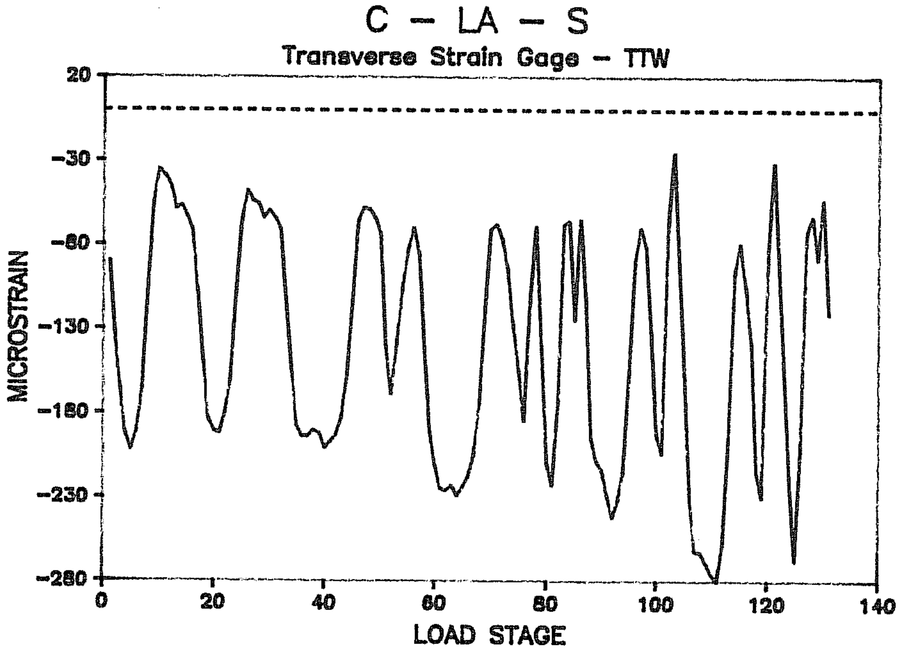


Figure 4.72

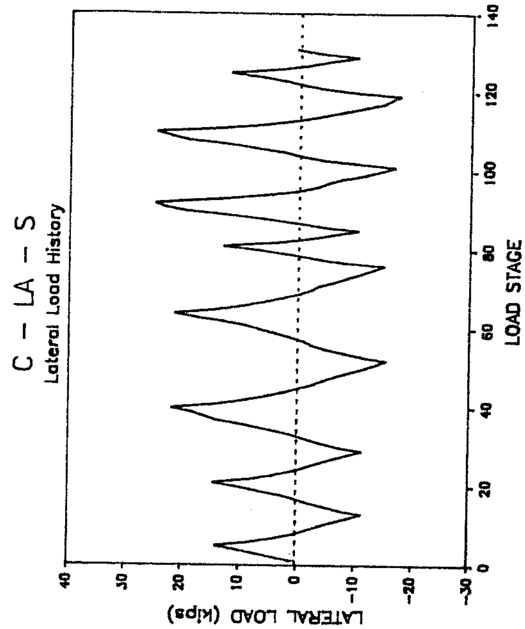


Figure 4.73

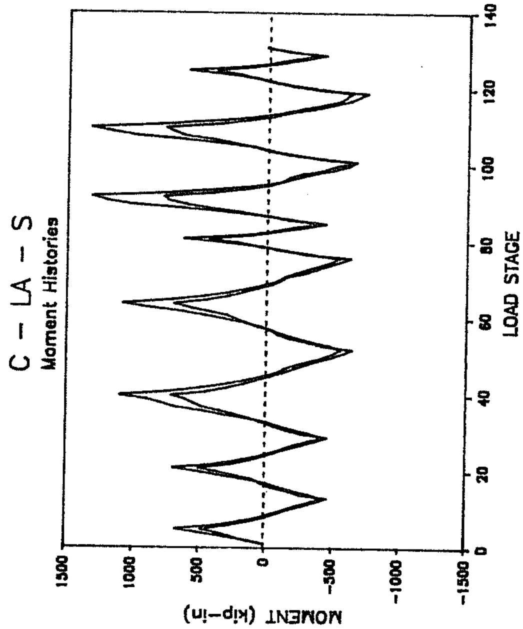


Figure 4.75

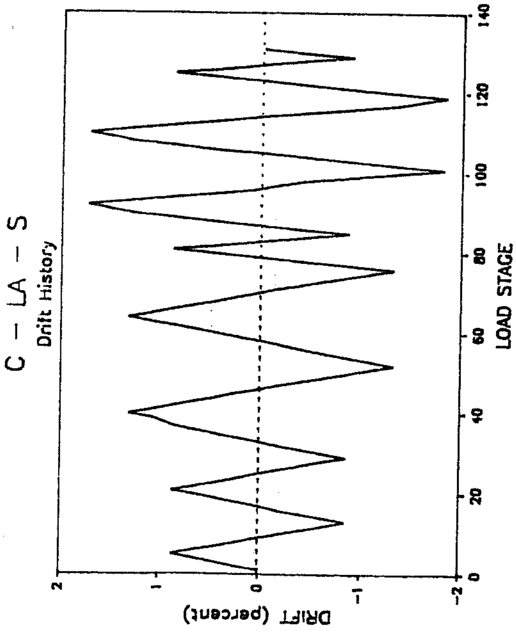


Figure 4.76

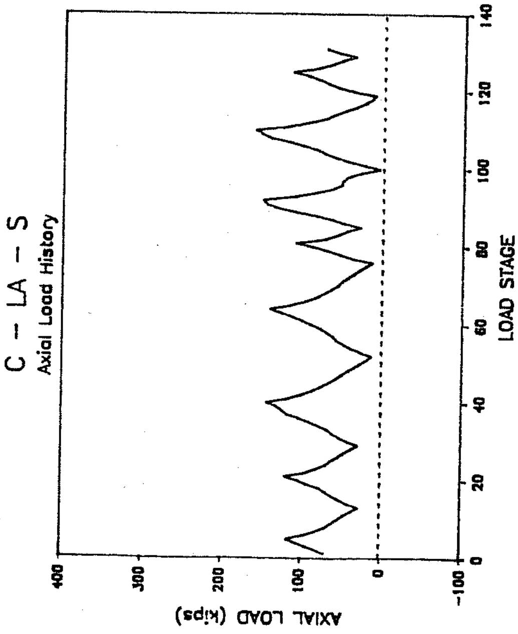


Figure 4.74

that the application of axial and lateral load were in phase. Also, they show that after an excursion to higher drift, returning to a previous drift level required less load than the previous cycle. This effect was more pronounced in the increasing compression direction (first quadrant) than the decreasing compression direction (third quadrant), as it was in all previous C series specimens.

4.6.2.2. Moment-axial load interaction. The calculated moment-axial load interaction diagram is shown in Figs. 4.77 and 4.78 along with the top and bottom moment-axial load path superimposed on it. The load paths remain below balance point because of the small constant of proportionality used during testing.

4.6.2.3. Lateral load-drift relationship. The lateral load-drift hysteresis relationship is shown in Fig. 4.79, and lateral secant stiffness at peak drift of each cycle is shown in Table 4.5. The hysteresis relationship shows the dissymmetry about the origin due to varying axial load, although this dissymmetry is not nearly as pronounced as the early stages of other constant eccentricity tests. Also, the increasing compression loop encompasses slightly less area than the decreasing compression loop, as was also the case for specimen C - LA - 1.

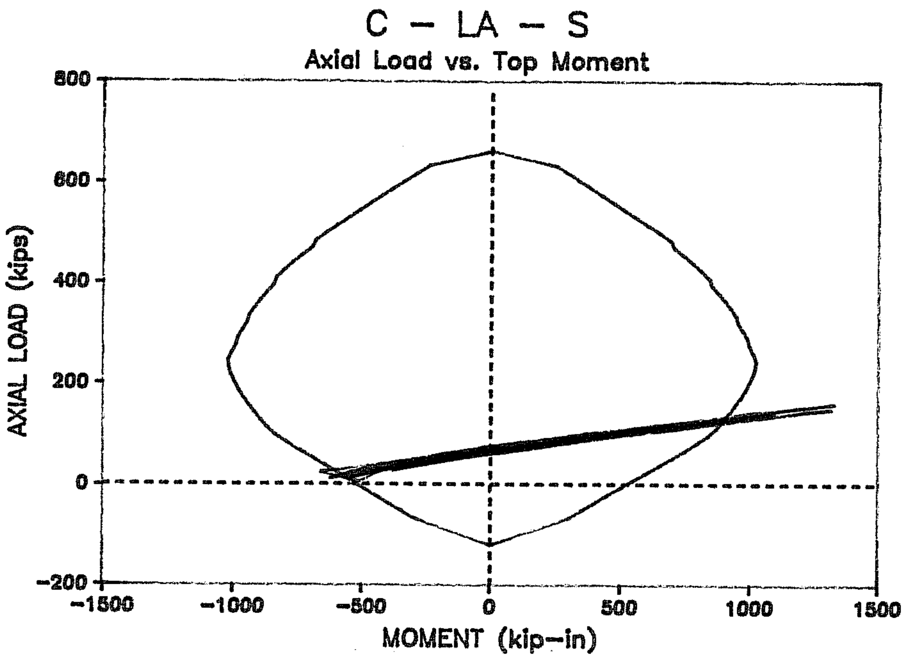


Figure 4.77

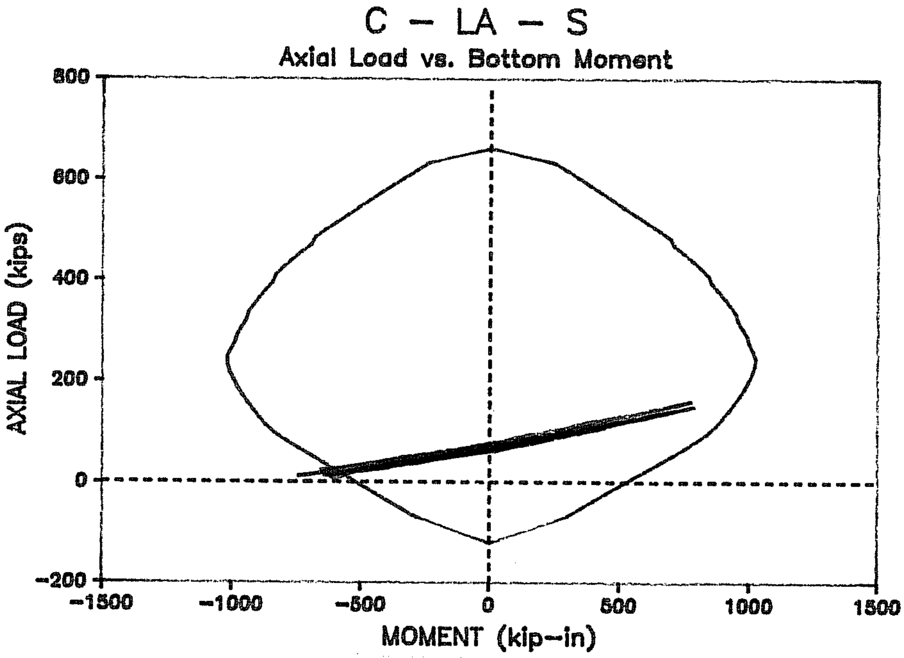


Figure 4.78

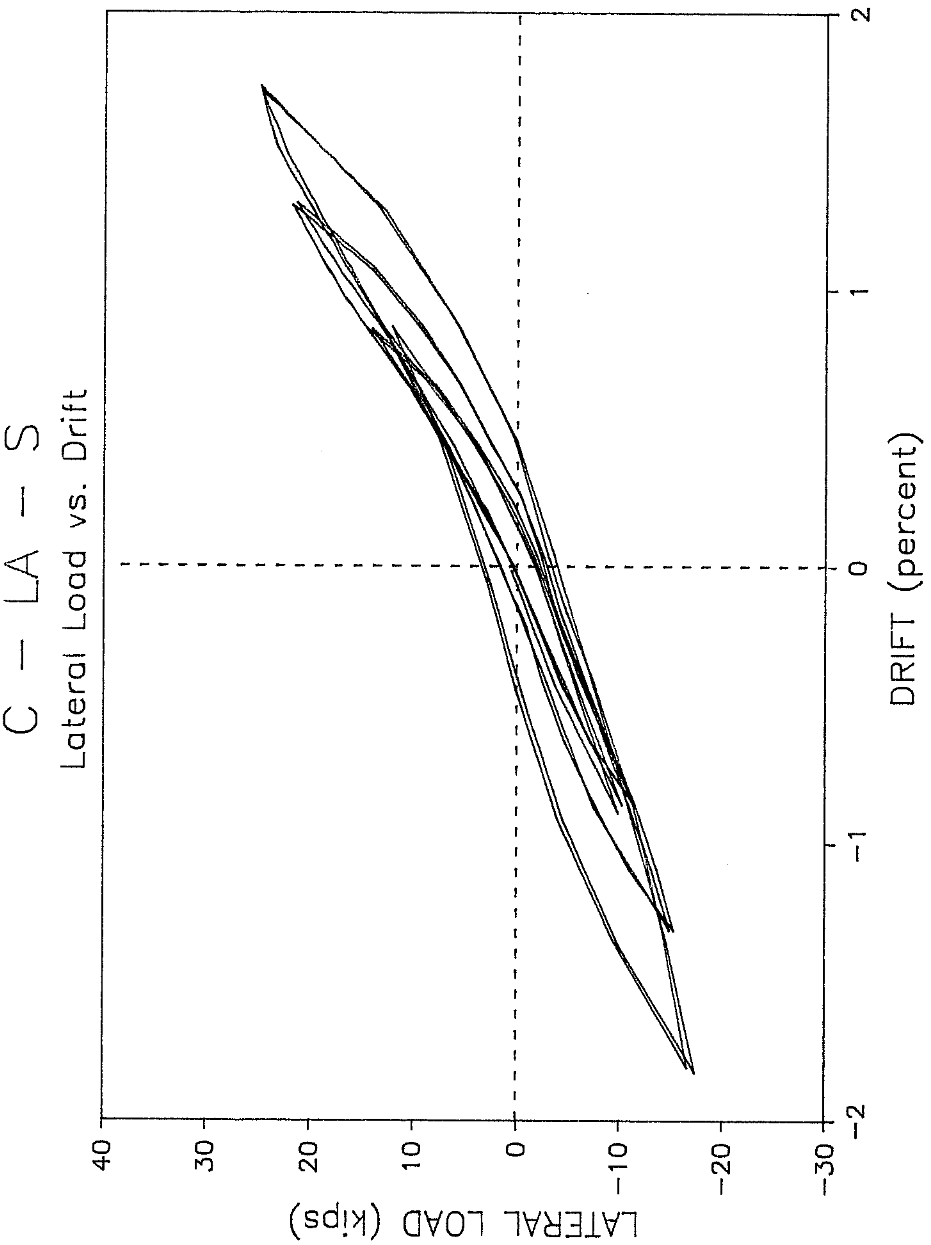

C - LA - S
Lateral Load vs. Drift
DRIFT (percent)
-2
-1
0
1
2
LATERAL LOAD (kips)
40
30
20
10
0
-10
-20
-30

Figure 4.79

SECANT STIFFNESSES
SPECIMEN C - LA - S

CYCLE NO.	DIRECTION	LATERAL LOAD (kips)	DRIFT (in.)	SECANT STIFFNESS (kip/in)
1	+	14.1	0.49	**27.3**
	-	-11.3	-0.54	**22.2**
2	+	14.6	0.49	**28.2**
	-	-11.3	-0.54	**22.1**
3	+	21.8	0.76	**27.8**
	-	-15.4	-0.82	**19.5**
4	+	21.4	0.76	**27.1**
	-	-15.0	-0.82	**19.0**
5	+	13.1	0.50	**25.1**
	-	-10.3	-0.54	**20.0**
6	+	24.9	1.02	**23.9**
	-	-16.7	-1.11	**18.0**
7	+	24.8	1.01	**24.0**
	-	-17.5	-1.12	**15.9**
8	+	12.2	0.50	**23.3**
	-	-10.0	-0.56	**18.6**

+ direction is increasing compression
- direction is decreasing compression

Table 4.5

Hysteresis loops of specimen C - LA - S are relatively slender when compared to previous constant eccentricity tests, indicating lower energy dissipation. This fact is not particularly surprising considering that the column had already been damaged by previous loading. Also noteworthy is the fact that after initial excursions to higher deflections, subsequent excursions to the same displacement return to nearly the same load. This seems to indicate that specimen behavior was controlled by reinforcement behavior, which is reasonable since loading remains below the balance point and cover concrete contribution was reduced due to cracking caused by the previous loading.

4.6.2.4. Strain histories. Strain gage histories from top longitudinal, bottom longitudinal, and transverse bars (Figs. 4.80, 4.81, and 4.72) show significant yielding of longitudinal steel and very low transverse steel strains. As was the case for specimen C - HA, longitudinal bars were strained in phase with principal loads, while transverse bars were strained at twice the frequency of the applied loads.

4.6.2.5. Moment-curvature relationships. Moment-curvature relationships (Figs. 4.82 and 4.83) show a considerable offset along the curvature axis. This indicates that some plastic deformation was left over from the previous loading applied to the column. Dissymmetry about the origin again

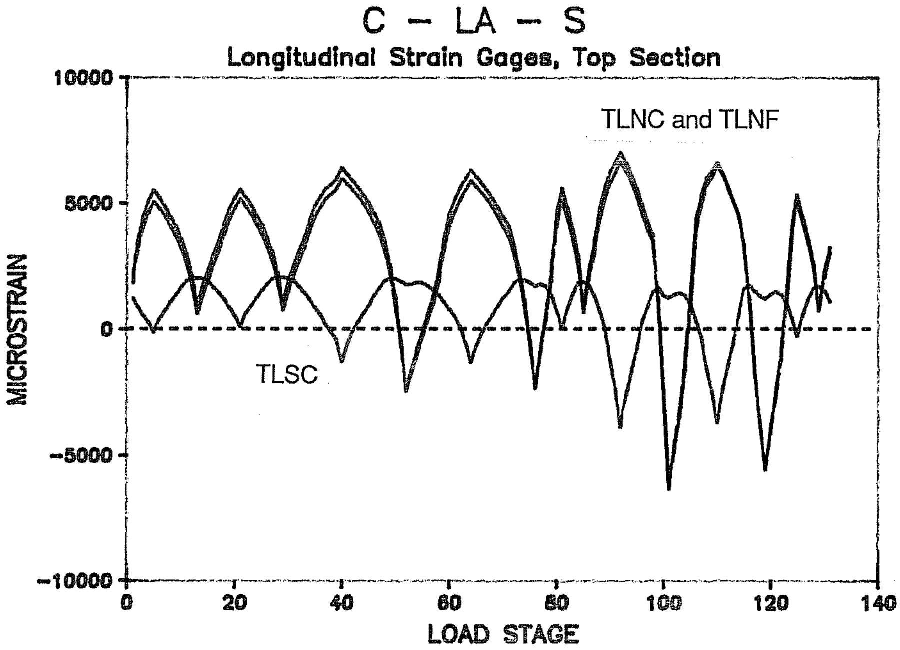


Figure 4.80

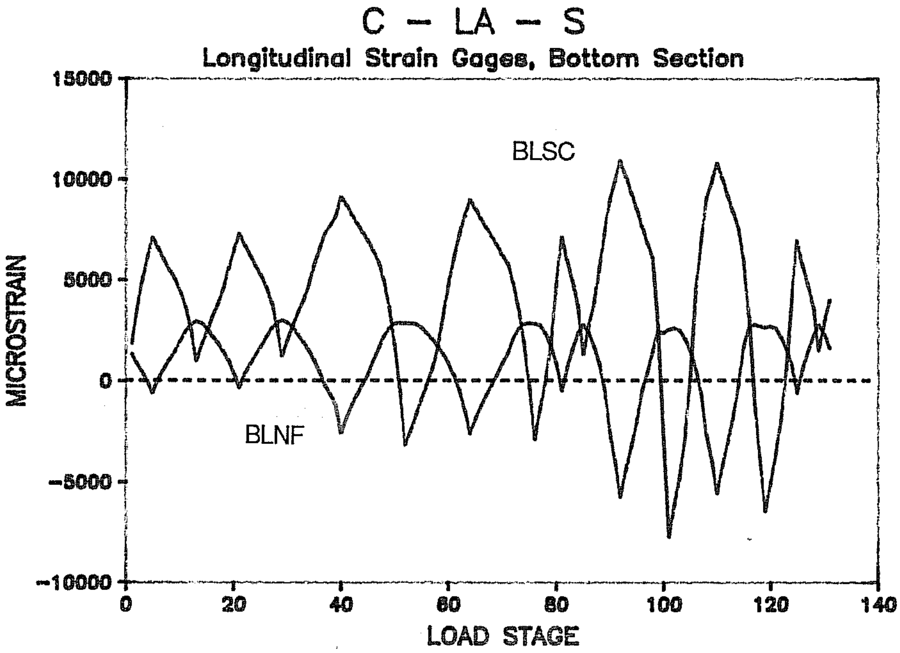


Figure 4.81

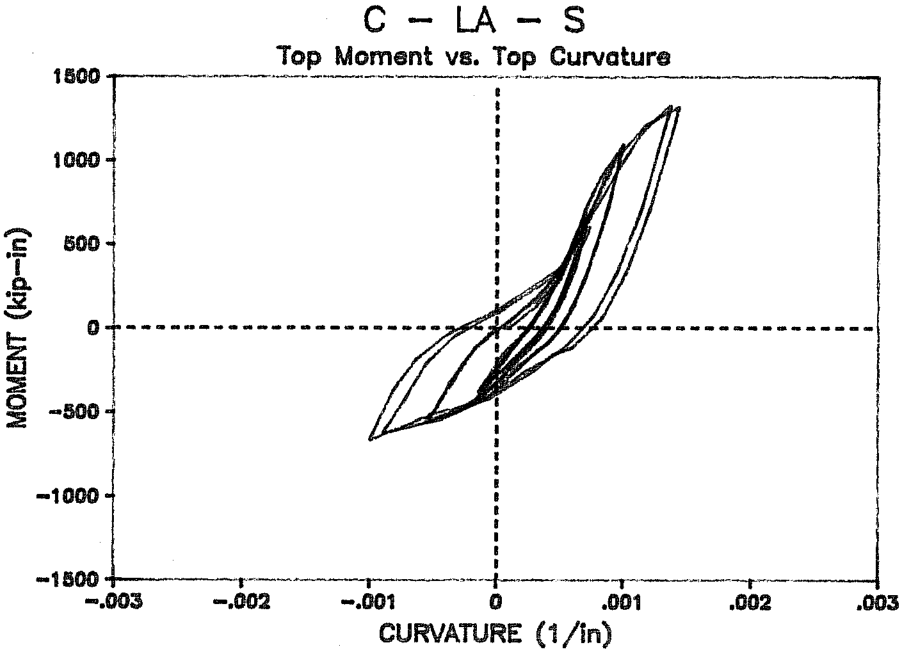


Figure 4.82

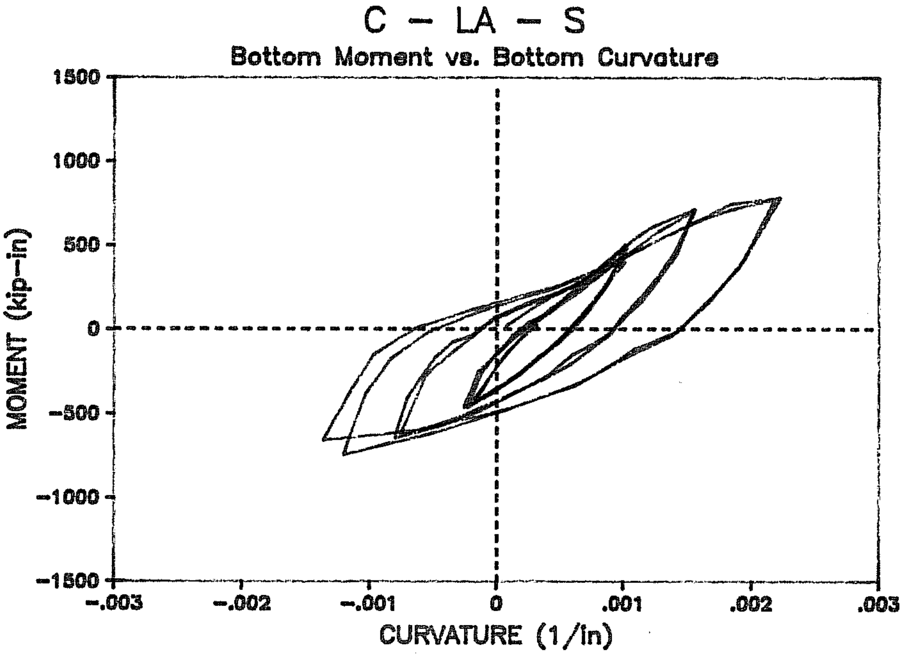


Figure 4.83

indicates the presence of varying axial load, a presence which is also noticable in moment-drift relationships (Figs. 4.84 and 4.85). Also, the same well-behaved unloading mentioned above is seen for these hysteresis relationships, and the decrease in stiffness after excursions to higher drifts is noticeable as the slope of the hysteresis loops decreases as the drift increases.

4.6.2.6. Cracking. Although this test was performed after the specimen was already damaged, cracking of specimen C - LA - S was quite similar to the cracking of specimen C - LA - 1. The cracks became inclined earlier for specimen C - LA - S because the shear strength of the column had been adversely affected by the previous test. Otherwise, the cracks were approximately the same in both specimens. Cracking of specimen C - LA - S at a late stage in the loading is shown in Fig. 4.86. Notice the large amount of cracking which occurred relative to the other C series specimens, probably due to damage sustained by previous loading.

4.7. Summary and Conclusions

General behavioral trends of tests performed in this experimental program include:

1. In constant eccentricity tests, after an initial excursion to lateral load-drift combinations, subsequent excursions to an

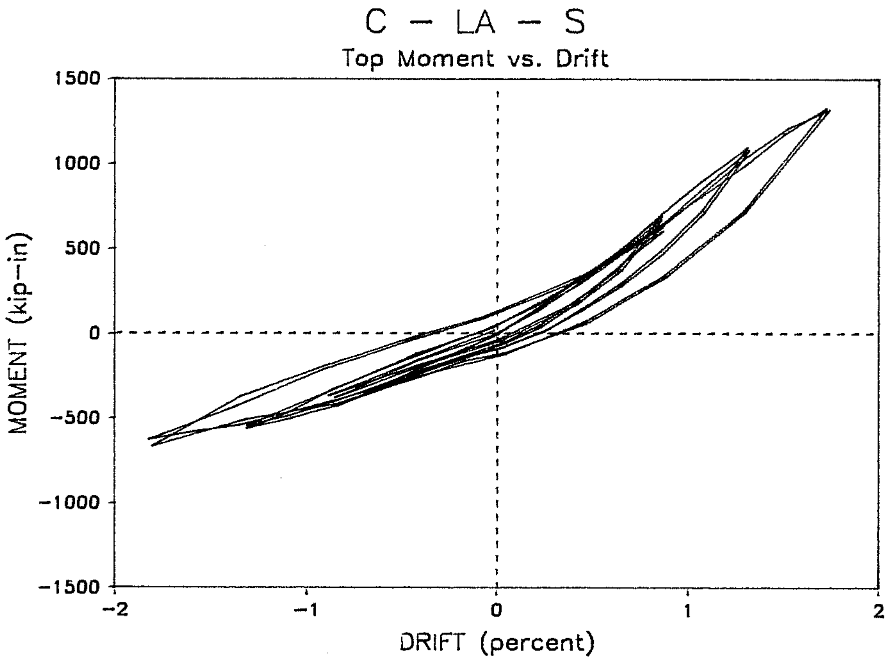


Figure 4.84

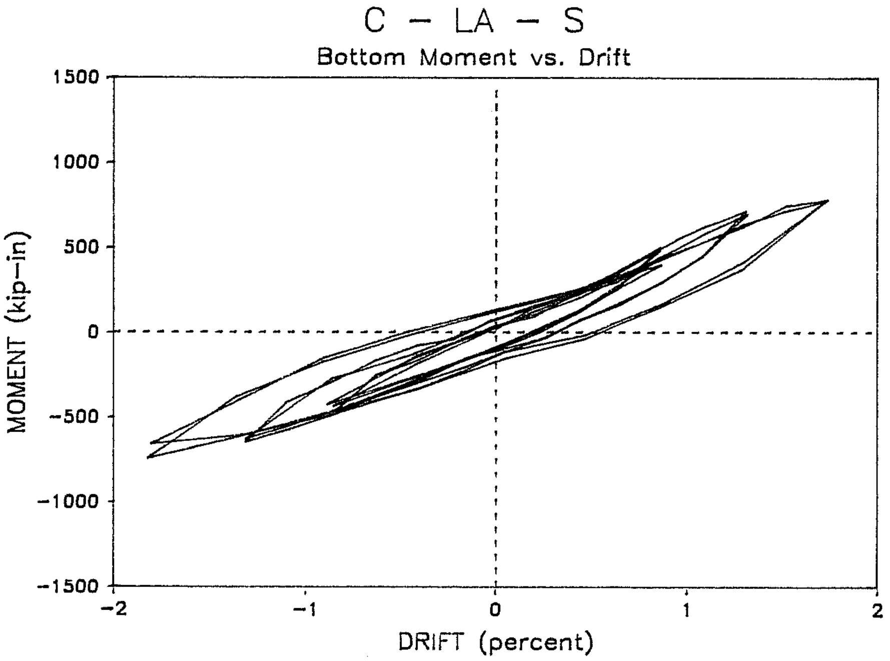


Figure 4.85

FINAL CRACK PATTERN
C - LA - S

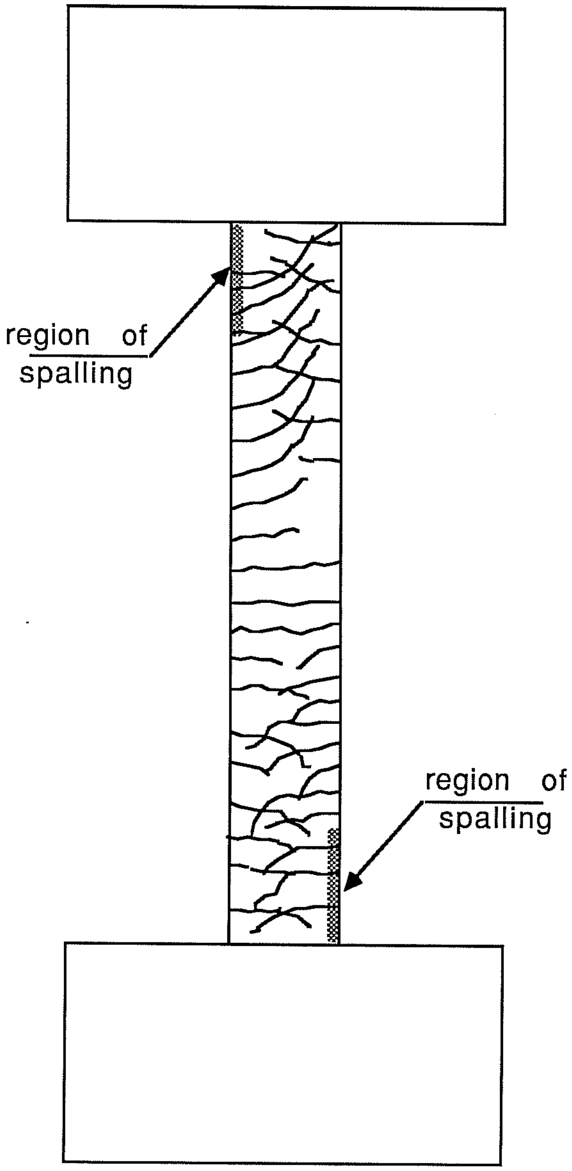


Figure 4.86

equal drift resulted in a lower magnitude of lateral load.

2. The reduction in lateral load of subsequent excursions in the increasing compression direction were different from corresponding reductions in the decreasing compression direction. Reductions in the increasing compression direction were due to loss of concrete strength (spalling, cracking, etc.) and were generally larger than reductions in the decreasing compression direction, which were caused by changes in steel stiffness due to the Bauschinger Effect.

3. Lateral load-drift, moment-drift, and moment-curvature relationships of reinforced concrete columns subjected to varying axial and lateral load are not symmetrical about the origin. This differs from similar relationships of columns subjected to constant axial load and reversed cyclic lateral load, which are symmetrical about the origin. In constant eccentricity tests, the amount of dissymmetry

was related to the magnitude of axial load variations.

4. Energy dissipation, which is related to hysteresis loop size, was a function of how much inelastic deformation was experienced by the column.

5. Specimen behavior was sensitive to maximum axial compression experienced during testing. When axial loads exceeded balanced axial load, lateral stiffness quickly degraded and lateral capacity of the column dropped well below ultimate. When axial loads were kept below balanced point, lateral stiffness did not degrade as quickly and lateral capacity of the column was maintained near ultimate capacity.

6. Hinge length increased slightly throughout the constant eccentricity tests. This was implied by the difference between moment-drift and moment-curvature relationships. However, based on observations of spalling in the specimens, a hinge length of ten inches appears to be reasonable to use for analysis.

7. More cracks formed in the decreasing compression direction than the increasing

compression direction, and cracking occurred over more of the column.

8. Lateral stiffness of a reinforced concrete column is profoundly influenced by axial load. As axial load decreases, so does lateral stiffness. This is clearly illustrated in the behavior of specimen I - HA. On two occasions, that specimen experienced a simultaneous decrease in lateral load and increase in drift resulting from a rapid decrease in axial load.

CHAPTER 5

DEVELOPMENT OF ANALYTICAL MODEL

5.1 Overview

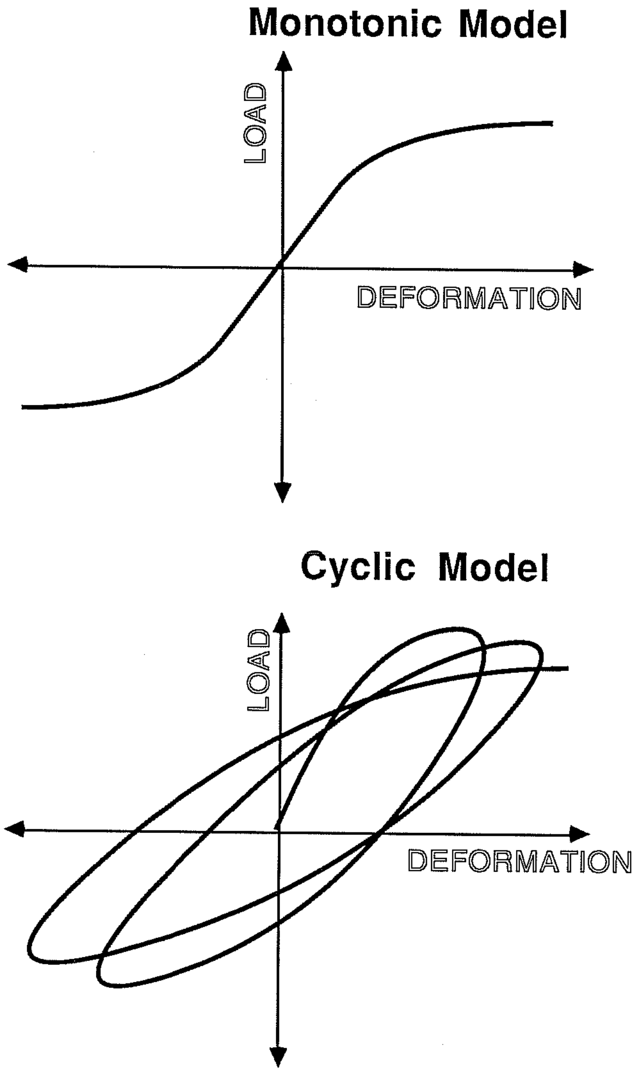
This chapter describes the development and implementation of an analytical model for a reinforced concrete column subjected to variations in both axial and lateral loads. Theoretical basis for the model is discussed along with the algorithms used in its implementation.

5.2. Scope of Model

Analytical models for behavior of reinforced concrete members fall into two general categories : monotonic and cyclic. A typical example of each is shown in Fig. 5.1. Monotonic models simulate member response to an imposed load or deformation without taking into account changes in the member which may have occurred due to previous loading. No attempt is made to model effects which occur due to load repetition or reversals, such as loss of stiffness after an initial excursion to a given load or deterioration of steel-concrete bond. Monotonic models can be used to generate envelopes of member response which can be a good estimate of the upper bound of the cyclic response.

Cyclic models, on the other hand, attempt to include all characteristics of the inelastic response of a member,

MONOTONIC VS. CYCLIC MODELS



Monotonic model is simple, generates failure envelope, upper-bound of cyclic behavior

Cyclic model is complex, gives hysteresis loops for load reversals

Figure 5.1

including those due to loading history. Such models are very complex, and often involve dozens of hysteresis rules for determining the member response based on both current and previous stress or strain states. While cyclic models may be useful for inelastic analysis of a structure, they are beyond the scope of this study. Rather, monotonic models will be used to aid in interpretation of test results. It is important to note, however, that many of the principles used in the development of a monotonic model have direct application to the cyclic problem.

5.3 Theoretical Development

Modelling a complex system is best acheived by dividing the system into a series of smaller subsystems which can be modelled seperately and then integrated into a single unit. This requires that the system be divided into levels of description, each of which operates on information passed to it by the level directly below. Results of the operation performed at a level is then passed on to the next highest level. Ideally, the lowest level is based on physical laws and the highest level is a simple description of complex phenomena.

An excellent example of this approach to problem solving is a computer program. The highest level is the function performed by the program, for instance making a copy of a disk file. This can be understood quite easily in conceptual terms.

Blocks of data are transferred from one file to another, much as paper in a manila folder is copied and placed in another folder. This conceptual model can then be broken into a series of specific computer instructions which can be written in a high level language, such as FORTRAN or Pascal. This language is then translated into assembly language, which is much more specific than a high level language but is also more difficult to grasp conceptually. Nevertheless, assembly language can be easily translated into machine instructions which can be executed by a computer. These machine instructions, in turn, correspond to electronic circuits which perform functions associated with the instruction. These circuits function through the movement of electrons, which move according to established physical principles such as Ohm's Law. Therefore, the complex model of a computer program for transferring data can be broken down into a series of simple submodels, each of which performs a function which is easily understood. However, trying to copy a disk file by describing the movement of electrons is an insurmountable task. That is why defining distinct levels of description for complex tasks is so useful; it allows a conceptual leap to be conveniently divided into several small steps.

For a reinforced concrete column, there are three levels of description into which the column can be divided. First of these is the material level where the behavior of steel and concrete is described. Material behavior is typically defined by stress-strain curves which are determined experimentally. The second level is that of the column section. Section behavior is expressed as a moment-curvature relationship which is derived from the material stress-strain curves. The highest level of description is the global column behavior which is defined by load-deformation relationships which are derived from the moment-curvature relationship of the section. Ultimately, these high-level descriptions of column behavior can be used to describe even higher level phenomena, such as frame behavior. For this study, however, global column behavior is the highest level of description. Figure 5.2 shows the hierarchy of levels and the information which is passed between them.

Each of these levels of description -- material, section, and column -- is associated with a type of nonlinearity. Nonlinearity, for the purposes of this discussion, is a change in the stiffness of the model of a level of description. Nonlinearity is a source of concern in analytical modelling, because solving nonlinear problems is considerably more difficult than solving linear problems. Hence, it is important to identify sources of nonlinearity when developing an analytical model.

On the material level, nonlinearity is the result of steel and concrete behavior. Steel behavior, although linear in

MODEL HIERARCHY

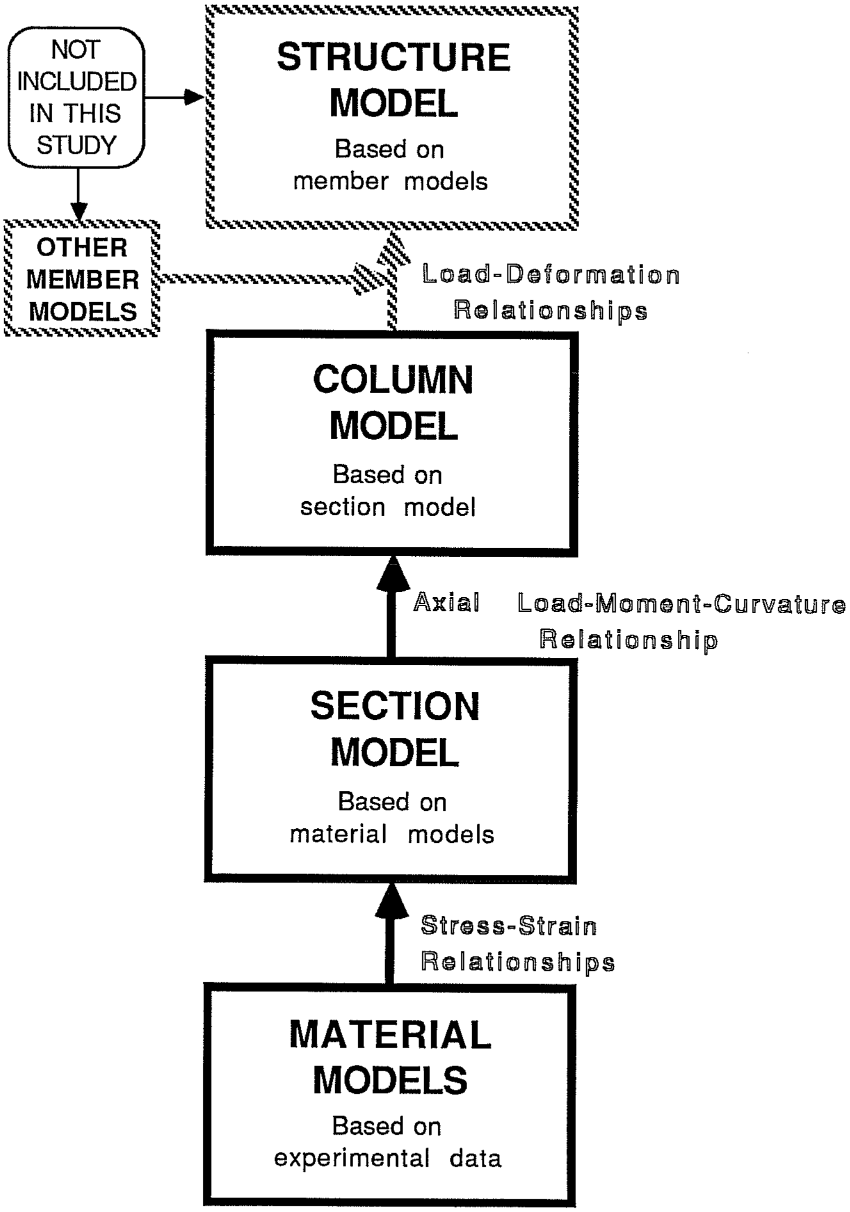

NOT INCLUDED IN THIS STUDY
STRUCTURE MODEL
Based on member models
OTHER MEMBER MODELS
Load-Deformation Relationships
COLUMN MODEL
Based on section model
Axial Load-Moment-Curvature Relationship
SECTION MODEL
Based on material models
Stress-Strain Relationships
MATERIAL MODELS
Based on experimental data

Figure 5.2

the elasic range, becomes nonlinear once strain exceeds yield strain, and concrete behavior is inelastic and nonlinear from very low strain levels because of microcracking. On the section level, nonlinearity results because section stiffness is very sensitive to axial load. Consequently, moment-curvature relationships must be determined for several axial loads in order to account for changes in section stiffness. On the column level, axial load also influences lateral stiffness of long columns because axial load produces additional moments when a column is in a deformed state. All three of these sources of nonlinearity were accounted for in the present analysis in a fashion described later in this chapter.

5.4. Material Models

In this section, material models used for steel and concrete are described.

5.4.1. Steel model. For steel, a stress-strain relationship of the type shown in Fig. 5.3 was used. An elastic stiffness of 29000 ksi (4205 MPA) and a yield plateau from yield strain up to a strain of one percent were assumed. Strain hardening was assumed to be linear, and continued until ultimate stress is reached at a strain of ten percent. Beyond the strain at ultimate stress, the steel was assumed to maintain the

STEEL MODEL

Multilinear Idealization

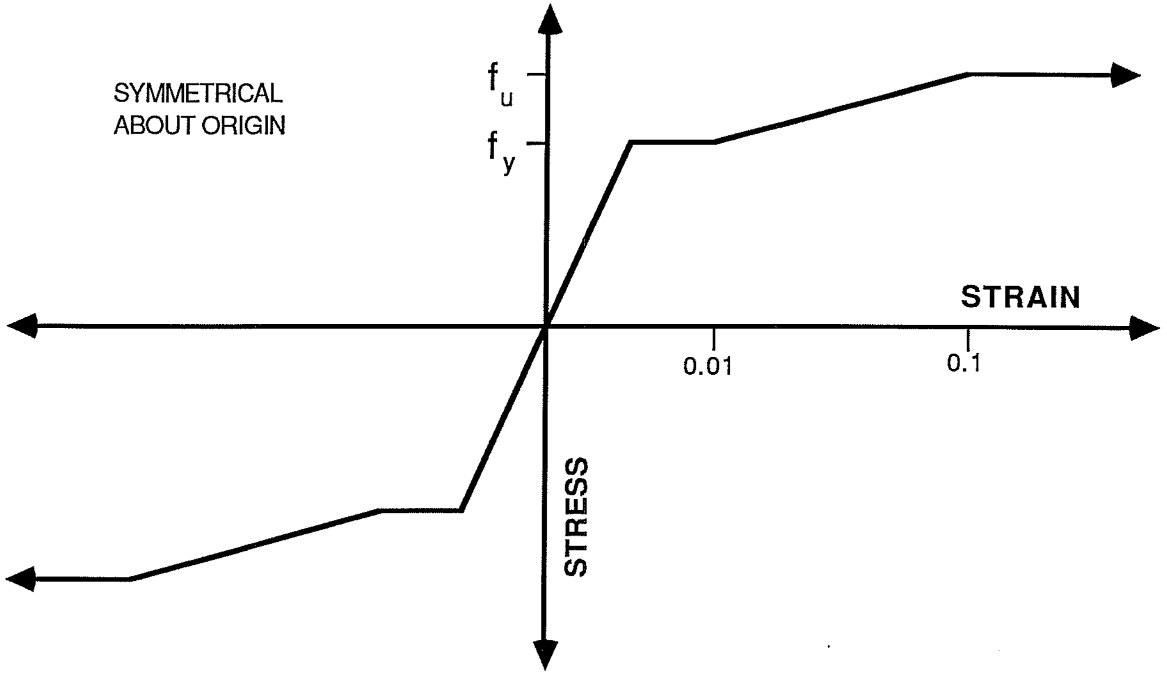


Figure 5.3

ultimate stress. This stress-strain relationship was assumed to be symmetrical about the origin.

To calibrate this model, it was necessary to obtain the yield stress and ultimate stress of reinforcing bars used in the tests. For this experimental program, these values were determined from tests performed on a 600 kip universal testing machine at the Phil M. Ferguson Structural Engineering Laboratory. Bars with lengths varying from 8 to 12 inches were tested. Because reinforcing bars used in the experimental program were from the same heat, steel properties were assumed to be identical for all specimens.

5.4.2. Concrete model. Concrete behavior was represented using a stress-strain relationship very similar to that determined by Scott, Park, and Priestley [8]. This model, which was developed for concrete loaded at low strain rates, assumes a parabolic function up to ultimate stress, followed by a linear descending branch which continues to some minimum stress. Confined and unconfined concrete behave differently, and the model accounts for this difference. Graphs of stress-strain relationships for both confined and unconfined concrete are shown in Fig. 5.4.

The ascending branch of confined concrete is defined by the equation :

$$\sigma(\varepsilon) = Kf_c'[2(\varepsilon/\varepsilon_{0c}) - (\varepsilon/\varepsilon_{0c})^2]$$

CONCRETE MODEL

Based on Scott, Park, and Priestley

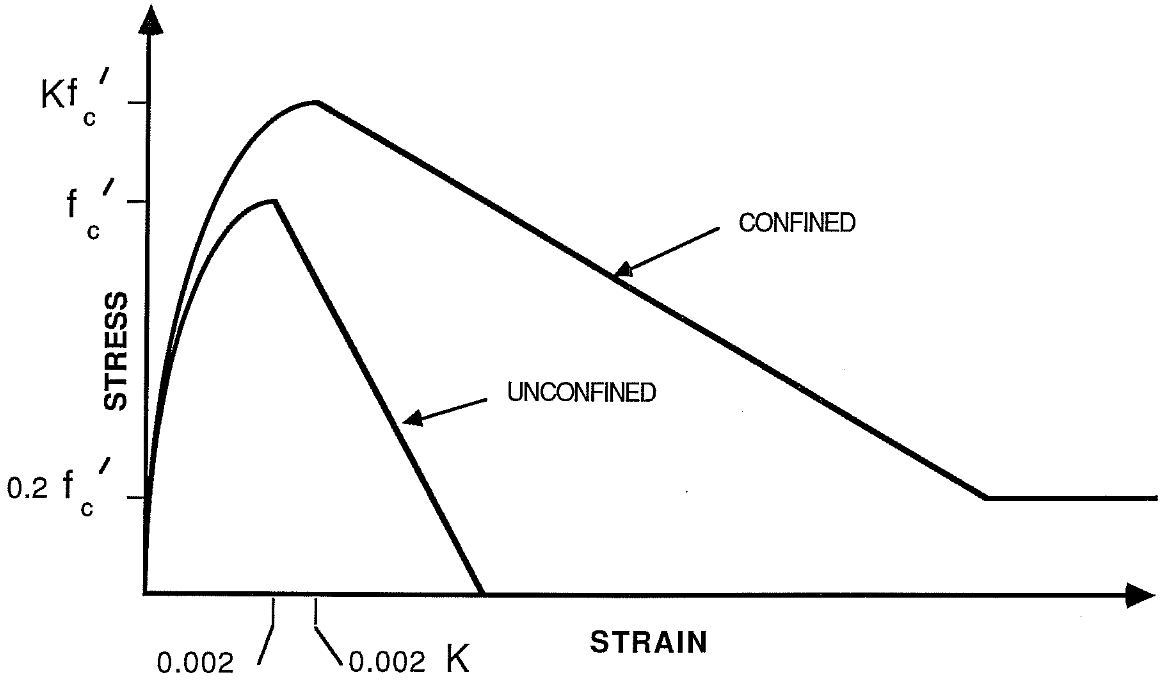


Figure 5.4

where

$$K = 1 + \frac{\rho_s f_y}{f'_c} \qquad \rho_s = \frac{\text{volume of transverse steel}}{\text{volume of confined core}}$$

$$\varepsilon_{0c} = 0.002K$$

f_y = yield stress of transvere steel (ksi)

f'_c = compressive strength of concrete (ksi)

The descending branch of confined concrete is defined by the equation :

$$\sigma(\varepsilon) = Kf'_c[1-Z_c(\varepsilon-\varepsilon_{0c})] \geq 0.2\ Kf'_c$$

where

$$Z_c = \frac{0.5}{\varepsilon_{50h}+\varepsilon_{50u}-\varepsilon_{0c}} \qquad \varepsilon_{50h} = 0.75\ \rho_s\sqrt{b''/5}$$

b'' = width of confined core

$$\varepsilon_{50u} = \frac{3 + 2f'_c}{1000(f'_c-1)}$$

s = stirrup spacing

Confined concrete is assumed to have a minimum strength for very large strains. The ascending branch of unconfined concrete is defined by the equation :

$$\sigma(\varepsilon) = f'_c[2(\varepsilon/\varepsilon_{0u}) - (\varepsilon/\varepsilon_{0u})^2]$$

where

$$\varepsilon_{0u} = 0.002$$

The descending branch of unconfined concrete is defined by the equation :

$$\sigma(\varepsilon) = f_c' [1 - Z_u (\varepsilon - \varepsilon_{0u})] \geq 0$$

where

$$Z_u = \frac{0.5}{\varepsilon_{50u} - \varepsilon_{0u}}$$

Unconfined concrete is assumed to have no strength at large strains. This models the loss of cover which may occur under high loads.

The tensile strength for all concrete was assumed to be zero. This is one difference between Scott, Park, and Priestley and the model used by the author. This is done mainly for simplicity in modelling the section response, as will be discussed later. Another difference between the two models was the descending branch of unconfined concrete. The Scott, Park, and Priestly model assumed a linear descending branch from a strain of two to three percent. At strains greater than three percent, unconfined concrete was assumed to have no strength. This instantaneous decrease in strength was difficult to use in numerical analysis so the descending branch in the author's model was assumed to continue at the same slope until it intersected the strain axis.

The model was calibrated using results from uniaxial compression tests on plain concrete cylinders cast simultaneously with column specimens. A minimum of three cylinders were tested for each column specimen, and the maximum compressive stresses reached during these tests were averaged. This average value was then used as f'c in the concrete stress-strain model. The values of f'c determined for each test are shown in Table 3.1.

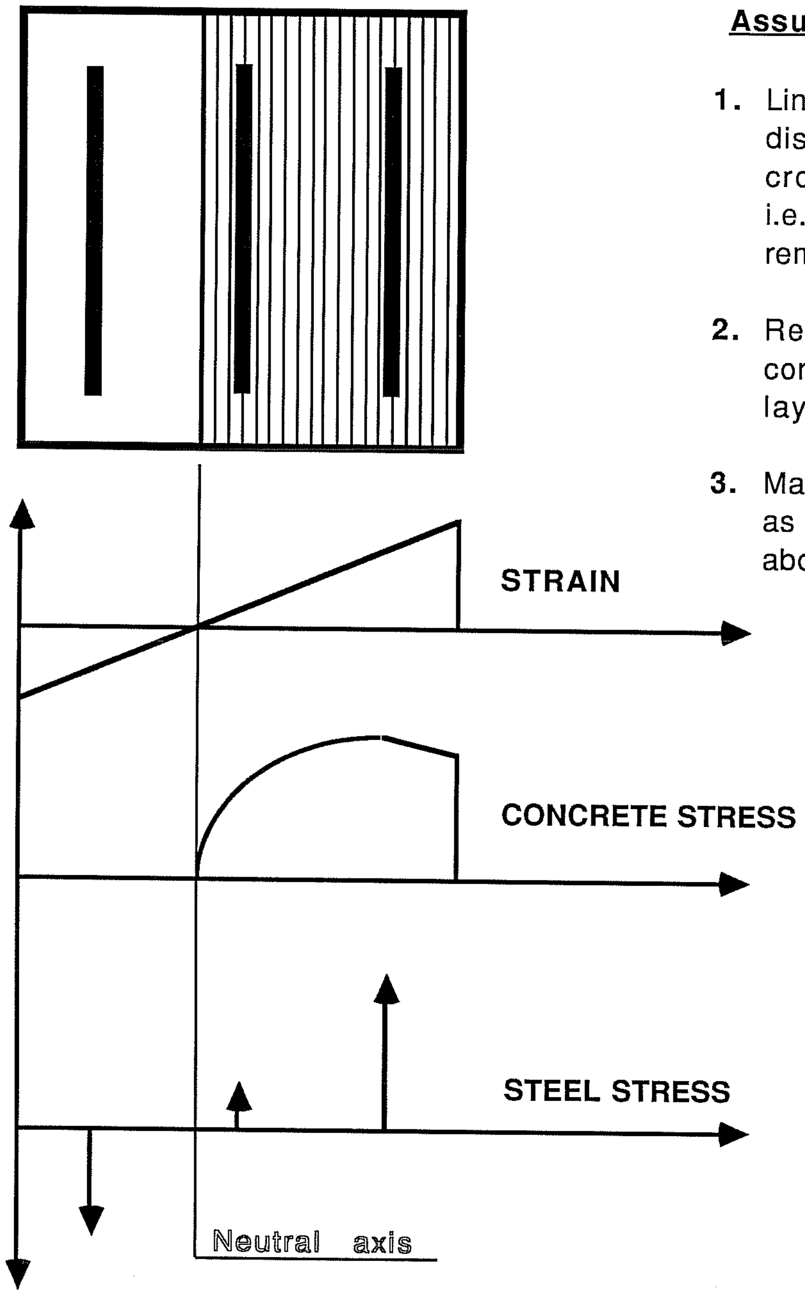
5.5. Section Model

The method used to obtain an axial load-moment-curvature relationship for a reinforced concrete section is based on a method which is well-known and often implemented [28-34]. An excellent brief description is given by Wakabayashi in Ref. 34. A schematic of the section model is shown in Fig. 5.5.

5.5.1. Idealizations. This analytical model is based on two idealizations of the section properties and one assumption about section response. First, reinforcing steel is assumed to be concentrated in layers. These layers are assigned an area equal to the actual area of reinforcing steel, but are assumed to have no thickness. Placing steel in such layers makes numerical "bookkeeping" straightforward, and assuming the layers have zero thickness improves the convergence of the solution algorithm. These points will be given additional treatment later in this chapter.

The second idealization is the concrete contribution to the section can be represented as a multi-slice stress zone (see

SECTION MODEL



Assumptions:

1. Linear strain distribution over cross-section, i.e. plane sections remain plane

2. Reinforcement concentrated in layers

3. Material behaves as predicted by above models

Figure 5.5

Fig. 5.5). Since concrete stress is assumed to be zero for tensile strains, only concrete above the neutral axis contributes to the behavior of the section. This stress zone, then, always has dimensions of the width of the section and the depth of the neutral axis and only contributes compressive force to the section response. Calculating force and moment contribution of the stress zone is accomplished by dividing it into a series of equal slices. Each slice is assumed to have constant strain, so the stress in each slice can be determined using the concrete stress-strain relationship. The force developed by a slice can then be found by multiplying that stress by the thickness and width of the slice. These slice forces, together with their location in the stress zone, can be used to determine the total force and force centroid of the stress zone.

The third and most important idealization of the section model is that plane sections remain plane for all load combinations. This is a crucial assumption for two reasons. First, it implies that no shear deformation occurs in the column. This is rarely, if ever, true. However, shear deformation, in many cases, is much smaller than flexural deformation and can be neglected for analysis purposes. Second, the plane-sections-remain-plane assumption implies a linear strain distribution over the cross section. Hence, the potentially unwieldly task of determining a strain distribution reduces to obtaining just two coefficients, which is enough to define a linear distribution. Data gathered in the experimental program described in Chapters 3 and 4 seems to support this assumption for all but the most extreme loads experienced by columns tested in that program.

5.5.2. Calculation of section model. Determination of the axial load-moment-curvature relationship for a reinforced concrete section is accomplished in two phases. First, limits of the response are calculated, based on some definition of failure. These limits keep the search for solutions both confined enough to be well-behaved and comprehensive enough to allow for accurate representation of the response. Once limits have been obtained, the second phase begins. This phase fills in the axial load-moment-curvature response between the limits obtained from the first phase of analysis.

In principle, there are many ways to accomplish the second phase. For instance, one approach is to pick a moment-curvature combination and calculate the corresponding axial load. This type of approach is useful only if the unknown variable is a single-valued function of the other two variables, otherwise it is very difficult to implement because there may not be a unique solution. As it turns out, this unique solution requirement is only satisfied when axial load and curvature are used to determine bending moment, so this approach was used in the current analysis. Consequently, the first phase of analysis is to determine limiting values of axial load and curvature for the section response.

Three axial load limits, shown in Fig. 5.6, are required for the section model. First, ultimate tensile capacity is determined from steel properties. Next, ultimate compressive capacity is determined from concrete and steel properties. One other value of axial load must be determined for the analysis because of the method used to obtain curvature limits.

Determination of curvature limits for the section is carried out in conjunction with the calculation of the moment-axial load interaction diagram. Calculation of the interaction diagram requires a definition of failure of the section. Following standard practice, as expressed in ACI 318-83 Standard Building Code and Commentary [22,23], failure was assumed to occur when the extreme fiber compressive strain of the section reached a level of 0.003. This definition is adequate for most values of axial load, but not all. When the neutral axis of the section approaches zero, strains in the steel grow very large. For this reason, a lower limit on axial load greater than the ultimate tensile capacity is necessary for analytical purposes. This lower limit is chosen so that the neutral axis is zero, thus making the extreme fiber strain zero as well, and the maximum

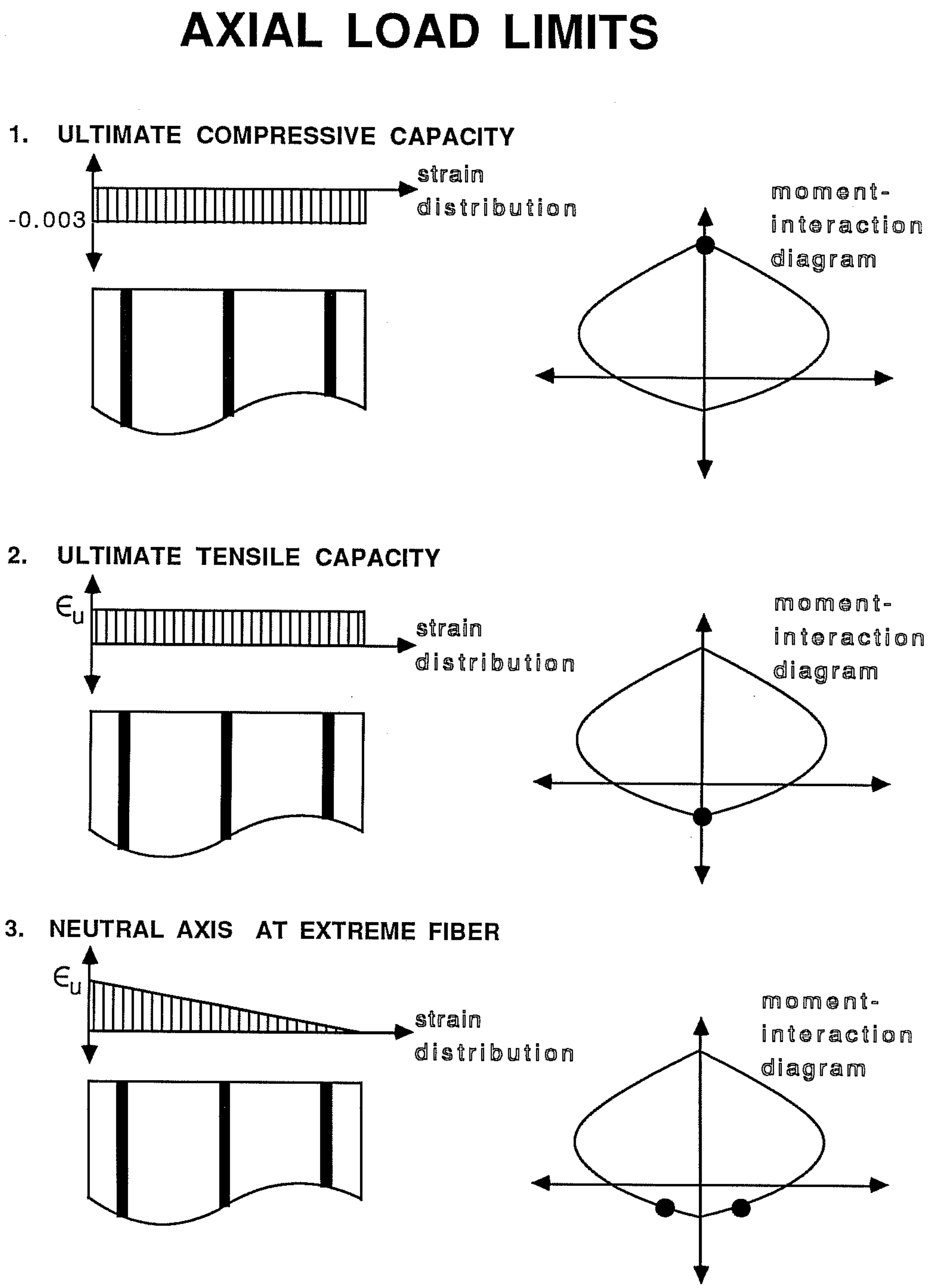

AXIAL LOAD LIMITS
1. ULTIMATE COMPRESSIVE CAPACITY
-0.003
strain distribution
moment-interaction diagram
2. ULTIMATE TENSILE CAPACITY
ϵ_u
strain distribution
moment-interaction diagram
3. NEUTRAL AXIS AT EXTREME FIBER
ϵ_u
strain distribution
moment-interaction diagram

Figure 5.6

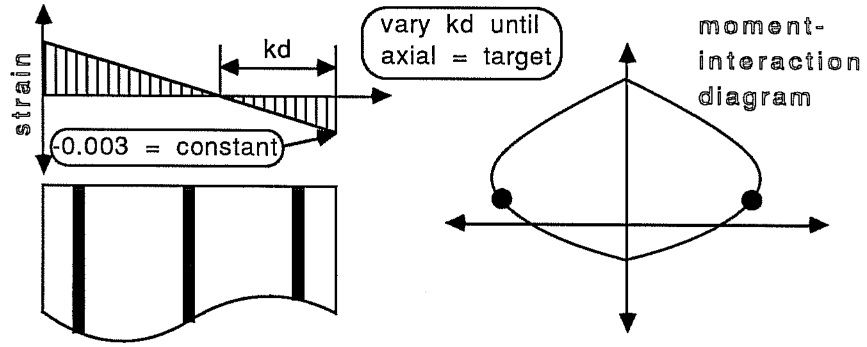
steel strain is the strain of the steel at ultimate stress. For axial load levels between this lower limit and the tensile capacity of the section, straight-line interpolation is used. This is a reasonable practice because the lower limit of axial load is generally a fairly large tensile value, and so axial load-moment combinations between the lower limit and the ultimate tensile capacity are rarely experienced and would be expected to be fairly linear.

Points on the interaction diagram are found using an iterative procedure. First, a target axial load is selected. Target values are determined by dividing the range of possible axial loads (between ultimate compressive load and the zero-neutral-axis load described above) into equal parts. Then, while keeping the extreme compressive fiber strain equal to the ultimate concrete strain, the neutral axis is varied until this target axial load is obtained. Once the corresponding neutral axis has been found, bending moment and curvature can be calculated. This procedure is then repeated for all values of axial load, thus generating a moment-interaction diagram complete with the curvature corresponding to each axial load-moment combination.

Curvature values associated with interaction points provide a starting point in the search for curvature limits. While these curvatures are, by definition, failure curvatures,

CALCULATION OF MOMENT-CURVATURE DIAGRAM FOR A GIVEN AXIAL LOAD

1. DETERMINATION OF CURVATURE LIMITS



Failure curvature = 0.003 / kd

→ **Curvature limit = Failure curvature X Ductility**

2. CALCULATION OF MOMENT-CURVATURE RELATIONSHIP

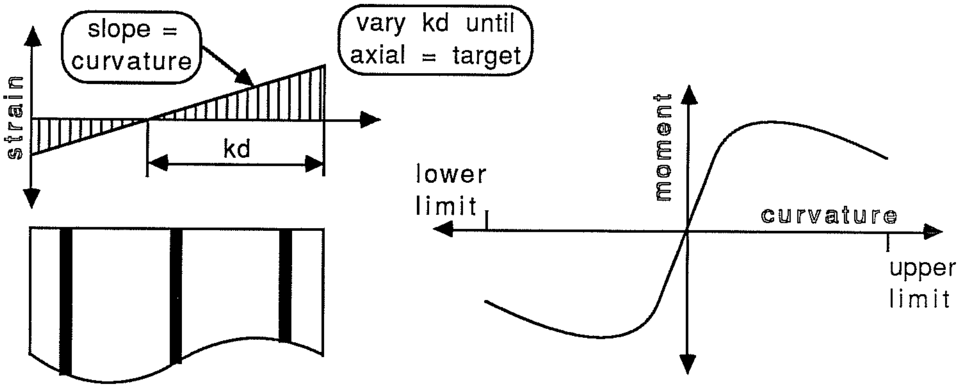


Figure 5.7

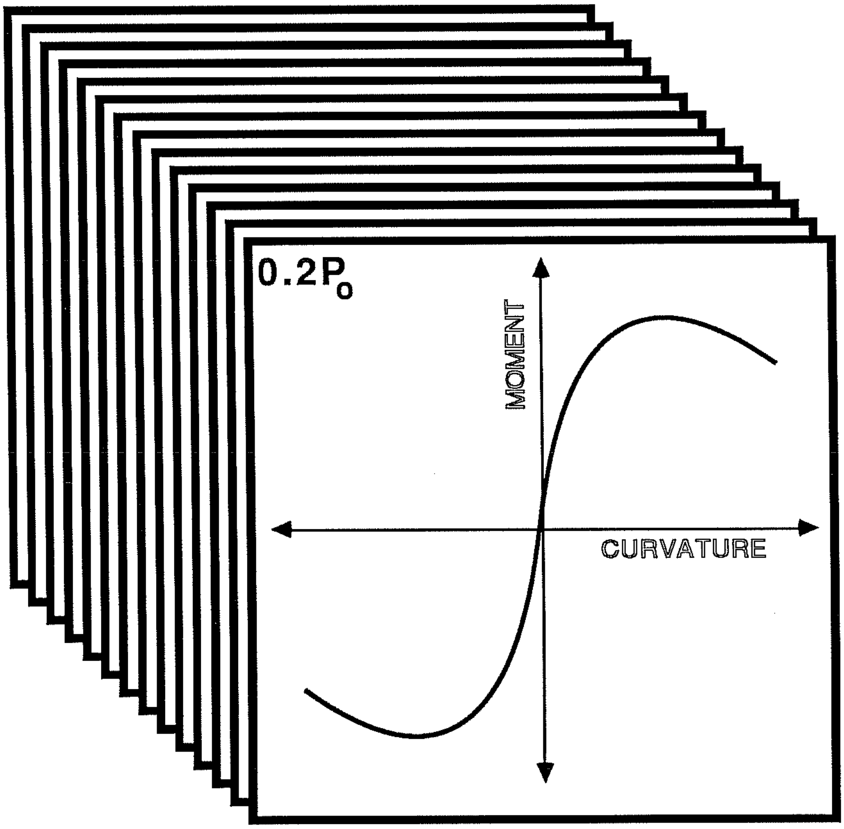
Using this relationship to obtain arbitrary axial load-moment-curvature combinations is straightforward but somewhat difficult to conceptualize. Figure 5.8 provides a means for visualizing this relationship as a series of "sheets," each one containing a moment-curvature relationship for a single axial load. To find the moment corresponding to a curvature-axial load combination it is necessary to determine between which "sheets" the point lies. Once these sheets are identified, the points on each sheet which bracket the target curvature are determined. This gives four curvature-axial load combinations, two from each sheet. Each of these points has a corresponding moment associated with it, so the moment for the target curvature-axial load pair can be obtained by using a linear interpolation between the four known points. In this manner, any axial load-moment-curvature combination can be obtained, provided enough sheets are generated with sufficiently precise moment-curvature diagrams.

5.6. Column Model

Two different methods are used in this study to obtain load-deformation relationships for reinforced concrete columns. Both models are based on the schematic shown in Fig. 5.9. The first is a load-control method which is well-known and often implemented. Wakabayashi [34] gives an excellent brief outline of the general method, although there are a few minor differences

AXIAL LOAD-MOMENT-CURVATURE RELATIONSHIP

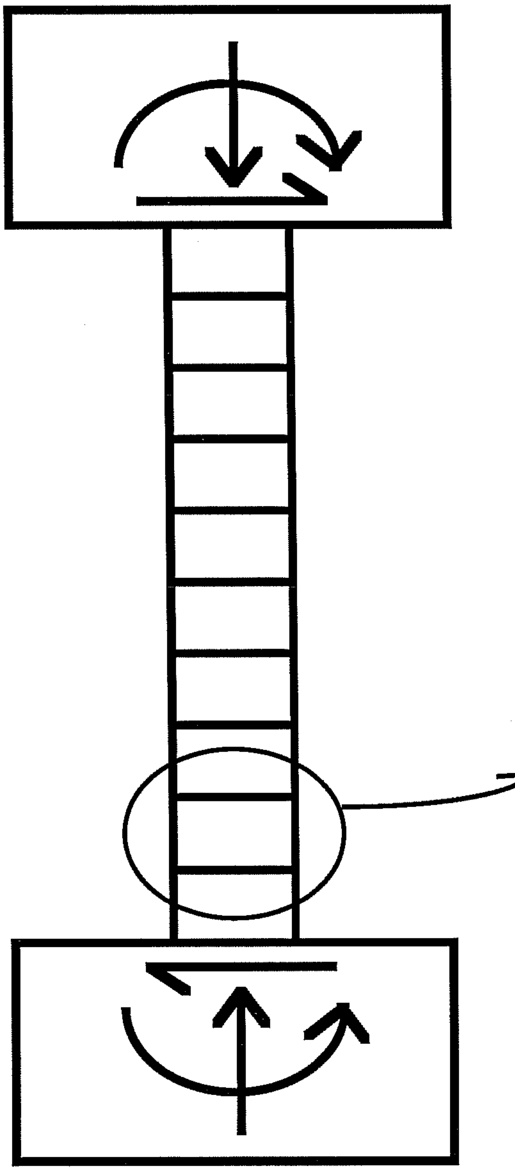
This is a conceptual representation of the relationship



Each sheet has a different axial load and contains the corresponding moment-curvature relationship

Figure 5.8

ANALYTICAL COLUMN MODEL

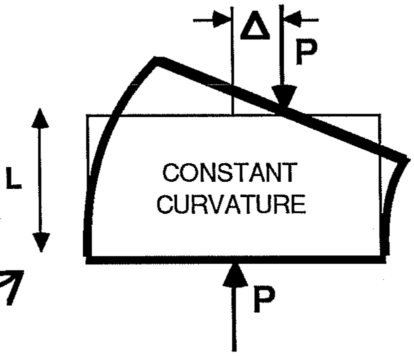


Assumptions:

1. Constant shear and axial force over length of column

2. Constant curvature over length of element

3. Moment-curvature-axial load relationship is defined for column section



Kinematics:

$$\Delta_{i+1} = \Delta_i + L(\Theta_i + L\phi_i)$$

$$\Theta_{i+1} = \Theta_i + L\phi_i$$

Equilibrium:

$$M_{i+1} = M_{end} - VLi - P\sum^{i}\Delta_j$$

Figure 5.9

between the method he describes and the implementation described below. The second method was developed by the author and is deflection-controlled. Each method has advantages and disadvantages which are discussed in detail below. In this section, only general descriptions of each method are given, with more specific algorithms given in the next section on the implementation of material, section, and column models in the computer program OPUS.

5.6.1. Load-control method. This method uses an iterative approach to solving for the lateral load-lateral displacement envelope. A flowchart of this method is shown in Fig. 5.10. The specified loading program is first divided into a number of equal steps and the column is divided into a number of equal segments. Then, for each load step an estimate of the column end moment is made and equilibrium is satisfied individually for each of the segments. Having satisfied local equilibrium for each segment, global equilibrium is checked. If global equilibrium is not satisfied, the initial estimate of the end moment is revised and the procedure repeated. Once global equilibrium is satisfied, the end deflection is recorded and the entire procedure is repeated for another load step. In this way, a lateral load-lateral deflection curve can be generated, one load step at a time.

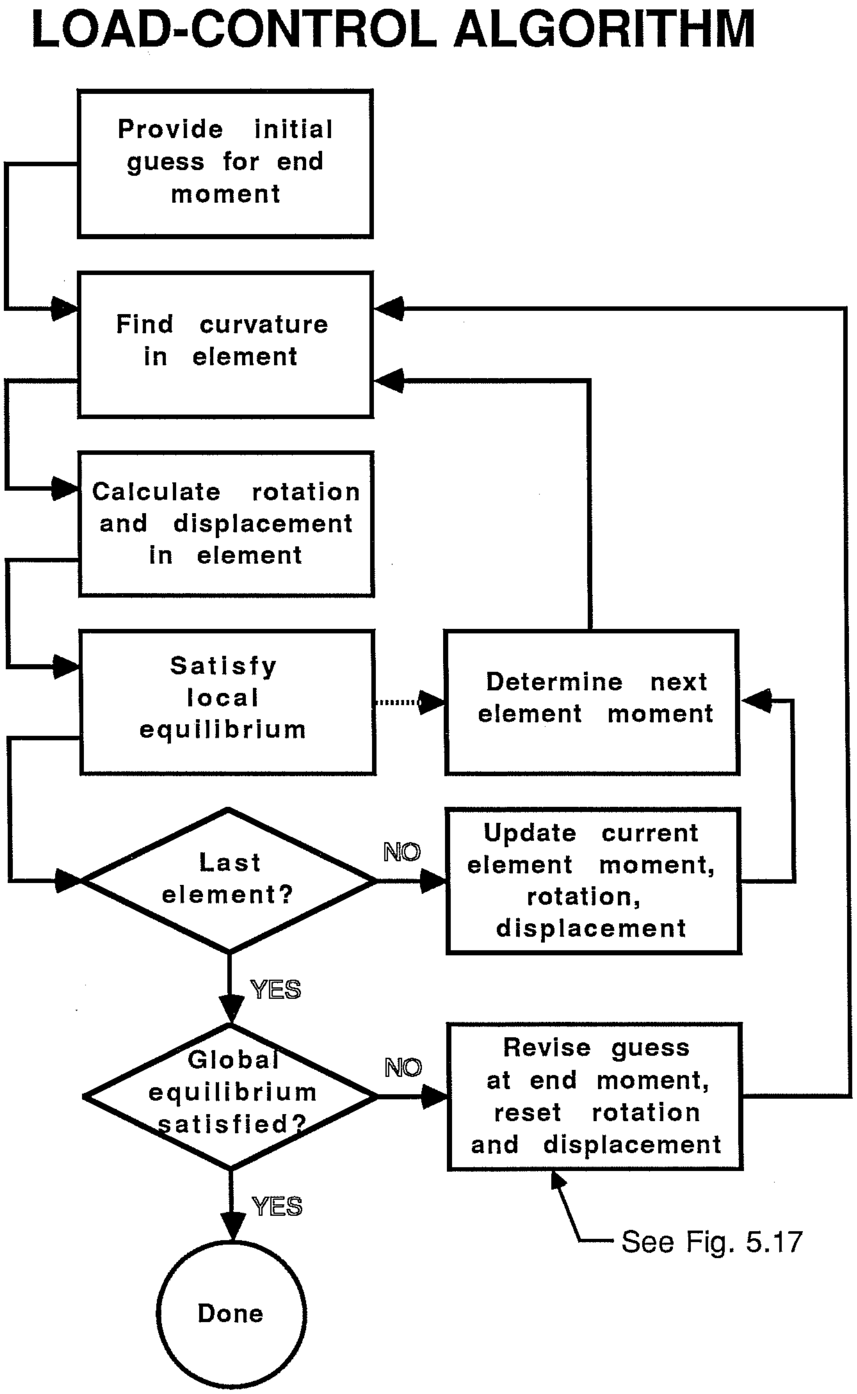

LOAD-CONTROL ALGORITHM
Provide initial guess for end moment
Find curvature in element
Calculate rotation and displacement in element
Satisfy local equilibrium
Determine next element moment
Last element?
NO
Update current element moment, rotation, displacement
YES
Global equilibrium satisfied?
NO
Revise guess at end moment, reset rotation and displacement
YES
Done
See Fig. 5.17

Figure 5.10

There are several benefits to determining the response with this method. First, calculations are rather fast. With thirty load steps and sixty column segments, an envelope was generated on an IBM PC-AT in under one minute. Also, the method is conceptually simple; only equilibrium need be satisfied to obtain an envelope.

Despite these benefits, there was a severe limitation of the load-control method: only ascending portions of the failure envelope, that is portions between positive and negative ultimate load, could be obtained. This was a significant restraint for this model because the purpose of the model was to study inelastic behavior, much of which often occurs at displacements higher than the displacement at ultimate load. Consequently, much of the response this study was designed to investigate could not be generated using a load-control method. For this reason, a deformation-control model was developed.

5.6.2. Deformation-control method. The deformation-control model, which was designed by the author, used the same basic approach as the load-control model described above. However, changes were made to allow for hinging, which would be expected to occur at deflections above the deflection at ultimate load.

A flowchart of the deformation-control method is shown in Fig. 5.11. The "loading" (lateral displacements) is divided

DEFORMATION-CONTROL ALGORITHM

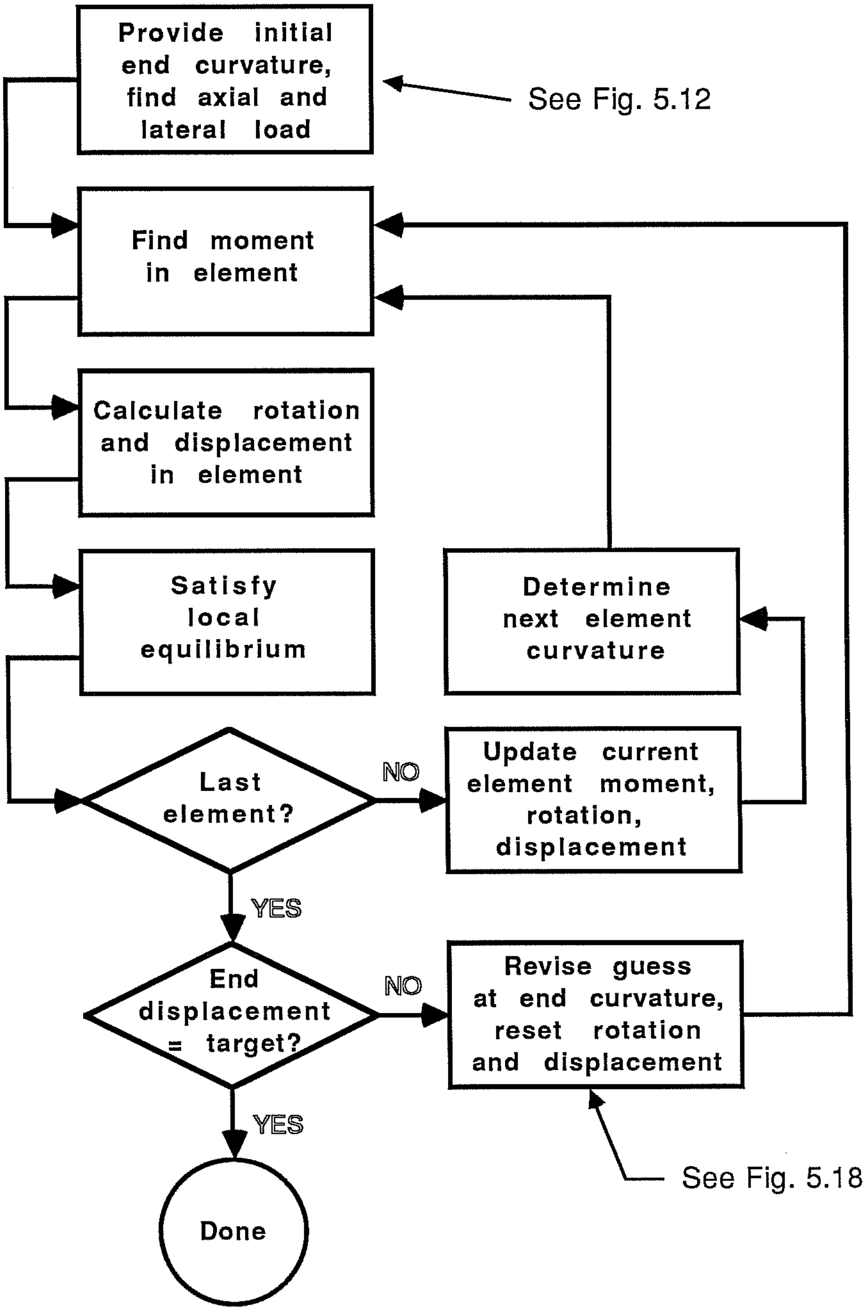

Provide initial end curvature, find axial and lateral load
See Fig. 5.12
Find moment in element
Calculate rotation and displacement in element
Satisfy local equilibrium
Determine next element curvature
Last element?
NO
Update current element moment, rotation, displacement
YES
End displacement = target?
NO
Revise guess at end curvature, reset rotation and displacement
YES
See Fig. 5.18
Done

Figure 5.11

into load steps and the column into segments, just as in the load-control method described above. Instead of estimating an initial end moment, however, an estimate of the initial end curvature is made. From end curvature and end conditions, axial load, end moments, and lateral load can be determined by satisfying global equilibrium. This determination was performed using the iterative process shown in Fig. 5.12. Using an initial estimate of axial load and the relative end conditions, end moments were obtained from the axial load-moment-curvature relationship of the section. Using the end moments, axial load, and lateral deflection, lateral loads were calculated from equilibrium. A new axial load was calculated from the lateral load and the process repeated until the difference between estimated and calculated axial loads was within acceptable tolerances.

Having satisfied global equilibrium, local equilibrium is then satisfied for each segment of the column. The method used to satisfy local equilibrium differs from the similar process in the load-control method in that the curvature for each segment must be chosen more carefully. In the load-control method, restricting axial load-moment combinations to those between positive and negative ultimate makes the curvature function single-valued. In the deformation-control method,

AXIAL AND LATERAL LOAD DETERMINATION FOR DEFORMATION CONTROL MODEL

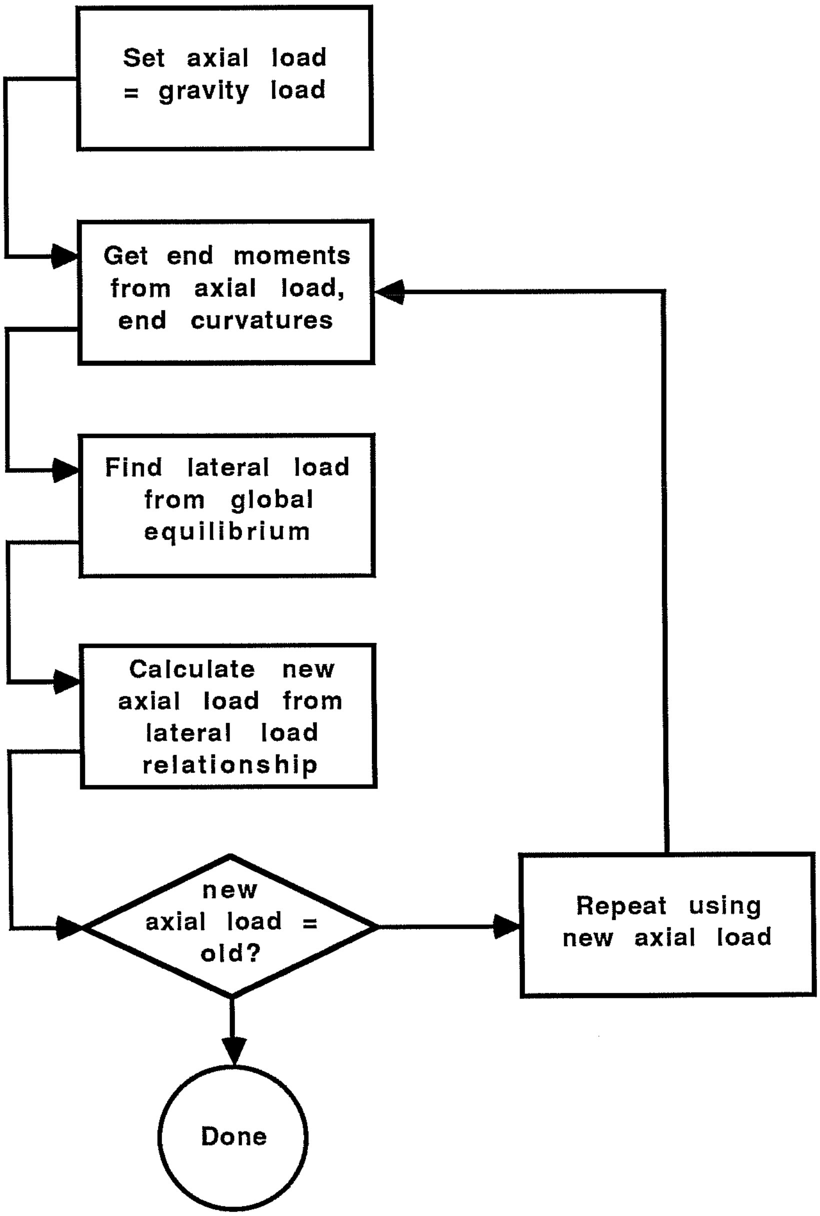

Set axial load = gravity load
Get end moments from axial load, end curvatures
Find lateral load from global equilibrium
Calculate new axial load from lateral load relationship
new axial load = old?
Repeat using new axial load
Done

Figure 5.12

however, two curvatures may exist for a given axial load-moment combination, so an additional restriction must be made to find a unique curvature for an axial load-moment combination. Therefore, it is assumed that the curvature along the column is either strictly increasing or strictly decreasing. Figure 5.13 shows graphically the difference between the search for curvature solutions in the load and deformation-control methods.

After satisfying local equilibrium for each segment, the end deflection is compared to the target deflection. If they are not equal, within a user-specified tolerance, a new estimate of end curvature is made and the above procedure repeated. If they are equal, a check is performed to see if hinging has occurred. This is done by comparing the curvature of the end segment with that of the adjacent segment. If the difference is not significant, hinging was not necessary to achieve the target displacement and the axial load- lateral load-lateral displacement combination is recorded and the method is repeated for another load step. If hinging did occur, however, some adjustments are required.

Hinging, for purposes of this analysis, is treated as a concentration of large deformation over a short length (called the hinge length) of a structural member. The smaller the hinge length, the larger the deformation required to obtain a given displacement. To model hinging it is necessary to know the hinge

SEARCH FOR CURVATURE VALUES

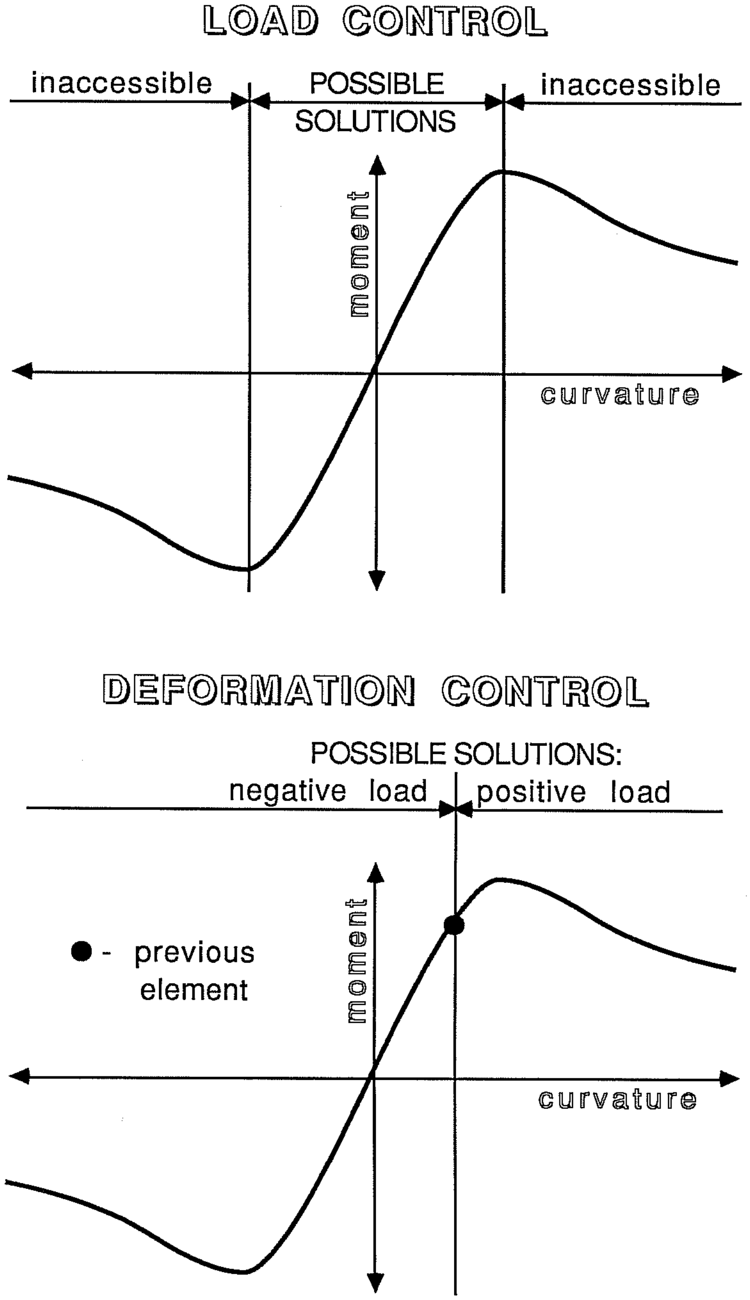


Figure 5.13

length so that the deformation is not unrealistically large. Several studies have examined hinging in a variety of members, and a number of simple methods for calculating hinge lengths have been proposed [35,36]. Generally, these methods proposed using hinge lengths which were approximately equal to the section depth. It is not clear, however, whether these methods are applicable to the current study because they were designed for use on beams, not columns. Therefore, observations of the experimental program described in Chapters 3 and 4 were also used to arrive at an estimate of hinge length. Based on cracking and spalling of test specimens, an estimate of a hinge length of ten inches was chosen. Hence, hinge length was assumed to be equal to the total section depth for this model.

With this estimate of hinge length, the entire deformation-control method described above was repeated, with one difference. When satisfying local equilibrium, curvature of segments within the hinge was assumed to be equal to the end curvature. Thus, the hinge curvature, instead of being concentrated in one column segment, was distributed over a number of segments. Once global and local equilibrium were satisfied and end displacement was equal to the target displacement (within user-specified tolerance), the axial load-lateral load-lateral section stiffnesses, reinforcement ratios, and the moment-interaction diagram.

Column loads are entered in procedure LOADING. Function keys are used to load and save loading data on disk files, clear current load entries, graph load paths, and switch between load and deformation-controlled analysis models. Arrow keys are used to select a data item to modify, and ENTER initiates the modification of the current item. Three loading modes are provided for both axial and lateral loads: constant, linear, and multi-linear. Multi-linear mode can be used to define a cyclic load or an approximation of a nonlinear load path. For axial loads, the form which the linear mode takes is different for load and deformation-controlled models. When operating under load control, linear axial loads are specified by the initial and final values of the load. Under deformation control, however, linear axial loads are specified by a gravity load and a ratio of the change in axial load to the change in lateral load. Axial load is then determined from lateral load, which is determined from lateral displacement. This is discussed in greater detail below in the section on the deformation-controlled model.

5.7.3. Material models. Material models are implemented as functions in OPUS. The function CONCRETE calculates either the confined or unconfined concrete stress for a given strain. This function is later used by the function STRESS_BLOCK to determine the concrete compression force in the column section. Steel stresses are calculated from steel strains in the function STEEL. These stresses are not pure steel stresses, but rather steel stress minus concrete stress. This is done so that the concrete compression zone can be treated as a complete rectangle without having to determine where "holes" in the compression zone occur due to the reinforcement.

5.7.4. Section model. The section model is calculated, displayed, and modified through procedure SECTION. Concrete and steel contributions to section behavior are determined separately then superimposed. This greatly simplifies the section model because each material can be handled in the optimum fashion.

Concrete contribution to the section is calculated in the function STRESS_BLOCK. STRESS_BLOCK uses the extreme fiber strain and neutral axis to obtain the strain distribution across the column section. The stress zone is then divided into a user-specified number of slices, and the strain in each slice is calculated from the strain distribution. Using the location of the slice in the section, the percentage of the slice which is confined is determined. Slices near the ends of the section are unconfined, and interior slices are partially confined. The strain in the slice is then used by function CONCRETE to determine the stress in confined and unconfined parts of the slice. Total force in the slice is calculated from slice thickness, slice width, confined concrete stress, unconfined concrete stress, and the percentage of the slice which is confined. This process is repeated for each slice, and the sum of the slice forces is the force in the compression zone. The centroid of the compression zone can also be determined by summing the product of the slice force and slice depth for all slices and dividing by the force in the compression zone.

Steel contribution in the section can be determined in a much simpler way than concrete contribution. First, strain in a steel layer is calculated from the strain distribution across the section. Then, using function STEEL, the stress in the layer is obtained, the concrete stress is subtracted, then this net stress multiplied by the area of steel to give the force in the layer.

The central procedure of OPUS' section model is GET_FORCES, which calculates axial load-moment-curvature combinations from material properties. There are two different modes used to obtain these combinations, one used by procedure MOMENT_INTERACTION to generate the failure envelope and one used by CALC_SECTION to generate the axial load-moment-curvature relationship. The general approach used by both modes is the same, and is flowcharted in Fig. 5.14. The modes differ in the constraints used in the search for solutions, a difference which is discussed in greater detail below.

The key to GET_FORCES is the technique used to adjust the neutral axis after each calculation. It is important that the technique used always converge on the solution, and that it do so as rapidly as possible. The solution technique used is based on the Newton-Raphson method for finding roots of nonlinear equations and is flowcharted in Fig. 5.15. The characteristic equation of the section is assumed to be

$$g(kd) = P(kd) - P_{target}$$

where

$P(kd)$ = axial load as a function of neutral axis depth

P_{target} = target axial load

kd = depth of neutral axis

So, the object of the solution technique is to find the value of kd such that g(kd) = 0. This is accomplished by revising kd each iteration according to the Newton-Raphson equation

$$x_{i+1} = x_i - \frac{g\ (x_i)}{g'(x_i)}$$

where

$g(x)$ = characteristic equation

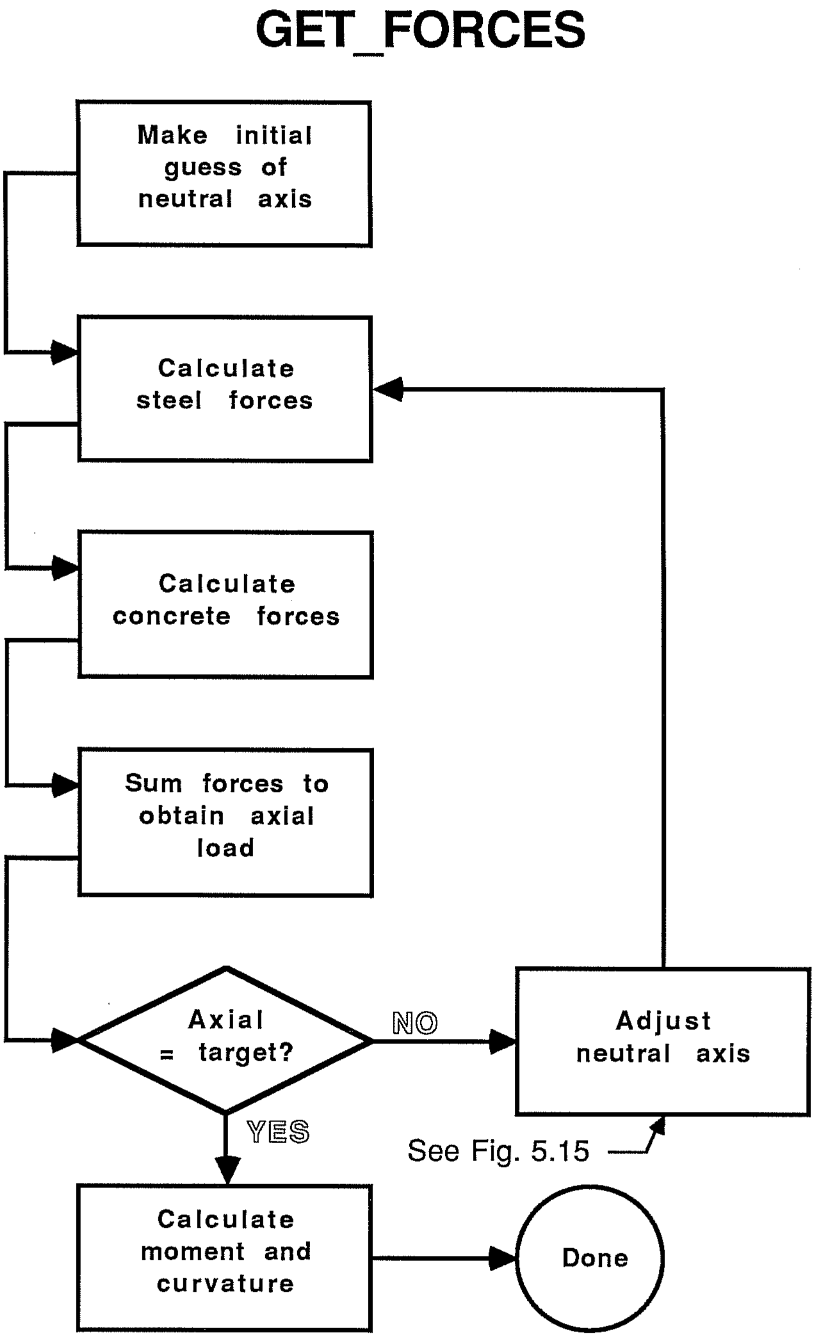

GET_FORCES
Make initial guess of neutral axis
Calculate steel forces
Calculate concrete forces
Sum forces to obtain axial load
Axial = target?
NO
Adjust neutral axis
YES
See Fig. 5.15
Calculate moment and curvature
Done

Figure 5.14

NEWTON-RAPHSON METHOD

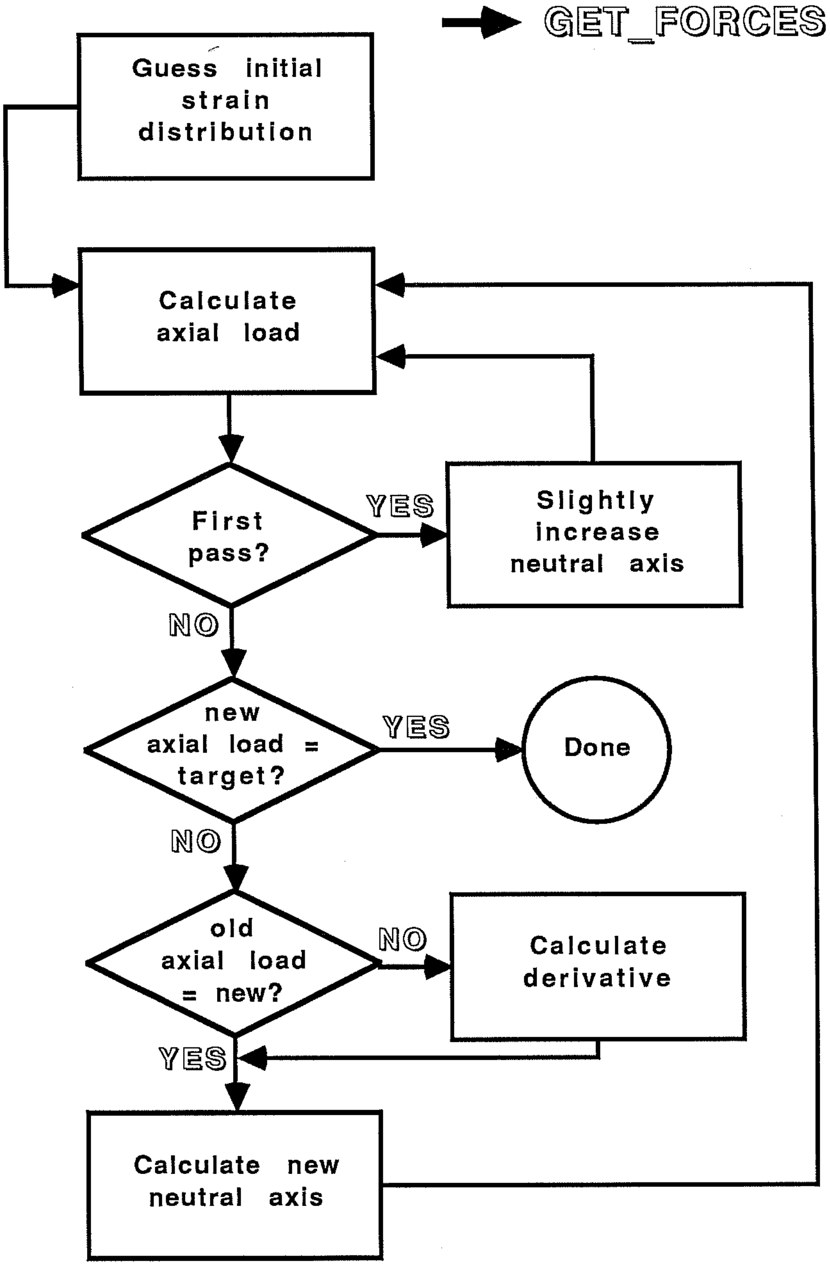


Figure 5.15

Because g(kd) does not have a simple closed-form expression capable of differentiation, the derivative must be determined numerically. So, the derivative is estimated using the first-order approximation

$$g'(x_i) = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}$$

This method works well for the vast majority of situations, but there are a few cases where it does not. The most important of these is where there is an instantaneous change in axial load, such as occurs at layers of reinforcement. In these cases, the derivative of g(kd) can get so large that it cannot be represented by the computer, resulting in premature termination of the program. Therefore, in each iteration axial load was checked for change. If no change occurred, the derivative was not changed, thus transforming the method from a second-order Newton-Raphson method to a first-order constant-stiffness method. This procedure worked well, and did not significantly affect the speed of convergence.

Function GET_FORCES, in the most general terms, searches for axial load-moment-curvature combinations by varying the strain distribution across the section. Because the strain distribution is assumed to be linear, two independent variables are required to define the distribution. Solving for two independent variables, however, is much more complex than solving for one, so some constraint must be placed on the strain distribution in order to reduce the number of independent variables. Each mode of GET_FORCES places a constraint on the strain distribution which depends on what is known about the distribution and what end result is desired.

Mode one calculates axial load-moment-curvature combinations which lie on the axial load-moment interaction diagram. The constraint on the search for solutions is the extreme compressive fiber strain be equal to the ultimate concrete strain. Varying the neutral axis varies the axial load, moment, and curvature while assuring that combinations calculated lie on the interaction diagram.

Mode two calculates axial load-moment-curvature combinations for a specified curvature. This mode is used to generate the axial load-moment-curvature relationship for the column section. The extreme fiber strain is determined for each iteration by multiplying the desired curvature by the neutral axis for that iteration. Varying the neutral axis varies the axial load, moment, and extreme fiber strain while maintaining a constant curvature.

5.7.5. Load-controlled column model. Both the load-controlled and deformation-controlled models are executed by procedure ANALYZE. This procedure determines which model to use, then proceeds with the calculation.

Procedure GET_DEFLECTION determines the lateral deflection which occurs under a given axial and lateral load. The flowchart shown in Fig. 5.16 gives a general overview of the model, and Fig. 5.17 is the flowchart of the modified Newton-Raphson method which was used to converge on the deflection. The Newton-Raphson method used in GET_DEFLECTION differs from the method used in GET_FORCES which was described in Section 5.8.4 only in the characteristic equation used. The equation used in GET_DEFLECTION is

$$g(M_o) = H(M_o) - H_{target}$$

where

$H(M_o)$ = lateral load as a function of end moment

H_{target} = target lateral load

M_o = end moment

Procedure GET_DEFLECTION relies heavily on the procedure GET_ELEM_CURV, which determines the curvature for an axial load-moment combination from the axial load-moment-curvature relationship calculated by the section model. Linear interpolation is used to extract points from the model which were not explicitly calculated by CALC_SECTION. Curvature values obtained by GET_ELEM_CURV always lie between the curvatures

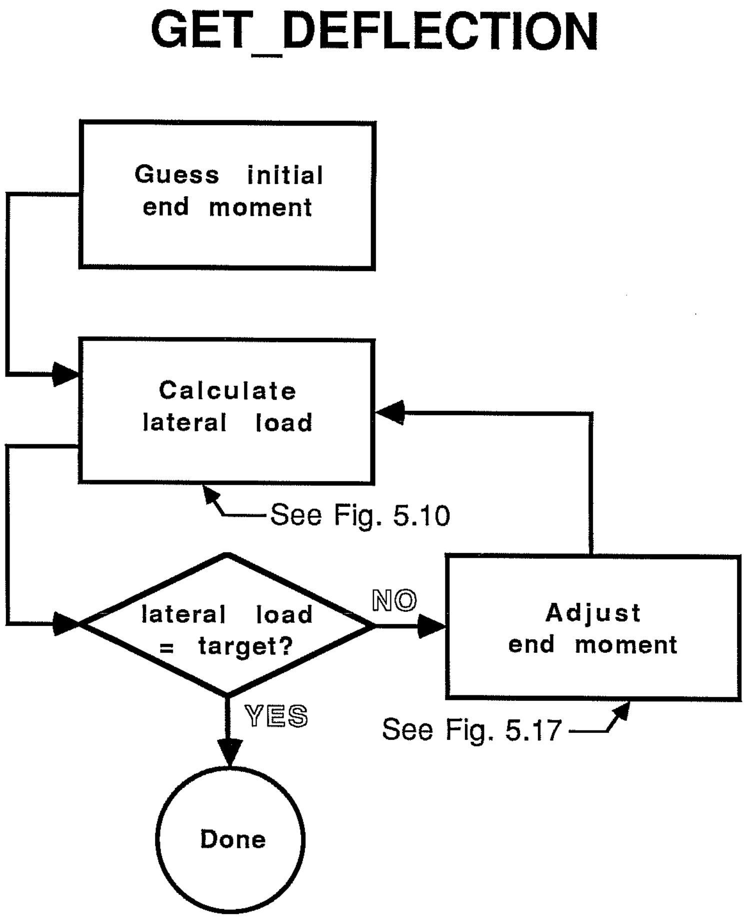

GET_DEFLECTION
Guess initial end moment
Calculate lateral load
See Fig. 5.10
lateral load = target?
NO
Adjust end moment
YES
See Fig. 5.17
Done

Figure 5.16

NEWTON-RAPHSON METHOD

→ GET_DEFLECTION

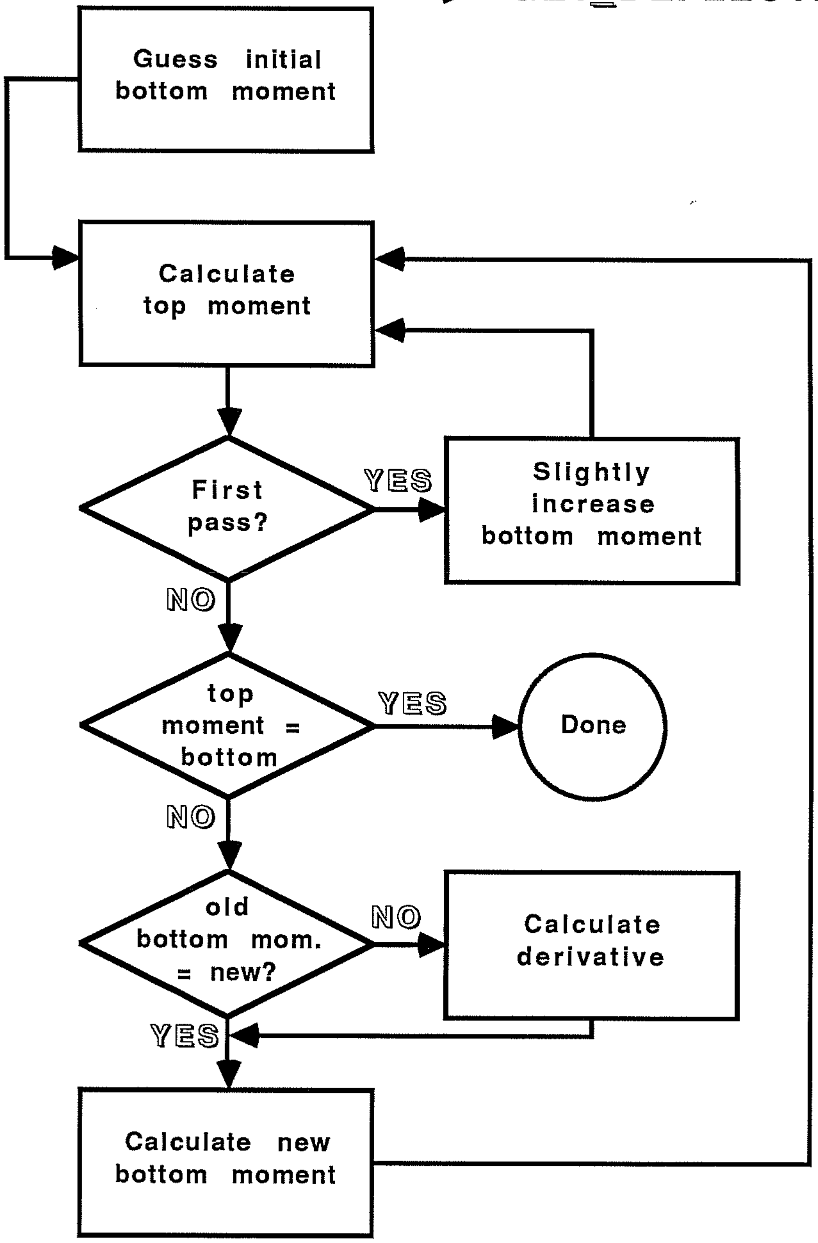


Figure 5.17

associated with maximum and minimum moments for a given axial load.

5.7.6. Deformation-controlled column model. The heart of the deformation-controlled model is procedure GET_LAT_LOAD, which calculates the lateral load, given a deflection and the relationship between axial and lateral load. The flowchart shown in Fig. 5.18 gives a general overview of the model, and Fig. 5.19 is the flowchart for the modified Newton-Raphson method which was used to converge on the deflection. Like GET_DEFLECTION, the Newton-Raphson method used in GET_LAT_LOAD differs from the method used in GET_FORCES which was described in Section 5.8.4 only in the characteristic equation used. The equation used in GET_LAT_LOAD is

$$g(\phi_o) = \delta(\phi_o) - \delta_{target}$$

where

$\delta(\phi_o)$ = lateral deflection as a function at end curvature

δ_{target} = target deflection

ϕ_o = end curvature

Procedure GET_LAT_LOAD uses procedure FIND_ELEM_CURV to obtain the curvature corresponding with an axial load-moment combination. Curvatures obtained from FIND_ELEM_CURV are always greater than the curvature of the previous element when the lateral deflection is negative, and less than the previous curvature when the lateral deflection is positive. Also, the

GET_LAT_LOAD

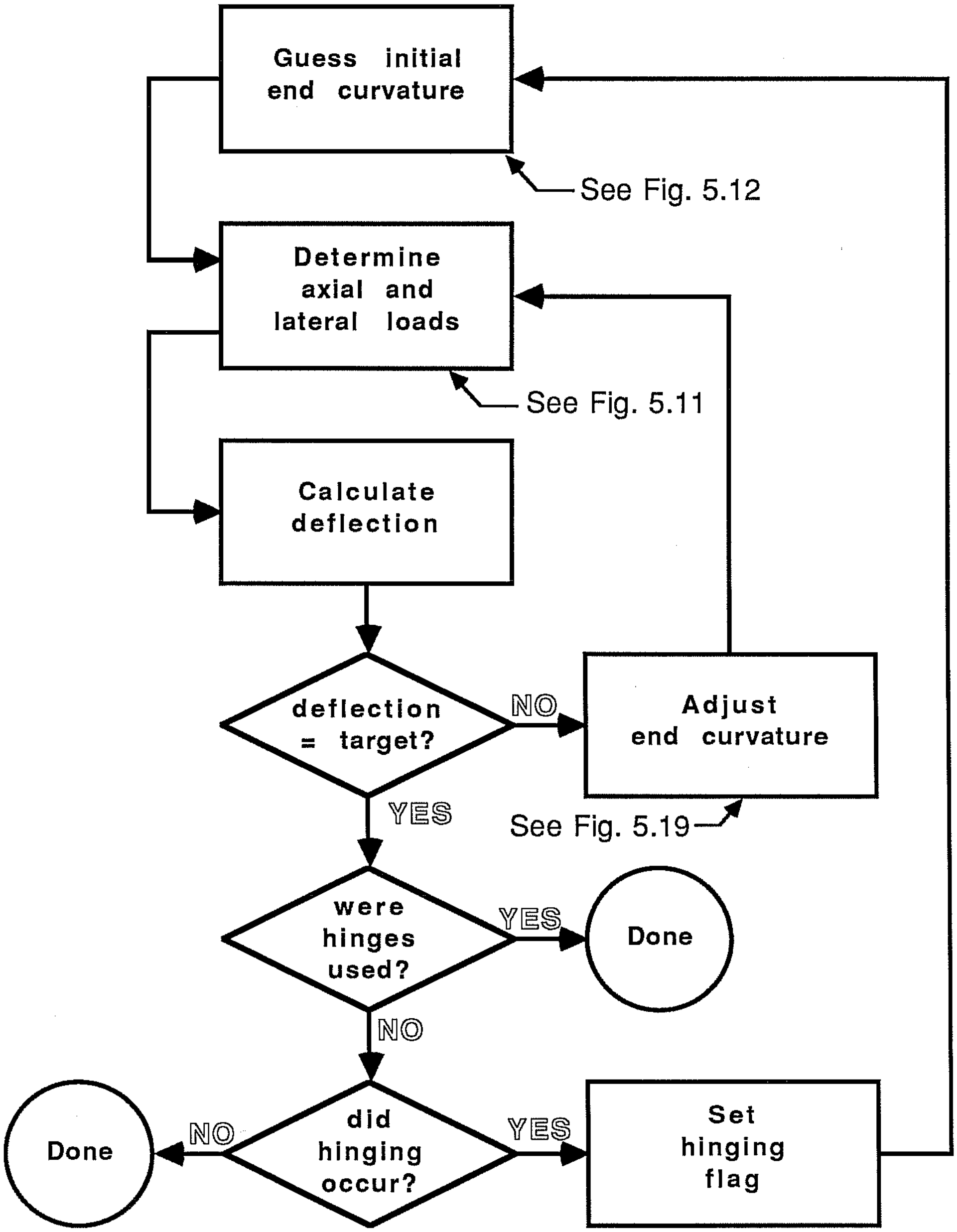


Figure 5.18

NEWTON-RAPHSON METHOD

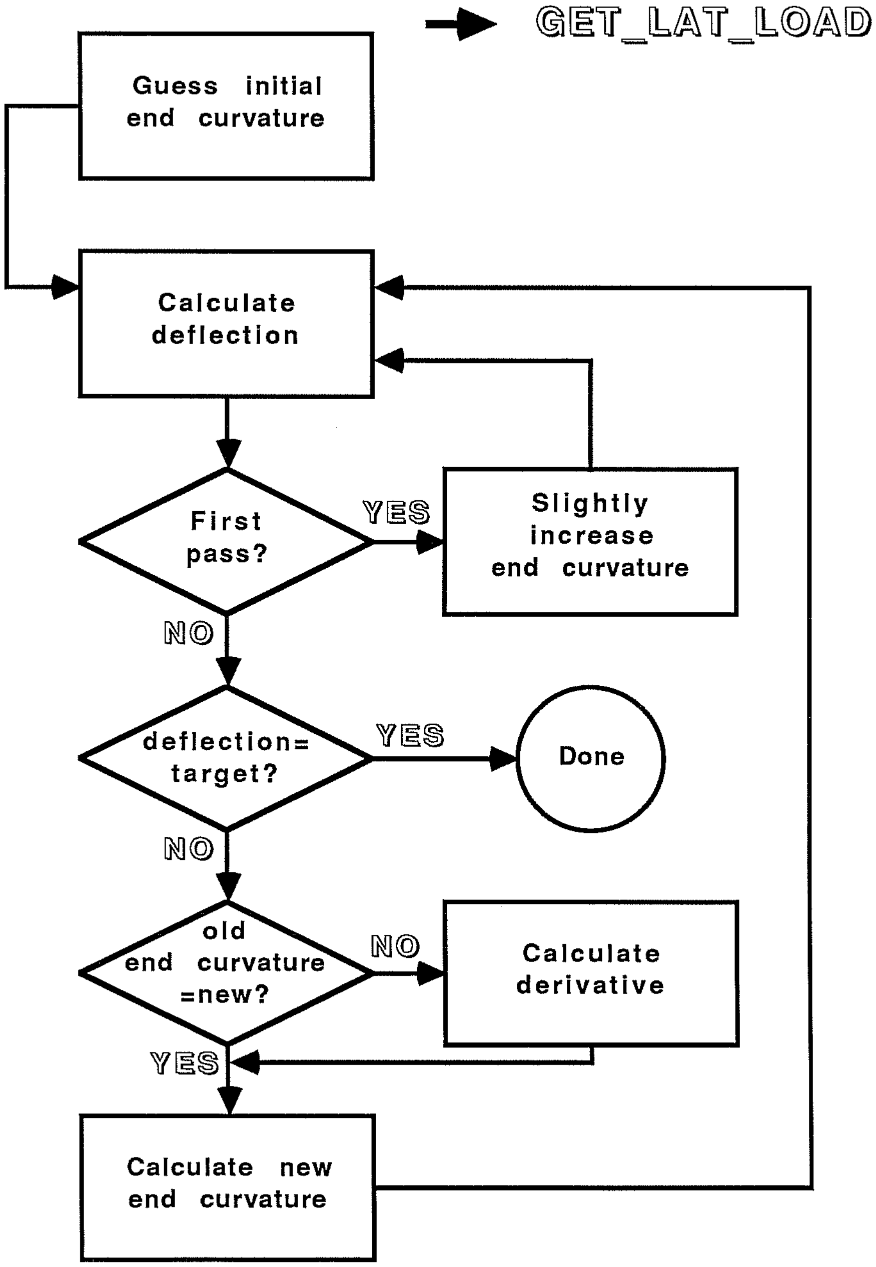


Figure 5.19

absolute value of curvatures in the hinge region are always taken to be greater than the absolute value of the curvature at maximum moment when there is a descending branch of the moment-curvature diagram.

5.7.7. *Other procedures.* The following is a list of other procedures used in OPUS, along with an explanation of the function they perform (input/output routines and procedures discussed above are not mentioned here) :

INITIALIZE	initializes variables when OPUS begins
GET_ELEM_MOM	obtains moment for an axial load-curvature combination
GET_LOAD_STEPS	divides loading into steps
GET_CURVATURE	calculates curvature limits and divides into steps for axial load-moment-curvature calculation
SHOW_GRAPH	2D and 3D graphing routines
CALC_PROP	calculates those section properties not specified by user input
RUN_COLUMN	supervisor routine which manages all OPUS functions

CHAPTER 6

RESULTS OF ANALYTICAL INVESTIGATION

6.1 Overview

In this chapter, results of the analytical investigation described in Chapter 5 are presented and discussed. Where possible, relationships generated by the OPUS column analysis program are compared with data gathered from the experimental investigation discussed in Chapters 3 and 4. Comparisons are made for section and column models, and discrepancies between measured and calculated values are discussed. Implications of analytical results with regard to design and behavior of reinforced concrete columns are also presented.

6.2 Calculated Section Response

The section model generated by OPUS produced two relationships which describe the behavior of a reinforced concrete section. The first, an axial load-moment interaction diagram, is a common design tool and probably the most widely used and understood section relationship. The second, an axial load-moment-curvature relationship, while considerably more complex than the axial load-moment interaction, gives a more complete picture of the response of a reinforced concrete section.

6.2.1. Axial load-moment interaction diagram. The axial load-moment interaction curve is the set of axial load-moment combinations at which a reinforced concrete section reaches it maximum usable compressive concrete strain. Interaction diagrams described in this section were based on a maximum usable strain of 0.003, which follows the recommendation of ACI 318-83 [22,23]. Axial load-moment interaction diagrams for specimens C - HA, C - LA - 1, and C - LA - 2 are shown in Figs. 6.1 through 6.3.

An axial load-moment interaction diagram can be used to obtain information about the behavior of a reinforced concrete section. For instance, from Fig. 6.1 it can be seen that for axial loads below the balance point, an increase in axial load results in an increase in moment capacity. Above the balance point the converse is true; increased axial load results in reduced moment capacity. In addition, behavior of the column changes from a ductile mode of failure, in which the column has the potential to resist substantial post-ultimate loads, to a brittle mode of failure which is sometimes accompanied by complete material failure of the column with virtually no post-ultimate capacity. This change in behavior above the balance point has important implications in design of exterior columns in frames. Because exterior columns will experience additional

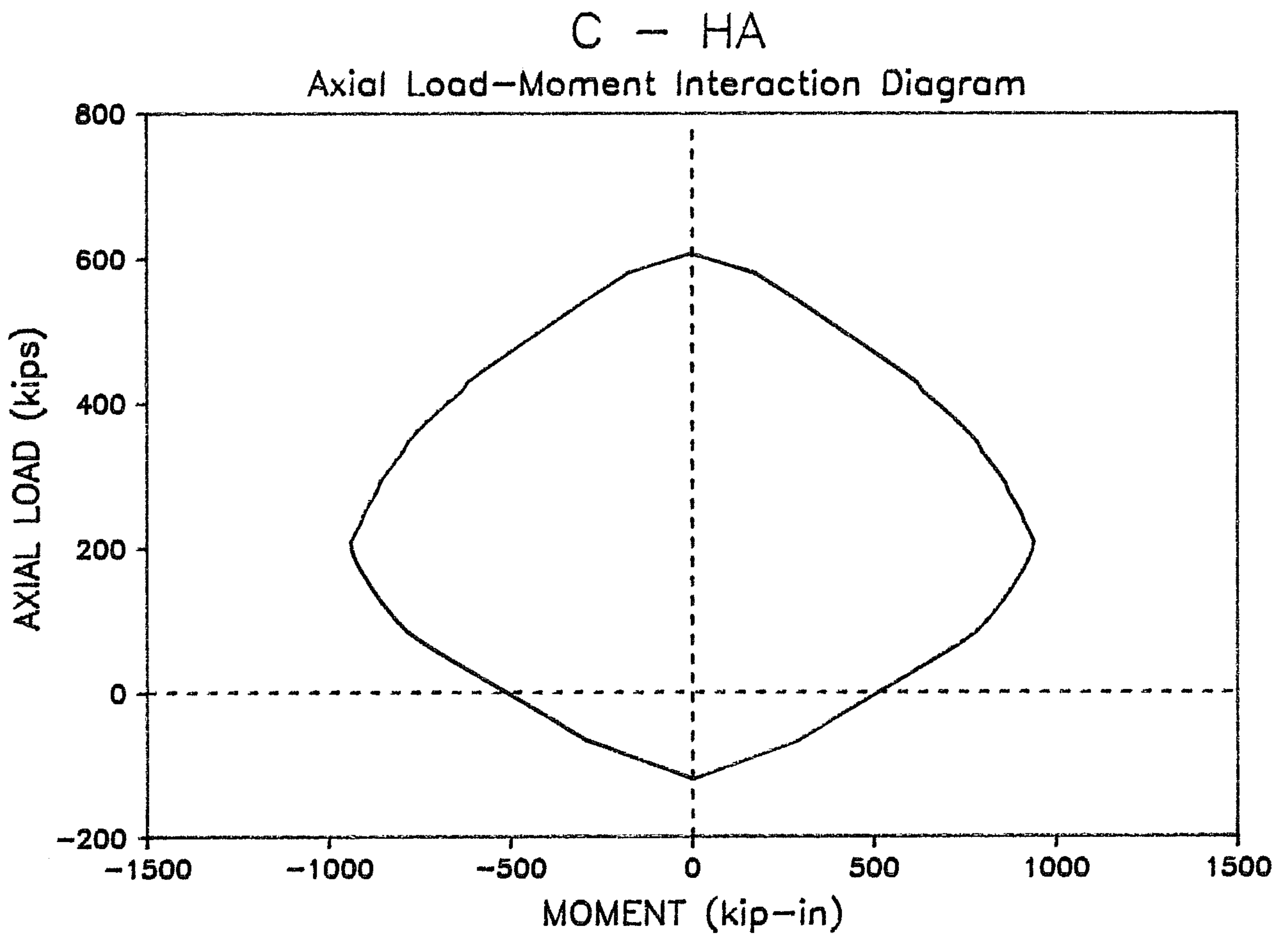

C - HA
Axial Load-Moment Interaction Diagram
AXIAL LOAD (kips)
800
600
400
200
0
-200
-1500
-1000
-500
0
500
1000
1500
MOMENT (kip-in)

Figure 6.1

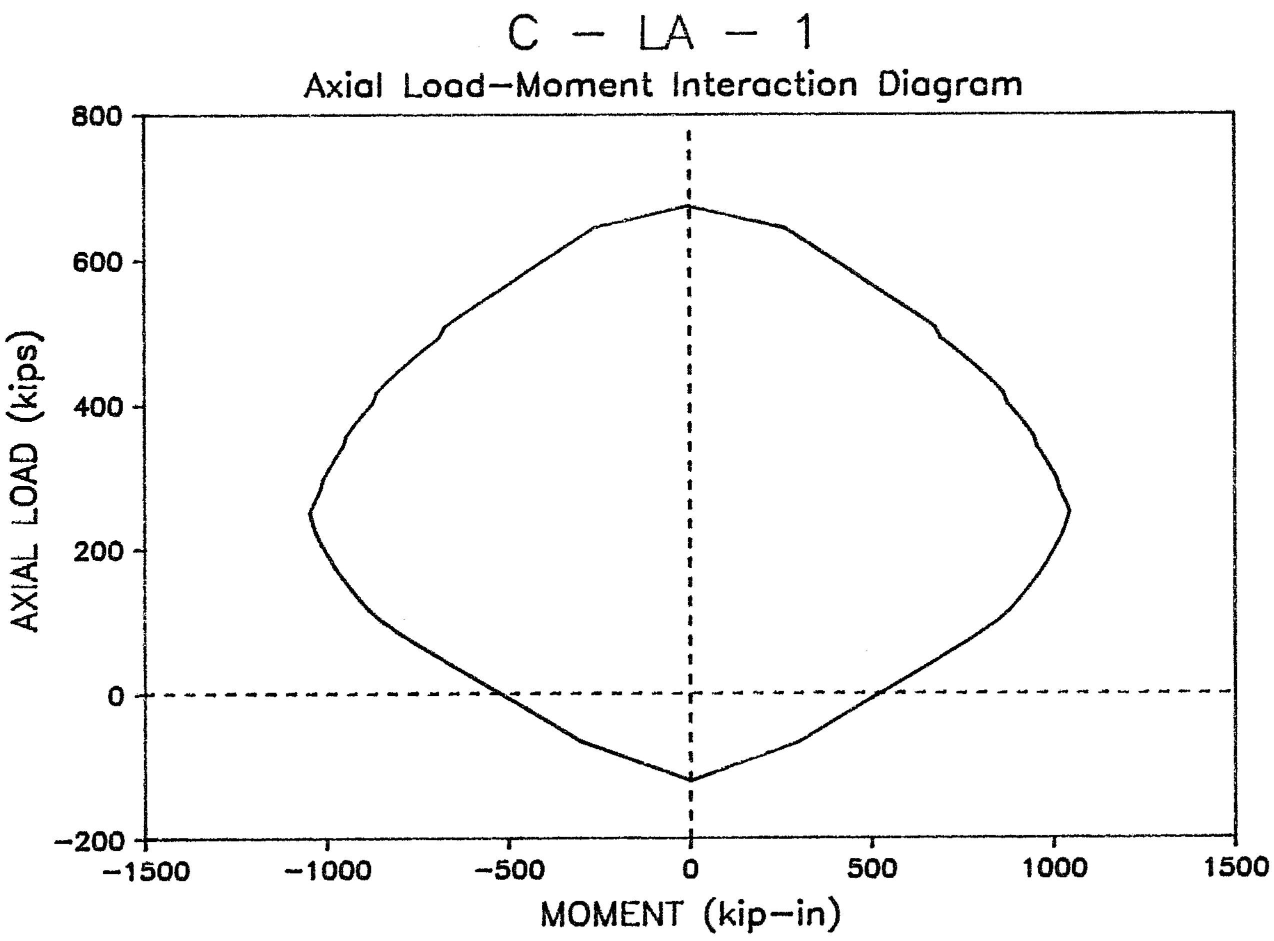

C – LA – 1
Axial Load–Moment Interaction Diagram
800
600
400
200
0
−200
AXIAL LOAD (kips)
−1500
−1000
−500
0
500
1000
1500
MOMENT (kip–in)

Figure 6.2

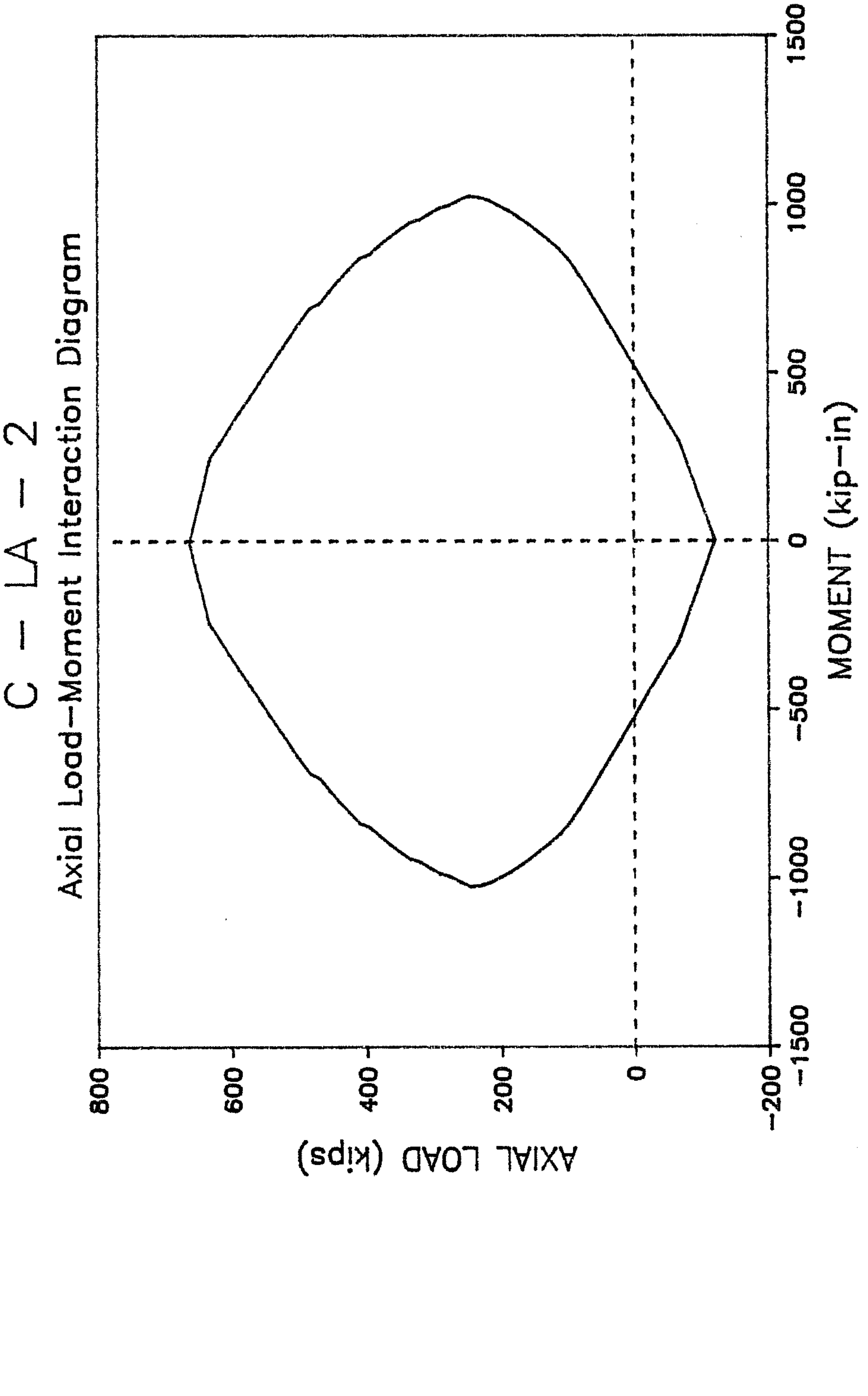

C - LA - 2
Axial Load-Moment Interaction Diagram
AXIAL LOAD (kips)
800
600
400
200
0
-200
MOMENT (kip-in)
-1500
-1000
-500
0
500
1000
1500

Figure 6.3

axial load due to frame response under lateral loads, it is important that the additional axial load be considered, especially if the resulting axial load-moment combination lies above the balance point of the column section.

6.2.2. Axial load-moment-curvature relationship. The most comprehensive description of section behavior generated by the analytical model is the axial load-moment-curvature relationship. This relationship, which can be represented as a surface in three dimensions, is the set of all reasonable monotonic axial load-moment-curvature combinations which the column may experience. Axial load is limited to values between ultimate tensile capacity and concentric axial compression at a strain of 0.003. Absolute value of curvatures at a given axial load are kept less than the product of the curvature associated with that axial load on the axial load-moment interaction diagram and a specified curvature ductility, taken arbitrarily to be twenty for this study.

Insight into the behavior of a reinforced concrete section can be gained through study of axial load-moment-curvature relationships. This can be accomplished in two-dimensions by examining "slices" through the surface, the most useful of which is a constant axial load slice. This type of slice yields a moment-curvature relationship for a given axial load. By examining changes in the moment-curvature relationship as the axial load changes, important information can be extracted about the behavior of the section. In the descriptions that follow, column specimen C - HA, which was tested in the experimental program described in Chapters 3 and 4, is used as the prototype for the analytical model.

Figure 6.4 is a moment-curvature diagram with zero axial load. The behavior is what would be expected for an under-reinforced beam, namely linear behavior until steel yields, a well-defined yield point, a slight increase in capacity after yield due to strain hardening of the reinforcing steel, and substantial ductility. All of these characteristics are desirable for any flexural member, although a column loading with zero axial load would be somewhat unusual.

Adding a small amount of axial load (ten percent of the concentric axial load capacity) to the column would result in the moment-curvature relationship shown in Fig. 6.5. There are three important differences between this moment-curvature relationship and the moment-curvature relationship for zero axial load. First, the stiffness of the curve in Fig. 6.5 is higher because axial load causes the neutral axis to be located deeper in the section, thus increasing the amount of concrete contributing to the stiffness. Second, the moment capacity of the section decreases slightly after yielding because at high compressive

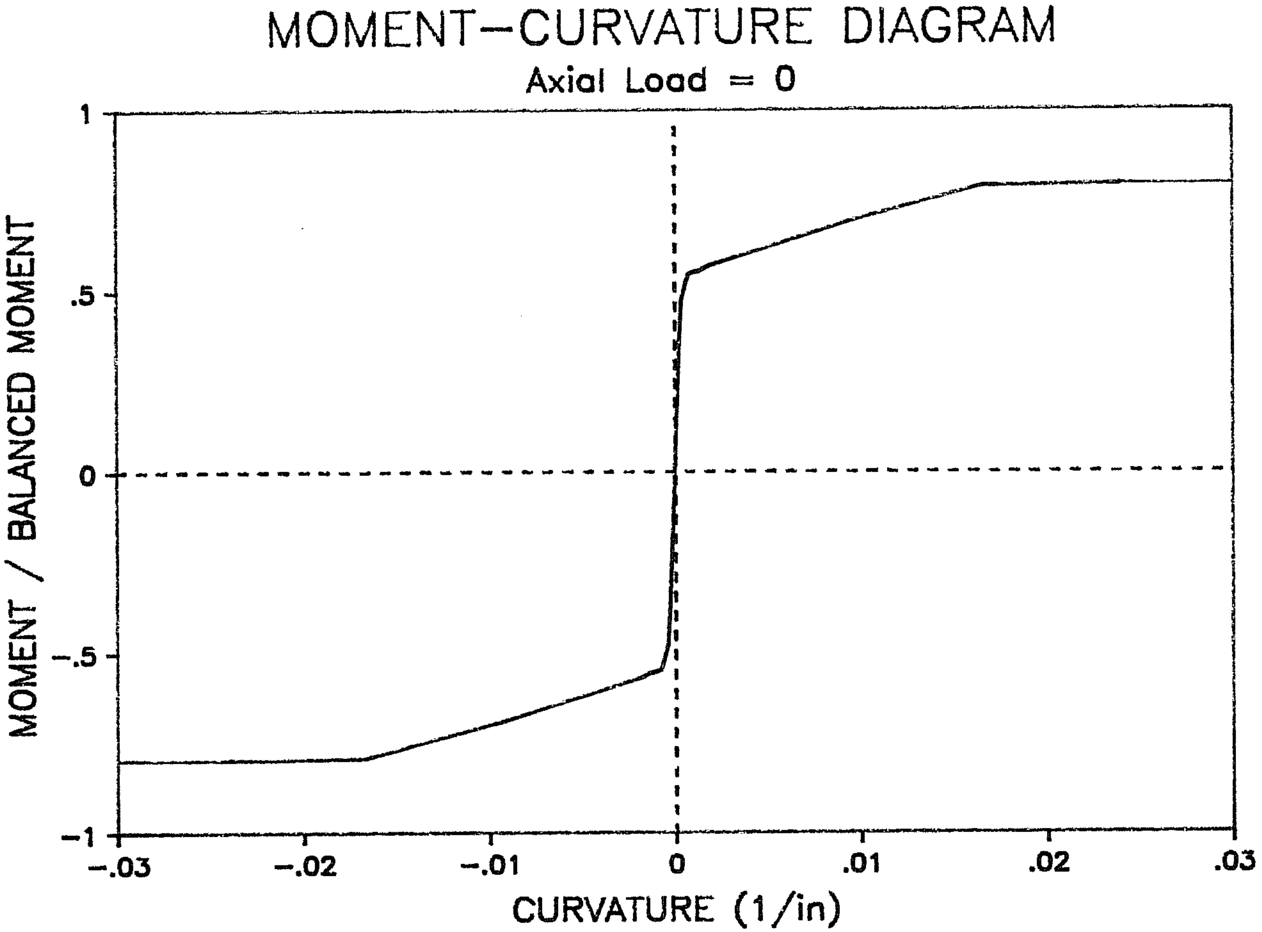

MOMENT–CURVATURE DIAGRAM
Axial Load = 0
MOMENT / BALANCED MOMENT
1
.5
0
–.5
–1
–.03
–.02
–.01
0
.01
.02
.03
CURVATURE (1/in)

Figure 6.4

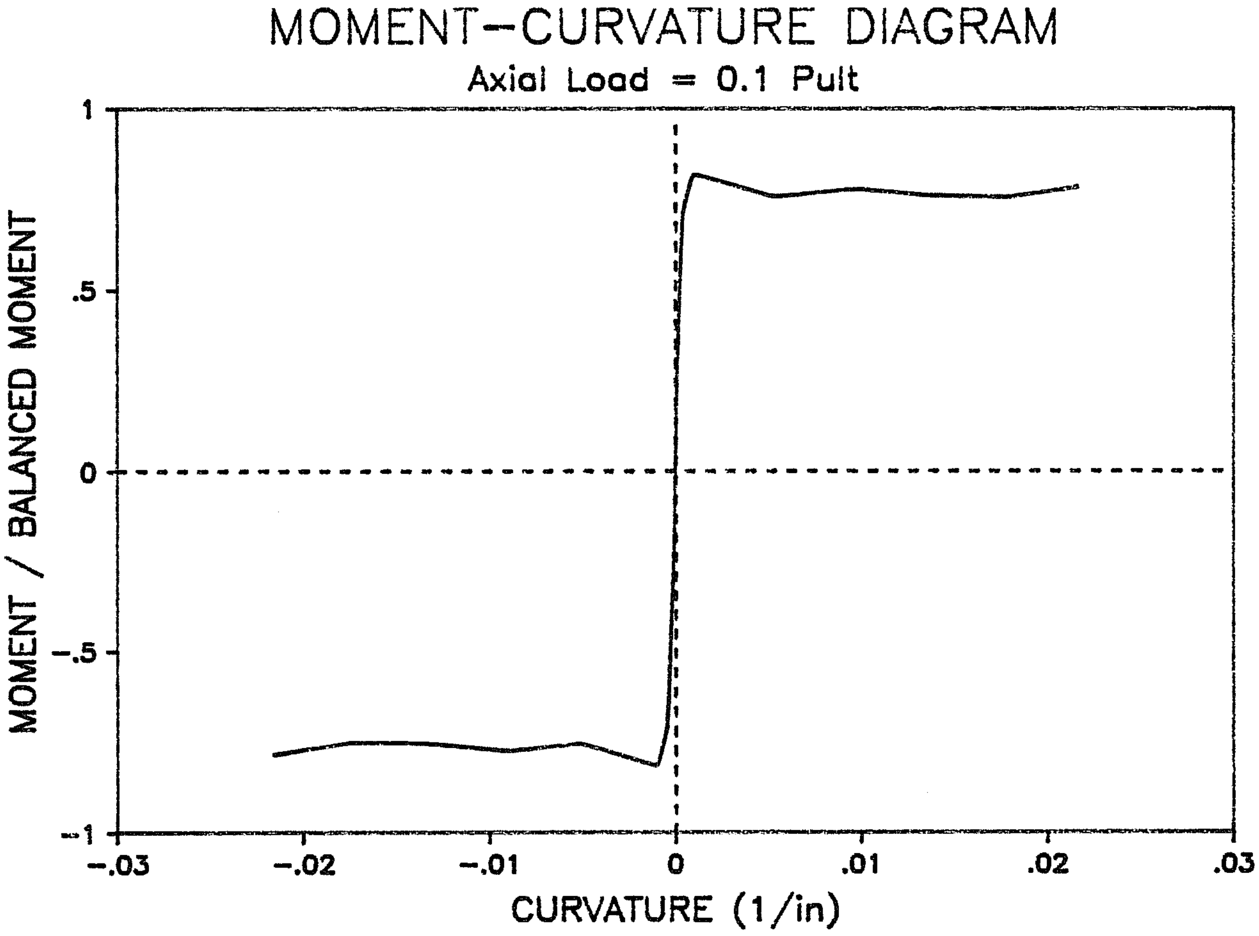

MOMENT—CURVATURE DIAGRAM
Axial Load = 0.1 Pult
MOMENT / BALANCED MOMENT
1
.5
0
−.5
−1
−.03
−.02
−.01
0
.01
.02
.03
CURVATURE (1/in)

Figure 6.5

strains concrete stresses decrease and the internal moment arm in the section is reduced. Third, the curvature at which yielding occurs increases slightly as compression is added to the section. This increase in yield curvature indicates that the range of elastic deformation is larger for this level of compression than for zero axial load.

As axial load is increased, there is a further increase in initial stiffness and the decrease in moment capacity at high curvatures becomes more pronounced as shown in Fig. 6.6. Once the maximum moment is reached, unconfined concrete loses most of its strength, representing spalling of cover concrete. Due to the effective decrease in section dimensions, moment capacity can not be regained. Moment then decreases as curvature increases beyond the curvature at maximum moment. Also, the larger axial load increases the range of elastic behavior of the column due to an increase in yield curvature.

The moment-curvature relationship at balanced axial load (the axial load-moment combination at which steel yields when concrete fails) is shown in Fig. 6.7. Notice that the decrease in capacity after maximum moment is very large for this axial load because much of the cover concrete spalls off, causing a significant decrease in the effective size of the section. This loss of section ductility due to axial load is an important consideration in the design of columns. Also, the higher

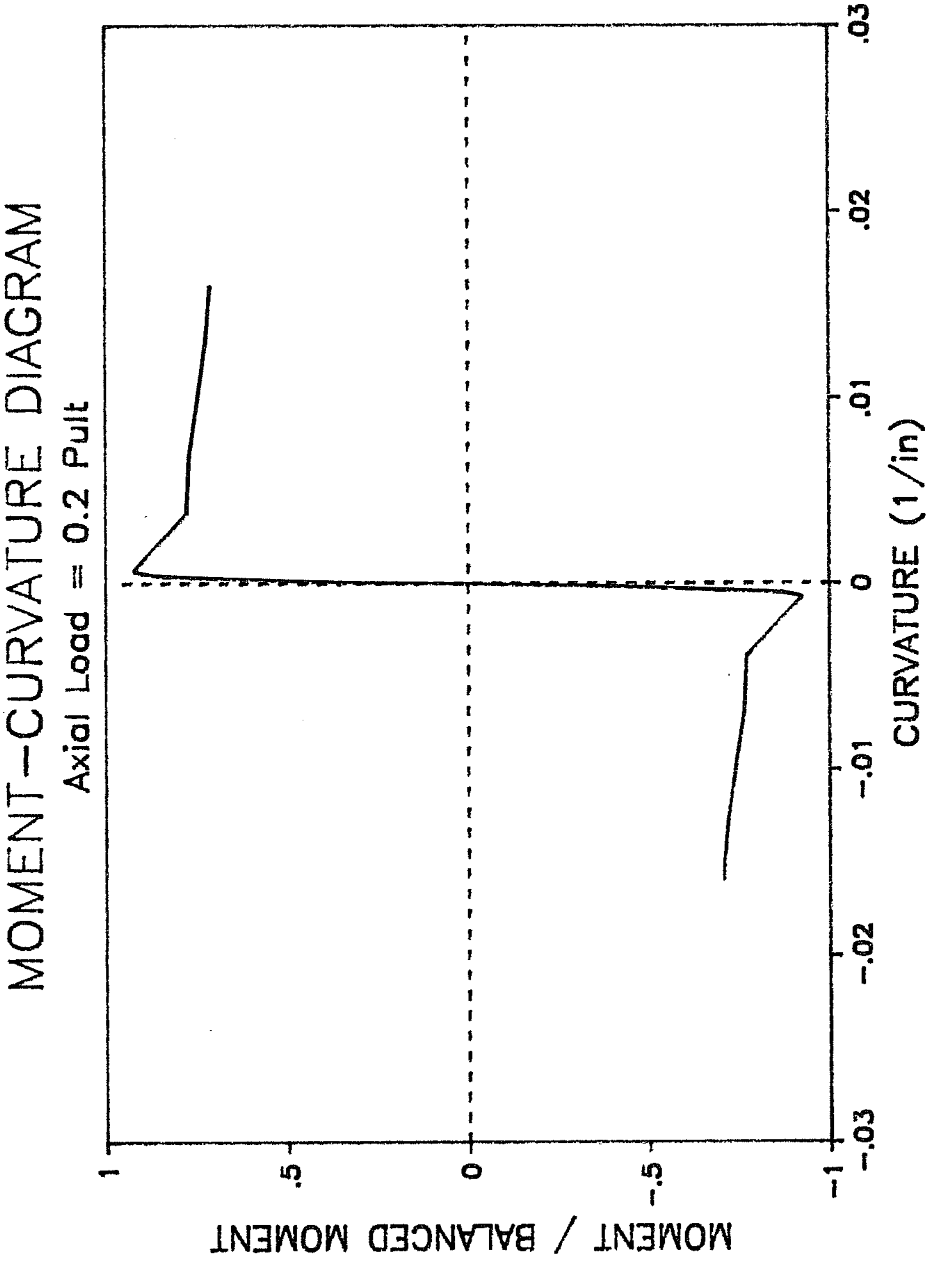

MOMENT—CURVATURE DIAGRAM
Axial Load = 0.2 Pult
MOMENT / BALANCED MOMENT
1
.5
0
-.5
-1
CURVATURE (1/in)
-.03
-.02
-.01
0
.01
.02
.03

Figure 6.6

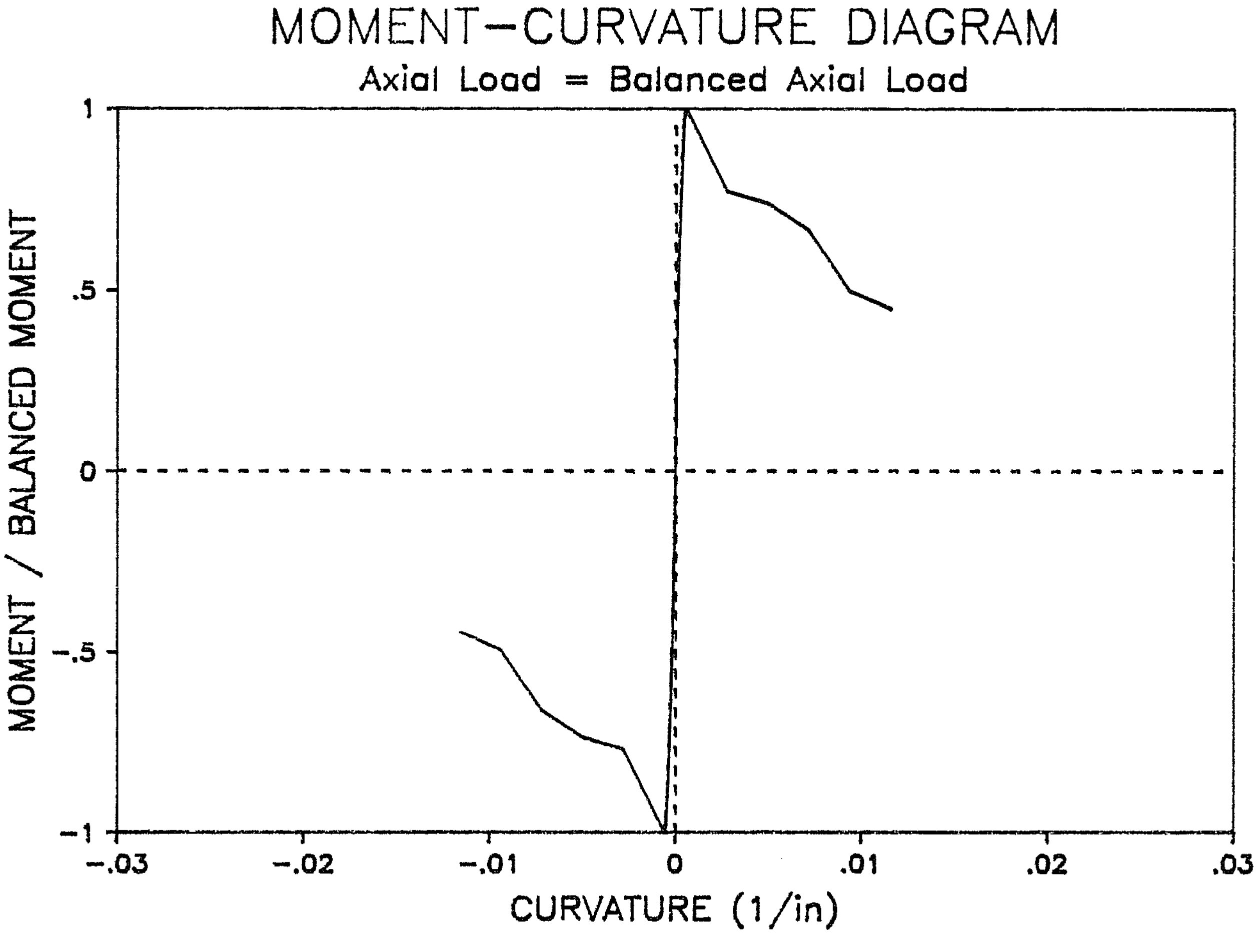

MOMENT—CURVATURE DIAGRAM
Axial Load = Balanced Axial Load
MOMENT / BALANCED MOMENT
1
.5
0
-.5
-1
-.03
-.02
-.01
0
.01
.02
.03
CURVATURE (1/in)

Figure 6.7

compressive load causes an increase in initial stiffness and yield curvature, as described previously.

As axial load becomes larger than the balanced axial load, the behavior of the column is begins to change. Figure 6.8 shows a moment-curvature diagram for a column with an axial load of one-half of the ultimate axial load. Section stiffness decreases relative to the stiffness at the balanced axial load because the compression steel in the section yields at a relatively low moment and because concrete loses stiffness at the high strains associated with this axial load. Also, curvature at maximum moment decreases because the extreme fiber strain is reached at a lower curvature, after which moment capacity decreases rapidly. This decrease in post-ultimate moment capacity is much more rapid and pronounced for axial loads above the balance point than for axial loads below the balance point. This rapid decrease occurs because additional increases in curvature cause increases in concrete compressive strain which result in lower concrete stress once strain exceeds the strain-at-ultimate-stress of the concrete. As axial load increases, the decrease in stiffness, decrease in curvature at maximum moment, and decrease in ductility become even more pronounced. This is apparent in Fig. 6.9, which is the moment-curvature diagram for 80 percent of ultimate compressive capacity of the column.

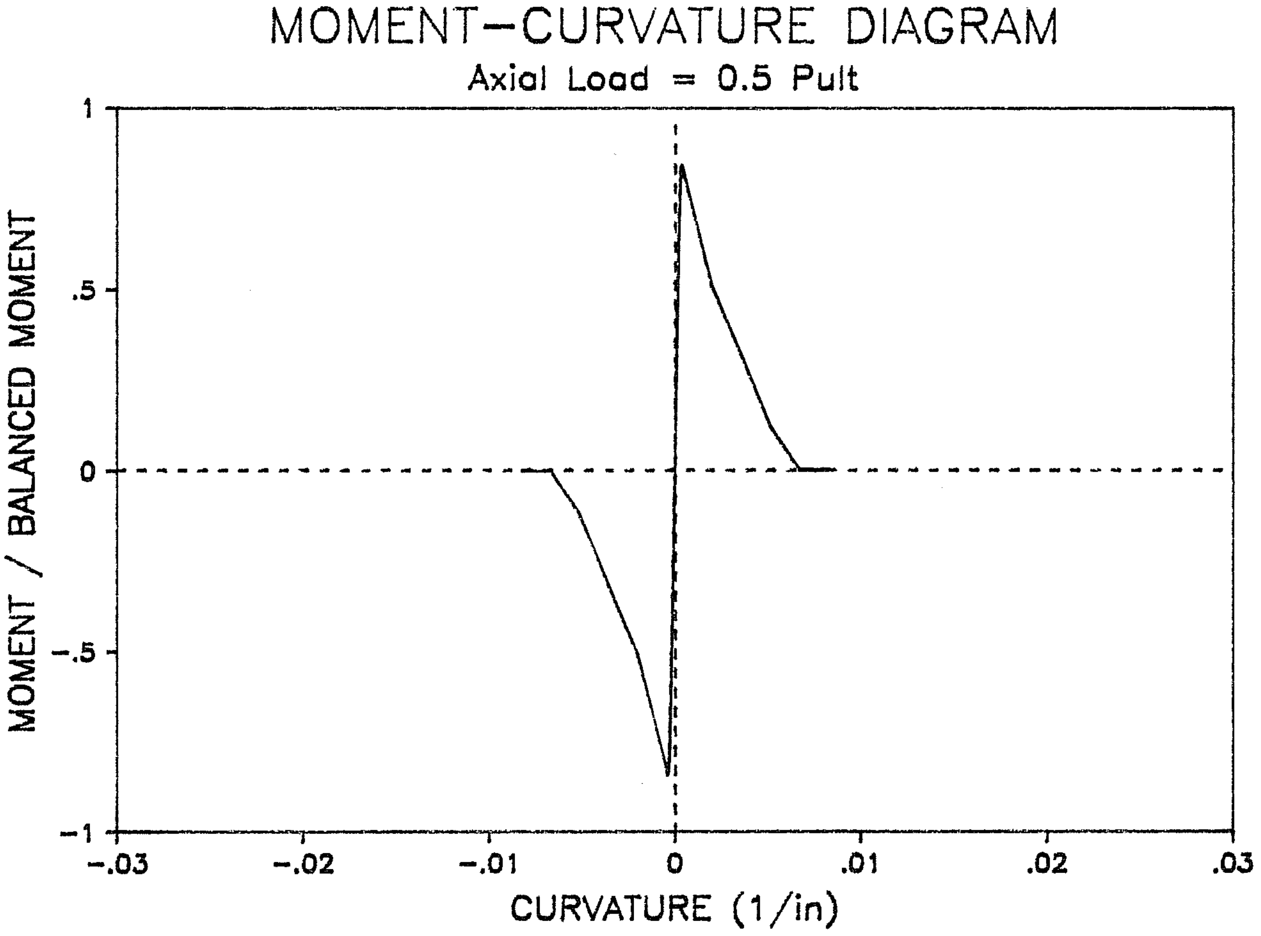

MOMENT–CURVATURE DIAGRAM
Axial Load = 0.5 Pult
MOMENT / BALANCED MOMENT
1
.5
0
–.5
–1
–.03
–.02
–.01
0
.01
.02
.03
CURVATURE (1/in)

Figure 6.8

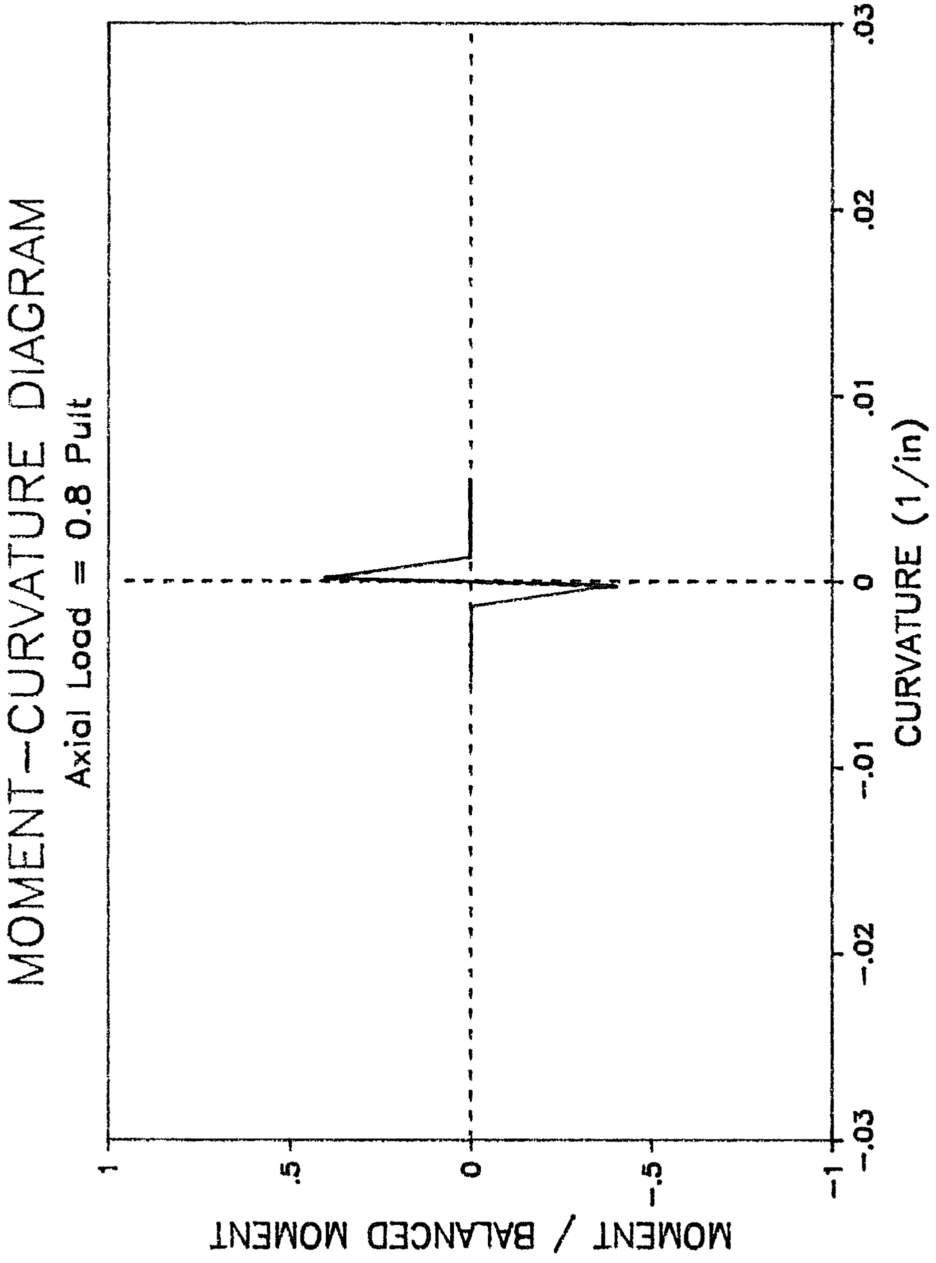

MOMENT—CURVATURE DIAGRAM
Axial Load = 0.8 Pult
MOMENT / BALANCED MOMENT
1
.5
0
-.5
-1
CURVATURE (1/in)
-.03
-.02
-.01
0
.01
.02
.03

Figure 6.9

As can be noted from the above dicussions, the effect of a change in axial load on moment-curvature relationships of reinforced concrete sections depends on whether the initial and final axial load are above or below the balance point. If the axial loads are below the balance point, an increase in axial load causes an increase in stiffness and yield curvature. If, on the other hand, axial loads are above the balance point, an increase in axial load causes a decrease in stiffness and a decrease in the curvature at maximum moment. In one respect, though, behavior above and below the balance point is similar; an increase in axial load results in a decrease in the ductility of a reinforced concrete section.

6.3 Calculated Column Response

The principal result of the analytical column model is a lateral load-drift response curve for a column subjected to monotonically increasing lateral load. The axial load is assumed to have two components, a static one due to gravity load, and a varying one resulting from lateral load on the frame. The changes in axial load are assumed to be proportional to changes in lateral load. This loading can be used to simulate the envelope of the loading imposed on the constant eccentricity specimens tested in the experimental phase of this study.

Figure 6.10 shows the calculated response of a column under constant axial load. As would be expected, the response is symmetrical with respect to the origin, indicating that the column responds in the same fashion regardless of which direction lateral load is applied. Figure 6.11 shows the response of the same column with the same initial axial load, but with an additional variation in axial load superimposed. Total axial load is never greater than the balanced axial load and never less than zero (net tension). The ratio of change in axial load to change in lateral load is approximately three. The changing axial load does have an effect on the response. The response curve is no longer symmetrical with respect to the origin because change in axial load modifies lateral stiffness. Stiffness is greater in the increasing compression direction (first quadrant) than the decreasing compression direction (third quadrant). In addition, maximum lateral load in the decreasing compression direction is significantly lower than the maximum lateral load in the increasing compression direction. This is true because the increase in axial load increases the moment capacity of the column, and as a result the lateral load capacity, while the decrease in axial load reduces the moment capacity. Figure 6.12 shows the same column with a greater variation in axial load. The ratio of change in axial load to change in lateral load is approximately eight, so the column experiences axial loads above

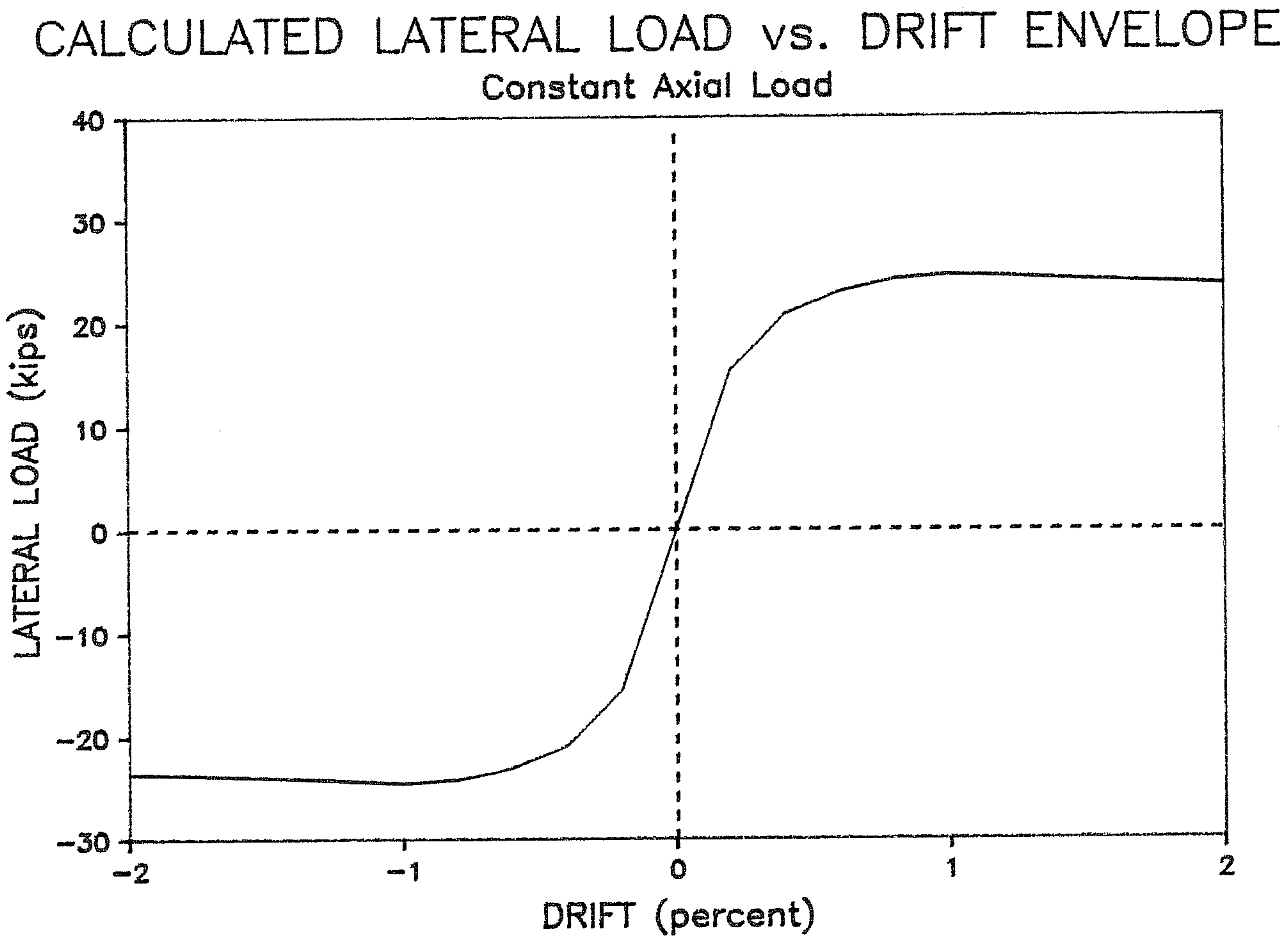

CALCULATED LATERAL LOAD vs. DRIFT ENVELOPE
Constant Axial Load
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
−2
−1
0
1
2
DRIFT (percent)

Figure 6.10

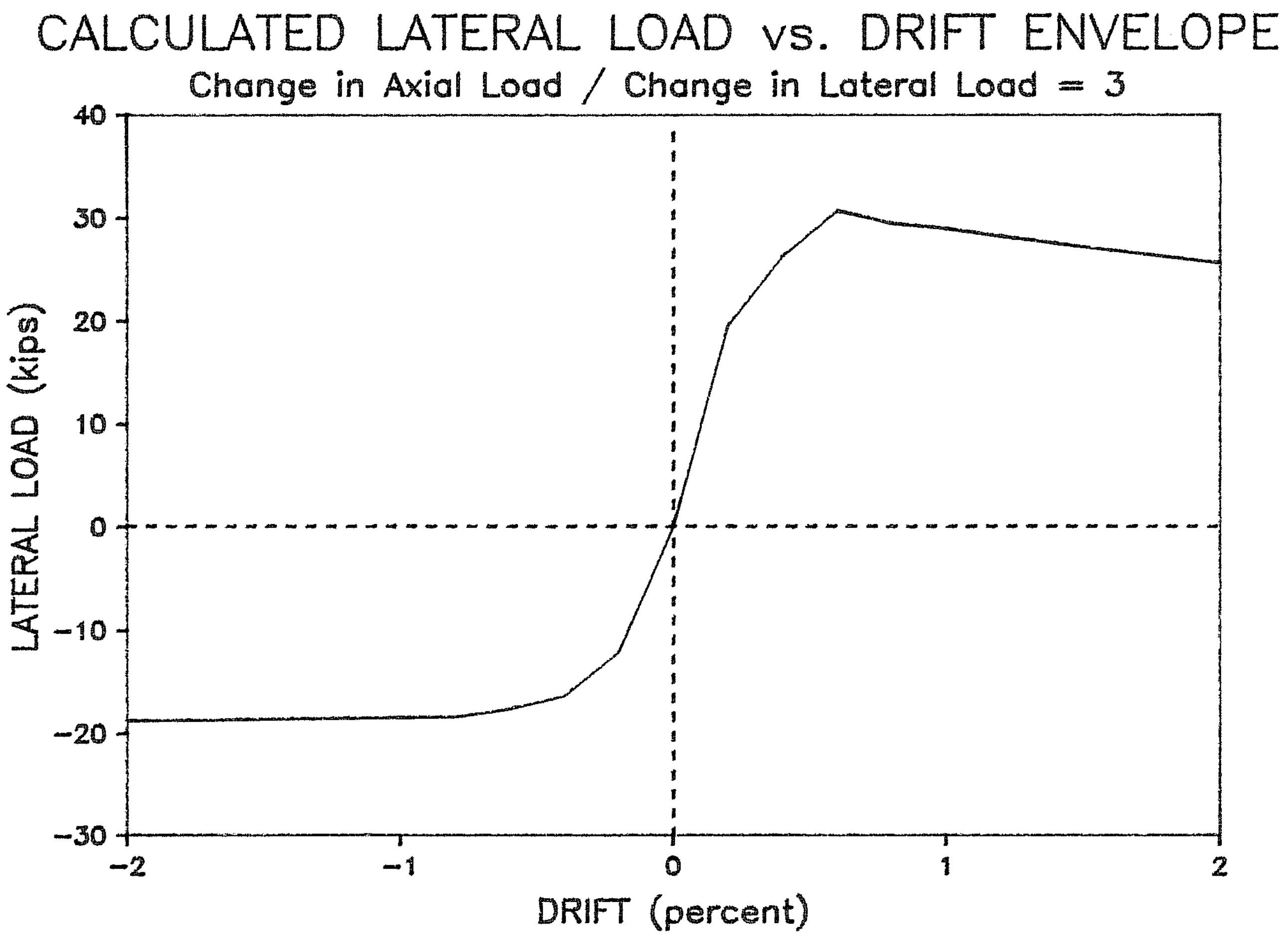

CALCULATED LATERAL LOAD vs. DRIFT ENVELOPE
Change in Axial Load / Change in Lateral Load = 3
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
−2
−1
0
1
2
DRIFT (percent)

Figure 6.11

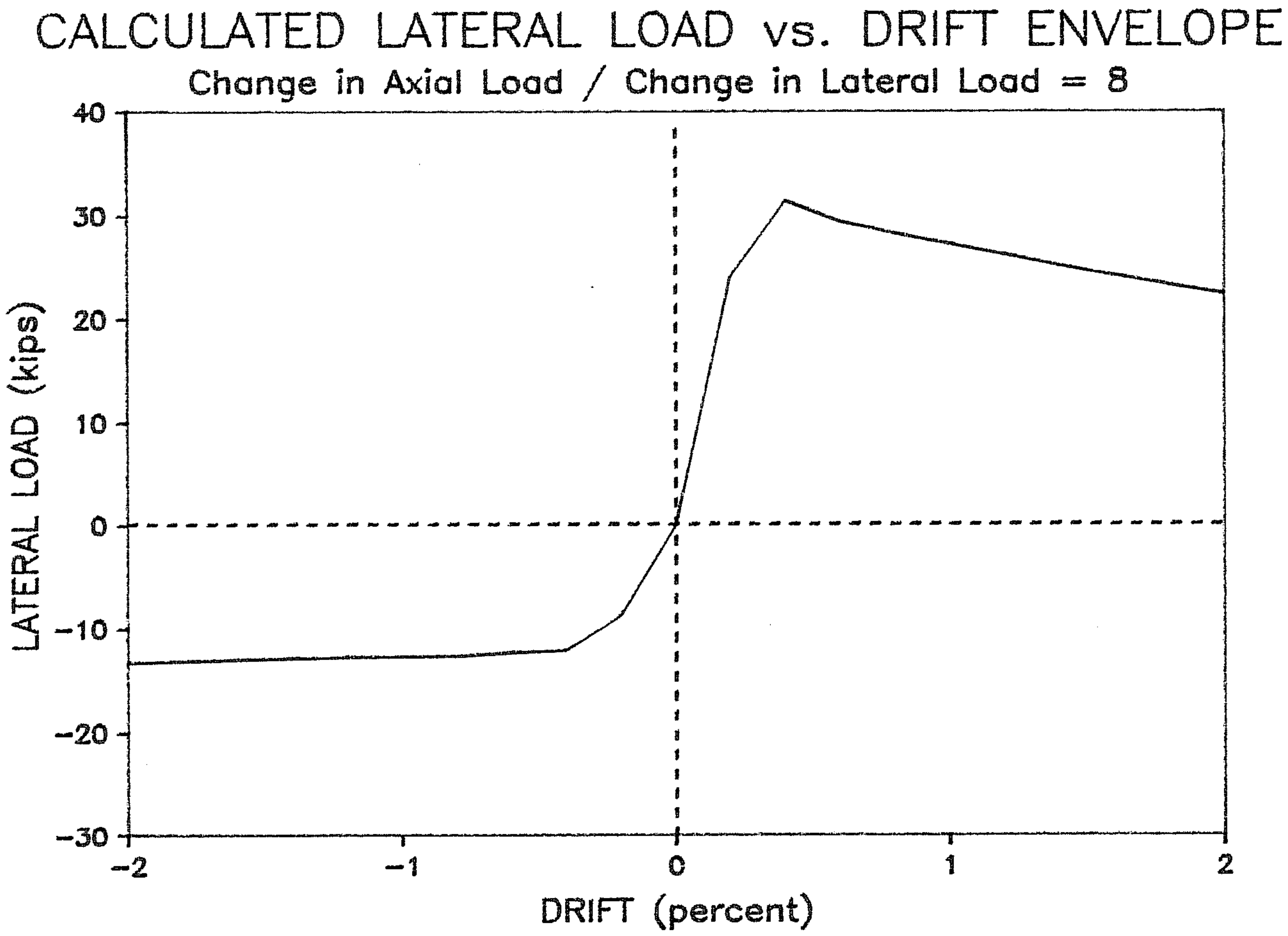

CALCULATED LATERAL LOAD vs. DRIFT ENVELOPE
Change in Axial Load / Change in Lateral Load = 8
LATERAL LOAD (kips)
40
30
20
10
0
-10
-20
-30
-2
-1
0
1
2
DRIFT (percent)

Figure 6.12

the balance point and below zero (net tension). The dissymmetry mentioned above is even more pronounced for this loading, and the increasing compression direction is much stiffer than the decreasing compression direction. Also, the maximum lateral load in the increasing compression direction is greater than the maximum lateral load in the decreasing compression direction due to the decrease in moment capacity which occurs with decreasing axial load.

Comparing the three responses in Figs. 6.10 through 6.12 reveals two further characteristics of column behavior. First, lateral stiffness in the increasing compression direction (first quadrant of lateral load vs drift graphs) increases as the ratio of change in axial load to change in lateral load increases. This increase in stiffness reflects the dependence of lateral stiffness on the level of axial compression in the column. Conversely, an increase in the ratio of change in axial load to change in lateral load causes a decrease in lateral stiffness in the decreasing compression direction (third quadrant of lateral load vs drift graphs). Second, lateral load capacity is related to the ratio of change in axial load to change in lateral load. In the increasing compression direction, an increase in that ratio will effect a higher lateral capacity until the total axial load exceeds the balanced axial load above which lateral load capacity will start to decrease. In the decreasing compression direction, an increase in that ratio will always cause a decrease in lateral load capacity, owing to the decrease in moment capacity which accompanies a decrease in axial load.

In conclusion, lateral load-drift response to varying axial and lateral loads can be characterized by several observations. First, the response is not symmetrical with respect to the origin. The dissymmetry becomes more pronounced as the change in axial load becomes larger. Second, the stiffness in the increasing compression direction is greater than the stiffness in the decreasing compression direction. The difference between these stiffnesses increases as the variation in axial load increases. Third, the lateral load capacity in the increasing compression direction is greater than the lateral load capacity in the decreasing compression direction. This occurs because of the decrease in moment capacity which follows from a decrease in axial load. Fourth, an increase in the ratio of change in axial load to change in lateral load will result in an increase in lateral stiffness in the increasing compression direction and a decrease in lateral stiffness in the decreasing compression direction. Finally, an increase in the ratio of change in axial load to change in lateral load affects the lateral load capacity in both the increasing and decreasing compression directions. In the increasing compression direction a higher ratio will result in a higher capacity until the total axial load exceeds the balance point, and in the decreasing compression direction a higher ratio decreases the lateral load capacity.

6.4. Comparison of Calculated and Measured Responses

The measured response of the three constant relative eccentricity tests and the single test with independently varying axial and lateral loads are compared with calculated responses generated by the analytical model. Experimental and analytical responses are shown together in Figs. 6.13 through 6.19. Lateral load-drift and moment-curvature relationships are compared for each constant relative eccentricity test and lateral load-drift is shown for the specimen subjected to independently varying axial and lateral loads. For comparison of moment-curvature relationships, the bottom end section of the column was used except for specimen C - HA, which lost some of the strain gages at that section. For C - HA, then, top moment and curvature values were used because they were a more reliable indicator of response.

6.4.1. Specimen C - HA. Figures 6.13 and 6.14 show the measured and calculated lateral load-drift and moment-curvature response for specimen C - HA. In the increasing

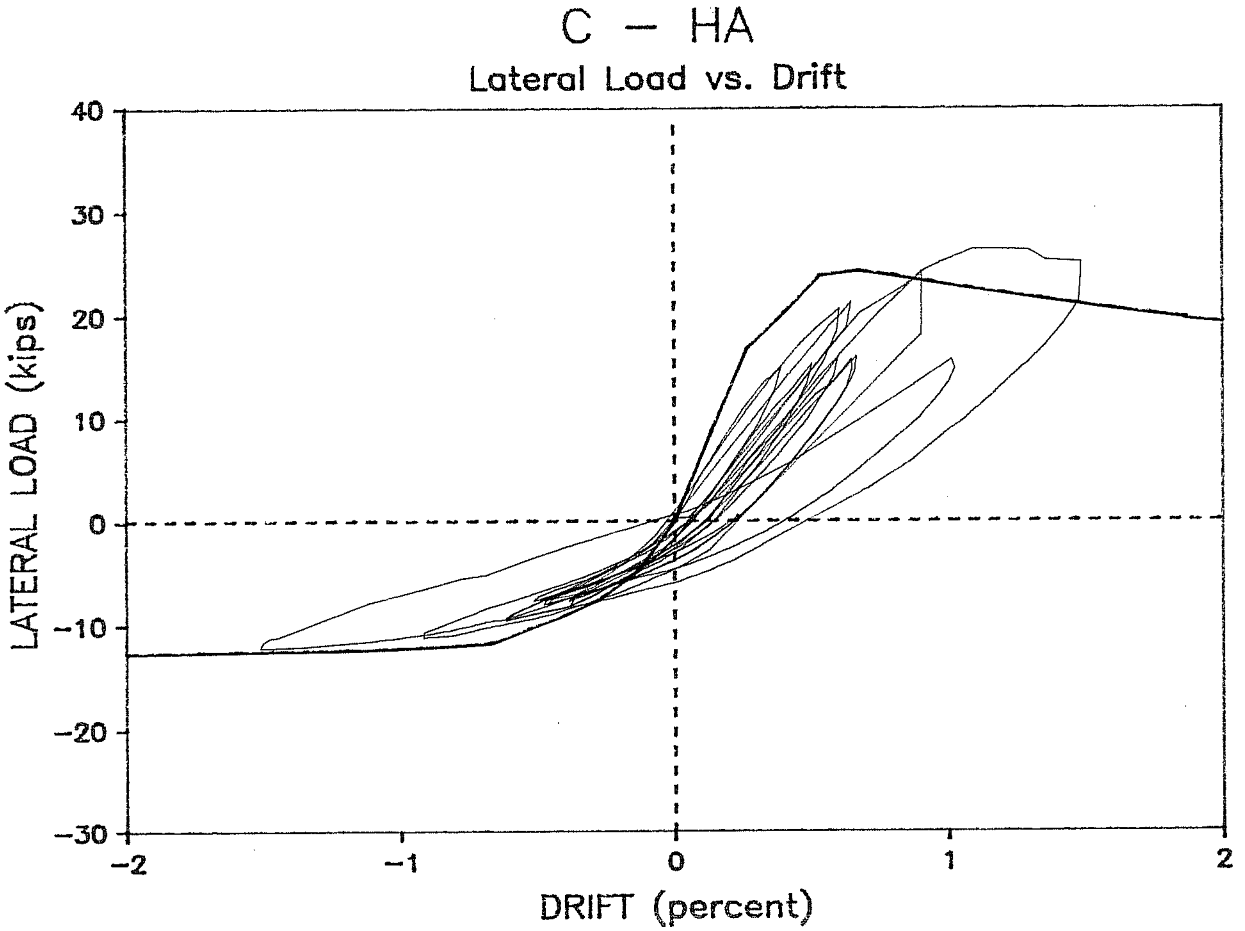

C – HA
Lateral Load vs. Drift
40
30
20
10
0
−10
−20
−30
LATERAL LOAD (kips)
−2
−1
0
1
2
DRIFT (percent)

Figure 6.13

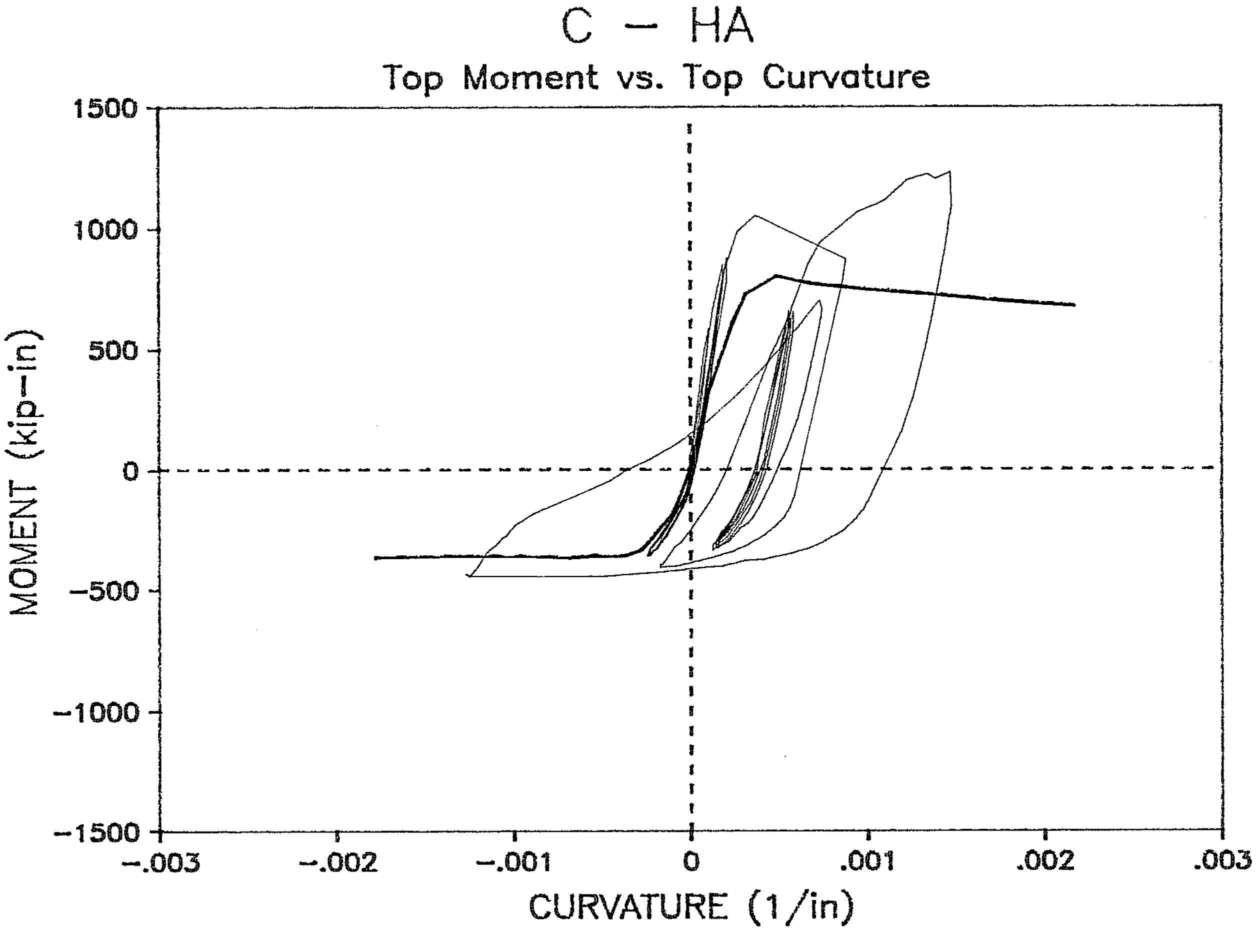

C – HA
Top Moment vs. Top Curvature
1500
1000
500
0
−500
−1000
−1500
MOMENT (kip−in)
−.003
−.002
−.001
0
.001
.002
.003
CURVATURE (1/in)

Figure 6.14

compression direction of the lateral load-drift response, the analytical model underestimates the maximum lateral load by 7.7 percent and overestimates the initial lateral stiffness by approximately 42 percent. The overestimate of stiffness is principally due to the fact that the analytical model does not take into account lack of fixity due to bar slip. Because there is some development length required to anchor the reinforcement, the column is not perfectly fixed to the endblock.

The moment calculated by the model is 35 percent lower than the measured moment. This discrepancy in maximum moment is principally due to the method used by the computer program to simulate a hinge in the column. In order to achieve a given level of drift, hinges at each end of the column must undergo a given amount of rotation. Hinge curvature is a function of this rotation and the length of the hinge. Because the hinge length used in the mode was merely an estimate, it is possible that the actual hinge length was larger than that estimated by the analytical model. This would result in a smaller end curvature and larger moment than that calculated by the model, which was in fact the case. Calculated initial section stiffness of specimen C - HA, however, is very nearly the same as the measured initial section stiffness. The difference due to bar slip noted above does not influence the section stiffness, thus allowing good estimates of section stiffness by the model.

In the decreasing compression direction, the calculated maximum moment is approximately 27 percent less than the measured moment. The influence of axial load is reduced in the decreasing compression direction; in fact, the column is in net tension when the maximum negative moment is reached.

6.4.2. Specimen C - LA - 1. The lateral load-drift relationships measured for specimen C - LA - 1 and calculated using the analytical model are shown in Fig. 6.15. Maximum lateral load in the increasing compression direction is reasonably reproduced by the analytical model and is underestimated by 13 percent in the decreasing compression direction. Initial stiffness calculated using the model is approximately 40 percent higher than the measured stiffness of the test specimen, again due to the bar slip effect discussed above. The general shape of the envelope is good, and lateral capacity at large deformations is calculated with reasonable accuracy.

The moment-curvature relationship measured for specimen C - LA - 1 is shown in Fig. 6.16 along with the moment-curvature envelope generated by the analytical model. The calculated stiffness is slightly less than the measured stiffness. Maximum moment in the decreasing compression direction is approximately

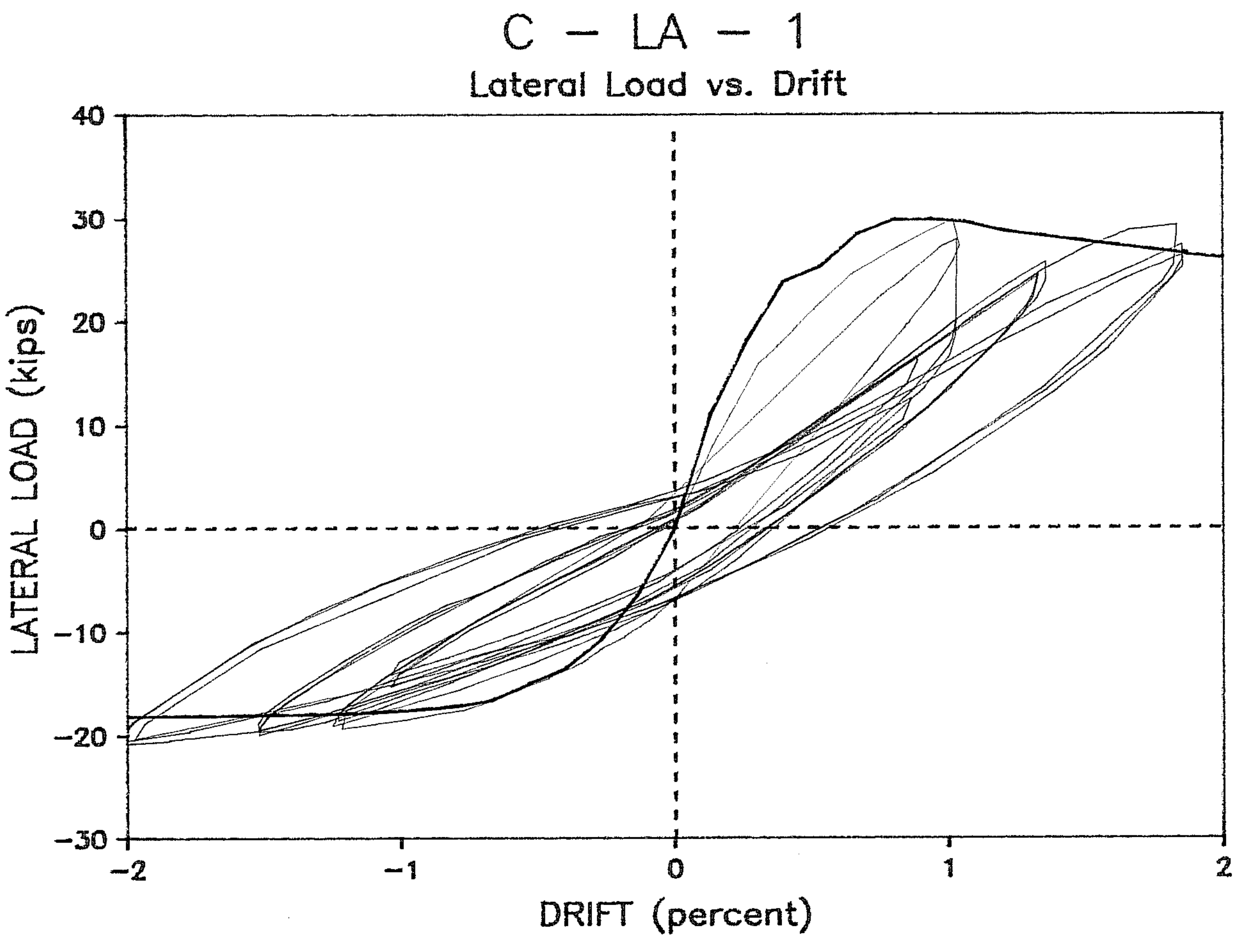

C – LA – 1
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
−2
−1
0
1
2
DRIFT (percent)

Figure 6.15

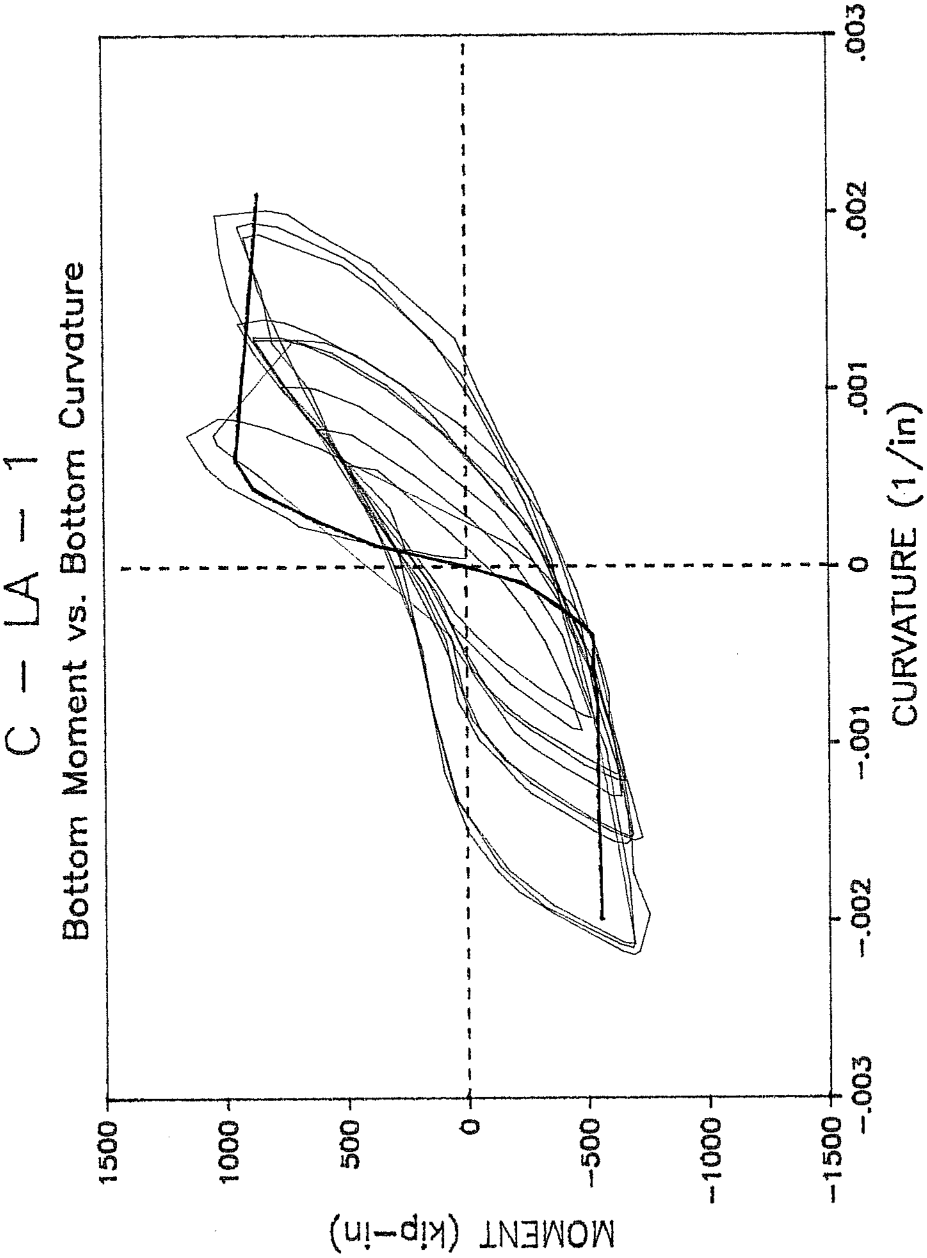

C – LA – 1
Bottom Moment vs. Bottom Curvature
MOMENT (kip-in)
1500
1000
500
0
-500
-1000
-1500
CURVATURE (1/in)
-.003
-.002
-.001
0
.001
.002
.003

Figure 6.16

27 percent less than the measured maximum moment, and the calculated maximum moment in the increasing compression direction is approximately 17 percent less than the measured moment. This difference is principally due to the analytical model emulated column hinging, as was discussed previously for specimen C - HA.

6.4.3. Specimen C - LA - 2. The measured lateral load-drift relationship for specimen C - LA - 2 is shown in Fig. 6.17 along with the calculated lateral load-drift envelope. As in C - HA and C - LA - 1, the initial calculated column stiffness is higher than the measured column stiffness, in this case by 45 percent. Also, the maximum calculated lateral load in the increasing compression direction is very close to the maximum measured lateral load, and the maximum measured lateral load in the decreasing compression direction is underestimated by 20 percent with the analytical model.

Measured and calculated moment-curvature relationships for specimen C - LA - 2 are shown in Fig. 6.18. The calculated and measured section stiffnesses are almost equal, but the calculated maximum moment in both the increasing and decreasing compression directions is lower than the measured value by 13 and 35 percent, respectively. As discussed above, this is at least mostly due to the method used by the analytical model to represent column hinging.

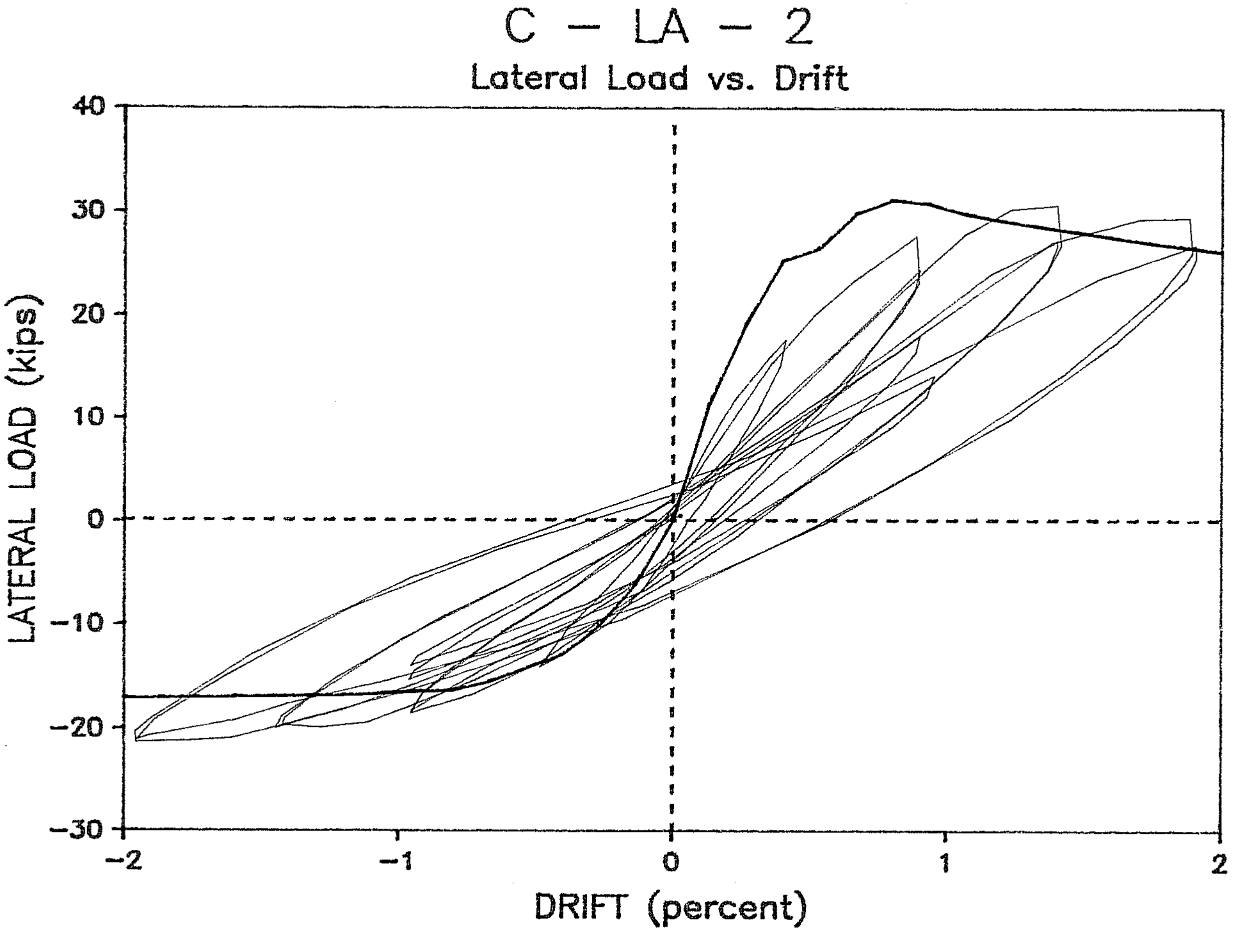

C – LA – 2
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
−2
−1
0
1
2
DRIFT (percent)

Figure 6.17

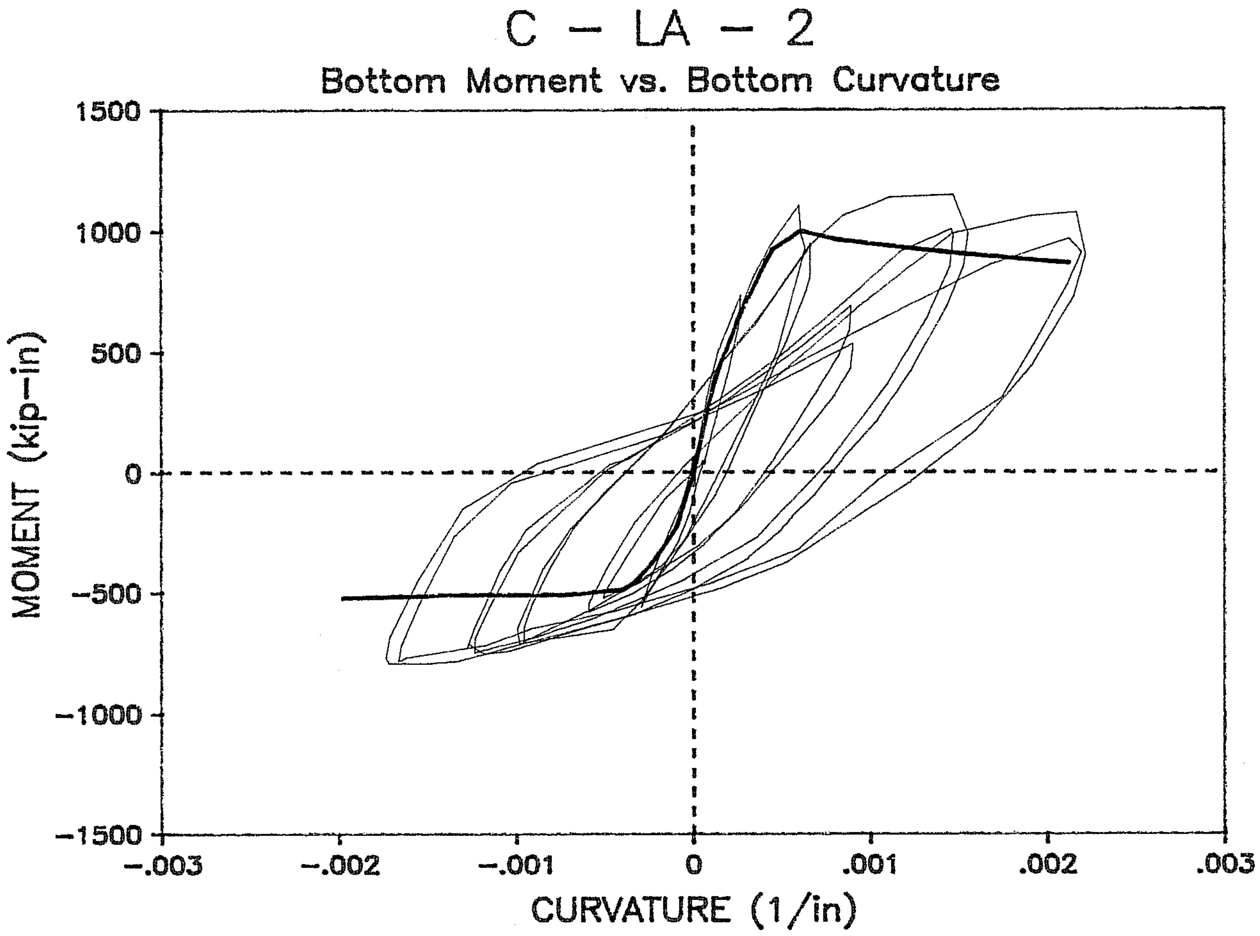

C – LA – 2
Bottom Moment vs. Bottom Curvature
MOMENT (kip–in)
1500
1000
500
0
–500
–1000
–1500
–.003
–.002
–.001
0
.001
.002
.003
CURVATURE (1/in)

Figure 6.18

6.4.4. General trends for constant relative eccentricity specimens. There are some clear trends in the calculated and measured responses of the three constant relative eccentricity test specimens. In each case, a reasonable estimate of the maximum lateral load is calculated for both the increasing and decreasing compression directions. When there is a difference between calculated and measured values, the calculated value is always smaller and therefore conservative. The column stiffness calculated using the analytical model is higher (at least 40 percent) than the measured value for all column specimens. The fact that initial section stiffnesses for ends of the columns are reproduced so well suggests that additional flexibility is due to slipping of reinforcement in the anchorage, a phenomenon not accounted for in the analytical column model. Bar slip, however, would not influence the results of the section model.

Measured and calculated stiffnesses of column end-sections, as determined by the moment-curvature relationships, are nearly identical for all three specimens. However, measured maximum moment in both the increasing and decreasing compression directions is determined to be at least 13 percent, and in most cases 20 percent, greater than the moment calculated by the analytical model. This is due, in part, to the high column stiffness noted above, which causes a smaller second-order effect in the models than in the actual column. Consequently the total moment on the section at high load levels is smaller.

6.4.5. Specimen I - HA. The lateral load vs. drift relationship for specimen I - HA is shown in Fig. 6.19, together with points generated by the analytical model. Points shown in Fig. 6.19 are either peak values of axial or lateral load cycles, and were calculated using the load-control algorithm. While stiffness comparisons are more difficult to make for this specimen due to the uncoupled axial and lateral loads, the maximum levels of lateral load attained in both directions are reproduced within 15 percent, as is the maximum level of drift in the first quadrant which is underestimated by approximately 11 percent. The maximum drift in the third quadrant, however, is greatly underestimated by the analytical model and the grouping of points about the lateral load axis is not representative of the measured response. Closer examination of the loading reveals one possible explanation for the differences in the calculated and measured responses.

Portions of the response which are closely reproduced by the analytical model were at stages with relatively low axial load. Under low axial load, steel behavior tends to dominate the response of the column because much of the concrete is cracked. The portions of the response which are not reproduced well by the

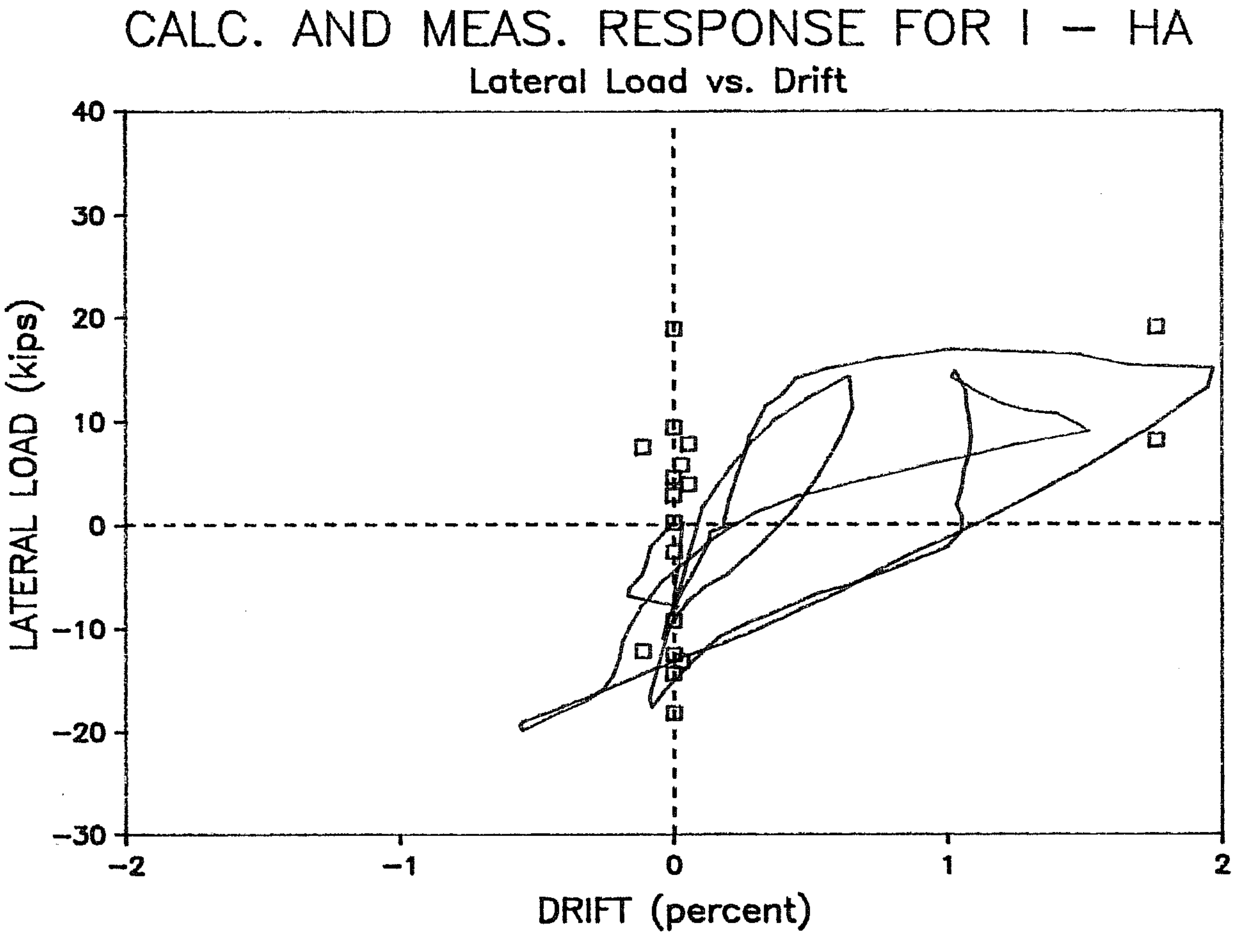

CALC. AND MEAS. RESPONSE FOR I — HA
Lateral Load vs. Drift
LATERAL LOAD (kips)
40
30
20
10
0
−10
−20
−30
−2
−1
0
1
2
DRIFT (percent)

Figure 6.19

analytical model correspond with load stages with high axial compression. In these regions of the response, concrete dominates the behavior of the column. Both steel and concrete models used in the analysis are path-independent, that is they do not account for previous damage done to the material. Because concrete behavior is more dependent on previous stress states, a greater discrepancy between measured and calculated response is found when concrete behavior dominates the response of the column, as is the case for high axial loads.

6.5. Stiffness Determination and Implications for Design

In this section measured and calculated stiffness values are presented and discussed. Calculated stiffnesses are obtained using the analytical model developed in Chapter 5 and with traditional elastic calculations using gross and transformed sections. Calculated and measured stiffnesses are evaluated to present recommendations for effective column stiffnesses for use in analysis of reinforced concrete frames subjected to severe lateral loads. Due to the similarity of both loading and material properties, stiffnesses from tests and analysis of specimens C - LA - 1 and C - LA - 2 are averaged and treated as a single specimen, hereafter referred to as C - LA.

6.5.1. Section stiffness. Traditionally, stiffness of a reinforced concrete section is estimated by multiplying the modulus of elasticity of concrete (E_c) by some effective moment of inertia for the section (I). The most commonly used values for E_cI are based on a secant modulus of elasticity proposed by Pauw [38] and an effective moment of inertia. There are several different methods for calculating the effective moment of inertia, ranging from the very simple gross section to that calculated by a more complex expression which takes into consideration cracking and steel contribution to section stiffness. Insight can be gained by comparing calculated and measured section stiffnesses with three idealized values of E_cI: the first using a moment of inertia calculated using gross section dimensions, the second based on the moment of inertia for an uncracked transformed section, and the third using a moment of inertia for a cracked transformed section. Figures 6.20 and 6.21 show these three stiffnesses along with the stiffnesses calculated using the analytical model in Chapter 5 and measured stiffnesses obtained from the experimental program described in Chapters 3 and 4. Stiffnesses are normalized with respect to the gross section stiffness (3115 kip-in_2 for C - HA and 3618 kip-in_2 for C - LA) for ease of comparison. Measured stiffnesses and stiffnesses calculated with the column model are secant stiffnesses obtained from moment-curvature relationships corresponding with a drift level of approximately 0.5 percent,

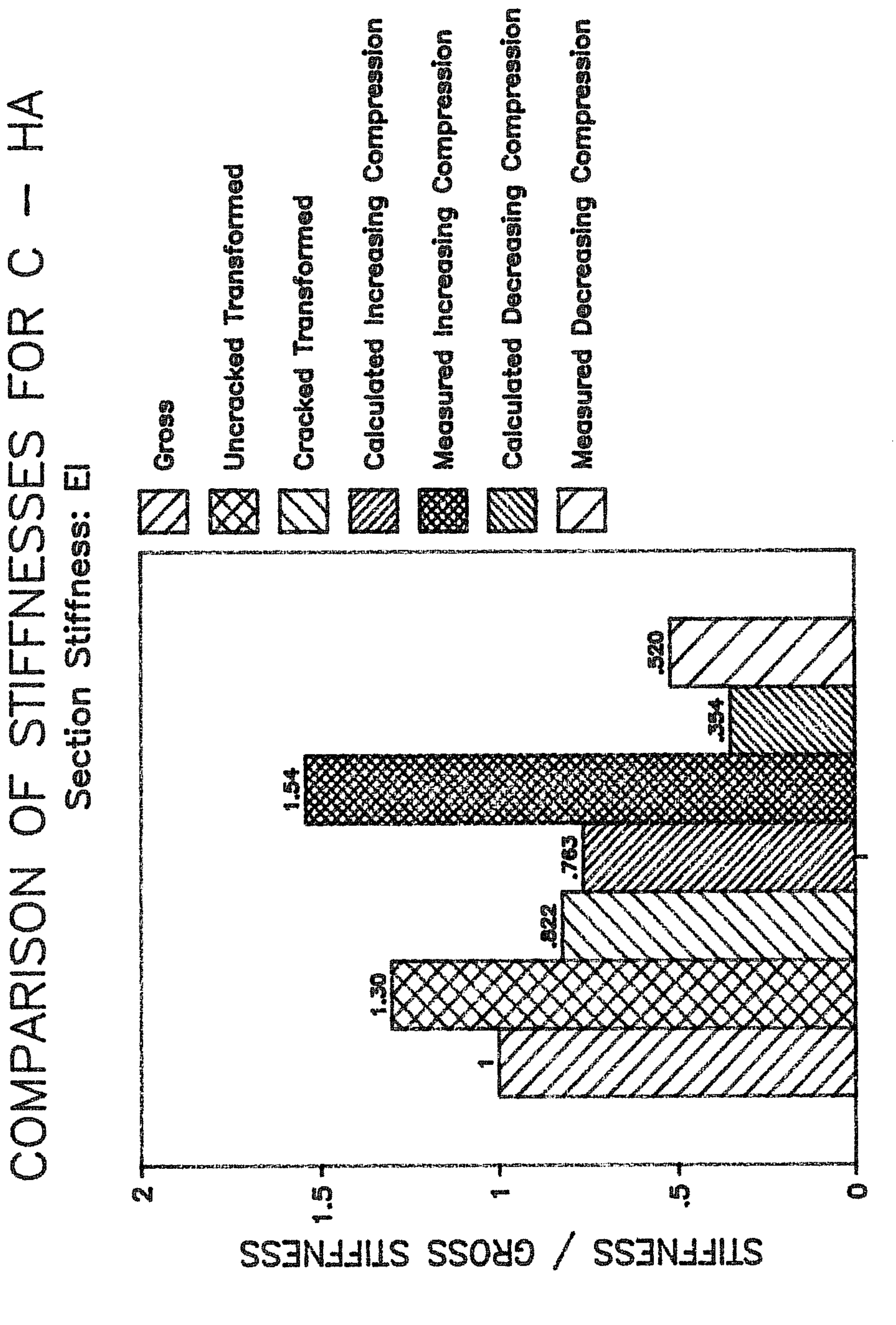

COMPARISON OF STIFFNESSES FOR C – HA
Section Stiffness: EI
Gross
Uncracked Transformed
Cracked Transformed
Calculated Increasing Compression
Measured Increasing Compression
Calculated Decreasing Compression
Measured Decreasing Compression
1
1.30
.822
.763
1.54
.354
.520
STIFFNESS / GROSS STIFFNESS
2
1.5
1
.5
0

Figure 6.20

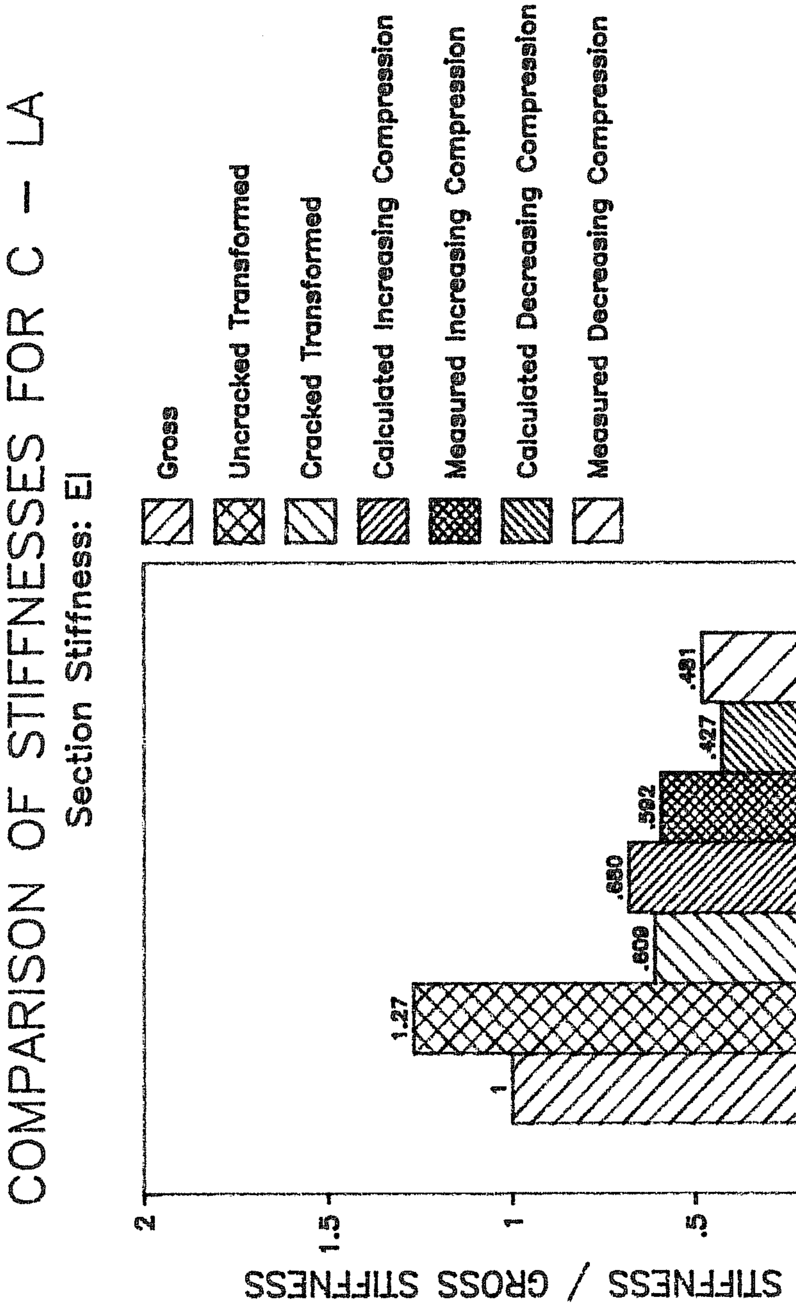

COMPARISON OF STIFFNESSES FOR C – LA
Section Stiffness: EI
▨ Gross
▩ Uncracked Transformed
▧ Cracked Transformed
▨ Calculated Increasing Compression
▩ Measured Increasing Compression
▧ Calculated Decreasing Compression
▨ Measured Decreasing Compression
1
1.27
.609
.680
.592
.427
.481
2
1.5
1
.5
0
STIFFNESS / GROSS STIFFNESS

Figure 6.21

which is the drift limit specified by the Uniform Building Code [39] for buildings subjected to seismic loads.

Calculated model stiffnesses in the increasing compression direction are within ten percent of the cracked transformed section stiffness for both C - HA and C - LA. Model stiffnesses in the decreasing compression direction are significantly less than the cracked transformed stiffness: 43 and 63 percent of the cracked transformed stiffnesses of C - HA and C - LA, respectively. The higher model stiffness in the increasing compression direction is due to the influence of axial load, which increases section stiffness by closing cracks thus making more of the concrete effective in resisting load.

This variation in stiffness due to axial load is also found in the measured stiffnesses. However, the range of measured stiffnesses for specimen C - LA is very different from specimen C - HA. Measured stiffnesses for specimen C - LA ranged between 48 and 59 percent of the gross section stiffness (fairly close to the model range of 43 to 68 percent), whereas measured stiffnesses for specimen C - HA ranged between 52 and 154 percent of the gross section stiffness (compared to 35 and 76 percent for the model). This difference in ranges demonstrates the profound effect which axial load can have on section stiffness. In the case of the increasing compression stiffness of C - HA, enough axial load was on the column (228 kips) to prevent almost any cracks from forming. The model predicts cracking in specimen C - HA at a lower lateral load than actually occurred during the test. With the exception of this test, however, the model overestimates the effect of axial load on section stiffness, that is the model produces a larger difference between increasing and decreasing compression stiffnesses. In fact, the model overestimates stiffness in the increasing compression direction and underestimates stiffness in the decreasing compression direction.

6.5.2. Column stiffness. Lateral stiffness of a column is generally determined from a section stiffness and end conditions. For instance, a prismatic column which is fixed at both ends has a lateral stiffness of $12EI/L_3$ and a column which is fixed at one end and pinned at the other has lateral stiffness of $3EI/L_3$.

Figures 6.22 and 6.23 show the column stiffnesses determined from each of the sources of section stiffness described above. The idealized column stiffnesses (gross, uncracked transformed, and cracked transformed sections) are based on a $12EI/L_3$ calculation appropriate for the end conditions associated with each test specimen. Actual end restraint for each test specimen was not fixed-fixed as idealized, however, resulting in some difference between end moments. For the

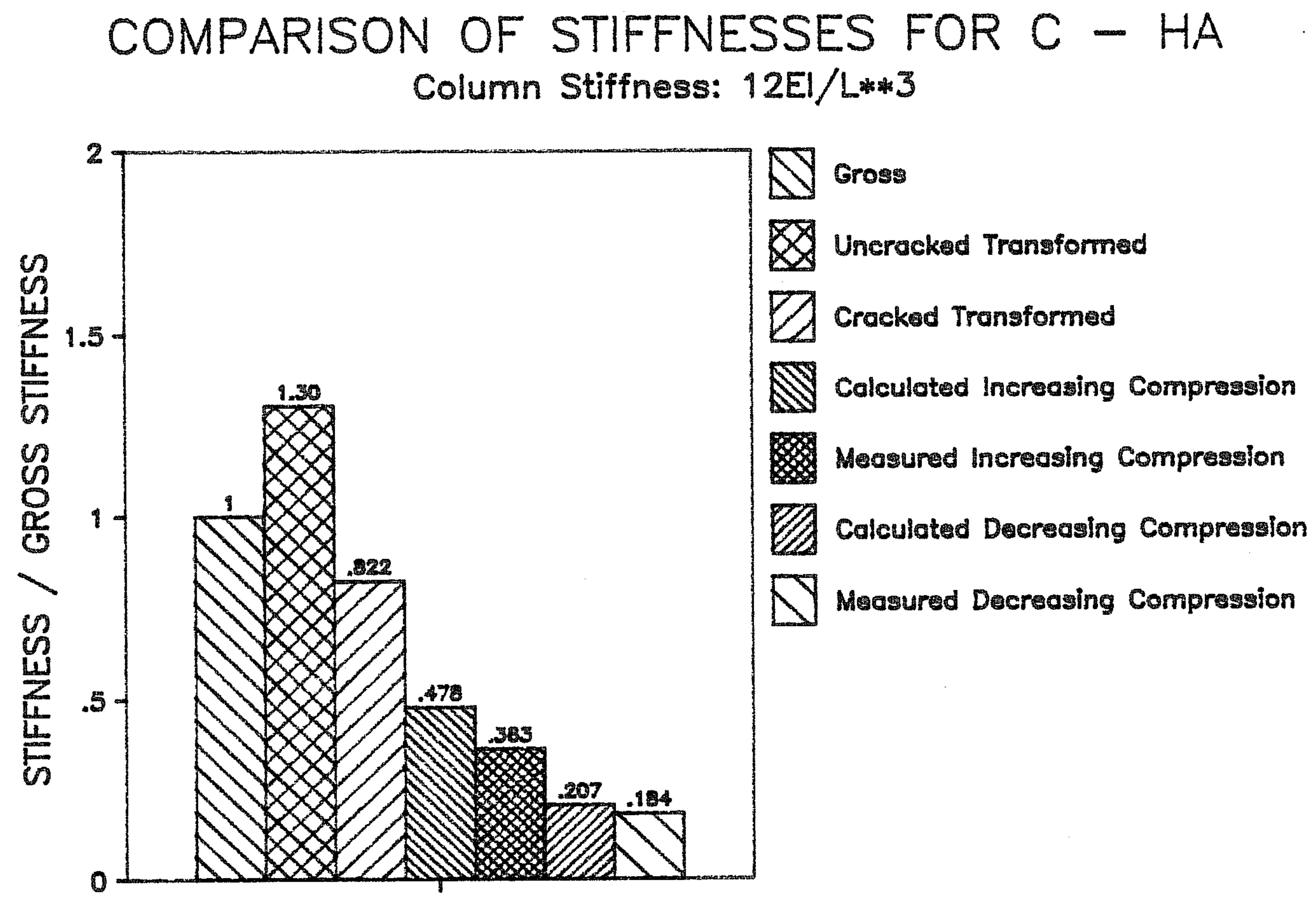

COMPARISON OF STIFFNESSES FOR C – HA
Column Stiffness: 12EI/L**3
STIFFNESS / GROSS STIFFNESS
2
1.5
1
.5
0
1
1.30
.822
.478
.383
.207
.184
Gross
Uncracked Transformed
Cracked Transformed
Calculated Increasing Compression
Measured Increasing Compression
Calculated Decreasing Compression
Measured Decreasing Compression

Figure 6.22

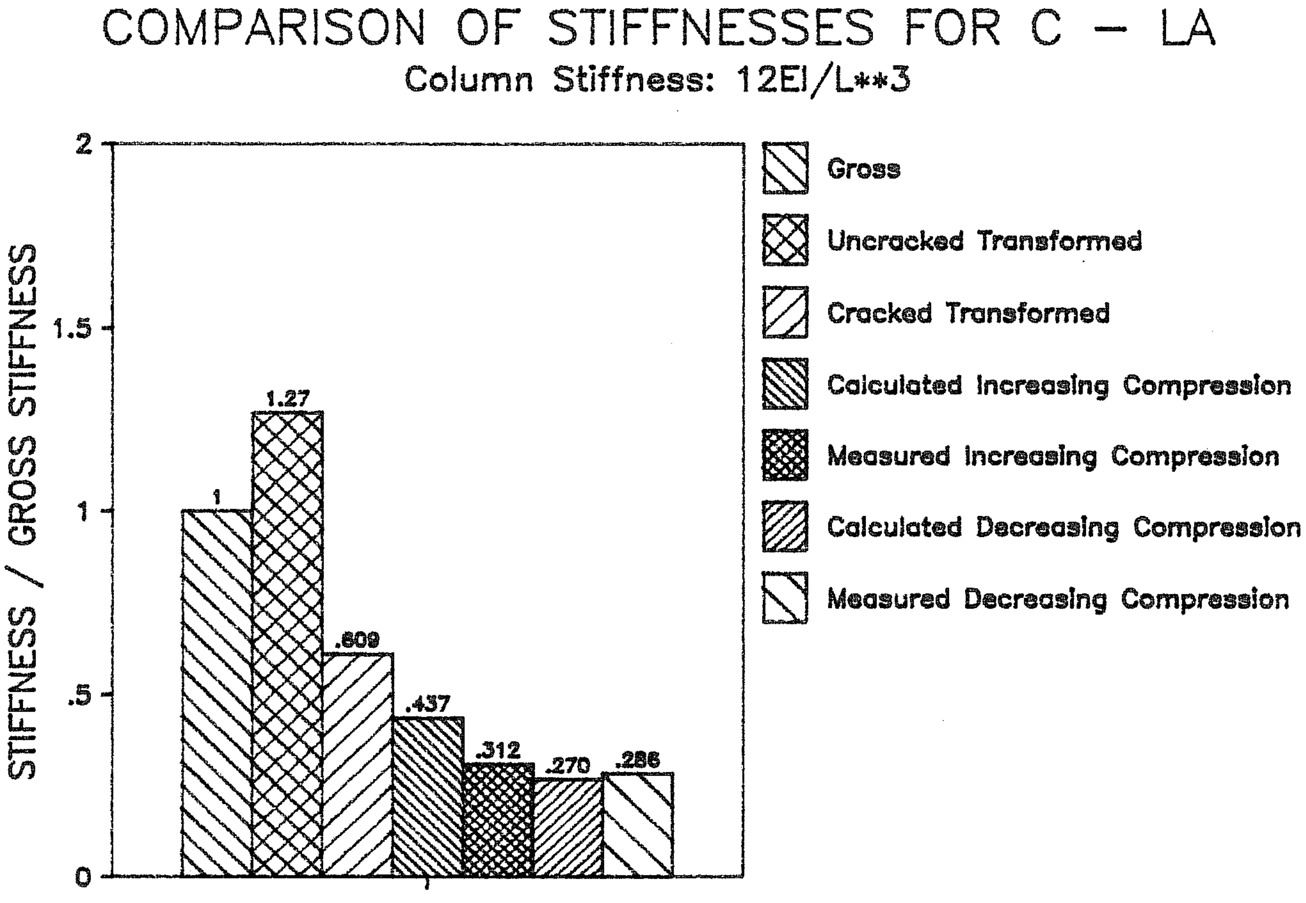

COMPARISON OF STIFFNESSES FOR C - LA
Column Stiffness: 12EI/L**3
STIFFNESS / GROSS STIFFNESS
2
1.5
1
.5
0
1
1.27
.609
.437
.312
.270
.286
Gross
Uncracked Transformed
Cracked Transformed
Calculated Increasing Compression
Measured Increasing Compression
Calculated Decreasing Compression
Measured Decreasing Compression

Figure 6.23

idealized column stiffness calculations mentioned above, the unequal end moments were considered to appropriately modify the $12EI/L_3$ column stiffness before the values were tabulated. Analytical and measured stiffnesses were calculated directly from lateral load-lateral deflection data at a drift of approximately 0.5 percent. All results are also normalized with respect to gross column stiffness for comparison.

All column stiffnesses determined by the analytical model and measured from the experimental tests are lower than the corresponding cracked transformed column stiffness. In fact, all measured and calculated stiffnesses are less than 70 percent of the cracked column stiffness, and most are less than 50 percent. Column stiffnesses calculated by the analytical model are closer to measured stiffnesses than are idealized stiffnesses. In the loading direction associated with increasing axial compression, the calculated model stiffness is at least 32 percent higher than the measured stiffness. This suggests that the analytical model overestimates the influence of increasing axial load on the lateral stiffness.

Measured stiffnesses range from 18 to 36 percent of the gross column stiffness, with most values near 30 percent. Columns subjected to smaller changes in axial load exhibited very little difference between the measured secant stiffness in the increasing and decreasing compression direction. It is therefore reasonable to assume that an interior column, with a constant axial load comparable to the initial axial load applied to specimens tested here (approximately ten percent of the axial compressive strength), would have a stiffness of approximately 30 percent of the gross column stiffness. Exterior columns in frames, one of which would experience increasing compression and the other decreasing compression, develop stiffnesses which do not completely offset each other. However, the difference in stiffness for either of the loadings used in the laboratory is not great and would be relatively insignificant in terms of the overall story stiffness. Consequently, a reasonable estimate of lateral stiffness for all columns in a story would be 30 percent of the stiffness based on gross section dimensions Because the stiffness based on a cracked section is approximately 60 to 80 percent of the gross stiffness, use of this stiffness would overestimate the stiffness of the column by more than 100 percent. Even use of the ACI Building Code estimate [22,23], based on 40 percent of the gross column stiffness, would overestimate column stiffness as indicated by tests performed in this study. The implications of overestimating the effective column stiffness are discussed below.

6.5.3. Implications on column design. While it is important to note the difference between calculated and measured column stiffness, it is more important to understand the effect of this difference on the behavior of a frame. To illustrate these effects, it is helpful to consider the analysis and design of a column in a frame. In general there are two criteria which must be satisfied in a design: strength and stiffness. Determination of required strength is based on either a first-order (axial load effects on member response ignored) or second-order (axial load effects on member response included) analysis. (Note that true second-order analyses are not often performed.) Stiffness requirements are generally specified as a maximum allowable drift for a story or an entire frame.

In a first-order analysis, distribution of forces between members of the frame is dependent on relative stiffnesses of the members. If the stiffness of one member is overestimated, that member resists more moment, while the other members at that joint resist less. If the design of those members is then based on forces calculated by that analysis, the member for which the stiffness was overestimated will be designed conservatively while the other members will be underdesigned because they will experience more moment than was anticipated.

In reality, of course, this may not be a problem because once a member reaches its capacity, hinges will form and moment will be redistributed. The total available strength of the joint will not be greatly affected as long as sufficient deformation capacity is provided to allow for redistribution of forces (this emphasizes the need for proper detailing). Consequently, precise determination of the relative stiffness of members at a joint is desirable but not crucial.

Knowledge of the actual relative member stiffnesses is important in the sequence of hinge formation. If the stiffness of a member is underestimated and the member is underdesigned, it will most probably hinge before other members at that joint. If that member is a column, hinging may seriously reduce the vertical load carrying capacity of that portion of the frame. In fact, as was noted earlier, particular combinations of axial load and moment can result in complete material failure of the column, not simply flexural hinging. Therefore, in a first-order analysis for determination of force distribution it is better to overestimate column stiffness and underestimate beam stiffness to insure the vertical load carrying capability of the frame by forcing a more desirable sequence of hinge formation. Current design recommendations in North America for beam-column joints [40] address this by requiring additional column strength beyond what is required to balance the flexural strength of beams framing into the same joint.

In performing a second-order analysis, that is an analysis which accounts for the contribution of axial load to the moment experienced by a column, a good estimate of the actual member stiffness is crucial to the determination of second-order effects. This is so because the amount of second-order moment is related to the deflection of the column, which is related to the actual stiffness of that column as well as the relative stiffness of that column in a joint. The stiffer the column, the less second-order moment it experiences, so an overestimate of the column stiffness results in an underestimate of the second-order moment it would be required to resist. In many cases this may not be a problem, because an overestimate of stiffness would result in an overestimate of first-order moment, which may offset the underestimate of second-order moment. For frames which are very sensitive to second-order effects, however, an overestimate of column stiffness could result in an unconservative design.

The relative stiffness also plays a part in the determination of second-order effects. The lateral stiffness of the column is a function of the end restraint provided. If the beams framing into the joint are very stiff relative to the column, the column has a stiffness of approximately $12EI/L_3$, but if the beams are very flexible, the lateral stiffness of the column is much less. Therefore, overestimating the stiffness of the beams at a joint results in an underestimate of the second-order moments in the columns framing into that joint.

It is especially important to err on the side of flexibility in a second-order analysis, much more so than in a first-order analysis. In a first-order analysis, overestimating the stiffness of a particular member will not significantly affect the total capacity of the joint because of the ability of understrength members to hinge and redistribute additional loads to adjacent members. In a second-order analysis, this redistribution cannot take place because formation of a hinge significantly decreases lateral stiffness, thus increasing the second-order moments. This increase in moment causes a further decrease in stiffness, and possibly more hinging. If the initial estimate of lateral stiffness is too high, it is possible that the structure may appear able to resist a load which, in reality, could cause instability of a column and perhaps the entire frame. Of course, there is some room for error because some nonstructural elements, such as partitions and non-load-bearing walls, do contribute additional stiffness to the structure which is not accounted for in most analyses. However, as lighter movable partitions grow in popularity this additional stiffness will decrease and structures will tend to behave more like the analytical model. Therefore, it is important that beam and column stiffnesses not be overestimated in order to fully account for second-order effects.

6.6. Summary and Conclusions

Based on the observations made in the previous sections, the following points may be made:

1. The analytical model developed in Chapter 5 reasonably reproduces the maximum lateral load of constant relative-eccentricity specimens in the direction of both increasing and decreasing axial compression.
2. Initial lateral stiffness of constant relative eccentricity specimens measured in the experimental program is overestimated by the analytical model. This is principally due to the effects of bar slip at the ends of the column.
3. Moment-curvature envelopes calculated for constant relative-eccentricity specimens using the analytical model accurately reproduce measured initial section stiffnesses, but underestimate the maximum moment attained by the section.
4. Maximum lateral load of specimen I - HA, which was subjected to independently varying axial and lateral loads, was reasonably predicted by the analytical model. Due to the strong path-dependence of concrete, however, deformations at high axial load levels are not reproduced.

5. Knowledge of the actual stiffness of a column is not necessary, although desirable, in a first-order analysis because relative stiffness of members at a joint are used to determine member forces. A moderate error in relative stiffnesses can be tolerated because of the ability of members to redistribute forces when capacities have been reached, assuming of course that members have been properly detailed and constructed.

6. A conservative approach to a first-order analysis would be to underestimate beam stiffnesses and overestimate column stiffnesses. Beams and columns which resist the forces determined by such an analysis would be more likely to redistribute forces in a desirable manner.

7. The use of stiffness values for beams and columns which reasonably approximate actual stiffnesses is crucial in a second-order analysis or an analysis to check drift limits. If estimated stiffnesses are higher than actual stiffnesses, results of these analyses may be unconservative.

8. Based on measured column stiffnesses obtained from the limited number of tests performed in the

experimental program described in Chapters 3 and 4, a reasonable estimate of column stiffness should be obtained by using a value of 0.3EI, where E is the secant modulus of concrete as recommended by ACI 318-83 and I is the moment of inertia based on gross section dimensions. Given the importance of relative beam-column stiffness, it is also necessary to use an appropriate beam stiffness in order to assure satisfactory performance. Recommending a value for beam stiffness, however, is beyond the scope of this study.

CHAPTER 7

SUMMARY AND CONCLUSIONS

The behavior of reinforced concrete columns subjected to cyclic axial and lateral load reversals was studied. The experimental program consisted of four column specimens subjected to one of three different loadings. The analytical program involved the development of an interactive computer program to calculate the response of a reinforced concrete column subjected to monotonically varying axial and lateral loads.

7.1. Experimental Program

All specimens had the same geometry and reinforcement. Each column specimen was 10 inches square and 5 feet long, with a 36 inch square, 18 inch deep anchorage block at each end. The specimens were 0.42-scale model of a prototype column which was 24 inches square and 12 feet long. Concrete compressive strength at 28 days for all specimens was between 4500 psi and 6000 psi. Longitudinal reinforcement consisted of ten #4 bars, four on each primary face and two at mid-depth, for a reinforcement ratio of two percent. Transverse reinforcement consisted of stirrups made from 1/4 inch diameter undeformed bar placed at 2-1/2 inch intervals over the entire length of the column. All reinforcement was Grade 60, and the specimen was designed in accordance with Appendix A, Special Provisions for Seismic Design, of ACI 318-83, Standard Building Code.

Lateral loading was controlled by monitoring drift. Three different load programs, corresponding to three different structural systems, were selected. The first two loading programs, which subjected specimens to cyclic axial and lateral load reversals which varied at the same rate, were representative of loads experienced by an exterior column in a plane frame. For these load programs, the change in axial load was held proportional to the change in lateral load throughout the test. Initial axial load imposed on each column specimen was 70 kips. The difference between the two load programs was the ratio of change in axial load to change in lateral load. For one load program this ratio was 8.4 and for the other the ratio was 3.3. The third load program imposed cyclic axial and lateral load reversals which varied at different rates, representative of a structure with two different lateral-force-resisting systems in orthogonal directions. The prototype structure used to develop this loading program was a staggered shear wall-frame structure. All specimens were tested in an apparatus in which relative rotation of their ends was restrained. Despite unexpected flexibility at low load levels, this leveling apparatus performed well.

The experimental program revealed several significant trends:

1. In constant-relative eccentricity tests, load-deformation hysteresis relationships were not symmetrical with respect to the origin. This contrasts with the symmetry found in tests of columns subjected to cyclic lateral load reversals and constant axial load.
2. The post-ultimate capacity of the specimen was strongly dependent on the axial load at ultimate. When ultimate lateral load was reached at axial load levels above the balance point, the lateral capacity of the column decreased at higher deformations.
3. Lateral stiffness of a reinforced concrete column was found to be profoundly influenced by axial load. The lateral stiffness of a column under moderate axial compression could be several times higher than the lateral stiffness of the same column under axial tension. This was clearly illustrated in the behavior of the specimen subjected to independently varying axial and lateral loads. On two occasions, that specimen experienced a simultaneous decrease in lateral load

and increase in drift due to a rapidly decreasing axial load.

4. Columns loaded to a new displacement level displayed a reduction in stiffness when loaded again to the same displacement. Additional loass in stiffness at this displacement level was experienced after the column was loaded to an even higher level of displacement. Consequently, it can be concluded that column stiffness degrades as higher levels of deformation are imposed on the column.

5. Specimens attained their peak lateral load at drifts in the range of 1.2 to 1.4 percent, and for higher levels of drift lateral load capacity decreased. All specimens maintained a substantial portion of their lateral capacity at high drifts, although specimen C - HA was not loaded beyond one and one-half percent drift because of electronic problems with the test apparatus.

6. Specimens C - LA - 1 and C - LA - 2, which had similar material properties but load programs with differences in the number of secondary load cycles, displayed similar behavior.

7.2. Analytical Program

An interactive computer program (OPUS) was developed to model the response of a reinforced concrete column subjected to varying axial and lateral loads. The program was implemented on a personal computer and written in Pascal. It performed the analysis in two phases. First, using section geometry and constitutive equations for concrete and steel, an axial load-moment-curvature relationship was generated for the column section. Then, using this section relationship and column end conditions, a monotonic response envelope was calculated. Two methods, one load-controlled and one deformation-controlled, were available for generating the envelope.

Column specimens tested in the experimental program were analyzed using the OPUS computer program. Results from the analysis were compared to the results of the experimental program. The computer model overestimated the lateral stiffness (due to bar slip which was not accounted for in the model), underestimated maximum moment at end sections (due to the method used to model the hinge regions of the column), and reasonably reproduced both the lateral strength and section stiffnesses of test specimens subjected to the first two load programs. For the third load program, maximum and minimum lateral loads were reproduced. Drift calculations were not correctly estimated, however, owing to the path dependency of concrete which was not accounted for in the analytical model.

7.3. Design Implications

Using traditional design tools, such as axial load-moment interaction diagrams, lateral load capacity of a reinforced concrete column can be conservatively calculated. This is done by determining the design axial load, obtaining the moment capacity corresponding to that axial load from the axial load-moment interaction diagram, and calculating the lateral load capacity from static equilibrium. For columns subjected to varying axial loads, both the maximum and minimum axial loads need to be checked in order to assure adequate column capacity. An additional factor of safety should be applied to axial load-moment combinations which lie above the balance point to ensure the strength of the column is never realized. The potential for redistribution of column forces is eliminated if the column sufferes a complete material failure rather than forming a flexural hinge, which is more likely to occur at axial load-moment combinations below the balance point.

Traditional design tools do not necessarily provide adequate estimates of column stiffness. Based on the limited number of specimens tested in this study, stiffness calculations performed using the product of the secant modulus of concrete and cracked transformed moment of inertia overestimate lateral stiffness of a reinforced concrete column subjected to severe lateral loads. This may result in an underestimate of interstory drift and second-order moments. A more realistic value for stiffness, based on measurements obtained from the experimental phase of this study, is $0.3E_cI_g$, where E_c is the secant modulus of concrete and I_g is the moment of inertia calculated from gross dimensions of the column section.

7.4. Future Research

Based on research conducted in this study, there are related areas which appear to be of interest for further study.

While computer programs are widely used in the analysis of reinforced concrete frames, there are few guidelines for choosing a realistic section stiffness to be used as input to these programs. Given the increasingly important role which computers are bound to play in the analysis and design of reinforced concrete structures, there is a need for development of rational approach to selection of member stiffnesses for use in analyzing building structures subjected to varying degrees of lateral load, and also a need for educating engineers in the proper selection of those stiffnesses.

Aspects of space frame behavior should be investigated. In all likelihood, some of the columns in a frame subjected to severe earthquake forces would experience biaxial bending together with axial load variations due to overturning effects. This would almost certainly be the case for corner columns in a space frame. The response of reinforced concrete columns to cyclic axial loads and biaxial bending is an area of interest for further experimental study.

REFERENCES

1. Richart, F. E., Brandtzaeg, A., and Brown, R. L., "A Study of the Failure of Concrete Under Combined Compressive Stresses," Bulletin No. 185, University of Illinois Engineering Experiment Station, Urbana, Nov. 1928, 102 pp.

2. Richart, F. E., Brandtzaeg, A., and Brown, R. L., "The Failure of Plain and Spirally Reinforced Concrete in Compression," Bulletin No. 190, University of Illinois Engineering Experiment Station, Urbana, Apr. 1929, 72 pp.

3. Richart, F. E., Draffin, J. O., Olson, T. A., and Heitman, "The Effect of Eccentric Loading, Protective Shells, Slenderness Ratios, and Other Variables in Reinforced Concrete Columns," Bulletin No. 308, University of Illinois Engineering Experiment Station, Vol. 46, No. 19, Oct. 1948, 128 pp.

4. Fowler, T. J., "Reinforced Concrete Columns Governed by Concrete Compression," PhD Dissertation, The University of Texas at Austin, Jan. 1966.

5. Kent, Dudley Charles, and Park, Robert, "Flexural Members with Confined Concrete," Journal of the Structural Division, ASCE, V. 97, ST7, July 1971, pp. 1969-1990.

6. Park, R., and Sampson, R. A., "Ductility of Reinforced Concrete Column Sections in Seismic Design," Journal of the American Concrete Institute, Proc. V. 69, No. 9, Sep. 1972, pp. 543-551.

7. Sheikh, S. A., and Uzumeri, S. M., "Strength and Ductility of Tied Concrete Columns," Journal of the Structural Division, ASCE, V. 106, ST5, May 1980, pp. 1079-1102.

8. Scott, B. D., Park, R., and Priestley, M. J. N., "Stress-Strain Behavior of Concrete Confined by Overlapping Hoops at Low and High Strain Rates," Journal of the American Concrete Institute, Proc. V. 79, No. 1, January-February 1982, pp. 13 - 27.

9. Sheikh, Shamim A., "A Comparative Study of Confinement Models," Journal of the American Concrete Institute, Proc. V. 79, No. 4, July-August 1982, pp. 296-306.

10. Hellesland, J., "A Study into the Sustained and Cyclic Load Behavior of Reinforced Concrete Columns," PhD Dissertation, Department of Civil Engineering, University of Waterloo, Waterloo, Ontario, Canada, August 1970.

11. Hirosawa, M., Ozaki, M., and Wakabayashi, M., "Experimental Study on Large Models of Reinforced Concrete Columns," Proceedings of Fifth World Conference on Earthquake Engineering, Rome, Italy, 1973, pp. 815-824.

12. Zagajeski, S. W., Bertero, V. V., and Boukamp, J. G., "Hysteretic Behavior of Reinforced Concrete Columns Subjected to High Axial and Cyclic Shear Forces," Report No. UCB/EERC 78/05, Earthquake Engineering Research Center, University of California, Berkeley, 1978.

13. Maruyama, K., Ramirez, H., and Jirsa, J. O., "Short RC Columns Under Bilateral Load Histories," Journal of the Structural Division, ASCE, V. 110, ST1, Jan. 1984, pp. 120-137.

14. Umehara, H., and Jirsa, J. O., "Short Rectangular RC Columns Under Bidirectional Loadings," Journal of the Structural Division, ASCE, V. 110, ST3, Mar. 1984, pp. 605-618.

15. Rabbat, B. G., Daniel, J. I., Weinmann, T. L., and Hanson, N. W., "Seismic Behavior of Lightweight and Normal Weight Concrete Columns," Journal of the American Concrete Institute, Proc. V. 83, No. 1, January-February 1986, pp. 69-79.

16. Ramirez, H., and Jirsa, J. O., "Effect of Axial Load on Shear Behavior of Short RC Columns Under Cyclic Lateral Deformations," PMFSEL Report, No. 80-1, The University of Texas at Austin, June 1980.

17. Gilbertson, N. D., and Moehle, J. P., "Experimental Study of Small-Scale RC Columns Subjected to Axial and Shear Force Reversals," Structural Research Series, No. 481, Report UILU-ENG-80-2015, University of Illinois, Urbana, July 1980, 95 pp.

18. Bedell, R. H., and Abrams, D. P., "Scale Relations for Concrete Columns," Structural Research Series No. 8302, Civil Engineering Department, University of Colorado, Boulder.

19. Davis, H., "Behavior of Columns at the Base of Concrete Frame Buildings," Master's Thesis, Department of Civil Engineering, University of Colorado, Boulder, 1983, 167 pp.

20. Epp, W. H., "The Effect of Loading Path on the Hysteretic Response of Concrete Columns," Master's Thesis, Department of Civil Engineering, University of Colorado, Boulder, 1984, 139 pp.

21. Kreger, M. E., "An Experimental/Analytical Investigation of the Dynamic Response of Staggered Structural Wall Systems," PhD Dissertation, Department of Civil Engineering, University of Illinois, Urbana, 1983, 321 pp.

22. ACI Committee 318, "Building Code Requirements for Reinforced Concrete (ACI 318-83)," American Concrete Institute, Detroit, 1983, 111 pp.

23. ACI Committee 318, "Commentary on Building Code Requirements for Reinforced Concrete (ACI 318-83)," American Concrete Institute, Detroit, 1983, 155 pp.

24. Mindess, Sidney, and Young, J. Francis, Concrete, Prentice-Hall, Englewood Cliffs, 1981.

25. Woodward, K. A., "Behavior Classification of Short Reinforced Concrete Columns Subjected To Cyclic Deformation." PhD Dissertation, The University of Texas, Austin, 1980, 334 pp.

26. Bett, B., "Behavior of Strengthened and/or Repaired Reinforced Concrete Columns Under Reversed Cyclic Deformations," Masters Thesis, The University of Texas, Austin, 1984, 122 pp.

27. SuperCalc4 User's Guide and Reference Manual, Computer Associates International, San Jose, 1986.

28. Pfrang, E. O., Siess, C. P., and Sozen, M. A., "Load-Moment-Curvature Characteristics of Reinforced Concrete Cross Sections," Journal of the American Concrete Institute, Proc. V. 61, No. 7, July 1964, pp. 763-777.

29. Alani, A. F., "Correlation of Physical Tests with Computer Simulation Models for Slab and Girder Bridge Systems," PhD Dissertation, The University of Texas at Austin, May 1971.

30. Berwanger, C., "Effect of Axial Load on the Moment-Curvature Relationship of Reinforced Concrete Members," Reinforced Concrete Columns, Publication SP-50, American Concrete Institute, Oct. 1975, pp. 263-288.

31. Ford, J. S., Chang, D. C., and Breen, J. E., "Behavior of Concrete Columns Under Controlled Lateral Deformation," Journal of the American Concrete Institute, Proc. V. 78, No. 1, January-February 1981, pp. 3 - 20.

32. Ford, J. S., Chang, D. C., and Breen, J. E., "Experimental and Analytical Modeling of Unbraced Multipanel Concrete Frames," Journal of the American Concrete Institute, Proc. V. 78, No. 1, January-February 1981, pp. 21 - 35.

33. Ford, J. S., "Behavior of Concrete Columns in Unbraced Multipanel Frames," PhD Dissertation, The University of Texas at Austin, Dec. 1977.

34. Wakabayashi, Minoru, Design of Earthquake Resistant Buildings, McGraw-Hill, New York, 1986, 309 pp.

35. Mattock, Alan H., "Rotational Capacity of Hinging Regions in Reinforced Concrete Beams," Flexural Mechanics of Reinforced Concrete, SP-12, American Concrete Institute/American Society of Civil Engineers, Detroit, 1965, pp. 143-181.

36. Corley, W. Gene, "Rotational Capacity of Reinforced Concrete Beams," Journal of the Structural Division, ASCE, V. 92, ST5, Oct. 1966, pp. 121 - 146.

37. Turbo Pascal User's Guide and Reference Manual, Borland International, Scotts Valley, California, 1984.

38. Pauw, A., "Static Modulus of Elasticity of Concrete as Affected by Density," Journal of the American Concrete Institute, Proc. V. 67, Nov. 1970, pp. 894-897.

39. International Conference of Building Officials, Uniform Building Code, Los Angeles, 1985.

40. ACI-ASCE Joint Committee 352, "Recommendations for Design of Beam-Column Joints in Monolithic Reinforced Concrete Structures," Journal of the American Concrete Institute, Proc. V. 82, No. 3, May-June 1985, pp. 266-283.

41. Blume, John A., Newmark, Nathan M., and Corning, Leo H., *Design of Multistory Reinforced Concrete Buildings for Earthquake Motions*, Portland Cement Association, Skokie, 1961, 318 pp.

42. MacGregor, J. G., Breen, J. E., and Pfrang, E. O., "Design of Slender Concrete Columns," *Journal of the American Concrete Institute*, Proc. V. 67, No. 1, January 1970, pp. 6-28.

43. Takeda, T., Sozen, M., and Nielsen, N., "Reinforced Concrete Response to Simulated Earthquakes," *Journal of the Structural Division, ASCE*, Vol. 96, ST12, December 1970, pp. 2557-2564.

44. Kreger, M. E., and Abrams, D. P., "Measured Hysteresis Relationships for Small-Scale Beam-Column Joints," *Structural Research Series*, No. 453, University of Illinois, Urbana, Aug. 1978.

45. Hayashi, S., Kokusho, S., Teramoto, E., and Osanai, T., "Study of the Bending Behavior of Reinforced Concrete Columns Under Changing Axial Force," *Bulletin D'Information*, No. 132, CEB Symposium on Structural Concrete Under Seismic Actions, April 1979, pp. 125-132.

46. Saiidi, M., and Sozen, M. A., "Simple and Complex Models for Nonlinear Seismic Response of Reinforced Concrete Structures," *Structural Research Series*, No. 465, Report UILU-ENG-79-2013, University of Illinois, Urbana, August 1979, 188 pp.

47. Selna, L., Martin, I., Park, R., Wyllie, L., "Strong and Tough Concrete Columns for Seismic Forces,"*Journal of the Structural Division, ASCE*, V. 106, ST8, August 1980, pp. 1717-1734.

48. Priestley, M. J. N., and Park, R., "Strength and Ductility of Bridge Substructures," *RRU Bulletin*, No. 71, National Roads Board, Wellington, New Zealand, 1984, 120 pp.

49. Keshavarzian, M., and Schnobrich, W. C., "Computed Nonlinear Seismic Response of RC Wall-Frame Structures," *Structural Research Series*, No. 515, Report UILU-ENG-84-2004, University of Illinois, Urbana, May 1984, 219 pp.

50. Moehle, J. P., "Seismic Response of Vertically Irregular Structures," *Journal of the Structural Division, ASCE*, V. 110, ST9, Sept. 1984, pp. 2002-2014.

51. Hellesland, J., Choudhury, D. and Scordelis, A. C., "Nonlinear Analysis and Design of RC Bridge Columns Subjected to Imposed Deformations," Report No. UCB/SESM-85/03, University of California, Berkeley, April 1985, 305 pp.

52. Klingner, R. E., Class Notes, Course on Reinforced Concrete Members, The University of Texas at Austin, August 1985.

VITA

Leo Edward Linbeck, III was born in Houston, Texas, on 30 November, 1961, the son of Constance Belle Baird Linbeck and Leo Edward Linbeck, Jr. After graduating from St. Thomas High School in Houston in 1979, he entered the University of Notre Dame, South Bend, Indiana. In May 1984, he received a Bachelor of Science in Civil Engineering and a Bachelor of Arts in the Program of Liberal Studies, Notre Dame's Great Books program. He was employed for three summers as an ironworker by Linbeck Construction Corporation, one summer as a structural engineer for CBM Engineers, Inc., and one summer as a structural engineer for Walter P. Moore and Associates. In August 1984, he entered the Graduate School at The University of Texas at Austin. Since that time, he has been concerned with the study of reinforced concrete columns in buildings subjected to earthquake loadings under the direction of Dr. Michael E. Kreger. He has a strong interest in the role of the engineer in modern society, and the development of civil engineering software for the personal computer.

Permanent address: 3404 Chevy Chase
Houston, Texas 77019

This thesis was typed by the author.

APPENDIX

The following pages are a listing of the OPUS column analysis program. It was written in Turbo Pascal, a Pascal compiler written by Borland International, and printed using Sideways, a landscape printing program from Funk Software.

```
     lt_blue    = 9;
     lt_green   =10;
     lt_cyan    =11;
     lt_red     =12;
     lt_magenta =13;
     yellow     =14;
     white      =15;
     max_picture_size = 2000;
     max_no_ax_loads = 50;
     alpha_increment = 90;
     beta_increment = 90;
     x_increment = 10;
     y_increment = 10;
     max_no_elem = 60;
     max_no_load_steps = 30;

(*---------------------------------------------------------------------------*)

TYPE

    MESSAGE                  = STRING[80] ;

    REGISTERS                = RECORD
                                    AX, BX, CX, DX, BP, SI, DI, DS, ES, FLAGS : INTEGER
                                 END ;

    pixel = record
          x : real;
          y : real;
          z : real;
          c : integer;
        end;

    data = array[0..99] of real;

    layer = record
          area : real;
          depth : real;
        end (* layer *);

    slice_data = record
          t : real;
          d : real;
```

```
      conf_per : real;
    end (* slice_data *);

prop_rec = record
         filename : string[20];
         length : real;
         width : real;
         depth : real;
         conc_str : real;
         steel_yield : real;
         steel_ult : real;
         layers : integer;
         steel : array [1..3] of layer;
         tie_spacing : real;
         tie_bar_area : real;
         tie1_no_legs : integer;
         tie2_no_legs : integer;
         ult_conc_strain : real;
         inter_flag : boolean;
         strain_at_ult : real;
         strain_hard : real;
         no_slices : integer;
       end (* prop_rec *);

calc_rec = record
         long_rho : real;
         trans_rho : real;
         eff_depth : real;
         area : real;
         neutral_axis : real;
         uncr_mom_iner : real;
         cr_mom_iner : real;
         crack_moment : real;
         conc_mod : real;
         conc_tens : real;
         beta : real;
         mod_rat : real;
         conf_depth : real;
         conf_width : real;
         bal_mom : real;
         bal_ax : real;
         bal_kd : real;
         ult_comp : real;
         ult_tens : real;
         ult_steel : real;
         yield_strain : real;
         mod_sh : real;
         e_0 : real;
         e_50_h : real;
         e_50_u : real;
         k : real;
         z_conf : real;
         z_unconf : real;
         ax_min : real;
         ax_min_curv : real;
         kd1_ax : real;
         kd2_ax : real;
         kd3_ax : real;
       end (* calc_rec *);

load_rec = record
         filename : string[20];
         load_control : boolean;
         lateral_type : integer;
         axial_type : integer;
         lateral : array[1..10] of real;
         axial : array[1..10] of real;
       end (* load_rec *);

sect_rec = record
         axial : real;
         no_points : integer;
         curvature : array[1..40] of real;
         moment : array[1..40] of real;
       end (* sect_rec *);

sect_file_rec = record
        filename : string[20];
        no_ax_loads : integer;
        ductility : real;
        data : array[1..max_no_ax_loads] of sect_rec;
      end (* sect_file_rec *);

defl_rec = record
         filename : string[20];
         no_elem : integer;
         toler : real;
         prop_filename : string[20];
         load_filename : string[20];
         sect_filename : string[20];
         auto_mode : boolean;
         load_control : boolean;
         lat_load : data;
         ax_load : data;
         deflection : data;
         no_load_steps : integer;
         ratio : real;
         hinge_length : integer;
       end (* defl_rec *);

int_array = array [1..50] of real;

inter_rec = record
         filename : string[20];
         prop_filename : string[20];
         moment : int_array;
         axial : int_array;
         curvature : int_array;
       end (* inter_rec *);

picture_pointer = ^picture_type;

picture_type = record
         data : array[0..max_picture_size] of pixel;
       end (* picture_type *);

seg_stuff = record
         moment : array[0..max_no_elem] of real;
         deflection : array[0..max_no_elem] of real;
         rotation : array[0..max_no_elem] of real;
         curvature : array[0..max_no_elem] of real;
       end (* seg_stuff *);

seg_data_pointer = ^ seg_data_type;

seg_data_type = record
         data : array[0..max_no_load_steps] of seg_stuff;
       end (* seg_data_type *);
```

```
(*----------------------------------------------------------------------------*)

VAR

   KEY_PRESSED   : BYTE    ;
   KEY           : BYTE    ;
   I , J , K     : INTEGER ;
   MESS          : MESSAGE ;
   DONE          : BOOLEAN ;
   old_ar_row,old_ar_col : integer;
   prop_file : file of prop_rec;
   load_file : file of load_rec;
   sect_file : file of sect_file_rec;
   defl_file : file of defl_rec;
   inter_file : file of inter_rec;
   menu_char,char_code : byte;
   ans : byte;
   prop : prop_rec;
   load : load_rec;
   sect : sect_file_rec;
   defl : defl_rec;
   calc : calc_rec;
   inter : inter_rec;
   curr_box,curr_ax_load : integer;
   prop_row,prop_col : array[1..18] of integer;
   load_row,load_col,load_arrow : array[1..22] of integer;
   sect_row,sect_col : array[1..44] of integer;
   load_type_descr : array[1..3] of string[12];
   no_load_boxes,side_jump : integer;
   no_load_steps,load_step : integer;
   graph_no,no_steps : integer;
   x_data,y_data : array[0..99] of real;
   converge : boolean;
   plast_cen : real;
   overflow : boolean;
   error : boolean;
   alpha,beta : real;
   lat_boxes,ax_boxes : integer;
   kd : real;
   scale_factor : real;
   title_color,data_color,descr_color,menu_color : integer;
   subtitle_color,error_color,mess_color,help_color : integer;
   index_color : integer;
   curr_page : integer;
   rec    : registers;
   xaddr  : array[0..639] of integer;
   yaddr  : array[0..349] of integer;
   point  : array[0..639] of integer;
   pen_color : integer;
   picture : picture_pointer;
   seg : seg_data_pointer;
   rotate : array[1..13] of real;
   scale,translate,initial_size,center : pixel;
   picture_size,link_no,picture_flag : integer;
   max,min : pixel;
   cos_a,sin_a,cos_b,sin_b,cos_c,sin_c : real;
   alpha_st,beta_st : string[3];
   x_mesh_fineness,y_mesh_fineness : integer;
   axes : array[1..3] of pixel;
   color : array[1..5] of integer;
   printout : boolean;
   plot_mom,plot_curv : real;
   x_abs_window,y_abs_window : integer;
   x_rel_window,y_rel_window : real;
   x_abs_offset,y_abs_offset : integer;
   boundary : boolean;

(*---------------------------------------------------------------------------*)

PROCEDURE CLS (STROW,STCOL,ENDROW,ENDCOL,COLOR : BYTE  ) ;
   VAR
      REGS : REGISTERS ;
   BEGIN
      WITH REGS DO
         BEGIN
            AX := $0700 ;
            BX := (COLOR  * 256) + 0      ;
            CX := ((STROW-1)  * 256) + STCOL-1 ;
            DX := ((ENDROW-1) * 256) + ENDCOL-1
         END ;
      INTR($10,REGS)
   END ;

(*---------------------------------------------------------------------------*)

PROCEDURE CURSOR (ROW,COL : BYTE) ;
```

```
   VAR
      REGS : REGISTERS ;
   BEGIN
      WITH REGS DO
         BEGIN
            AX := $0200 ;
            BX := CURR_PAGE   * 256 ;
            DX := (ROW * 256) + COL
         END ;
      INTR($10,REGS)
   END ;

(*-----------------------------------------------------------------------------*)

PROCEDURE GETC (VAR KEY : BYTE) ;
   VAR
      REGS : REGISTERS ;
   BEGIN
      REGS.AX := $0000 ;
      INTR($16,REGS)   ;
      KEY         := REGS.AX AND $00FF ;
      KEY_PRESSED := (REGS.AX AND $FF00) DIV 256
   END ;

(*-----------------------------------------------------------------------------*)

PROCEDURE HRCOL(ROW, COL, COLOR : INTEGER ; MESS : MESSAGE) ;
   VAR
      I, J, K : INTEGER    ;
      REGS    : REGISTERS  ;
   BEGIN
      K := COL-1 ;
      J := ORD(MESS[0]) ;
      FOR I := 1 TO J DO
         BEGIN
            CURSOR(ROW-1,K) ;
            WITH REGS DO
               BEGIN
                  AX  := (9 * 256) + ORD(MESS[I]) ;
                  BX  := (CURR_PAGE * 256) + COLOR ;
                  CX  := $0001 ;
                  INTR($10,REGS) ;
                  K   := K + 1
               END
         END
   END ;

(*---------------------------------------------------------------------------*)

FUNCTION NUMERIC(CH : BYTE) : BOOLEAN ;
   BEGIN
      NUMERIC := FALSE ;
      IF ((CH >= 48) AND (CH <= 57))
         THEN NUMERIC := TRUE
   END ;

(*---------------------------------------------------------------------------*)

FUNCTION SPECIAL(CB : BYTE) : BOOLEAN ;
   BEGIN
      SPECIAL := FALSE ;
      IF   (CB = 13) OR
           (KEY_PRESSED = UP_ARROW) OR (KEY_PRESSED = DN_ARROW) OR
           (KEY_PRESSED = LT_ARROW) OR (KEY_PRESSED = RT_ARROW) OR
           (KEY_PRESSED = F1)  OR  (KEY_PRESSED = F2)  OR
           (KEY_PRESSED = F3)  OR  (KEY_PRESSED = F4)  OR
           (KEY_PRESSED = F5)  OR  (KEY_PRESSED = F6)  OR
           (KEY_PRESSED = F7)  OR  (KEY_PRESSED = F8)  OR
           (KEY_PRESSED = F9)  OR  (KEY_PRESSED = ESC)
        THEN SPECIAL := TRUE
   END ;

(*---------------------------------------------------------------------------*)

PROCEDURE GETNUM(VAR NUMBER : REAL ; WL, FL : INTEGER)  ;
   VAR
      ENTRY : ARRAY [1..10] OF CHAR ;
      I , J : INTEGER ;
      WHOLE , FRAC , MUL : REAL ;
      WLEN , FLEN  : INTEGER ;
      DONE , WDONE , FDONE : BOOLEAN ;
      CB : BYTE ;
      CH : CHAR ;
      SIGN : BOOLEAN;
   BEGIN
      SIGN := FALSE;
```

```
FOR I := 1 TO 10 DO ENTRY[I] := ' ' ;
WLEN := 0 ;
IF (WL > 0) THEN WDONE := FALSE  ELSE WDONE := TRUE ;
DONE  := FALSE ;
WHILE NOT DONE DO
   BEGIN
      FLEN := 0 ;
      IF FL > 0 THEN FDONE := FALSE ELSE FDONE := TRUE ;
      WHILE NOT WDONE DO
         BEGIN
            GETC(CB) ;
            IF NUMERIC(CB)
               THEN
                  BEGIN
                     WLEN := WLEN + 1 ;
                     ENTRY[WLEN] := CHR(CB) ;
                     WRITE(ENTRY[WLEN]:1) ;
                     IF WLEN = WL THEN IF FDONE THEN BEGIN
                                                        WDONE := TRUE ;
                                                        DONE  := TRUE
                                                     END
                                                ELSE BEGIN
                                                        WDONE := TRUE ;
                                                        WRITE('.')
                                                     END
                  END
               ELSE IF SPECIAL(CB)
                       THEN BEGIN
                               WDONE := TRUE ;
                               FDONE := TRUE ;
                               DONE  := TRUE
                            END
                        ELSE IF (CB = 8) AND ((WLEN > 0) or ((WLEN=0) and (SIGN=TRUE)))
                                THEN BEGIN
                                        WRITE(CHR(CB):1) ;
                                        WRITE(' ')  ;
                                        WRITE(CHR(CB):1) ;
                                        WLEN := WLEN - 1 ;
                                        IF WLEN = 0 THEN SIGN := FALSE;
                                     END
                                ELSE IF (CB = 46) AND (WLEN > 0)
                                        THEN BEGIN
                                                IF (FL > 0)
```

```
                                            THEN WRITE(CHR(CB):1) ;
                                          WDONE := TRUE
                                       END
                                  ELSE IF CB = 45
                                          THEN BEGIN
                                                  SIGN := TRUE;
                                                  WRITE(CHR(CB):1);
                                               END;
   END ;
WHILE NOT FDONE DO
   BEGIN
      GETC(CB) ;
      IF NUMERIC(CB) THEN BEGIN
                             FLEN := FLEN + 1 ;
                             ENTRY[HLEN+FLEN] := CHR(CB) ;
                             WRITE(ENTRY[HLEN+FLEN]:1) ;
                             IF FLEN = FL THEN BEGIN
                                                  FDONE := TRUE ;
                                                  DONE  := TRUE
                                               END
                          END
                     ELSE
                        IF SPECIAL(CB)
                           THEN BEGIN
                                   FDONE := TRUE ;
                                   DONE  := TRUE
                                END
                           ELSE
                              IF (CB = 8)
                                 THEN
                                    IF FLEN > 0
                                       THEN
                                          BEGIN
                                             WRITE(CHR(CB):1) ;
                                             WRITE(' ')  ;
                                             WRITE(CHR(CB):1) ;
                                             FLEN := FLEN - 1
                                          END
                                       ELSE
                                          BEGIN
                                             WRITE(CHR(CB):1) ;
                                             WRITE(CHR(CB):1) ;
                                             WRITE('  ') ;
```

```
                                                           WRITE(CHR(CB):1) ;
                                                           WRITE(CHR(CB):1) ;
                                                           WDONE := FALSE ;
                                                           WLEN := WLEN - 1 ;
                                                           FDONE := TRUE
                                                        END;
               END
         END ;
      IF (WL > 0) AND (WLEN > 0)
         THEN
            BEGIN
               J := WLEN ;
               WHOLE := 0.0 ;
               MUL := 1.0 ;
               WHILE J > 0 DO
                  BEGIN
                     WHOLE := WHOLE + (TRUNC(ORD(ENTRY[J])-48))*MUL ;
                     MUL := MUL * 10.0 ;
                     J := J - 1
                  END;
               FRAC := 0.0 ;
               MUL := 10.0 ;
               J := WLEN + 1;
               WHILE (J <= WLEN+FLEN) DO
                  BEGIN
                     FRAC := FRAC + (TRUNC(ORD(ENTRY[J])-48))/MUL ;
                     MUL := MUL * 10.0 ;
                     J := J + 1
                  END ;
               NUMBER := WHOLE + FRAC;
               IF SIGN THEN NUMBER := -NUMBER;
            END
   END ;

(*-----------------------------------------------------------------------------*)

FUNCTION F_TO_A(VLU : REAL ; WL , FLEN : INTEGER) : MESSAGE ;
   VAR
      SIGN , I , J , K , WLEN  : INTEGER ;
      TEMP1 , TEMP2 , FACTOR  : REAL    ;
      ALPHA : MESSAGE ;
   BEGIN
      SIGN  := 1 ;
```

```
IF VLU < 0.0 THEN BEGIN
                     VLU  := - VLU ;
                    SIGN := -1
                  END ;
TEMP1 := VLU ;
WLEN  := 0 ;
WHILE NOT (TEMP1 < 1.0) DO BEGIN
                              TEMP1 := TEMP1 / 10.0 ;
                              WLEN  := WLEN + 1
                           END ;
TEMP1 := VLU ;
TEMP2 := VLU ;
FOR J := 1 TO WLEN DO BEGIN
                         K := J ;
                         FACTOR := 1.0 ;
                         WHILE K < WLEN DO BEGIN
                                              FACTOR := FACTOR * 10.0 ;
                                              K := K + 1
                                           END ;
                         TEMP1 := TEMP2 / FACTOR ;
                         ALPHA[J] := CHR(TRUNC(TEMP1)+48) ;
                         TEMP2 := TEMP2 - (FACTOR * TRUNC(TEMP1))
                      END ;
IF WLEN = 0
   THEN BEGIN
           FOR J := 1 TO WL DO
              BEGIN
                 WLEN := WLEN + 1 ;
                 ALPHA[WLEN] := '0'
              END
        END ;
ALPHA[WLEN+1] := '.' ;
FOR I := 1 TO FLEN DO BEGIN
                         TEMP2 := TEMP2 * 10.0 ;
                         ALPHA[WLEN+1+I] := CHR(TRUNC(TEMP2)+48) ;
                         TEMP2 := TEMP2 - TRUNC(TEMP2)
                      END ;
ALPHA[0] := CHR(WLEN+1+FLEN) ;
IF SIGN = -1 THEN BEGIN
                     FOR I := (WLEN+1+FLEN) DOWNTO 1 DO ALPHA[I+1] := ALPHA[I] ;
                     ALPHA[1] := '-' ;
                     ALPHA[0] := CHR(WLEN+1+FLEN+1)
                  END ;
```

```
      F_TO_A   := ALPHA
   END ;

(*---------------------------------------------------------------------------*)

FUNCTION I_TO_A(IVLU : INTEGER) : MESSAGE ;
   VAR
      SIGN   , I , J , K , WLEN  : INTEGER ;
      TEMP1 , TEMP2 , FACTOR  : INTEGER    ;
      ALPHA : MESSAGE ;
   BEGIN
      IF IVLU < 0 THEN BEGIN
                         IVLU := - IVLU ;
                         SIGN := -1
                      END ;
      IF IVLU = 0 THEN BEGIN
                         ALPHA[0] := CHR(1) ;
                         ALPHA[1] := '0' ;
                         I_TO_A   := ALPHA
                      END
                 ELSE BEGIN
                         TEMP1 := IVLU ;
                         WLEN  := 0 ;
                         WHILE NOT (TEMP1 < 1) DO BEGIN
                                                     TEMP1 := TEMP1 DIV 10 ;
                                                     WLEN  := WLEN + 1
                                                 END ;
                         TEMP1 := IVLU ;
                         TEMP2 := IVLU ;
                         FOR J := 1 TO WLEN DO BEGIN
                                                  K := J ;
                                                  FACTOR := 1 ;
                                                  WHILE K < WLEN DO BEGIN
                                                                       FACTOR := FACTOR * 10 ;
                                                                       K := K + 1
                                                                    END ;
                                                  TEMP1 := TEMP2 DIV FACTOR ;
                                                  ALPHA[J] := CHR(TEMP1+48) ;
                                                  TEMP2 := TEMP2 - (FACTOR * TEMP1)
                                               END ;
                         ALPHA[0] := CHR(WLEN) ;
                         IF SIGN = -1 THEN BEGIN
                                              FOR I := WLEN DOWNTO 1 DO ALPHA[I+1] := ALPHA[I] ;
                                              ALPHA[1] := '-' ;
                                              ALPHA[0] := CHR(WLEN+1)
                                           END ;
                         I_TO_A   := ALPHA
                      END
   END ;

(*----------------------------------------------------------------------------*)

procedure switch(var v1,v2:integer);
var
  s : integer;
begin
  s := v1; v1 := v2; v2 := s
end;

(*----------------------------------------------------------------------------*)

procedure set_color(c:integer);
begin
  pen_color := c;
  inline($A1/pen_color/      {mov  ax,pen_color  - get color value}
         $BA/$03C4/          {mov  dx,$03C4      - sequencer port}
         $B4/$02/            {mov  ah,$02        - addr map mask register}
         $86/$C4/            {xchg al,ah }
         $EE/                {out  dx,al }
         $42/                {inc  dx    }
         $86/$C4/            {xchg al,ah }
         $EE)                {out  dx,al }
end;

(*----------------------------------------------------------------------------*)

procedure write_dot(tot,pt:integer);
{Note TURBO bug! tot and pt would generate $0006 and $0004}
begin
  inline($B8/$A000/          {mov  ax,$A000      - get map base}
         $8E/$C0/            {mov  es,ax }
         $BA/$03CE/          {mov  dx,$03CE      - graphics chip port}
         $B0/$08/            {mov  al,$08        - addr map mask register}
         $8B/$5E/$06/        {mov  bx,tot[bp]    - get map offset}
         $8A/$66/$04/        {mov  ah,pt[bp]     - get bit mask}
         $EE/                {out  dx,al }
         $42/              {inc  dx    }
         $86/$C4/          {xchg al,ah }
         $EE/              {out  dx,al          - set bit mask}
         $4A/              {dec  dx    }
         $26/$8A/$07/      {mov  al,es:[bx]     - latch byte}
         $B0/$FF/          {mov  al,0FFh }
         $26/$88/$07/      {mov  es:[bx],al     - write dot}
         $86/$C4/          {xchg al,ah }
         $EE/              {out  dx,al }
         $42/              {inc  dx    }
         $86/$C4/          {xchg al,ah }
         $EE)              {out  dx,al          - turn all bits back on}
end;

(*-----------------------------------------------------------------------------*)

procedure plotc(x,y:integer);
begin
  write_dot(xaddr[x]+yaddr[y],point[x])
end;

(*-----------------------------------------------------------------------------*)

procedure drawc(xx1,yy1,xx2,yy2:integer);
var
  lg_delta, sh_delta, cycle, lg_step, sh_step : integer;
begin
  lg_delta := xx2-xx1; sh_delta := yy2-yy1;
  if lg_delta < 0 then
    begin
      lg_delta := -lg_delta; lg_step := -1
    end
  else lg_step := 1;
  if sh_delta < 0 then
    begin
      sh_delta := -sh_delta; sh_step := -1
    end
  else sh_step := 1;
  if sh_delta < lg_delta then
    begin
      cycle := lg_delta shr 1;
      while xx1 <> xx2 do
        begin
```

```
          write_dot(xaddr[xx1]+yaddr[yy1],point[xx1]);
          xx1 := xx1 + lg_step; cycle := cycle + sh_delta;
          if cycle > lg_delta then
            begin
              yy1 := yy1 + sh_step; cycle := cycle-lg_delta
            end
        end
    end
  else
    begin
      cycle := sh_delta shr 1;
      switch(lg_delta,sh_delta);
      switch(lg_step,sh_step);
      while yy1 <> yy2 do
        begin
          write_dot(xaddr[xx1]+yaddr[yy1],point[xx1]);
          yy1 := yy1 + lg_step; cycle := cycle + sh_delta;
          if cycle > lg_delta then
            begin
              cycle := cycle - lg_delta; xx1 := xx1 + sh_step
            end
        end
    end
end;

(*---------------------------------------------------------------------------*)

procedure blank_line(linenum : integer);
          begin
               gotoXY(1,linenum);
               cls(linenum,1,linenum,80,0);
          end;

(*---------------------------------------------------------------------------*)

procedure draw_box(x1,y1,x2,y2 : integer);
          var
             mode,width,page : byte;
          begin
             set_color(off_white);
             drawc(x1,y1,x1,y2);
             drawc(x1,y2,x2,y2);
             drawc(x2,y2,x2,y1);
          drawc(x2,y1,x1,y1);
        end;

(*----------------------------------------------------------------------*)

procedure draw_arrow(r,c : integer);
        var x,y : integer;
        begin
            gotoXY(old_ar_col,old_ar_row);
            write(' ');
            y := r * 14;
            x := c * 8;
            set_color(yellow);
            drawc(x-7,y-7,x-1,y-7);
            drawc(x-7,y-8,x-1,y-8);
            drawc(x-1,y-8,x-5,y-1);
            drawc(x-1,y-7,x-5,y-14);
            old_ar_row := r;
            old_ar_col := c;
        end;

(*----------------------------------------------------------------------*)

overlay procedure save_inter_file;
        begin
          if prop.inter_flag
            then begin
              assign(inter_file,inter.filename);
              rewrite(inter_file);
              write(inter_file,inter);
              close(inter_file);
            end;
        end;

(*----------------------------------------------------------------------*)

overlay function load_inter_file : boolean;
        var ext_pos : integer;
        begin
          error := false;
          inter.prop_filename := prop.filename;
          ext_pos := pos('.prp',prop.filename);
          inter.filename := copy(prop.filename,1,ext_pos);
```

```
             inter.filename := concat(inter.filename,'int');
             if prop.inter_flag
                then begin
                   assign(inter_file,inter.filename);
{$I-}
                   reset(inter_file);
{$I+}
                   if IOresult = 0
                      then begin
                         read(inter_file,inter);
                         close(inter_file);
                         load_inter_file := true;
                      end
                      else load_inter_file := false;
                end
                else load_inter_file := false;
          end;

(*------------------------------------------------------------------------------*)

overlay procedure load_prop_file;
          var complete : boolean;
          begin
             complete := false;
             repeat
                blank_line(24);
                wrcol(24,1,mess_color,'Properties input file name : ');
                getc(char_code);
                if key_pressed<>CARRIAGE
                   then begin
                        write(chr(char_code));
                        readln(prop.filename);
                        prop.filename := concat(chr(char_code),prop.filename,'.prp');
                        assign(prop_file,prop.filename);
{$I-}
                        reset(prop_file);
{$I+}
                        if IOresult = 0
                           then complete := true
                           else begin
                              blank_line(25);
                              wrcol(25,1,mess_color,'Invalid file name; try again');
                              complete := false;
                           end;
                   end
                   else complete := true;
             until complete;
             if key_pressed <> CARRIAGE
                then if key_pressed <> F1
                   then begin
                      read(prop_file,prop);
                      close(prop_file);
                   end;
             blank_line(25);
             blank_line(24);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure save_prop_file;
          var
             doit : boolean;
             temp : string[20];
          begin
               blank_line(24);
               wrcol(24,1,mess_color,'Properties output file name : ');
               getc(char_code);
               if key_pressed<>CARRIAGE then
                  begin
                       write(chr(char_code));
                       readln(temp);
                       temp := concat(chr(char_code),temp,'.prp');
                       if temp <> prop.filename
                          then prop.inter_flag := false;
                       prop.filename := temp;
                       assign(prop_file,prop.filename);
{$I-}
                       reset(prop_file);
{$I+}
                       if IOresult=0
                          then begin
                             blank_line(24);
                             mess := concat('Overwrite current ',prop.filename,'?');
                             wrcol(24,1,mess_color,mess);
                             getc(ans);
                             if Upcase(chr(ans))='Y'
```

```
                                then doit := true
                                else doit := false;
                          end
                          else doit := true;
                  end;
               close(prop_file);
               if doit
                  then begin
                     rewrite(prop_file);
                     write(prop_file,prop);
                     close(prop_file);
                  end (* if doit *);
               blank_line(24);
          end;

(*----------------------------------------------------------------------------*)

overlay procedure save_load_file;
          var doit : boolean;
          begin
               blank_line(24);
               wrcol(24,1,mess_color,'Loading output file name : ');
               getc(char_code);
               if key_pressed<>CARRIAGE then
                  begin
                       write(chr(char_code));
                       readln(load.filename);
                       load.filename := concat(chr(char_code),load.filename,'.lod');
                       assign(load_file,load.filename);
{$I-}
                       reset(load_file);
{$I+}
                       if IOresult=0
                          then begin
                             blank_line(24);
                             mess := concat('Overwrite current ',load.filename,'?');
                             wrcol(24,1,mess_color,mess);
                             getc(ans);
                             if Upcase(chr(ans))='Y'
                                then doit := true
                                else doit := false;
                          end
                          else doit := true;
                       close(load_file);
                       if doit
                          then begin
                             rewrite(load_file);
                             write(load_file,load);
                             close(load_file);
                          end (* if doit *);
                  end;
               blank_line(24);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure load_load_file;
          var complete : boolean;
          begin
             complete := false;
             repeat
                blank_line(24);
                wrcol(24,1,mess_color,'Loading input file name : ');
                getc(char_code);
                if key_pressed<>CARRIAGE
                   then begin
                      write(chr(char_code));
                      readln(load.filename);
                      load.filename := concat(chr(char_code),load.filename,'.lod');
                      assign(load_file,load.filename);
{$I-}
                      reset(load_file);
{$I+}
                      if IOresult = 0
                         then complete := true
                         else begin
                            blank_line(25);
                            wrcol(25,1,mess_color,'Invalid file name; try again');
                            complete := false;
                         end;
                   end
                   else complete := true;
                until complete;
                if key_pressed <> CARRIAGE
                   then if key_pressed <> F1
                      then begin
```

```
                         read(load_file,load);
                         close(load_file);
                      end;
                blank_line(25);
                blank_line(24);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure load_section_file;
          var complete : boolean;
          begin
             complete := false;
             repeat
                blank_line(24);
                wrcol(24,1,mess_color,'Section input file name : ');
                getc(char_code);
                if key_pressed<>CARRIAGE
                   then begin
                      write(chr(char_code));
                      readln(sect.filename);
                      sect.filename := concat(chr(char_code),sect.filename,'.sec');
                      assign(sect_file,sect.filename);
{$I-}
                      reset(sect_file);
{$I+}
                      if IOresult = 0
                         then complete := true
                         else begin
                            blank_line(25);
                            wrcol(25,1,error_color,'Invalid file name; try again');
                            complete := false;
                         end;
                   end
                   else complete := true;
             until complete;
             if (key_pressed<>CARRIAGE)
                then if (key_pressed<>F1)
                   then begin
                      read(sect_file,sect);
                      close(sect_file);
                   end;
             blank_line(25);
             blank_line(24);
          end;

(*-----------------------------------------------------------------------------*)

overlay procedure save_section_file;
          var doit : boolean;
          begin
               blank_line(24);
               wrcol(24,1,mess_color,'Section output file name : ');
               getc(char_code);
               if key_pressed<>CARRIAGE then
                  begin
                       write(chr(char_code));
                       readln(sect.filename);
                       sect.filename := concat(chr(char_code),sect.filename,'.sec');
                       assign(sect_file,sect.filename);
{$I-}
                       reset(sect_file);
{$I+}
                       if IOresult=0
                          then begin
                             blank_line(24);
                             mess := concat('Overwrite current ',sect.filename,'?');
                             wrcol(24,1,error_color,mess);
                             getc(ans);
                             if Upcase(chr(ans))='Y'
                                then doit := true
                                else doit := false;
                          end
                          else doit := true;
                       close(sect_file);
                       if doit
                          then begin
                             rewrite(sect_file);
                             write(sect_file,sect);
                             close(sect_file);
                          end (* if doit *);
                  end;
               blank_line(24);
          end;

(*-----------------------------------------------------------------------------*)
```

```
overlay procedure sect_to_sc3;
          var
             filename : string[20];
             csv : text;
             ext_pos : integer;
             i,j : integer;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'...generating ASCII file, please wait...');
             ext_pos := pos('.sec',sect.filename)-1;
             filename := copy(sect.filename,1,ext_pos);
             filename := concat(filename,'_s','.csv');
             assign(csv,filename);
             rewrite(csv);
             for i := 1 to sect.no_ax_loads do with sect.data[i] do
                for j := 1 to no_points do
                   writeln(csv,f_to_a(axial,6,3),',',
                           f_to_a(moment[j],8,3),',',
                           f_to_a(curvature[j],1,9));
             close(csv);
             blank_line(25);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure inter_to_csv;
          var
             filename : string[20];
             csv : text;
             ext_pos : integer;
             i,j : integer;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'...generating ASCII file, please wait...');
             ext_pos := pos('.int',inter.filename)-1;
             filename := copy(inter.filename,1,ext_pos);
             filename := concat(filename,'_i','.csv');
             assign(csv,filename);
             rewrite(csv);
             for j := 1 to 50 do with inter do
                writeln(csv,f_to_a(axial[j],6,3),',',
                            f_to_a(moment[j],8,3));
             for j := 50 downto 1 do with inter do
                writeln(csv,f_to_a(axial[j],6,3),',',
                           f_to_a(-moment[j],8,3));
             close(csv);
             blank_line(25);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure load_defl_file;
          var complete : boolean;
          begin
             complete := false;
             repeat
                blank_line(24);
                wrcol(24,1,mess_color,'Deflection input file name : ');
                getc(char_code);
                if key_pressed<>CARRIAGE
                   then begin
                        write(chr(char_code));
                        readln(defl.filename);
                        defl.filename := concat(chr(char_code),defl.filename,'.dfl');
                        assign(defl_file,defl.filename);
{$I-}
                        reset(defl_file);
{$I+}
                        if IOresult = 0
                           then complete := true
                           else begin
                              blank_line(25);
                              wrcol(25,1,error_color,'Invalid file name; try again');
                              complete := false;
                           end;
                   end
                   else complete := true;
             until complete;
             if key_pressed <> CARRIAGE
                then if key_pressed <> F1
                   then begin
                      read(defl_file,defl);
                      close(defl_file);
                   end;
             blank_line(25);
             blank_line(24);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure save_defl_file;
          var doit : boolean;
          begin
               blank_line(24);
               wrcol(24,1,mess_color,'Deflection output file name : ');
               getc(char_code);
               if key_pressed<>CARRIAGE then
                  begin
                       write(chr(char_code));
                       readln(defl.filename);
                       defl.filename := concat(chr(char_code),defl.filename,'.dfl');
                       assign(defl_file,defl.filename);
{$I-}
                       reset(defl_file);
{$I+}
                       if IOresult=0
                          then begin
                             blank_line(24);
                             mess := concat('Overwrite current ',defl.filename,'?');
                             wrcol(24,1,error_color,mess);
                             getc(ans);
                             if Upcase(chr(ans))='Y'
                                then doit := true
                                else doit := false;
                          end
                          else doit := true;
                       close(defl_file);
                       if doit
                          then begin
                             rewrite(defl_file);
                             write(defl_file,defl);
                             close(defl_file);
                          end (* if doit *);
                  end;
               blank_line(24);
          end;

(*---------------------------------------------------------------------------*)
```

```
function get_props : boolean;
          var
             ext_pos : integer;
          begin
             draw_box(0,0,x_abs_window-1,y_abs_window-1);
             load_prop_file;
             ext_pos := pos('.prp',prop.filename);
             sect.filename := copy(prop.filename,1,ext_pos);
             inter.filename := concat(sect.filename,'int');
             sect.filename := concat(sect.filename,'sec');
             assign(sect_file,sect.filename);
{$I-}
             reset(sect_file);
{$I+}
             if IOresult = 0
                then begin
                   blank_line(24);
                   mess := concat('WARNING : ',sect.filename,' already exists. If you continue, it will be destroyed');
                   wrcol(24,1,mess_color,mess);
                   wrcol(25,1,mess_color,'Press return to continue, any other key to abort...');
                   getc(ans);
                end
                else key_pressed := CARRIAGE;
             if key_pressed=CARRIAGE
                then get_props := true
                else get_props := false;
             blank_line(25);
             blank_line(24);
             close(sect_file);
          end;

(*-------------------------------------------------------------------------------*)

overlay procedure help(screen : integer);
          begin[
             case screen of
                0  : begin (* main menu *)
                     end;
                1  : begin (* prop menu *)
                     end;
                2  : begin (* load menu *)
                     end;
                3  : begin (* sect menu *)
                     end;
                4  : begin (* unassigned *)
                     end;
                5  : begin (* load_prop_file *)
                     end;
                6  : begin (* save_prop_file *)
                     end;
                7  : begin (* load_load_file *)
                     end;
                8  : begin (* save_load_file *)
                     end;
                9  : begin (* load_sect_file *)
                     end;
                10 : begin (* save_sect_file *)
                     end;
                11 : begin (* load_defl_file *)
                     end;
                12 : begin (* save_defl_file *)
                     end;
             end (* case *);}
          end;

(*--------------------------------------------------------------------------*)

procedure dummy3;
          begin
          end;

(*--------------------------------------------------------------------------*)

overlay procedure prop_init;
          begin
               with prop do begin
                  filename := '                    ';
                  length := 0;
                  width := 0;
                  depth := 0;
                  conc_str := 0;
                  steel_yield := 0;
                  steel_ult := 0;
                  layers := 3;
                  steel[1].area := 0;
```

```
   steel[1].depth := 0;
   steel[2].area := 0;
   steel[2].depth := 0;
   steel[3].area := 0;
   steel[3].depth := 0;
   tie_spacing := 0;
   tie_bar_area := 0;
   tie1_no_legs := 0;
   tie2_no_legs := 0;
   ult_conc_strain := 0;
   inter_flag := false;
   strain_at_ult := 0.1;
   strain_hard := 0.01;
   no_slices := 50;
end (* with prop*);
prop_row[1] := 4;
prop_col[1] := 12;
prop_row[2] := 6;
prop_col[2] := 11;
prop_row[3] := 8;
prop_col[3] := 11;
prop_row[4] := 11;
prop_col[4] := 17;
prop_row[5] := 12;
prop_col[5] := 19;
prop_row[6] := 15;
prop_col[6] := 21;
prop_row[7] := 16;
prop_col[7] := 19;
prop_row[8] := 18;
prop_col[8] := 18;
prop_row[9] := 19;
prop_col[9] := 17;
prop_row[10] := 4;
prop_col[10] := 53;
prop_row[11] := 7;
prop_col[11] := 45;
prop_row[12] := 8;
prop_col[12] := 46;
prop_row[13] := 11;
prop_col[13] := 45;
prop_row[14] := 12;
prop_col[14] := 46;
```

```
            prop_row[15] := 15;
            prop_col[15] := 45;
            prop_row[16] := 16;
            prop_col[16] := 46;
            prop_row[17] := 18;
            prop_col[17] := 55;
            prop_row[18] := 19;
            prop_col[18] := 56;
         end (* prop_init *);

(*-----------------------------------------------------------------------------*)

overlay procedure load_init;
          begin
             load_type_descr[1] := 'constant';
             load_type_descr[2] := 'linear';
             load_type_descr[3] := 'multi-linear';
             with load do begin
                filename := '                    ';
                lateral_type := 1;
                axial_type := 1;
                load_control := true;
                for i := 1 to 10 do begin
                   lateral[i] := 0;
                   axial[i] := 0;
                end (* for *);
             end (* with *);
             lat_boxes := 2;
             ax_boxes := 2;
             load_row[1] := 6;
             load_col[1] := 18;
             load_row[2] := 8;
             load_col[2] := 10;
             load_row[3] := 6;
             load_col[3] := 48;
             load_row[4] := 8;
             load_col[4] := 40;
             for i := 5 to 22 do begin
                load_row[i] := 0;
                load_col[i] := 0;
             end;
          end (* load_init *);
```

```
(*--------------------------------------------------------------------------*)

overlay procedure switch_load_type;
          var indx : integer;
          begin
             indx := 10*load.lateral_type + load.axial_type;
             case indx of
                11 : begin
                        lat_boxes := 2;
                        ax_boxes := 2;
                        load_row[2] := 8;
                        load_col[2] := 10;
                        load_row[3] := 6;
                        load_col[3] := 48;
                        load_row[4] := 8;
                        load_col[4] := 40;
                     end;
                12 : begin
                        lat_boxes := 2;
                        ax_boxes := 3;
                        load_row[2] := 8;
                        load_col[2] := 10;
                        load_row[3] := 6;
                        load_col[3] := 48;
                        load_row[4] := 8;
                        load_col[4] := 48;
                        load_row[5] := 10;
                        load_col[5] := 46;
                     end;
                13 : begin
                        lat_boxes := 2;
                        ax_boxes := 11;
                        load_row[2] := 8;
                        load_col[2] := 10;
                        load_row[3] := 6;
                        load_col[3] := 48;
                        load_row[4] := 11;
                        load_col[4] := 35;
                        load_row[5] := 13;
                        load_col[5] := 35;
                        load_row[6] := 15;
                        load_col[6] := 35;
                        load_row[7] := 17;
```

```
            load_col[7] := 35;
            load_row[8] := 19;
            load_col[8] := 35;
            load_row[9] := 11;
            load_col[9] := 50;
            load_row[10] := 13;
            load_col[10] := 50;
            load_row[11] := 15;
            load_col[11] := 50;
            load_row[12] := 17;
            load_col[12] := 50;
            load_row[13] := 19;
            load_col[13] := 50;
         end;
    21 : begin
            lat_boxes := 3;
            ax_boxes := 2;
            load_row[2] := 8;
            load_col[2] := 18;
            load_row[3] := 10;
            load_col[3] := 16;
            load_row[4] := 6;
            load_col[4] := 48;
            load_row[5] := 8;
            load_col[5] := 40;
         end;
    22 : begin
            lat_boxes := 3;
            ax_boxes := 3;
            load_row[2] := 8;
            load_col[2] := 18;
            load_row[3] := 10;
            load_col[3] := 16;
            load_row[4] := 6;
            load_col[4] := 48;
            load_row[5] := 8;
            load_col[5] := 48;
            load_row[6] := 10;
            load_col[6] := 46;
         end;
    23 : begin
            lat_boxes := 3;
            ax_boxes := 11;
```

```
          load_row[2] := 8;
          load_col[2] := 18;
          load_row[3] := 10;
          load_col[3] := 16;
          load_row[4] := 6;
          load_col[4] := 48;
          load_row[5] := 11;
          load_col[5] := 35;
          load_row[6] := 13;
          load_col[6] := 35;
          load_row[7] := 15;
          load_col[7] := 35;
          load_row[8] := 17;
          load_col[8] := 35;
          load_row[9] := 19;
          load_col[9] := 35;
          load_row[10] := 11;
          load_col[10] := 50;
          load_row[11] := 13;
          load_col[11] := 50;
          load_row[12] := 15;
          load_col[12] := 50;
          load_row[13] := 17;
          load_col[13] := 50;
          load_row[14] := 19;
          load_col[14] := 50;
       end;
  31 : begin
          lat_boxes := 11;
          ax_boxes := 2;
          load_row[2] := 11;
          load_col[2] := 3;
          load_row[3] := 13;
          load_col[3] := 3;
          load_row[4] := 15;
          load_col[4] := 3;
          load_row[5] := 17;
          load_col[5] := 3;
          load_row[6] := 19;
          load_col[6] := 3;
          load_row[7] := 11;
          load_col[7] := 18;
          load_row[8] := 13;
```

```
            load_col[8] := 18;
            load_row[9] := 15;
            load_col[9] := 18;
            load_row[10] := 17;
            load_col[10] := 18;
            load_row[11] := 19;
            load_col[11] := 18;
            load_row[12] := 6;
            load_col[12] := 48;
            load_row[13] := 8;
            load_col[13] := 40;
        end;
   32 : begin
            lat_boxes := 11;
            ax_boxes := 3;
            load_row[2] := 11;
            load_col[2] := 3;
            load_row[3] := 13;
            load_col[3] := 3;
            load_row[4] := 15;
            load_col[4] := 3;
            load_row[5] := 17;
            load_col[5] := 3;
            load_row[6] := 19;
            load_col[6] := 3;
            load_row[7] := 11;
            load_col[7] := 18;
            load_row[8] := 13;
            load_col[8] := 18;
            load_row[9] := 15;
            load_col[9] := 18;
            load_row[10] := 17;
            load_col[10] := 18;
            load_row[11] := 19;
            load_col[11] := 18;
            load_row[12] := 6;
            load_col[12] := 48;
            load_row[13] := 8;
            load_col[13] := 48;
            load_row[14] := 10;
            load_col[14] := 46;
        end;
   33 : begin
```

```
lat_boxes := 11;
ax_boxes := 11;
load_row[2] := 11;
load_col[2] := 3;
load_row[3] := 13;
load_col[3] := 3;
load_row[4] := 15;
load_col[4] := 3;
load_row[5] := 17;
load_col[5] := 3;
load_row[6] := 19;
load_col[6] := 3;
load_row[7] := 11;
load_col[7] := 18;
load_row[8] := 13;
load_col[8] := 18;
load_row[9] := 15;
load_col[9] := 18;
load_row[10] := 17;
load_col[10] := 18;
load_row[11] := 19;
load_col[11] := 18;
load_row[12] := 6;
load_col[12] := 48;
load_row[13] := 11;
load_col[13] := 35;
load_row[14] := 13;
load_col[14] := 35;
load_row[15] := 15;
load_col[15] := 35;
load_row[16] := 17;
load_col[16] := 35;
load_row[17] := 19;
load_col[17] := 35;
load_row[18] := 11;
load_col[18] := 50;
load_row[19] := 13;
load_col[19] := 50;
load_row[20] := 15;
load_col[20] := 50;
load_row[21] := 17;
load_col[21] := 50;
load_row[22] := 19;
```

```
                        load_col[22] := 50;
                     end (* 33 *);
             end (* case *);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure sect_init;
          var i,j : integer;
          begin
               for j := 1 to 20 do begin
                  for i := 1 to 20 do begin
                      sect.data[j].moment[i] := 0;
                      sect.data[j].curvature[i] := 0;
                  end (* for i *);
                  sect.data[j].no_points := 20;
                  sect.data[j].axial := 0;
               end (* for j *);
               sect.no_ax_loads := 20;
               sect.ductility := 0;
               sect.filename := '                    ';
               sect_row[1] := 4;
               sect_col[1] := 32;
               sect_row[2] := 5;
               sect_col[2] := 34;
               sect_row[3] := 5;
               sect_col[3] := 41;
               sect_row[4] := 6;
               sect_col[4] := 41;
               sect_row[5] := 10;
               sect_col[5] := 5;
               sect_row[6] := 11;
               sect_col[6] := 5;
               sect_row[7] := 12;
               sect_col[7] := 5;
               sect_row[8] := 13;
               sect_col[8] := 5;
               sect_row[9] := 14;
               sect_col[9] := 5;
               sect_row[10] := 15;
               sect_col[10] := 5;
               sect_row[11] := 16;
               sect_col[11] := 5;
```

```
sect_row[12] := 17;
sect_col[12] := 5;
sect_row[13] := 18;
sect_col[13] := 5;
sect_row[14] := 19;
sect_col[14] := 5;
sect_row[15] := 10;
sect_col[15] := 20;
sect_row[16] := 11;
sect_col[16] := 20;
sect_row[17] := 12;
sect_col[17] := 20;
sect_row[18] := 13;
sect_col[18] := 20;
sect_row[19] := 14;
sect_col[19] := 20;
sect_row[20] := 15;
sect_col[20] := 20;
sect_row[21] := 16;
sect_col[21] := 20;
sect_row[22] := 17;
sect_col[22] := 20;
sect_row[23] := 18;
sect_col[23] := 20;
sect_row[24] := 19;
sect_col[24] := 20;
sect_row[25] := 10;
sect_col[25] := 35;
sect_row[26] := 11;
sect_col[26] := 35;
sect_row[27] := 12;
sect_col[27] := 35;
sect_row[28] := 13;
sect_col[28] := 35;
sect_row[29] := 14;
sect_col[29] := 35;
sect_row[30] := 15;
sect_col[30] := 35;
sect_row[31] := 16;
sect_col[31] := 35;
sect_row[32] := 17;
sect_col[32] := 35;
sect_row[33] := 18;
```

```
          sect_col[33] := 35;
          sect_row[34] := 19;
          sect_col[34] := 35;
          sect_row[35] := 10;
          sect_col[35] := 50;
          sect_row[36] := 11;
          sect_col[36] := 50;
          sect_row[37] := 12;
          sect_col[37] := 50;
          sect_row[38] := 13;
          sect_col[38] := 50;
          sect_row[39] := 14;
          sect_col[39] := 50;
          sect_row[40] := 15;
          sect_col[40] := 50;
          sect_row[41] := 16;
          sect_col[41] := 50;
          sect_row[42] := 17;
          sect_col[42] := 50;
          sect_row[43] := 18;
          sect_col[43] := 50;
          sect_row[44] := 19;
          sect_col[44] := 50;
     end (* sect_init *);

(*---------------------------------------------------------------------------*)

overlay procedure init_graphics;
var
  indx     : integer;
  switches : byte;
  info     : byte;
  mono     : boolean;
  ch       : char;
begin
  switches := mem[$40:$88] and $0F;
  info     := mem[$40:$87];
  mono     := odd((info and $02) shr 1);
  if mono then rec.ax := $000F {640 x 350 monochrome}
  else with rec do
    case switches of
      6 : ax := $0D;     {color 40x25 - 320x200}
      7 : ax := $0E;     {color 80x25 - 640x200}
      8 : ax := $10;     {enhanced color normal mode}
      9 : ax := $10;     {enhanced color enhanced mode}
      else rec.ax := $0E;
    end; {case}

  intr($10,rec);
  {Arrays used to avoid repetitive address calculations}
  for indx := 0 to 349 do yaddr[indx] := 80*indx;
  for indx := 0 to 639 do
    begin
      xaddr[indx] := indx div 8;
      point[indx] := $80 shr (indx mod 8)
    end;
  pen_color := green
end;

(*---------------------------------------------------------------------------*)

overlay procedure welcome_screen;
       var seed : integer;
       begin
wrcol(2,1,title_color,'                         W E L C O M E   T O   O P U S                    ');
wrcol(5,1,mess_color,'        This is an interactive program to analyze and abuse various        ');
wrcol(7,1,mess_color,'        reinforced concrete columns.  Its name, in addition to being       ');
wrcol(9,1,mess_color,'          a world-famous cartoon character, stands for the guiding         ');
wrcol(11,1,mess_color,'       principle of its design: Organized to Please its User''s Senses.      ');
wrcol(13,1,mess_color,'       If you ever need more information than a given screen provides,     ');
wrcol(15,1,mess_color,'        just hit the F1 function key on the left of the keyboard.  If      ');
wrcol(17,1,mess_color,'      this doesn''t give enough information, well, too bad: the program     ');
wrcol(19,1,mess_color,'            was free anyway so you don''t have any reason to gripe.         ');
wrcol(21,1,index_color,'                    Written in 1986 by Leo Linbeck, III                    ');
wrcol(22,1,index_color,'                    at the University of Texas at Austin                   ');
wrcol(25,1,white,'PRESS ANY KEY TO CONTINUE....');
          getc(ans);
          seed := round(random(9)+0.5);
          case seed of
             0 : wrcol(25,30,subtitle_color,'Someday all this could be yours....');
             1 : wrcol(25,30,subtitle_color,'Fly low and avoid the radar....');
             2 : wrcol(25,30,subtitle_color,'Beam me up, Scotty....');
             3 : wrcol(25,30,subtitle_color,'and Jerry Mathers as The Beaver....');
             4 : begin
                    wrcol(25,30,subtitle_color,'Today is your lucky day....');
                    delay(1000);
                    wrcol(25,57,error_color,'or is it?');
                 end;
             5 : wrcol(25,30,subtitle_color,'Ground control to Major Tom....');
             6 : wrcol(25,30,subtitle_color,'Opus says : RELAX....');
             7 : wrcol(25,30,subtitle_color,'Dammit Jim I''m a doctor, not a veterinarian....');
             8 : wrcol(25,30,subtitle_color,'I love the smell of napalm in the morning....');
             9 : wrcol(25,30,subtitle_color,'Patience, grasshopper....');
          end (* case *);
          if seed=9
             then delay(7000)
             else delay(1500);
          cls(1,1,25,80,0);
       end;

(*---------------------------------------------------------------------------*)

procedure initialize;
          begin
               randomize;
               mess := '                                                                                ';
               x_abs_window := 480;
               y_abs_window := 320;
               x_abs_offset := 4;
               y_abs_offset := -7;
               curr_page := 0;
               color[5] := white;
               color[4] := green;
               color[3] := lt_blue;
               color[2] := lt_magenta;
               color[1] := lt_red;
               title_color := yellow;
               data_color := lt_magenta;
               menu_color := green;
               descr_color := white;
               subtitle_color := off_white;
               mess_color := brown;
               error_color := lt_red;
               help_color := cyan;
               index_color := lt_cyan;
               init_graphics;
               welcome_screen;
               curr_box := 1;
               curr_ax_load := 1;
               old_ar_col := 79;
               old_ar_row := 24;
               prop_init;
               load_init;
               sect_init;
               scale_factor := 1.1;
               key_pressed := 0;
          end;

(*---------------------------------------------------------------------------*)

overlay procedure get_elem_mom(ax : real; var elem_curv,elem_mom : real);
          var
             i,j : integer;
             finished : boolean;
             mom1,mom2 : real;
             low_boundary,high_boundary : boolean;
             i_incr : integer;
    begin
       j := 0;
       high_boundary := false;
       low_boundary := false;
       finished := false;
       if (elem_curv = 0) or (ax < sect.data[1].axial) or (ax >= sect.data[sect.no_ax_loads].axial)
          then elem_mom := 0
          else begin
             repeat
                j := j + 1;
                i_incr := round(abs(elem_curv)/elem_curv);
                if (sect.no_ax_loads=1) or
                  ((ax>=sect.data[j].axial) and (ax<sect.data[j+1].axial))
                   then with sect.data[j] do if elem_curv < curvature[1]
                      then begin
                         finished := true;
                         low_boundary := true;
                         boundary := true;
                      end
                      else if elem_curv > curvature[no_points]
                         then begin
                            finished := true;
                            high_boundary := true;
                            boundary := true;
                         end
```

```
            else begin
               if i_incr > 0
                  then i := (no_points div 2) - 1
                  else i := (no_points div 2) + 2;
               repeat
                  i := i + i_incr;
                  if (elem_curv>=curvature[i]) and
                    (elem_curv<curvature[i+1])
                     then finished := true;
               until finished;
            end
      else if j>=sect.no_ax_loads
         then begin
            finished := true;
            overflow := true;
         end;
until finished;
if high_boundary
   then begin
      elem_mom :=  (ax - sect.data[j].axial)
                 * (sect.data[j+1].moment[40] - sect.data[j].moment[40])
                 / (sect.data[j+1].axial - sect.data[j].axial)
                 + sect.data[j].moment[40];
      elem_curv :=  (ax - sect.data[j].axial)
                  * (sect.data[j+1].curvature[40] - sect.data[j].curvature[40])
                  / (sect.data[j+1].axial - sect.data[j].axial)
                  + sect.data[j].curvature[40];
   end
   else if low_boundary
      then begin
         elem_mom :=  (ax - sect.data[j].axial)
                    * (sect.data[j+1].moment[1] - sect.data[j].moment[1])
                    / (sect.data[j+1].axial - sect.data[j].axial)
                    + sect.data[j].moment[1];
         elem_curv :=  (ax - sect.data[j].axial)
                     * (sect.data[j+1].curvature[1] - sect.data[j].curvature[1])
                     / (sect.data[j+1].axial - sect.data[j].axial)
                     + sect.data[j].curvature[1];
      end
      else begin
         mom1 :=  (elem_curv - sect.data[j].curvature[i])
                * (sect.data[j].moment[i+1] - sect.data[j].moment[i])
                / (sect.data[j].curvature[i+1] - sect.data[j].curvature[i])
```

```
                             +  sect.data[j].moment[i];
                      if (sect.no_ax_loads=1) or (j=sect.no_ax_loads-1)
                         then elem_mom := mom1
                         else begin
                            mom2 :=  (elem_curv - sect.data[j+1].curvature[i])
                                   * (sect.data[j+1].moment[i+1] - sect.data[j+1].moment[i])
                                   / (sect.data[j+1].curvature[i+1] - sect.data[j+1].curvature[i])
                                   +  sect.data[j+1].moment[i];
                            elem_mom :=  (ax - sect.data[j].axial)
                                       * (mom2 - mom1)
                                       / (sect.data[j+1].axial - sect.data[j].axial)
                                       + mom1;
                         end (* else *);
                end (* if boundary *);
          end (* else *);
    end;

(*--------------------------------------------------------------------------*)

overlay procedure get_elem_curv(ax : real; var elem_mom,elem_curv : real);
          var
             i,j : integer;
             finished : boolean;
             curv1,curv2 : real;
             low_boundary,high_boundary : boolean;
             i_incr : integer;
    begin
       j := 0;
       high_boundary := false;
       low_boundary := false;
       finished := false;
       if (elem_mom = 0) or (ax < sect.data[1].axial) or (ax >= sect.data[sect.no_ax_loads].axial)
          then elem_curv := 0
          else begin
             repeat
                j := j + 1;
                i_incr := round(abs(elem_mom)/elem_mom);
                if (sect.no_ax_loads=1) or
                  ((ax>=sect.data[j].axial) and (ax<sect.data[j+1].axial))
                   then with sect.data[j] do if elem_mom <= moment[1]
                      then begin
                         finished := true;
                         low_boundary := true;
```

```
         end
         else if elem_mom >= moment[no_points]
            then begin
               finished := true;
               high_boundary := true;
            end
            else begin
               if i_incr > 0
                  then i := (no_points div 2) - 1
                  else i := (no_points div 2) + 2;
               repeat
                  i := i + i_incr;
                  if (elem_mom>=moment[i]) and
                    (elem_mom<moment[i+1])
                     then finished := true;
               until finished
            end
      else if j>=sect.no_ax_loads
         then begin
            finished := true;
            overflow := true;
         end;
until finished;
if high_boundary
   then begin
      elem_mom :=  (ax - sect.data[j].axial)
                 * (sect.data[j+1].moment[40] - sect.data[j].moment[40])
                 / (sect.data[j+1].axial - sect.data[j].axial)
                 + sect.data[j].moment[40];
      elem_curv :=  (ax - sect.data[j].axial)
                  * (sect.data[j+1].curvature[40] - sect.data[j].curvature[40])
                  / (sect.data[j+1].axial - sect.data[j].axial)
                  + sect.data[j].curvature[40];
   end
   else if low_boundary
      then begin
         elem_mom :=  (ax - sect.data[j].axial)
                    * (sect.data[j+1].moment[1] - sect.data[j].moment[1])
                    / (sect.data[j+1].axial - sect.data[j].axial)
                    + sect.data[j].moment[1];
         elem_curv :=  (ax - sect.data[j].axial)
                     * (sect.data[j+1].curvature[1] - sect.data[j].curvature[1])
                     / (sect.data[j+1].axial - sect.data[j].axial)
```

```
                                  + sect.data[j].curvature[1];
                   end
                   else begin
                      curv1 :=  (elem_mom - sect.data[j].moment[i])
                              * (sect.data[j].curvature[i+1] - sect.data[j].curvature[i])
                              / (sect.data[j].moment[i+1] - sect.data[j].moment[i])
                              +  sect.data[j].curvature[i];
                      if (sect.no_ax_loads=1) or (j=sect.no_ax_loads-1)
                         then elem_curv := curv1
                         else begin
                            curv2 :=  (elem_mom - sect.data[j+1].moment[i])
                                    * (sect.data[j+1].curvature[i+1] - sect.data[j+1].curvature[i])
                                    / (sect.data[j+1].moment[i+1] - sect.data[j+1].moment[i])
                                    +  sect.data[j+1].curvature[i];
                            elem_curv :=  (ax - sect.data[j].axial)
                                        * (curv2 - curv1)
                                        / (sect.data[j+1].axial - sect.data[j].axial)
                                        + curv1;
                         end (* else *);
                end (* if boundary *);
          end (* else *);
    end;

(*----------------------------------------------------------------------------*)

overlay procedure smoother;
          var i,j : integer;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'...smoothing data, please wait...');
             for i := 2 to sect.no_ax_loads-1 do with sect.data[i] do begin
                for j := 1 to (no_points div 2) do
                   if moment[j] >= 0
                      then begin
                         moment[j] := 0;
                         moment[no_points+1-j] := 0;
                      end;
                for j := 2 to no_points-1 do
                   if (abs(moment[j]) < 0.000001) and (moment[j+1] <> 0) and (moment[j-1] <> 0)
                      then moment[j] := (moment[j+1] + moment[j-1]) / 2;
             end;
             blank_line(25);
          end;
```

```
(*---------------------------------------------------------------------------*)

overlay procedure get_load_steps;
          var
             i,j,count : integer;
             lat_inc,ax_inc : real;
          begin
               if (load.lateral_type=3) or (load.axial_type=3) then defl.no_load_steps := 89;
               no_load_steps := defl.no_load_steps;
               if load.load_control
                  then case load.lateral_type of
                           1 : for i := 0 to no_load_steps do defl.lat_load[i] := load.lateral[1];
                           2 : begin
                                  lat_inc := (load.lateral[2]-load.lateral[1]) / no_load_steps;
                                  for i := 0 to no_load_steps do defl.lat_load[i] := load.lateral[1] + i*lat_inc;
                               end;
                           3 : for i := 1 to 9 do begin
                                  lat_inc := (load.lateral[i+1]-load.lateral[i]) / 10.0;
                                  for j := 0 to 9 do
                                     begin
                                        count := 10 * (i-1) + j;
                                        defl.lat_load[count] := load.lateral[i] + lat_inc;
                                     end (* for j *);
                              end (* for i *);
                  end (* then case *)
                  else case load.lateral_type of
                           1 : for i := 0 to no_load_steps do defl.deflection[i] := load.lateral[1];
                           2 : begin
                                  lat_inc := (load.lateral[2]-load.lateral[1]) / no_load_steps;
                                  for i := 0 to no_load_steps do defl.deflection[i] := load.lateral[1] + i*lat_inc;
                               end;
                           3 : for i := 1 to 9 do begin
                                  lat_inc := (load.lateral[i+1]-load.lateral[i]) / 10.0;
                                  for j := 0 to 9 do
                                     begin
                                        count := 10 * (i-1) + j;
                                        defl.deflection[count] := load.lateral[i] + lat_inc;
                                     end (* for j *);
                              end (* for i *);
                  end (* else case *);
               case load.axial_type of
                     1 : for i := 0 to no_load_steps do defl.ax_load[i] := load.axial[1];
```

```
                    2 : begin
                             ax_inc := (load.axial[2]-load.axial[1]) / no_load_steps;
                             for i := 0 to no_load_steps do defl.ax_load[i] := load.axial[1] + i*ax_inc;
                        end;
                    3 : for i := 1 to 9 do
                            begin
                                 ax_inc := (load.axial[i+1]-load.axial[i]) / 10.0;
                                 for j := 0 to 9 do
                                     begin
                                          count := 10 * (i-1) + j;
                                          defl.ax_load[count] := load.axial[i] + ax_inc;
                                     end (* for j *);
                            end (* for i *);
                  end (* case *);
          end;

(*---------------------------------------------------------------------------*)

function find_elem_curv(ax,last_mom,last_curv,target_mom,curv_inc : real) : real;
         var
            mom,curv : real;
            old_mom : real;
            okee_dokee : boolean;
            pass : integer;
         begin
            okee_dokee := false;
            curv := last_curv;
            mom := last_mom;
            pass := 0;
            repeat
               pass := pass + 1;
               old_mom := mom;
               get_elem_mom(ax,curv,mom);
               if curv_inc > 0
                  then if (old_mom<=target_mom) and (mom>=target_mom)
                     then okee_dokee := true
                     else curv := curv + curv_inc
                  else if (old_mom>=target_mom) and (mom<=target_mom)
                     then okee_dokee := true
                     else curv := curv + curv_inc;
            until okee_dokee or (pass > 100);
            if pass > 100
               then find_elem_curv := 0
               else find_elem_curv :=  curv - curv_inc
                                     + curv_inc * (target_mom-old_mom)
                                     / (mom-old_mom);
          end;

(*-----------------------------------------------------------------------------*)

procedure get_curvature(indx : integer);
          var
             i : integer;
             complete : boolean;
             max_curv,min_curv,curv_inc : real;
             ax_load : real;
             no_pts : integer;
          begin
             ax_load := sect.data[indx].axial;
             no_pts := 40;
             i := 1;
             complete := false;
             repeat
                if (ax_load>=inter.axial[i]) and (ax_load<inter.axial[i+1])
                   then complete := true
                   else i := i + 1;
             until complete;
             max_curv :=  (ax_load - inter.axial[i])
                        * (inter.curvature[i+1] - inter.curvature[i])
                        / (inter.axial[i+1] - inter.axial[i])
                        - inter.curvature[i];
             min_curv := -max_curv;
             curv_inc := max_curv * (sect.ductility-1) / 5;
             for i := 1 to 6 do
                sect.data[indx].curvature[i] :=  min_curv * sect.ductility
                                               + (i-1)*curv_inc;
             curv_inc := max_curv / (no_pts div 2 - 5);
             for i := 7 to (no_pts div 2) do
                sect.data[indx].curvature[i] := min_curv + (i-6)*curv_inc;
             for i := 1 to (no_pts div 2) do
                sect.data[indx].curvature[no_pts+1-i] := -sect.data[indx].curvature[i];
          end;

(*-----------------------------------------------------------------------------*)
(*-----------------------------------------------------------------------------*)
(*                                                                             *)
(*  The following routines include the three dimensional graphics procedures *)
(*    used to draw the moment-axial load-curvature diagram.  Big deal, eh?   *)
(*                                                                           *)
(*---------------------------------------------------------------------------*)
(*---------------------------------------------------------------------------*)

overlay procedure show_graph;

(*---------------------------------------------------------------------------*)
(*---------------------------------------------------------------------------*)

function rel_to_abs (rel : real; axis : char) : integer;
         var xw,yw : real;
         begin
              xw := x_abs_window;
              yw := y_abs_window;
              case axis of
                   'x' : rel_to_abs := round((rel-min.x) * xw / x_rel_window) + x_abs_offset;
                   'y' : rel_to_abs := round((max.y-rel) * yw / y_rel_window) + y_abs_offset-1;
              end;
         end;

(*---------------------------------------------------------------------------*)

procedure max_min;
         var i : integer;
         begin
            max.x := picture^.data[1].x;
            min.x := max.x;
            max.y := picture^.data[1].y;
            min.y := max.y;
            max.z := picture^.data[1].z;
            min.z := max.z;
            for i := 2 to picture_size do begin
               if picture^.data[i].x > max.x then max.x := picture^.data[i].x;
               if picture^.data[i].x < min.x then min.x := picture^.data[i].x;
               if picture^.data[i].y > max.y then max.y := picture^.data[i].y;
               if picture^.data[i].y < min.y then min.y := picture^.data[i].y;
               if picture^.data[i].z > max.z then max.z := picture^.data[i].z;
               if picture^.data[i].z < min.z then min.z := picture^.data[i].z;
            end;
         end;
```

```
(*--------------------------------------------------------------------------*)

procedure draw_axis (axis : char);
          var x0,y0,i,x_dash,y_dash,x_d,y_d,st,fn : integer;
          begin
             x_d := 18;
             y_d := 10;
             x0 := rel_to_abs(0,'x');
             y0 := rel_to_abs(0,'y');
             x_dash := round(x_abs_window / x_d) - 1;
             y_dash := round(y_abs_window / y_d) - 1;
             set_color(white);
             case axis of
                'x' : if (min.y < 0) and (max.y > 0) then
                         for i := 0 to x_dash do begin
                            st := x_d * i + 3;
                            fn := st + x_d div 2;
                            drawc(st,y0,fn,y0);
                         end;
                'y' : if (min.x < 0) and (max.x > 0) then
                         for i := 0 to y_dash do begin
                            st := y_d * i + 3;
                            fn := st + y_d div 2;
                            drawc(x0,st,x0,fn);
                         end;
                '3' : for i := 1 to 3 do
                         drawc(x0,y0,rel_to_abs(axes[i].x,'x'),rel_to_abs(axes[i].y,'y'))
             end (* case *);
          end;

(*--------------------------------------------------------------------------*)
(*--------------------------------------------------------------------------*)

overlay procedure show_picture;

(*--------------------------------------------------------------------------*)

   function clip_line(a,b : integer; var a1,b1,a2,b2 : integer) : boolean;

         type outcode = array [1..4] of boolean;

         var
            outcode1,outcode2 : outcode;
            s1,t1,s2,t2,s_max,t_max,s_min,t_min : real;

(*---------------------------------------------------------------------------*)

      function reject_check : boolean;
            begin
               reject_check := false;
               for i := 1 to 4 do if (outcode1[i]) and (outcode2[i]) then reject_check := true;
            end;

(*---------------------------------------------------------------------------*)

      function accept_check : boolean;
            var and_check : boolean;
            begin
               accept_check := true;
               for i := 1 to 4 do if (outcode1[i]) or (outcode2[i]) then accept_check := false;
            end;

(*---------------------------------------------------------------------------*)

      procedure swap(var a1,b1,a2,b2 : integer);
             var
                b_temp : boolean;
                i_temp : integer;
                r_temp : real;
             begin
                for i := 1 to 4 do begin
                   b_temp := outcode1[i];
                   outcode1[i] := outcode2[i];
                   outcode2[i] := b_temp;
                end;
                i_temp := a1;
                a1 := a2;
                a2 := i_temp;
                i_temp := b1;
                b1 := b2;
                b2 := i_temp;
                r_temp := s1;
                s1 := s2;
                s2 := r_temp;
                r_temp := t1;
                t1 := t2;
            t2 := r_temp;
         end;

(*-------------------------------------------------------------------------*)

      var
         accept,reject,done : boolean;
      begin
         a1 := rel_to_abs(picture^.data[a].x,'x');
         a2 := rel_to_abs(picture^.data[b].x,'x');
         b1 := rel_to_abs(picture^.data[a].y,'y');
         b2 := rel_to_abs(picture^.data[b].y,'y');
         s1 := a1;
         t1 := b1;
         s2 := a2;
         t2 := b2;
         s_max := x_abs_window-2;
         t_max := y_abs_window-2;
         s_min := 1;
         t_min := 1;
         accept := false;
         reject := false;
         done := false;
         clip_line := false;
         repeat
            for i := 1 to 4 do outcode1[i] := false;
            if y_abs_window <= b1 then outcode1[1] := true;
            if b1 <= 1 then outcode1[2] := true;
            if x_abs_window <= a1 then outcode1[3] := true;
            if a1 <= 0 then outcode1[4] := true;
            for i := 1 to 4 do outcode2[i] := false;
            if y_abs_window <= b2 then outcode2[1] := true;
            if b2 <= 1 then outcode2[2] := true;
            if x_abs_window <= a2 then outcode2[3] := true;
            if a2 <= 0 then outcode2[4] := true;
            reject := reject_check;
            if reject
               then done := true
               else begin
                  accept := accept_check;
                  if accept
                     then done := true
                     else begin
```

```
                           if not   ((outcode1[1]) or (outcode1[2])
                                  or (outcode1[3]) or (outcode1[4]))
                              then swap(a1,b1,a2,b2);
                           if outcode1[1]
                              then begin
                                 b1 := y_abs_window-1;
                                 if b2 <> b1
                                    then a1 := round(s1 + (s2-s1)*(t_max-t1)/(t2-t1));
                                 s1 := a1;
                              end
                              else if outcode1[2]
                                 then begin
                                    b1 := 2;
                                    if b2 <> b1
                                       then a1 := round(s1 + (s2-s1)*(t_min-t1)/(t2-t1));
                                    s1 := a1;
                                 end
                                 else if outcode1[3]
                                      then begin
                                         a1 := x_abs_window-1;
                                         if a2 <> a1
                                            then b1 := round(t1 + (t2-t1)*(s_max-s1)/(s2-s1));
                                         t1 := b1;
                                      end
                                      else if outcode1[4]
                                         then begin
                                            a1 := 1;
                                            if a2 <> a1
                                               then b1 := round(t1 + (t2-t1)*(s_min-s1)/(s2-s1));
                                            t1 := b1;
                                         end;
                        end;
                  end;
            until done;
            clip_line := accept;
         end;

(*---------------------------------------------------------------------------*)

          var
             i,a,b,leo_mod,leo_div : integer;
             fac : real;
             a1,b1,a2,b2 : integer;
```

```
      begin
         max_min;
         x_rel_window := scale_factor * (max.x - min.x);
         if x_rel_window = 0 then x_rel_window := 0.01;
         y_rel_window := scale_factor * (max.y - min.y);
         if y_rel_window = 0 then y_rel_window := 0.01;
         cls(1,1,23,61,0);
         draw_box(0,0,x_abs_window-1,y_abs_window-1);
         for i := 1 to picture_size-1 do begin
            leo_div := i div y_mesh_fineness;
            leo_mod := i mod y_mesh_fineness;
            if leo_mod <> 0
               then begin
                  a := i;
                  b := i + 1;
                  if pen_color <> picture^.data[a].c
                     then set_color(picture^.data[a].c);
                  if clip_line(a,b,a1,b1,a2,b2)
                     then drawc(a1,b1,a2,b2);
               end;
            if (leo_div < x_mesh_fineness-1)
               then begin
                  a := i;
                  b := i + y_mesh_fineness;
                  if pen_color <> picture^.data[a].c
                     then set_color(picture^.data[a].c);
                  if clip_line(a,b,a1,b1,a2,b2)
                     then drawc(a1,b1,a2,b2);
               end;
        end;
        a := picture_size;
        b := picture_size - y_mesh_fineness;
        if clip_line(a,b,a1,b1,a2,b2)
           then drawc(a1,b1,a2,b2);
        blank_line(25);
        wrcol(25,1,mess_color,mess);
      end;

(*--------------------------------------------------------------------------*)
(*--------------------------------------------------------------------------*)

overlay procedure rotate_picture;
```

```
        var a1,b1 : real;

(*---------------------------------------------------------------------------*)

   overlay procedure define_picture;
          var
             i,j,k,leo,clr,indx : integer;
          begin
             x_mesh_fineness := sect.no_ax_loads;
             y_mesh_fineness := sect.data[1].no_points;
             for i := 1 to x_mesh_fineness do with sect.data[i] do begin
                for k := 1 to 5 do
                   if (axial>inter.axial[10*(k-1)+1])
                    and (axial<=inter.axial[10*(k-1)+10])
                       then clr := color[k];
                for j := 1 to y_mesh_fineness do begin
                   leo := y_mesh_fineness*(i-1) + j;
                   picture^.data[leo].z := -curvature[j];
                   picture^.data[leo].y := moment[j];
                   picture^.data[leo].x := axial;
                   picture^.data[leo].c := clr;
                end;
             end;
             picture^.data[0].x := 1.0;
             picture^.data[0].y := 1.0;
             picture^.data[0].z := 1.0;
             picture^.data[0].c := 0;
             picture_size := x_mesh_fineness * y_mesh_fineness;
             link_no := (x_mesh_fineness-1)*y_mesh_fineness + x_mesh_fineness*(y_mesh_fineness-1);
             for i := 1 to 3 do begin
                axes[i].x := 0;
                axes[i].y := 0;
                axes[i].z := 0;
             end;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure set_up_rotation_matrix;
          var
             cos_a,cos_b,cos_c : real;
             sin_a,sin_b,sin_c : real;
          begin
```

```
             cos_b := cos(alpha) * cos(beta);
             cos_a := cos(beta);
             cos_c := sin(alpha) * cos(beta);
             sin_a := sqrt(1 - sqr(cos_a));
             sin_b := sqrt(1 - sqr(cos_b));
             sin_c := sqrt(1 - sqr(cos_c));
             rotate[1] :=  cos_a * cos_b;
             rotate[2] :=  sin_a * cos_b * cos_c  +         sin_b * sin_c;
             rotate[3] :=  sin_a * cos_b * sin_c  -         sin_b * cos_c;
             rotate[4] :=  0;
             rotate[5] := -sin_a;
             rotate[6] :=  cos_a *        cos_c;
             rotate[7] :=  cos_a *        sin_c;
             rotate[8] :=  0;
             rotate[9] :=  cos_a * sin_b;
             rotate[10] := sin_a * sin_b * cos_c  -         cos_b * sin_c;
             rotate[11] := sin_a * sin_b * sin_c  +         cos_b * cos_c;
             rotate[12] := 0;
             rotate[13] := 1;
          end;
(*--------------------------------------------------------------------------*)

   overlay procedure rotation;
          var
             temp : pixel;
             i : integer;
          begin
             for i := 0 to picture_size do begin
                 temp.x :=  picture^.data[i].x * rotate[1]
                          + picture^.data[i].y * rotate[2]
                          + picture^.data[i].z * rotate[3];
                 temp.y :=  picture^.data[i].x * rotate[5]
                          + picture^.data[i].y * rotate[6]
                          + picture^.data[i].z * rotate[7];
                 temp.z :=  picture^.data[i].x * rotate[9]
                          + picture^.data[i].y * rotate[10]
                          + picture^.data[i].z * rotate[11];
                 temp.c := picture^.data[i].c;
                 picture^.data[i] := temp;
             end;
             for i := 1 to 3 do begin
                 temp.x :=  axes[i].x * rotate[1]
```

```
                          + axes[i].y * rotate[2]
                          + axes[i].z * rotate[3];
                 temp.y :=  axes[i].x * rotate[5]
                          + axes[i].y * rotate[6]
                          + axes[i].z * rotate[7];
                 temp.z :=  axes[i].x * rotate[9]
                          + axes[i].y * rotate[10]
                          + axes[i].z * rotate[11];
                 temp.c := axes[i].c;
                 axes[i] := temp;
             end;
          end;

(*----------------------------------------------------------------------------*)
{
overlay procedure shear;
        var
           i : integer;
        begin
           for i := 0 to picture_size do begin
              picture^.data[i].x :=  picture^.data[i].x
                                   + picture^.data[i].z * a1;
              picture^.data[i].y :=  picture^.data[i].y
                                   + picture^.data[i].z * b1;
           end;
           for i := 1 to 3 do begin
               axes[i].x := axes[i].x + axes[i].z * a1;
               axes[i].y := axes[i].y + axes[i].z * b1;
           end;
        end;
}
(*----------------------------------------------------------------------------*)

   overlay procedure scaling;
          var  i : integer;
          begin
             for i := 0 to picture_size do begin
                 picture^.data[i].x := picture^.data[i].x * scale.x;
                 picture^.data[i].y := picture^.data[i].y * scale.y;
                 picture^.data[i].z := picture^.data[i].z * scale.z;
             end;
             for i := 1 to 3 do begin
                 axes[i].x := axes[i].x * scale.x;
                 axes[i].y := axes[i].y * scale.y;
                 axes[i].z := axes[i].z * scale.z;
             end;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure translation;
          var i : integer;
          begin
             for i := 0 to picture_size do begin
                picture^.data[i].x := picture^.data[i].x + translate.x;
                picture^.data[i].y := picture^.data[i].y + translate.y;
                picture^.data[i].z := picture^.data[i].z + translate.z;
             end;
             for i := 1 to 3 do begin
                axes[i].x := axes[i].x + translate.x;
                axes[i].y := axes[i].y + translate.y;
                axes[i].z := axes[i].z + translate.z;
             end;
          end;

(*---------------------------------------------------------------------------*)

          var
             i : integer;
          begin
             define_picture;
             max_min;
             axes[1].x := 1;
             axes[2].y := 1;
             axes[3].z := 1;
             initial_size.x := max.x - min.x;
             initial_size.y := max.y - min.y;
             initial_size.z := max.z - min.z;
             translate.x := -min.x;
             translate.y := -min.y;
             translate.z := -min.z;
             translation;
             set_up_rotation_matrix;
             blank_line(25);
             wrcol(25,1,mess_color,'Rotating matrix           ');
             rotation;
```

```
              blank_line(25);
              wrcol(25,1,mess_color,'Shearing matrix            ');
{              shear;          }
              blank_line(25);
              wrcol(25,1,mess_color,'Scaling matrix             ');
              max_min;
              scale.x := initial_size.x / (max.x-min.x);
              scale.y := initial_size.y / (max.y-min.y);
              scale.z := initial_size.z / (max.z-min.z);
              scaling;
              translate.x := -translate.x;
              translate.y := -translate.y;
              translate.z := -translate.z;
              blank_line(25);
              wrcol(25,1,mess_color,'Translating matrix         ');
              translation;
              str(round(alpha*180/pi),alpha_st);
              str(round(beta*180/pi),beta_st);
              mess := concat('alpha = ',alpha_st,', ','beta = ',beta_st);
              blank_line(24);
              blank_line(25);
              wrcol(25,1,mess_color,'Drawing picture            ');
           end;

(*---------------------------------------------------------------------------*)
(*---------------------------------------------------------------------------*)

overlay procedure draw_plot;
           var
              i : integer;
              x1,x2,y1,y2 : integer;
           begin
              draw_box(0,0,x_abs_window-1,y_abs_window-1);
              x_rel_window := 1.1*(max.x - min.x);
              y_rel_window := 1.1*(max.y - min.y);
              x_abs_offset := 16;
              y_abs_offset := 14;
              if abs(x_rel_window)<0.000001 then x_rel_window := 1.1 * max.x;
              if abs(y_rel_window)<0.000001 then y_rel_window := 1.1 * max.y;
              draw_axis('x');
              draw_axis('y');
              x1 := rel_to_abs(x_data[0],'x');
              y1 := rel_to_abs(y_data[0],'y');
```

```
             set_color(cyan);
             for i := 1 to no_steps do begin
                x2 := rel_to_abs(x_data[i],'x');
                y2 := rel_to_abs(y_data[i],'y');
                drawc(x1,y1,x2,y2);
                x1 := x2;
                y1 := y2;
             end (* for *);
          end;

(*----------------------------------------------------------------------------*)

overlay procedure defl_to_csv;
          var
             filename : string[20];
             csv : text[$400];
             ext_pos : integer;
             i,j : integer;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'...generating ASCII file, please wait...');
             ext_pos := pos('.dfl',defl.filename)-1;
             filename := copy(defl.filename,1,ext_pos);
             filename := concat(filename,'_d','.csv');
             assign(csv,filename);
             rewrite(csv);
             with defl do
                for j := 0 to no_load_steps do with seg^.data[j] do
                   if (no_elem <= 60) and (no_load_steps <= 30)
                      then for i := 0 to no_elem do
                         writeln(csv,ax_load[j]:12,',',
                                     lat_load[j]:12,',',
                                     moment[i]:12,',',
                                     curvature[i]:12,',',
                                     rotation[i]:12,',',
                                     deflection[i]:12)
                      else writeln(csv,f_to_a(ax_load[j],8,3),',',
                                       f_to_a(lat_load[j],8,3),',',
                                       f_to_a(deflection[j],3,9));

             close(csv);
             blank_line(25);
          end;
```

```
(*---------------------------------------------------------------------------*)
(*---------------------------------------------------------------------------*)

procedure run_move;

(*---------------------------------------------------------------------------*)

   overlay procedure up_alpha;
          begin
             alpha :=  alpha + pi / alpha_increment;
             rotate_picture;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure down_alpha;
          begin
             alpha := alpha - pi / alpha_increment;
             rotate_picture;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure up_beta;
          begin
             beta := beta + pi / beta_increment;
             rotate_picture;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure down_beta;
          begin
             beta:= beta - pi / beta_increment;
             rotate_picture;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure up_x;
          begin
             translate.x := -x_increment;
             x_abs_offset := x_abs_offset - round(translate.x);
             show_picture;
          end;

(*--------------------------------------------------------------------------*)

   overlay procedure down_x;
          begin
             translate.x := x_increment;
             x_abs_offset := x_abs_offset + round(translate.x);
             show_picture;
          end;

(*--------------------------------------------------------------------------*)

   overlay procedure up_y;
          begin
             translate.y := y_increment;
             y_abs_offset := y_abs_offset + round(translate.y);
             show_picture;
          end;

(*--------------------------------------------------------------------------*)

   overlay procedure down_y;
          begin
             translate.y := -y_increment;
             y_abs_offset := y_abs_offset - round(translate.y);
             show_picture;
          end;

(*--------------------------------------------------------------------------*)

   overlay procedure toggle2_menu;
          begin
               if x_abs_window = 640 then x_abs_window := 480
                  else x_abs_window := 640;
               if y_abs_window = 350 then y_abs_window := 320
                  else y_abs_window := 350;
               cls(1,1,25,80,0);
               show_picture;
               key_pressed := 0;
          end;
```

```
(*---------------------------------------------------------------------------*)

   overlay procedure change_scale;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'Input scale factor > ');
             getnum(scale_factor,4,4);
             show_picture;
          end;

(*---------------------------------------------------------------------------*)

   overlay procedure move_menu;
          begin
             if x_abs_window = 480
                then begin
                   cls(1,62,23,80,0);
                   wrcol(2,62,menu_color,'F1  : + alpha');
                   wrcol(3,62,menu_color,'F2  : - alpha');
                   wrcol(4,62,menu_color,'F3  : + beta');
                   wrcol(5,62,menu_color,'F4  : - beta');
                   wrcol(6,62,menu_color,'F5  : + x');
                   wrcol(7,62,menu_color,'F6  : - x');
                   wrcol(8,62,menu_color,'F7  : + y');
                   wrcol(9,62,menu_color,'F8  : - y');
                   wrcol(10,62,menu_color,'F9  : close in or');
                   wrcol(11,62,menu_color,'      back away');
                   wrcol(12,62,menu_color,'F10 : toggle');
                   wrcol(13,62,menu_color,'      menu');
                   wrcol(14,62,menu_color,'ESC : return to');
                   wrcol(15,62,menu_color,'      sect menu');
                end;
          end;

(*---------------------------------------------------------------------------*)

          begin
             move_menu;
             repeat
                getc(ans);
                case key_pressed of
                   F1  : up_alpha;
                   F2  : down_alpha;
                   F3  : up_beta;
                   F4  : down_beta;
                   F5  : up_x;
                   F6  : down_x;
                   F7  : up_y;
                   F8  : down_y;
                   F9  : change_scale;
                   F10 : toggle2_menu;
                   else begin end;
                end (* case *);
             until key_pressed = ESC;
             key_pressed := 0;
          end;

(*-----------------------------------------------------------------------------*)
(*-----------------------------------------------------------------------------*)
(*-----------------------------------------------------------------------------*)

          var
             i,indx : integer;
             x_row,y_col,x_lim,y_lim : integer;
             x_save,y_save : integer;
             temp : real;
          begin
             draw_box(0,0,x_abs_window-1,y_abs_window-1);
             case graph_no of
                1 : begin (* lateral load vs. lateral deflection *)
                       max.y := defl.lat_load[0];
                       min.y := max.y;
                       max.x := defl.deflection[0];
                       min.x := max.x;
                       i := 0;
                       for indx := 0 to defl.no_load_steps do
                          if defl.deflection[indx] <> 0
                             then begin
                                y_data[i] := defl.lat_load[indx];
                               x_data[i] := defl.deflection[indx];
                                if y_data[i]>max.y then max.y := y_data[i];
                                if y_data[i]<min.y then min.y := y_data[i];
                                if x_data[i]>max.x then max.x := x_data[i];
                                if x_data[i]<min.x then min.x := x_data[i];
                                i := i + 1;
             end (* if / for *);
       no_steps := i - 1;
       draw_plot;
       x_row := round(rel_to_abs(0,'y')/14);
       y_col := round(rel_to_abs(0,'x')/8);
       x_lim := x_abs_window div 8;
       y_lim := y_abs_window div 14;
       wrcol(x_row,2,white,f_to_a(min.x,2,6));
       wrcol(2,y_col-6,white,f_to_a(max.y,6,2));
       wrcol(x_row+2,x_lim-9,white,f_to_a(max.x,2,6));
       wrcol(y_lim,y_col+1,white,f_to_a(min.y,6,2));
       if key_pressed <> F7
          then begin
             blank_line(24);
             wrcol(24,1,mess_color,'Do you want an ASCII file of the load-deflection curve (y/n)?');
             getc(ans);
             blank_line(24);
             if upcase(chr(ans)) = 'Y'
                then defl_to_csv;
          end;
    end (* case 1 *);
2 : begin (* lateral and axial load vs. time *)
       min.x := 0;
       max.y := load.lateral[1];
       min.y := max.y;
       if load.lateral_type=3 then no_steps:=9 else no_steps:=1;
       max.x := no_steps;
       for i := 0 to no_steps do begin
          y_data[i] := load.lateral[i+1];
          x_data[i] := i;
          if y_data[i]>max.y then max.y := y_data[i];
          if y_data[i]<min.y then min.y := y_data[i];
       end (* for *);
       y_abs_window := 90;
       y_abs_offset := 0;
       draw_plot;
       max.y := load.axial[1];
       min.y := max.y;
       if load.axial_type=3 then no_steps:=9 else no_steps:=1;
       max.x := no_steps;
       for i := 0 to no_steps do begin
          y_data[i] := load.axial[i+1];
          x_data[i] := i;
```

```
            if y_data[i]>max.y then max.y := y_data[i];
            if y_data[i]<min.y then min.y := y_data[i];
         end (* for *);
         y_abs_offset := 90;
         draw_plot;
         y_abs_window := 320;
         y_abs_offset := 0;
      end (* case 2 *);
   3 : begin (* moment interaction diagram *)
         max.y := inter.axial[50];
         min.y := inter.axial[1];
         max.x := calc.bal_mom;
         min.x := -max.x;
         no_steps := 99;
         for i := 0 to 49 do begin
            x_data[i] := inter.moment[i+1];
            x_data[99-i] := -inter.moment[i+1];
            y_data[i] := inter.axial[i+1];
            y_data[99-i] := inter.axial[i+1];
         end;
         draw_plot;
         x_row := round(rel_to_abs(0,'y')/14);
         y_col := round(rel_to_abs(0,'x')/8);
         x_lim := x_abs_window div 8;
         y_lim := y_abs_window div 14;
         wrcol(x_row,2,white,f_to_a(-calc.bal_mom,6,2));
         wrcol(2,y_col-6,white,f_to_a(inter.axial[50],6,2));
         wrcol(x_row+2,x_lim-9,white,f_to_a(calc.bal_mom,6,2));
         wrcol(y_lim,y_col+1,white,f_to_a(inter.axial[1],6,2));
      end (* case 3 *);
   4 : begin (* 3 dimensional graph of moment-curvature-axial surface *)
         new(picture);
         blank_line(24);
         wrcol(24,1,mess_color,'Input alpha : ');
         getnum(alpha,3,3);
         wrcol(24,23,mess_color,';   beta : ');
         getnum(beta,3,3);
         blank_line(24);
         wrcol(24,1,mess_color,'...calculating, please wait...');
         alpha := alpha * pi / 180;
         beta := beta * pi / 180;
         rotate_picture;
         show_picture;
         run_move;
         dispose(picture);
      end (* case 4 *);
   5 : with sect.data[curr_ax_load] do begin (* moment-curvature for a given axial load *)
         x_save := x_abs_window;
         y_save := y_abs_window;
         x_abs_window := 640;
         y_abs_window := 350;
         max.y := moment[1];
         max.x := curvature[1];
         i := 0;
         for indx := 1 to no_points do begin
            y_data[i] := moment[indx];
            x_data[i] := curvature[indx];
            if y_data[i]>max.y then max.y := y_data[i];
            if x_data[i]>max.x then max.x := x_data[i];
            i := i + 1;
         end (* for *);
         min.x := -max.x;
         min.y := -max.y;
         no_steps := i - 1;
         cls(1,1,23,61,0);
         draw_plot;
         x_row := round(rel_to_abs(0,'y')/14);
         y_col := round(rel_to_abs(0,'x')/8);
         x_lim := x_abs_window div 8;
         y_lim := y_abs_window div 14;
         wrcol(x_row,2,white,f_to_a(min.x,2,6));
         wrcol(2,y_col-6,white,f_to_a(max.y,6,2));
         wrcol(x_row+2,x_lim-9,white,f_to_a(max.x,2,6));
         wrcol(y_lim,y_col+1,white,f_to_a(min.y,6,2));
         x_abs_window := x_save;
         y_abs_window := y_save;
      end (* case 5 *);
   6 : begin (* moment-axial load for a given curvature *)
         x_save := x_abs_window;
         y_save := y_abs_window;
         x_abs_window := 640;
         y_abs_window := 350;
         max.x := 0;
         max.y := sect.data[sect.no_ax_loads-1].axial;
         min.y := sect.data[2].axial;
         indx := 0;
```

```
       overflow := false;
       for i := 2 to sect.no_ax_loads-1 do begin
          y_data[indx] := sect.data[i].axial;
          temp := plot_curv;
          get_elem_mom(y_data[indx],temp,x_data[indx]);
          if overflow
             then overflow := false
             else begin
                if abs(x_data[indx])>max.x then max.x := abs(x_data[indx]);
                indx := indx + 1;
             end;
       end (* for *);
       min.x := -max.x;
       no_steps := indx - 1;
       indx := 2 * no_steps + 1;
       for i := 0 to no_steps do begin
          x_data[indx-i] := -x_data[i];
          y_data[indx-i] := y_data[i];
       end;
       no_steps := indx;
       cls(1,1,23,61,0);
       draw_plot;
       x_row := round(rel_to_abs(0,'y')/14);
       y_col := round(rel_to_abs(0,'x')/8);
       x_lim := x_abs_window div 8;
       y_lim := y_abs_window div 14;
       wrcol(x_row,2,white,f_to_a(min.x,2,6));
       wrcol(2,y_col-6,white,f_to_a(max.y,6,2));
       wrcol(x_row+2,x_lim-9,white,f_to_a(max.x,2,6));
       wrcol(y_lim,y_col+1,white,f_to_a(min.y,6,2));
       x_abs_window := x_save;
       y_abs_window := y_save;
    end (* case 6 *);
7 : begin (* curvature-axial load for a given moment *)
       x_save := x_abs_window;
       y_save := y_abs_window;
       x_abs_window := 640;
       y_abs_window := 350;
       max.x := 0;
       max.y := sect.data[sect.no_ax_loads-1].axial;
       min.y := sect.data[2].axial;
       indx := 0;
       overflow := false;
```

```
                       for i := 2 to sect.no_ax_loads-1 do begin
                          y_data[indx] := sect.data[i].axial;
                          temp := plot_mom;
                          get_elem_curv(y_data[indx],temp,x_data[indx]);
                          if overflow
                             then overflow := false
                             else begin
                                if abs(x_data[indx])>max.x then max.x := abs(x_data[indx]);
                                indx := indx + 1;
                             end;
                       end (* for *);
                       min.x := -max.x;
                       no_steps := indx - 1;
                       indx := 2 * no_steps + 1;
                       for i := 0 to no_steps do begin
                          x_data[indx-i] := -x_data[i];
                          y_data[indx-i] := y_data[i];
                       end;
                       no_steps := indx;
                       min.x := -max.x;
                       cls(1,1,23,61,0);
                       draw_plot;
                       x_row := round(rel_to_abs(0,'y')/14);
                       y_col := round(rel_to_abs(0,'x')/8);
                       x_lim := x_abs_window div 8;
                       y_lim := y_abs_window div 14;
                       wrcol(x_row,2,white,f_to_a(min.x,2,6));
                       wrcol(2,y_col-6,white,f_to_a(max.y,6,2));
                       wrcol(x_row+2,x_lim-9,white,f_to_a(max.x,2,6));
                       wrcol(y_lim,y_col+1,white,f_to_a(min.y,6,2));
                       x_abs_window := x_save;
                       y_abs_window := y_save;
                    end (* case 7 *);
             end (* case *);
          end;

(*---------------------------------------------------------------------------*)

function concrete(conf : boolean; strain : real) : real;
         var rat,stress : real;
         begin
            stress := 0;
            if strain > 0
```

```
               then begin
                  rat := strain / calc.e_0;
                  if conf
                     then if strain < calc.e_0
                        then stress :=  calc.k * prop.conc_str * (2.0*rat - sqr(rat))
                        else begin
                           stress :=  calc.k * prop.conc_str
                                    * (1.0 - calc.z_conf * (strain - calc.e_0));
                           if stress < 0.2 * calc.k * prop.conc_str
                              then stress := 0.2 * calc.k * prop.conc_str;
                        end
                     else if strain < calc.e_0
                        then stress := prop.conc_str * (2.0*rat - sqr(rat))
                        else stress := prop.conc_str * (1.0 - calc.z_unconf * (strain-calc.e_0));
               end;
            if stress > 0
               then concrete := stress
               else concrete := 0;
         end;

(*---------------------------------------------------------------------------*)

function steel(strain : real) : real;
         var
            stress : real;
            zone : integer;
         begin
            if strain > prop.strain_at_ult
               then zone := 1
               else if strain > prop.strain_hard
                  then zone := 2
                  else if strain > calc.yield_strain
                     then zone := 3
                     else if strain > -calc.yield_strain
                        then zone := 4
                        else if strain > -prop.strain_hard
                           then zone := 5
                           else if strain > -prop.strain_at_ult
                              then zone := 6
                              else zone := 7;
            case zone of
               1 : stress := prop.steel_ult;
               2 : stress :=  prop.steel_yield
```

```
                            + calc.mod_sh * (strain - prop.strain_hard);
               3 : stress := prop.steel_yield;
               4 : stress := 29500.0 * strain;
               5 : stress := -prop.steel_yield;
               6 : stress := -prop.steel_yield
                            + calc.mod_sh * (strain + prop.strain_hard);
               7 : stress := -prop.steel_ult
            end (* case *);
            steel := stress - concrete(true,strain);
         end;

(*--------------------------------------------------------------------------*)

function stress_block(kd,ext_fib_strain : real; var x_bar : real) : real;
         var
            strain : real;
            force : real;
            str_inc : real;
            i : integer;
            x : real;
            moment : real;
            t_slce : real;
            d_slce : real;
            sb_depth : real;
            min_str : real;
            force_inc : real;
            unconf_width : real;
            cover,cover_inc : real;
            core,core_inc : real;
         begin
            x_bar := 0.0;
            unconf_width := prop.width - calc.conf_width;
            core := 0;
            core_inc := 0;
            cover := 0;
            cover_inc := 0;
            force := 0;
            force_inc := 0;
            if (kd < 0) or (ext_fib_strain < 0)
               then stress_block := 0.0
               else begin
                  force := 0.0;
                  moment := 0.0;
                  if kd > prop.depth
                     then begin
                        sb_depth := prop.depth;
                        min_str := ext_fib_strain * (kd-prop.depth) / kd;
                     end
                     else begin
                        sb_depth := kd;
                        min_str := 0;
                     end;
                  str_inc := (ext_fib_strain-min_str) / prop.no_slices;
                  t_slce := sb_depth / prop.no_slices;
                  for i := 1 to prop.no_slices do begin
                     x := i;
                     strain := min_str + str_inc * (x-0.5);
                     cover_inc :=  concrete(false,strain)
                                * unconf_width * t_slce;
                     core_inc := 0;
                     d_slce := t_slce * (prop.no_slices + 0.5 - x);
                     if (d_slce<prop.steel[1].depth) or (d_slce>calc.eff_depth)
                        then cover_inc :=  cover_inc
                                        + concrete(false,strain)
                                        * calc.conf_width * t_slce
                        else core_inc :=   concrete(true,strain)
                                        * calc.conf_width * t_slce;
                     force_inc := core_inc + cover_inc;
                     core := core + core_inc;
                     cover := cover + cover_inc;
                     force := force + force_inc;
                     moment := moment + force_inc * d_slce;
                  end;
                  if abs(force) < 0.0001
                     then x_bar := 0
                     else x_bar := moment / force;
                  stress_block := force;
               end;
         end;

(*----------------------------------------------------------------------------*)

procedure get_forces(mode : integer; axial:real; var mom,curv:real);
          var
             old_kd : real;
             str1,str2,str3 : real;
```

```
   force1,force2,force3,force4 : real;
   complete : boolean;
   ax,ax_load,old_ax : real;
   pass : integer;
   factor : real;
   x_bar : real;
   deriv : real;
   strain : real;
   alpha : real;
   zone1,zone2,zone3,zone4 : boolean;
   first_pass : boolean;
begin
   ax := axial;
   if axial = 0 then ax := 0.1;
   zone4 := (axial > calc.kd3_ax);
   zone3 := (axial <= calc.kd3_ax) and (axial > calc.kd2_ax) and (prop.layers = 3);
   zone2 := (axial <= calc.kd2_ax) and (axial > calc.kd1_ax);
   zone1 := axial <= calc.kd1_ax;
   strain := prop.ult_conc_strain;
   factor := 1.01;
   first_pass := false;
   if mode = 1
      then if zone4
         then begin
            old_kd := calc.eff_depth;
            old_ax := calc.kd3_ax;
            kd := factor * old_kd;
         end
         else if zone3
            then begin
               old_kd := prop.steel[2].depth;
               old_ax := calc.kd2_ax;
               kd := factor * old_kd;
            end
            else begin
               old_kd := prop.steel[1].depth;
               old_ax := calc.kd1_ax;
               if zone2
                  then kd := factor * old_kd
                  else kd := old_kd / factor;
            end
      else begin
         kd := prop.depth / 2;
      first_pass := true;
   end;
pass := 0;
alpha := 1;
complete := false;
repeat
   pass := pass + 1;
   if mode = 3 then strain := kd * curv;
   str1 := strain * (kd - prop.steel[1].depth) / kd;
   force1 := steel(str1) * prop.steel[1].area;
   str3 := strain * (kd - calc.eff_depth) / kd;
   force3 := steel(str3) * prop.steel[prop.layers].area;
   if prop.layers = 3
      then begin
         str2 := strain * (kd - prop.steel[2].depth) / kd;
         force2 := steel(str2) * prop.steel[2].area;
      end
      else force2 := 0;
   force4 := stress_block(kd,strain,x_bar);
   ax_load := force1 + force2 + force3 + force4;
   if first_pass
      then begin
         first_pass := false;
         old_ax := ax_load;
         old_kd := kd;
         kd := kd * factor;
      end
      else if abs(ax_load-ax) < 0.1
         then complete := true
         else begin
            if ax_load = old_ax
               then deriv := deriv * 100.0
               else deriv := (ax_load - old_ax) / (kd - old_kd);
            old_kd := kd;
            kd := kd - alpha*(ax_load - ax) / deriv;
            old_ax := ax_load;
         end;
   if (pass > 20) and (alpha = 1)
      then begin
         pass := 0;
         alpha := 0.5;
      end;
until (complete) or (pass>20);
```

```
          mom :=  force4 * (plast_cen - x_bar)
             + force3 * (plast_cen - calc.eff_depth)
             + force1 * (plast_cen - prop.steel[1].depth)
             + force2 * (plast_cen - prop.steel[2].depth);
          if mode = 1 then curv := strain / kd;
       end;

(*------------------------------------------------------------------------------*)

overlay procedure moment_interaction;
       var
          ax_inc : real;
          i : integer;
          mom,curv : real;
       begin
          with inter do begin
             moment[1] := 0.0;
             axial[1] := calc.ult_tens;
             curvature[1] := 0.0;
             moment[50] := 0.0;
             axial[50] := calc.ult_comp;
             curvature[50] := 0.0;
             ax_inc := (axial[50] - calc.ax_min) / 49;
             wrcol(25,38,descr_color,'calculating interaction point no.    of 50');
             i := 49;
             while (i>1) and not(keyPressed) do begin
                wrcol(25,72,data_color,i_to_a(50-i));
                axial[i] := calc.ax_min + ax_inc*(i-2);
                get_forces(1,axial[i],mom,curv);
                moment[i] := mom;
                curvature[i] := curv;
                i := i - 1;
             end;
             prop.inter_flag := true;
             save_inter_file;
          end (* with *);
       end;

(*------------------------------------------------------------------------------*)

overlay procedure show_calc_prop;
       var
          area : real;
```

```
   x_hold,y_hold : integer;
begin
   with calc do begin
     cls(1,1,25,80,0);
     draw_box(0,0,639,y_abs_window-1);
     drawc(0,35,639,35);
     drawc(0,37,639,37);
     mess := 'CALCULATED PROPERTIES';
     wrcol(2,30,title_color,mess);
     mess := 'Reinforcement ratio -';
     wrcol(5,5,subtitle_color,mess);
     long_rho := long_rho * 100.0;
     mess := concat('   longitudinal : ',f_to_a(long_rho,2,2),' percent');
     wrcol(6,5,menu_color,mess);
     long_rho := long_rho / 100.0;
     trans_rho := trans_rho * 100.0;
     mess := concat('   transverse : ',f_to_a(trans_rho,1,4),' percent');
     wrcol(7,5,menu_color,mess);
     trans_rho := trans_rho / 100.0;
     mess := concat('Effective depth : ',f_to_a(eff_depth,3,3),' in.');
     wrcol(9,5,subtitle_color,mess);
     mess := concat('Concrete modulus : ',f_to_a(conc_mod,5,2),' ksi');
     wrcol(11,5,subtitle_color,mess);
     mess := concat('Concrete tensile str : ',f_to_a(conc_tens,4,2),' psi');
     wrcol(13,5,subtitle_color,mess);
     mess := concat('Modular ratio : ',f_to_a(mod_rat,2,2));
     wrcol(15,5,subtitle_color,mess);
     mess := 'Balanced condition -';
     wrcol(17,5,subtitle_color,mess);
     mess := concat('   moment : ',f_to_a(bal_mom,6,2),' kip in.     ');
     wrcol(18,5,menu_color,mess);
     mess := concat('   axial load : ',f_to_a(bal_ax,5,2),' kips    ');
     wrcol(19,5,menu_color,mess);
     mess := concat('   neutral axis : ',f_to_a(bal_kd,3,3),' in.   ');
     wrcol(20,5,menu_color,mess);
     mess := concat('Transformed area : ',f_to_a(area,5,2),' sq.in.');
     wrcol(5,45,subtitle_color,mess);
     mess := 'Transformed moment of inertia -';
     wrcol(7,45,subtitle_color,mess);
     mess := concat('   uncracked : ',f_to_a(uncr_mom_iner,7,2),' in.^4');
     wrcol(8,45,menu_color,mess);
     mess := concat('   cracked : ',f_to_a(cr_mom_iner,7,2),' in.^4');
     wrcol(9,45,menu_color,mess);
     mess := concat('Neutral axis @ P=0 : ',f_to_a(neutral_axis,3,3),' in.');
     wrcol(11,45,subtitle_color,mess);
     mess := 'Confined core -';
     wrcol(13,45,subtitle_color,mess);
     mess := concat('   width : ',f_to_a(conf_width,3,3),' in.');
     wrcol(14,45,menu_color,mess);
     mess := concat('   depth : ',f_to_a(conf_depth,3,3),' in.');
     wrcol(15,45,menu_color,mess);
     area := conf_depth * conf_width;
     mess := concat('   area : ',f_to_a(area,5,2),' sq.in.');
     wrcol(16,45,menu_color,mess);
     mess := 'Ultimate axial loads -';
     wrcol(18,45,subtitle_color,mess);
     mess := concat('   compression : ',f_to_a(ult_comp,5,2),' kips    ');
     wrcol(19,45,menu_color,mess);
     mess := concat('   tension : ',f_to_a(ult_tens,3,2),' kips  ');
     wrcol(20,45,menu_color,mess);
     wrcol(22,5,subtitle_color,'File name : ');
     wrcol(22,17,data_color,prop.filename);
     blank_line(24);
     if error
        then wrcol(24,1,error_color,'Error was detected in moment interaction routine')
        else wrcol(24,1,mess_color,'Press return to graph moment interaction diagram.');
     blank_line(25);
     wrcol(25,1,mess_color,'Press any other key to return to properties menu....');
     getc(ans);
     if (key_pressed=CARRIAGE)
        then begin
           cls(1,1,25,80,0);
           x_hold := x_abs_window;
           y_hold := y_abs_window;
           x_abs_window := 640;
           graph_no := 3;
           show_graph;
           blank_line(24);
           blank_line(25);
           wrcol(25,1,mess_color,'Press any key to return to properties menu....');
           repeat until keyPressed;
           x_abs_window := x_hold;
        end;
   end (* with *);
   cls(1,1,25,80,0);
end;
```

```
(*-----------------------------------------------------------------------------*)

overlay procedure calc_prop;
          var
             gr_area : real;
             steel_area : real;
             a,b,c,deter : real;
             core_vol,trans_vol : real;
             fsmax,fcmax : real;
             comp_stress : real;
             d_prime : real;
             force1,force2,force3,force4 : real;
             str1,str2,str3 : real;
             x_bar : real;
             i,j : integer;
        begin
           prop.no_slices := 50;
           prop.strain_at_ult := 0.1;
           with calc do begin
              eff_depth := prop.steel[prop.layers].depth;
              conf_depth := abs(eff_depth-prop.steel[1].depth);
              conf_width := prop.width - (prop.depth-conf_depth);
              gr_area := prop.width * prop.depth;
              steel_area := prop.steel[1].area + prop.steel[2].area + prop.steel[3].area;
              long_rho := steel_area / gr_area;
              if prop.width=prop.depth
                then trans_vol :=  prop.tie_bar_area * conf_depth * prop.tie1_no_legs
                else if prop.width>prop.depth
                   then trans_vol :=  prop.tie_bar_area
                                    * (conf_width*prop.tie1_no_legs
                                      +conf_depth*prop.tie2_no_legs)
                   else trans_vol :=  prop.tie_bar_area
                                    * (conf_depth*prop.tie1_no_legs
                                      +conf_width*prop.tie2_no_legs);
             core_vol := conf_depth * conf_width * prop.tie_spacing;
             trans_rho := trans_vol / core_vol;
             conc_mod := 57.0 * sqrt(prop.conc_str*1000.0);
             mod_rat := 29500.0 / conc_mod;
             area := (gr_area-steel_area) + mod_rat*(steel_area);
             conc_tens := 7.5 * sqrt(prop.conc_str*1000.0);
             uncr_mom_iner :=  prop.width * prop.depth * sqr(prop.depth) / 3.0
                                  + mod_rat *
```

```
                          (prop.steel[1].area * sqr(prop.steel[1].depth)
                          +prop.steel[2].area * sqr(prop.steel[2].depth)
                          +prop.steel[3].area * sqr(prop.steel[3].depth))
                     - area * sqr(prop.depth/2.0);
crack_moment := conc_tens * uncr_mom_iner / prop.depth * 2.0;
a := prop.width / 2.0;
b :=  (mod_rat-1)
    * (prop.steel[1].area
      +prop.steel[2].area
      +prop.steel[3].area);
c :=  (mod_rat-1)
    * (prop.steel[1].area * prop.steel[1].depth
      +prop.steel[2].area * prop.steel[2].depth
      +prop.steel[3].area * prop.steel[3].depth);
deter := sqr(b) + 4.0 * a * c;
neutral_axis := (b + sqrt(deter)) / 2.0 / a;
cr_mom_iner :=  prop.width * neutral_axis * sqr(neutral_axis) / 3.0
                   + (mod_rat-1)
                   * (prop.steel[1].area * sqr(neutral_axis-prop.steel[1].depth)
                     +prop.steel[2].area * sqr(neutral_axis-prop.steel[2].depth)
                     +prop.steel[3].area * sqr(neutral_axis-prop.steel[3].depth));
fcmax := prop.conc_str * (1 + trans_rho*prop.steel_yield/prop.conc_str);
yield_strain := prop.steel_yield / 29500.0;
prop.strain_hard := 0.01;
mod_sh :=  (prop.steel_ult - prop.steel_yield - prop.strain_hard)
         / (prop.strain_at_ult - prop.strain_hard);
k := 1.0 + trans_rho * prop.steel_yield / prop.conc_str;
e_0 := 0.002 * k;
e_50_h :=  0.75 * trans_rho
              * sqrt(conf_width/prop.tie_spacing);
e_50_u := (3.0 + 2*prop.conc_str) / (prop.conc_str - 1.0) / 1000.0;
z_conf := 0.5 / (e_50_h + e_50_u - e_0);
z_unconf := 0.5 / (e_50_u - e_0);
ult_comp :=  stress_block(1.0E12,prop.ult_conc_strain,x_bar)
                + steel(prop.ult_conc_strain) * steel_area;
ult_tens := -prop.steel_yield * steel_area;
beta := 0.85 - 0.05*(prop.conc_str-4.0);
if beta>0.85 then beta := 0.85;
if beta<0.65 then beta := 0.65;
bal_kd :=   eff_depth * prop.ult_conc_strain
                / (prop.ult_conc_strain + yield_strain);
str1 :=   prop.ult_conc_strain
       * (bal_kd - prop.steel[1].depth)
       / bal_kd;
force1 := steel(str1) * prop.steel[1].area;
if prop.layers = 3
   then begin
      str2 :=   prop.ult_conc_strain
             * (bal_kd - prop.steel[2].depth)
             / bal_kd;
      force2 := steel(str2) * prop.steel[2].area;
   end
   else force2 := 0.0;
force3 := -prop.steel_yield * prop.steel[prop.layers].area;
force4 := stress_block(bal_kd,prop.ult_conc_strain,x_bar);
bal_ax := force1 + force2 + force3 + force4;
plast_cen := prop.depth / 2.0;
bal_mom :=  force4 * (plast_cen - x_bar)
          + force1 * (plast_cen - prop.steel[1].depth)
          + force2 * (plast_cen - prop.steel[2].depth)
          + force3 * (plast_cen - eff_depth);
if sect.ductility = 0 then sect.ductility := 1;
force3 := steel(-yield_strain) * prop.steel[prop.layers].area;
str1 := -yield_strain * prop.steel[1].depth / eff_depth;
force1 := steel(str1) * prop.steel[1].area;
if prop.layers = 3
   then begin
      str2 := -yield_strain * prop.steel[2].depth / eff_depth;
      force2 := steel(str2) * prop.steel[2].area;
   end
   else force2 := 0.0;
ax_min := force1 + force2 + force3;
ax_min_curv := yield_strain / eff_depth;
force1 := 0;
str3 := prop.ult_conc_strain * (prop.steel[1].depth-eff_depth) / prop.steel[1].depth;
force3 := steel(str3) * prop.steel[prop.layers].area;
if prop.layers = 3
   then begin
      str2 := prop.ult_conc_strain * (prop.steel[1].depth-prop.steel[2].depth) / prop.steel[1].depth;
      force2 := steel(str2) * prop.steel[2].area;
   end
   else force2 := 0;
force4 := stress_block(prop.steel[1].depth,prop.ult_conc_strain,x_bar);
kd1_ax := force1 + force2 + force3 + force4;
force3 := 0;
str1 := prop.ult_conc_strain * (eff_depth-prop.steel[1].depth) / eff_depth;
```

```
          force1 := steel(str1) * prop.steel[1].area;
          if prop.layers = 3
             then begin
                str2 := prop.ult_conc_strain * (eff_depth-prop.steel[2].depth) / eff_depth;
                force2 := steel(str2) * prop.steel[2].area;
             end
             else force2 := 0;
          force4 := stress_block(eff_depth,prop.ult_conc_strain,x_bar);
          kd3_ax := force1 + force2 + force3 + force4;
          if prop.layers = 3
             then begin
                force2 := 0;
                str3 := prop.ult_conc_strain * (prop.steel[2].depth-eff_depth) / prop.steel[2].depth;
                force3 := steel(str3) * prop.steel[prop.layers].area;
                str1 := prop.ult_conc_strain * (prop.steel[2].depth-prop.steel[1].depth) / prop.steel[2].depth;
                force1 := steel(str1) * prop.steel[1].area;
                force4 := stress_block(prop.steel[2].depth,prop.ult_conc_strain,x_bar);
                kd2_ax := force1 + force2 + force3 + force4;
             end
             else kd2_ax := kd1_ax;
          if not(load_inter_file)
             then moment_interaction;
       end (* with *);
     end;

(*----------------------------------------------------------------------*)

procedure dummy5;
          begin
          end;

(*----------------------------------------------------------------------*)

overlay procedure write_prop;
          var i : integer;
          begin
             with prop do begin
               for i := 1 to 18 do
                  case i of
                     1  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(length,4,2));
                     2  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(width,3,2));
                     3  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(depth,3,2));
                     4  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(conc_str,2,3));
                     5  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(ult_conc_strain,1,4));
                     6  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel_yield,3,2));
                     7  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel_ult,3,2));
                     8  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(tie_bar_area,1,3));
                     9  : wrcol(prop_row[i],prop_col[i],data_color,f_to_a(tie_spacing,2,3));
                     10 : wrcol(prop_row[i],prop_col[i],data_color,i_to_a(layers));
                     11 : if layers>0
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[1].area,3,2));
                     12 : if layers>0
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[1].depth,3,2));
                     13 : if layers>1
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[2].area,3,2));
                     14 : if layers>1
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[2].depth,3,2));
                     15 : if layers>2
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[3].area,3,2));
                     16 : if layers>2
                             then wrcol(prop_row[i],prop_col[i],data_color,f_to_a(steel[3].depth,3,2));
                     17 : wrcol(prop_row[i],prop_col[i],data_color,i_to_a(tie1_no_legs));
                     18 : if width<>depth
                             then wrcol(prop_row[i],prop_col[i],data_color,i_to_a(tie2_no_legs))
                             else cls(prop_row[i],prop_col[i],prop_row[i],prop_col[i]+2,0);
               end (* case *);
               wrcol(22,17,data_color,filename);
             end (* with *);
          end;

(*----------------------------------------------------------------------------*)

overlay procedure write_load;
          var i,offset : integer;
          begin
             with load do begin
                wrcol(load_row[1],load_col[1],data_color,load_type_descr[lateral_type]);
                case lateral_type of
                   1 : begin
                          if load_control
                             then wrcol(load_row[2],load_col[2],data_color,f_to_a(lateral[1],4,1))
                             else wrcol(load_row[2],load_col[2],data_color,f_to_a(lateral[1],1,4));
                          wrcol(load_row[3],load_col[3],data_color,load_type_descr[axial_type]);
                          offset := 1;
                       end;
                   2 : begin
```

```
                          if load_control
                             then wrcol(load_row[2],load_col[2],data_color,f_to_a(lateral[1],4,1))
                             else wrcol(load_row[2],load_col[2],data_color,f_to_a(lateral[1],1,4));
                          if load_control
                             then wrcol(load_row[3],load_col[3],data_color,f_to_a(lateral[2],4,1))
                             else wrcol(load_row[3],load_col[3],data_color,f_to_a(lateral[2],1,4));
                          wrcol(load_row[4],load_col[4],data_color,load_type_descr[axial_type]);
                          offset := 2;
                       end;
                   3 : begin
                          for i := 1 to 10 do
                             if load_control
                                then wrcol(load_row[i+1],load_col[i+1],data_color,f_to_a(lateral[i],4,1))
                                else wrcol(load_row[i+1],load_col[i+1],data_color,f_to_a(lateral[i],1,4));
                          wrcol(load_row[12],load_col[12],data_color,load_type_descr[axial_type]);
                          offset := 10;
                       end (* begin *);
                end (* case *);
                case axial_type of
                   1 : wrcol(load_row[3+offset],load_col[3+offset],data_color,f_to_a(axial[1],4,1));
                   2 : begin
                          wrcol(load_row[3+offset],load_col[3+offset],data_color,f_to_a(axial[1],4,1));
                          wrcol(load_row[4+offset],load_col[4+offset],data_color,f_to_a(axial[2],4,1));
                       end;
                   3 : for i := 1 to 10 do
                          wrcol(load_row[i+2+offset],load_col[i+2+offset],data_color,f_to_a(axial[i],4,1));
                end (* case *);
                wrcol(22,17,data_color,filename);
             end (* with *);
          end (* write_load *);

(*---------------------------------------------------------------------------*)

overlay procedure write_sect;
          begin
               cls(sect_row[1],sect_col[1],sect_row[1],sect_col[1]+10,0);
               wrcol(sect_row[1],sect_col[1],data_color,f_to_a(sect.data[curr_ax_load].axial,6,2));
               cls(sect_row[2],sect_col[2],sect_row[2],sect_col[2]+2,0);
               wrcol(sect_row[2],sect_col[2],data_color,i_to_a(curr_ax_load));
               cls(sect_row[3],sect_col[3],sect_row[3],sect_col[3]+2,0);
               wrcol(sect_row[3],sect_col[3],data_color,i_to_a(sect.no_ax_loads));
               cls(sect_row[4],sect_col[4],sect_row[4],sect_col[4]+2,0);
               wrcol(sect_row[4],sect_col[4],data_color,i_to_a(sect.data[curr_ax_load].no_points));
               for i := 1 to 20 do begin
                  cls(sect_row[i+4],sect_col[i+4],sect_row[i+4],sect_col[i+4]+9,0);
                  wrcol(sect_row[i+4],sect_col[i+4],data_color,f_to_a(-sect.data[curr_ax_load].moment[i],6,2));
                  cls(sect_row[i+24],sect_col[i+24],sect_row[i+24],sect_col[i+24]+9,0);
                  wrcol(sect_row[i+24],sect_col[i+24],data_color,f_to_a(-sect.data[curr_ax_load].curvature[i],1,7));
               end (* for *);
               wrcol(22,17,data_color,sect.filename);
          end;

(*---------------------------------------------------------------------------*)

overlay procedure print_sect;
          var
             i,j : integer;
             finished : boolean;
          begin
             blank_line(25);
             wrcol(25,1,mess_color,'...printing, hit any key to pause or stop...');
             finished := false;
             i := 1;
             while (i <= sect.no_ax_loads) and not(finished) do
                with sect.data[i] do begin
                   writeln(lst);
                   writeln(lst,'AXIAL = ',axial);
                   writeln(lst);
                   j := 1;
                   while (j <= no_points) and not(finished) do begin
                      if keyPressed
                         then begin
                            blank_line(24);
                            wrcol(24,1,mess_color,'Hit ESC to abort, any other key to continue...');
                            getc(ans);
                            blank_line(24);
                            if key_pressed = ESC
                               then finished := true
                               else finished := false;
                         end;
                      writeln(lst);
                      writeln(lst,'MOMENT = ',moment[j]);
                      writeln(lst,'CURVATURE = ',curvature[j]);
                      j := j + 1;
                   end;
                i := i + 1;
                end;
          end;

(*--------------------------------------------------------------------------*)

procedure dummy2;
          begin
          end;

(*--------------------------------------------------------------------------*)

overlay procedure prop_menu;
          var bottom,i : integer;
          begin
               cls(1,1,25,80,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
               drawc(0,35,479,35);
               drawc(0,280,479,280);
               wrcol(2,18,title_color,'COLUMN AND SECTION PROPERTIES');
               wrcol(4,3,descr_color,'Length :          in.');
               wrcol(6,3,descr_color,'Width :         in.');
               wrcol(8,3,descr_color,'Depth :         in.');
               wrcol(10,3,descr_color,'Concrete properties -');
               wrcol(11,6,descr_color,'strength :           ksi');
               wrcol(12,6,descr_color,'ult strain :');
               wrcol(14,3,descr_color,'Steel properties -');
               wrcol(15,6,descr_color,'yield stress :         ksi');
               wrcol(16,6,descr_color,'ult stress :         ksi');
               wrcol(18,3,descr_color,'Tie bar area :        sq.in.');
               wrcol(19,3,descr_color,'Tie spacing :         in.');
               wrcol(4,35,descr_color,'Layers of steel -');
               if prop.layers <= 0 then prop.layers := 3;
               for i := 0 to prop.layers-1 do
                   begin
                        wrcol(6+4*i,35,descr_color,concat('Layer ',i_to_a(i+1)));
                        wrcol(7+4*i,38,descr_color,'Area :         sq.in.');
                        wrcol(8+4*i,38,descr_color,'Depth :         in.');
                   end;
               if prop.width=prop.depth
                  then wrcol(18,35,descr_color,'No. stirrup legs  :')
                  else begin
                     wrcol(18,35,descr_color,'No. long tie legs :');
                     wrcol(19,35,descr_color,'No. short tie legs :');
                  end;
               wrcol(22,5,menu_color,'File name :');
               wrcol(2,62,menu_color,'F1   : help');
               wrcol(3,62,menu_color,'F2   : load file');
               wrcol(4,62,menu_color,'F3   : save file');
               wrcol(5,62,menu_color,'F4   : clear data');
               wrcol(6,62,menu_color,'F5   : make ASCII');
               wrcol(7,62,menu_color,'       inter file');
               wrcol(8,62,menu_color,'PGDN : calculated');
               wrcol(9,62,menu_color,'       properties');
               wrcol(10,62,menu_color,'RET  : insert new');
               wrcol(11,62,menu_color,'       value');
               wrcol(12,62,menu_color,'ESC  : return to');
               wrcol(13,62,menu_color,'       main menu');
          end;

(*---------------------------------------------------------------------------*)

overlay procedure load_menu;
          var i : integer;
          begin
               cls(1,1,25,80,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
               drawc(0,35,479,35);
               drawc(0,33,479,33);
               drawc(0,63,479,63);
               drawc(240,35,240,280);
               drawc(0,280,x_abs_window-1,280);
               wrcol(2,28,title_color,'LOADING');
               wrcol(4,12,subtitle_color,'LATERAL');
               wrcol(4,42,subtitle_color,'AXIAL');
               wrcol(6,3,descr_color,'Loading type :');
               wrcol(6,33,descr_color,'Loading type :');
               with load do begin
                  case lateral_type of
                    1 : if load_control
                          then wrcol(8,3,descr_color,'Load :          kips')
                          else wrcol(8,3,descr_color,'Disp :          in.');
                    2 : begin
                            if load_control
                               then wrcol(8,3,descr_color,'Initial load :        kips')
                               else wrcol(8,3,descr_color,'Initial disp :        in.');
                            if load_control
```

```
                                then wrcol(10,3,descr_color,'Final load :         kips')
                                else wrcol(10,3,descr_color,'Final disp :         in.');
                        end;
                    3 : if load_control
                           then wrcol(8,10,descr_color,'Load values (kips) :')
                           else wrcol(8,10,descr_color,'Disp values (inches):');
                  end (* case *);
                  case axial_type of
                    1 : wrcol(8,33,descr_color,'Load :');
                    2 : if load.load_control
                           then begin
                              wrcol(8,33,descr_color,'Initial load :       kips');
                              wrcol(10,33,descr_color,'Final load :       kips');
                           end
                           else begin
                              wrcol(8,33,descr_color,'Gravity load :       kips');
                              wrcol(10,33,descr_color,'L-R slope :               ');
                           end;
                    3 : wrcol(8,40,descr_color,'Load values (kips) :');
                  end (* case *);
                  wrcol(22,5,menu_color,'File name :');
                  wrcol(2,62,menu_color,'F1  : help');
                  wrcol(3,62,menu_color,'F2  : load file');
                  wrcol(4,62,menu_color,'F3  : save file');
                  wrcol(5,62,menu_color,'F4  : clear data');
                  wrcol(6,62,menu_color,'F5  : show loads');
                  wrcol(7,62,menu_color,'F7  : switch to');
                  if load_control
                     then wrcol(8,62,menu_color,'      disp control')
                     else wrcol(8,62,menu_color,'      load control');
                  wrcol(9,62,menu_color,'RET : insert new');
                  wrcol(10,62,menu_color,'      value');
                  wrcol(11,62,menu_color,'ESC : return to');
                  wrcol(12,62,menu_color,'      main menu');
               end (* with *);
          end;

(*----------------------------------------------------------------------------*)

overlay procedure sect_menu;
          begin
               cls(1,1,25,90,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
```

```
drawc(0,34,479,34);
drawc(0,36,479,36);
drawc(0,280,479,280);
drawc(0,92,479,92);
drawc(0,115,234,115);
drawc(236,115,479,115);
drawc(22,115,22,280);
drawc(114,115,114,280);
drawc(116,115,116,280);
drawc(139,115,139,280);
drawc(234,92,234,280);
drawc(236,92,236,280);
drawc(262,115,262,280);
drawc(354,115,354,280);
drawc(356,115,356,280);
drawc(380,115,380,280);
wrcol(10,2,index_color,'1');
wrcol(11,2,index_color,'2');
wrcol(12,2,index_color,'3');
wrcol(13,2,index_color,'4');
wrcol(14,2,index_color,'5');
wrcol(15,2,index_color,'6');
wrcol(16,2,index_color,'7');
wrcol(17,2,index_color,'8');
wrcol(18,2,index_color,'9');
wrcol(19,1,index_color,'10');
wrcol(10,16,index_color,'11');
wrcol(11,16,index_color,'12');
wrcol(12,16,index_color,'13');
wrcol(13,16,index_color,'14');
wrcol(14,16,index_color,'15');
wrcol(15,16,index_color,'16');
wrcol(16,16,index_color,'17');
wrcol(17,16,index_color,'18');
wrcol(18,16,index_color,'19');
wrcol(19,16,index_color,'20');
wrcol(10,32,index_color,'1');
wrcol(11,32,index_color,'2');
wrcol(12,32,index_color,'3');
wrcol(13,32,index_color,'4');
wrcol(14,32,index_color,'5');
wrcol(15,32,index_color,'6');
wrcol(16,32,index_color,'7');
```

```
          wrcol(17,32,index_color,'8');
          wrcol(18,32,index_color,'9');
          wrcol(19,31,index_color,'10');
          wrcol(10,46,index_color,'11');
          wrcol(11,46,index_color,'12');
          wrcol(12,46,index_color,'13');
          wrcol(13,46,index_color,'14');
          wrcol(14,46,index_color,'15');
          wrcol(15,46,index_color,'16');
          wrcol(16,46,index_color,'17');
          wrcol(17,46,index_color,'18');
          wrcol(18,46,index_color,'19');
          wrcol(19,46,index_color,'20');
          wrcol(2,23,title_color,'SECTION RESPONSE');
          wrcol(4,19,descr_color,'Axial load =           kips');
          wrcol(5,16,descr_color,'Axial load number    of    ');
          wrcol(6,17,descr_color,'Number of data points :');
          wrcol(8,8,subtitle_color,'MOMENT(kip-in):');
          wrcol(8,38,subtitle_color,'CURVATURE (1/in):');
          wrcol(22,5,menu_color,'File name :');
          wrcol(2,62,menu_color,'F1   : help');
          wrcol(3,62,menu_color,'F2   : load file');
          wrcol(4,62,menu_color,'F3   : save file');
          wrcol(5,62,menu_color,'F4   : smooth data');
          wrcol(6,62,menu_color,'F5   : clear data');
          wrcol(7,62,menu_color,'F6   : 3D graph');
          wrcol(8,62,menu_color,'F7   : print data');
          wrcol(9,62,menu_color,'F8   : ASCII file');
          wrcol(10,62,menu_color,'F9   : 2D graph');
          wrcol(11,62,menu_color,'PGDN : next ax load');
          wrcol(12,62,menu_color,'PGUP : prev ax load');
          wrcol(13,62,menu_color,'RET  : input new');
          wrcol(14,62,menu_color,'       value');
          wrcol(15,62,menu_color,'ESC  : return to');
          wrcol(16,62,menu_color,'      main menu');
          draw_box(0,0,x_abs_window-1,y_abs_window-1);
      end;

(*-------------------------------------------------------------------*)

procedure dummy1;
      begin
      end;
```

```
(*---------------------------------------------------------------------------*)

overlay procedure get_lat_load(def,gravity,slope : real; var ax,lat : real);
   var
      old_lat,curv_inc : real;
      elem_disp,elem_rot : real;
      elem_mom,elem_curv,elem_len : real;
      end_mom,end_curv : real;
      elem : integer;
      pass : integer;
      good : boolean;
      complete : boolean;
      redo : boolean;
      top_hinging,bottom_hinging : boolean;
      old_th,old_bh : boolean;
      top_curv,top_mom : real;
      test_curv : real;
      old_mom,old_curv,old_defl  : real;
      deriv : real;
      trans_mom,test_mom : real;
      transition : boolean;
      alpha : real;
      subpass : integer;
      sine,cosine : real;
   begin
      overflow := false;
      if def = 0
         then begin
            lat := 0.0;
            ax := gravity;
         end
         else begin
            alpha := 1;
            elem_len := prop.length / defl.no_elem;
            complete := false;
            pass := 0;
            end_curv := def / sqr(prop.length/2);
            old_curv := 0;
            old_defl := 0;
            trans_mom := 0;
            bottom_hinging := false;
            top_hinging := false;
```

```
repeat with seg^.data[load_step] do begin
   pass := pass + 1;
   mess := concat(i_to_a(pass),'  ');
   wrcol(21,72,data_color,mess);
   lat := 0;
   ax := gravity;
   good := false;
   top_curv := defl.ratio * end_curv;
   subpass := 0;
   repeat
      subpass := subpass + 1;
      old_lat := lat;
      get_elem_mom(ax,end_curv,end_mom);
      get_elem_mom(ax,top_curv,top_mom);
      lat := (end_mom - top_mom - gravity*def)
            / (prop.length + slope*def);
      if abs(lat-old_lat) < defl.toler * abs(lat)
         then good := true
         else lat := (lat + old_lat) / 2;
      ax := slope * lat + gravity;
      if subpass > 100
         then good := true;
   until good;
   curv_inc := -end_curv / defl.no_elem;
   curvature[0] := end_curv;
   moment[0] := end_mom;
   if top_hinging or bottom_hinging
      then defl.hinge_length := round(prop.depth/elem_len)
      else defl.hinge_length := -1;
   rotation[0] := 0;
   deflection[0] := 0;
   get_elem_mom(ax,top_curv,moment[defl.no_elem]);
   curvature[defl.no_elem] := top_curv;
   elem := 1;
   transition := false;
   redo := false;
   while (elem<=defl.no_elem) and (not(overflow)) do begin
      mess := concat(i_to_a(elem),'  ');
      wrcol(22,75,data_color,mess);
      rotation[elem] :=  rotation[elem-1]
                       + curvature[elem-1]*elem_len;
      sine := sin(rotation[elem]);
      cosine := cos(rotation[elem]);
   deflection[elem] :=  deflection[elem-1]
                      + rotation[elem]*elem_len;
   if ((elem <= defl.hinge_length) and (bottom_hinging))
      then begin
         moment[elem] := moment[0];
         curvature[elem] := curvature[0];
      end
      else if (elem = (defl.no_elem-defl.hinge_length)) and top_hinging
         then begin
            transition := true;
            trans_mom :=  moment[0]
                        - (lat*cosine+ax*sine)*elem*elem_len
                        - (ax*cosine-lat*sine)*def;
            moment[elem] := moment[defl.no_elem];
            curvature[elem] := curvature[defl.no_elem];
         end
         else if (elem > (defl.no_elem-defl.hinge_length)) and top_hinging
            then begin
               moment[elem] := moment[defl.no_elem];
               curvature[elem] := curvature[defl.no_elem];
            end
            else begin
               moment[elem] :=  moment[0]
                              - (lat*cosine+ax*sine)*elem_len*elem
                              - (ax*cosine-lat*sine)*deflection[elem];
               curvature[elem] := find_elem_curv(ax*cosine-lat*sine,moment[elem-1],
                                                  curvature[elem-1],
                                                  moment[elem],curv_inc);
            end;
   if (curvature[elem]=0) and (moment[elem]<>0)
      then redo := true;
   elem := elem + 1;
   if keyPressed
      then begin
         blank_line(24);
         wrcol(24,1,mess_color,'Interrupted - press ESC to abort, any other key to continue...');
         getc(ans);
         blank_line(24);
         if key_pressed = ESC
            then overflow := true;
      end;
end;
if transition and (abs(moment[defl.no_elem]-trans_mom) < defl.toler*abs(moment[defl.no_elem]))
```

```
  then redo := redo or false;
if abs(1-deflection[defl.no_elem]/def) < defl.toler
  then if redo
    then if top_hinging
      then begin
        old_curv := end_curv;
        old_defl := deflection[defl.no_elem];
        end_curv := end_curv * moment[defl.no_elem] / trans_mom;
      end
      else begin
        if end_curv <> old_curv
          then deriv := (deflection[defl.no_elem] - old_defl) / (end_curv - old_curv);
        if deriv = 0 then deriv := 0.00000001;
        old_curv := end_curv;
        old_defl := deflection[defl.no_elem];
        end_curv := end_curv - alpha * (deflection[defl.no_elem] - def) / deriv;
      end
    else if top_hinging and bottom_hinging
      then complete := true
      else begin
        bottom_hinging := bottom_hinging or (1.1*abs(curvature[1]) < abs(end_curv));
        top_hinging := bottom_hinging or (1.1*abs(curvature[defl.no_elem-1]) < abs(top_curv));
        if top_hinging then bottom_hinging := true;
        if not(top_hinging or bottom_hinging)
          then complete := true
          else redo := true;
        pass := 0;
        end_curv := def / sqr(prop.length/2);
        old_curv := 0;
        old_defl := 0;
      end
  else begin
    if end_curv <> old_curv
      then deriv := (deflection[defl.no_elem] - old_defl) / (end_curv - old_curv);
    old_curv := end_curv;
    old_defl := deflection[defl.no_elem];
    end_curv := end_curv - alpha * (deflection[defl.no_elem] - def) / deriv;
  end;
if (pass = 20) and (not(bottom_hinging and top_hinging))
  then begin
    pass := 0;
    bottom_hinging := true;
    top_hinging := true;
```

```
                      end_curv := def / sqr(prop.length/2);
                      old_curv := 0;
                      old_defl := 0;
                      redo := true;
                   end;
             end;
             until ((complete) or (pass>40)) and not(redo);
             converge := complete;
          end;
    end;

(*---------------------------------------------------------------------------*)

overlay procedure get_deflection(ax,lat : real; var def : real);
          var
             end_mom : real;
             elem_mom,elem_curv,elem_rot,elem_disp,elem_len : real;
             elem : integer;
             pass : integer;
             done : boolean;
             sine,cosine : real;
      begin
         overflow := false;
         if (lat = 0) or (ax < sect.data[2].axial)
            then def := 0
            else begin
               end_mom := lat * prop.length / 2;
               elem_len := prop.length / defl.no_elem;
               done := false;
               pass := 0;
               repeat
                  pass := pass + 1;
                  mess := concat(i_to_a(pass),'  ');
                  wrcol(21,72,data_color,mess);
                  elem_mom := end_mom;
                  elem_rot := 0;
                  elem_disp := 0;
                  elem := 1;
                  while (elem<defl.no_elem+1) and (not(done)) do begin
                     wrcol(22,75,data_color,i_to_a(elem));
                     get_elem_curv(ax,elem_mom,elem_curv);
                     if not(overflow)
                        then begin
```

```
                           elem_rot := elem_rot - elem_curv*elem_len;
                           sine := sin(elem_rot);
                           cosine := cos(elem_rot);
                           elem_disp := elem_disp + elem_rot*elem_len;
                           elem_mom :=  end_mom
                                      - (lat*cosine+ax*sine)*elem_len*elem
                                      + (ax*cosine-lat*sine)*elem_disp;
{ NOTE: alternative =>     elem_mom := end_mom - lat*elem_len + ax*elem_rot*elem_disp;  }
                           elem := elem + 1;
                        end
                        else done := true;
                     if keyPressed
                        then begin
                           blank_line(24);
                           wrcol(24,1,mess_color,'Interrupted - press ESC to abort, any other key to continue...');
                           getc(ans);
                           if key_pressed = ESC
                              then begin
                                 overflow := true;
                                 done := true;
                              end;
                           blank_line(24);
                        end;
                  end (* while *);
                  if abs(elem_mom/defl.ratio-end_mom) < abs(defl.toler*end_mom)
                     then done := true
                     else end_mom := (end_mom + elem_mom/defl.ratio) / 2;
               until (done) or (pass>30);
               if not(done)
                  then converge := false
                  else if (defl.no_load_steps>30) or (defl.no_elem>60)
                     then def := elem_disp
                     else with seg^.data[load_step] do begin
                        moment[0] := end_mom;
                        rotation[0] := 0;
                        deflection[0] := 0;
                        curvature[0] := 0;
                        sine := 0;
                        cosine := 1;
                        for elem := 1 to defl.no_elem do begin
                           get_elem_curv(ax*cosine-lat*sine,moment[elem-1],curvature[elem]);
                           rotation[elem] :=  rotation[elem-1]
                                                       - curvature[elem]*elem_len;
                           sine := sin(rotation[elem]);
                           cosine := cos(rotation[elem]);
                           deflection[elem] :=  deflection[elem-1]
                                                        + rotation[elem]*elem_len;
                           moment[elem] :=  (lat*cosine+ax*sine)*elem_len*elem
                                          + moment[0]
                                          + (ax*cosine-lat*sine)*deflection[elem];
                        end (* for *);
                        def := deflection[defl.no_elem];
                     end (* else with seg^ *);
            end (* else *);
      end;

(*------------------------------------------------------------------------------*)

overlay procedure calc_section;
          var
             new_num : real;
             name : string[20];
             finished : boolean;
             aborted : boolean;
             ax_inc : real;
             mom : real;
             i,j : integer;
             no_ax,no_pts : integer;
          begin
             if get_props then begin
                prop_menu;
                cls(1,62,23,80,0);
                draw_box(0,0,x_abs_window-1,y_abs_window-1);
                wrcol(2,62,descr_color,'Section file :');
                wrcol(3,64,data_color,sect.filename);
                write_prop;
                blank_line(24);
                wrcol(24,1,mess_color,'Input number of axial loads to use in calculation : ');
                getnum(new_num,2,0);
                no_ax := round(new_num);
                sect.no_ax_loads := no_ax;
                wrcol(5,62,descr_color,'Number of');
                mess := concat('axial loads : ',i_to_a(no_ax));
                wrcol(6,63,data_color,mess);
                blank_line(24);
                wrcol(24,1,mess_color,'Input number of section slices to use : ');
```

```
                            then begin
                               blank_line(24);
                               wrcol(24,1,mess_color,'Press ESC to abort, any other key to continue...');
                               getc(ans);
                               if key_pressed = ESC
                                  then aborted := true;
                            end;
                         j := j + 1;
                      end (* for j *);
                      i := i + 1;
                   end (* with sect.data[i] / for i *);
                for j := 1 to sect.data[1].no_points do begin
                   sect.data[1].moment[j] := 0.0;
                   sect.data[1].curvature[j] := 2 * sect.data[2].curvature[j];
                   sect.data[no_ax].moment[j] := 0.0;
                   sect.data[no_ax].curvature[j] := sect.data[no_ax-1].curvature[j];
                end;
                smoother;
                if not(aborted) then begin
                   rewrite(sect_file);
                   write(sect_file,sect);
                   close(sect_file);
                   blank_line(24);
                   blank_line(25);
                   wrcol(25,1,mess_color,'Analysis complete. Press any key to return to main menu...');
                   repeat until keyPressed;
                end;
             end (* if flag *);
             cls(1,1,25,80,0);
          end;

(*----------------------------------------------------------------------------*)

overlay function get_info : boolean;
          var
             name : string[20];
             temp : real;
             ext_pos : integer;
          begin
               cls(1,1,25,80,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
               blank_line(24);
               wrcol(24,1,mess_color,'How many elements do you want to break the column into? ');
```

```
getnum(temp,3,0);
defl.no_elem := round(temp);
wrcol(2,62,descr_color,'Number of');
wrcol(3,62,descr_color,' elements : ');
wrcol(3,74,data_color,i_to_a(defl.no_elem));
blank_line(24);
wrcol(24,1,mess_color,'How many load steps do you want? ');
getnum(temp,2,0);
defl.no_load_steps := round(temp);
wrcol(5,62,descr_color,'Number of');
wrcol(6,62,descr_color,' load steps : ');
wrcol(6,76,data_color,i_to_a(defl.no_load_steps));
blank_line(24);
wrcol(24,1,mess_color,'Input ratio of top to bottom moment : ');
getnum(temp,1,3);
defl.ratio := temp;
wrcol(8,62,descr_color,'Mom ratio =');
wrcol(8,74,data_color,f_to_a(defl.ratio,1,3));
blank_line(24);
wrcol(24,1,mess_color,'What level of tolerance is acceptable (in percent)? ');
getnum(temp,2,3);
wrcol(10,62,descr_color,'Tolerance : ');
wrcol(11,64,data_color,f_to_a(temp,1,3));
wrcol(11,69,descr_color,' percent');
defl.toler := temp / 100.0;
load_prop_file;
wrcol(13,62,descr_color,'prop file :');
wrcol(14,64,data_color,prop.filename);
defl.prop_filename := prop.filename;
load_load_file;
switch_load_type;
wrcol(15,62,descr_color,'load file :');
wrcol(16,64,data_color,load.filename);
defl.load_filename := load.filename;
defl.load_control := load.load_control;
blank_line(24);
wrcol(24,1,mess_color,'How should section properties be calculated: (A)uto or (M)anual? ');
repeat
   getc(ans);
   case upcase(chr(ans)) of
      'A' : begin
               ext_pos := pos('prp',prop.filename)-1;
               sect.filename := copy(prop.filename,1,ext_pos);
```

```
                              sect.filename := concat(sect.filename,'sec');
                              assign(sect_file,sect.filename);
                              {$I-}
                              reset(sect_file);
                              {$I+}
                              if IOresult = 0
                                 then begin
                                    read(sect_file,sect);
                                    close(sect_file);
                                    get_info := true;
                                 end
                                 else get_info := false;
                              done := true;
                              defl.auto_mode := true;
                           end;
                     'M' : begin
                              load_section_file;
                              defl.auto_mode := false;
                              done := true;
                           end
                     else done := false;
                  end (* case *);
               until done;
               wrcol(17,62,descr_color,'section file :');
               wrcol(18,64,data_color,sect.filename);
               defl.sect_filename := sect.filename;
               blank_line(24);
               blank_line(25);
          end;

(*---------------------------------------------------------------------------*)

procedure dummy4;
          begin
             writeln('This procedure is never called');
             writeln('If you see this while the program is running,');
             writeln('all known principles of physics are invalid...');
          end;

(*---------------------------------------------------------------------------*)
(*---------------------------------------------------------------------------*)

overlay procedure properties;
```

```
var
   new_num : real;
   arrow_col : integer;
   max_box : integer;
begin
     prop_menu;
     write_prop;
     repeat
        if prop.width=prop.depth
           then max_box := 17
           else max_box := 18;
        if curr_box>9
           then arrow_col:=34
           else arrow_col:=2;
        draw_arrow(prop_row[curr_box],arrow_col);
        getc(menu_char);
        case key_pressed of
            F2          :    begin
                                  load_prop_file;
                                  write_prop;
                             end;
            F3          :    begin
                                  save_prop_file;
                                  write_prop;
                             end;
            F4          :    begin
                                prop_init;
                                write_prop;
                             end;
            F5          :    begin
                                inter_to_csv;
                                write_prop;
                             end;
            PG_DN       :    begin
                                blank_line(25);
                                wrcol(25,1,mess_color,'...calculating, please wait...');
                                calc_prop;
                                show_calc_prop;
                                cls(1,1,25,80,0);
                                prop_menu;
                                write_prop;
                             end;
            UP_ARROW    :    begin
```

```
                      if curr_box=1
                         then curr_box := max_box
                         else curr_box := curr_box - 1;
                      if ((curr_box=15) or (curr_box=16)) and (prop.layers<>3)
                         then curr_box := 14;
                   end;
DN_ARROW   :       begin
                      if curr_box=max_box
                         then curr_box := 1
                         else curr_box := curr_box + 1;
                      if ((curr_box=15) or (curr_box=16)) and (prop.layers<>3)
                         then curr_box := 17;
                   end;
LT_ARROW,
RT_ARROW   :       begin
                      if curr_box>9
                         then curr_box := curr_box - 9
                         else curr_box := curr_box + 9;
                      if ((curr_box=15) or (curr_box=16)) and (prop.layers<>3)
                         then curr_box := 14;
                   end;
CARRIAGE    :      begin
                        blank_line(24);
                        wrcol(24,1,mess_color,'Input new value : ');
                        getnum(new_num,10,10);
                        prop.inter_flag := false;
                        with prop do
                           case curr_box of
                             1  : length := new_num;
                             2  : width := new_num;
                             3  : depth := new_num;
                             4  : conc_str := new_num;
                             5  : ult_conc_strain := new_num;
                             6  : steel_yield := new_num;
                             7  : steel_ult := new_num;
                             8  : tie_bar_area := new_num;
                             9  : tie_spacing := new_num;
                             10 : begin
                                         layers := round(new_num);
                                         cls(1,1,25,80,0);
                                         prop_menu;
                                  end;
                             11 : steel[1].area := new_num;
```

```
                                                12 : steel[1].depth := new_num;
                                                13 : steel[2].area := new_num;
                                                14 : steel[2].depth := new_num;
                                                15 : steel[3].area := new_num;
                                                16 : steel[3].depth := new_num;
                                                17 : tie1_no_legs := round(new_num);
                                                18 : tie2_no_legs := round(new_num);
                                              end (* case curr_box *);
                                           blank_line(24);
                                           write_prop;
                                      end (* CARRIAGE *);
                      else begin end;
                  end (* case key_pressed *);
               until key_pressed = ESC;
               cls(1,1,25,80,0);
          end (* properties *);

(*---------------------------------------------------------------------------*)

overlay procedure loading;
          var
             arrow_col : integer;
             new_num : real;
          begin
             load_menu;
             write_load;
             with load do
               repeat
                  if key_pressed=CARRIAGE then write_load;
                  if (load_row[curr_box]>10)
                     then arrow_col := load_col[curr_box] - 1
                     else if load_row[curr_box]=6
                        then arrow_col := load_col[curr_box] - 16
                        else if load_col[curr_box]<30
                           then arrow_col := 2
                           else arrow_col := 32;
                  draw_arrow(load_row[curr_box],arrow_col);
                  getc(menu_char);
                  case key_pressed of
                     F2          : begin
                                      load_load_file;
                                      switch_load_type;
                                      cls(1,1,23,61,0);
```

```
                 load_menu;
                 write_load;
              end;
F3          : begin
                 save_load_file;
                 write_load;
              end;
F4          : begin
                 load_init;
                 write_load;
              end;
F5          : begin
                 graph_no := 2;
                 show_graph;
                 cls(1,1,23,61,0);
                 load_menu;
                 write_load;
              end;
F7          : begin
                 if load_control
                    then load_control := false
                    else load_control := true;
                 cls(1,1,25,80,0);
                 load_menu;
                 write_load;
              end;
UP_ARROW    : if curr_box=1
                 then curr_box := lat_boxes + ax_boxes
                 else curr_box := curr_box - 1;
DN_ARROW    : if curr_box=(lat_boxes+ax_boxes)
                 then curr_box := 1
                 else curr_box := curr_box + 1;
RT_ARROW    : if curr_box>lat_boxes
                 then curr_box := 1
                 else curr_box := lat_boxes + 1;
LT_ARROW    : if curr_box>lat_boxes
                 then curr_box := 1
                 else curr_box := lat_boxes + 1;
CARRIAGE    : if curr_box = 1
                 then begin
                    wrcol(19,62,help_color,'LOADING TYPES :');
                    wrcol(20,62,help_color,'1 - constant');
                    wrcol(21,62,help_color,'2 - linear');
```

```
      wrcol(22,62,help_color,'3 - multi-linear');
      blank_line(24);
      wrcol(24,1,mess_color,'Input number for load type : ');
      getnum(new_num,1,0);
      blank_line(24);
      if (new_num<1) or (new_num>3) then new_num := lateral_type;
      lateral_type := round(new_num);
      switch_load_type;
      cls(1,1,25,80,0);
      load_menu;
      write_load;
   end
   else if curr_box = (lat_boxes+1)
      then begin
         wrcol(19,62,help_color,'LOADING TYPES :');
         wrcol(20,62,help_color,'1 - constant');
         wrcol(21,62,help_color,'2 - linear');
         wrcol(22,62,help_color,'3 - multi-linear');
         blank_line(24);
         wrcol(24,1,mess_color,'Input number for load type : ');
         getnum(new_num,1,0);
         blank_line(24);
         if (new_num<1) or (new_num>3) then new_num := lateral_type;
         axial_type := round(new_num);
         switch_load_type;
         cls(1,1,25,80,0);
         load_menu;
         write_load;
      end
      else begin
         blank_line(24);
         wrcol(24,1,mess_color,'Input new value : ');
         getnum(new_num,8,8);
         blank_line(24);
         if curr_box=2 then lateral[1] := new_num;
         case lateral_type of
            1 : axial[curr_box-3] := new_num;
            2 : if curr_box = 3
                   then lateral[2] := new_num
                   else axial[curr_box-4] := new_num;
            3 : if curr_box < 12
                   then lateral[curr_box-1] := new_num
                   else axial[curr_box-12] := new_num;
```

```
                                           end;
                                           cls(load_row[curr_box],load_col[curr_box],
                                               load_row[curr_box],load_col[curr_box]+7,0);
                                           if load_control
                                              then mess := f_to_a(new_num,4,1)
                                              else mess := f_to_a(new_num,1,4);
                                           wrcol(load_row[curr_box],load_col[curr_box],data_color,mess);
                                        end;
                     else begin end;
                  end (* case key_pressed *);
               until key_pressed = ESC;
               cls(1,1,25,80,0);
          end (* loading *);

(*--------------------------------------------------------------------------*)

overlay procedure section;
          var
             arrow_col : integer;
             new_num : real;
             indx : integer;
             ready : boolean;
          begin
               sect_menu;
               write_sect;
               repeat
                  if curr_box=1
                     then arrow_col:=18
                     else if curr_box=3
                        then arrow_col := 40
                        else if curr_box=4
                           then arrow_col := 16
                           else arrow_col:=sect_col[curr_box]-1;
                  draw_arrow(sect_row[curr_box],arrow_col);
                  getc(char_code);
                  case key_pressed of
                     F2 : begin
                             load_section_file;
                             write_sect;
                          end;
                     F3 : save_section_file;
                     F4 : smoother;
                     F5 : sect_init;
```

```
F6 : begin
        graph_no := 4;
        show_graph;
        sect_menu;
        write_sect;
     end;
F7 : print_sect;
F8 : sect_to_sc3;
F9 : begin
        ready := false;
        repeat
           blank_line(25);
           wrcol(25,1,help_color,'1 = constant axial, 2 = constant curvature, 3 = constant moment');
           blank_line(24);
           wrcol(24,1,mess_color,'Input selection : ');
           getnum(new_num,1,0);
           indx := round(new_num);
           if (indx > 0) and (indx < 4) then ready := true;
        until ready;
        blank_line(24);
        blank_line(25);
        case indx of
           1 : begin
                  graph_no := 5;
                  ready := false;
                  repeat
                     wrcol(24,1,mess_color,'Input axial load no. :  ');
                     getnum(new_num,2,0);
                     if (new_num > 1) and (new_num < sect.no_ax_loads)
                        then begin
                           ready := true;
                           curr_ax_load := round(new_num);
                        end;
                  until ready;
                  blank_line(24);
                  cls(1,62,25,80,0);
                  show_graph;
                  wrcol(1,26,descr_color,' AXIAL LOAD = ');
                  wrcol(1,40,data_color,f_to_a(sect.data[curr_ax_load].axial,6,2));
               end (* case 1 *);
           2 : begin
                  graph_no := 6;
                  wrcol(24,1,mess_color,'Input curvature :  ');
                  getnum(plot_curv,10,10);
                  blank_line(24);
                  cls(1,62,25,80,0);
                  show_graph;
                  wrcol(1,27,descr_color,' CURVATURE = ');
                  wrcol(1,40,data_color,f_to_a(plot_curv,1,7));
               end (* case 2 *);
           3 : begin
                  graph_no := 7;
                  wrcol(24,1,mess_color,'Input moment :  ');
                  getnum(plot_mom,10,10);
                  cls(1,62,25,80,0);
                  blank_line(24);
                  show_graph;
                  wrcol(1,30,descr_color,' MOMENT = ');
                  wrcol(1,40,data_color,f_to_a(plot_mom,7,1));
               end (* case 3 *);
        end (* case *);
        wrcol(25,1,mess_color,'Press any key to restore data...');
        repeat until keyPressed;
        sect_menu;
        write_sect;
     end;
UP_ARROW : if curr_box=1
              then curr_box := 44
              else if curr_box=3
                 then curr_box := 1
                 else curr_box := curr_box - 1;
DN_ARROW : if curr_box=44
              then curr_box := 1
              else if curr_box=1
                 then curr_box := 3
                 else curr_box := curr_box + 1;
LT_ARROW : if curr_box>4 then
              if curr_box>14
                 then curr_box := curr_box - 10
                 else curr_box := curr_box + 30;
RT_ARROW : if curr_box>4 then
              if curr_box<34
                 then curr_box := curr_box + 10
                 else curr_box := curr_box - 30;
PG_DN    : begin
              if curr_ax_load=1
```

```
                                      then curr_ax_load := sect.no_ax_loads
                                      else curr_ax_load := curr_ax_load - 1;
                                   write_sect;
                                end;
                     PG_UP    : begin
                                   if curr_ax_load=sect.no_ax_loads
                                      then curr_ax_load := 1
                                      else curr_ax_load := curr_ax_load + 1;
                                   write_sect;
                                end;
                     CARRIAGE : with sect.data[curr_ax_load] do begin
                                   blank_line(24);
                                   wrcol(24,1,mess_color,'Input new value : ');
                                   getnum(new_num,10,10);
                                   blank_line(24);
                                   if curr_box=1
                                      then axial := new_num
                                      else if curr_box=3
                                         then sect.no_ax_loads := round(new_num)
                                         else if curr_box=4
                                            then no_points := round(new_num)
                                            else if curr_box < 25
                                               then begin
                                                  moment[curr_box-4] := -new_num;
                                                  moment[no_points-curr_box+5] := new_num;
                                               end
                                               else begin
                                                  curvature[curr_box-24] := -new_num;
                                                  curvature[no_points-curr_box+25] := new_num;
                                               end;
                                   write_sect;
                                end (* CARRIAGE *);
                     else begin end;
                  end (* case *);
               until key_pressed = ESC;
               cls(1,1,25,80,0);
          end (* section *);

(*----------------------------------------------------------------------------*)

overlay procedure analyze;
          var
             stop : boolean;
```

```
   error : boolean;
begin
    new(seg);
    converge := true;
    overflow := false;
    error := false;
    if not(get_info)
       then calc_section;
    wrcol(24,1,mess_color,'Do you want a printout of the results (y/n)?');
    getc(ans);
    blank_line(24);
    printout := (chr(ans) = 'y');
    wrcol(24,1,mess_color,'...analysis underway, please wait...');
    wrcol(20,62,descr_color,'Load step :');
    wrcol(21,62,descr_color,'Pass no :');
    wrcol(22,62,descr_color,'Element no :');
    get_load_steps;
    load_step := 0;
    stop := false;
    while (load_step<no_load_steps+1) and not(stop) do begin
       mess := concat(i_to_a(load_step),' ');
       wrcol(20,74,data_color,mess);
       if load.load_control
          then get_deflection(defl.ax_load[load_step],defl.lat_load[load_step],defl.deflection[load_step])
          else get_lat_load(defl.deflection[load_step],load.axial[1],load.axial[2],
                            defl.ax_load[load_step],defl.lat_load[load_step]);
       error := error or overflow;
       overflow := false;
       if not(converge)
          then wrcol(25,1,error_color,'Convergence problem detected -- check data for equilibrium');
       if printout then begin
          writeln(lst);
          writeln(lst,'Load step number ',load_step);
          writeln(lst,'Axial load = ',defl.ax_load[load_step]);
          writeln(lst,'Lateral load = ',defl.lat_load[load_step]);
          writeln(lst,'Deflection = ',defl.deflection[load_step]);
          writeln(lst);
       end;
       if keyPressed
          then begin
             blank_line(24);
             wrcol(24,1,mess_color,'Press ESC to abort, any other key to continue...');
             getc(ans);
```

```
                  if key_pressed = ESC
                     then stop := true;
               end;
            load_step := load_step + 1;
         end (* while *);
         if error
            then begin
               blank_line(25);
               blank_line(24);
               wrcol(24,1,error_color,'The column has failed to converge on a solution');
               wrcol(25,1,mess_color,'Press ESC to abort results, any other key to continue...');
               getc(ans);
            end
            else begin
               blank_line(25);
               wrcol(25,1,mess_color,'Analysis complete.');
               key_pressed := 0;
            end;
         blank_line(25);
         blank_line(24);
         if key_pressed <> ESC
            then begin
               save_defl_file;
               cls(1,1,23,61,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
               graph_no := 1;
               show_graph;
               blank_line(24);
               blank_line(25);
               wrcol(25,1,mess_color,'Press any key to return to main menu...');
               repeat until keyPressed;
               blank_line(25);
            end;
         dispose(seg);
      end;

(*-------------------------------------------------------------------------*)

overlay procedure toggle_menu;
      begin
         if x_abs_window = 640 then x_abs_window := 480
            else x_abs_window := 640;
         if y_abs_window = 350 then y_abs_window := 320
```

```
                  else y_abs_window := 350;
               cls(1,1,25,80,0);
               show_graph;
               key_pressed := 0;
            end;

(*----------------------------------------------------------------------------*)

overlay procedure main_menu;
            begin
               cls(1,62,23,80,0);
               draw_box(0,0,x_abs_window-1,y_abs_window-1);
               wrcol(2,62,menu_color,'F1  : help');
               wrcol(3,62,menu_color,'F2  : properties');
               wrcol(4,62,menu_color,'F3  : loading');
               wrcol(5,62,menu_color,'F4  : view or edit');
               wrcol(6,62,menu_color,'      section');
               wrcol(7,62,menu_color,'      response');
               wrcol(8,62,menu_color,'F5  : calculate');
               wrcol(9,62,menu_color,'      section');
               wrcol(10,62,menu_color,'      response');
               wrcol(11,62,menu_color,'F6  : analyze');
               wrcol(12,62,menu_color,'F7  : turn off menu');
               wrcol(13,62,menu_color,'F9  : show graph');
               wrcol(14,62,menu_color,'F10 : clear graph');
               wrcol(15,62,menu_color,'ESC : exit to DOS');
            end;

(*----------------------------------------------------------------------------*)

procedure run_column;
            begin
                 repeat
                    if key_pressed = 0
                       then if x_abs_window = 480
                          then main_menu
                          else wrcol(25,62,white,'Hit F7 for menu');
                    getc(menu_char);
                    case key_pressed of
                       F2  : begin
                                curr_box := 1;
                                old_ar_row := 24;
                                old_ar_col := 79;
```

```
          properties;
          key_pressed := 0;
       end;
F3  : begin
          curr_box := 1;
          old_ar_row := 24;
          old_ar_col := 79;
          loading;
          key_pressed := 0;
       end;
F4  : begin
          curr_box := 1;
          old_ar_row := 24;
          old_ar_col := 79;
          section;
          key_pressed := 0;
       end;
F5  : begin
          curr_box := 1;
          old_ar_row := 24;
          old_ar_col := 79;
          calc_section;
          key_pressed := 0;
       end;
F6  : begin
          analyze;
          cls(1,62,23,80,0);
          key_pressed := 0;
       end;
F7  : toggle_menu;
F9  : begin
          load_defl_file;
          graph_no := 1;
          show_graph;
          blank_line(25);
          wrcol(25,1,mess_color,'Graphing complete; press any key to restore main menu...');
          repeat until keyPressed;
          blank_line(25);
          cls(1,62,23,80,0);
          key_pressed := 0;
       end;
F10 : begin
          cls(1,1,23,61,0);
```

```
                              draw_box(0,0,x_abs_window-1,y_abs_window-1);
                           end;
                     ESC : begin
                              blank_line(25);
                              wrcol(25,1,error_color,'ARE YOU SURE YOU WANT TO STOP (y/n)? ');
                              getc(ans);
                              if chr(ans) <> 'y'
                                 then blank_line(25)
                                 else key_pressed := ESC;
                           end;
                     else begin end;
                  end (* case *)
               until key_pressed = ESC;
          end;

(*---------------------------------------------------------------------------*)

begin
   initialize;
   run_column;
   TextMode;
end.
```