IMPLICATIONS OF P-DELTA ANALYSIS AND LRFD OF GABLE FRAMES

by

Eric J. Wishart

Thesis submitted to the faculty of

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

CIVIL ENGINEERING

APPROVED:

Thomas M. Murray, Chairman

Richard M. Barker

Siegfried M. Holzer

W. Samuel Easterling

December 1990

Blacksburg, Virginia 24061

IMPLICATIONS OF P-DELTA ANALYSIS AND LRFD OF GABLE FRAMES

by

Eric J. Wishart

Chairman: Thomas M. Murray

Charles E. Via Department of Civil Engineering

(ABSTRACT)

Recent developments in the philosophy of structural steel design have led to design specifications that incorporate second-order geometric effects. The use of second-order elastic analysis (SOEA) in the design of structural frameworks may lead to more economically designed structures and increased knowledge of structural stability.

The research presented here concerns economy of design between the available steel design specifications as they apply to the metal building industry. Since these buildings are primarily for industrial use, their optimization suggests the use of gabled rigid frames with tapered elements to provide the required load carrying capacity.

Results of the research indicate that elastic stability considering geometric nonlinearity is not a primary concern for these types of frames. Rather, the fully-stressed design approach leads to the optimally designed frame.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

ACKNOWLEDGEMENTS

The writer wishes to express his sincere gratitude to Dr. Thomas M. Murray, without whom this research would not have been possible. He would also like to thank his academic advisory committee members, W. Samuel Easterling, Siegfried M. Holzer, Richard M. Barker, and especially Thomas M. Murray for their direction and guidance during the course of this research. He would also like to thank his parents James & Teresa, and his fiancee Grace, for their constant support and patience. Special thanks are also due to Craig S. Young, for his programming assistance. A hearty thanks also to Ann W. Crate, who helped with the manuscript. Last, but by no means least, the writer wishes to thank NUCOR Metal Building Systems for funding the research project.

Dedicated to the late James L. Baldwin...

LIST OF FIGURES

LIST OF FIGURES (continued)

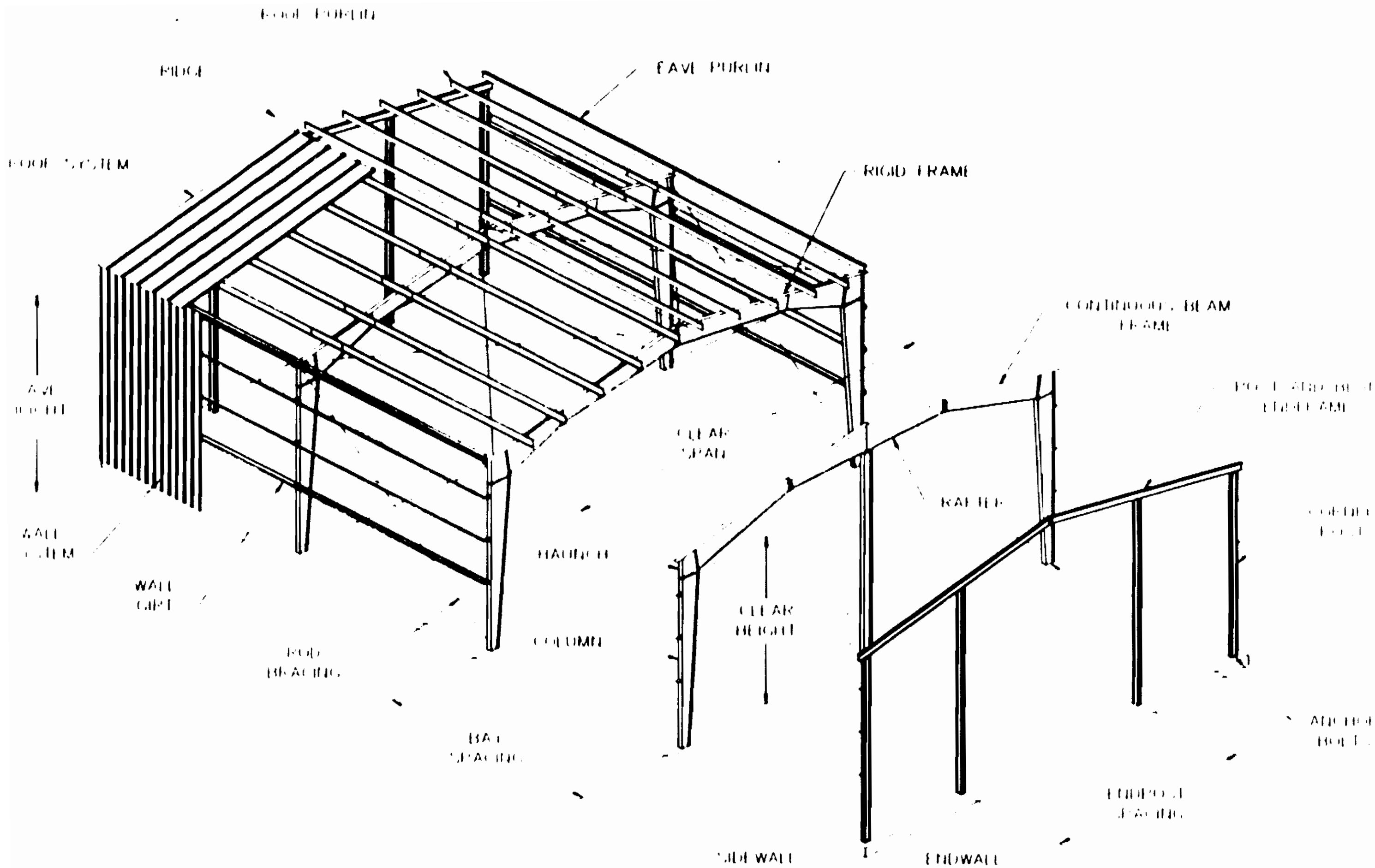
LIST OF TABLES

CHAPTER I
INTRODUCTION

1.1 Background

Tapered and Gabled Rigid Framing Systems are manufactured by the metal building Industry primarily for light, industrial buildings. The purpose of tapering is to optimize the frame cross-sections to develop a fully-stressed design. Such a frame is shown in Figure 1.1. The gable is introduced in order to slope the roof, allowing free drainage, and to provide vertical clearance where necessary. Light industrial buildings are those buildings for which the design Live-to-Dead Load ratio (L/D) is greater than five, which is typical with these types of building systems. The manufacturer of the frames studied here are those of NUCOR Metal Building Systems, hereinafter referred to as NUCOR. These systems may be designed in the United States under the provisions of the AISC Allowable Stress Design (ASD) Specifications (AISC, 1978 and 1989), the AISC Load and Resistance Factor Design (LRFD) Specification (AISC, 1986), the Uniform Building Code (UBC) (ICBO, 1988), and the Low Rise Building Systems Manual (LRBSM) (MBMA, 1986), along with local requirements where applicable.

These structures are typically designed with unbraced frames in the in-plane direction, and braced frames in the out-of-plane direction. The framing of concern in this study are the repetitive primary interior unbraced frames, typically spaced at 25 ft. intervals. Unbraced frames are those types of frames that



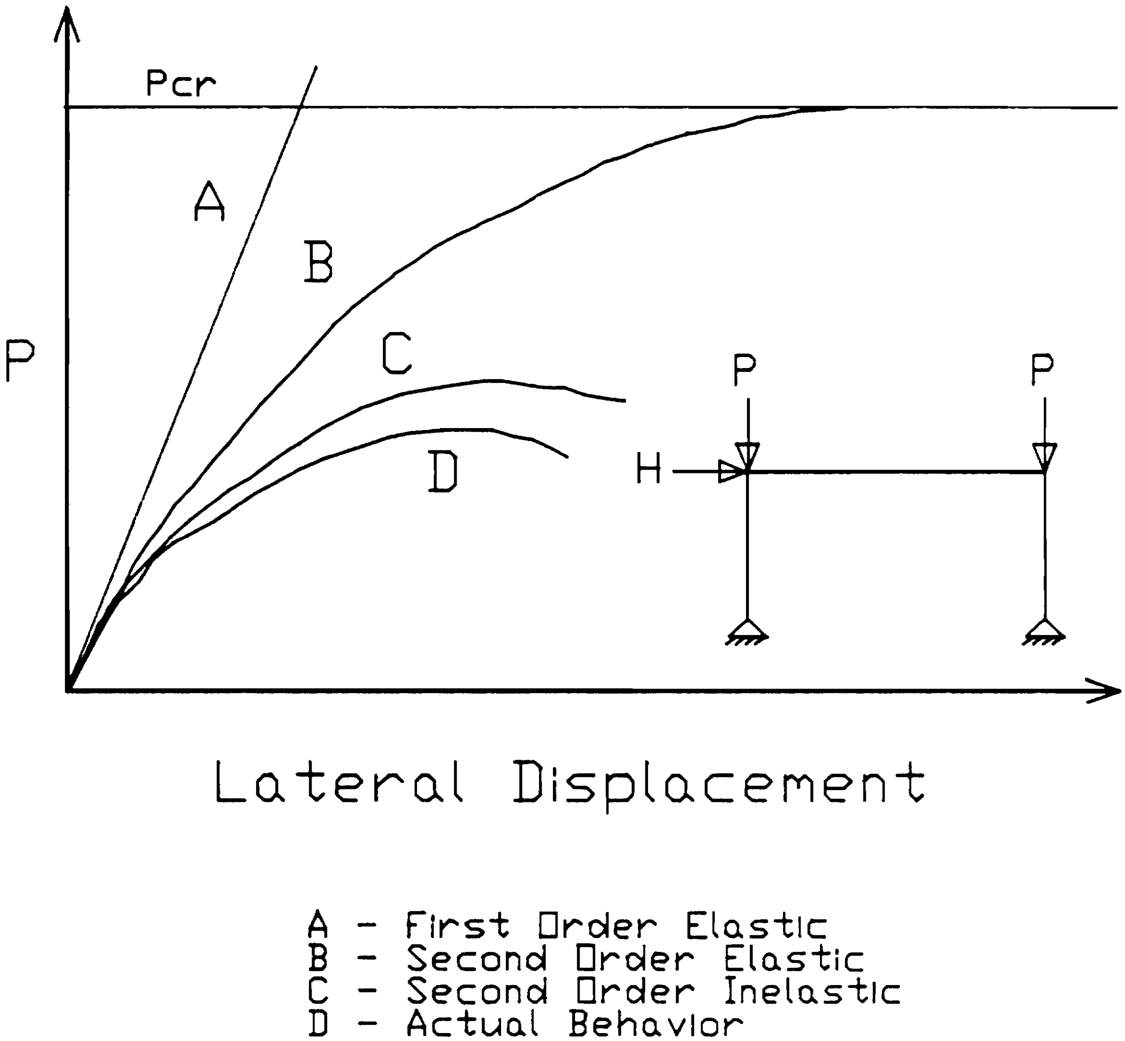
Typical Gable Frame
Figure 1.1

depend on their own flexural stiffness for lateral stability. All of the frames involved in this study are unbraced.

Local effects due to concentrated loads, end frames, post-buckling shear strength, and out-of-plane behavior are not considered in this study. Rather, the repetitive interior frames of these industrial structures are studied, since by weight, they comprise the largest portion of the load-resisting system.

Because a linear, first-order elastic analysis of a structure under compression loads does not account for reductions in flexural stiffness, flexural stiffness is overestimated and therefore the lateral stability (Figure 1.2). The reduction in flexural stiffness or stability is investigated in ASD via a stability interaction equation. Since the ASD approach in steel design focuses on individual member behavior rather than a system approach, this investigation is carried out by the use of an amplification factor on the individual frame components. The procedure involves the use of an effective length factor for column strength (relating the buckling strength of the column to an equivalent pinned-end Euler column). For unbraced, prismatic frames the in-plane effective length factor is always equal to or greater than 1.0. This equation attempts to estimate the degree of second-order effects in the design moments by an amplification factor based on the amount of compressive axial load in the member being designed. Similarly, investigation of structural capacity of the individual components involves the use of a yielding interaction equation, summing stresses on the cross-section.

By the LRFD Specification, the interaction equations are based on combined forces vs. combined strengths, where the design forces in the members are at their factored values. The LRFD interaction equations for



Analysis Response Curves

Figure 1.2

strength are stability-based, since the axial term of the equation is based on the nominal compressive strength and not the nominal yield strength. Stresses are not calculated in LRFD, since plastification is recognized for flexural design. An elastic stress is meaningless if the section strength is above the theoretical yield moment.

1.2 Scope and Purpose of Research

This study served many functions. First, it investigated the possible use of an in-plane effective length factor of 1.0 in the ASD stability interaction equation and the LRFD strength interaction equation. This is accomplished by performing true second-order analyses under the controlling load combinations and checking the interaction equations. Second, since the ASD procedure that the fabricator uses was modified recently (AISC, 1989) an attempt to estimate its implications to the metal building industry is made. Third, investigation of column and rafter stability was performed in order to determine the effective lengths of tapered elements, and compared with those available in the tapered column alignment charts. Fourth, because of the added difficulties in following the slender cross-section design provisions for column, flexural, and interaction behavior, examples are provided in Appendix D. Fifth, it was found that critical load combinations had not been investigated for multi-span frames. Sixth, a convergence study was performed in order to quantify the analysis errors introduced by using prismatic elements to model tapered framing.

The study focuses on the economy of design for two second-order elastic formats, using a modified ASD Specification and the LRFD Specification. These studies concern behavior and design proposals for the interaction of axial compression and strong-axis flexure of beam-columns. It is hoped that the use of an (in-plane) effective length factor of 1.0 for the interaction behavior can result in increased capacities for the range of frames typically designed by NUCOR. There are 16 recommendations regarding both the analysis and capacity sides of the design equations and further study suggestions.

The initial proposal for this research involved only an investigation of frame design strengths, comparing first and second-order analysis and design formats. During the course of the investigation, a significant number of problems appeared that warranted further investigation, including but not limited to, flexural design of slender-web members and compression design of cross-sections with slender elements. These problems are detailed, verified, and discussed as an integral part of this thesis. Also, implications of a flange/web local buckling criterion in the 1989 ASD Specification are analytically discussed.

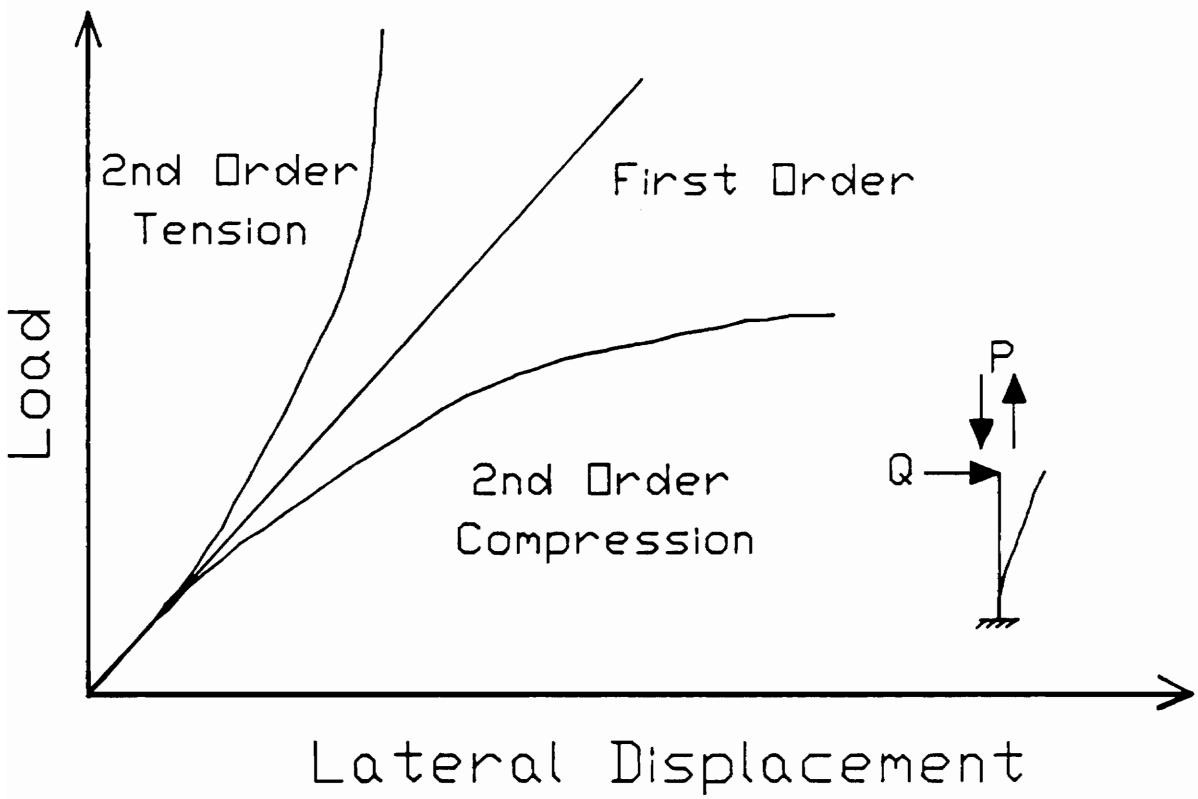
1.3 Assumptions

1.3.1 General Assumptions

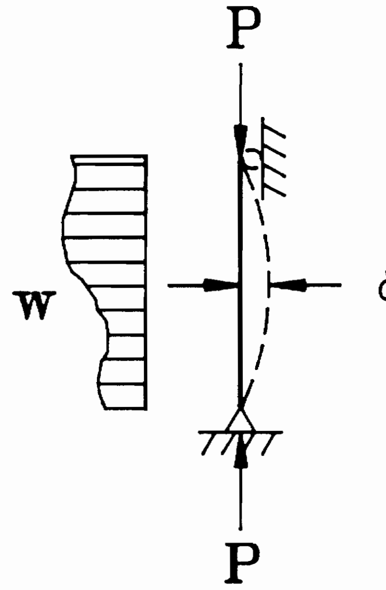
- The 10 symmetric gable frames submitted by NUCOR were designed according to the provisions of the 1978 ASD Specification, 1988 Uniform Building Code, and 1986 Low Rise Building Systems Manual.
- Basic design wind load pressures and their distributions are taken from the LRBSM, since the UBC provisions are currently being modified at this time by the ICBO (ICBO, 1990).

1.3.2 Analysis Assumptions

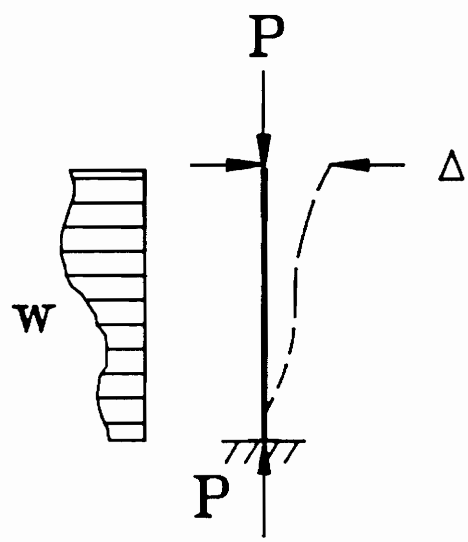
- Material is elastic, and remains elastic throughout the loading history.
- All members are rigidly connected at the joints (ASD: Type I, LRFD Type FR).
- Member lengths are defined by lines passing through their centers-of-gravity.
- Only strong-axis bending is considered, i.e., no biaxial bending is introduced from the out-of-plane direction.
- Their is no initial crookedness or out-of-plumbness in the columns or rafters.
- Maximum moments occur at girt or purlin locations, or at element connectivity points.
- Bernoulli-Euler beam theory (small strains).
- Static loading governs behavior.
- Second-Order Elastic Analysis (SOEA) includes both the member (P-δ) and frame (P-Δ) effects, under axial compression or tension (Figures 1.3 and 1.4).
- Basic loads are taken from appropriate design codes and there are no special loads or requirements of local codes.
- Frames are unbraced in the in-plane direction.
- Frames are braced in the out-of-plane direction.
- Individual elements of the frame resist the calculated first and second-order design moments without local buckling taking place (attainment of the design section strength is required in order for the design moments to be correct, since unacceptable local buckling will have the effect of load redistribution not accounted for in the analysis.



Second Order Effects
Figure 1.3



P−δ Effect



P−Δ Effect

Geometry of Deformation

Figure 1.4

- Nonlinear member curvature (bowing effect) is neglected in the analyses, i.e., the change in member length does not vary with the difference between the flexurally bent member and the unloaded member (curvature shortening).

1.3.3 Capacity Assumptions

- First-order ASD effective-length factors are taken as 1.5 (In-Plane) and 1.0 (Out-of-Plane) for columns, and 1.95 (In-Plane) and 1.0 (Out-of-Plane) for rafters.
- Second-order ASD and LRFD effective-length factors are taken as 1.0 for interaction behavior.
- The 1989 ASD Specification is followed. The 1989 ASD is chosen as future NUCOR frame designs will be required to conform to this Specification.
- The 1986 LRFD Specification is followed.
- Shear deformation is negligible in the displacement analysis of the section.
- Inelastic action and the leaning column principle is neglected in the determination of capacities.

CHAPTER II
LITERATURE REVIEW

2.1 Overview

Since first-order elastic analysis overestimates lateral stiffness under the action of compressive loads, it is required to assess the magnitude of this reduced lateral stiffness. Further, a first-order elastic analysis and a design based on a lateral drift limit does not assure adequate lateral stability, since first-order analysis completely neglects any axial deformation and decreased bending stiffness effects. By the first-order allowable stress design procedure, second-order effects are assessed approximately by an amplification factor method. This method may be unconservative, especially for unbraced frames (Salmon and Johnson 1989).

Second-order elastic effects are those effects induced by axial loads acting through the laterally displaced configuration of a beam-column. Second-order elastic analysis considers the geometric changes in the structural assembly, i.e., how the displacements and rotations affect the structure as a system. The major disadvantage to modeling these geometric changes is the loss of superposition of the results.

Considering these effects results in amplification of first-order analysis moments, since the geometric changes create further lateral displacements which, in turn, amplify the moments, and so on. This second-order elastic amplification has been assessed by a number of recent researchers, and their methods have been recognized by the industry as being accurate and appropriate. Because of the second-order effect, equilibrium must be formulated on the displaced structural configuration, which is not known in advance, and continues to change under the application of the loads.

The actual response depicted in Figure 1.2 can be most closely followed by so-called second-order inelastic analysis, which includes not only nonlinear geometrical effects, but also material nonlinearity such as yielding and plastification. This type of analysis is being studied at Cornell University and Purdue University, although there is no practical method to date of performing such an analysis, nor a design specification that recognizes the analysis results. Its purpose is to completely eliminate the effective-length procedure of column design.

To assess the magnitude of second-order effects and their implications to NUCOR frames, it is necessary to study and implement a second order analysis technique. The second-order member effect is caused by the axial loads acting through the laterally displaced portion between the ends of the beam-column. The second-order frame effect is caused by the axial loads acting through the laterally displaced frame as it undergoes a relative translation between member ends. Under compressive axial loads, these forces tend to destabilize the response of the frame. Under tensile loads, these forces tend to stabilize the frame. The magnitude of second-order effects is also dependent on the loading, as they increase under proportional loading, and decrease under nonproportional loading. As mentioned in Chapter I, in accordance with the ASD approach, second-order effects are calculated using an amplification of the elastic first-order member moments. These moments are subsequently used in the ASD stability interaction equation (Eqn 1.6-1a, 1978 ASD or Eqn. H1-1, 1989 ASD);

$$\frac{f_a}{F_a} + \frac{C_m}{(1-\frac{f_a}{F'_e})}\frac{f_b}{F_b} \tag{2.1}$$

where the amplification factor is the term

$$\frac{C_m}{(1-\frac{f_a}{F'_e})} \tag{2.2}$$

where C_m is dependent on loading and end restraints, and $F'_e = \pi^2 EI/(KL)^2$, divided by a factor-of-safety of 23/12.

Since there is no restriction on this quantity, it is conceivable that this amplification factor can be less than one (1.0), attenuating the second-order effects. The equation was developed for prismatic cross-sections, utilizing the end moments and axial force. It is to be used to define the factor-of-safety against buckling of the member between points of support. If used incrementally along the length of a beam-column, the writer suggests that it provides a check of the factor-of-safety against local buckling but not lateral buckling, especially when slender cross-sections are present.

Note that ASD does not recognize the attenuation of the design moments under tensile loads. Rather, only the yielding interaction check is required, based on the first-order analysis. Though not usually the governing load combination, there is a definite benefit for recognizing this interaction effect. For the purposes of this study, attention is given primarily to the interaction with compression, as it is the most commonly occurring state for the design of NUCOR frames.

The LRFD second-order analysis approach involves the use of dual amplification factors, one for member effects and one for frame effects. Again, these equations were developed for prismatic, rectangular geometry, not necessarily applicable for use with tapered members. The LRFD method requires two analyses to be performed for each load combination. The procedure is outlined as follows:

- Analyze structure with fictitious floor level restraints that resist lateral displacement of the structure;
- From this analysis, obtain the maximum member moments (M_{nt}) and the lateral forces at the fictitious supports required to resist lateral translation;
- Perform an analysis applying only the lateral forces found at the fictitious supports (in the reverse direction);
- From this second analysis, obtain the maximum member moments (M_{lt});
- Calculate amplification factors for each of the moments found in Steps 2 and 4, by use of the LRFD Specification Equations for B_1 and B_2, respectively;
- Calculate the ultimate moments for the member design by summing the moments times their respective amplification factors,

$$M_u = B_1 M_{nt} + B_2 M_{lt} \tag{2.3}$$

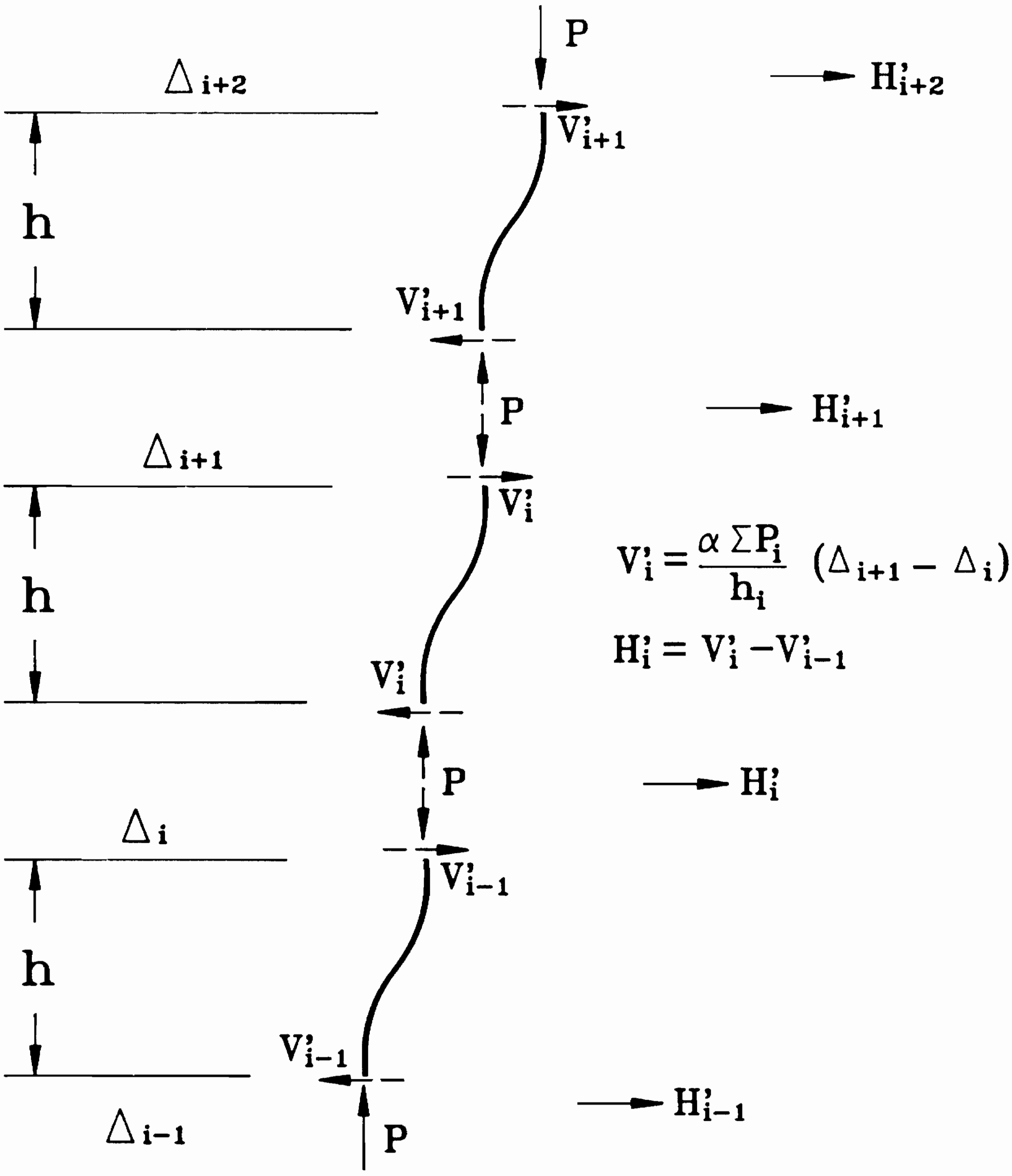
This procedure further assumes that the maximum design moments are synergistic, i.e., occur at the same point, a member end. This is not always correct, and can be overly conservative. Some of the alternate methods available and their applicability are discussed here. From this information, the most efficient and general method is chosen.

2.2 P-Delta Methods

Under the ASD Specification, recent literature (Adams 1976, Smith 1988) states that an unbraced frame may be designed as thought it were a braced frame if the analyst uses second order member forces in the design of the structure. This means that effective-length factors are less than or equal to 1.0. In order to apply the LRFD Specification, second order member forces are required for all designs.

In order to assess the second order effects of structures, the analyst may choose any of a number of techniques introduced over the past few years in the available technical literature. Each of these techniques has its own advantages and disadvantages. Some assess only frame (P-Δ) effects while others assess both frame (P-Δ) and member (P-δ) effects. Some of these second order analyses involve modified first-order analyses, and others involve a nonlinear stability function approach to model the geometric changes and decrease in frame stiffness under the design loads. A true second-order elastic analysis considers both the member and frame effect, though for most practical cases of low-rise buildings, the member effect is negligible (Galambos 1968).

Frame (P-Δ) Effect Only. The most popular model developed to assess the frame (P-Δ) effect was introduced by Adams (1976). Adams' sway subassemblage model, Figure 2.1 utilizes a method that involves calculating additional sway forces (applied at each story) that are calculated based on magnitude of the vertical loads and the current lateral displacements. This



Subassemblage Beam–Column Model for Adams' P–Δ Method

Figure 2.1

procedure requires iteration, since the structure continues to displace laterally under the increased lateral loads, which requires further increases in the lateral load. The calculated lateral displacements will converge for most structures, although Adams has recognized that his method is unreliable and unconservative for very flexible framing. Under these circumstances, the analysis will not converge and the structure is regarded as unstable. It should also be noted that this model was only intended for prismatic, rectangular geometry. Its applicability to tapered and/or non-rectangular geometry has neither been proven nor disproven.

Under the presence of gravity loading only (no lateral loads), Adams recommends that the structure be analyzed assuming initial imperfections due to out-of-plumbness of the columns or .002h per story. White (1990) reports that a statistical model developed to determine the distribution of column out-of-plumbness in common building construction is

$$\Delta_i = \frac{0.006h}{n^{0.445}} \tag{2.4}$$

where h = story height, n = number of columns in the story, and Δ_i = initial out-of-plumbness for story i. This initial imperfection can be introduced into the structural model in two ways. First, the out-of-plumbness can be used in a computer analysis by specifying the joint coordinates as being geometrically out-of-plumb, and then perform a second-order analysis. The second method, more suited to hand computation, is to calculate fictitious initial sway forces caused by the fictitious initial imperfections, and then perform first-order analysis iterations. This procedure is suggested by Adams (1976).

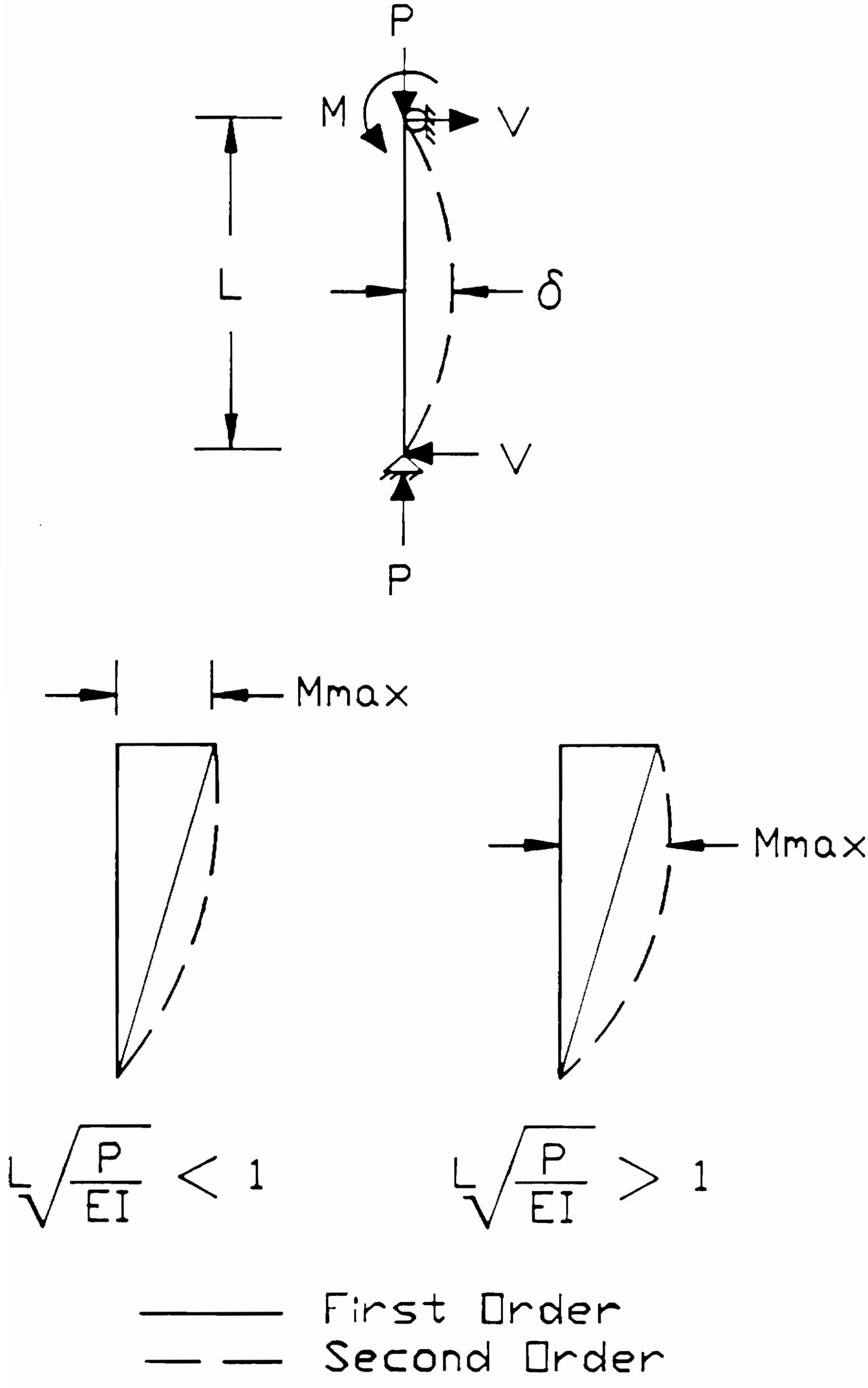
Though the frame (sway) effect is the most easily quantifiable second-order effect, the member (nonsway) effect is more subtle and can outweigh the effects of the sway case under high axial loads, since the maximum moment can move in-span. The location of the maximum moment under high axial loads is explained below. Since Adams' method does not include the member effect, it is required to find a method that can model both effects accurately. These methods are discussed next.

Frame (P-Δ) and Member (P-δ) Effects. Complicating the assessment of member P-δ effects is the possibility of the maximum moment not occurring at a joint or under a load. Depending on the value of the axial load, the maximum moment may move away from the point at which the maximum moment is first thought to occur. This effect is shown in Figure 2.2. This behavior is dependent on a stability parameter that varies with the axial load, flexural rigidity, and unbraced member length. When

$$L \sqrt{\left(\frac{P}{EI}\right)} < 1.0 \tag{2.5}$$

the axial force influence accounts for less than five percent of the second-order behavior (Galambos 1968). When this parameter is greater than one, it is recommended that a complete SOEA be performed, as the response can change dramatically with small increases of the applied load.

Therefore, an analyst must pay close attention to the significance of this quantity, as the maximum moment at the joints or under loads may not be the correct maximum moment. Chen and Lui (1987) describe a method to calculate the location and value of the maximum in-span moment, should the axial loads be



Location of Mmax

Figure 2.2

of such magnitude to force the maximum along the member length. For the purposes of this study, the maximum moment is assumed to occur at preselected joint coordinates, since NUCOR frames generally possess adequate in-plane flexural rigidity such that the stability parameter described above is less than 1.0.

Convenient methods for assessing the frame and member second-order effects that involve modifications to first-order analyses were developed by LeMessurier (1977) and Lui (1988). Both are approximate in application, though Lui's method will converge to a correct solution with a sufficient number of iterations.

LeMessurier's method is an amplification factor approach that does not require iteration, but does require lateral translation. It involves calculation of an individual or story lateral load and a parameter that accounts for the presence of compressive axial loads. LeMessurier accounts for compressive axial force influence and also was also intended for prismatic, rectangular framing. It will not assess the member effect without a lateral translation between the ends of the member.

A limitation to both the Adams and LeMessurier methods is that amplification of the design moments is only carried out for columns. The subassemblage models in these procedures assume the interconnecting beams act as rigid bodies, and therefore do not participate in the moment redistribution. The amplification of column moments is not carried over to the connecting beams or lateral force resisting elements of the bracing system. For these cases, Lui's method may be advantageous.

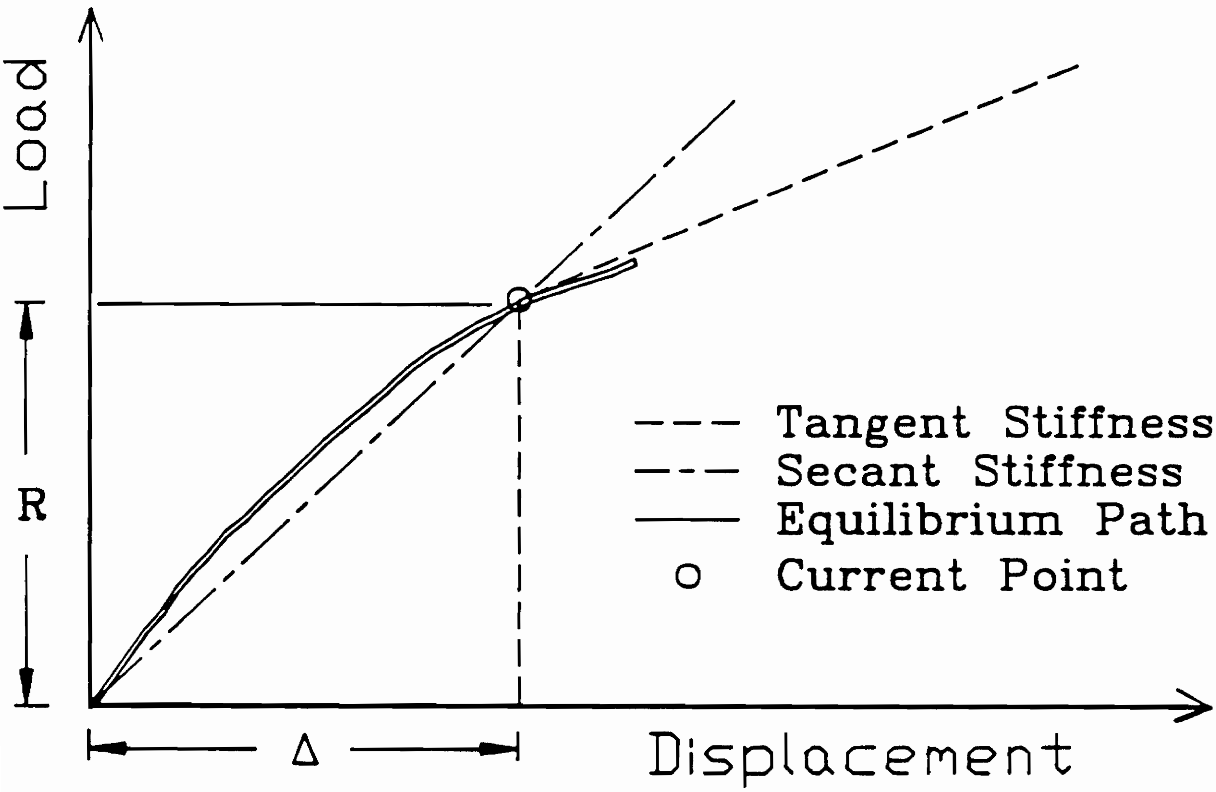
Lui's method, which has no limiting assumptions (besides the normal Bernoulli-Euler beam theory assumptions), is applicable to both ASD Types 1, 2, and 3 framing, and correspondingly, LRFD Type FR and PR framing. It accounts for both member (P-Δ) and frame (P-δ) effects, and can be applied to tapered framing.

This analysis requires iteration and becomes quite complicated for even simple loading distributions, though Lui states that one iteration is sufficient for preliminary design. The frame P-Δ effects are accounted for by pseudo lateral joint forces at each story, similar to the Adams' method described above. The member P-δ effect is more difficult to calculate. It requires scaling the moment diagram for each structural member by the factor P/EI, and applying this diagram as a distributed load to the individual members. Hence, the member effect on a tapered column can be calculated. This loading generally is one degree higher than the loads originally on the structure. For example, a pinned base, single bay portal frame subjected to a single concentrated load at the center of the rafter member produces a linear moment in the columns, which, when scaled with the P/EI factor, becomes a linearly increasing distributed load on the column. For the rafter, the scaled P/EI loading becomes three triangular loads. Because of this, iteration by hand is very tedious and the designer may become swamped in the bookkeeping. For a multi-story frame under x number of load cases, the designer can see that this analysis could take weeks to perform without computer assistance.

Probably the most efficient and general approaches are computer methods (Chen and Lui 1987, Oran 1973, Smith 1988, Wang 1973) which account for both P-Δ and P-δ effects, and are variations on the displacement method of matrix structural analysis. These procedures utilize the stability functions coupled with the slope deflection equations (including axial load). These approaches also recognize the increase in member stiffness under the influence of tensile loads and the decrease in member stiffness under the action of compressive loads. These methods also require iteration for the nonlinear response, since the stability functions are dependent on the magnitude and type of axial loads in the members. However, they are programmable for a computer and iteration becomes a trivial matter in the overall analysis and design.

Since a matrix-structural analysis approach yields the most efficient and generally applicable method of analysis, it was chosen as the preferred analysis tool. The specifics of the solution of equations and system model development are described in Chapter III. NUCOR currently uses a first-order elastic matrix-displacement approach with uniform elements for the analysis of their tapered frames. With some modifications, it could include second-order elastic analysis results.

The method of second-order elastic analysis for this study is a two-fold computer model approach. The structural system model is developed using element tangent stiffness matrices developed by Oran (1973), and the internal force-deformation response is developed using element secant stiffness matrices. In this incremental/iterative approach, response is traced using the system tangent stiffness as a predictor and the element secant stiffness is used as a corrector. Figure 2.3 is a graphical description of this approach. As shown in the figure, the tangent stiffness represents a tangent to the response curve, for the values of the current displacements. The secant stiffness is calculated from the initially undeformed configuration (Total-Lagrangian method) and accurately represents the internal force-deformation response. Applying this method, the response of the structure up to the elastic stability limit state can be traced.



Tangent/Secant Stiffness
Figure 2.3

Further, since this method uses an exact solution to the beam-column differential equation, the results are consistent for prismatic geometries (in accordance with small strains and neglecting shear deformations), but approximate for the finite element model necessary for tapered beam-columns.

Since the path is nonlinear, the structure must be loaded incrementally, and iterations performed to converge to an equilibrium point on the response curve. It is not recommended to apply more than one load increment for a preliminary analysis/design, as the second-order effects may be overestimated with the initial structural model elements (White 1990). As the design stage progresses, the number of load increments can be increased to choose the most efficient structural sections.

A design criterion has been suggested (Adams 1976, LeMessurier 1977, Smith 1988) such that the in-plane effective-length factor can be taken as 1.0 when the results of second-order elastic analysis have been used in the member selection process. Assuming that the in-plane slenderness controls stability, additional requirements for attempting to utilize $K_x = 1.0$ have also been suggested:

- Under ASD, the upper limit on the ratio of fa/Fa, calculated at a first-order effective-length, can be conservatively taken as 0.85 (White 1990). Intuitively then, for an LRFD approach, the $Pu/\phi Pn$ ratio should be limited to approximately 0.9. Except for the lower half of the tapered columns of the frames studied, this criterion seems to be met.
- The interaction strength as a beam-column may be computed based on an effective-length factor of 1.0, if the ratio P_u/Py is less than 0.5 (Yura 1987). This also seems to be met generally for typical NUCOR frames.

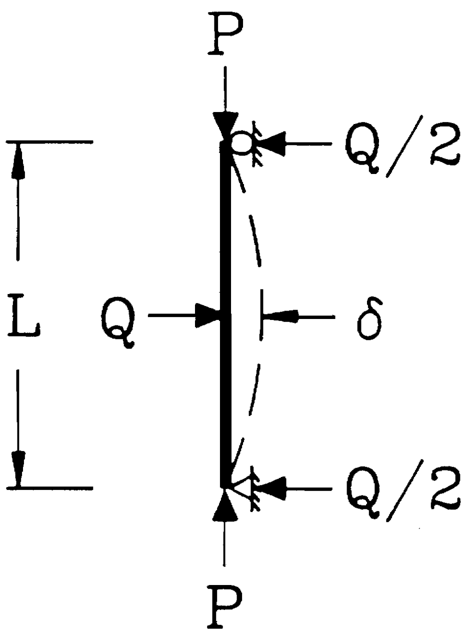
- Shear deformation is negligible for the buckling mode. Shear deformation has the effect of lowering the buckling load, and increasing the effective-length of the column (Galambos 1976). It is only of concern for deep beams.

2.3 Beam-Column Examples

Presented here are two beam-column examples. The first, a member effect example, and second, a sway effect example. It is shown that the methods described above may produce different results, supporting the writer's choice of using the computer solution method. The same analysis assumptions listed in the introduction apply to these examples.

Member Effect Example. Given the beam-column shown in Figure 2.4, which is subjected to a lateral load, Q, at midheight of the column, and compressive axial force, P, at the top of the column. The member is rotationally unrestrained at its ends, and not subject to lateral translation (nonsway). The member is typical of the proportions of H-sections found in the AISC Manuals, approximately a W14x90 section. It is desired to find the maximum second-order elastic moments by the theories discussed above.

By inspection, the maximum moment occurs under the lateral load, Q. The solution of this problem is given by Timoshenko and Gere (1961). The linear analysis shows no amplification of the design moment, consistent with its theory. However, when the applicable amplification factor (ASD stability interaction equation factor on the moment term) is calculated, the degree of second-order effects has been overestimated according to a correct solution. Notice that with no sway involved, both the Adams' and LeMessurier's methods do not recognize the increase in moment at center-span (LeMessurier's model must have chord



Parameters

$P = 100$ kips
$Q = 20$ kips

$E = 29000$ ksi
$I = 987$ in^4
$L = 200$ in

Analysis Method	M℄ [in-k]
Timoshenko & Gere	1011.81
Linear	1000.00
Adams P–Delta	1000.00
LeMessurier P–Delta	1000.00
ASD Amplification	1027.88
Lui's P–Delta	1011.64
Proposed PDELTA	1011.81

Beam–Column Problem
(Non–Sway)

Figure 2.4

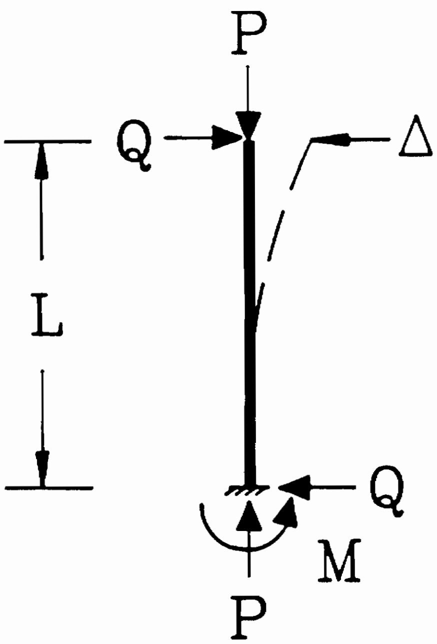
rotation). Lui's method with one iteration (as he suggests) presents a lower bound, and yields a correct solution. The solution afforded by the proposed computer method developed as part of this research yields a corrects solution, since the given problem satisfies the assumptions of the system model.

Frame Effect Example. Similarly, given the cantilever beam-column shown in Figure 2.5, which is subjected to a lateral load, Q, and a compressive axial force, P, at the top of the column. The member is rotationally unrestrained at the free end and is subject to lateral translation. The column is a typical H-section found in the AISC Manuals, approximately a W14x90 section. Again, it is desired to find the maximum second-order elastic moments according to the theories discussed above.

By inspection, the maximum moment occurs at the fixed end. The solution of this problem is given by Timoshenko and Gere (1961). Also, a linear analysis shows no amplification of the design moment, but when the applicable amplification factor is calculated, the degree of second-order effects has been bounded by the LRFD amplification factor method. This is due to the different approaches used in the development of the sway-amplification factors (the LRFD Eqn. H1-5 (B_2) was developed from Adams' theory, and LRFD Eqn. H1-6 (B_2) was developed from LeMessurier's approach).

LRFD Equation H1-5:

$$B_2 = \frac{1}{1 - \Sigma P_u\left(\frac{\Delta_{oh}}{\Sigma HL}\right)} \tag{2.6}$$



Parameters			
P	=	100	kips
Q	=	20	kips
E	=	29000	ksi
I	=	995	in^4
L	=	200	in

Analysis Method	M [in-k]
Timoshenko & Gere	4195.69
Linear	4000.00
Adams P-Delta	4193.79
LeMessurier P-Delta	4195.84
LRFD (H1-5)	4238.11
LRFD (H1-6)	4193.78
Lui's P-Delta	4193.40
Proposed PDELTA	4195.66

Beam-Column Problem
(Sway)

Figure 2.5

LRFD Equation H1-6:

$$B_2 = \frac{1}{1 - \frac{\Sigma P_u}{\Sigma P_e}} \tag{2.7}$$

Lui's solution with one iteration is close to a correct solution again, as a lower bound. The proposed computer method developed as part of this research yields a correct solution, since the given problem satisfies the assumptions of the system model.

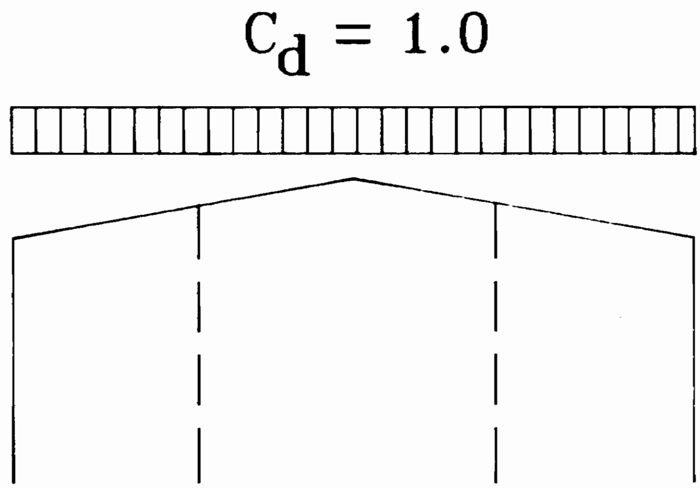
As shown in the beam-column examples, the proposed computer method gives a correct solution, and is the most convenient and efficient method available to solve large degree-of-freedom problems. For this reason, the proposed computer method is employed for this research.

CHAPTER III
RIGID FRAME MODELING

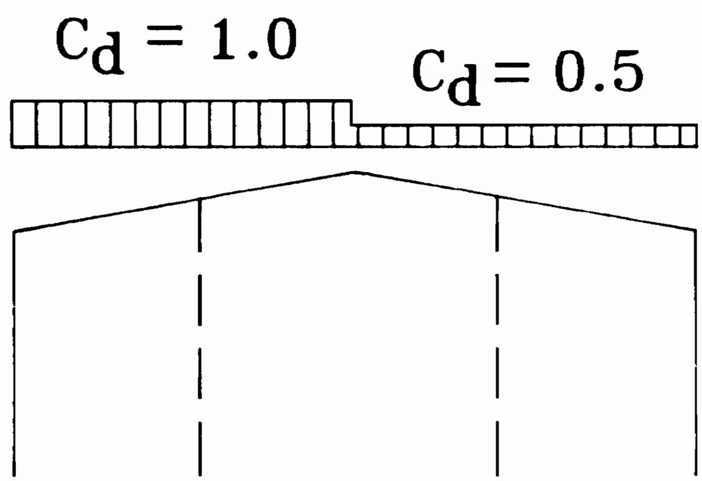
3.1 Design Loads

The basic design loads were established in order to represent the most frequently occurring structural loadings. They represent, under a given set of conditions dependent on geography, the expected loading conditions to which the buildings will be subjected. To a large extent, NUCOR's prefabricated buildings are marketed and designed for the midwestern U.S. region. Typical structural loads for this region can be found in ANSI A58.1, "Minimum Design Loads for Buildings and Other Structures," and the Uniform Building Code (ICBO 1988).

Ground snow loads for this area are typically on the order of 30 psf to 40 psf. The basic wind speed used for the design wind pressures is 80 mph. The distribution of snow and/or wind loads on the structural frame have been determined through statistical analysis. The distributions used for this study were taken from the UBC and the LRBSM. For the balanced and unbalanced snow load cases, Figure 3.1 graphically depicts the distribution of the snow load. Tributary areas for each purlin location are computed, and the vertical projection of the snow load is calculated as a concentrated load, and applied to the rafter. The wind load distributions shown in Figure 3.2 are taken from the LRBSM, since the UBC wind load exposure coefficients are currently in a process of amendment (ICBO 1990). Similarly, the tributary areas of the purlins and girts are calculated,



a. Full Snow Load

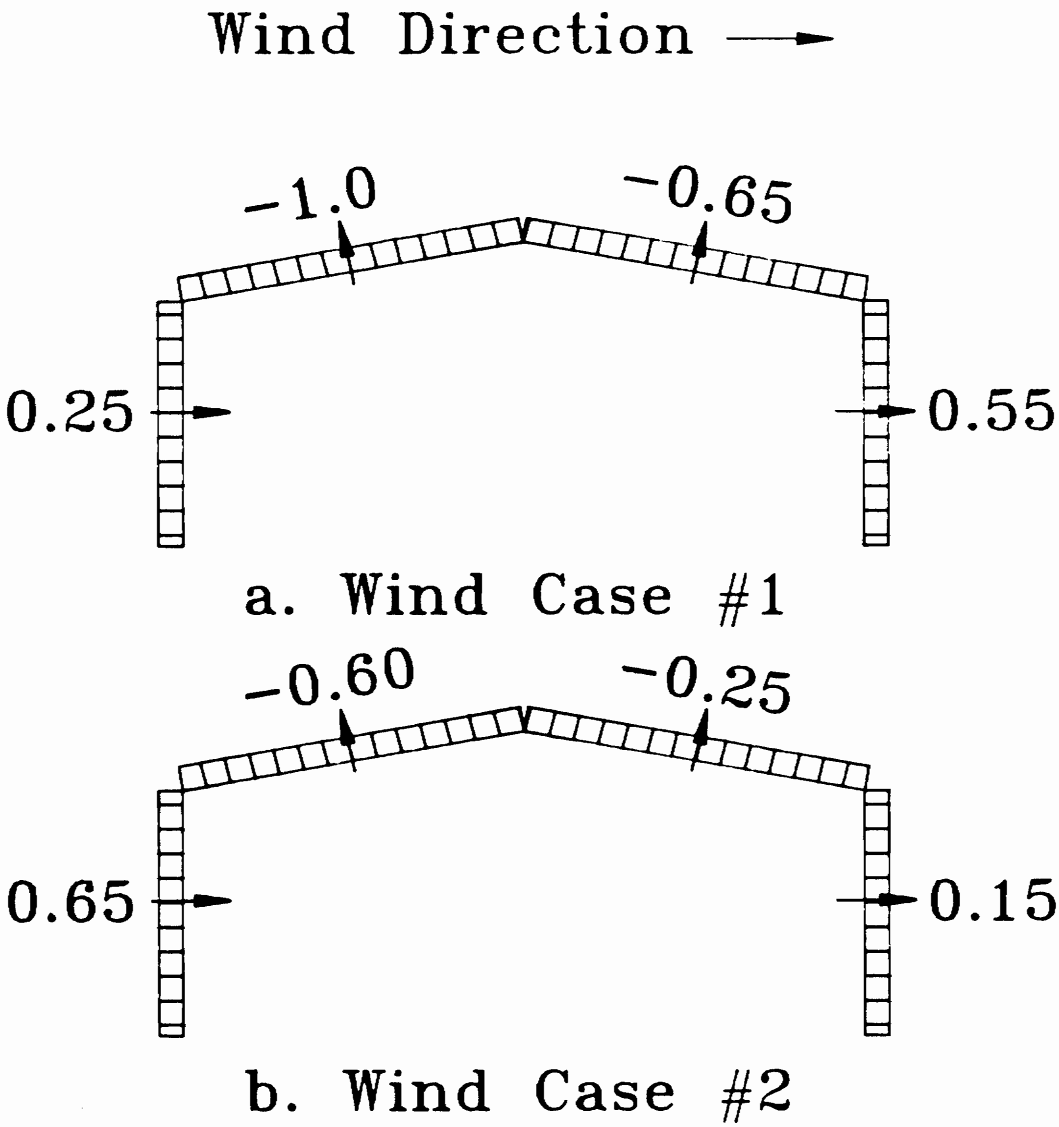


b. Unbalanced Snow Load

Key: – — – Optional Column

p_f $0.7 C_d p_g$

Snow Load Distributions
1988 Uniform Building Code
Figure 3.1



Note: $q = 0.00256V^2 (\frac{H}{33})^{2/7}$ [psf]

H = Eave Height [ft]

V = Wind Speed [mph]

Wind Load Distributions
1986 MBMA Design Code

Figure 3.2

and the horizontal and vertical components of the forces are calculated and applied to the frame components. The distribution coefficients for both snow and wind have been established statistically.

According to the UBC and the LRBSM, critical load cases are to be investigated, which cause maximum load effects. Under usual circumstances, the critical load combinations for single-span, solid-webbed, gabled frames are those of dead and snow acting together, and dead with snow and wind acting together. The probability factors of the maximum values of loads occurring simultaneously is better defined by the LRFD approach, but the ASD load combination factors have been in use for decades, and are still acceptable.

Acting in combination, i.e., tabulating the loads according to the statistically-factored load combinations (ANSI A58.1 1982), the effects of snow and wind occurring simultaneously are developed in accordance with the UBC provisions. The combinations shown in Table 3.1 are used in this research. The 16 LRFD combinations are shown in order to contrast them with their equivalent ASD combinations. Notice that the magnitude of the probability factors on the load combinations are widely different between the ASD and LRFD combinations, but analytically, are very similar (i.e., maximum snow and half wind, maximum wind and half snow, etc.). Each of the 16 load combinations needs to be checked in order to safely establish where and under what combination the maximum effects occur. The load combination most frequently governing is dead+live though in some locations other combinations govern. In general, it is necessary to apply all the combinations to locate the maximum interaction values. This is true especially for tapered framing, as a frame optimized for one combination may be unconservative under another. The 16 combinations require the frame to

Load Combinations

Allowable Stress Design

1) 1.00 * (D + SL + SR)
2) 1.00 * (D + SL + SR/2)
3) 0.75 * (D + WLL1 + WLR1)
4) 0.75 * (D + WLL2 + WLR2)
5) 0.75 * (D + SL + SR + WLL1/2 + WLR1/2)
6) 0.75 * (D + SL + SR + WLL2/2 + WLR2/2)
7) 0.75 * (D + SL + SR/2 + WLL1/2 + WLR1/2)
8) 0.75 * (D + SL + SR/2 + WLL2/2 + WLR2/2)
9) 0.75 * (D + SL + SR/2 + WRL1/2 + WRR1/2)
10) 0.75 * (D + SL + SR/2 + WRL2/2 + WRR2/2)
11) 0.75 * (D + SL/2 + SR/4 + WLL1 + WLR1)
12) 0.75 * (D + SL/2 + SR/2 + WLL2 + WLR2)
13) 0.75 * (D + SL/2 + SR/4 + WLL1 + WLR1)
14) 0.75 * (D + SL/2 + SR/4 + WLL2 + WLR2)
15) 0.75 * (D + SL/2 + SR/4 + WRL1 + WRR1)
16) 0.75 * (D + SL/2 + SR/4 + WRL2 + WRR2)

Load & Resistance Factor Design

1) 1.2D + 1.6SL + 1.6SR
2) 1.2D + 1.6SL + 0.5SR
3) 1.2D + 1.3WLL1 + 1.3WLR1
4) 1.2D + 1.3WLL2 + 1.3WLR2
5) 1.2D + 1.6SL + 1.6SR + 0.8WLL1 + 0.8WLR1
6) 1.2D + 1.6SL + 1.6SR + 0.8WLL2 + 0.8WLR2
7) 1.2D + 1.6SL + 0.5SR + 0.8WLL1 + 0.8WLR1
8) 1.2D + 1.6SL + 0.5SR + 0.8WLL2 + 0.8WLR2
9) 1.2D + 1.6SL + 0.5SR + 0.8WRL1 + 0.8WRR1
10) 1.2D + 1.6SL + 0.5SR + 0.8WRL2 + 0.8WRR2
11) 1.2D + 1.3WLL1 + 1.3WLR1 + 0.5SL + 0.5SR
12) 1.2D + 1.3WLL2 + 1.3WLR2 + 0.5SL + 0.5SR
13) 1.2D + 1.3WLL1 + 1.3WLR1 + 0.5SL + 0.25SR
14) 1.2D + 1.3WLL2 + 1.3WLR2 + 0.5SL + 0.25SR
15) 1.2D + 1.3WRL1 + 1.3WRR1 + 0.5SL + 0.25SR
16) 1.2D + 1.3WRL2 + 1.3WRR2 + 0.5SL + 0.25SR

Key:
SL: Snow on Left
SR: Snow on Right
WLL1: Wind from Left on Left, Case 1
WLR2: Wind from Left on Right, Case 2
WRL1/2: 0.5*(Wind from Right on Left, Case 1)

Table 3.1

withstand a variety of structural loadings, from maximum snow to maximum wind and all variations in-between. The critical load combination generally not known a priori.

Since the loads to be applied to the main structural framing are given in terms of a pressure distribution, a discretization is employed to model the load transfer. Since the given loads are transferred to the primary frames through simple-span purlins, the loads induced on the frames are of a concentrated type. The concentrated loads are determined by calculating the tributary area of each purlin, and multiplying this square-foot area by the values of the tabulated pounds-per-square-foot basic load. These equivalent concentrated loads are then applied to the structural model, just as they would be applied to the actual frame. The calculations must also account for slope of the roof, in order to determine vertical and horizontal concentrated loads at each purlin or girt. These calculations are trivial and not repeated here, but are developed by the standard vector transformations.

Note: The analysis program written for the study does not account for uniform loads, since the frames studied are subjected only to concentrated loads, provided at purlin or girt locations. Element selfweight is actually a distributed load, but was discretized into equivalent concentrated vertical forces at the ends of the elements. The fixed-end-moments due to the uniformly distributed selfweight, consistent with the FEM, was neglected. This is done due to the fact that the dead load of the structure is negligible for these types of frames, and would only further increase the analysis time by accounting for these trivially small moments.

3.2 Frame Discretization

The structural model is developed through a segmental discretization, called elements. These elements join points that are of particular interest to the analyst. The joints are selected in order to describe the behavior of the structure in an overall way, as the element model does not give the continuum behavior. Of particular interest in unbraced rigid frames, are the lateral movements of the tops of the columns (drift), the vertical movement at the gable, the rotations at the bases of the columns, etc. With the knowledge of behavior at specific points of interest, the frame can be designed to withstand the structural loadings.

A prismatic, single-bay portal frame can be represented with only three elements, one each for the columns, and one interconnecting beam, according to the first-order elastic matrix displacement method (MDM) of structural analysis. However, the use of these prismatic elements to model the behavior of a tapered beam can introduce severe discretization errors (Yang 1986). This discretization must be chosen carefully in order to minimize the potential error. The use of uniform elements in the model of a tapered beam goes beyond the theory of the matrix displacement method. Its roots are in the finite element method (FEM), and should be treated as such. Note that the FEM degenerates to the MDM, for the case of prismatic members.

In the first-order elastic analysis of prismatic structures then, not much thought is required in developing a suitable model, once the analyst is familiar with the process. However, for a second-order elastic prismatic beam-column problem, selection of the proper number of elements for an accurate model is another subject. White (1990) suggests a criterion for element selection, based on the value of the expected maximum axial load and its Euler buckling load. This criterion is that the quantity P/P_e is held to be below 0.4. It is suggested that a typical beam-column of prismatic section requires a maximum number of three elements to hold the error to less than 1%. (White 90) Since this study is to model tapered beam-columns, the writer recommends that this upper bound be used as a lower bound for the element selection, as discretization errors are introduced by selecting too few elements. Appendix C discusses the reason for this suggestion.

The error introduced by the use of uniform elements can be quite substantial, and is the subject of Appendix C. Minimization of this error can be accomplished by using a higher number of prismatic elements, or using an element model that was developed as possessing a taper. The use of tapered elements can reduce the number of degrees of freedom in the analysis, thereby reducing analysis time. The discretization of the frame into the elements described above is the subject of the next section.

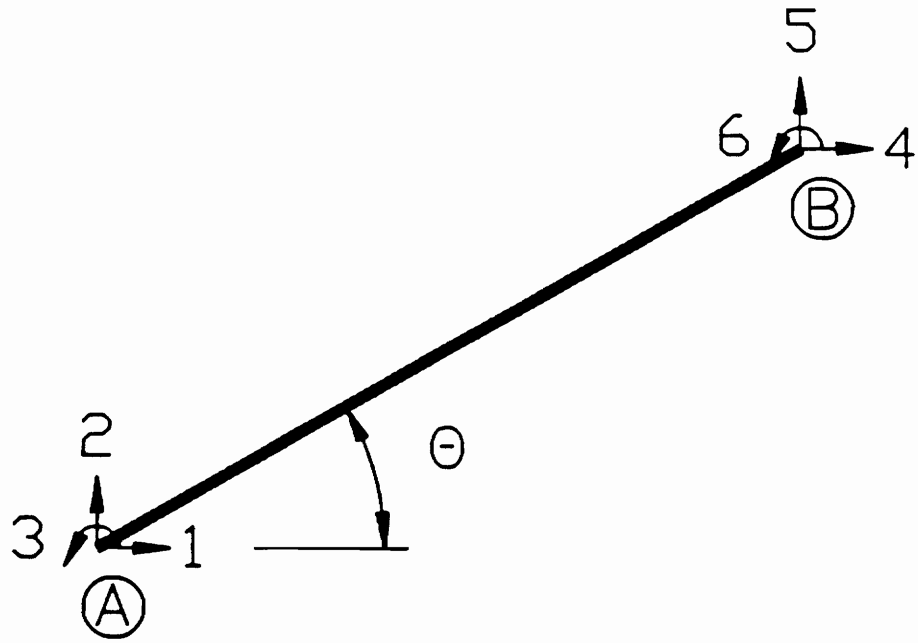
3.3 Element Models

Section 3.2 describes the discretization of a structural model through the analysts' identification of points of interest. These points of interest are labeled as joints or nodes. Whether or not two or more members actually intersect at the so-called joint is not important. A joint can be placed anywhere. In a two-dimensional plane frame, each joint possesses three degrees of freedom (DOF) i.e., the horizontal and lateral translational components, and a rotation. Quantifying the amount of deformation at each of these DOFs, the behavior under a particular load combination can be shown by drawing the displaced state of the structure. Between each of these joints, a beam-column element must be defined to store the induced flexural and axial energies introduced by the loading. These elements are defined next.

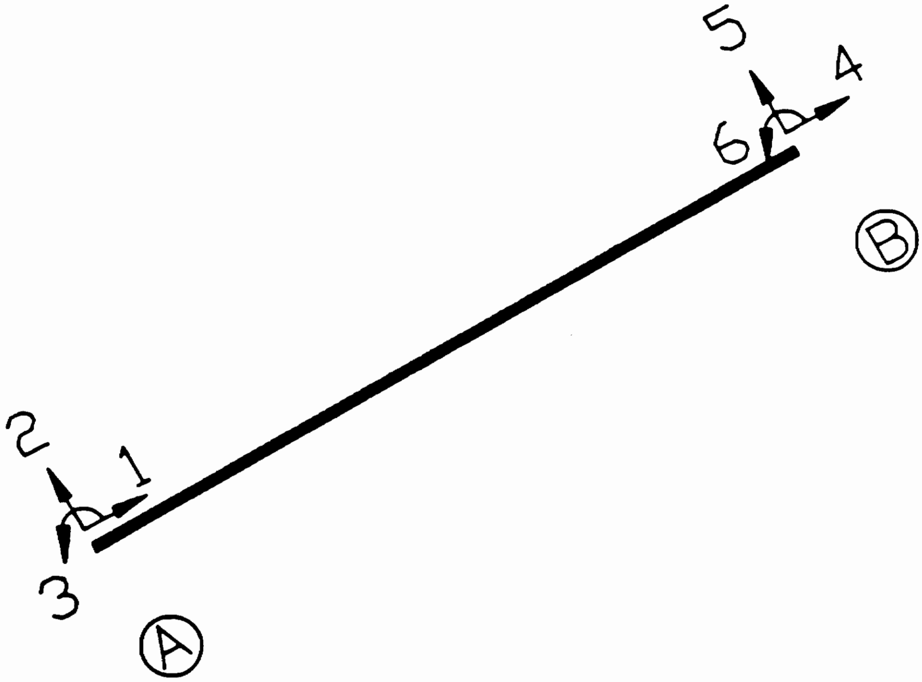
After global degrees of freedom are selected, the elements between them are identified and transformed into the global set of stiffness equations. Figures 3.3 and 3.4 show a typical prismatic beam-column element labeled with its elemental degrees of freedom for both the global and member system. Once these elements are mapped into the system of stiffness equations, and the system is solved for the displacements at the joints, the element forces can be determined. Figures 3.5 and 3.6 show the beam-column element labeled with its elemental global and local force systems, which are calculated first by the joint forces, and then rotating them with the standard transformation matrix.

It is important to remember that this research does not follow the usual procedure of the MDM, not only because of the tapered frame behavior, but also, and primarily, because of the required nonlinear analysis. As described above, a tangent-stiffness was used to predict response and a secant-stiffness was used to model the internal force-deformation behavior. Because of these differences, the element model used to map element stiffnesses into the system model is different from the element model used to determine the internal response.

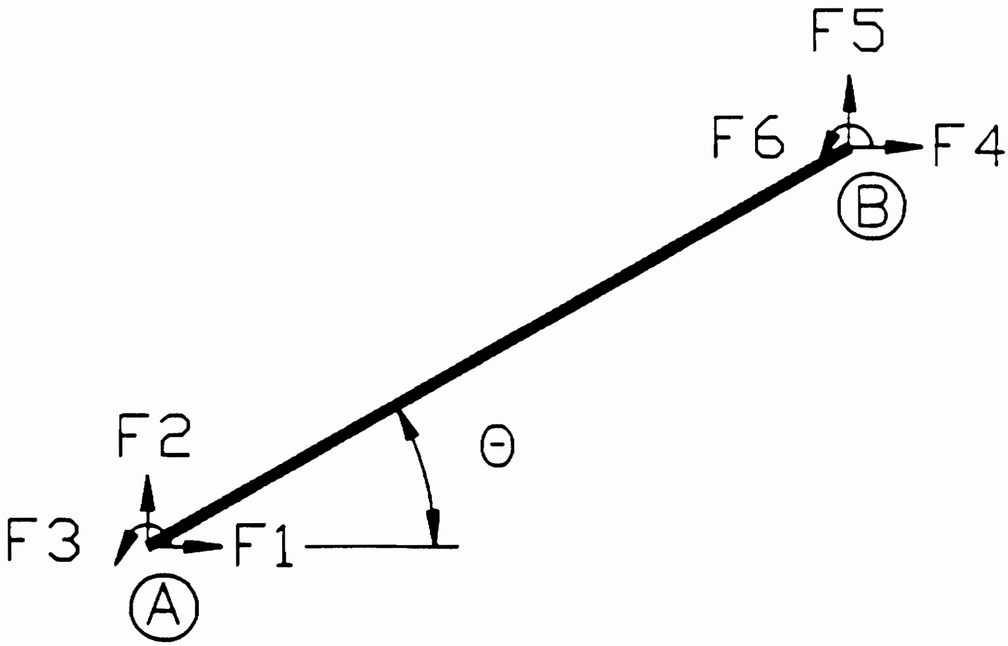
The element tangent stiffness matrices used in this research were developed by Oran (1973). Figure 3.7 shows an element global tangent-stiffness matrix, in notation identical to that of Holzer (1985), except for the ϕ terms on the rotational stiffness terms. These ϕ terms reduce(increase) the amount of stiffness present at these terms under axially compressive(tensile) loads. If the axial load is neglected or zero, the stiffness matrix will reduce to the standard first-order beam-column element matrix.



Element GDOF Definition
Figure 3.3

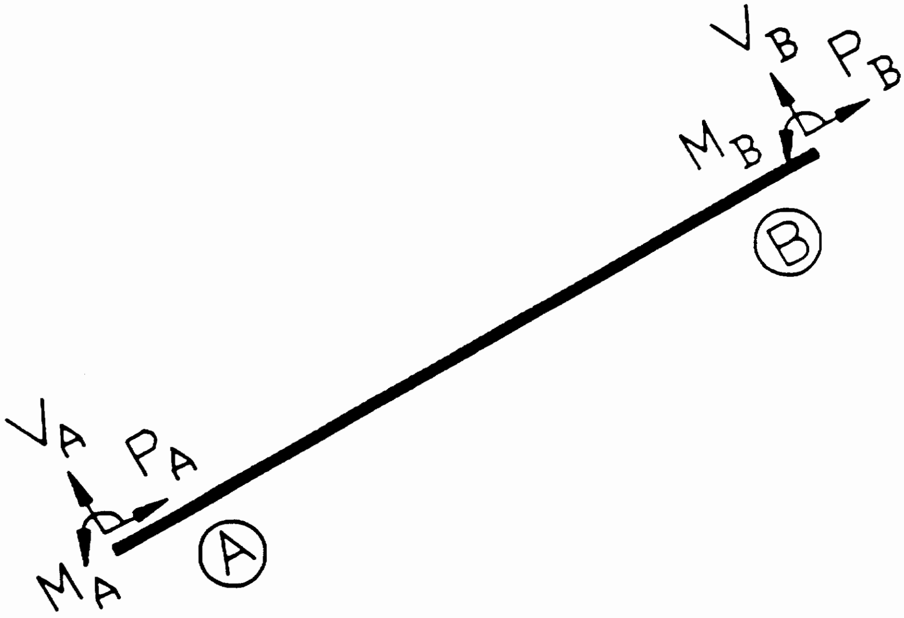


Element LDOF Definition
Figure 3.4



Element Global Forces

Figure 3.5



Element Local Forces

Figure 3.6

$$
K_t = \begin{bmatrix}
g_1 & g_2 & g_4 & -g_1 & -g_2 & g_4 \\
 & g_3 & g_5 & -g_2 & -g_3 & g_5 \\
 & & g_6 & -g_4 & -g_5 & g_7 \\
 & & & g_1 & g_2 & -g_4 \\
 & & & & g_3 & -g_5 \\
 & \text{Symmetric} & & & & g_6
\end{bmatrix}
$$

$$\alpha = \frac{EI}{L^3}$$

$$\beta = \frac{AL^2}{I}$$

$$c_1 = \cos\Theta$$

$$c_2 = \sin\Theta$$

$$g_1 = \alpha(\beta c_1^2 + 2c_2^2\phi_3)$$

$$g_2 = \alpha c_1 c_2(\beta - 2\phi_3)$$

$$g_3 = \alpha(\beta c_2^2 + 2c_1^2\phi_3)$$

$$g_4 = -\alpha\phi_3 L c_2$$

$$g_5 = \alpha\phi_3 L c_1$$

$$g_6 = \alpha\phi_1 L^2$$

$$g_7 = \alpha\phi_2 L^2$$

Compression

$$\phi_1 = kL(\sin(kL) - kL\cos(kL))/\phi_c$$

$$\phi_2 = kL(kL - \sin(kL))/\phi_c$$

Tension

$$\phi_1 = kL(kL\cosh(kL) - \sinh(kL))/\phi_t$$

$$\phi_2 = kL(\sinh(kL) - kL)/\phi_t$$

Where:

$$\phi_c = 2 - 2\cos(kL) - kL\sin(kL)$$

$$\phi_t = 2 - 2\cosh(kL) + kL\sinh(kL)$$

$$\phi_3 = \phi_1 + \phi_2$$

Tangent Stiffness Matrix

Figure 3.7

As an abstraction, it is interesting to note that the geometric stiffness matrix for a beam-column that is usually used to represent the tangent stiffness was compared with that of Oran's. It was found that Oran's tangent stiffness converged to the solution faster than that of the usual matrix. For this reason, Oran's matrix was used rather than staying with usual practice.

The element secant stiffness matrices (shown in Figure 3.8) used in this research are taken from Chen and Lui (1987), and represent the internal force-deformation behavior of the element, consistent with the exact solution to the differential equation. The element model was developed for end moments, shears, and forces, with no in-span lateral loads. Again, should the effect of axial load on the flexural stiffnesses be neglected, the stiffness matrix will reduce to the standard first-order beam-column element matrix.

3.4 System Model

The system model is assembled according to the usual mapping of element stiffness terms into the global stiffness matrix. The computer program stores the stiffness terms in a single column vector, and the locations of elements of the main diagonal are stored in a column vector. This technique is employed in order to optimize the solution process, taking advantage of symmetry of the system stiffness matrix. Because of symmetry, and the fact that the system stiffness matrix is closely banded to the main diagonal, a solution technique called the active column solver is utilized. This facilitates faster solution, since storing the entire DOFxDOF system stiffness matrix would unnecessarily require the storage of large quantities of zeroes, and mathematical operations on them. By

$$k_s = \frac{EI}{L}\begin{bmatrix} \frac{A}{I} & 0 & 0 & -\frac{A}{I} & 0 & 0 \\ & \frac{12}{L^2}\phi_1 & \frac{6}{L}\phi_2 & 0 & -\frac{12}{L^2}\phi_1 & \frac{6}{L}\phi_2 \\ & & 4\phi_3 & 0 & -\frac{6}{L}\phi_2 & 2\phi_4 \\ & & & \frac{A}{I} & 0 & 0 \\ & & & & \frac{12}{L^2}\phi_1 & -\frac{6}{L}\phi_2 \\ \text{Symmetric} & & & & & 4\phi_3 \end{bmatrix}$$

$$K_s = \Lambda^T k_s \Lambda$$

	Compression	**Tension**
ϕ_1	$\frac{(kL)^3\sin(kL)}{12\phi_c}$	$\frac{(kL)^3\sinh(kL)}{12\phi_t}$
ϕ_2	$\frac{(kL)^2(1-\cos(kL))}{6\phi_c}$	$\frac{(kL)^2(\cosh(kL)-1)}{6\phi_t}$
ϕ_3	$\frac{(kl)(\sin(kL)-kL(\cos(kL))}{4\phi_c}$	$\frac{(kL)(kL\cosh(kL)-\sinh(kL))}{4\phi_t}$
ϕ_4	$\frac{(kL)(kL-\sin(kL))}{2\phi_c}$	$\frac{(kL)(\sinh(kL)-kL)}{2\phi_t}$

Where: $\phi_c = 2-2\cos(kL)-kl\sin(kL)$

$\phi_t = 2-2\cosh(kL)+kL\sinh(kL)$

Secant Stiffness Matrix

Figure 3.8

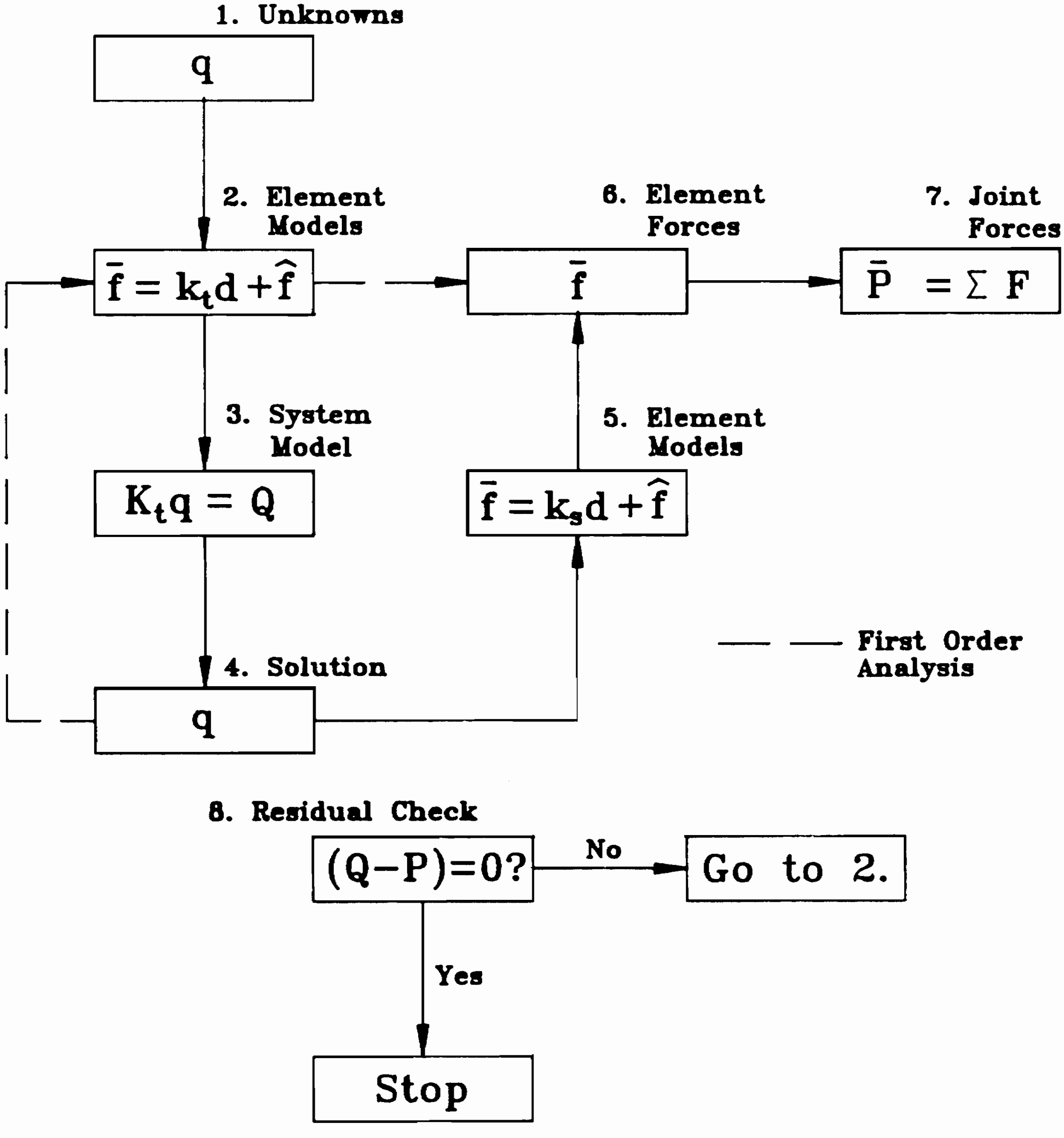
this method, computer memory requirements are reduced, and the solution progresses at a faster pace.

Probably the largest disadvantage to a nonlinear problem is the solution effort required to solve the systems of equations. With the element stiffness matrix coefficients dependent on the magnitude and direction of the axial load, the loss of superposition is of a serious nature. With a second-order elastic analysis of the type used here, the system of equations must be solved for each load combination, load increment, and equilibrium iteration. This means that for sixteen load combinations, three load increments/combination, and (say) five iterations per increment is required for equilibrium, the entire system of equations is solved 32*3*5 = 480 times. Obviously, this is best suited to computer solution. The analysis scheme is discussed next.

3.5 Solution of System Equations

Figure 3.9 shows the differences between the requirements of a first-order elastic analysis and a nonlinear, second-order elastic analysis technique. The flowchart is shown in the format presented by Holzer (1985), with the modifications required for the nonlinear analysis shown. This is done to begin to describe the effort involved in performing a nonlinear analysis. As indicated, an FOEA follows the dashed lines on the chart.

The response is solved by Newton-Raphson iteration. Since the displacements are unknown initially, so are the axial forces. When the first increment of load is applied to the system, a first-order elastic analysis is performed. With this information, the axial loads for this level of displacement and applied load can be determined. Then, the axial loads are used to check the



Note: Process identical to conventional first order analysis, with modifications to trace the nonlinear equilibrium path.

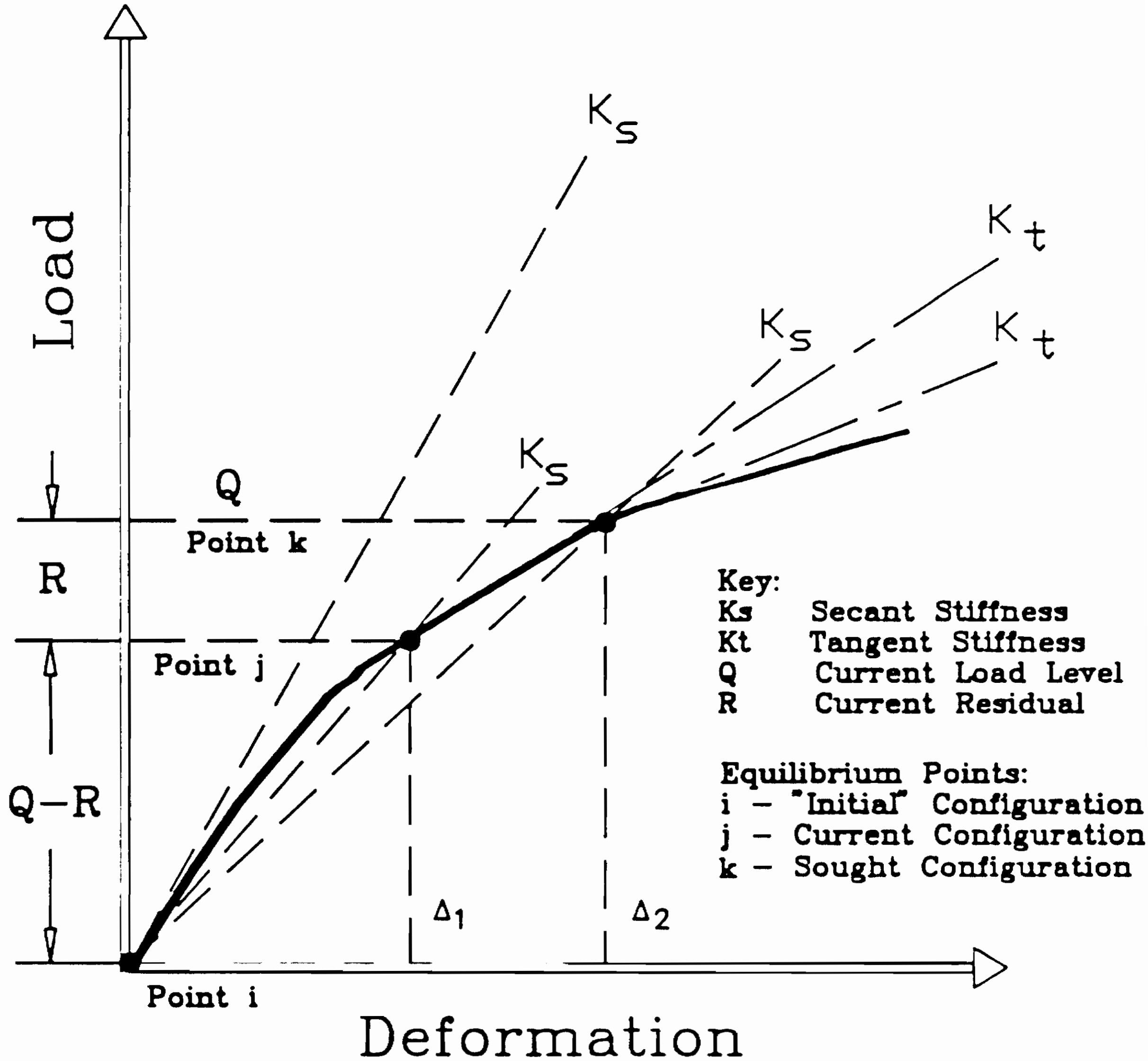
Matrix Displacement Method for Second Order Analysis
Figure 3.9

equilibrium of the structure, and a residual load vector, if necessary, is applied to the structure. Essentially, the nonlinear response is solved by employing a series of linear analyses. This procedure is sometimes called quasilinearization. The process for locating a single equilibrium point is shown in Figure 3.10. A single equilibrium point using the Newton-Raphson solution technique usually requires three or four iterations to locate. Because of the nonlinear response, both accrued load level and accrued displacements must be maintained. Convergence of the process requires two criteria to be satisfied for this Newton-Raphson scheme.

In order to determine if an equilibrium point has been located, two separate criteria must be checked. In a general first-order elastic analysis utilizing the FEM, displacement analysis is a convergence problem. Here, because of the nonlinear equilibrium path, both displacements and the internal forces must converge. This can be demonstrated by two diagrams.

The displacement convergence criteria is required to determine whether or not the structure has deformed an appreciable amount, i.e., enough to warrant applying another iteration. Bathe (1976) and Cook (1989) both recommend a relative displacement target value of 1/1000 be a cutoff point for convergence. This behavior may be visualized by the response of a beam-column under compression and transverse loading. This is a stress-softening phenomenon, as shown in Figure 3.11. It shows that even though the internal/external forces are within a tolerable difference, the increment in displacement is not within tolerances.

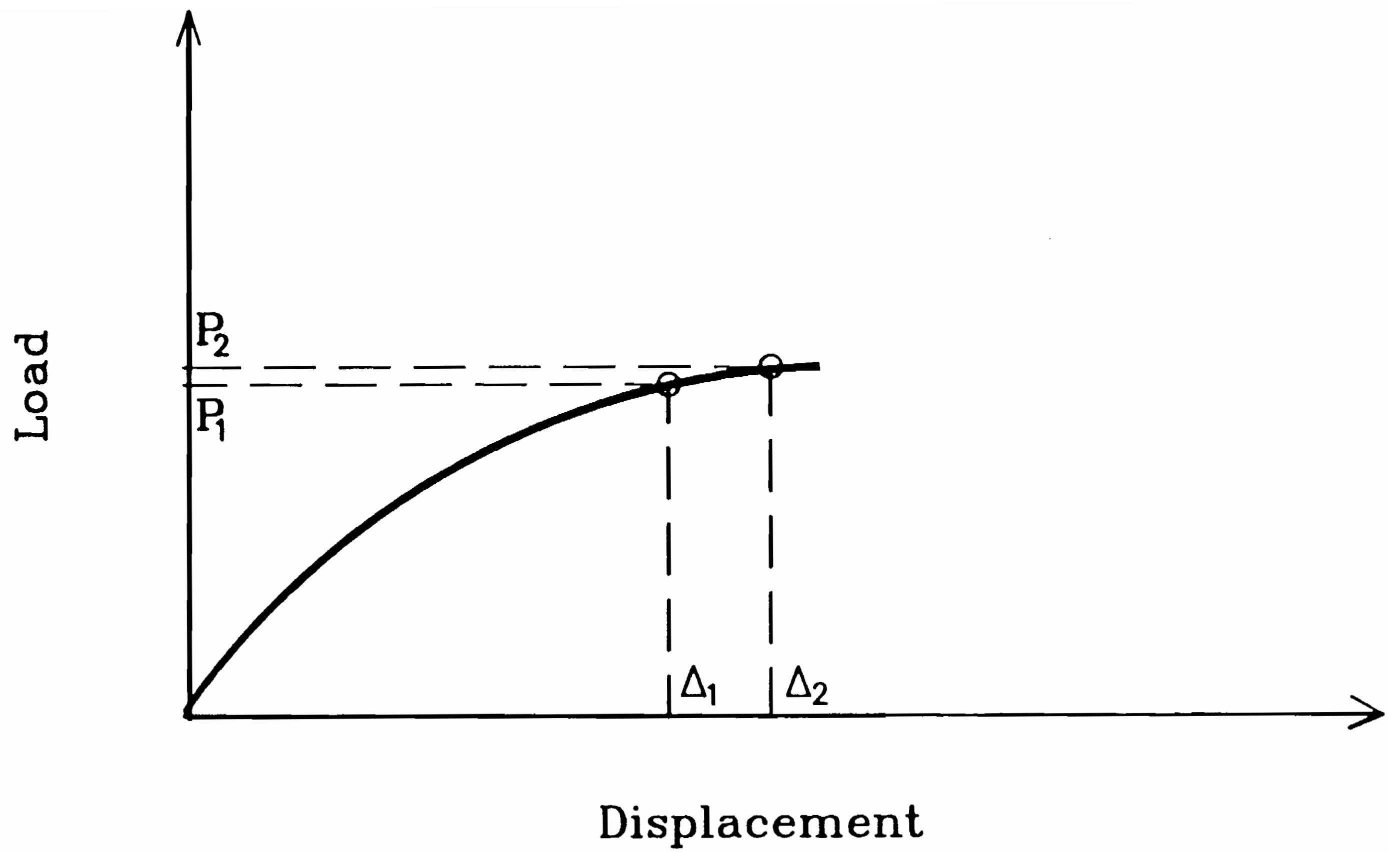
Similarly, force convergence criteria is required to determine whether or not the load imbalance between that stored internally and that applied externally is



Newton Raphson Iteration to Load Level "Q." Tangent and Secant Stiffnesses are formulated at the last equilibrium point found. Convergence is attained when both the increment in deformation and the value of the residual are within their respective tolerances.

Full Newton-Raphson Iteration

Figure 3.10



Stress Softening; Not Converged

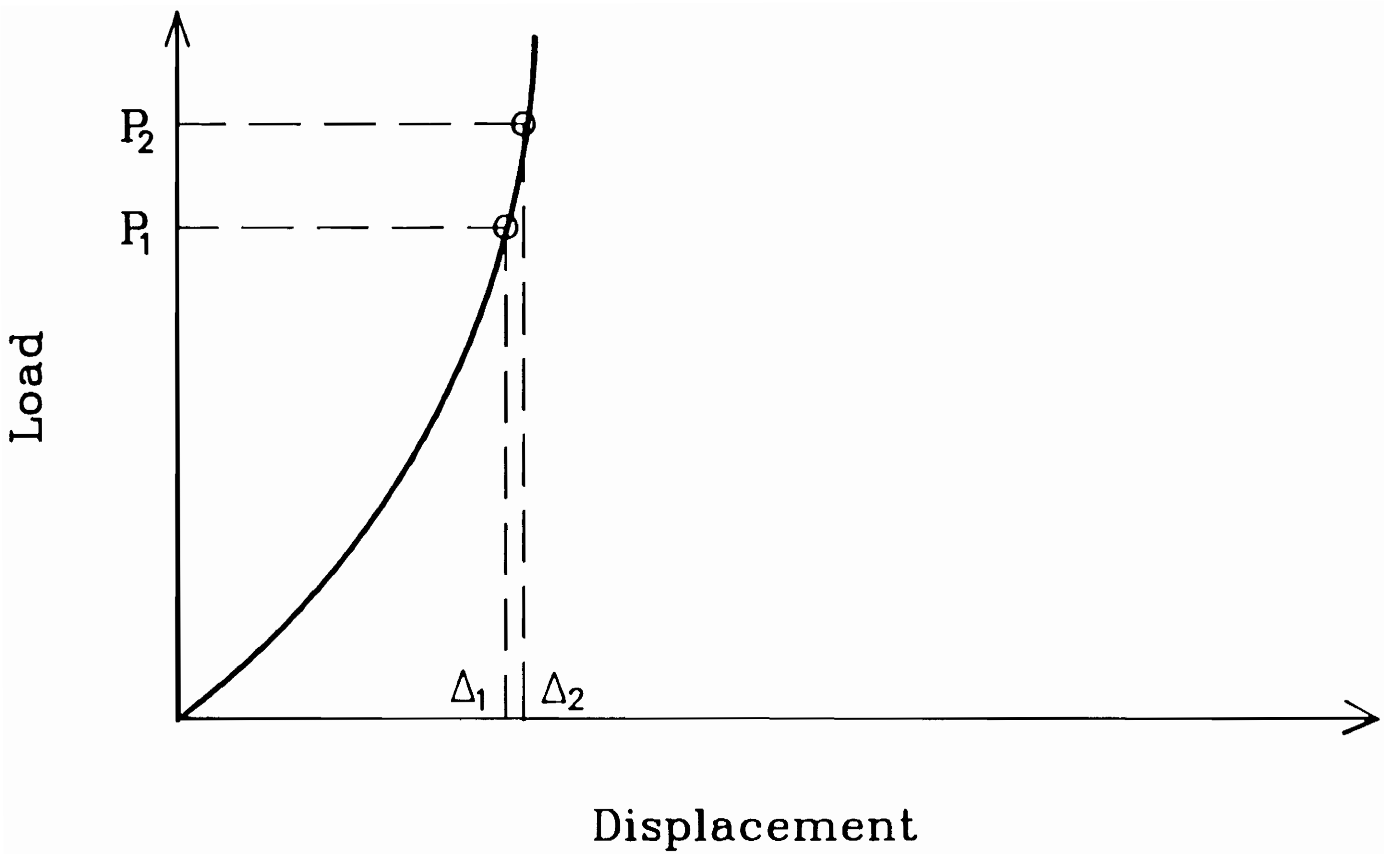
Figure 3.11

enough to warrant another iteration. Bathe (1976) recommends a relative force target value of 0.1 be the cutoff point for convergence. This behavior may be visualized by the response of a beam-column under tension and transverse loading. This is a stress-stiffening phenomenon, as shown in Figure 3.12. This shows that even though the increment in displacement is within a tolerable difference, the imbalance in the load is not within tolerance.

By the above discussion then, it is necessary to satisfy both of these convergence criteria simultaneously. The computer program developed as a part of this study uses these criteria, with a slight change in application. Rather than tracing and forcing convergence at each degree of freedom, those locations accruing the largest incremental deformations or largest load imbalances were compared with the convergence criteria. Because of the interdependence of the displacements and forces, the technique utilized here will bring about sufficient accuracy, such that when an equilibrium point is assumed, the residual vector is essentially zero, and the displacement increments are likewise, zero.

Normally, the number of load increments applied to each of the structures is also a variable, because with fewer updates of the system matrix, the solution may drift from the theoretical solution, and result in the system stiffness being overestimated. This situation is avoided by the tangent/secant stiffness model employed here, because the secant stiffness matrix presented correctly models the internal force-deformation response.

The final variable of these analyses is that of the iterations to be performed. Again, since the loads are of practical concern, the maximum number of iterations to find a single equilibrium point was set at 20. White (1990), states that a practical limit of 10 can be imposed.



Stress Stiffening; Not Converged

Figure 3.12

CHAPTER IV
SECOND ORDER ELASTIC ANALYSIS AND DESIGN

4.1 Introduction

This section of the study focuses on the development of the design equations of the ASD and LRFD Specifications. Although the ASD Specification was not developed formally to be used with second-order elastic analysis, the analyst can judge by analogy the rules to follow. The LRFD Specification always requires second-order elastic analysis of beam-column members. Because of the difficulties of implementing Specification requirements for slender cross-sections, example capacity determinations are provided in Appendix D.

4.2 Allowable Stress Design, First-Order Approach

The use of first-order elastic analysis with the allowable stress design procedure is common practice. The philosophy of ASD is to design elements to resist imposed loads by a margin of safety. This margin, known as a factor-of-safety (FS), varies with the type of applied stress (flexural, axial, shear). For example, the FS for shear on rolled beams is 1.44. This can be back-calculated by using the von-Mises octahedral shear yielding criterion, which defines shear strength to be $F_y/\sqrt{3}$. AISC specifies the allowable stress to be $0.4F_y$. Calculating the ratio of $(F_y/\sqrt{3})/0.4F_y$ obtains the value 1.44. In general, the format can be described by:

$$\frac{R_n}{FS} \geq \sum^{m} Q_m \tag{4.1}$$

The notation of Equation 4.1 is: Rn/FS = nominal capacity / factor-of-safety, and Σ Qm = sum of the service load effects. Or, essentially that the section strength, divided by a margin of safety, must be greater than the effects caused by the expected loads. Notice that "safety" is measured in terms of the expected service load range.

Moving onto specifics, two interaction equations must be satisfied for a compressive beam-column member. For tensile beam-columns, only one of the equations need be checked. These equations are known as the stability and yield interaction equations. Note here that the yielding equation is somewhat of a misnomer, as the flexural term of this equation can sometimes be based on lateral-torsional buckling or local-buckling.

The ASD Stability Interaction Equation (1978 or 1989) is

$$\frac{f_a}{F_a} + \frac{C_m}{\left(1 - \frac{f_a}{F'_e}\right)} \frac{f_b}{F_b} \leq 1.0 \tag{4.2}$$

and the ASD Yielding Interaction Equation (1989) is

$$\frac{f_a}{0.6QF_y} + \frac{f_b}{F_b} \leq 1.0 \tag{4.3}$$

The stability interaction ratio (compression only) is based on the axial capacity and flexural strength of the section as though failure was imminent, according to Salmon and Johnson (1989). This means that the column strength used in the equation is the absolute design maximum value of stress that the section can safely resist. This should be emphasized because of a slender member design problem, discussed later. The effective-length of the column is chosen by a rational method and is an inconvenient though necessary evil of first-order elastic design. The flexural strength is developed as though the member were subjected to purely flexural stresses.

Emphasized again, Fa is the absolute design maximum value of the compressive strength at effective-length and effective-cross-section. For prismatic, rectangular structures, the effective length factor [K] is usually determined by the use of Julian and Lawrence's effective-length factor nomographs (AISC 1989). For tapered members, Lee (1972) developed the effective-length nomographs using the Rayleigh-Ritz procedure and infinite series solution. The development of the nomographs is beyond the scope of this study. NUCOR assumes that the in-plane effective-length factor for tapered columns is 1.5. They also assume that the in-plane effective-length factor for tapered rafters is 1.95. Other means are available to estimate this factor, and are considered in Chapter VI. The factor-of-safety in the stability interaction equation is approximately 1.67 against elastic instability.

For the yielding interaction ratio, the factor-of-safety in this equation is approximately 1.67 against yielding of the extreme fibers of the cross-section.

4.3 Allowable Stress Design, Second-Order Approach

The philosophy of these design rules is based partly on criteria suggested by SOEA researchers Adams (1976), LeMessurier (1977), and Smith (1988). The design interaction equations are identical in format with those of the first-order design rules, with some modifications of the terms involved.

Practically speaking, the amplification of moments in the stability interaction equation provides an estimate of the second order effects. Logically then, if frame and member second-order results were accounted for directly at the analysis stage, the amplification factor could be eliminated in this equation. The extension suggested by Adams is that the effective length factor may be taken as 1.0 for the interaction check. White (1990) suggests that Kx = 1.0 is only applicable when fa/Fa < 0.85, and Yura (1987) suggests that Kx = 1.0 may be applicable when P_u/P_y < 0.5, as noted previously. Assuming that these criteria are met, the incrementally-applied second-order ASD stability interaction equation is given by:

$$\frac{f_a}{F_a} + \frac{f_b}{F_b} \leq 1.0 \tag{4.4}$$

where f_b and F_b are respectively, the calculated second-order moment and allowable moment at specific locations under investigation. When used in this format, it intuitively compares with the LRFD interaction equations, shown below as equations (4.6) and (4.7).

The difference in the yielding interaction equation is that the second-order elastic moment is used in the interaction check, rather than the first-order elastic moment. Though the first-order ASD yielding interaction equation uses first-order elastic moments, it was thought to be inconsistent with the development here, so the calculated second-order moments are used instead of the first-order moments.

The proposed benefit of an ASD second-order analysis and design philosophy is in the stability interaction equation. The increase in the flexural term by SOEA may be larger than the increase predicted by the first-order ASD amplification factor, and could possibly offset the accompanying decrease in the axial term. It is noted here that if the out-of-plane slenderness ratio controls the column strength, there may not be a benefit in using second-order elastic analysis, since the effective-length factor cannot be modified. If out-of-plane slenderness controls, the only possibility of benefit is that the actual second-order moments may be smaller than those obtained by the first-order ASD amplification factor analysis. Under the current first-order design rules, if the yielding ratio is governing, the use of SOEA only results in higher moments under axial compression. Therefore, no benefit is possible. If an SOEA converges, the structure is not unstable elastically, but may still be unstable inelastically. The stability interaction equation is really an inelastic stability interaction equation, but is more commonly called the stability interaction equation.

4.4 Load & Resistance Factor Design Approach

Load & Resistance Factor Design requires the use of second-order elastic analysis results. Also, the provisions of the LRFD Specification were developed assuming factored loading. The format of LRFD is to design structural systems to resist maximum expected lifetime loads, hence the need for factored loads. The load factors and material reliability factors were determined by probabilistic analysis. Load factors were probabilistically determined according to measured maximum values, and the probability of different types of loadings occurring at their maximum values is considered in the load combinations. Material strength reduction factors are determined based on observed statistical data on specified versus delivered section properties. This philosophy results in a more uniform reliability of the structure under different load combinations. The LRFD Specification attempts to obtain a uniform reliability over the entire structural system.

In general, the philosophy of the design approach is,

$$\phi R_n \geq \sum^{m} \gamma Q_m \tag{4.5}$$

Where ϕ = strength reduction factor, R_n = nominal strength, Σ = summation over *m*, γ = load factor, Q = service load effect *m*. Or, essentially that the nominal section strength multiplied by a strength reduction factor must be greater than the effects caused by the sum of the factored loads. Notice that safety here is measured in terms of a factored load range. One stability-based bilinear interaction equation must be satisfied for a beam-column member. The portion of the design curve to be followed is determined according to the axial force ratio.

The 1986 LRFD Specification interaction equations are:

For $P_u/\phi P_n \geq 0.2$

$$\frac{P_u}{\phi P_n} + \frac{8}{9}\frac{M_u}{\phi M_n} \leq 1.0 \tag{4.6}$$

For $P_u/\phi P_n < 0.2$

$$\frac{P_u}{2\phi P_n} + \frac{M_u}{\phi M_n} \leq 1.0 \tag{4.7}$$

The significance of the first term of the interaction ratio is the same as that of the ASD stability interaction equation. All unity strength checks are performed with the denominators (strengths) at their nominal capacities. The nominal capacities are determined based on a limit-states analysis approach and represent the strength that can be obtained, as in a laboratory experiment under controlled conditions. This strength is then reduced by ϕ factors, which are based on tolerances of the available section strength.

When compared with the alternate interaction equations (braced frames only) in Appendix H of the 1986 LRFD Specification, Equations 4.7 and 4.8 seem overly conservative for all values of column slenderness ratios. Though not specifically stated, it is believed that the Appendix H equations are to be used only when plastic strength is attainable.

4.5 Comparison of Specification Provisions

The following discussions concern the nominal strengths of cross-sections, neglecting the *load effect* side of the total design equations.

4.5.1 Flexural Strengths

For the plastic region of flexural strength, LRFD utilizes the correct plastic moment, not an assumed 10% increase from the moment which causes first-yield on compact cross-sections. The cross-sections of NUCOR frames are rarely capable of developing a fully-plastic moment, as the compact section criteria are usually violated. Though LRFD utilizes a higher strength in this region, it is generally unavailable to typical metal buildings.

For the lateral-torsional buckling strength region, although the strengths in LRFD directly account for both the warping and St. Venant torsional resistance terms, ASD permits use of the vector sum of these individual allowable flexural stresses. A modified r_T (r_T is normally the weak-axis radius of gyration of a portion of the cross-section, comprising the compression flange and 1/3 of the compression web area) must be calculated for this purpose. Therefore, the difference between the two specifications has been eliminated in this respect. The modified r_T equation is shown in Chapter VII.

LRFD has separate limit state strengths for flange local buckling (FLB) and web local buckling (WLB). However, ASD does not have a comparable WLB provision. This is an ASD benefit, but WLB can still occur even though it is not a recognized concern of the ASD Specifications.

LRFD was calibrated to produce similar results as a first-order ASD produces at an L/D ratio of 3.0. However, this calibration seems to apply only when dead and live load are involved. When wind loadings are included, LRFD does not recognize a 1/3 strength increase.

The lower curve of Figure 4.1 describes a first-order elastic behavior required strength comparison between ASD and LRFD for tensile or flexural members (with noncompact section), varying the L/D ratio. Values on the graph less than 1.0 indicate economy in favor of LRFD, and values greater than 1.0 indicate economy in favor of ASD. This plot is only for the dead+live load combination. The lower plot describes a first-order comparison of the results to

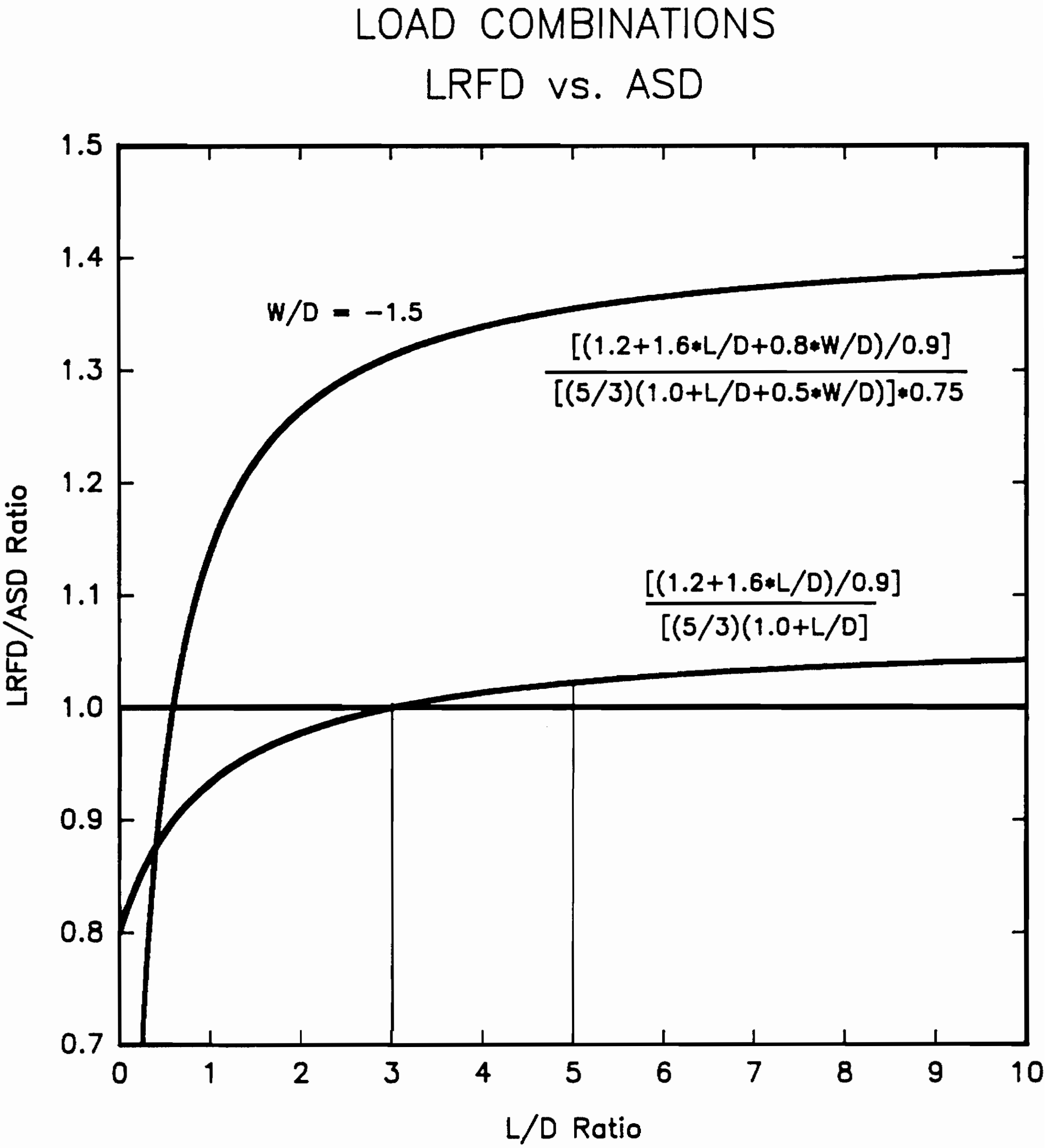

LOAD COMBINATIONS
LRFD vs. ASD
LRFD/ASD Ratio
L/D Ratio
W/D = −1.5
[(1.2+1.6*L/D+0.8*W/D)/0.9]
[(5/3)(1.0+L/D+0.5*W/D)]*0.75
[(1.2+1.6*L/D)/0.9]
[(5/3)(1.0+L/D]
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0
1
2
3
4
5
6
7
8
9
10

Figure 4.1

be expected when comparing ASD with LRFD. Consider a tension member or a noncompact flexural member to be the member being designed. As the graph shows, at a L/D ratio of 5.0, LRFD requires a section with 2.2% higher strength.

However, this situation is only part of the whole story. This 2.2% required increase is not typical, as the upper plot describes. When other ASD load combinations are compared, such as dead+snow+wind/2, ASD allows a 1/3 stress increase, whereas LRFD does not, since it is based on a philosophy of maximum expected lifetime loads. The second plot in Figure 4.1 shows a pure flexure or pure tension member with dead, live, and wind loads. The wind load is held constant here (uplift is assumed), and there are no second-order effects considered. Notice that the curve approaches a value of 1.33, indicating that ASD is more economical by nearly 33% over an LRFD approach for this range. However, because of the magnitude of loads and geometry of the typical NUCOR frame, the controlling ASD combination is usually D+L, and in LRFD it is either 1.2D+1.6L or 1.2D+1.6L+0.8W.

For weak-axis flexure of H cross sections, LRFD provides a great deal of strength compared to that of the ASD. This is due to the assumption of full plastification about the weak-axis. Unfortunately, weak-axis bending is not utilized for the primary load-resisting elements of these rigid frame gabled structures.

4.5.2 Axial Compressive Strengths

The equations governing compressive strength of noncompact sections in ASD and LRFD essentially follow the same capacity curve, although LRFD assumes a slightly different residual stress distribution of $0.44F_y$ instead of $0.50F_y$. This has the effect of allowing it a larger inelastic strength zone. However, when the LRFD closed-form tapered member provisions are followed, the inelastic/elastic cutoff point occurs earlier, (lower column slenderness ratios) indicating a larger elastic zone. Whenever the elastic zone is used, the strength is smaller than that achieved through inelastic action.

When slender cross-sections are involved, LRFD also possesses greater strengths, especially when compared to the 1989 ASD Specification. This situation arises from the fact that the Q_S factors in the new ASD Specification drop off sharply due to the flange/web local buckling interaction formulas.

4.6 Future Specification Modifications

As seen in Figure 4.1, inconsistencies with LRFD need to be worked out. There are other problems with the 1986 LRFD Specification, including but not limited to shear strength. For compact/noncompact sections compared with ASD, the LRFD shear strength is approximately 10% lower (Wishart 1990), and the strength equation may be changed in future editions of this Specification. The strengths are nearly identical for slender-web sections. Other potential problems involve the tapered member design provisions (though these are not typically followed by the industry). It is believed that the metal building industry follows the prismatic beam provisions, developing the section strengths at incremental locations along tapered members. This procedure is also followed in this research.

Further, it was recognized that research sponsored by Butler Manufacturing concerning flange/web local buckling that found its place in the 1989 ASD Specification was not incorporated into the LRFD Specification. Because of the omission, there exists an envelope of economy in favor of LRFD. Under pure flexure, and an L/D ratio of 5, there is an approximate 12% increase in flexural strength, depending on flange and web proportions. This can be visualized by Figure 4.2. The writer believes that either this 1989 ASD Specification criterion is incorrect, or the 1986 LRFD Specification is incorrect for this range. This should be studied further for applicability.

The upper graph of Figure 4.2 represents the available allowable stresses (ASD) and the reduced nominal capacities (LRFD) according to each of the design equations (labelled on the graph for convenience). FLB, WLB, and PG are abbreviations for flange local buckling, web local buckling, and plate girder, respectively. The graphs were constructed for a beam with 7 in. x 0.3125 in. flanges, 0.2060 in. thickness web, Fy = 50 ksi, and welded plate construction. The depth of the section was varied in order to trace the response that a tapered member would provide. Lateral-torsional buckling is assumed to be prevented.

According to the 1978 ASD Specification and the 1986 LRFD Specification, the section is partially compact up to h/t = 138. By the 1989 ASD Specification however, the flanges are partially compact up to an h/t value of 70. Past this point, the flange slenderness local buckling limit is dependent on the web h/t ratio. With this difference, the allowable stress then follows a transition curve from 0.63Fy to an intersection point somewhere on the plate girder design curve. The 0.63Fy is obtained from the provisions of F2 (1989 ASD), since the section possesses a noncompact flange and compact web in this region of h/t.

An economic envelope was found to exist in the region between this new ASD curve and the LRFD curves. As an example, assume the following:

- Live + Dead load combination controls.
- L/D = 5.
- h/t = 120.

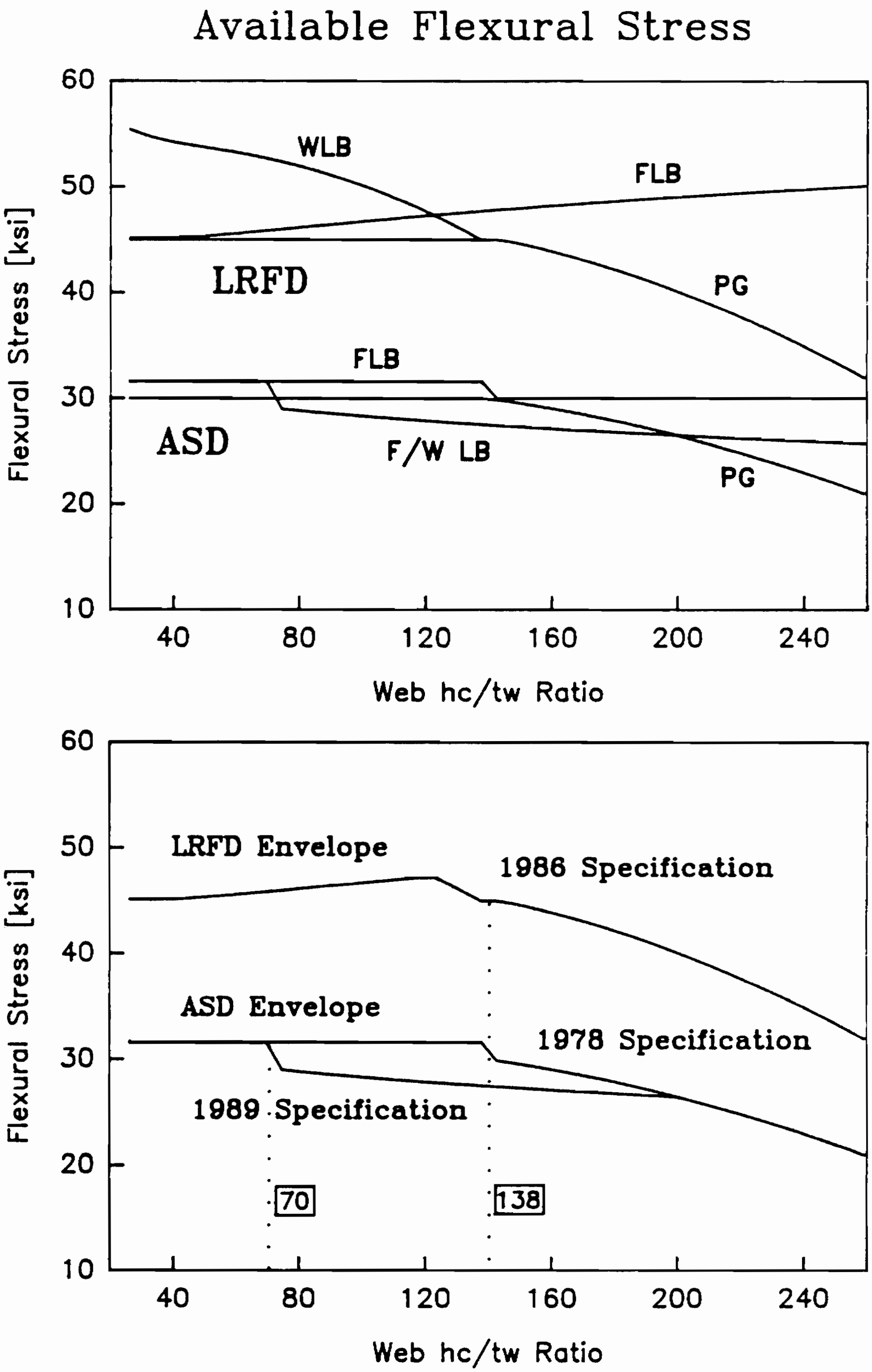

Available Flexural Stress
60
50
40
30
20
10
Flexural Stress [ksi]
WLB
FLB
LRFD
PG
FLB
ASD
F/W LB
PG
40
80
120
160
200
240
Web hc/tw Ratio
60
50
40
30
20
10
Flexural Stress [ksi]
LRFD Envelope
1986 Specification
ASD Envelope
1978 Specification
1989 Specification
70
138
40
80
120
160
200
240
Web hc/tw Ratio

Figure 4.2

Design stresses from the plots are:

1978 ASD: F_b = 32 ksi

1989 ASD: F_b = 27 ksi

1986 LRFD: ϕF_n = 47 ksi

Comparing the design stresses (unfactoring the LRFD effect):

1986 LRFD versus 1978 ASD:

$[47 \text{ ksi}]/[32 \text{ ksi}]/[(1.2+1.6(5))/(1+5)] = 0.96$;

therefore ASD is economical by 4%

1986 LRFD versus 1989 ASD:

$[47 \text{ ksi}]/[27 \text{ ksi}]/[(1.2+1.6(5))/(1+5)] = 1.14$;

therefore LRFD is economical by 14%

This difference of 14% is applicable only under pure flexural loading. When second-order beam-column effects are included, the economic envelope gets smaller, but is still greater than approximately 3%.

This omission will be under study by the LRFD Specification Committee for possible inclusion in the 2nd or 3rd Edition. However, it is not on the current agenda for the 2nd LRFD Specification, indicating that this envelope will be available for at least four more years. Also, the AISC had taken the position that the flange/web local web buckling interaction would not have much affect on the metal building industry. The writer feels that differences of approximately 10% are a serious consequence, and that the range of applicability of the ASD equation be researched further. With this current criteria, a significant portion of flexural strengths for slender- flanged cross-sections will fall into this category.

CHAPTER V
CASE STUDIES

5.1 Introduction

This chapter focuses on the results of applying the SOEA rules to the frame geometries submitted by the sponsor. There are three alternative design calculations for each frame, not including first-order ASD results.

Each of the 10 frames is analyzed for 16 ASD load combinations, and the corresponding 16 LRFD load combinations. The results of the analyses are then input to specification checking programs and the maximum interaction ratios are tabulated. The relative economy of each of the three alternatives is then compared by calculating the average percent difference between the first-order ASD ratios and each of the three alternatives.

5.2 Method of Comparison

For all the frames studied, the results are compared to the first-order analysis results. For each element of each frame, the maximum interaction ratio from its governing limit conditions is calculated. These limits may occur over one or more load combinations, i.e., stability under load combination 1, yielding under load combination 6, shear under load combination 8, etc. Generally, for symmetric gable frames, the first-order interaction ratios are controlled by the D+S and D+S+W/2 load combinations. Under LRFD, the governing load combinations are 1.2D+1.6S and 1.2D+1.6S+0.8W. Summary graphs of each frame and the design proposal interactions are found in Appendix A.

The economic comparisons are obtained by dividing the individual element interaction ratios by their first-order interaction ratios. By this method, multiples of the controlling first-order interaction ratios are found.

As an example, assume that a theoretical comparison is being performed between ASD and LRFD. Further assume that the LRFD strength interaction ratio is calculated to be 0.90. Likewise, assume that a first-order ASD is performed on the same member, and the governing interaction ratio is calculated to be 0.95. Normalizing the results,

$$LRFD/ASD = 0.90/0.95 = 0.9474$$

gives the multiple of the first-order interaction for LRFD vs. ASD. For this example then, LRFD is more economical than ASD by an amount:

$$(1.0-0.9474)*100\% = 5.26\%$$

When comparing ASD with LRFD, there can be confusion between strength comparisons and load effects comparisons because of the load and strength factoring. Comparing normalized interaction ratios eliminates any confusion.

Economy is provided by those multiples that are less than 1.0. These multiples are then analytically assessed as to their relative economies on a percentage basis. This was accomplished by defining a spreadsheet for each frame, listing the maximum interaction values for each of the analysis types. With this information, the multiples of the first-order ASD ratios are easily found. Graphs were then created in order to facilitate a quick data presentation. Specification checking computer programs written to accept the results of the analysis program were implemented here. The maximum interaction ratio of the selected cross-sections of each frame was extracted by these programs. The spreadsheets were also used to develop the average increase or decrease in the responses.

5.3 Economic Criterion

Economy, as used in this study, is defined as an available increase in design strength of at least 3%. It was assumed that an increase in design strength smaller than 3% would not justify the increase in analysis time required by second-order analysis.

5.4 Preliminary Data Cautions

Originally, the frames were designed under the 1978 ASD Specification. Due to the availability of the 1989 ASD Specification and its amendments to slender cross-section design, this research attempted to investigate frame integrity according to this new ASD Specification. The writer believes that there may be some confusion in application of slender cross-section design rules. To assist in alleviating this confusion, the provisions for slender cross-sections are discussed in Appendix D.

At various locations along the frames, the first-order design interaction ratios are considerably greater than 1.0. This is primarily due to four causes. They are:

- The flange/web local buckling interaction formula of the 1989 ASD Specification was included for flexural strengths;
- Compression allowable stresses were calculated for the condition of imminent failure (Salmon and Johnson 1989), iterating with $f=F_a$ when slender-web cross-sections are involved;
- Inclusion of the form-factor Q in denominator of the axial term of the 1989 ASD yielding interaction formula;
- Investigation of unbalanced snow load distributions, which were not originally investigated.

With the intent of satisfying the research proposal requirements of assessing P-Delta effects and critiquing LRFD of Gable Frames, three alternate design procedures (proposals) are defined as follows:

- The first alternate is a second-order elastic analysis and design check according to the Allowable Stress Design rules developed in Chapter 4, Section 3. This is identified in the individual discussions by the notation ASD\so.
- The second alternate is a second-order elastic analysis and design check according to the Load and Resistance Factor Design rules developed in Chapter 4, Section 4. This is identified in the individual discussions by the notation LRFD.
- The third alternate is a first-order elastic analysis and design check according to the Allowable Stress Design rules developed in Chapter 4, Section 2,

but does not use the effective-length factors that NUCOR usually assumes. According to elastic stability analyses and the tapered-column effective-length factor charts found in the Commentary to the ASD Specification (AISC 1978, AISC 1989), reduced (in-plane) effective-length factors were found to be available. They were K_X = 1.3 (columns) and K_X = 1.85 (rafters). Their development is the subject of Chapter VI. This is identified in the individual discussions as ASD\p.

Also, ASD\fo denotes the ASD first-order design, according to the assumed NUCOR K_X values.

In Appendix A, these results of each of the alternate design checks are plotted over their location on each of the frames. All interaction ratio multiples are measured with respect to the ASD\fo behavior.

5.5 Frame Discussions

The designation of each of the frames has the following format:

aWIDE **b**EAVE **c**/12 **d**PSF

Where a = total width of building frame (ft.), b = vertical height of the eave (ft.), c = roof slope from the eave to the gable point, as a function of 12 in., and d = the value of the snow load (psf.) applied to the horizontal projection of the frame.

The horizontal axes of the graphs in Appendix A represent cross-section locations along the frame. These locations start at the base of the left column and traverse the frame to the right column.

5.5.1 Frame #1; Designation: 50WIDE 30EAVE 2/12 30PSF

Symmetric, Clear-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 15.9 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. From this point, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.18, and the maximum capacity ratio is 1.19. Since the stability interaction ratio controls only the lower half of the column, there will be no benefit from second-order analysis rules.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 3%. The average increase in the capacity equation is 3%. Therefore, no benefit is achieved using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 20%. LRFD has no potential benefit here. The use of factored loads outweighs the benefits of an in-plane Kx = 1.0.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 5%, and the yielding interaction ratios increase by less than 1%. This indicates that this is a viable alternative. The 5% increase in design strength is directly achievable, without SOEA. The slight increase in the capacity equation is due to the decrease in the form-factor, Q. The form-factor decreases due to the increase in the column allowable stress.

5.5.2 Frame #2; Designation: 60WIDE 20EAVE 1/12 30PSF

Symmetric, Clear-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 14.2 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. Then, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.16, and the maximum capacity ratio is 1.18. As in Frame #1, since the stability ratio is not critical along most of the frame, there will be no benefit in using second-order ASD rules.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 10%. The average increase in the capacity equation is 1%. There is no foreseen benefit in using the modified second-order ASD rules for this frame.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 4%. The use of factored loads outweighs the benefits of an in-plane $K_x = 1.0$. Therefore, LRFD has no benefit here.

D. ASD\p versus ASD\fo Results. The stability interaction ratios decrease by less than 1%, and the yielding interaction ratios increase by less than 1%. Since yielding is the governing criterion, no benefit is attainable by strength increases in the stability equation.

5.5.3 Frame #3; Designation: 60WIDE 35EAVE .5/12 30PSF

Symmetric, Clear-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 16.7 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. Then, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.05, and the maximum capacity ratio is 1.11.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 8%. The average increase in the capacity equation is 1%. Therefore, no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 18%. Therefore, LRFD has no benefit here.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 2%, and the yielding interaction ratios increase by less than 1%. There is no benefit obtainable with this alternative, by the cutoff value assumed, and the fact that the capacity equations controls along most of the structure.

5.5.4 Frame #4; Designation: 80WIDE 40EAVE 2/12 30PSF

Symmetric, Clear-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 17.3 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. Then, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.05, and the maximum capacity ratio is 1.12.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 5%. The average increase in the capacity equation is 2%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 10%. Therefore, LRFD has no benefit here.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 3%, and the yielding interaction ratios increase by less than 1%. This is a beneficial case, directly obtainable without second-order elastic analysis.

5.5.5 Frame #5; Designation: 120WIDE 25EAVE 1/12 30PSF

Symmetric, Clear-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 15.3 psf

A. ASD\fo Results. The stability interaction ratio does not control design of the columns. The capacity interaction ratio controls most of the frame, except for isolated sections of the rafters. The maximum stability interaction ratio is 0.97, and the maximum capacity ratio is 1.12.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 5%. The average increase in the capacity equation is 2%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equations results in a relative decrease of the ratio of 2%. For this situation, LRFD does not support the economy criterion of at least 3%. The use of factored loads and second-order analysis outweighs the increased flexural capacities available in LRFD.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 1%, and the yielding interaction ratios increase by less than 1%. There are no potential economic benefits with this proposal.

5.5.6 Frame #6; Designation: 100WIDE 30EAVE .25/12 30PSF

Symmetric, Multi-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 15.9 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. From this point on, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.82, and the maximum capacity ratio is 1.95. These are considerably greater than 1.0 primarily because of the effects of the unbalanced snow load combinations, and to a lesser extent, the 1989 ASD Specification slender provisions.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 6%. The average increase in the capacity equation is 4%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 35%. Therefore, LRFD has no benefit here.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 3%, and the yielding interaction ratios increase by less than 1%. This is a borderline case, obtainable without second-order analysis.

5.5.7 Frame #7; Designation: 120WIDE 20EAVE .5/12 40PSF

Symmetric, Multi-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 40 psf Wind: 14.2 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. From this point on, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.33, and the maximum capacity ratio is 1.37. As in Frame #6, these are considerably greater than 1.0 primarily because of the effects of unbalanced snow load combinations, and to a lesser extent, the 1989 ASD Specification slender provisions.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 7%. The average increase in the capacity equation is 1%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in an increase of the ratio of 10%. LRFD has no available benefit for this frame.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 2%, and the yielding interaction ratios increase by less than 1%. The benefit is not above the economic cutoff value stipulated, but is available directly without second-order analysis.

5.5.8 Frame #8; Designation: 180WIDE 35EAVE 1/12 30PSF

Symmetric, Multi-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 16.7 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. Then, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.18, and the maximum capacity ratio is 1.30.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 7%. The average increase in the capacity equation is 2%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equations results in an increase of the ratio of 26%. LRFD has no benefit here.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 2%, and the yielding interaction ratios increase by less than 1%. Again, the benefit is not greater than the cutoff value stipulated, and the capacity equation is generally the governing ratio, but second-order analysis need not be used to obtain the decrease in the stability interaction ratios.

5.5.9 Frame #9; Designation: 250WIDE 50EAVE .25/12 40PSF

Symmetric, Multi-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 40 psf Wind: 18.4 psf

A. ASD\fo Results. The stability and capacity interaction equation values are nearly identical along the entire frame, resulting in a "balanced design" envelope. The maximum stability interaction ratio is 1.20, and the maximum capacity ratio is 1.30.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 6%. The average increase in the capacity equation is 2%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a decrease of the ratio of 2%. Though capacities have increased, LRFD has not exceeded the designated economic criterion.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 3%, and the yielding interaction ratios increase by less than 1%.

5.5.10 Frame #10; Designation: 80WIDE 40EAVE 2/12 30PSF

Symmetric, Multi-Span Gable Frame

Design Loads; Dead: 5 psf Snow: 30 psf Wind: 17.3 psf

A. ASD\fo Results. The stability interaction ratio controls design of the columns up to approximately midheight. Then, the capacity interaction ratio becomes critical, and remains critical along the rafters. The maximum stability interaction ratio is 1.05, and the maximum capacity ratio is 1.20.

B. ASD\so versus ASD\fo Results. The average increase in the stability equation is 12%. The average increase in the capacity equation is also 12%. Therefore, there is no benefit in using the modified second-order ASD rules.

C. LRFD versus ASD\fo Results. The average LRFD interaction equation versus the ASD\fo stability equation results in a relative increase of the ratio of 33%. Therefore, LRFD has no benefit here. The second-order effects under factored loads outweigh the benefits of an in-plane K_x = 1.0.

D. ASD\p versus ASD\fo Results. For this proposal, the stability interaction ratios decrease by 2%, and the yielding interaction ratios increase by less than 1%. No real benefit is obtained since the yielding equation controls along most of the frame.

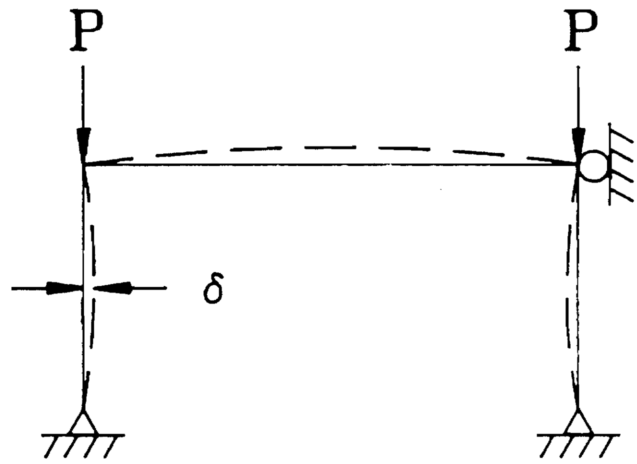
CHAPTER VI
ELASTIC STABILITY ANALYSIS OF TEST FRAMES

6.1 Introduction

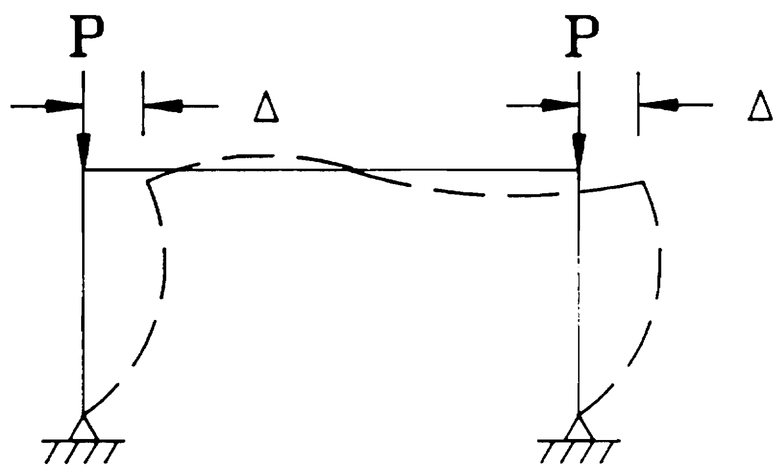
This section describes methods available to determine the effective-lengths of columns in braced or unbraced frames. All of these methods are accepted as fitting the rational method described in the AISC Specifications. This study was performed in order to find an alternative first-order effective-length factor for in-plane stability checks under allowable stress design because the second-order elastic analysis and design proposals failed to produce economy according to the set criteria.

Elastic stability is esoterically defined as a point at which a small lateral disturbance causes a large increase in displacements. This point is identified as the elastic stability limit point. The inclusion of shear effects in stability analysis results in lower buckling loads, but involving them in the analysis is more rigorous than required. The shear effect is normally only pronounced when deep beams under large shears are concerned.

From the deformational behavior in an elastic stability mode, the role transverse deformations play in creating the second-order elastic member and frame effects can easily be seen. Figure 6.1 shows the lowest energy level buckling configurations of typical rectangular, prismatic frames, both braced and unbraced. Note that for these frames, the first-order effective-length factor of



Sidesway Prevented



Sidesway Not Prevented

Elastic Stability Modes
of Rigid Frames

Figure 6.1

columns in braced frames is less than or equal to 1.0, and for the unbraced frame, the effective length factor is always greater than or equal to 1.0.

Research on the inelastic behavior of framed structures is being performed in an attempt to eliminate the effective-length method. At this time there is no generally accepted practical method available to replace the effective-length method. When a column is part of a framed system, the restraint provided by the varying end conditions and intermediate lateral bracing needs to be accounted for in the buckling load. For this reason, the concept of effective-length was initiated.

Accounting for the translational and lateral restraint provided by the interconnecting framework, the analyst may calculate a dimensionless effective length factor, K. This factor is determined by equating the strength of an equivalent length, pinned-end column, with the actual buckling load. The K-factor is then calculated by:

$$K = \sqrt{\frac{P_e}{P_{cr}}} \qquad (6.1)$$

Where P_e = Euler Buckling Load, P_{cr} = critical buckling load, and K = effective-length factor. Simplified methods to find this factor have been published, and are widely available. The determination of this factor can vary from using a simple nomograph for prismatic members (Julian and Lawrence 1962) or complicated mathematical analyses (eigenvalue, second-order elastic). Obviously, for the purposes of design, the quickest and most practical method needs to be chosen. The methods available and their practicality are discussed below.

6.2 Current Methodology

For prismatic, rectangular structures, the effective lengths of columns in frames are normally determined using the effective-length factor nomographs provided in the AISC steel construction manuals. The method involves calculating approximate factors that account for rotational restraint at the column ends. These factors are then used with the alignment charts, and the approximate K-factor is found. There are many underlying assumptions imposed by the nomograph method, many of which are violated for typical building frames. These assumptions are:

1) Behavior is purely elastic,
2) All members have constant cross-section,
3) All joints are rigid,
4) For braced frames, buckling causes a symmetric, single curvature in the connecting beams,
5) For unbraced frames, buckling causes a anti-symmetric double curvature in the connecting beams,
6) The stability parameter $L\sqrt{\frac{P}{EI}}$ of all columns are equal
7) Joint restraint is shared by distributing the I/L values between the columns,
8) All columns in a story buckle simultaneously.

For the symmetric NUCOR type frames, assumptions 2 and 5 are obviously violated. Elastic behavior is usually present as they primarily store flexural energy. The inelastic K-factor procedure developed by Yura (1971), is not applicable. There is also the question of assumption 7, since the rafter members are sloped and tapered.

Because of the violations when tapered members are considered, Lee (1974), set out to provide effective-length alignment charts for tapered columns. The analytical method of the calculations are quite complex and are not repeated here. They were derived using stiffness equations and the Rayleigh-Ritz method with tapered beam-column elements, which were solved with infinite series summations. It is not practical to calculate the K-factors in this manner.

Other methods to determine critical buckling loads are suggested by the finite element method. There are two such procedures most often utilized. The first is through an eigenvalue analysis, which involves calculating the lowest positive value of the column loads that result in a non-positive definite structure stiffness matrix. This can be a computationally expensive tool. One method of solving for the eigenvalues requires expanding the structure stiffness matrix by minors, and determining the lowest root of the characteristic equation. A typical discretized frame may have 500 degrees-of-freedom, for which there are 500 roots to be found. Another method of eigenvalue analysis is performed by a Jacobian method. This method is performed by a computer, and the lowest eigenvalue is extracted in the solution process. It can provide a quick answer for individual columns, but not when they are part of a system. Both of these methods are time-consuming, and are of little practical significance for design.

6.3 Second-Order Elastic Method

For the present research, a second-order elastic method in finding the elastic buckling loads was employed. In this method, the same tangent/secant stiffness method described earlier was applied here. An upper-bound estimate of the buckling load was calculated and applied to each of the structures incrementally. As the solution progresses, the frame response is monitored until the frame drifts to a specified amount, indicating instability.

In this research, a displacement criterion was used to determine buckling. This type of analysis requires that the vertical loads are placed perfectly on the columns, and a small lateral disturbance is introduced during each cycle of calculation. This small disturbance is introduced by specifying a lateral force applied to one of the column tops. As an estimate of the magnitudes of these loads, a typical frame upper-bound buckling load was chosen as 200 kips. The small disturbance (which is also applied incrementally) specified was 10 lbs.

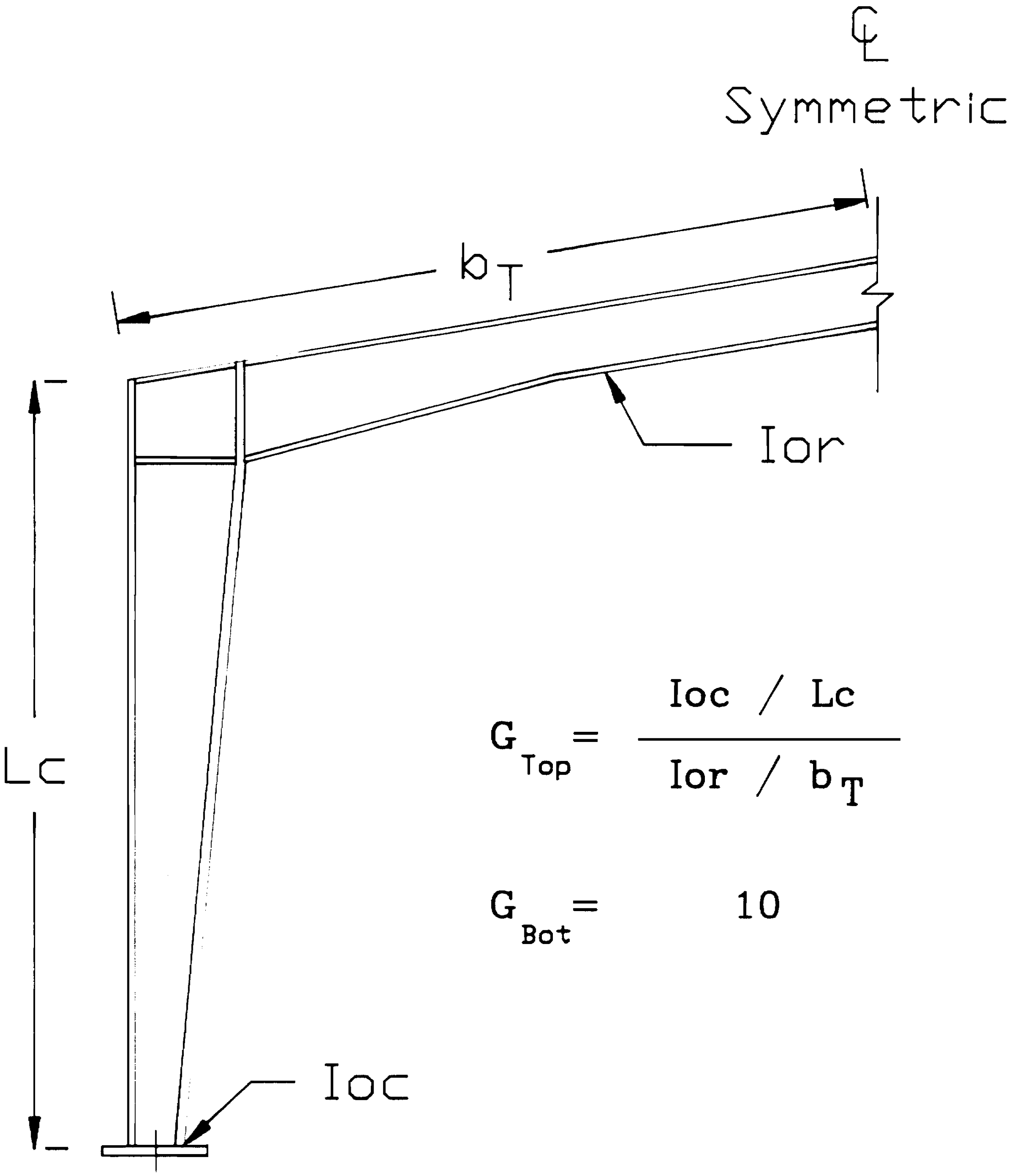
The response of the frame is traced using an incremental load analysis, in order to determine whether or not any chosen DOF has accrued a value of 12 in. in one load increment. When the frame reaches this point, the solution is terminated. The response is traced from a load of zero up to the calculated buckling load. The response must be traced in this manner because the first mode of buckling must be found, which occurs at some minimum energy level. For all the analyses, the load was applied in 0.5% increments of the assumed upper-bound buckling load.

Results of the SOEA show the percent of the total load applied, which is then multiplied by the value of the total load specified, obtaining the estimated critical buckling load. This method was applied to nine of the symmetric gable frames submitted by NUCOR.

6.4 Effective Length Study

In order to compare the critical buckling loads obtained in these analyses with those estimated by others, a K_γ-factor approach was used based on the tapered-column alignment charts found in the Commentary to the ASD Specification (AISC 1978 or 1989). The writer modified the calculation of the joint-rotation factors by using the minimum moment of inertia for the rafters, since the rafters are also tapered. The calculation of the joint-rotation factors in this application were as shown in Figure 6.2. Notice that the AISC recommended value of 10 for the restraint provided for the pinned-end is used and not the theoretical infinity value. Using this method, the K-factors were tabulated and the buckling loads calculated. Then, following the second-order elastic procedure described above, buckling loads were found, and the K-factors were calculated. Note that in the second-order elastic analysis, the theoretical critical buckling load is calculated. The computer model developed does not recognize the practical limitations of a pinned-support, i.e., the joint-rotation factor of an ideal pinned support is infinity.

The results of these analyses are shown in Table 6.1. P_e is the Euler Buckling load, assuming K_x = 1.0 (in-plane), and the column-base moment of inertia Io is used in all calculations. P_{buck} is the critical buckling load obtained from second-order elastic analysis. Chart K_γ is the K-factor determined by the modified alignment chart approach, and Calc'd K_γ is the K obtained through second-order elastic analysis. For all the cases studied, the modified alignment chart method was conservative, and lower relative to NUCOR's typical assumption of K_x = 1.5. This indicates that NUCOR may be able to use a K_x lower than 1.5 for most of their first-order frame designs. Because of this finding, first-order interaction checks of the ten frames were checked assuming K_x = 1.3 for the columns, and K_x = 1.85 for the rafters (though not shown, K_x = 1.85 is



$$G_{Top} = \frac{Ioc \;/\; Lc}{Ior \;/\; b_T}$$

$$G_{Bot} = 10$$

K_γ Calculation Parameters
Figure 6.2

Elastic Stability Analysis of Symmetric Gable Frames

Effective Length Study

Frame #	Taper Ratio	Chart K_γ	P_e	P_{Buck}	Calc'd K_γ
F1	2.00	1.00	125	135	0.96
F2	2.20	1.12	700	580	1.10
F3	2.00	1.00	240	255	0.97
F4	2.80	0.85	220	320	0.83
F5	3.00	0.97	1670	2045	0.90
F6	2.25	1.15	135	110	1.11
F7	2.00	1.25	330	270	1.11
F8	2.00	1.30	125	110	1.07
F9	3.30	1.30	145	195	0.86

Key:

P_e = Euler Buckling Load

P_{Buck} = Second Order Elastic Buckling Load

Table 6.1

based on similar application of second-order elastic analysis to the rafter members).

The use of K_X = 1.3 (in-plane, columns) and K_X = 1.85 (in-plane, rafters) is available. A conservative requirement to be checked before these values are used though, is the following, for symmetric clear-span gable frames, where the column taper ratio is greater than or equal to 2.0:

$$\frac{\text{Column Base Moment of Inertia/Eave Height}}{\text{Minimum Rafter Moment of Inertia/Frame Width}} \leq 1.0 \qquad (6.2)$$

This criterion is recommended by the writer, and is satisfied for the proportions of those frames involved in this research. This requirement can be verified by interpolation on the tapered column effective-length charts. For sway-permitted cases with taper ratios greater than 2.0, G_B = 10, and G_T = 0.5.

Note that the second-order elastic method calculated K_γ factors of approximately 1.0, but these were not recommended for the ASD first-order stability interaction equation because of the errors introduced by modeling tapered frames with uniform elements. The amount of error in the displacement analysis is approximated by a convergence study in Appendix C.

CHAPTER VII
RECOMMENDATIONS

7.1 Observations

The following observations were made during the course of the research:

- In order for second-order elastic analysis to be a benefit, stability interaction must control the design. Logistically, the member must be *more of a column than a beam*. This is not true for the range of NUCOR frames studied here, since yielding interaction is generally governing, according to Allowable Stress Design.

- It has been shown that elastic stability failures of rigid-frames of the dimensions studied in this research is not of primary concern. Rather, the *fully-stressed* design approach can be continued, assuming that *local buckling* is prevented. The directly calculated second-order effects under the ASD load combinations are *small*, indicating that these frames do possess adequate lateral stiffness to prevent a sidesway buckling mode.

- The alternate design checks using the direct application of second-order elastic analysis for either ASD or LRFD failed to provide economy according to the assumed economic criterion of a 3% strength increase versus an ASD first-order elastic analysis and design. This is due primarily to the yielding interaction equation generally governing the design of the frames, according to the ASD rules. By these comparisons, LRFD fails to provide economy due to both the factored load combinations and the direct inclusion of second-order effects.

Naturally, there are locations on the frames that do show marked increases in capacities, but the portions that do not comprise a greater influence area of the frames.

- Though the 1989 ASD Specification provides significant strength reductions for flexural design of slender cross-sections, it generally remains more economical than the 1986 LRFD.

7.2 Analysis Recommendations

Since the research conducted recognized potential problems in the structural analysis of rigid, tapered frames, the following should be recognized:

- There is approximately a 5% to 10% error in the first-order displacement analysis of tapered beam-columns when 2.5 ft. length prismatic elements are used (see Appendix D). A more highly refined discretization could be employed, though it would not be as efficient as using tapered elements. A change to tapered elements using the current discretization lengths would be most beneficial. A suggested model is given in Appendix C.
- For multi-span symmetric gable frames, the unbalanced snow load combination needs to be investigated as it may be critical.

7.3 Capacity Determination Recommendations

Research on the capacity side of the design equations applied to rigid, tapered frames recognized the following:

- Economically, NUCOR should continue to utilize first-order elastic analysis and design in accordance with the *Allowable Stress Design*

Specification. It was shown that LRFD will not provide economy for the range of frames and loadings commonly encountered in practice.

- Recommendations for effective-length factors given in Chapter VI may be used in the first-order ASD stability interaction equation, if the suggested criteria is met.

- Compressive allowable stress of slender cross-sections is determined as though *failure is imminent*, i.e., the stress used to calculate effective section properties is $f = F_a$. Iteration is required *anytime* the web is *slender*. Compressive strength under LRFD is calculated similarly, except for the substitution of $f = \phi P_n/A_g$ in its effective-width equation for the stiffened web element.

- Flexural allowable stress of slender cross-sections is taken as the *minimum* F_b from the slender provisions (1978 ASD: Appendix C, 1989 ASD: Appendix B), or the plate girder provisions (1978 ASD: Section 1.10.6, 1989 ASD: Chapter G). The 1989 ASD Specification criteria for flange/web local buckling interaction is under investigation by the AISC Specification Committees.

7.4 Future Research Needs

Due to inconsistencies with the application of prismatic design rules to tapered members, the following should be investigated:

- The determination of the in-plane K_x for tapered members is suggested to be chosen as the maximum value from:

$$K_x = \text{max.}[\ 1.5,\ r_x C'c/H\ ] \tag{7.1}$$

or, if the frame proportions meet the suggested Chapter VI criteria,

$$K_x = \max.[\, 1.3,\ r_x C'c/H \,] \tag{7.2}$$

Where r_x = In-plane radius of gyration at the smaller end of the column, $C'c = Cc/\sqrt{Q}$ (if slender cross-sections are involved), and H = the frame eave height. This is a requirement of the *true* tapered member design rules of ASD, in order to check stresses incrementally along the length of a tapered member (1989 ASD Appendix F). This requirement assures that the frame is behaving elastically under the application of combined stresses.

- Lateral-torsional flexural design under ASD, the C_b factor is suggested to be calculated as:

$$C_b = 1.30 + 1.05\left(\frac{f_{b1}}{f_{b2}}\right) + 0.3\left(\frac{f_{b1}}{f_{b2}}\right) \tag{7.3}$$

where

$f_{b1} = \min(\, M_1/S_1,\ M_2/S_2)$

$f_{b2} = \max(\, M_1/S_1,\ M_2/S_2\,)$

This is suggested because C_b was developed for *prismatic* members. The change to *stresses* is warranted because although M_1/M_2 is identically less than 1.0, in terms of stresses, $(M_1/S_1)/(M_2/S_2)$ may be greater than 1.0. This also follows from the *closed-form* tapered beam provisions. There, the C_b factor appears as *B*, and is calculated based on stresses.

- Also for lateral-torsional flexural design under ASD, it is appropriate to utilize the vector sum of the warping and St. Venant allowable flexural stresses, using a modified r_T equation, as the following criteria suggests (AISC, 1978):

Let $$r_T = \frac{I_y}{2S_x} \sqrt{d^2 + \frac{0.156L^2J}{I_y}} \tag{7.4}$$

Then:

For $$\frac{L}{r_T} > \sqrt{\left(\frac{102000*C_b}{F_y}\right)} \tag{7.5}$$

$$F_b = \sqrt{(F_{sv}^2 + F_w^2)} \leq = 0.60F_y \tag{7.6}$$

For $$\frac{L}{r_T} > \sqrt{\left(\frac{102000*C_b}{F_y}\right)} \text{ but less than Eqn. 7.8,} \tag{7.7}$$

$F_w = F_b$ from 1978 ASD 1.5-6a

$F_{sv} = F_b$ from 1978 ASD 1.5-7

For $$\frac{L}{r_T} > \sqrt{\left(\frac{510000*C_b}{F_y}\right)} \tag{7.8}$$

$F_w = F_b$ from 1978 ASD 1.5-6b

$F_{sv} = F_b$ from 1978 ASD 1.5-7

- Assuming F_y = 50 ksi, for (70 < h/t < 138), F_b will most likely come from the new flange/web local buckling criteria, if NUCOR continues to select slender flange proportions. This will require the use of thicker webs or flanges for future NUCOR frames. The applicability of the 1989 ASD flange/web local buckling interaction should be verified experimentally for the range of cross-sections proportioned by NUCOR.

- Should it be applicable, compliance with new Uniform Building Code wind pressures should be verified upon it next publication.

- A study on base plate stiffening could be initiated in order to determine a value for the actual rotational restraint provided by NUCOR column bases, instead of the value *10*, usually assumed for the K_γ determination. However, since elastic stability does not seem to be of concern, a decrease of the in-plane effective-length factor for the tapered columns may not result in an increase of a frames' capacity.

REFERENCES

Adams, P. F., Wood, B. R., and Beaulieu, D., (1976) "Column Design by P-Delta Method", *Journal of the Structural Division*, ASCE, Vol. 102, No. ST2.

Adams, P. F., Wood, B. R., and Beaulieu, D., (1976) "Further Aspects of Design by P-Delta Method", *Journal of the Structural Division*, ASCE, Vol, 102, No. ST3.

AISC, (1978) *Specification for the Design, Fabrication, and Erection of Structural Steel for Buildings*, AISC, Chicago, Il.

AISC, (1986) "Load and Resistance Factor Design Specification for Structural Steel Buildings", AISC, Chicago, Il.

AISC, (1989) *Specification for the Design, Fabrication, and Erection of Structural Steel for Buildings*, AISC, Chicago, Il.

AISC, (1989a) "Allowable Stress Design in Steel, A New Lecture Program from AISC", AISC, Chicago, Il.

AISI, (1968) *Specification for the Design of Cold-Formed Steel Structural Members*, AISI, Washington, D.C.

AISI, (1977) *Specification for the Design of Cold-Formed Steel Structural Members*, AISI, Washington, D.C.

AISI, (1986) *Specification for the Design of Cold-Formed Steel Structural Members*, AISI, Washington, D.C.

Bathe, K. J., and Wilson, E. J., (1976) Numerical Methods in Finite Element Analysis, Prentice Hall Inc., Englewood Cliffs, NJ.

Butler, M., (1983) "Comparison of Two Models for Geometrically Nonlinear Finite Element Analysis of Plane Frames", Master of Science Thesis, Virginia Polytechnic Institute & State University.

Chen, W. F., and Lui, E. M., (1987) Structural Stability - Theory and Implementation, Elsevier Press, New York, NY.

Cheong-Siat-Moy, F., et. al., (1977) "Strength of Steel Frames Under Gravity Loads", *Journal of the Structural Division*, ASCE, No. ST6.

Cook, R. D., et al.,(1989) Concepts and Applications of Finite Element Analysis, Third Edition, John Wiley & Sons, New York, NY.

Gaiotti, R., and Smith, B. S., (1989) "P-Delta Analysis of Building Structures", *Journal of the Structural Division*, ASCE, Vol. 115, No 4.

Galambos, T. V., (1968) Structural Members and Frames, Prentice-Hall Inc., Englewood Cliffs, NJ.

Galambos, T. V., Ed., (1976) Guide to Stability Design Criteria for Metal Structures, John Wiley & Sons, New York, NY.

Goto, Y. and Chen, W. F. (1987) "Second Order Elastic Analysis for Frame Design", *Journal of the Structural Division*, ASCE, Vol. 113, No. 7.

Holzer, S. M., (1985) Computer Analysis of Structures, Elsevier Press Inc., New York, NY.

International Conference of Building Officials, (1988) Uniform Building Code, ICBO, Whittier, Ca.

International Conference of Building Officials, (1990) "Building Standards", Part III, Code Change Agenda, ICBO, Whittier, Ca.

Iwankiw, N. (1990) Personal Communication, letter dated August 6, 1990.

Korn, A. and Galambos, T. V., (1968) "Behavior of Elastic-Plastic Frames", *Journal of the Structural Division*, ASCE, No. ST5.

Lee, G. C., et al., (1972) "Design of Tapered Members", WRC Bulletin No. 173, June.

Lee, G. C., et al., (1974) "Allowable Stress for Web-Tapered Beams with Lateral Restraints", WRC Bulletin No. 192, February.

Lee, G. C., et. al, (1981) Design of Single Story Rigid Frames, MBMA, Cleveland, Oh.

LeMessurier, W. J., (1977) "A Practical Method of Second Order Analysis / Part 2 - Rigid Frames", *AISC Engineering Journal*, Vol. 14, No. 2, Second Quarter.

Lui, E. M., (1988) "A Practical P-Delta Analysis Method for Type PR and FR Frames", *AISC Engineering Journal*, Third Quarter.

Metal Building Manufacturers Association, (1986) Low Rise Building Systems Manual, MBMA, Cleveland, Oh.

Nixon, D., Beaulieu, D., and Adams, P. F., (1975) "Simplified Second Order Frame Analysis", *Canadian Journal of Civil Engineering*, 2(4).

Oran, C. P., (1973) "Tangent Stiffness in Plane Frames", *Journal of the Structural Division*, ASCE, No. ST6.

Oran, C. P., (1973) "Tangent Stiffness in Space Frames", *Journal of the Structural Division*, ASCE, No. ST6.

Salmon, C. G., and Johnson, J. E., (1980) Steel Structures, Design and Behavior, Second Edition, Harper & Row, New York, NY.

Salmon, C. G., and Johnson, J. E., (1989) Steel Structures, Design and Behavior, Third Edition, Harper & Row, New York, NY.

Smith, J. C., (1988) Structural Analysis, Harper & Row, Inc., New York, NY.

Spangler, D. Jr., (1989) "P-Delta And Partial Knee Stiffness Effects Of Rigid Primary Framing Systems", MS Thesis, Virginia Polytechnic Institute & State University.

Timoshenko, S. P. and Gere, J. M., (1961) Theory of Elastic Stability, McGraw-Hill Inc., New York, NY.

Wang, C. K., (1973) Computer Methods in Advanced Structural Analysis, Intext Educational Publishers, New York, NY.

White, D. W., and Hajjar, J. F., (1990) "Application of Second-Order Elastic Analysis in LRFD: Research to Practice", *Proceedings of the 1990 Steel Construction Conference*, AISC, Kansas City, Mo., May 1990.

Wishart, Eric J. (1990) Spreadsheet Shear-Strength Provision Comparison of LRFD and ASD, Unpublished, May, 1990

Yang, T. Y., (1986) Finite Element Structural Analysis, Prentice-Hall Inc., Englewood Cliffs, NJ.

Young, C. S., (1989) "Symmetric, Variable-Bandwidth Linear Equation Solver, C Source Code", Unpublished, December, 1989.

Yu, W. W., (1985) Cold-Formed Steel Design, Wiley Interscience, New York, NY.

Yura, J. A., (1971) "The Effective Length of Columns in Unbraced Frames", *AISC Engineering Journal*, April.

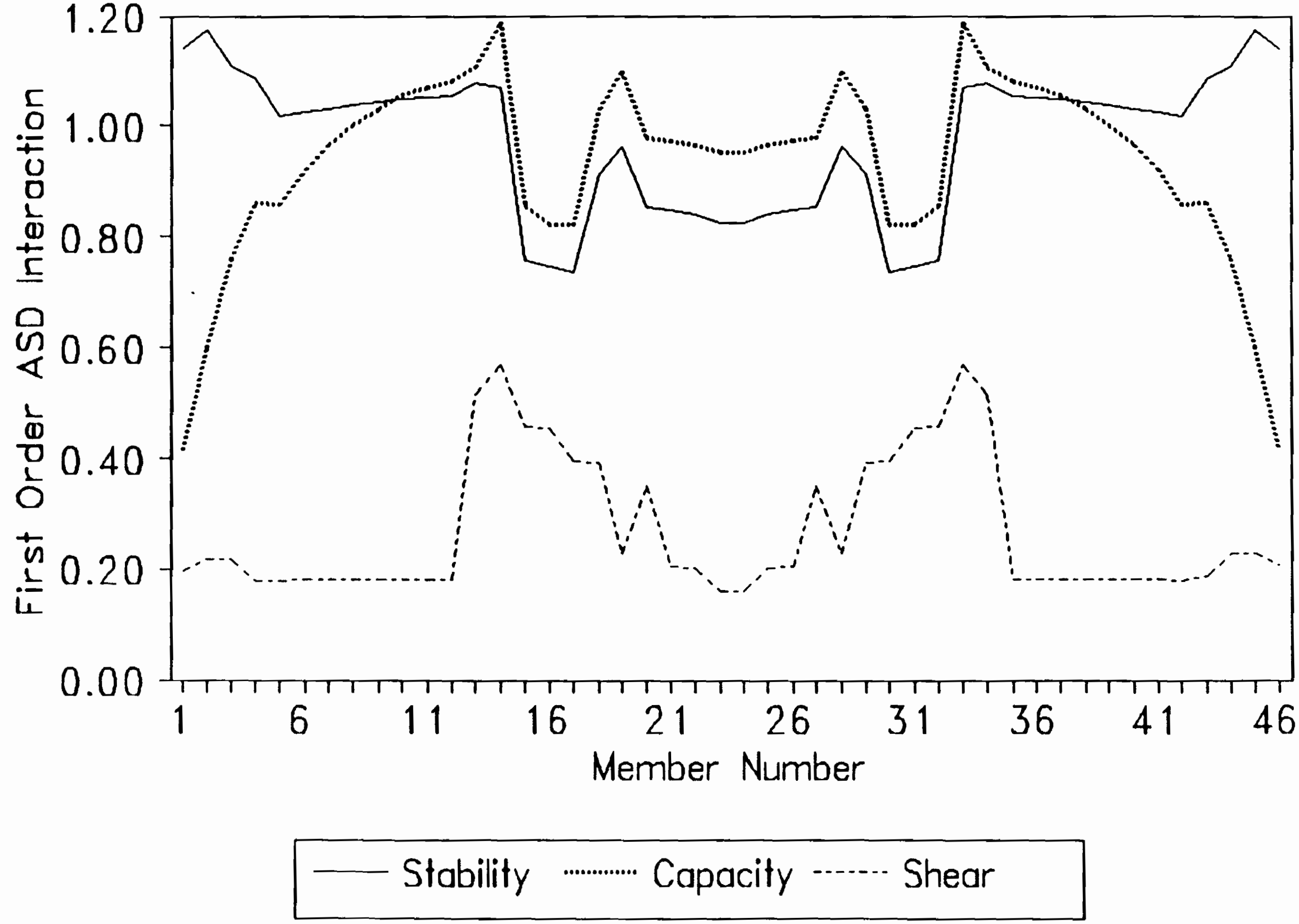
Yura, J. A., "Elements for Teaching Load and Resistance Factor Design", AISC, Chicago, Ill. 1987.

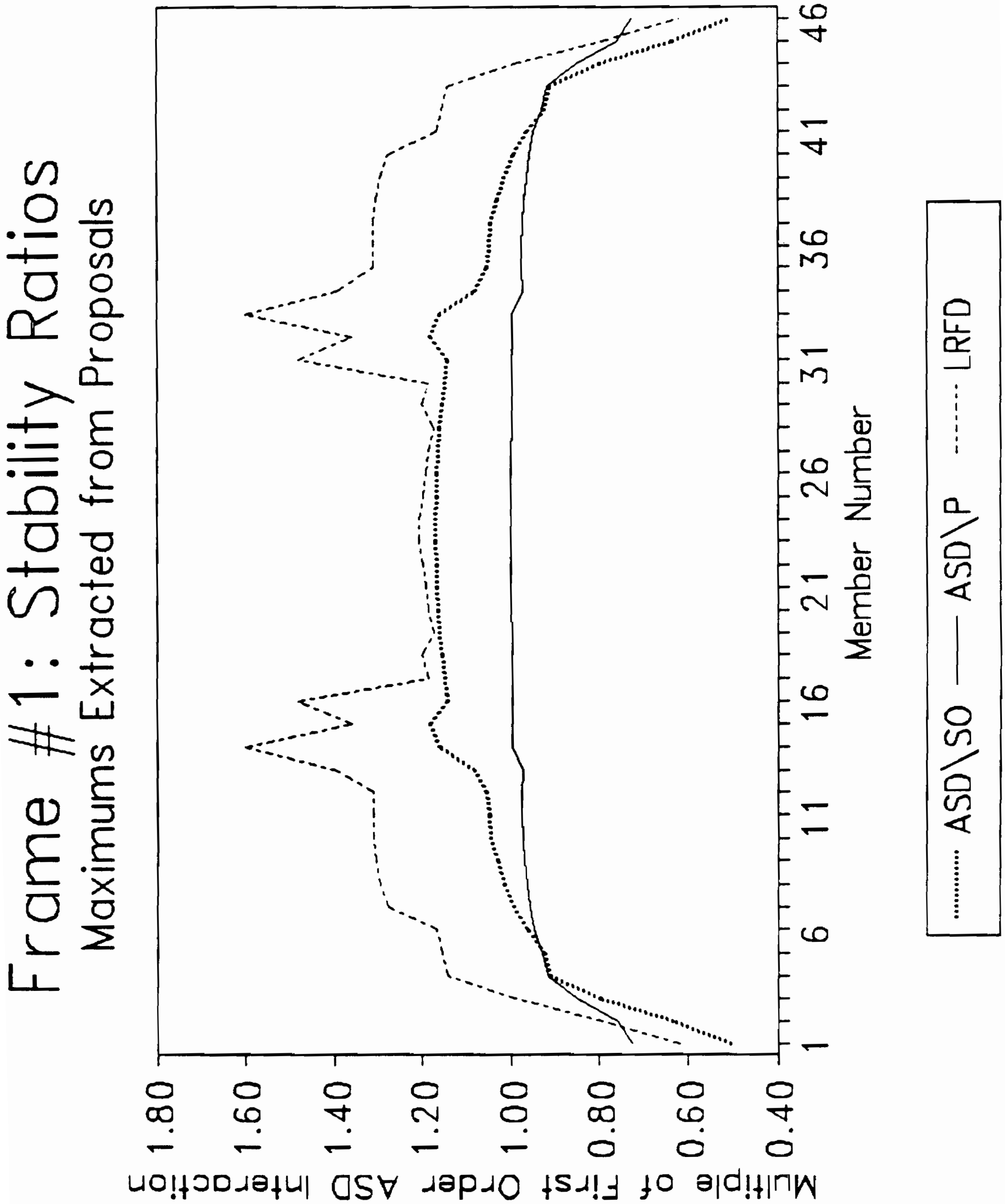
APPENDIX A

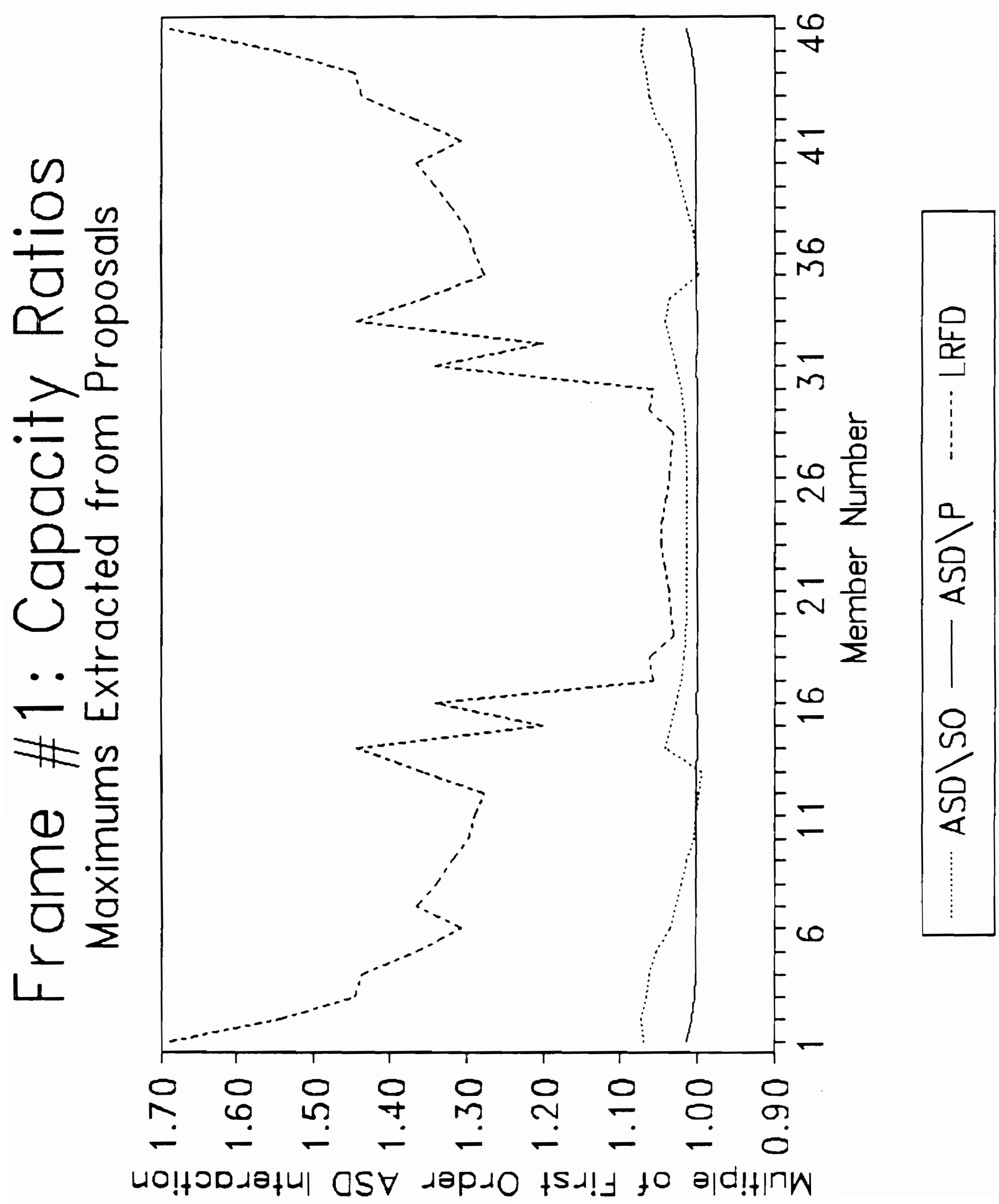
MAXIMUM INTERACTION RATIOS OF FRAMES

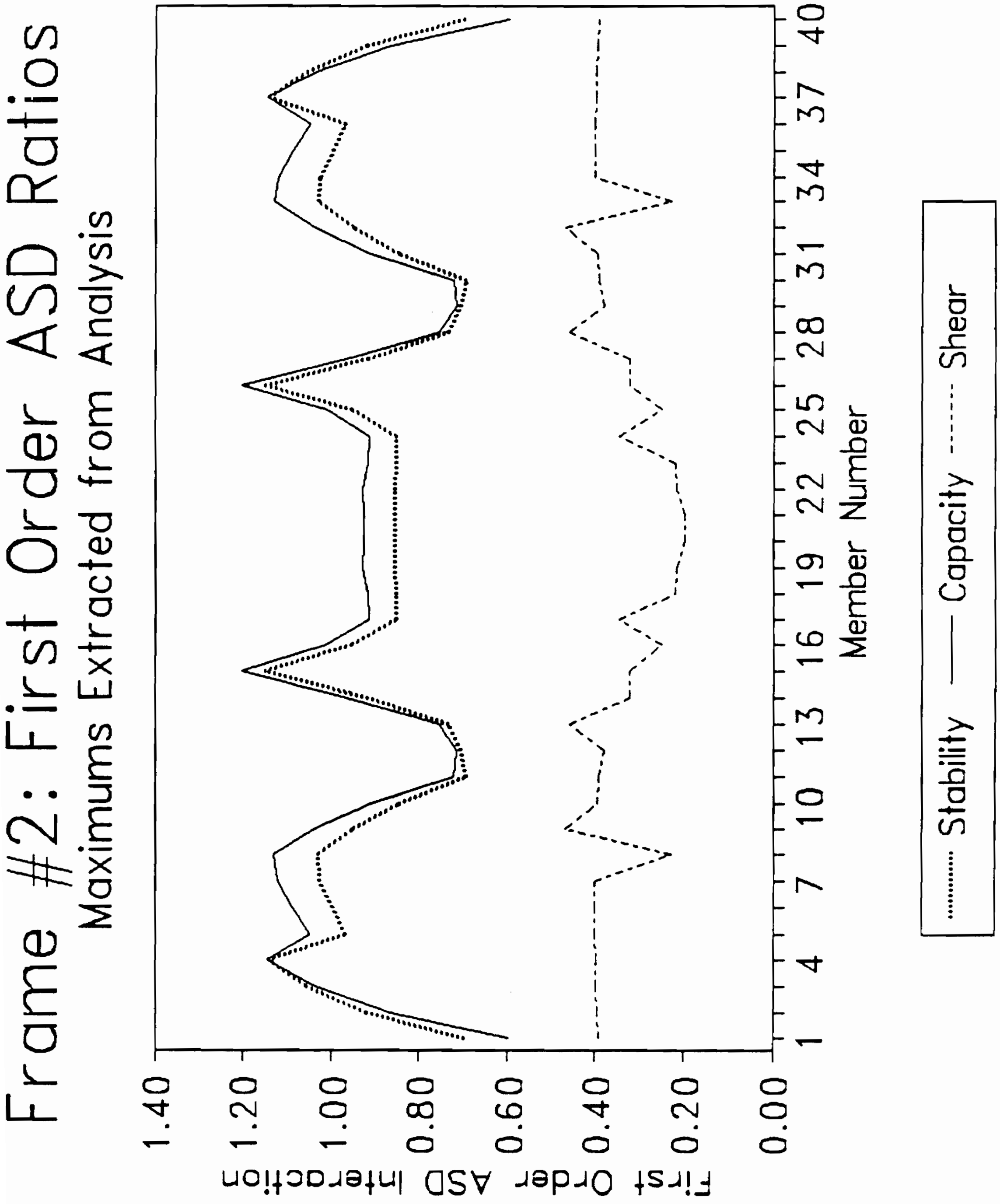
Frame #1: First Order ASD Ratios

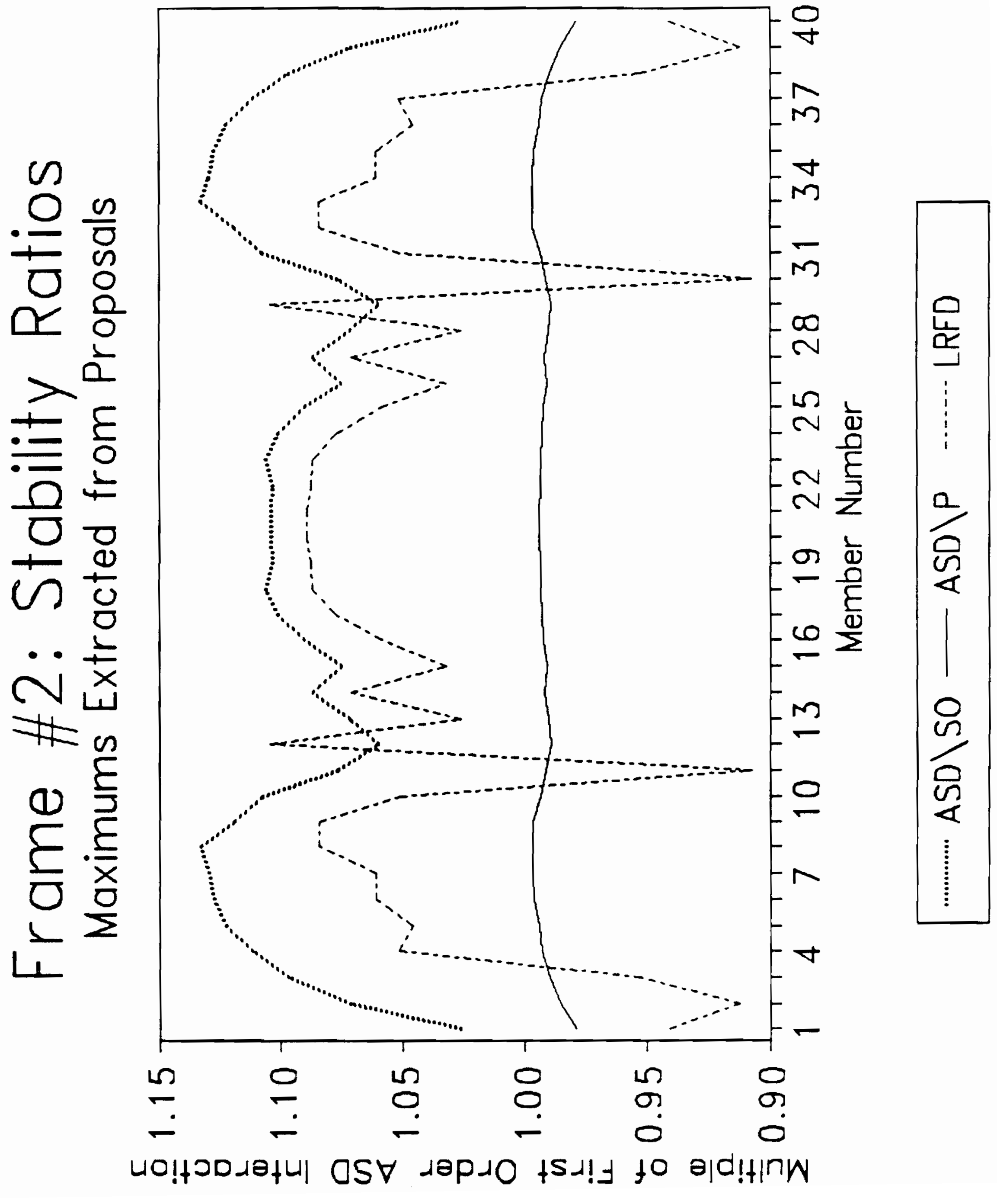
Maximums Extracted from Analysis

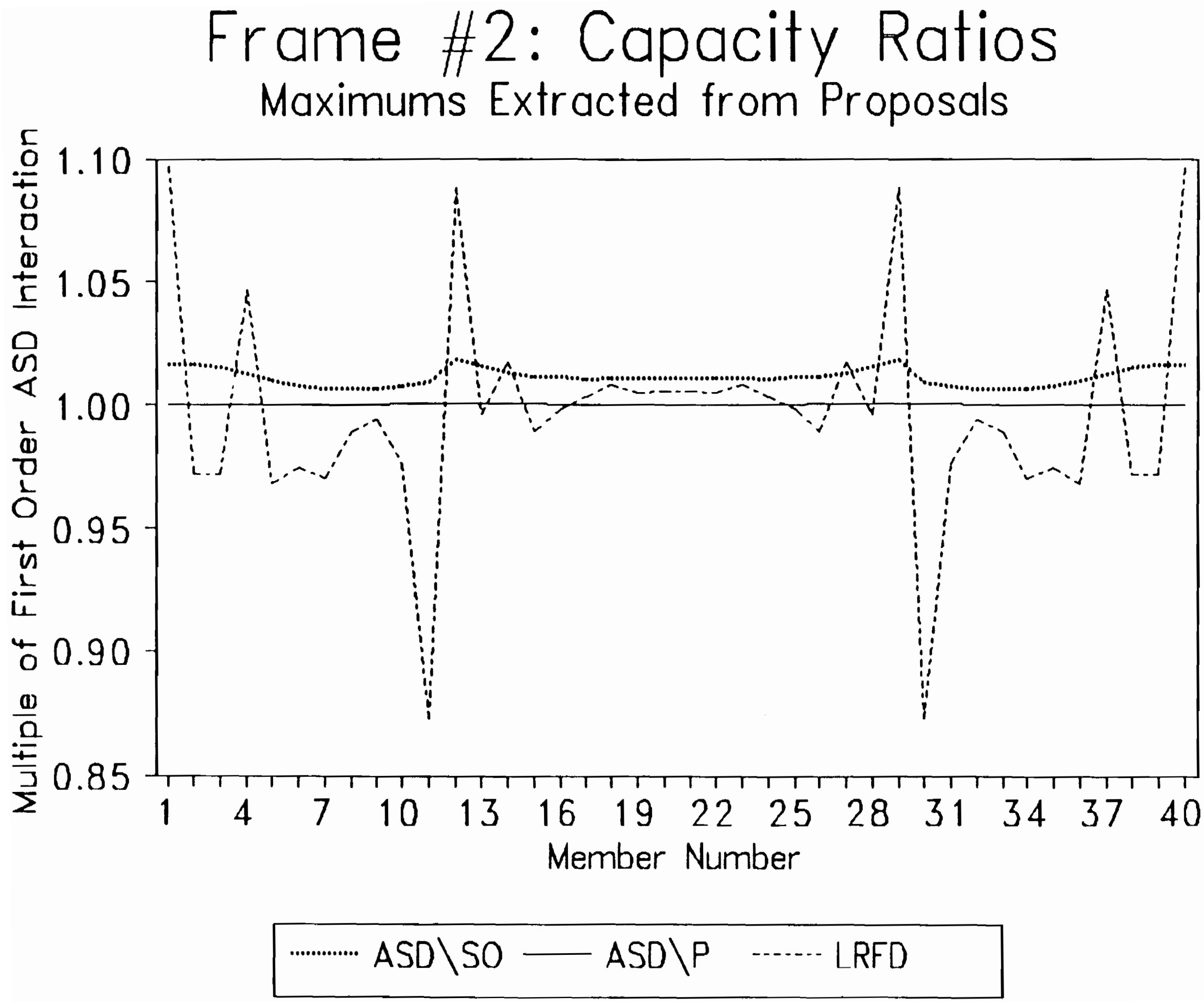


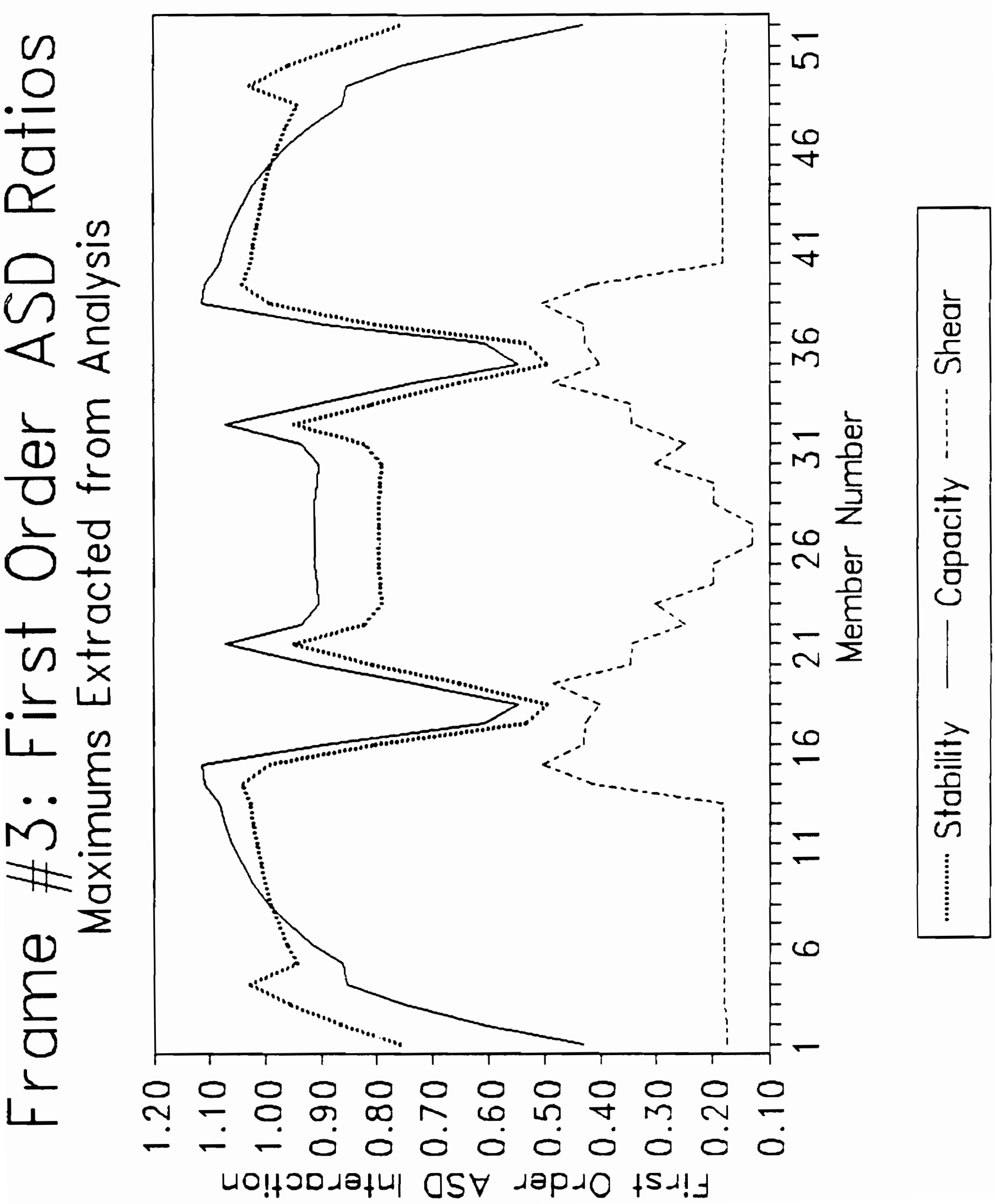

Frame #1: Stability Ratios
Maximums Extracted from Proposals
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
Multiple of First Order ASD Interaction
1
6
11
16
21
26
31
36
41
46
Member Number
ASD\SO
ASD\P
LRFD

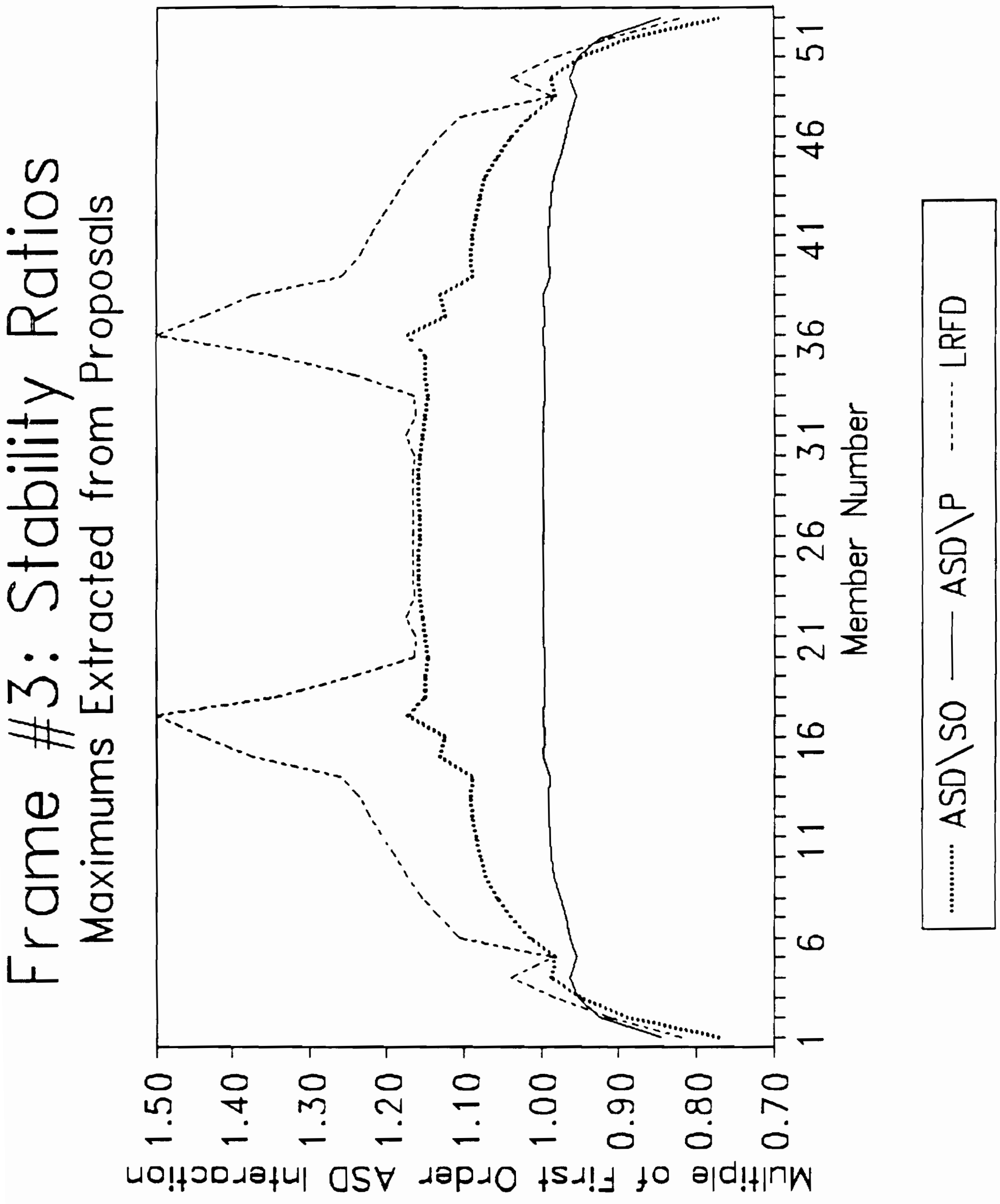

Frame #1: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.70
1.60
1.50
1.40
1.30
1.20
1.10
1.00
0.90
1
6
11
16
21
26
31
36
41
46
Member Number
ASD\SO
ASD\P
LRFD

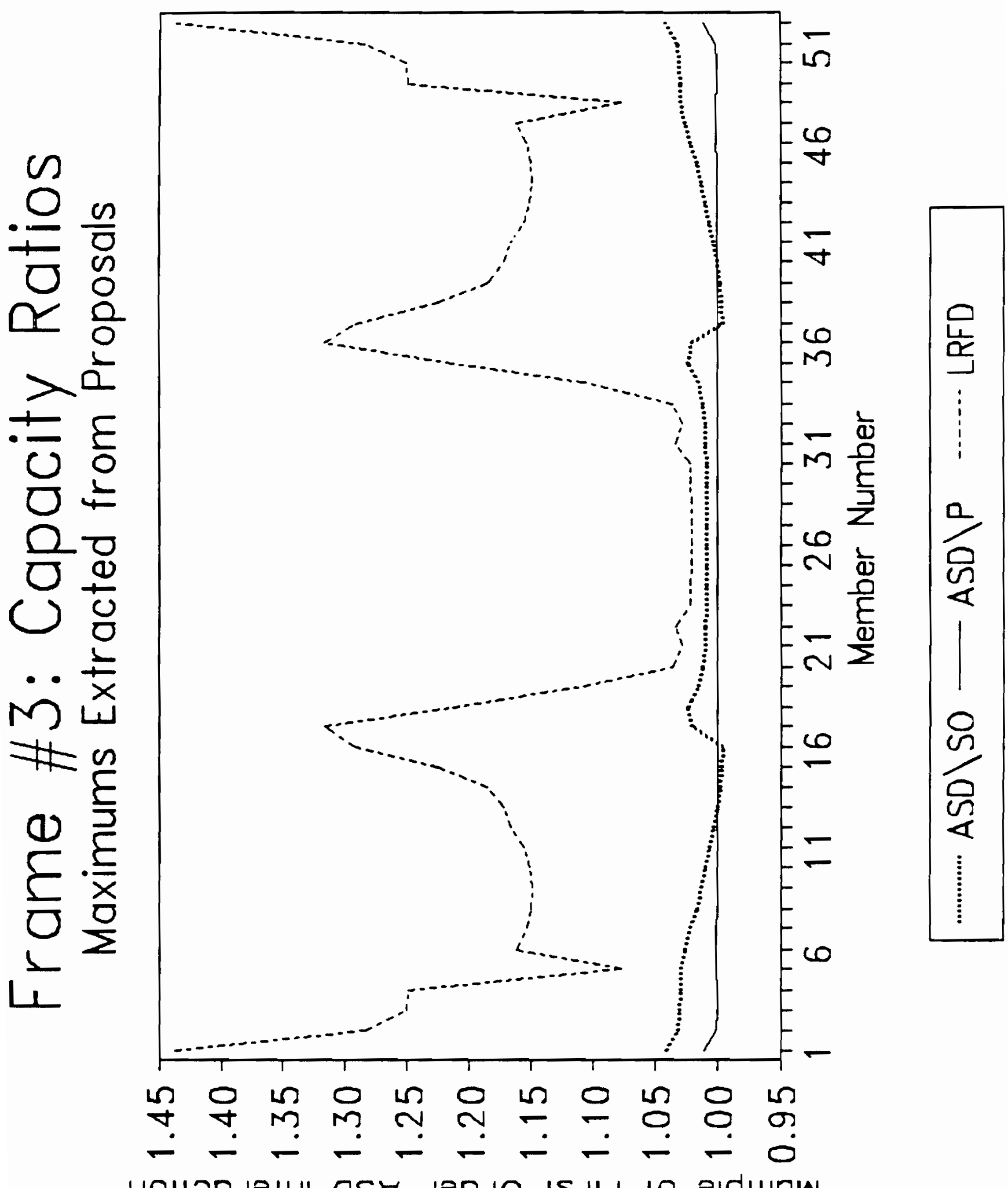

Frame #2: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
Member Number
1
4
7
10
13
16
19
22
25
28
31
34
37
40
Stability
Capacity
Shear

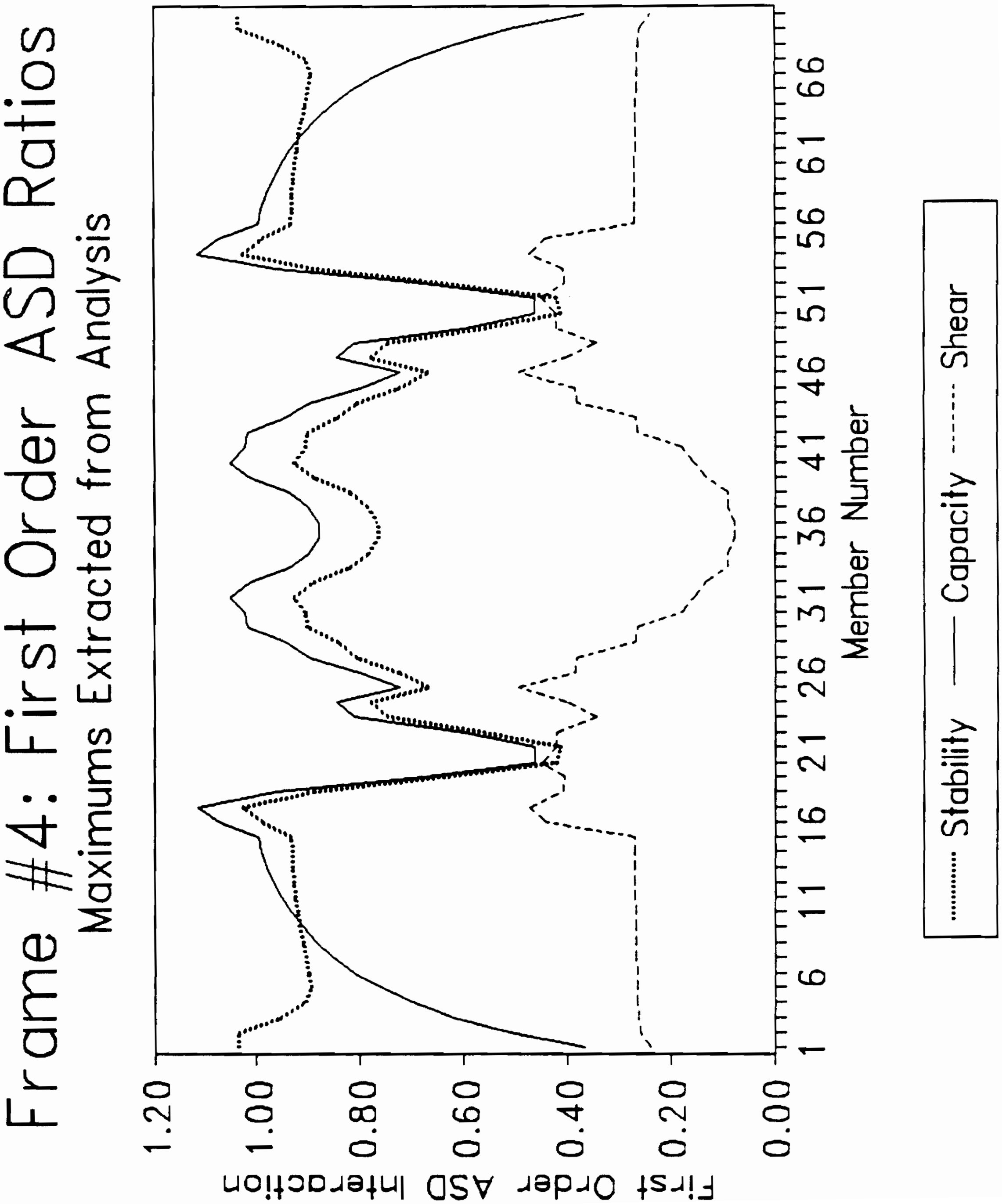

Frame #2: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.15
1.10
1.05
1.00
0.95
0.90
1
4
7
10
13
16
19
22
25
28
31
34
37
40
Member Number
ASD\SO
ASD\P
LRFD

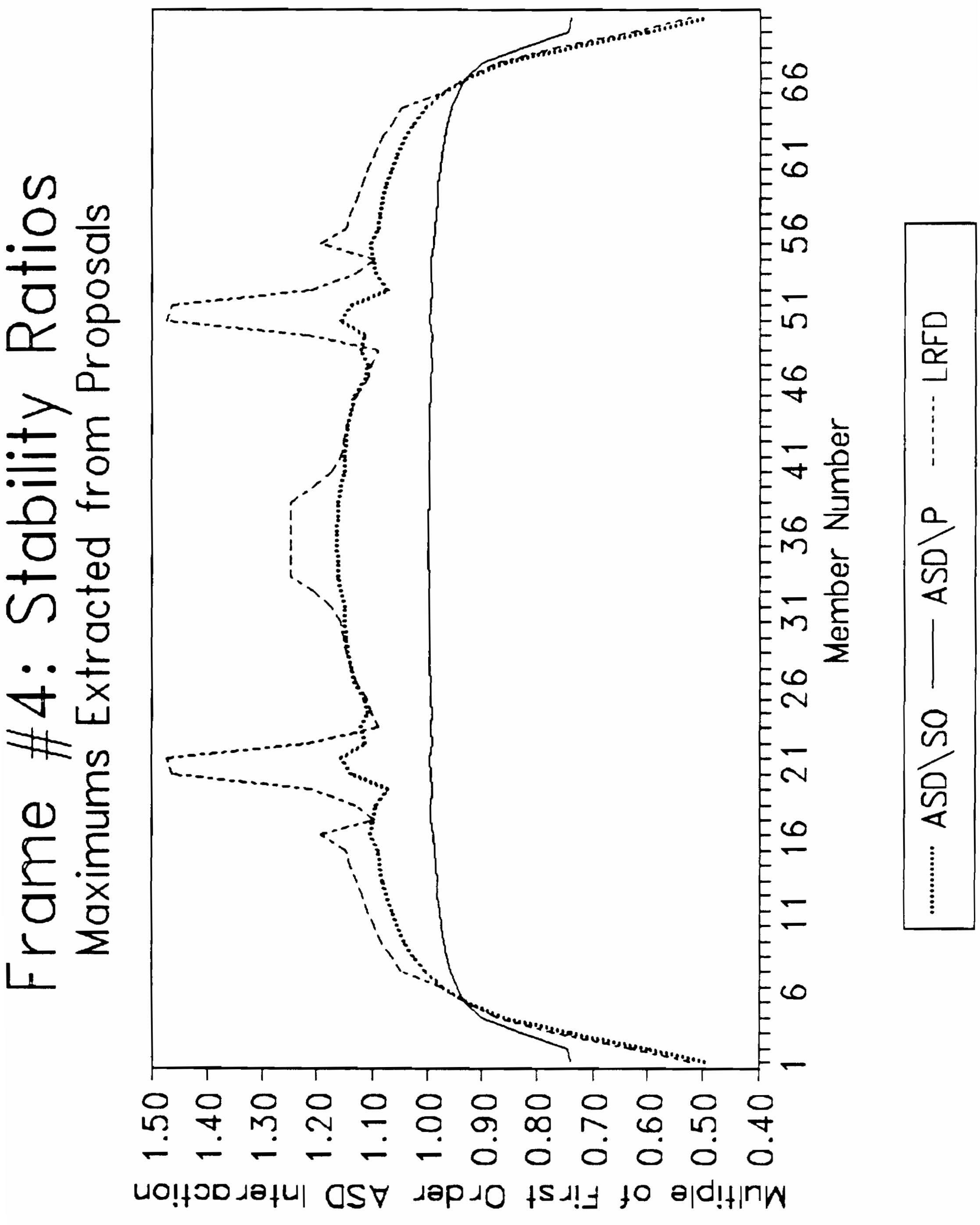

Frame #2: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.10
1.05
1.00
0.95
0.90
0.85
1
4
7
10
13
16
19
22
25
28
31
34
37
40
Member Number
ASD\SO
ASD\P
LRFD

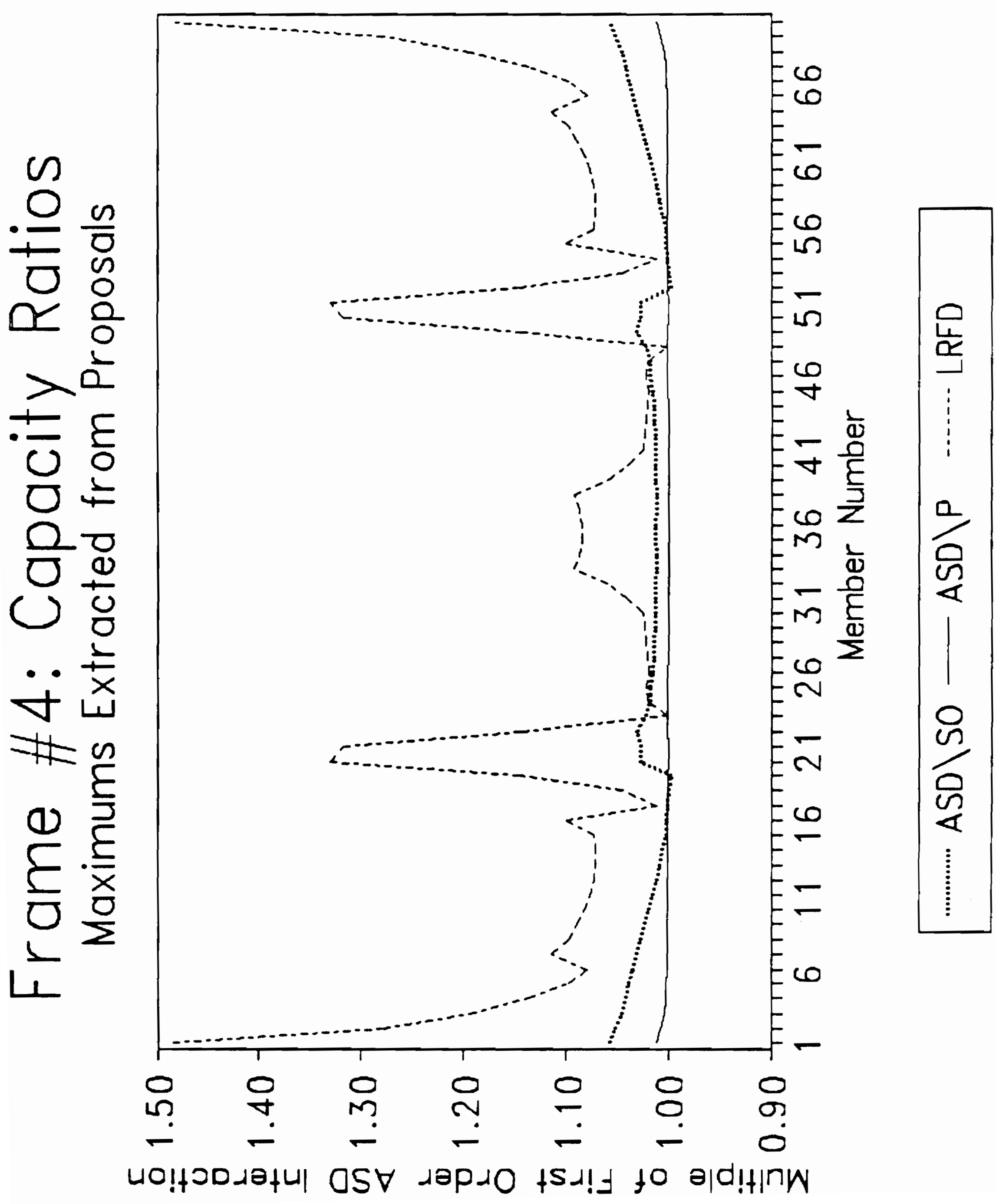

Frame #3: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
1
6
11
16
21
26
31
36
41
46
51
Member Number
Stability
Capacity
Shear

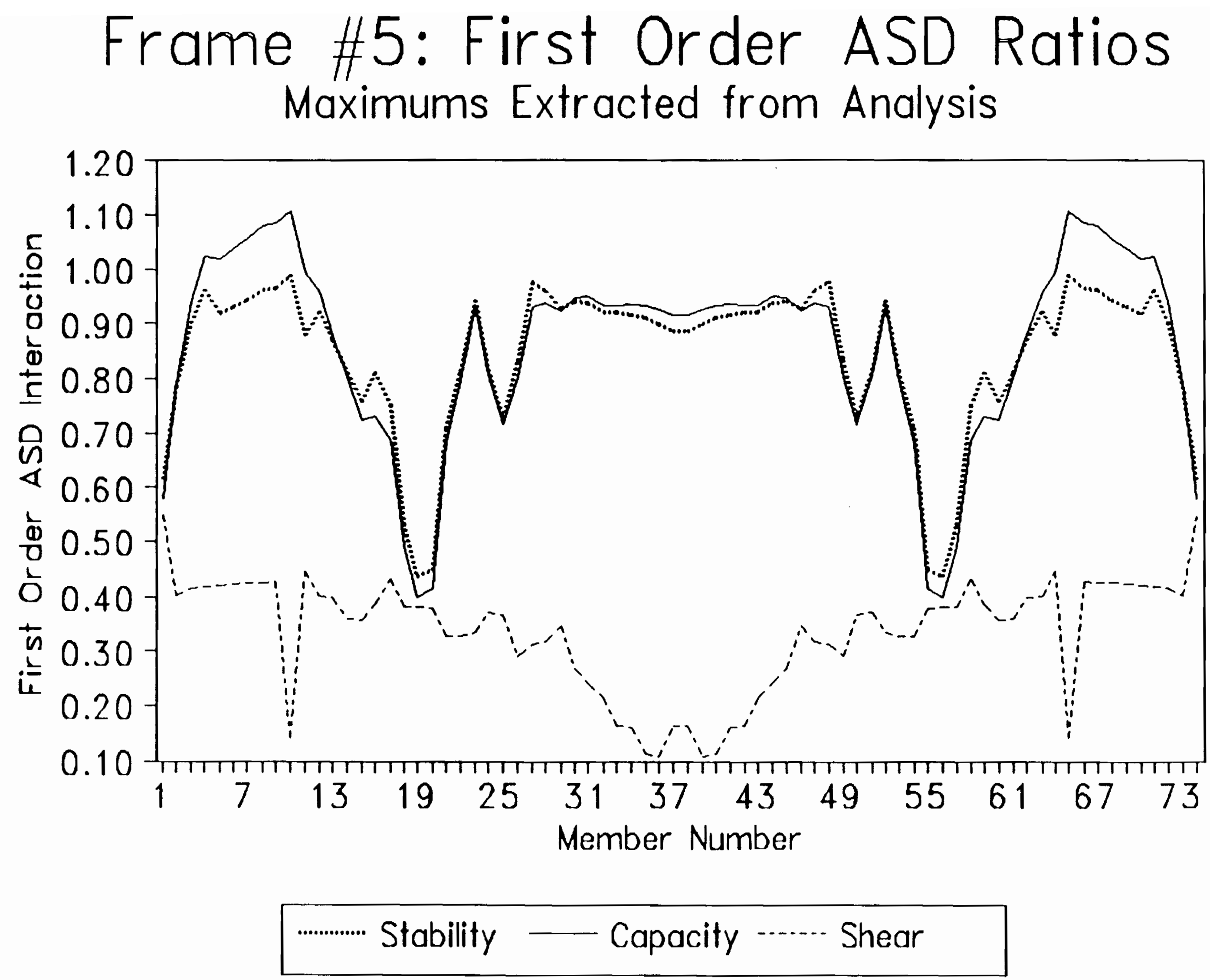

Frame #3: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.50
1.40
1.30
1.20
1.10
1.00
0.90
0.80
0.70
1
6
11
16
21
26
31
36
41
46
51
Member Number
ASD\SO
ASD\P
LRFD

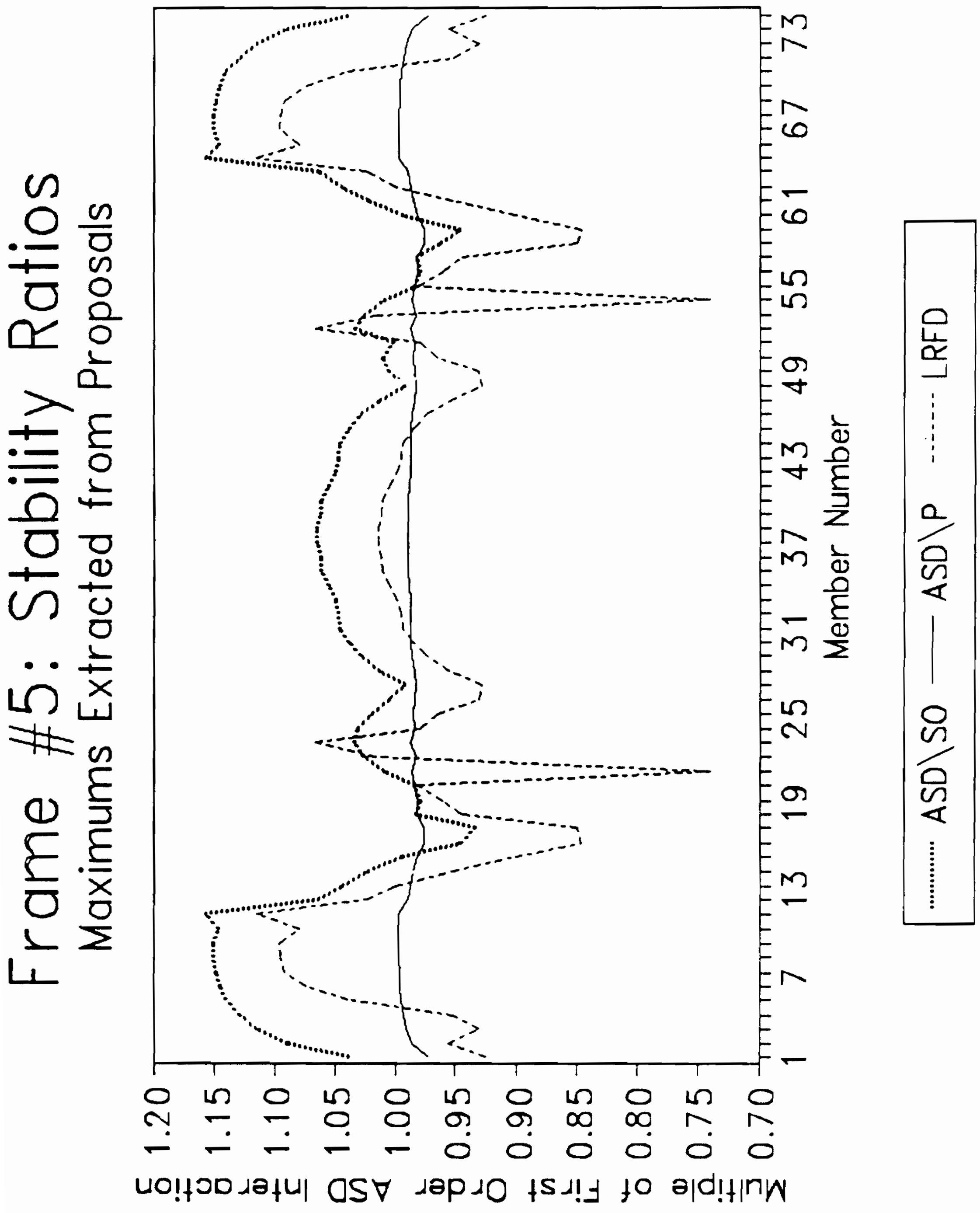

Frame #3: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.45
1.40
1.35
1.30
1.25
1.20
1.15
1.10
1.05
1.00
0.95
1
6
11
16
21
26
31
36
41
46
51
Member Number
ASD\SO
ASD\P
LRFD


Frame #4: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1
6
11
16
21
26
31
36
41
46
51
56
61
66
Member Number
Stability
Capacity
Shear

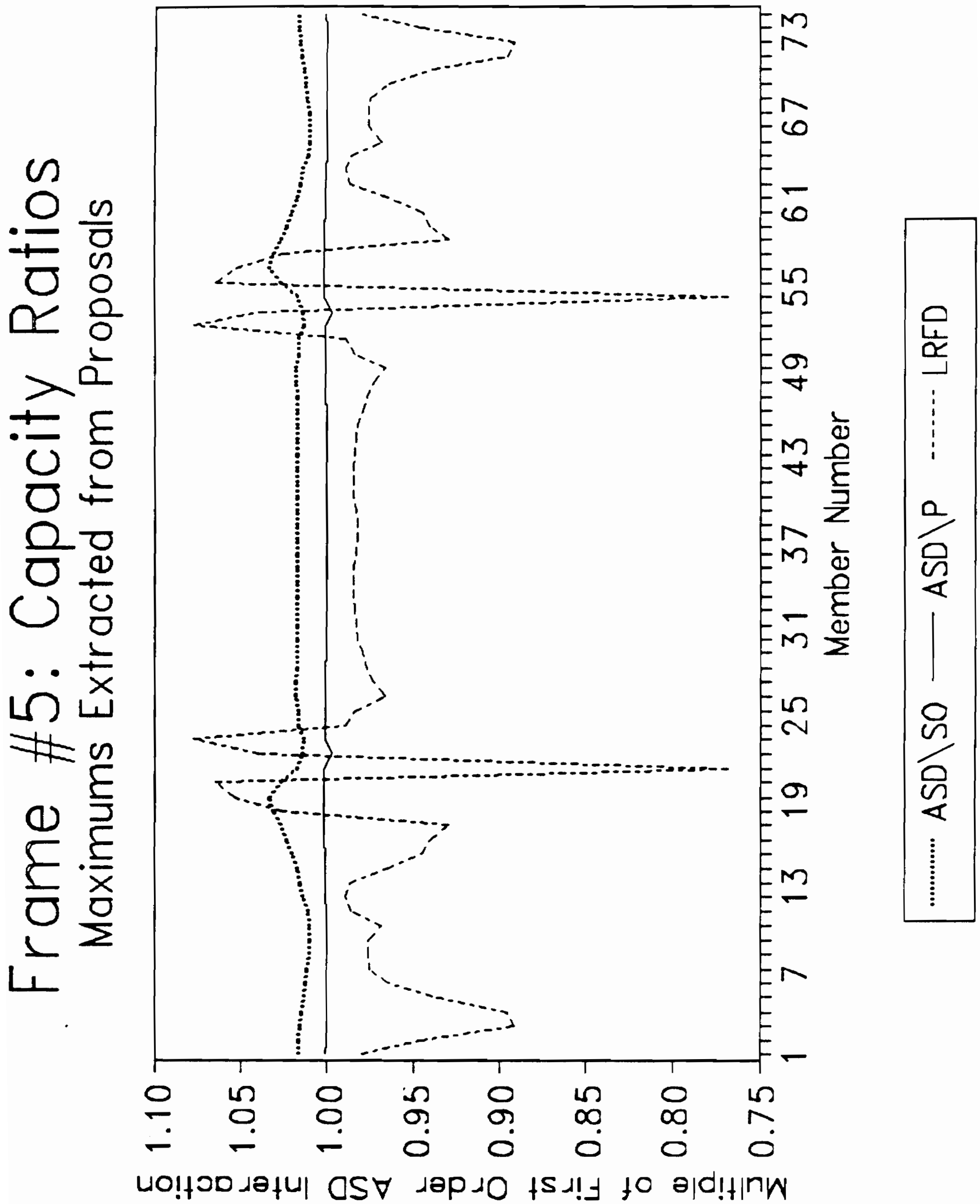

Frame #4: Stability Ratios
Maximums Extracted from Proposals
1.50
1.40
1.30
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
Multiple of First Order ASD Interaction
1
6
11
16
21
26
31
36
41
46
51
56
61
66
Member Number
ASD\SO
ASD\P
LRFD

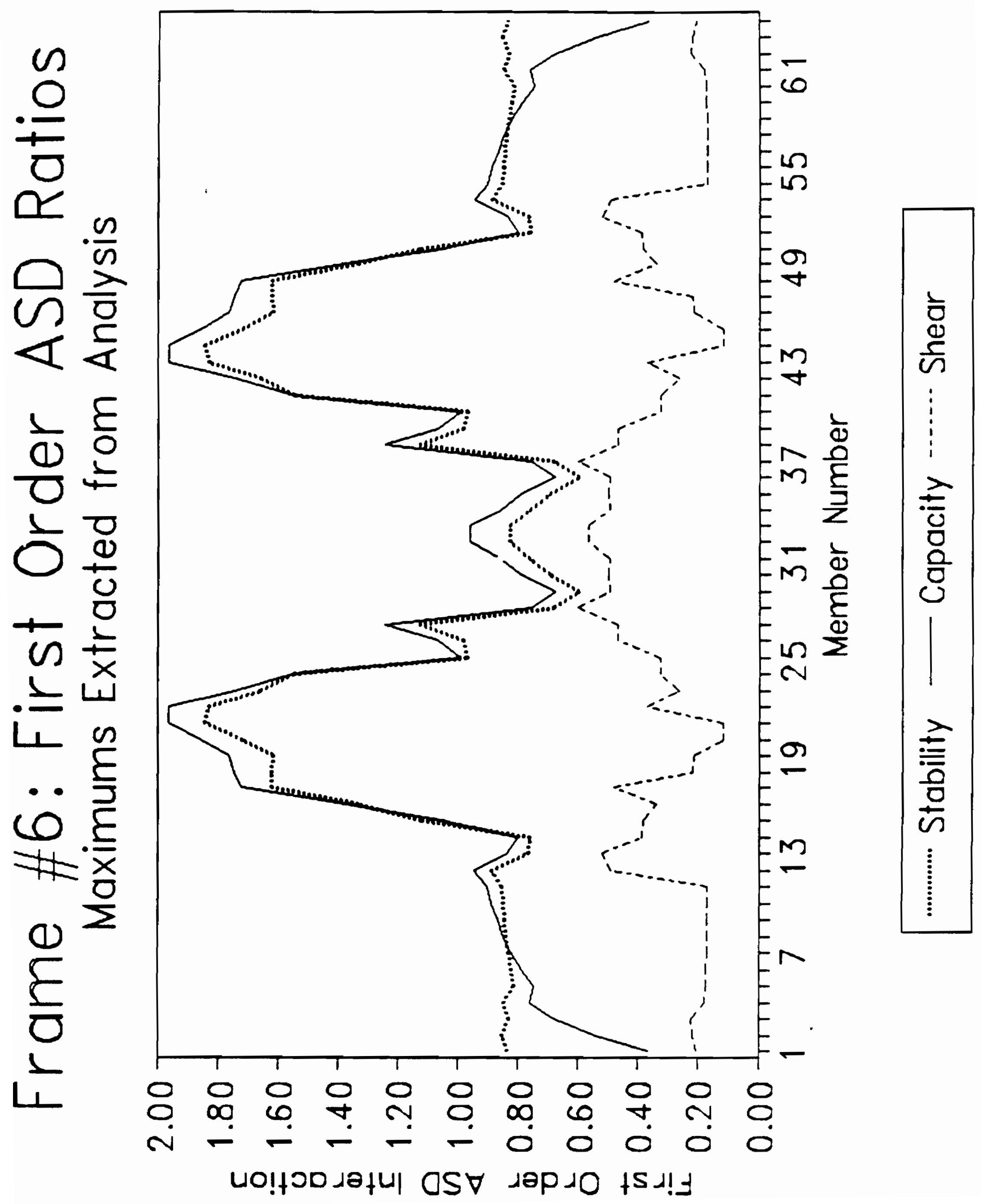

Frame #4: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.50
1.40
1.30
1.20
1.10
1.00
0.90
1
6
11
16
21
26
31
36
41
46
51
56
61
66
Member Number
ASD\SO
ASD\P
LRFD

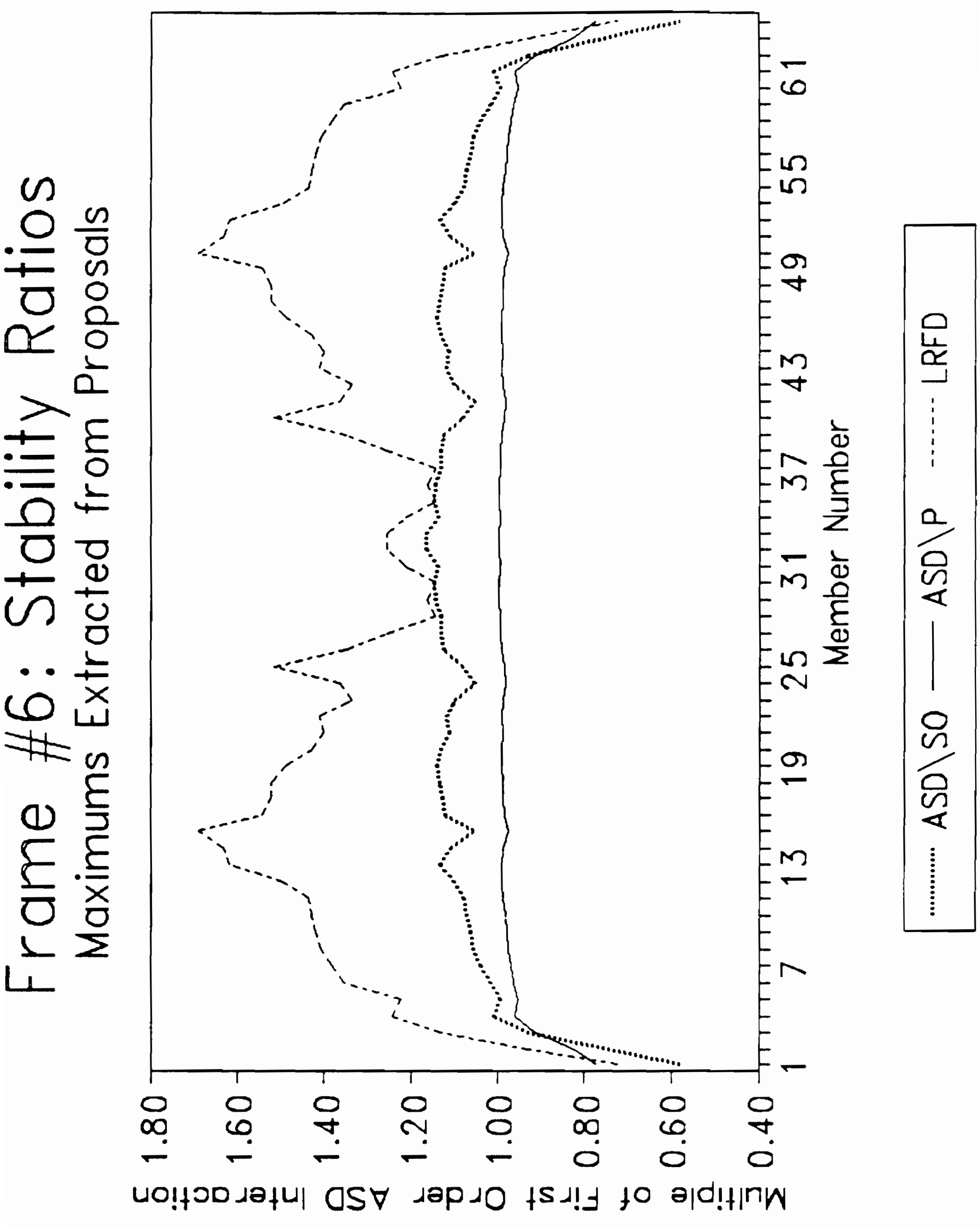

Frame #5: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
1
7
13
19
25
31
37
43
49
55
61
67
73
Member Number
Stability
Capacity
Shear

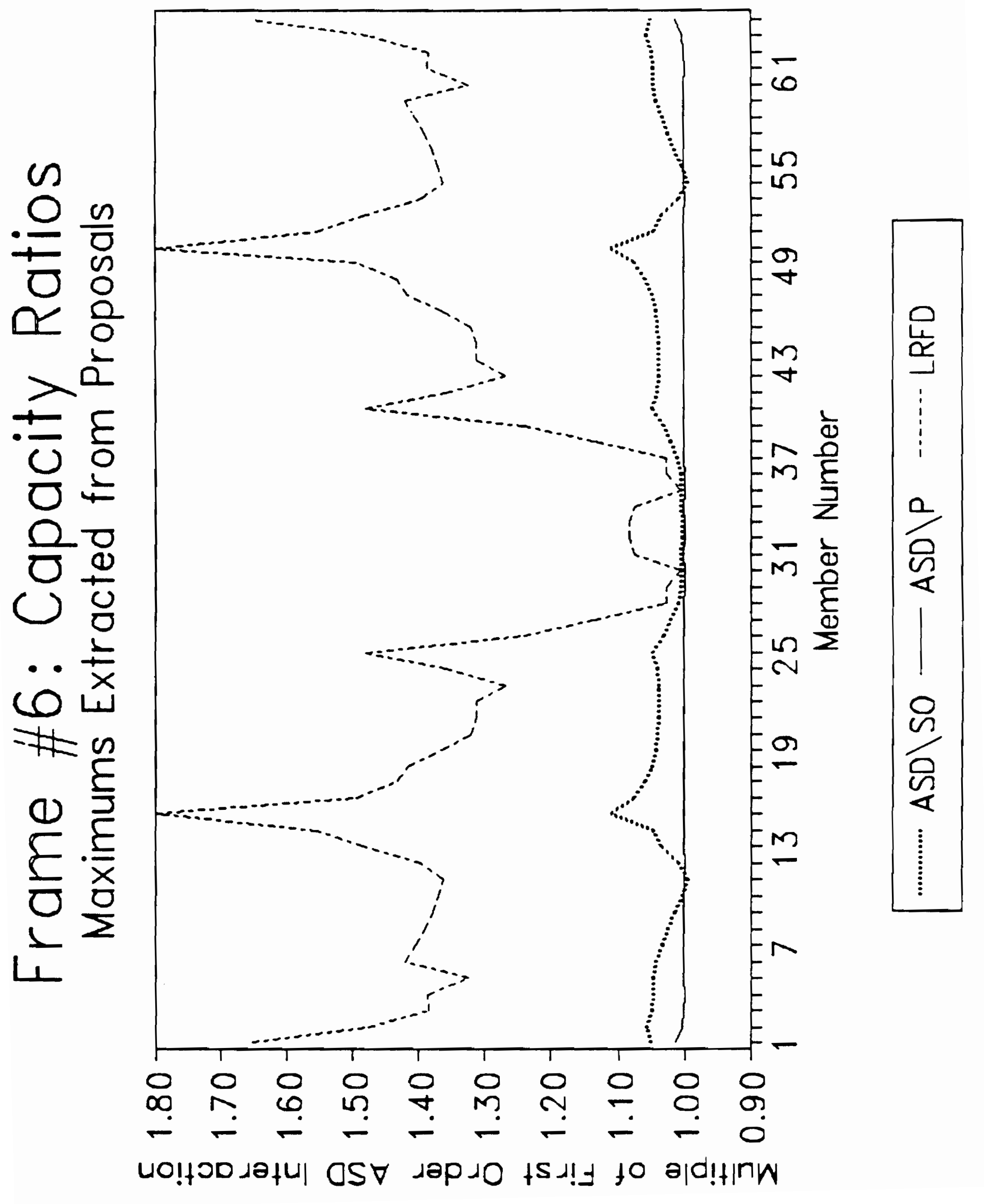


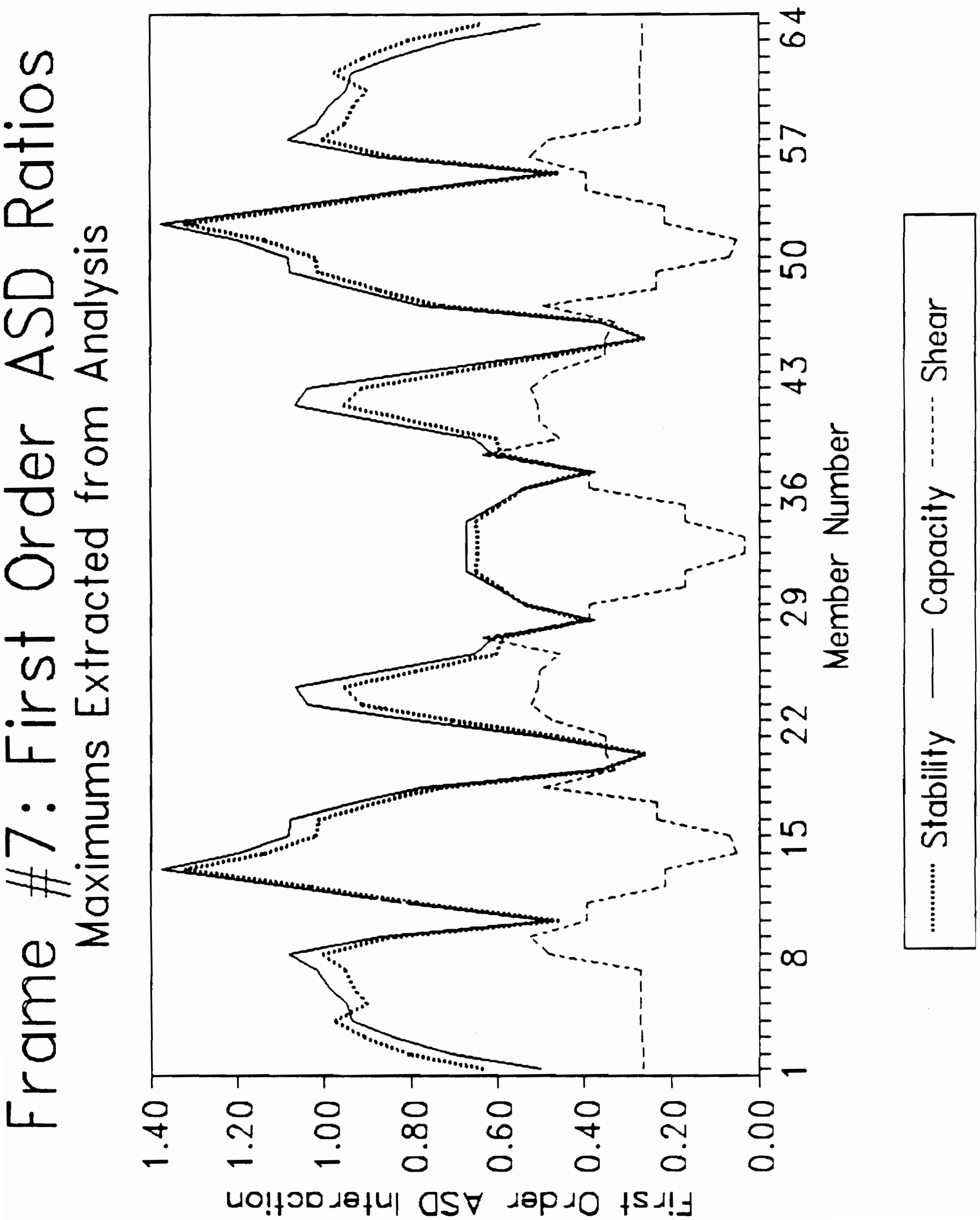
Frame #5: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
1
7
13
19
25
31
37
43
49
55
61
67
73
Member Number
ASD\SO
ASD\P
LRFD

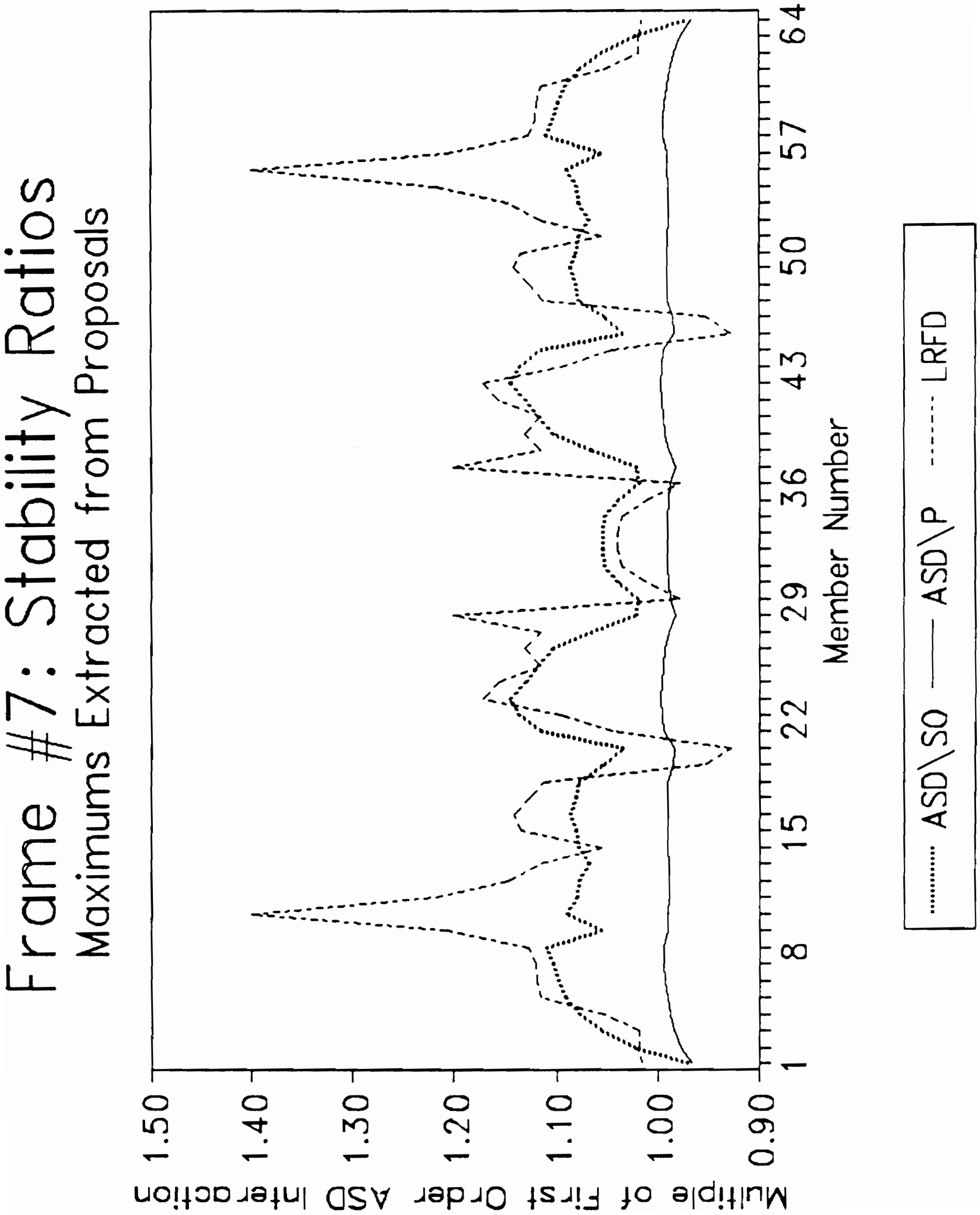

Frame #5: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
1
7
13
19
25
31
37
43
49
55
61
67
73
Member Number
ASD\SO
ASD\P
LRFD

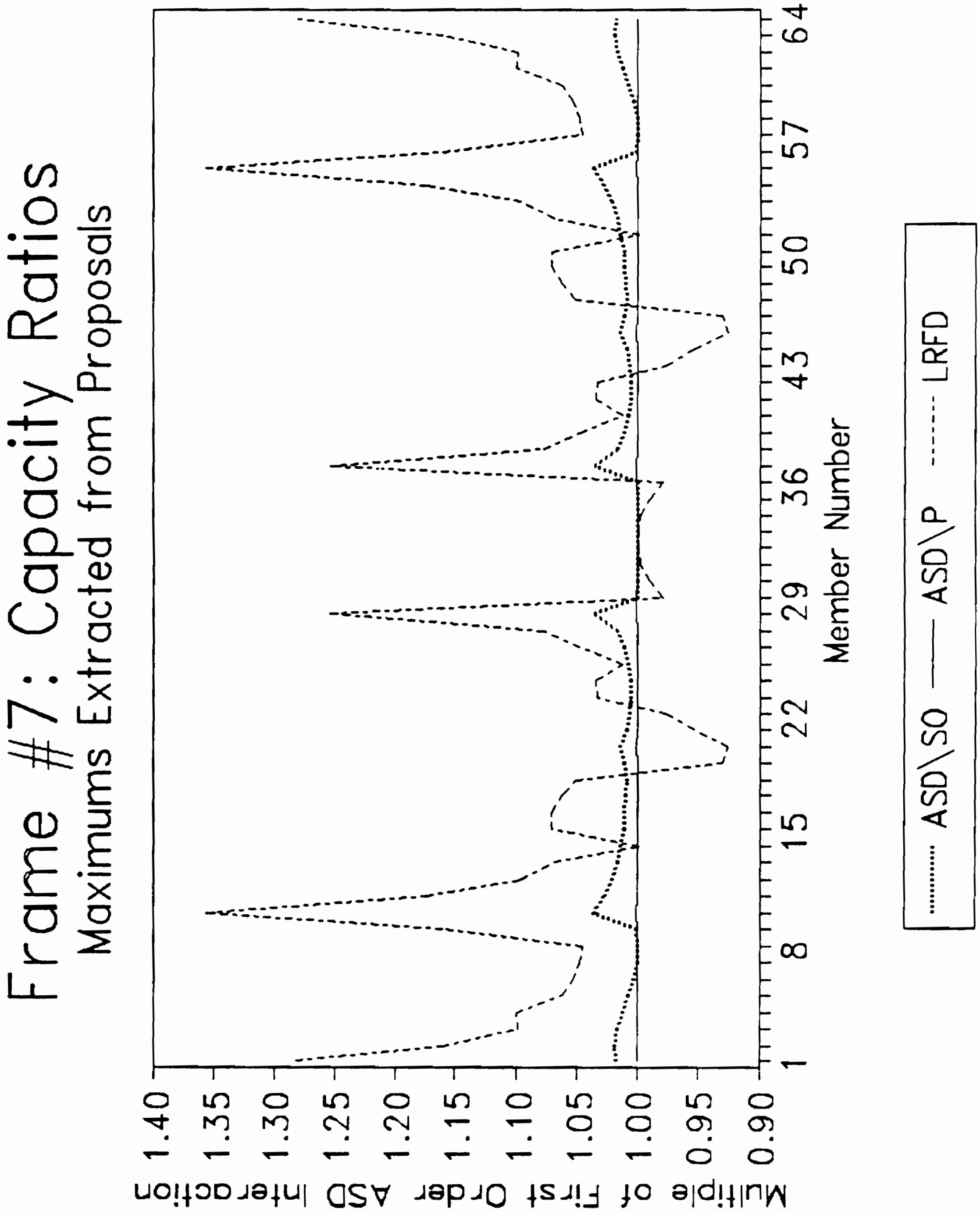

Frame #6: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1
7
13
19
25
31
37
43
49
55
61
Member Number
Stability
Capacity
Shear

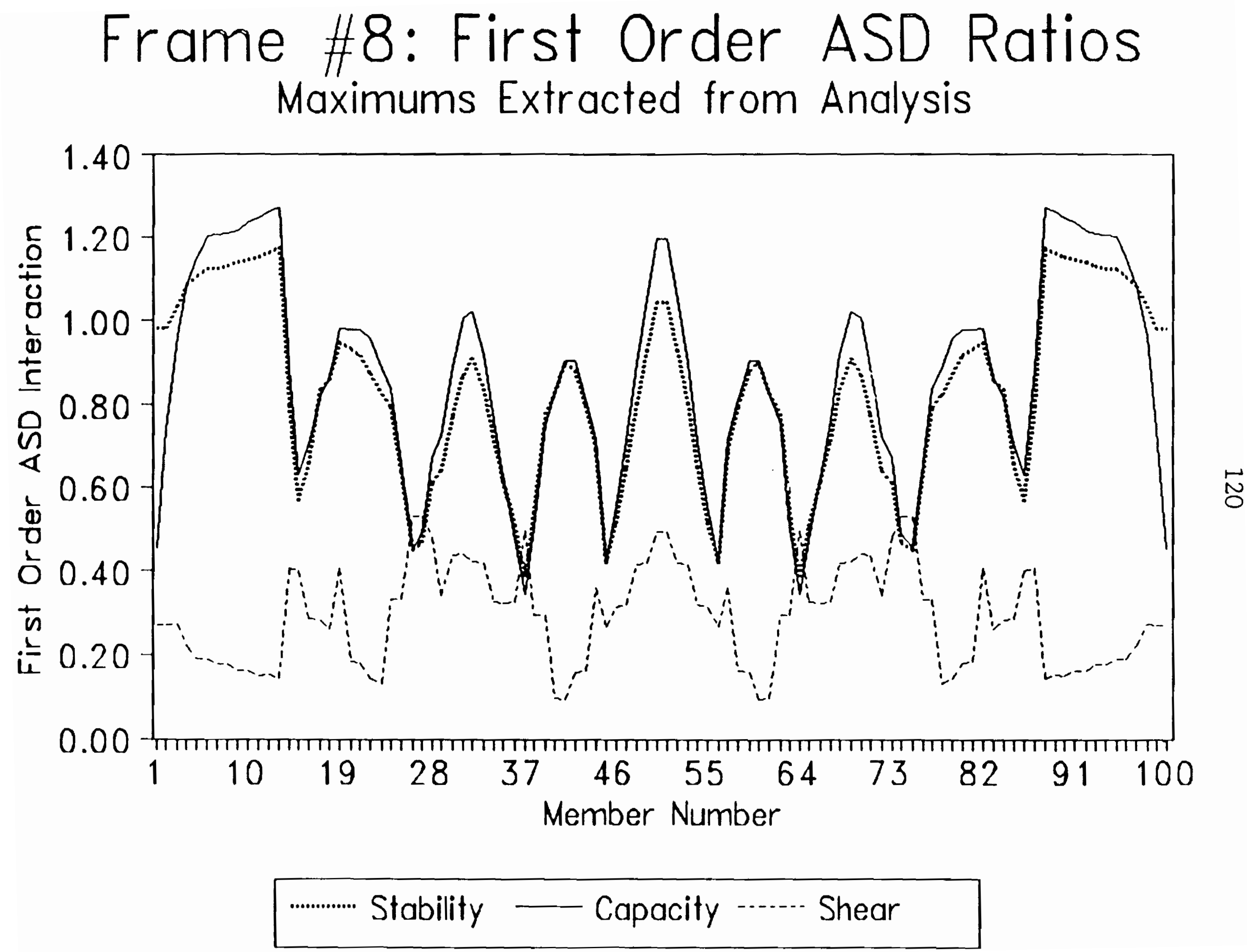

Frame #6: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
1
7
13
19
25
31
37
43
49
55
61
Member Number
ASD\SO
ASD\P
LRFD

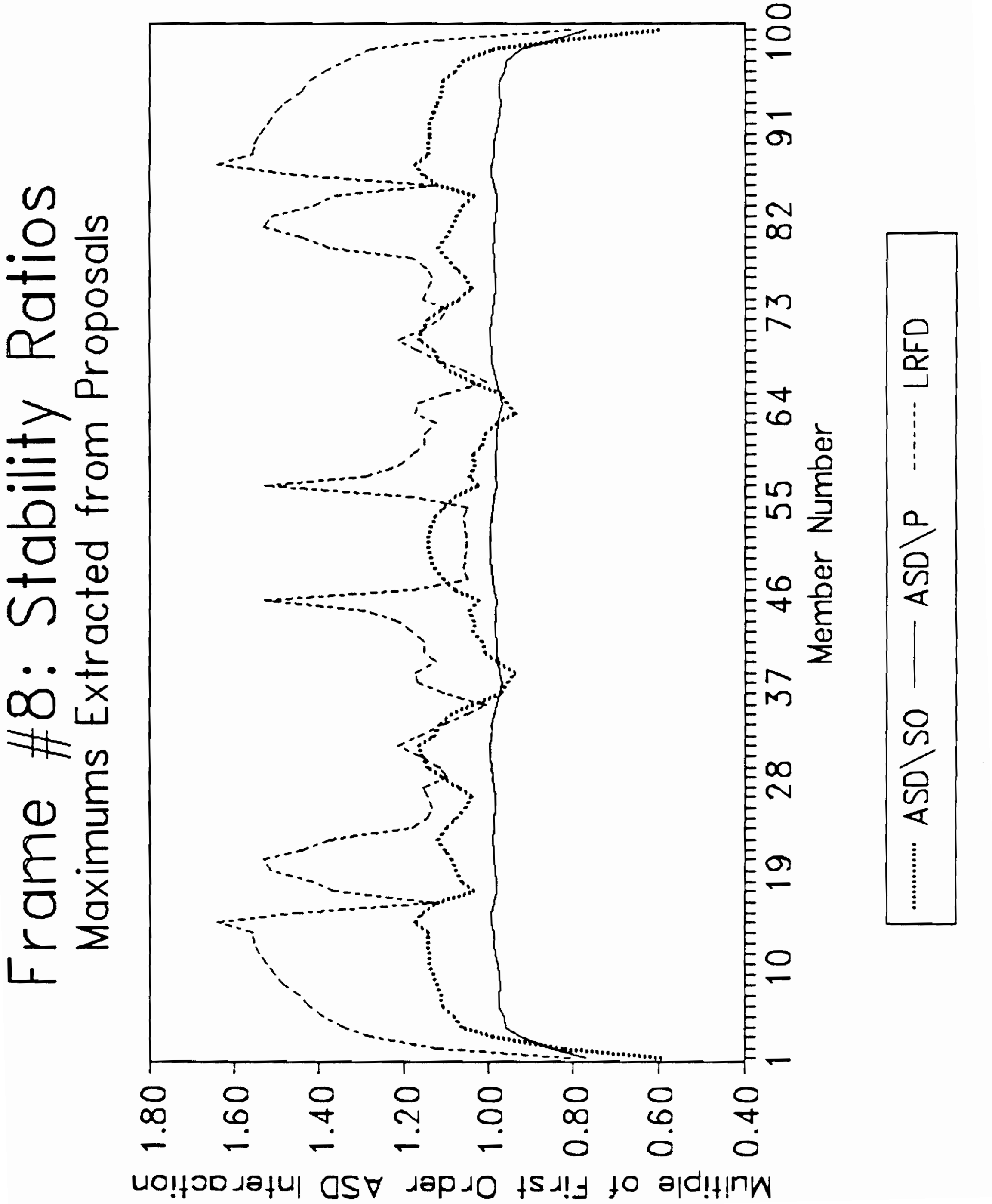

Frame #6: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.80
1.70
1.60
1.50
1.40
1.30
1.20
1.10
1.00
0.90
1
7
13
19
25
31
37
43
49
55
61
Member Number
ASD\SO
ASD\P
LRFD

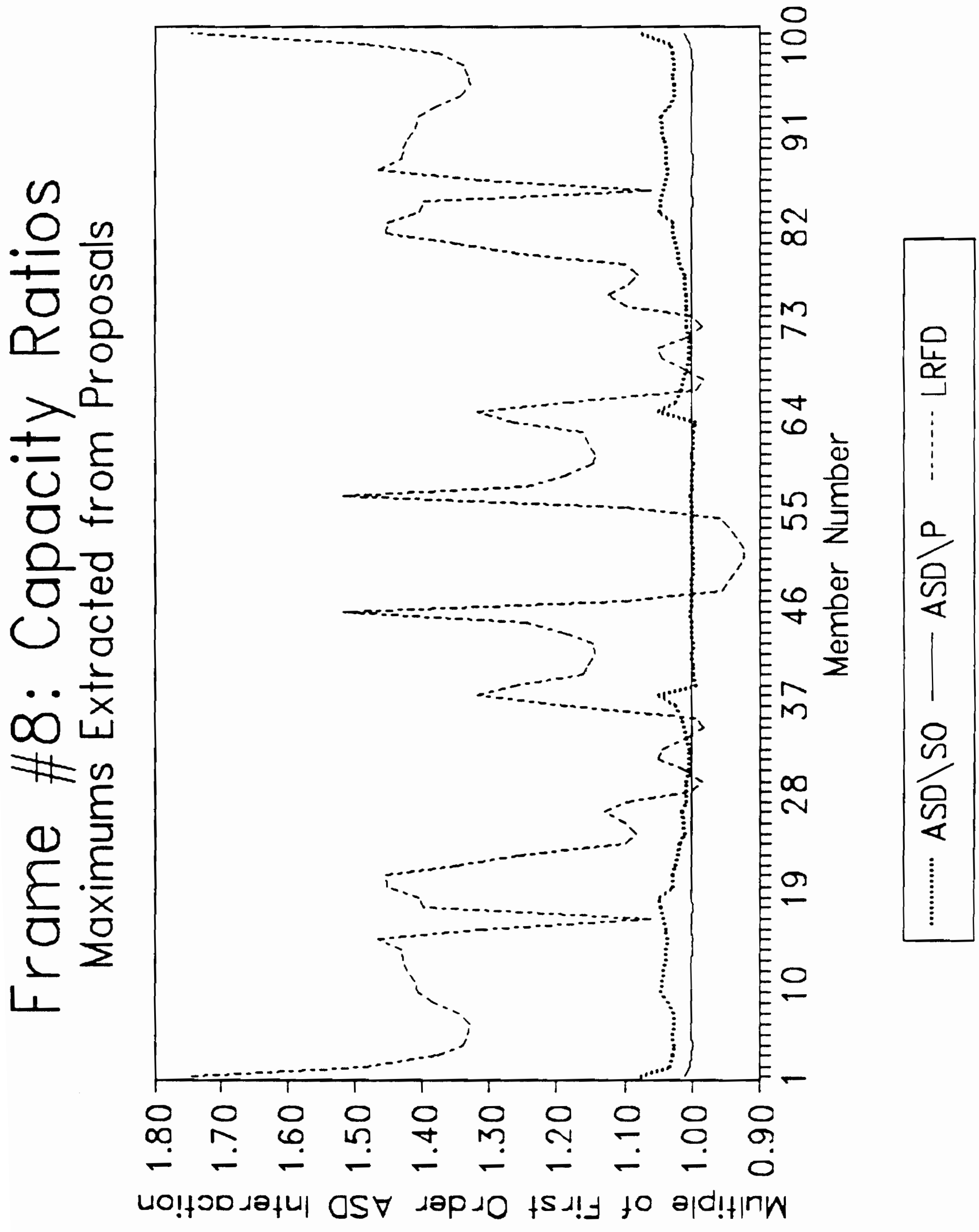

Frame #7: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1
8
15
22
29
36
43
50
57
64
Member Number
Stability
Capacity
Shear

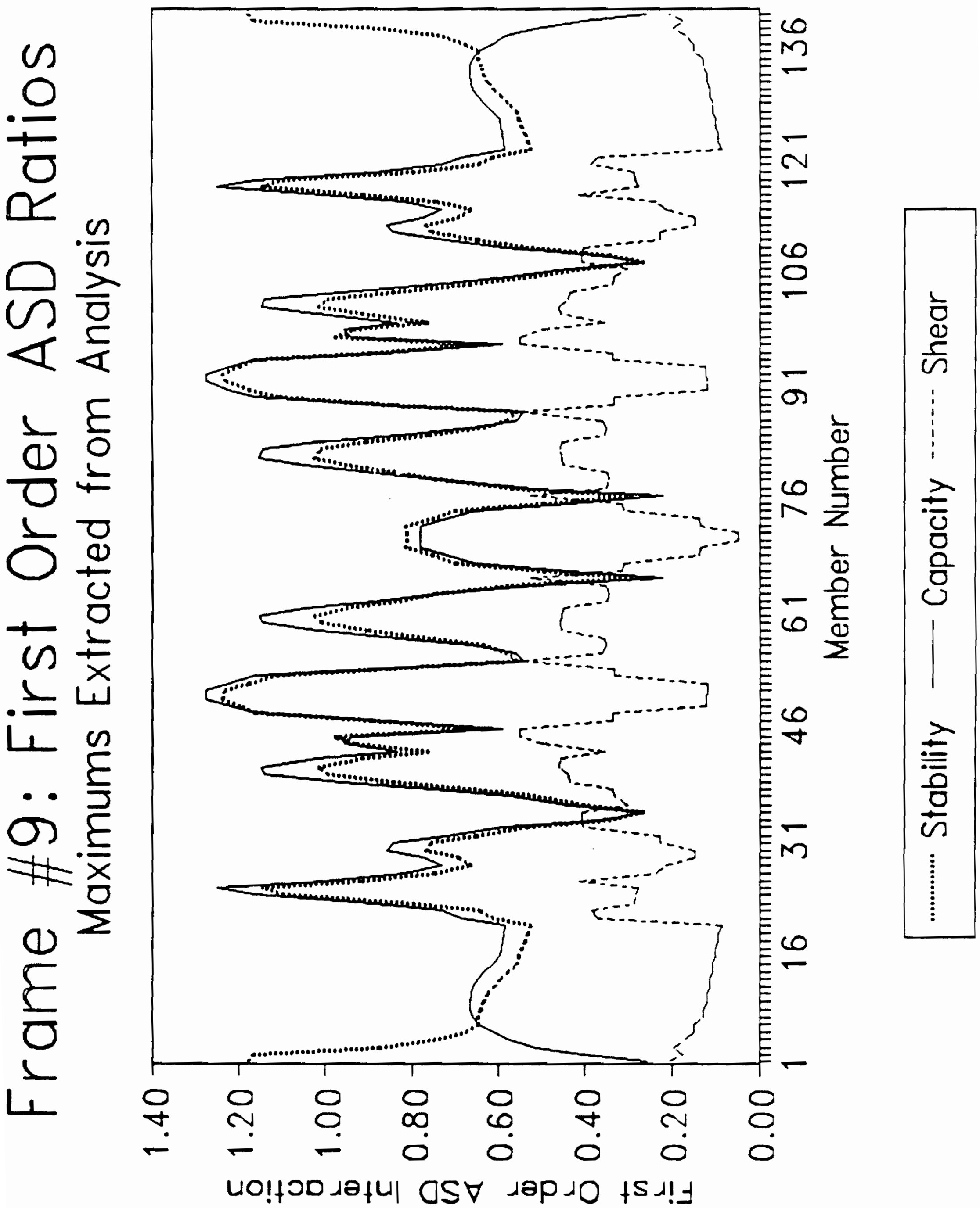

Frame #7: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.50
1.40
1.30
1.20
1.10
1.00
0.90
1
8
15
22
29
36
43
50
57
64
Member Number
ASD\SO
ASD\P
LRFD

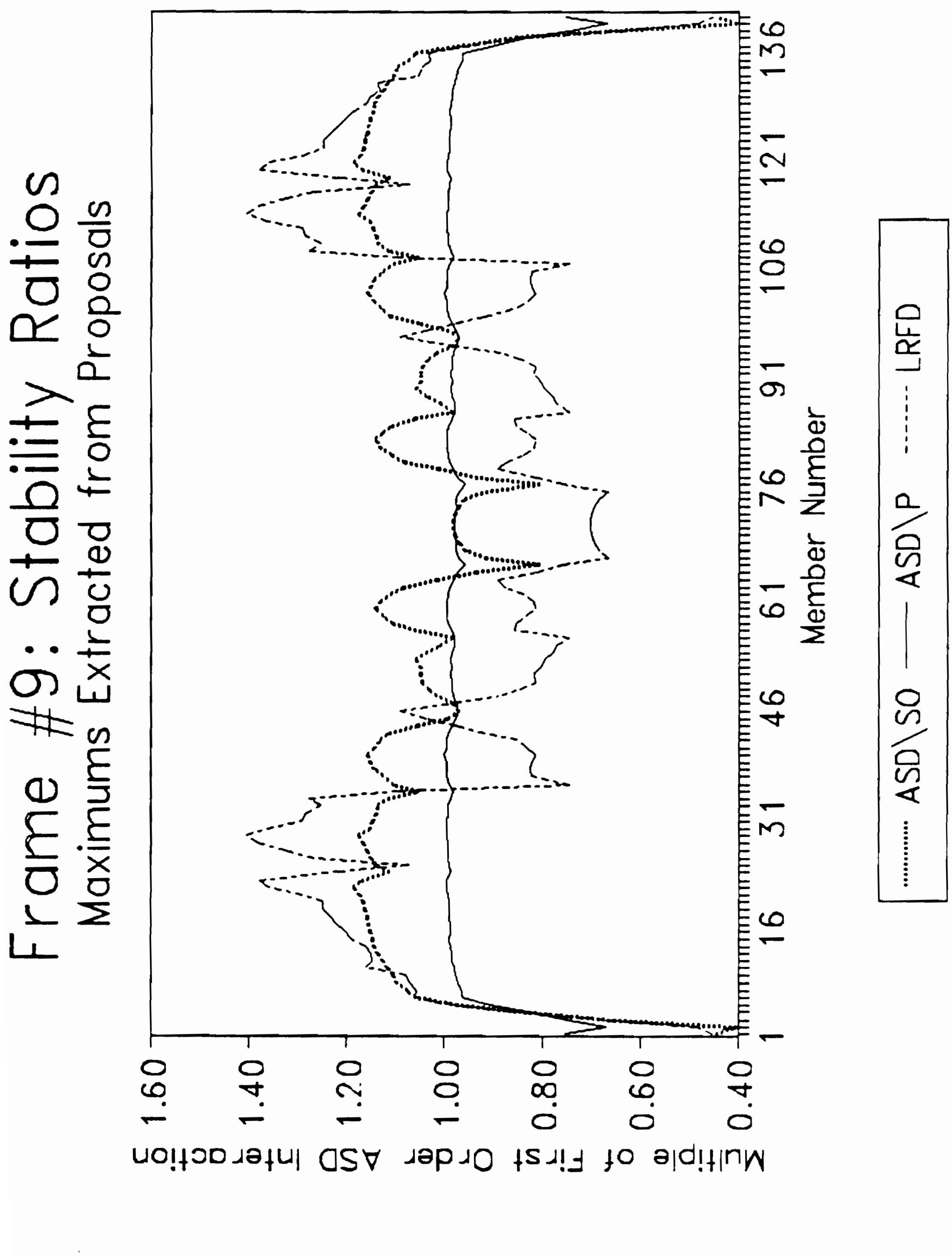

Frame #7: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.40
1.35
1.30
1.25
1.20
1.15
1.10
1.05
1.00
0.95
0.90
Member Number
1
8
15
22
29
36
43
50
57
64
ASD\SO
ASD\P
LRFD

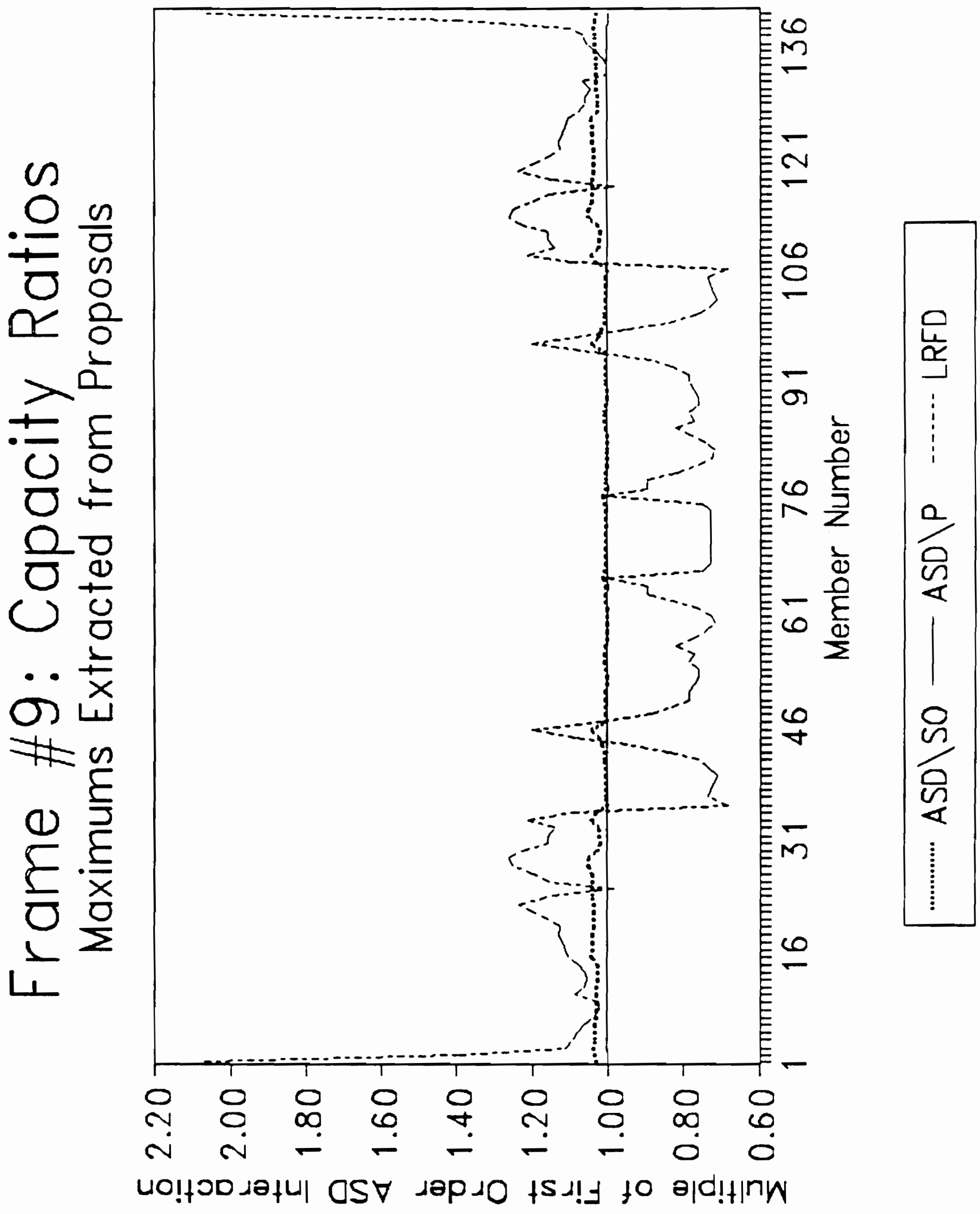

Frame #8: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1
10
19
28
37
46
55
64
73
82
91
100
Member Number
Stability
Capacity
Shear

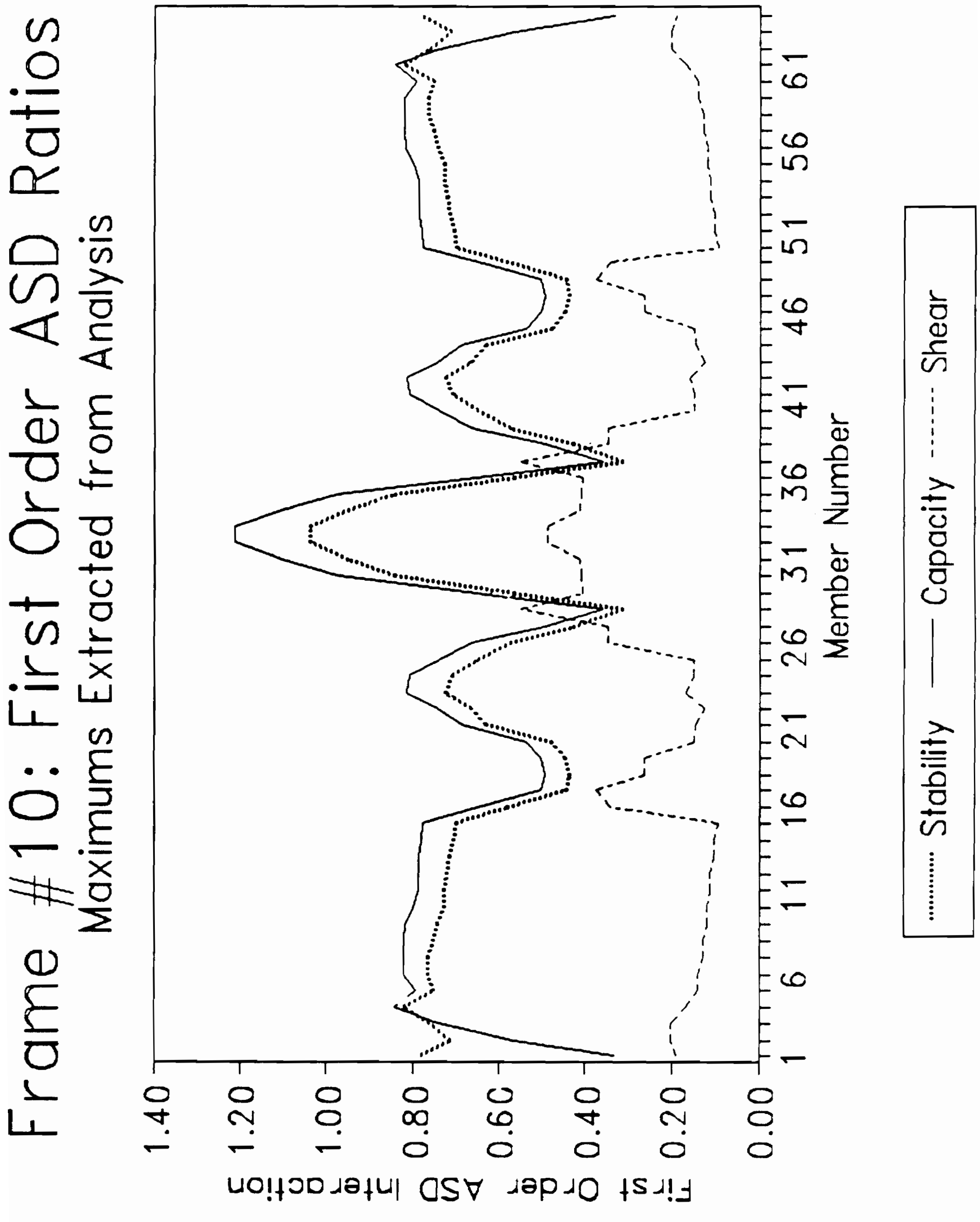

Frame #8: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
1
10
19
28
37
46
55
64
73
82
91
100
Member Number
ASD\SO
ASD\P
LRFD

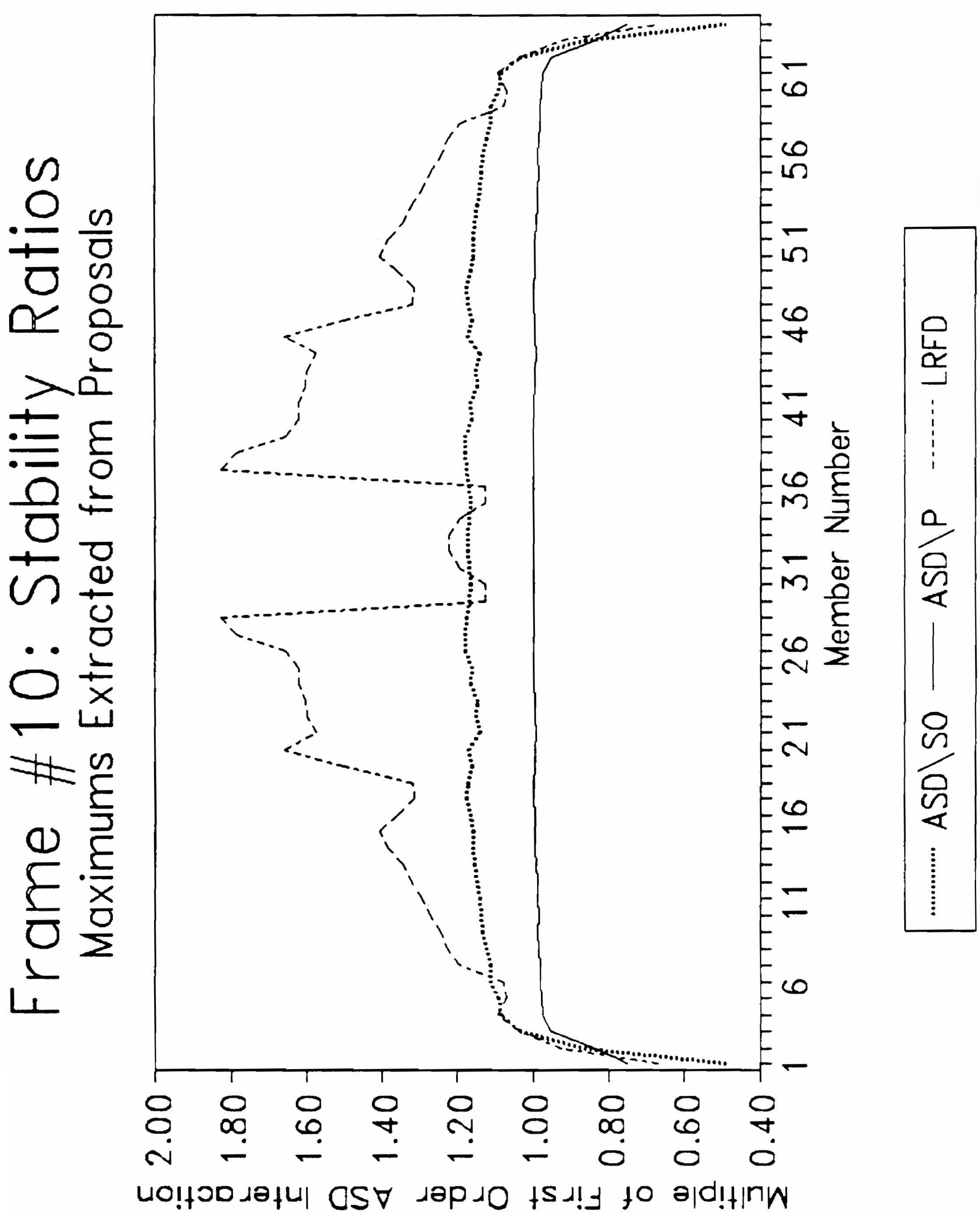

Frame #8: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.80
1.70
1.60
1.50
1.40
1.30
1.20
1.10
1.00
0.90
1
10
19
28
37
46
55
64
73
82
91
100
Member Number
ASD\SO
ASD\P
LRFD

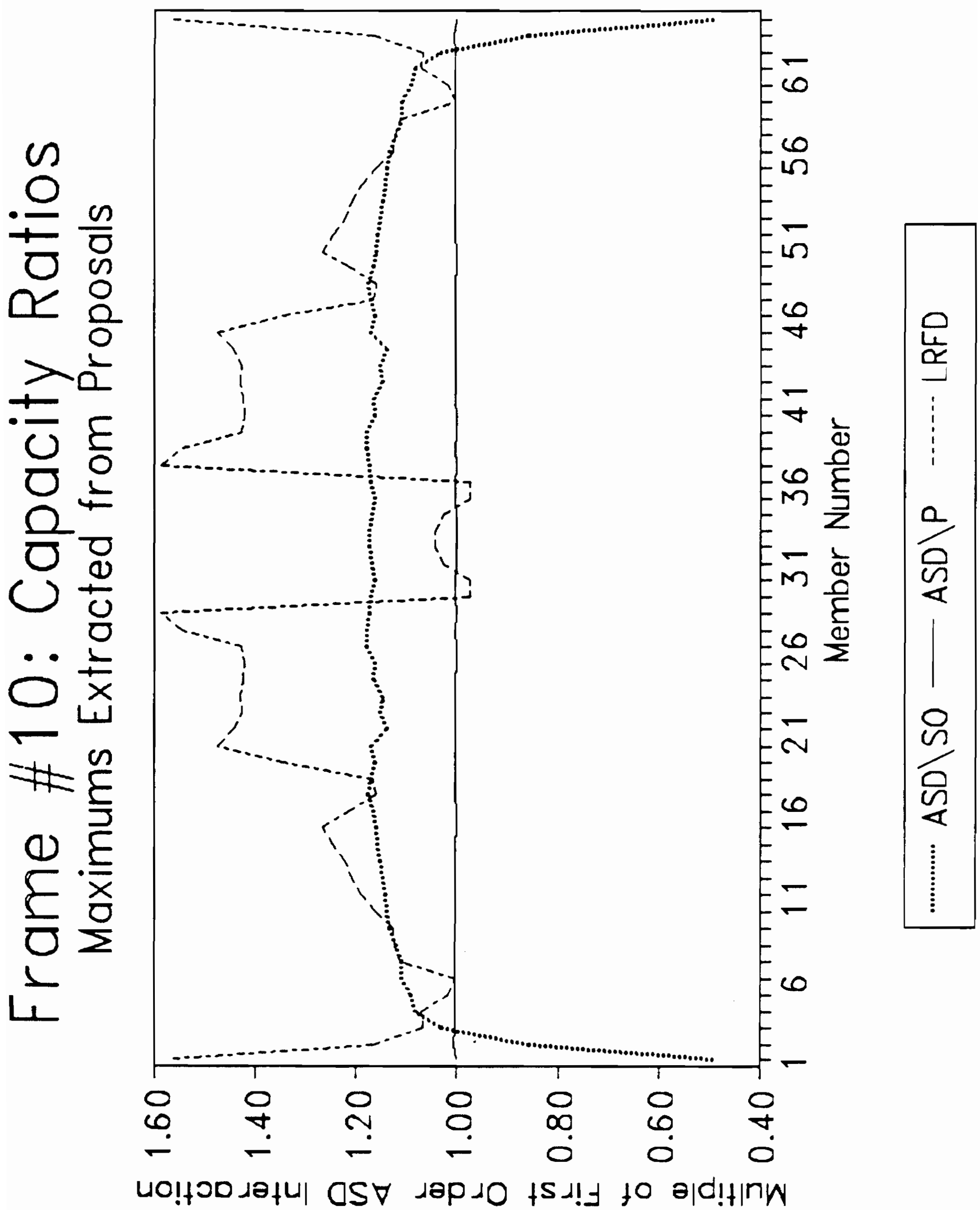

Frame #9: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1
16
31
46
61
76
91
106
121
136
Member Number
Stability
Capacity
Shear


Frame #9: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.60
1.40
1.20
1.00
0.80
0.60
0.40
1
16
31
46
61
76
91
106
121
136
Member Number
ASD\SO
ASD\P
LRFD


Frame #9: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
1
16
31
46
61
76
91
106
121
136
Member Number
ASD\SO
ASD\P
LRFD


Frame #10: First Order ASD Ratios
Maximums Extracted from Analysis
First Order ASD Interaction
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1
6
11
16
21
26
31
36
41
46
51
56
61
Member Number
Stability
Capacity
Shear


Frame #10: Stability Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
1
6
11
16
21
26
31
36
41
46
51
56
61
Member Number
ASD\SO
ASD\P
LRFD

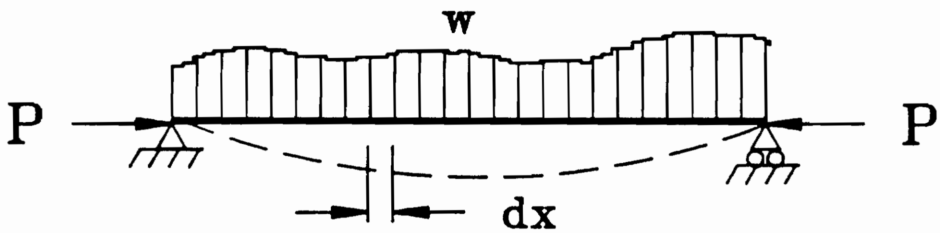

Frame #10: Capacity Ratios
Maximums Extracted from Proposals
Multiple of First Order ASD Interaction
1.60
1.40
1.20
1.00
0.80
0.60
0.40
Member Number
1
6
11
16
21
26
31
36
41
46
51
56
61
ASD\SO
ASD\P
LRFD

APPENDIX B

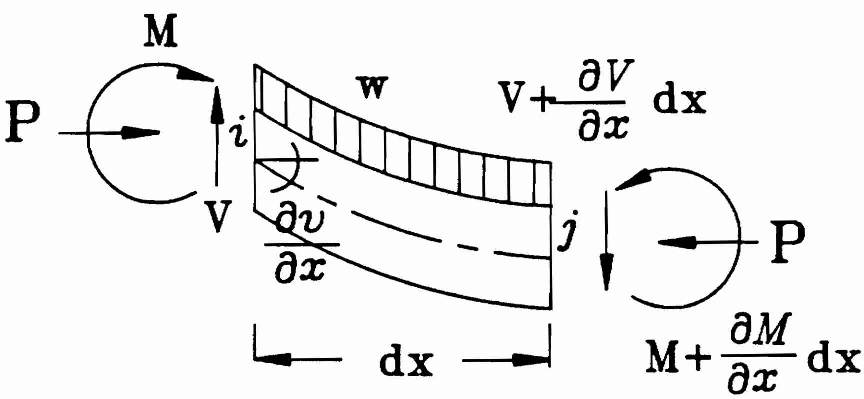
BEAM-COLUMN DIFFERENTIAL EQUATION DERIVATION

Differential Equation Derivation

1. Beam–Column w/ Transverse Loading



2. Differential Segment



3. Axial Force Influence

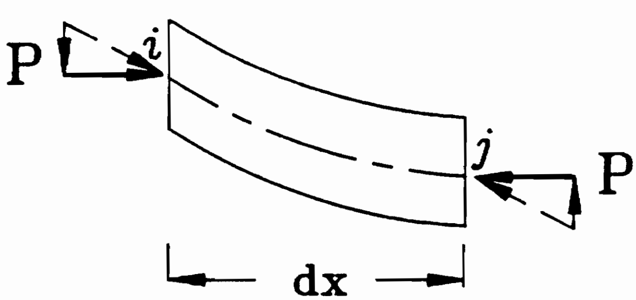


Figure B.1

Differential Equation Derivation

4. Equilibrium of Differential Segment:

$$M + \left(M + \frac{\partial M}{\partial x} dx\right) + w\frac{(dx)^2}{2} + \left(V + \frac{\partial V}{\partial x} dx\right) + P\left(\frac{\partial v}{\partial x}\right) dx = 0$$

Neglecting terms in dx^2 and differentiating,

$$\frac{\partial^2 M}{\partial x^2} - \frac{\partial V}{\partial x} - P\frac{\partial^2 v}{\partial x^2} = 0$$

For Vertical Equilibrium,

$$\frac{\partial V}{\partial x} - w = 0$$

5. From Kinematics,

$$M = -EI\frac{\partial^2 v}{\partial x^2}$$

6. The Differential Equation!

$$EI\frac{\partial^4 v}{\partial x^4} = w - P\frac{\partial^2 v}{\partial x^2}$$

Which may be solved by the matrix displacement method. The element model is developed as follows;

7. Flexural Strain Energy

$$U_b = \int_0^L \frac{EI}{2}\left(\frac{\partial^2 v}{\partial x^2}\right)^2 dx$$

Figure B.1 – continued

Differential Equation Derivation

8. Solution of Equation

Neglecting the axial force, the beam solution is

$$\frac{\partial^4 v}{\partial x^4} = 0$$

$$v(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3$$

9. Selecting Nodal Displacements,

$$v(x) = f_1(x) v_1 + f_2(x)\Theta_1 + f_3(x) v_2 + f_4(x)\Theta_2$$

$$\frac{\partial^2 v}{\partial x^2} = f_1''(x) v_1 + f_2''(x)\Theta_1 + f_3''(x) v_2 + f_4''(x)\Theta_2$$

The influence coefficients are found by

$$\mathbf{k}_{ij} = \mathbf{EI}\int_0^L f_i''(x) f_j''(x)\,\mathrm{dx}$$

Notes:
1. Neglects Axial Deformation and Second-Order Effects
2. {f} is determined from the Element Boundary Conditions Coefficients a_i are matched to Nodal DOFs to form the required Shape Functions {f}.

10. Modification for Beam-Column

To account for second-order effects, the influence of the axial load on the bending stiffness coefficients is determined by factoring the flexural stiffness coefficients (derived above) by the phi (ϕ) factors shown in Chapter III (derived from the Stability Functions).

Figure B.1 – continued

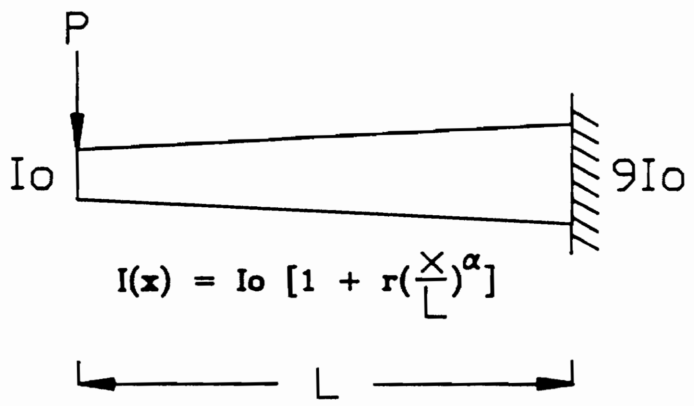
APPENDIX C
TAPERED MEMBER CONVERGENCE STUDY

C.1 Beam Behavior

Obviously, a stepped beam analysis is an approximate estimate of the continuum model of the tapered beam. Although first-order elastic analysis does not require iteration, this application of the displacement method is technically a convergence problem. The use of a stepped beam model is a transition from the realm of the matrix displacement method *(MDM)* into the world of finite element analysis *(FEA)*. As a consequence of discretization, error in analysis of the displacements is introduced. The magnitude of these errors depends on the similitude of the model and the continuum.

The more elements that are used in the analysis, the closer to the exact solution. However, as the number of elements increases, so does the time required to perform the analysis. A typical tapered beam is studied in Table C.1. This example is taken from Yang (1986). This tapered cantilever shows that eight uniform elements were required to reduce the error in the tip displacements to below 1%. Consequently, the error is less than 1% using only two tapered elements. The exact solution may be converged upon by selecting a higher degree of freedom discretization, as Table C.1 indicates. This convergence process is known as *monotonic convergence from below.* If a plot was constructed with a displacement as the vertical axis and the number of elements as the horizontal axis, the exact solution would be

Error Analysis of Uniform Element Approach to Tapered Beam



Parameters

$\alpha = 1$
$r = 8$

Number of Elements	Uniform Elements: Error in Tip Deflection (%)	Uniform Elements: Error in Tip Slope (%)	Tapered Elements: Error in Tip Deflection (%)	Tapered Elements: Error in Tip Slope (%)
1	30.30	10.29	−0.62	2.82
2	8.58	5.04	−0.17	0.77
3	3.92	2.99		
4	2.22	1.97		
6	0.98	1.02		
8	0.55	0.62		
12	0.24	0.30		
16	0.14	0.17		

Table C.1

approached asymptotically as the chosen number of degrees of freedom increases. Such a graph is presented later. This can be explained by the fact that in an FEA, the model always possesses a greater degree of *stiffness* than the continuum, because of the discretization.

The NUCOR software used for analysis and design of rigid frames generally specifies a *joint* or *nodal* coordinate every 2.5 ft. For a tapered beam of length 20 ft., this results in eight elements for the discretization. By Yang's analysis, the analysis should possess a maximum error under one percent. A positive value of the percent error signifies that the displacement has been underestimated.

C.2 Beam-Column Behavior

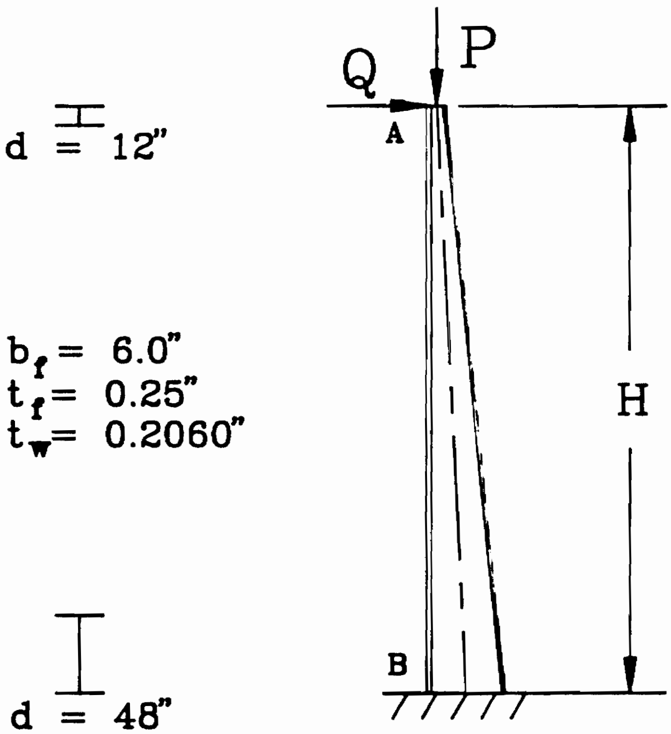
A typical NUCOR beam-column has a taper ratio (γ) of 3.0, and a vertical height of 30 ft (γ = (depth of larger end - depth of smaller end)/depth of smaller end). It also possesses representative cross-sectional dimensions of a 12 in. depth at the base, flange widths and thicknesses of 6.0 in. and 0.25 in. respectively, and a web thickness of 0.2060 in. The ratio of the larger end moment of inertia to that of the smaller end is approximately twelve. These dimensions were selected to represent the type of construction normally used by NUCOR frames. The boundary conditions were selected to maximize errors in the analysis, as the free end is subjected to a vertical and horizontal load and the larger end is fixed. The loads are typical to a combination case of dead and snow loads. Though these boundary conditions are not representative of NUCOR frame construction (an individual fix/free column), the errors involved in a frame analysis should be of the same order. A NUCOR beam-column would have the smaller end pinned (allowing rotation but no displacement), and the larger end rigidly connected to a rafter (allowing both rotation and displacement, but the angle between the column and rafter remains unchanged). The analysis is performed using the proposed P-Delta program developed in this study, and the results are shown in a form similar to that of Yang, described above.

A NUCOR software discretization would have used approximately 30.0 ft / 2.5 ft/element = 12 elements. By Table C.2, the error in deformations at the tip for lateral and rotational displacements would be on the order of 15%. Correspondingly, for second-order elastic beam-column behavior, the same discretization would also produce an error on the order of 15%. With this amount of error in discretizations, stress calculations would be unconservative by this same amount by first-order analysis. To obtain a solution with less than 1%, 128 elements would be required. Figure C.1 shows the convergence study plots for this beam-column, for the tip-displacement and base moment. Because of the discretization error, the NUCOR software should be called an FEA program, not an MDM program.

For practical reasons, it is not recommended to model a tapered column with a uniform element every six inches. What is recommended is that the software use more elements than currently employed, and recognition of the amount of error in the discretization process. This error is probably on the order of 10%. The best recommendation is to utilize tapered elements instead of uniform elements for the analysis. With tapered elements, the current discretization could be held, and the accuracy of the results improved. Figure C.2 (Galambos, 1976) could be used as a guide in the development of a

Tapered Beam-Column Modeling with Uniform Elements

Convergence Study

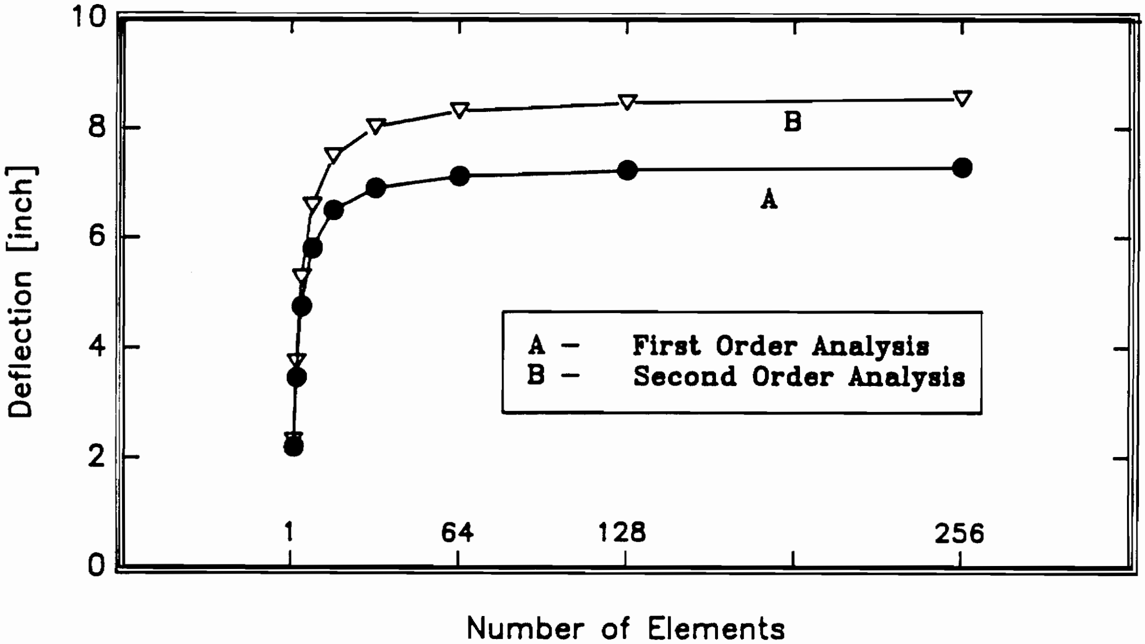


Parameters

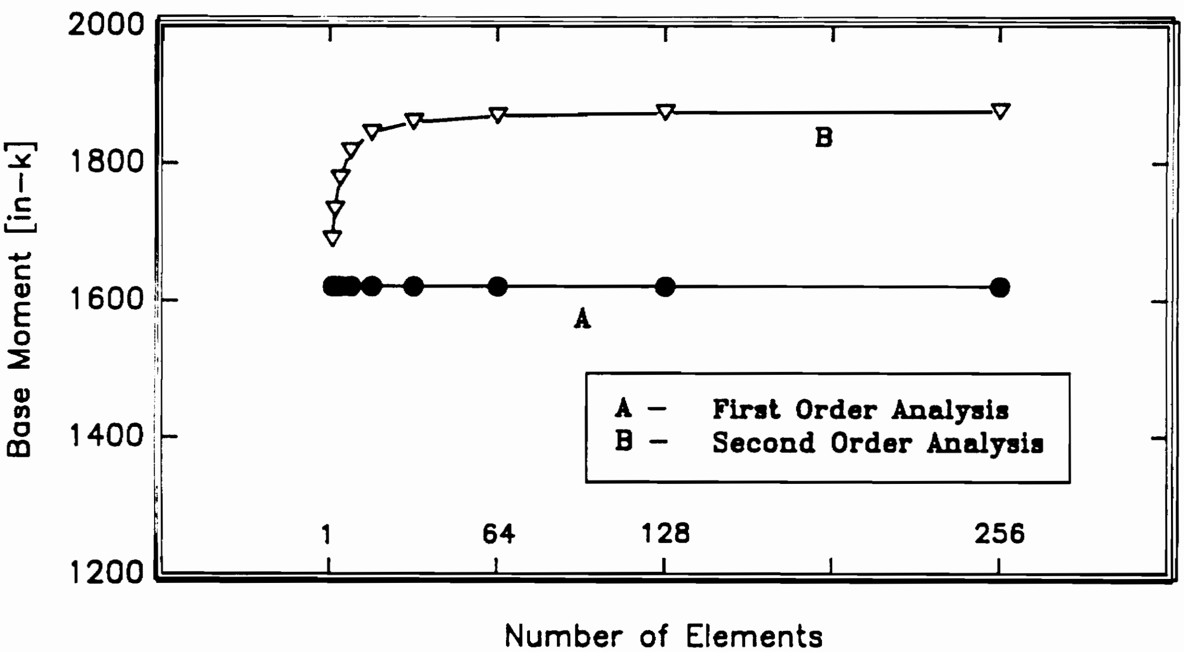
P = 30 kips
Q = 6 kips
H = 30 ft

Number of Elements	First Order Analysis: Error in Tip Deflection (%)	First Order Analysis: Error in Tip Slope (%)	Second Order Analysis: Error in Tip Deflection (%)	Second Order Analysis: Error in Tip Slope (%)
1	69.93	62.79	73.03	66.62
2	52.61	46.32	56.30	50.50
4	34.84	30.66	38.29	34.33
8	20.39	15.21	22.93	20.65
16	10.82	9.62	12.36	11.19
32	5.29	4.51	6.10	5.54
64	2.32	2.07	2.69	2.45
128	0.78	0.70	0.91	0.83
256	0.00	0.00	0.00	0.00

Table C.2



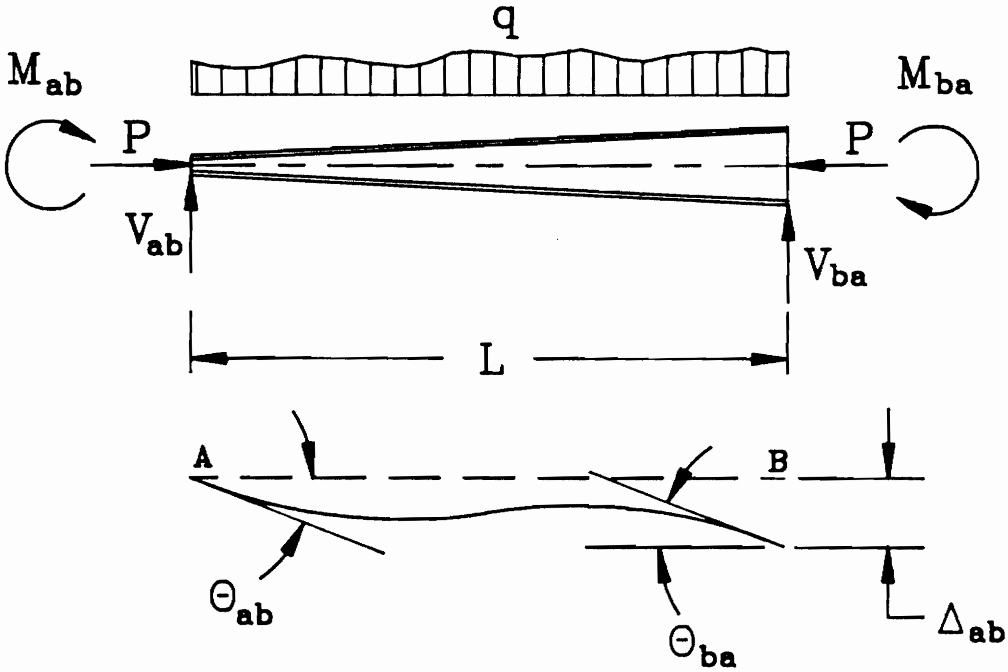
a. Convergence in Lateral Displacement



b. Convergence in Base Moment

Figure C.1

Sign Convention used in the following derivation;



Slope Deflection Equation for Tapered Members:

$$\Theta_{ab} = \frac{M_{ab}L}{EI_o}C_{aa} - \frac{M_{ba}L}{EI_o}C_{ab} + \frac{\Delta_{ab}}{L} + \bar{\Theta}_{ab}$$

$$\Theta_{ba} = -\frac{M_{ab}L}{EI_o}C_{ba} + \frac{M_{ba}L}{EI_o}C_{bb} + \frac{\Delta_{ab}}{L} - \bar{\Theta}_{ba}$$

Θ_{ab}, Θ_{ba} **End Rotations of Tangents to the Elastic Curve**

$\bar{\Theta}_{ab}, \bar{\Theta}_{ba}$ **End Rotations of Tangents to the Elastic Curve for a Simply Supported Beam**

C_{ij} **Unit Moment/Rotation Coefficients for Tapered Beam (functions of geometry and loading).**

Tapered Beam-Column Element

Figure C.2

$$a = \frac{\pi^2 P}{P_{exo}}$$

$$P_{exo} = \frac{\pi^2 EI_0}{L^2}$$

$$\gamma = \frac{d_1 - d_0}{d_0}$$

$$m = (1+\gamma)^{\rho}$$

$$n = (1+\gamma)^{-\rho}$$

$$\rho = \sqrt{\frac{1}{4} - \frac{a}{\gamma^2}}$$

$$\beta = \sqrt{\frac{a}{\gamma^2} - \frac{1}{4}}$$

For: $a/\gamma^2 \leq 1/4$

$$C_{aa} = \frac{1}{a} - \frac{\gamma}{a}\left(\rho\frac{m+n}{m-n} - \frac{1}{2}\right)$$

$$C_{ab} = C_{ba} = \frac{\gamma}{a\sqrt{1+\gamma}}\frac{2\rho}{m-n} + \frac{1}{a}$$

$$C_{bb} = \frac{1}{a} - \frac{\gamma}{a(1+\gamma)}\left(\rho\frac{m+n}{m-n} + \frac{1}{2}\right)$$

For: $a/\gamma^2 > 1/4$

$$C_{aa} = \frac{1}{a} - \frac{\gamma}{a}\beta\cot(\beta\ln(1+\gamma)) + \frac{\gamma}{2a}$$

$$C_{ab} = C_{ba} = \frac{\gamma}{a\sqrt{1+\gamma}}\beta\csc(\beta\ln(1+\gamma)) - \frac{1}{a}$$

$$C_{bb} = \frac{1}{a} - \frac{\gamma}{a(1+\gamma)}\beta\cot(\beta\ln(1+\gamma)) - \frac{\gamma}{2a(1+\gamma)}$$

Tapered Beam-Column Element

Figure C.2 – continued

tapered beam-column element, though there are other tapered element models possible.

The frames that were studied in the research used the same discretizations that NUCOR software had used. Therefore, the error in the analysis results of the research are of the same order as originally introduced by NUCOR.

APPENDIX D
SLENDER MEMBER DESIGN

D.1 Axial Capacity

D.1.1 General

The development of axial compressive strength for slender I cross-sections depends on both the maximum slenderness ratio for overall buckling, and the level of stress on the cross-section for local buckling. The AISC Specification provisions governing axial capacity were borrowed from the 1968 AISI Specification provisions for cold-formed steel design with some modifications. The design equations for column strength in the 1968 AISI Specification, for load determination analysis, are as follows:

For

$$KL/r < \frac{C_c}{\sqrt{Q}}$$

$$F_{a1} = \frac{12QF_y}{23} - \frac{3(QF_y)^2(\frac{KL}{r})^2}{23\pi^2 E} \qquad (D.1)$$

For

$$KL/r < \frac{C_c}{\sqrt{Q}}$$

$$F_{a1} = \frac{12\pi^2 E}{23(\frac{KL}{r})^2} \qquad (D.2)$$

Where the *Q* form-factor is determined according to cross-section element slenderness ratios, which establish the maximum allowable stresses to prevent a local buckling failure before lateral buckling. This Q depends on whether the cross-section has slender stiffened and/or unstiffened elements. These AISI provisions form the basis of slender-cross-section design in the AISC Specifications. Depending on whether all or some elements of the cross-section are slender, application of the AISC provisions results in replacing the full cross-section with an effective cross-section.

D.1.2 Unstiffened Elements Slender Only

For unstiffened elements (e.g., the flanges of an I-shape), the AISC provisions utilize a reduced average stress to account for softening due to local buckling. By the 1978 ASD or 1986 LRFD Specification, the Q_s factor reduces the average stress, according to its slenderness ratio ($b_f/2t_f$) and yield stress. By the 1989 ASD Specification, the Q_s factor depends not only on the element slenderness and yield stress, but also the web h/t ratio for a flange/web local buckling interaction effect. The 1989 AISC ASD equations for Q_s are as follows:

$$k_c = \frac{4.05}{(h/t)^{0.46}} \tag{D.3}$$

$$Q_s = 1.293 - 0.00309\,\frac{b_f}{2t_f}\,\sqrt{F_y/k_c} \tag{D.4}$$

$$Q_s = \frac{26{,}200\,k_c}{[F_y(\frac{b_f}{2t_f})]^2} \tag{D.5}$$

These Q_s factors are defined as the ratio of the local buckling strength to the yield strength. This calculation is not iterative as it is not dependent on the level of applied stress. The effect of this factor can be seen in Figure D.1a. An example illustrating the effect on a cross-section when the unstiffened elements are slender is provided below.

Example: F_a Calculation, 1989 AISC ASD Specification

Given Data;

$KL/r = 45.0$ $A = 5.5\ in^2$
$F_y = 50\ ksi$ $Cc = 107.0$
$b_f = 7\ in$ $t_f = 0.25\ in$
$h_c = 8\ in$ $t_w = 0.25\ in$

Table B5.1, pg. 5-36

$h_c/t_w = 32.0 < (253/\sqrt{50} = 35.78)$; Noncompact
$h_c/t_w = 32.0 < 70$; $k_c = 1.0$
$b_f/2t_f = 14.0 > (95/\sqrt{50} = 13.44)$; A/B, $Q_s < 1.0$
$b_f/2t_f = 14.0 < (195/\sqrt{50} = 27.58)$; Eqn. A-B5-3

Appendix B

$Q_s = 1.293 - 0.00309(14.0)(7.07) = 0.9871$
$Q_a = 1.0$ (Web Noncompact)
$Q = Q_sQ_a = (0.9871)(1.0) = 0.9871$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.9871} = 107.7$

Table 3, pg. 5-119

$KL/rC'c = 0.4178$
$Ca = 0.503$

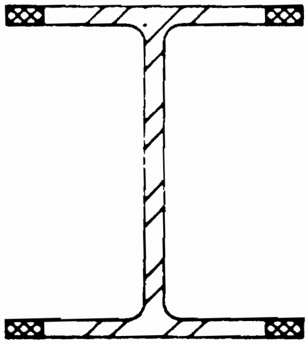
Allowable Compressive Stress

$F_a = CaQ_sQ_aF_y$
$F_a = (0.503)(0.9871)(1.0)(50\ ksi) = 24.83\ ksi$

$$\mathbf{F_a = 24.83\ ksi}$$

D.1.3 Stiffened Elements Slender Only

For stiffened elements (e.g., the web of an I-shape), the AISC provisions utilize an effective width concept. By the 1978, 1989 AISC ASD Specifications and

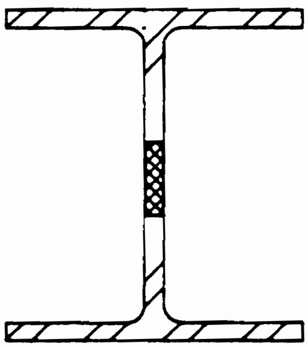


Equivalent Reduction of Flange Area

$Qa = 1.0$

$Qs < 1.0$

a) Slender Unstiffened Elements



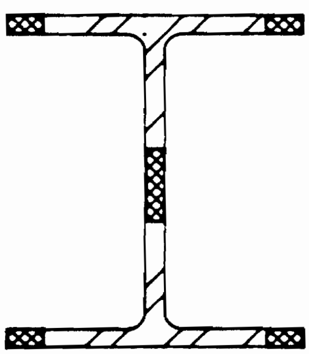
Deducted Area

$Qa < 1.0$

$Qs = 1.0$

$$b_{eff} = \frac{253t}{\sqrt{f}}\left[1 - \frac{44.3}{(b/t)\sqrt{f}}\right]$$

b) Slender Stiffened Elements



Deducted Area

$Qa < 1.0$

$Qs < 1.0$

c) Slender Unstiffened & Stiffened Elements

Reduction Factor Analysis

Figure D.1

the 1986 AISC LRFD Specification, the form-factor Q_a modifies the cross-section area according to the web slenderness ratio (h/t_w) and the level of compression stress considered. This Q_a factor is defined as the ratio of the local buckling strength to the yield strength. According to the 1968 AISI Specification, the stress level used in computing the cross-section properties is the local buckling stress of the unstiffened elements, Q_sF_y. This is a conservative approach, and was not adopted by the AISC. In ASD, a lower bound on the reduction in effective width is set so that b_{eff} is not less than $253t/\sqrt{F_y}$. By the 1978 or 1989 AISC ASD Specifications, the effective width of the stiffened element is determined from (with a factor of safety of 1.66):

$$b_{eff} = \frac{253\, t_w}{\sqrt{f}} \left[1 - \frac{44.3}{(b/t)\sqrt{f}}\right] \tag{D.6}$$

and

$$\textit{Minimum}\ b_{eff} = \frac{253\, t_w}{\sqrt{F_y}} \tag{D.7}$$

where $f = F_a$. Since F_a is unknown initially, an assumption of $0.6F_y$ may be used to obtain a lower bound on b_{eff}. Then, iterations are performed until b_{eff} and F_a converge to a constant value.

By the 1986 AISC LRFD Specification (factor of safety is removed):

$$b_{eff} = \frac{326\, t_w}{\sqrt{f}} \left[1 - \frac{57.3}{(b/t)\sqrt{f}}\right] \tag{D.8}$$

where $f = \phi P_n/A_g$. Since ϕP_n is unknown initially, an assumption of $0.85A_gF_y$ may be used to obtain a lower bound on b_{eff}. Then, iterations are performed until b_{eff} and ϕP_n converge to a constant value.

In LRFD, their is no minimum b_{eff} specified. However, it is assumed that LRFD intended to provide a *minimum* b_{eff} value. Logically, this value can be assumed to be:

$$\textit{Minimum}\ b_{eff} = \frac{326\, t_w}{\sqrt{F_y}} \tag{D.9}$$

The effect of the Q_a factor can be seen in Figure D.1b. The b_{eff} equation shown is taken from the AISC ASD Specifications. The effective widths calculated by this procedure are greater than those that would be obtained by directly applying the AISI provisions, as the level of stress f used is not necessarily the maximum level that can be applied to the unstiffened elements.

An example which illustrates the effective width concept when the stiffened element is slender is provided below.

Example: F_a Calculation, 1989 AISC ASD Specification

Given Data;

$KL/r = 45.0$ $A = 10.5\ in^2$
$F_y = 50$ ksi $Cc = 107.0$
$b_f = 6$ in $t_f = 0.375$ in
$h_c = 24$ in $t_w = 0.25$ in

Table B5.1, pg. 5-36

$h_c/t_w = 96.0 > (253/\sqrt{50} = 35.78)$; Slender A/B
$h_c/t_w = 96.0 > 70$; $k_c = 0.5663$
$b_f/2t_f = 8.0 < (95/\sqrt{(50/k_c)} = 10.11)$; Noncompact

<< Iteration 1 >>

Assume $f = 0.6F_yQ_s = 0.6(50\ ksi)(1.0) = 30\ ksi$
$Q_s = 1.0$
$b_{eff} = (253(.25)/\sqrt{30})\ [1-44.3/((96)\sqrt{30})] = 10.57"$
$Q_a = (10.5-(24-10.57)(.25))/10.5 = 0.6802$
$Q = Q_sQ_a = (1.0)(0.6802) = 0.6802$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.6802} = 129.7$
$KL/rC'c = 45.0/129.7 = 0.3470$
$Ca = 0.525$
$F_a = CaQ_sQ_aF_y$
$F_a = (0.525)(1.0)(0.6802)(50ksi) = 17.86\ ksi < 30\ ksi$

<< Iteration 2 >>

Assume $f = 17.86\ ksi$
$b_{eff} = (253(.25)/\sqrt{17.86})\ [1-44.3/(96\sqrt{17.86})] = 13.33"$
$Q_a = (10.5-(24-13.33)(.25))/10.5 = 0.7460$
$Q = Q_sQ_a = (1.0)(0.7460) = 0.7460$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.7460} = 123.9$
$KL/rC'c = 45.0/123.9 = 0.3632$
$Ca = 0.520$
$F_a = CaQ_sQ_aF_y$
$F_a = (0.520)(1.0)(0.7460)(50ksi) = 19.40\ ksi > 17.86\ ksi$

<< Iteration 3 >>

Assume $f = 19.40\ ksi$
$b_{eff} = (253(.25)/\sqrt{19.40})\ [1-44.3/(96\sqrt{19.40})] = 12.86"$
$Q_a = (10.5-(24-12.86)(.25))/10.5 = 0.7348$
$Q = Q_sQ_a = (1.0)(0.7348) = 0.7348$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.7348} = 124.8$
$KL/rC'c = 45.0/124.8 = 0.3606$
$Ca = 0.520$
$F_a = CaQ_sQ_aF_y$
$F_a = (0.520)(1.0)(0.7348)(50ksi) = 19.10\ ksi \sim 19.40\ ksi$

$F_a = 19.10$ ksi

D.1.4 Unstiffened Slender and Stiffened Slender Elements

When the cross-section contains both unstiffened and stiffened slender elements, the procedure is also iterative. The AISI provisions would apply the maximum design stress in the unstiffened elements to the stiffened elements. In this fashion, the AISI procedure would result in the web always possessing its minimum available cross-section area. However, this is not the case by the AISC procedure. By the AISC method the web cross-section area is greater, iterating to locate the effective width of the web. The effect of both Q_s and Q_a factors on the cross-section is shown in Figure D.1c. The allowable stress by this procedure will be higher than that of the AISI procedure, since the converged effective width of the section will be higher than that found using the maximum level of stress in the unstiffened elements.

It is reiterated here that the column allowable stress is not dependent on the service load compressive stress, but the maximum level of stress that can be applied to cause local buckling. The allowable compressive stress is determined as though the member were a stand-alone column. This is true whether or not the member is a column or a beam-column. The effective-length factor is still a necessary entity, governing the overall column behavior. The column strength should be developed independent of actual service load stress conditions. The following example illustrates the procedure for column allowable stress when both the flanges and web are slender.

Example: F_a Calculation, 1989 AISC ASD Specification

Given Data;

$KL/r = 45.0$ $A = 10.5 \text{ in}^2$
$F_y = 50$ ksi $Cc = 107.0$
$b_f = 7$ in $t_f = 0.25$ in
$h_c = 24$ in $t_w = 0.25$ in

Table B5.1, pg. 5-36

$h_c/t_w = 96.0 > (253/\sqrt{50} = 35.78)$; Slender, $Q_a < 1.0$
$h_c/t_w = 96.0 > 70.0$; $k_c = 0.5663$
$b_f/2t_f = 14.0 > (95/\sqrt{(50/k_c)} = 10.11)$; Slender, $Q_s < 1.0$
$b_f/2t_f = 14.0 < (195/\sqrt{(50/k_c)} = 20.75$; Eqn. A-B5-3

<< Iteration 1 >>

$Q_s = 1.293 - 0.00309(14.0)(9.40) = 0.8864$
$Q_a = 0.70$ (assumed)
$Q = (0.8864)(0.70) = 0.6205$
Assume $f = 0.6F_yQ = 0.6(50\text{ ksi})(0.6205) = 18.62$ ksi
$b_{eff} = (253(.25)/\sqrt{18.62})\ [1\text{-}44.3/(96\sqrt{18.62})] = 13.09''$
$Q_a = (10.5\text{-}(24\text{-}13.09)(.25))/10.5 = 0.7402$
$Q = Q_sQ_a = (0.8864)(0.7402) = 0.6561$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.6561} = 132.1$
$KL/rC'c = 45.0/132.1 = 0.3407$
$Ca = 0.527$
$F_a = CaQ_sQ_aF_y$
$F_a = (0.527)(0.8864)(0.6561)(50\text{ksi}) = 15.32\text{ ksi} < 18.62\text{ ksi}$

< < Iteration 2 > >

Assume $f = 15.32$ ksi
$b_{eff} = (253(.25)/\sqrt{15.32})\ [1\text{-}44.3/(96\sqrt{15.32})] = 14.25''$
$Q_a = (10.5\text{-}(24\text{-}14.25)(.25))/10.5 = 0.7679$
$Q = Q_sQ_a = (0.8864)(0.7679) = 0.6807$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.6807} = 129.7$
$KL/rC'c = 45.0/129.7 = 0.3470$
$Ca = 0.525$
$F_a = CaQF_y$
$F_a = (0.525)(0.6807)(50\text{ksi}) = 17.87\text{ ksi} > 15.32\text{ ksi}$

< < Iteration 3 > >

Assume $f = 17.87$ ksi
$b_{eff} = (253(.25)/\sqrt{17.87})\ [1\text{-}44.3/(96\sqrt{17.87})] = 13.33''$
$Q_a = (10.5\text{-}(24\text{-}13.33)(.25))/10.5 = 0.7460$
$Q = Q_sQ_a = (0.8864)(0.7460) = 0.6613$
$C'c = Cc/\sqrt{Q} = 107.0/\sqrt{0.6613} = 131.6$
$KL/rC'c = 45.0/131.6 = 0.3419$
$Ca = 0.526$
$F_a = CaQF_y$
$F_a = (0.526)(0.6613)(50\text{ksi}) = 17.40\text{ ksi} \sim 17.87\text{ ksi}$

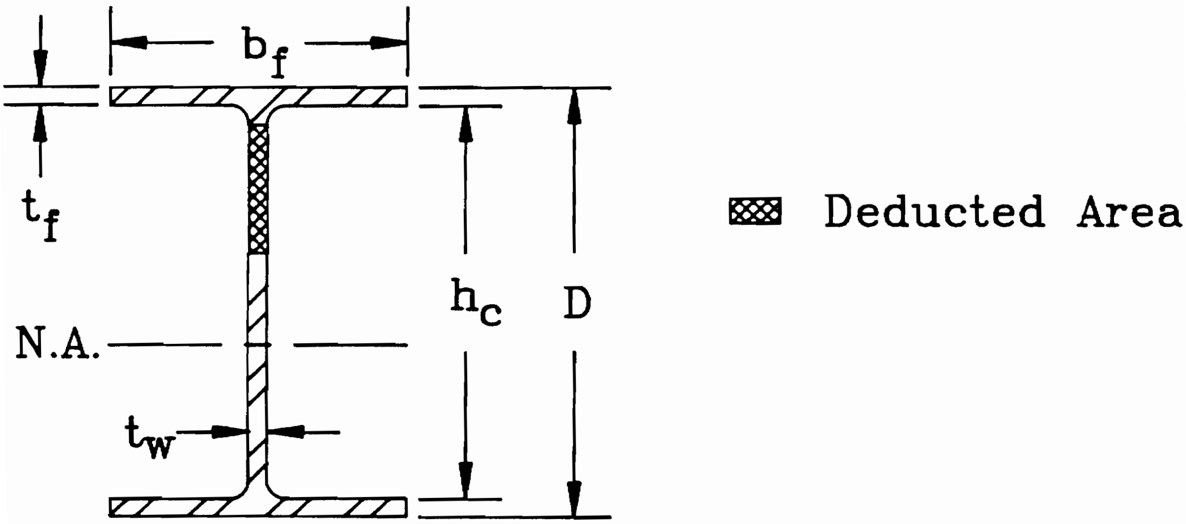
$$\mathbf{F_a = 17.40\ ksi}$$

D.2 Flexural Capacity

By the 1978 or 1989 AISC ASD Specifications, the allowable flexural stress of an I-shaped cross-section (when both the flanges and web are slender) is developed as the minimum of $0.6F_yQ_s$, or the *plate girder* allowable stress. The effect of Q_s on the cross-section was shown in Figure D.1a, and is not an iterative process to determine. The $0.6F_yQ_s$ (ASD) is applicable to prevent local buckling of the compression flange, and is calculated as described above. When the web h/t ratio exceeds $760/\sqrt{F_b}$, plate girder provisions apply (The $760/\sqrt{F_b}$ limit may be changed to $970/\sqrt{F_y}$ [Iwankiw, 1990]). The plate girder provisions account for bend buckling of the web, and the accompanying increased stress in the compression flange, and are found Section 1.10.6 of the 1978 ASD Specification, or Chapter G of the 1989 ASD Specification.

By the 1986 LRFD Specification, the flexural strength is also developed according to the web slenderness ratio. If the flanges are slender, and the web h/t ratio is less than $970/\sqrt{F_y}$, the provisions of Appendix F apply. Regardless of whether or not the flanges are slender, if the web h/t ratio is greater than $970/\sqrt{F_y}$, the plate girder provisions of Appendix G apply. Appendix G directly accounts for both lateral-torsional buckling and flange local buckling.

Application of the plate girder provisions of both ASD and LRFD has the effect of replacing the original cross-section with an effective cross-section. This effective cross-section results from bend buckling of the web. As the web buckles in regions of high compressive stress and moves out-of-plane, the compression flange receives the stress transfer. Rather than performing lengthy calculations to obtain the effective cross-section properties for this condition, the AISC Specifications use an equation to reduce the full section modulus to an effective section modulus (the formula shows a reduction of the allowable bending stress, but this is not what is really happening). This effect is shown in Figure D.2. In ASD, the basic F_b used in the plate girder equation is selected from the lateral-torsional buckling allowable stresses, or $0.6F_y$, whichever is smaller. In LRFD, the critical buckling stress is developed according to the lateral-torsional buckling strength, and the flange local buckling strength.



ASD Plate Girder Section Factor

$$R_{pg} = 1 - 0.0005\left(\frac{A_w}{A_f}\right)\left[\frac{760}{\sqrt{F_b}} - \frac{h_c}{t_w}\right]$$

LRFD Plate Girder Section Factor

$$R_{pg} = 1 - 0.0005\left(\frac{A_w}{A_f}\right)\left[\frac{970}{\sqrt{F_{cr}}} - \frac{h_c}{t_w}\right]$$

Plate Girder Effective Section
Figure D.2

The flange local buckling (FLB) limits in the 1989 ASD Specification involve the web h/t ratio, but the 1986 LRFD Specification FLB limits do not. This is the reason for the economic envelope shown earlier in Figure 4.2. By the 1989 ASD Specification, the allowable flexural stress must be selected as the minimum of the plate girder and slender flange provisions, if applicable. Utilizing LRFD in the range of $70 < h/t < 970/\sqrt{F_y}$, this condition can result in beams having flexural strengths approximately twelve percent higher than the same cross-section by the ASD Specification, including the load-effects side of the LRFD format equation.

The following example illustrates the calculation of the flexural allowable stress on an I-shaped cross-section whose flanges are compact but the web is slender.

Example: F_b Calculation, 1989 AISC ASD Specification

Given Data;
Compression Flange Fully Braced

F_y = 50 ksi S_{xc} = 199.2 in^3
b_f = 6 in t_f = 0.375 in
h_c = 48 in t_w = 0.25 in

Table B5.1, pg. 5-36

Assume F_b = 30 ksi
h_c/t_w = 192.0 > $(760/\sqrt{30})$ = 138.8) ; C/G or A/B
h_c/t_w = 192.0 > 70 ; k_c = 0.3607
$b_f/2t_f$ = 8.0 < ($95/\sqrt{(50/k_c)}$ = 8.07) ; Compact, C/G

Allowable Stress

Basic F_b = 30 ksi, as assumed above
R_{pg} = 1.0-0.0005((48*0.25)/(6*0.375))(192.0-138.8) = 0.8581
S_{xc}' = $R_{pg}S_{xc}$
S_{xc}' = (0.8581)(199.2in^3) = 170.93 in^3

$$F_b = 30 \text{ ksi}$$
$$S_{xc}' = 170.93 \text{ in}^3$$

D.3 Beam-Column Interaction

As described earlier, the column allowable stress (ASD) or the column capacity (LRFD) determined independently of service loading is utilized in the interaction formulas. An omission in the 1978 ASD Specification, and now appearing in the 1989 Specification, is the use of the combined form-factor Q ($Q=Q_sQ_a$) in the denominator of the axial term of the yielding equation. This results in higher interaction values than with the 1978 ASD Specification. The 1989 ASD equation is as follows:

$$\frac{f_a}{0.6QF_y} + \frac{f_b}{F_b} \leq 1.0 \qquad \text{(D.10)}$$

As an example, if the form factor is 0.85, the axial term would become 1/0.85 = 1.18, or 18% higher than in the 1978 ASD. The increase in the interaction ratio is lower, depending on the magnitude of the flexural term.

The allowable stresses (ASD) or the strengths (LRFD) calculated as though the member was a column or a beam, respectively, are used in the interaction equations. In other words, the denominators of the interaction equations are independent of the applied loads, but the numerators vary according to actual load combination effects.

The following example illustrates the difference between the 1978 and 1989 ASD Specification interaction equations.

Example: Interaction Ratios, AISC ASD Specifications

Given Data;

Unbraced Frame, First-Order Analysis & Design
F_y = 50.0 ksi
F'_e = 60.0 ksi
f_a = 4.0 ksi F_a = 16.0 ksi   Q = 0.60
f_b = 24.0 ksi F_b = 30.0 ksi

Stability: 1978 or 1989 ASD Specification

$f_a/F_a + C_m/(1-f_a/F'_e)\ (f_b/F_b)$
4.0/16.0 + 0.85/(1-4.0/60.0) (24.0/30.0) = 0.9786 < 1.0   **OK**

Yielding:

a) 1978 ASD Specification
fa/0.6Fy + fb/Fb
4.0/30.0 + 24.0/30.0 = 0.9333 < 1.0   **OK**

b) 1989 ASD Specification
fa/0.6QFy + fb/Fb
4.0/(0.60*30.0) + 24.0/30.0 = 1.0222 > 1.0   **NG**

Difference in Yielding Interactions:

(1.0222-0.9333)/0.9333 * 100(%) = 9.52 %

APPENDIX E

PROGRAM LISTINGS

SOAPREP.c

```
/****************************************************************************/
/*                                                                          */
/*      SOAPREP.c                                                           */
/*                                                                          */
/*      Preprocessor for Second Order Elastic Analysis of Rigid Frame       */
/*      Structures.  Defines the Structure Geometry and Basic Loads.        */
/*                                                                          */
/*      Research Team:                                                      */
/*      Eric J. Wishart..............................Research Assistant     */
/*      Thomas M. Murray, PhD.....................Principal Investigator    */
/*                                Of The                                    */
/*           Charles Edward Via Department of Civil Engineering             */
/*           Virginia Polytechnic Institute & State University              */
/*                         Blacksburg, Va. 24061                            */
/*                                                                          */
/*      Inputfiles:     RAW.dat     -> Joint Coordinates, Member Properties, */
/*                                     Member Incidences, Tributary Widths  */
/*                                                                          */
/*      Outputfiles:    COORDS.def -> Input Check File                      */
/*                      NUCOR.def  -> Computer use ONLY!                    */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/*      1.  Written for an IBM PC or Compatible Personal Computer           */
/*          running PC-DOS 3.3 or equivalent.                               */
/*      2.  Developed using the Microsoft Version 6.0 C Compiler            */
/*      3.  Dynamic Memory Allocation accesses Expanded/Extended Memory,    */
/*          if present, to solve large DOF problems.                        */
/*      4.  All REAL computations performed in double precision format.     */
/*      5.  Load Factors and Magnitude/Direction Coefficients are stored    */
/*          in separate, user editable files for future revision.          */
/*      6.  Loads are input in [psf].  Load Vectors are stored as [kip]     */
/*      7.  All routines expect quantities are in [kips] and/or [inches]    */
/*                                                                          */
/********************************** INCLUDEs ********************************/
/* ---------------------------- Microsoft Files ---------------------------- */
#include <stdio.h>                              /* i/o functions            */
#include <stdlib.h>                             /* library functions        */
#include <malloc.h>                             /* memory functions         */
#include <math.h>                               /* math functions           */
#include <memory.h>                             /* initialization functions */
#include <conio.h>                              /* console i/o              */
#include <graph.h>                              /* screen i/o               */
/****************************************************************************/

/********************************** DEFINEs *********************************/
/* -------------------------------- Constants ------------------------------ */
#define PI              3.141592653589793                                /**/
#define MAXJOINTS       500                                              /**/
/* --------------------------------- Macros -------------------------------- */
#define forget(x)       { if(x) free((void *)x) ; x = NULL ; }           /**/
/************************* File Pointer Declarations ************************/
FILE *fp_lf ;                                   /* Load Factors File        */
FILE *fp_ds ;                                   /* Dead/Snow Coef. File     */
FILE *fp_wind ;                                 /* Wind Coef. File          */
FILE *fp ;                                      /* Geometry and Load File   */
FILE *fp_coord ;                                /* User Check/Coord File    */
FILE *fp_raw ;                                  /* Raw Data File            */
/****************************************************************************/
/* Program Main                                                             */
/****************************************************************************/
main()
{
/* -- Geometry Definition Variables -- */
double bf, tf, hc, tw, Da, Db, length, theta, dx, dy, Fy,
       *xcoord, *ycoord, Roof_Slope ;
int nmem, njoints, ndofs, loc_vec[6], nsupports, dof, a_end, b_end,
    process, i, j, k, nj, nsup[50], suptype[50], total_bc, dummy ;
```

```
static char *type_buf[] = { "Fixed" , "Pinned" , "Roller-X" , "Roller-Y" } ;
char structure_description[80] ;

/* -- Load Vector Definition Variables -- */
double *load,  dead_psf, snow_psf, wind_psf, *trib_area, DL, SL, SR, WL, WR,
       ds_coef[4], bay_space, trib, w1, w2, w3, w4, direction, R_DL,
       cos_rs, sin_rs ;
int *lv_left, *lv_right, *lh_left, *lh_right, nv_l, nv_r, nh_l, nh_r ;

/* -- Initialize Geometry Variables -- */
  nsupports = total_bc = 0 ;
  memset( nsup, 0, 50*sizeof(int) ) ;
  memset( suptype, 0, 50*sizeof(int) ) ;

  /* -- Initialize Load Variables -- */
  bay_space = 0.0 ;
  nv_l = nv_r = nh_l = nh_r = 0 ;

  /* ------------------------------------------------------------------ */
  /* -- File Handles Initializations                                 -- */
  /* ------------------------------------------------------------------ */

  /* Open Load Factors File */
  _clearscreen(_GCLEARSCREEN) ;
  if ( ( fp_raw = fopen("RAW.DAT","r") ) == NULL )
  {
    /* -- Terminate On Error -- */
    _clearscreen(_GCLEARSCREEN) ;
    printf("\n DOS Error: Cannot Read File RAW.DAT") ;
    exit(0) ;
  }



  /* Open Load Factors File */
  if ( ( fp_lf = fopen("LOADFACT.DAT","r") ) == NULL )
  {
    /* -- Terminate On Error -- */
    _clearscreen(_GCLEARSCREEN) ;
    printf("\n DOS Error: Cannot Read File LOADFACT.DAT") ;
    exit(0) ;
  }

  /* Open Dead & Snow Magnitude/Direction File */
  if ( ( fp_ds = fopen("DSCOEF.DAT","r") ) == NULL )
  {
    /* -- Terminate On Error -- */
    _clearscreen(_GCLEARSCREEN) ;
    printf("\n DOS Error: Cannot Read File DSCOEF.DAT") ;
    exit(0) ;
  }

  /* Open Wind Magnitude/Direction File */
  if ( ( fp_wind = fopen("WINDCOEF.DAT","r") ) == NULL )
  {
    /* -- Terminate On Error -- */
    _clearscreen(_GCLEARSCREEN) ;
    printf("\n DOS Error: Cannot Read File WINDCOEF.DAT") ;
    exit(0) ;
  }

  /* Open Geometry File */
  if ( ( fp = fopen("NUCOR.DEF","wb") ) == NULL )
  {
    /* -- Terminate On Error -- */
    _clearscreen(_GCLEARSCREEN) ;
    printf("\n DOS Error: Cannot Create File NUCOR.DEF") ;
    exit(0) ;
```

```
}

/* Open Coordinate File */
if ( ( fp_coord = fopen("COORDS.DEF","w") ) == NULL )
{
  /* -- Terminate On Error -- */
  _clearscreen(_GCLEARSCREEN) ;
  printf("\n DOS Error: Cannot Create File COORDS.DEF") ;
  exit(0) ;
}

/* ---------------------------------------------------------------------- */
/* -- Control Variable Initializations                                 -- */
/* ---------------------------------------------------------------------- */

_clearscreen(_GCLEARSCREEN) ;
printf("\n * * * Structure Definition Control Parameters * * * \n") ;
printf("\n Title for Job: ") ;
gets(structure_description) ;
fprintf(fp_coord, "\n TITLE: ") ;
fputs( structure_description, fp_coord ) ;
printf("\n Total Number of Joints in Structure  : ") ;
scanf("%d",&njoints) ;
printf(" Total Number of Members in Structure : ") ;
scanf("%d",&nmem) ;

/* Test for Zero Joints or Members */
if ( njoints == 0 || nmem == 0 )
{
  _clearscreen(_GCLEARSCREEN) ;
  exit(0) ;
}

/* Structural dofs assignment */
ndofs = 3*njoints ;
if ( njoints > MAXJOINTS )
{
  printf("\n Degrees of Freedom > Program Capacity, ") ;
  printf("\n Maximum Number of Joints = 500 ") ;
  _clearscreen(_GCLEARSCREEN) ;
  exit(0) ;
}

/* Determine Fabrication Procedure for Specification Checking */
printf("\n Fabrication Process: 0 = Rolled, 1 = Welded  ") ;
scanf("%d",&process) ;

/* Determine Steel Yield Stress */
printf("\n Steel Yield Stress, Fy [ksi] ") ;
scanf("%lf",&Fy) ;

/* Tributary Area Information */
printf("\n Bay Spacing [FT]: ") ;
scanf("%lf", &bay_space) ;
printf("\n Roof Slope x /12 ? x = ") ;
scanf("%lf", &Roof_Slope ) ;

/* Save Info to User Check/Coordinate File for Backup */
fprintf(fp_coord, "\n\n Bay Spacing  Roof Slope") ;
fprintf(fp_coord, "\n    [ft]        [Unity] ") ;
fprintf(fp_coord, "\n ----------  ----------") ;
fprintf(fp_coord, "\n %9.2lf     %3.2lf / 12  ", bay_space, Roof_Slope ) ;

/* Write info to file */
fprintf( fp, "%d %d %d %lf", njoints, nmem, process, Fy ) ;
```

```
/* ------------------------------------------------------------------ */
/* -- Joint Coordinate Generation                                  -- */
/* ------------------------------------------------------------------ */

/* -- Dynamic Allocation -- */
xcoord    = (double *) calloc( njoints, sizeof(double) ) ;
ycoord    = (double *) calloc( njoints, sizeof(double) ) ;
trib_area = (double *) calloc( ndofs,   sizeof(double) ) ;
load      = (double *) calloc( ndofs,   sizeof(double) ) ;
lv_left   = (int    *) calloc( njoints, sizeof(int)    ) ;
lv_right  = (int    *) calloc( njoints, sizeof(int)    ) ;
lh_left   = (int    *) calloc( njoints, sizeof(int)    ) ;
lh_right  = (int    *) calloc( njoints, sizeof(int)    ) ;

_clearscreen(_GCLEARSCREEN) ;
printf("\n * * * Structure Joint Coordinates & Element Geometry * * * \n") ;
printf("\n Reading Data File...") ;

if ( xcoord == NULL || ycoord == NULL )
{
  printf("\n Insufficient System Memory, Redefine Structure") ;
  printf("\n with Fewer Joint Coordinates\n") ;
  exit(0) ;
}
if ( load == NULL || lv_left == NULL || lv_right == NULL || lh_left == NULL ||
lh_right == NULL ||      trib_area == NULL )
{
  printf("\n Insufficient System Memory, Redefine Structure") ;
  printf("\n with Fewer Degrees of Freedom\n") ;
  exit(0) ;
}

/* -- Read Raw Data File --*/
for ( i = 0 ; i < njoints ; i++ )
{
  fscanf( fp_raw, "%d %lf %lf", &dummy, &xcoord[i], &ycoord[i] ) ;
  xcoord[i] *= 12.0 ;
  ycoord[i] *= 12.0 ;
}

/* -- Store Coordinates for Later Printout &/Or Visual Checking -- */
fprintf(fp_coord, "\n\n JOINT COORDINATES") ;
fprintf(fp_coord, "\n -----------------\n") ;
fprintf(fp_coord, "\n  JT#     X [in]      Y [in]") ;
fprintf(fp_coord, "\n -----  ---------  ---------" ) ;
for ( i = 0 ; i < njoints ; i++ )
  fprintf(fp_coord, "\n %5d %10.4lf %10.4lf", i+1, xcoord[i], ycoord[i] ) ;
fprintf(fp_coord, "\n\n") ;

/* ------------------------------------------------------------------ */
/* -- Element Generation                                           -- */
/* ------------------------------------------------------------------ */

fprintf(fp_coord, "\n ELEMENT INFORMATION") ;
fprintf(fp_coord, "\n -------------------\n") ;
fprintf(fp_coord, "\n Member    bf      tf       hc       tw     Length      Theta
Incidences");
fprintf(fp_coord, "\n  No.    [in]    [in]     [in]     [in]     [in]       [rad]
A-End  B-End");
fprintf(fp_coord, "\n ------ ------ ------- ------- ------- --------- --------
-----  -----");

for ( i = 0 ; i < nmem ; i++ )
{
  /* -- Member Properties -- */
  fscanf(fp_raw, "%d %lf %lf %lf %lf %lf %d %d",
                  &dummy, &bf, &tf, &Da, &Db, &tw, &a_end, &b_end ) ;
```

```
  /* -- Avg. Member Web Height -- */
  hc = 0.5*(Da+Db) - 2.0*tf ;

  /* -- Member Length -- */
  dx = xcoord[b_end-1] - xcoord[a_end-1] ;
  dy = ycoord[b_end-1] - ycoord[a_end-1] ;
  length = sqrt( dx*dx + dy*dy ) ;

  /* -- Calculate Θ Angle -- */
  if ( fabs(dx) < 1.0e-3 )
    theta = ( dy > 0.0 ) ? PI/2.0 : -PI/2.0 ;
  else
  {
    theta = atan(dy/dx) ;
    if ( dx < 0.0 ) theta += PI ;
  }

  /* Element MCODE Assignment */
  loc_vec[0] = 3 * a_end - 3 ;
  loc_vec[1] = 3 * a_end - 2 ;
  loc_vec[2] = 3 * a_end - 1 ;
  loc_vec[3] = 3 * b_end - 3 ;
  loc_vec[4] = 3 * b_end - 2 ;
  loc_vec[5] = 3 * b_end - 1 ;

  /* Write info to file */
  fprintf( fp, " %.4lf %.4lf %.4lf %.4lf %.4lf %.4lf %.4lf %.6lf",
                        bf, tf, hc, tw, Da, Db, length, theta ) ;
  for ( j = 0 ; j < 6 ; ++j ) fprintf( fp, " %4d",loc_vec[j]) ;

  /* -- Store Element Information for Later Printout &/Or Visual Checking -- */
  fprintf(fp_coord, "\n %5d  %7.4lf  %5.4lf  %7.4lf  %7.4lf  %9.4lf  % 8.4lf  %4d
%4d",
                    (i+1), bf, tf, hc, tw, length, theta, a_end, b_end ) ;

}

/* ------------------------------------------------------------------ */
/* -- Boundary Condition Generation                                -- */
/* ------------------------------------------------------------------ */


REDO:
_clearscreen(_GCLEARSCREEN) ;
printf("\n * * * Structure Boundary Conditions * * * \n") ;
printf("Enter Structure Joint Numbers where Supports are Provided,\n") ;
printf("Followed by the Support Type.  Terminate entry with a '0 0'\n") ;
printf("Fixed = 0     Pinned = 1     Roller-X = 2     Roller-Y = 3\n\n") ;

do
{
  scanf("%d %d", &nj, &j ) ;                       // Joint #, Type
  nsup[nsupports] = nj - 1 ;                        // (Int) Joint #
  suptype[nsupports] = j ;                          // Hold Type
  nsupports += 1 ;
}
while ( nj != 0 ) ;
nsupports -= 1 ;                                    // Correct nsupports

/* Parse Support Conditions... */
for ( k = 0 ; k < nsupports ; k++ )
{
  if ( suptype[k] == 0 )         total_bc += 3 ;     // Fixed
  else if ( suptype[k] == 1 )    total_bc += 2 ;     // Pinned
  else                           total_bc += 1 ;     // Roller
}

/* Error Check Boundary Conditions */
```

```
if ( total_bc > 50 )
{
  _clearscreen(_GCLEARSCREEN) ;
  printf("\n Input Error: Fifty (50) Restrained DOFS Maximum Allowable.") ;
  printf("\n Please Re-enter the Boundary Conditions.") ;
  printf("\n\n Press Any Key to Continue...") ;
  nsupports = total_bc = 0 ;
  while ( !kbhit() ) ;                              // Kill Time
  getch() ;                                         // Key pressed
  goto REDO ;                                       // Loop Back
}

/* -- Boundary Conditions OK, Save to Geometry File -- */
fprintf( fp, " %d", total_bc ) ;

/* -- Write BC Information to File -- */
for ( k = 0 ; k < nsupports ; k++ )
{
  if ( suptype[k] == 0 )                            // Fixed
  {
    for ( j = 0 ; j <= 2 ; j++ )
      fprintf( fp, " %d", 3*nsup[k]+j ) ;
  }
  else if ( suptype[k] == 1 )                       // Pinned
  {
    for ( j = 0 ; j <= 1 ; j++ )
      fprintf( fp, " %d", 3*nsup[k]+j ) ;
  }
  else if ( suptype[k] == 2 )                       // Roller-x
    fprintf( fp, " %d", 3*nsup[k]+1 ) ;
  else if ( suptype[k] == 3 )                       // Roller-y
    fprintf( fp, " %d", 3*nsup[k] ) ;
}

/* -- Store Boundary Condition Information for Later Printout &/Or Visual Checking
-- */
fprintf(fp_coord, "\n\n\n BOUNDARY CONDITIONS") ;
fprintf(fp_coord,     "\n -------------------\n") ;
fprintf(fp_coord, "\n Joint  Support") ;
fprintf(fp_coord, "\n  No.    Type") ;
fprintf(fp_coord, "\n -----  -------") ;
for ( k = 0 ; k < nsupports ; k++ )
{
  i = suptype[k] ;
  fprintf(fp_coord, "\n %4d    %s", nsup[k]+1, type_buf[i] ) ;
}

/* ---------------------------------------------------------------------- */
/* -- Load Combination Generation                                       -- */
/* ---------------------------------------------------------------------- */

_clearscreen(_GCLEARSCREEN) ;
printf("\n * * * Basic Load Information * * *\n ") ;
printf("\n Enter Basic Load Magnitudes [psf] ") ;

printf("\n Uniform Dead + Collateral Load: ") ;
scanf("%lf", &dead_psf) ;
printf(" Uniform Ground Snow Load: ") ;
scanf("%lf", &snow_psf) ;
printf(" Wind Stagnation Pressure: ") ;
scanf("%lf", &wind_psf) ;
printf("\n Beginning Load Combination Generation:\n\n ") ;

/* Save Info to User Check/Coordinate File for Backup */
fprintf(fp_coord, "\n\n\n EXTERNAL BASIC LOADS") ;
fprintf(fp_coord,     "\n --------------------\n") ;
fprintf(fp_coord, "\n   D+C     Snow     Wind  ") ;
fprintf(fp_coord, "\n  [psf]    [psf]    [psf] ") ;
```

```
fprintf(fp_coord, "\n -------  -------  -------") ;
fprintf(fp_coord, "\n %7.2lf  %7.2lf  %7.2lf", dead_psf, snow_psf, wind_psf ) ;
fprintf(fp_coord, "\n\n") ;


/* Close User Check/Coordinate File */
fclose ( fp_coord ) ;

/* Vertical Load Locations --> Left Rafter */
do
{
  fscanf(fp_raw, "%d %lf", &nj, &trib ) ;                    // Joint #
  lv_left[nv_l] = 3*nj - 2 ;                                  // (Int) DOF
  trib_area[lv_left[nv_l]] = trib * bay_space / 1000.0 ;     // sf/1000p/k
  nv_l += 1 ;
}
while ( nj != 0 ) ;
nv_l -= 1 ;                                                   // Correct nv_l

/* Vertical Load Locations --> Right Rafter */
do
{
  fscanf(fp_raw, "%d %lf", &nj, &trib ) ;                     // Joint #
  lv_right[nv_r] = 3*nj - 2 ;                                  // (Int) DOF
  trib_area[lv_right[nv_r]] = trib * bay_space / 1000.0 ;     // sf/1000p/k
  nv_r += 1 ;
}
while ( nj != 0 ) ;
nv_r -= 1 ;                                                    // Correct nv_r

/* Horizontal Load Locations --> Left Rafter */
do
{
  fscanf(fp_raw, "%d %lf", &nj, &trib ) ;                    // Joint #
  lh_left[nh_l] = 3*nj - 3 ;                                  // (Int) DOF
  trib_area[lh_left[nh_l]] = trib * bay_space / 1000.0 ;     // sf/1000p/k
  nh_l += 1 ;
}
while ( nj != 0 ) ;
nh_l -= 1 ;                                                   // Correct nh_l

/* Horizontal Load Locations --> Right Rafter */
do
{
  fscanf(fp_raw, "%d %lf", &nj, &trib ) ;                    // Joint #
  lh_right[nh_r] = 3*nj - 3 ;                                 // (Int) DOF
  trib_area[lh_right[nh_r]] = trib * bay_space / 1000.0 ;    // sf/1000p/k
  nh_r += 1 ;
}
while ( nj != 0 ) ;
nh_r -= 1 ;                                                   // Correct nh_r

/* Read Dead & Snow Magnitude/Direction Factors */
printf("\r                                                   ") ;
printf("\r Reading Direction Factors") ;
for ( i = 0 ; i < 4 ; i++ ) fscanf(fp_ds, "%lf", &ds_coef[i]) ;

/* Factors for Wind Load Application Points */
Roof_Slope = atan(fabs(Roof_Slope/12.0)) ;
cos_rs = cos(Roof_Slope) ;
sin_rs = sin(Roof_Slope) ;
R_DL = 1.0/cos_rs ;

/* -- Read First Load Case Factors -- */
i = 1 ;
printf("\r                                        ") ;
printf("\r Assembling Load Combination No. %d", i) ;
fscanf(fp_lf, "%lf %lf %lf %lf %lf", &DL, &SL, &SR, &WL, &WR) ;
```

```
fscanf(fp_wind, "%lf %lf %lf %lf", &w1, &w2, &w3, &w4) ;

/* Assemble the Load Vectors by Load Combination */
/* -- k = Internal DOF# Assigned the Concentrated Load -- */
do
{
  /* -- Initialize Load Vector -- */
  memset( load, 0, ndofs*sizeof(double) ) ;

  /* -- << Left Column >> -- */
  for ( j = 0 ; j < nh_l ; j++ )
  {
     k = lh_left[j] ;
     if ( k >= 0 ) load[k] += WL*w1*trib_area[k]*wind_psf ;
  }

  /* -- << Left Rafter >> -- */
  for ( j = 0 ; j < nv_l ; j++ )
  {
     /* Global Direction Vertical Loads */
     k = lv_left[j] ;
     if ( k > 0 ) load[k] += R_DL*DL*ds_coef[1]*trib_area[k]*dead_psf +
                                SL*ds_coef[1]*trib_area[k]*snow_psf +
                                 WL*cos_rs*w2*trib_area[k]*wind_psf ;

     /* Global Direction Horizontal Loads */
     direction = -1.0 ;                         // Global X <- Direction
     if ( k >= 0 && WL >= 0.0 )
       load[k-1] += WL*w2*sin_rs*sin_rs*R_DL*direction*trib_area[k]*wind_psf ;
  }

  /* -- << Right Rafter >> -- */
  for ( j = 0 ; j < nv_r ; j++ )
  {
     /* Global Direction Vertical Loads */
     k = lv_right[j] ;
     if ( k > 0 ) load[k] += R_DL*DL*ds_coef[2]*trib_area[k]*dead_psf +
                                SR*ds_coef[2]*trib_area[k]*snow_psf +
                                 WR*cos_rs*w3*trib_area[k]*wind_psf ;

     /* Global Direction Horizontal Loads */
     direction = 1.0 ;                          // Global X -> Direction
     if ( k > 0 && WR >= 0.0 )
       load[k-1] += WR*w3*sin_rs*sin_rs*R_DL*direction*trib_area[k]*wind_psf ;
  }

  /* -- << Right Column >> -- */
  for ( j = 0 ; j < nh_r ; j++ )
  {
     k = lh_right[j] ;
     if ( k > 0 && WR > 0.0 ) load[k] += WR*w4*trib_area[k]*wind_psf   ;
  }

  /* -- Save Load Vector to File -- */
  for ( j = 0 ; j < ndofs ; j++ ) fprintf( fp, " % 8.4lf", load[j] ) ;

  /* -- Read Next Load Case Factors -- */
  i += 1 ;
  printf("\r                                        ") ;
  if ( i < 33 ) printf("\r Assembling Load Combination No. %d", i) ;
  fscanf(fp_lf, "%lf %lf %lf %lf %lf", &DL, &SL, &SR, &WL, &WR) ;
  fscanf(fp_wind, "%lf %lf %lf %lf", &w1, &w2, &w3, &w4) ;
}
while ( DL != 0.0 ) ;

/* ---------------------------------------------------------------------- */
/* -- Clean-Up                                                          -- */
/* ---------------------------------------------------------------------- */
```

```
  /*-- Close Files --*/
  fcloseall() ;


  /* -- Return System Memory -- */
  forget ( xcoord ) ;
  forget ( ycoord ) ;
  forget ( load ) ;
  forget ( lv_left ) ;
  forget ( lv_right ) ;
  forget ( lh_left ) ;
  forget ( lh_right ) ;

  /* -- Progress Report -- */
  printf("\r Done! \n") ;

} /* End of Main Program */
/*****************************************************************************/
/* End of SOAPREP Program                                                   */
/*****************************************************************************/
```

PDELTA.c

```
/*******************************************************************************/
/*
* PDELTA.h       Header File for PDELTA Program
*/
/********************************** INCLUDEs **********************************/
#include <stdio.h>                          /* i/o functions                 */
#include <conio.h>                          /* keyboard functions            */
#include <memory.h>                         /* initialization functions      */
#include <math.h>                           /* math functions                */
#include <stdlib.h>                         /* library functions             */
#include <malloc.h>                         /* memory functions              */
#include <search.h>                         /* sorting routine               */
#include <ctype.h>                          /* character conversionS         */
/********************************** DEFINEs ***********************************/
#define NOTPOSITIVEDEF  -1                  /* Error in Factorization of SS  */
#define ZERO_STIFFNESS  1.0e-3              /* Tolerance for RIGID BODY MODE */
#define TRUE            1                   /* Logical Test Value            */
#define FALSE           0                   /* Logical Test Value            */
#define NO_ERROR        1                   /* Error Check -> Proceed        */
#define ERROR           0                   /* Error Check -> Halt           */
#define L_C             32                  /* # Load Combinations           */
#define E               29000.0             /* Elastic Modulus of Steel      */
#define NUMLOADSTEPS    5                   /* Maximum Load Increments       */
#define MAXCYCLE        20                  /* Maximum Iterations/Increment  */
#define CPFB            0.1                 /* Maximum Jt. Force Imbalance   */
#define CPDB            0.001               /* Displacement Tolerance Ratio  */
#define DENSITY         0.283e-3            /* Steel Density, k/in^3         */
#define forget(x)                  { if(x) free((void *)x) ; x = NULL ; }
/********************************* STRUCTURES *********************************/
typedef struct ELEMENT
{
  double BF,                                /* Flange Width                  */
         TF,                                /* Flange Thickness              */
         HC,                                /* Avg. Web Height               */
         TW,                                /* Web Thickness                 */
         Da,                                /* Total Depth at A-End          */
         Db,                                /* Total Depth at B-End          */
         LENGTH,                            /* Length of Member (Element)    */
         COST,                              /* cos Ö  (Direction Cosine  x)  */
         SINT,                              /* sin Ö  (Direction Cosine  y)  */
         Ag,                                /* Gross Area of Element         */
         Ix,                                /* X-X Moment of Inertia         */
         SW,                                /* Element Selfweight            */
         P ;                                /* Axial Force (+C , -T )        */
  int MCODE[6] ;                            /* Element MCODE (DOF Matching)  */
} ELEMENT ;
```

```
/*******************************************************************************/
/*
* PDELTA.fun              Function Prototype File for PDELTA Program
*/
/*******************************************************************************/
/******************************* PROTOTYPES *******************************/
int  main(int  argc,char  * *argv);
int  member_properties(struct  ELEMENT *element);
int  support_conditions(int  *nsupports,int  *bc);
int  skyline(int  nmem,int  neqn,struct  ELEMENT *element,int  *Kht,int  *Maxa);
int   assemble_stiffness_matrix(int   neqn,int   nmem,int   nsupports,int   *bc,int
*Maxa,double  *ss,struct  ELEMENT *element,double  (*km)[6]);
void member_stiffness(struct  ELEMENT *element,double  (*km)[6]);
int   assemble_load_vector(int   neqn,int   cycle,double   *Q,double   *QQ,double
*Residual,struct  ELEMENT *element,int  nmem,int  load_case);
int  solve(double  *ss,double  *QQ,int  *Maxa,int  neqn,int  load_case);
int  factor(double  *ss,int  *Maxa,int  neqn);
void forsub(double  *ss,double  *QQ,int  *Maxa,int  neqn);
void bacsub(double  *ss,double  *QQ,int  *Maxa,int  neqn);
void print_displacements(int  load_case,int  njoints,double  *TotQQ);
void support_reactions(int  load_case,int  nsupports,double  *SumGJF,int  *bc);
void member_end_forces(int loadstep,int load_case,int nmem,int neqn,int cycle,int
nsupports,struct ELEMENT *element,double (*km)[6],double *SumGJF,double *TotQQ,int
numloadsteps);
void element_props(int fo_flag,int nmem,struct ELEMENT *element );
int   calc_residual(double   *Residual,double   *Q,double   *QQ,double   *SumGJF,int
loadstep,int  neqn,int  nsupports,int  *bc,int  cycle,double  *deltaQQ);
int  sort_descending(double  *value1,double  *value2);
void stg_toupper(char  *stg);
void pause(void);
/****************************** Utility Functions ******************************/
void stg_toupper( char *stg )
{
  char *temp_ptr ;

  temp_ptr = stg ;
  while( *temp_ptr ) *temp_ptr++ = toupper( *temp_ptr ) ;
  return ;
}

void pause(void)
{
  printf("\n\n Strike a key when ready...") ;
  while ( !kbhit() ) ;
  getch() ;
}
```

```
/*****************************************************************************/
/*                                                                           */
/*      PDELTA.c                                                             */
/*                                                                           */
/*      Program to Perform Second Order Elastic Analysis of Rigid Frame      */
/*      Frame Structures.  Program follows the Matrix Displacement Method    */
/*      of Structural Analysis.  Uses NEWTON-RHAPSON solution technique      */
/*      in order to solve the nonlinear problem.                             */
/*                                                                           */
/*      Research Team:                                                       */
/*      Eric J. Wishart...............................Research Assistant     */
/*      Thomas M. Murray, PhD.....................Principal Investigator     */
/*                               Of The                                      */
/*             Charles Edward Via Department of Civil Engineering            */
/*             Virginia Polytechnic Institute & State University             */
/*                         Blacksburg, Va. 24061                             */
/*                                                                           */
/*      Inputfile:      NUCOR   .def    ; Geometry & Loads                   */
/*      Outputfiles:    ASD     .dis    ; Second Order Displacements         */
/*                      LRFD    .dis    ; Second Order Displacements         */
/*                      ASD     .tmp    ; Second Order Member Forces         */
/*                      LRFD    .tmp    ; Second Order Member Forces         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/*      Program Notes:                                                       */
/*                                                                           */
/*      1.  Written for an IBM PC or Compatible Personal Computer            */
/*          running PC-DOS 3.3 or equivalent.                                */
/*      2.  Developed using the Microsoft Version 6.0 C Compiler             */
/*      3.  Dynamic Memory Allocation accesses Expanded/Extended Memory,     */
/*          if present, to solve large DOF problems.                         */
/*      4.  All REAL computations performed in double precision format.      */
/*      5.  Tangent Stiffness is used for Structural Response to Loading.    */
/*      6.  Secant Stiffness is used for Member Force-Deformation Response.  */
/*                                                                           */
/****************************** Local INCLUDEs *******************************/
#include "pdelta.h"                             /* constants, macros, types  */
#include "pdelta.fun"                           /* function prototypes       */
/************************ Global Variable Declarations ***********************/
int numloadsteps, fo_flag, ss_comps ;
/************************* File Pointer Declarations *************************/
FILE *fp_input    ;                              /* Geometry & Loads File    */
FILE *fp_asd_dis  ;                              /* ASD Displacements file   */
FILE *fp_asd_tmp  ;                              /* ASD Spec. Check file     */
FILE *fp_lrfd_dis ;                              /* LRFD Displacements file  */
FILE *fp_lrfd_tmp ;                              /* LRFD Spec. Check file    */
/*****************************************************************************/
/* Program Main                                                              */
/*****************************************************************************/

main( int argc, char *argv[] )
{

/* Local Variables */
double Fy, km[6][6], *ss, *Q, *QQ, *TotQQ, *deltaQQ, *SumGJF, *Residual ;
int i, j, loadstep, cycle, load_case, mem, result, eqn,
    process, stop, njoints, nmem, neqn, nsupports,
    *bc, *Kht, *Maxa, status = NO_ERROR, Active_Load_Cases ;
char *p_char, *filename ;
ELEMENT *element ;

  /* Print Header for Program */
  puts(" PDELTA - Ver. 1.00") ;
  puts(" Virginia Polytechnic Institute & State University") ;
  puts(" Charles E. Via Department of Civil Engineering") ;
  puts(" Blacksburg, Va. 24061") ;
  puts(" Copyright 1990, All Rights Reserved") ;
```

```
printf("\n") ;

/* Check Argument List */
if ( argc <= 2 )
{
  puts(" Usage: PDELTA order filename") ;
  status = ERROR ;
}

/* Begin Definition of Problem */
if ( status ) /* <<<<< begin first set of status checks >>>>>> */
{
  /* Parse Command Line Arguments */
  p_char = argv[1] ;                          // order
  stg_toupper( p_char ) ;                     // ORDER
  filename = argv[2] ;                        // filename
  stg_toupper( filename ) ;                   // FILENAME

  /* Input File Handle */
  status = ((fp_input = fopen(filename,"rb")) == NULL ) ? ERROR : NO_ERROR ;
  if ( !status ) printf("\n File %s Not Found!\n", filename ) ;

  /* -- Initialize Output Files -- */
  status = ((fp_asd_dis = fopen( "ASD.dis", "w" )) == NULL ) ? ERROR : NO_ERROR ;
  status = ((fp_asd_tmp = fopen( "ASD.tmp", "wb")) == NULL ) ? ERROR : NO_ERROR ;
  if ( !fo_flag )
  {
    status = ((fp_lrfd_dis = fopen( "LRFD.dis", "w" )) == NULL ) ? ERROR : NO_ERROR
;
    status = ((fp_lrfd_tmp = fopen( "LRFD.tmp", "wb")) == NULL ) ? ERROR : NO_ERROR
;
  }
  if ( !status ) puts(" Unable to Open Disk Solution Files...") ;

  if ( status )
  {
    /* Set Environment for Analysis Type */
    if (!strcmp(p_char,"SECOND"))
    {
      /* -- Assign StepSize for Basic Incremental Load -- */
      printf(" Number of Load Increments [1-5]: ") ;
      scanf("%d", &numloadsteps ) ;
      fo_flag = FALSE ;
      Active_Load_Cases = L_C ;               // Process ASD & LRFD
      if ( numloadsteps > NUMLOADSTEPS ) numloadsteps = NUMLOADSTEPS ;
      if ( numloadsteps <= 1 ) numloadsteps = 1 ;
    }
    else                                      // First Order selected or
    {                                         // Default to FIRST_ORDER
      numloadsteps = 1 ;
      fo_flag = TRUE ;
      Active_Load_Cases = L_C/2 ;             // Process ASD
    }
  }

} /* <<<<<< end first set of status checks >>>>>> */

if ( status ) /* <<<<<< begin second set of status checks >>>>>> */
{
  /* --  Read General Structure Data -- */
  fscanf( fp_input, "%d %d %d %lf", &njoints, &nmem, &process, &Fy ) ;

  /* -- Degrees of Freedom for Structure (GLOBAL) -- */
  neqn = njoints * 3 ;

  /* -- Dynamic Allocation for Structural Model -- */
  element  =  (ELEMENT *) calloc( nmem,     sizeof(ELEMENT) ) ;
  bc       =  (int *)     calloc( njoints,  sizeof(int)     ) ;
```

```
    Q        = (double *) calloc( neqn,     sizeof(double)  ) ;
    QQ       = (double *) calloc( neqn,     sizeof(double)  ) ;
    TotQQ    = (double *) calloc( neqn,     sizeof(double)  ) ;
    deltaQQ  = (double *) calloc( neqn,     sizeof(double)  ) ;
    SumGJF   = (double *) calloc( neqn,     sizeof(double)  ) ;
    Residual = (double *) calloc( neqn,     sizeof(double)  ) ;
    Kht      = (int *)    calloc( neqn,     sizeof(int)     ) ;
    Maxa     = (int *)    calloc( (neqn+1), sizeof(int)     ) ;

    /* -- Check Dynamic Memory Allocations -- */
    if ( element == NULL || bc == NULL )
    {
      puts(" GEOMETRY allocation failed...") ;
      status = ERROR ;
    }
    if ( Q == NULL || QQ == NULL || TotQQ == NULL || deltaQQ == NULL )
    {
      puts(" LOAD/DEFORMATION allocation failed...") ;
      status = ERROR ;
    }
    if ( SumGJF == NULL || Residual == NULL )
    {
      puts(" EQUILIBRIUM allocation failed...") ;
      status = ERROR ;
    }
    if ( Kht == NULL || Maxa == NULL )
    {
      puts(" SKYLINE allocation failed...") ;
      status = ERROR ;
    }
  } /* <<<<<< end second set of status checks >>>>>> */

  if ( status ) /* <<<<<< begin third set of status checks >>>>>> */
  {
    /* -- Set Up Models for Specification Check Post-Processors -- */
    fprintf( fp_asd_tmp, "%d %.2lf %d", nmem, Fy, fo_flag ) ;
    if ( !fo_flag ) fprintf( fp_lrfd_tmp, "%d %d %.2lf", nmem, process, Fy ) ;

    /* -- Read Element Geometry Data -- */
    printf("\n Reading Structure Geometry...") ;
    for ( mem = 0 ; mem < nmem ; mem++ )
      status = member_properties( &element[mem] ) ;

    /* -- Read Boundary Conditions -- */
    printf("\r Reading Boundary Conditions...") ;
    status = support_conditions( &nsupports, bc ) ;

    /* -- Calculate Skyline -- */
    printf("\r Determining Skyline Profile...") ;
    status = skyline( nmem, neqn, element, Kht, Maxa ) ;

    /* -- Pointer to Structure Stiffness Matrix  -- */
    if ( (ss = (double *) calloc(Maxa[neqn], sizeof(double))) == NULL )
    {
      puts(" STIFFNESS allocation failed...") ;
      status = ERROR ;
    }
  } /* <<<<<< end third set of status checks >>>>>> */

  if ( status ) /* <<<<<< begin fourth set of status checks >>>>>> */
  {
    /* Report Analysis Progress */
    if ( fo_flag )
      printf("\r Performing First-Order Elastic Analysis...\n\n" ) ;
    else
      printf("\r Performing Second-Order Elastic Analysis...\n\n" ) ;

    /* --- Assemble ONCE: First Order Analysis --- */
```

```
if ( fo_flag ) /* fo_flag == TRUE */
{
  ss_comps = FALSE ;
  result = assemble_stiffness_matrix( neqn, nmem, nsupports,
                                      bc, Maxa, ss, element, km ) ;
}
else /* fo_flag == FALSE */
  ss_comps = TRUE ;

/* Begin Analysis for All Load_Cases */
for ( load_case = 1 ; load_case <= Active_Load_Cases ; load_case++ )
{
  /* -- Write Progress to Output Device -- */
  printf("\r Solving Load Combination # %d", load_case) ;

  /* -- Initialize Member Parameters -- */
  if ( load_case == 1 )
    element_props( fo_flag, nmem, element ) ;

  /* -- Initialize Load Case Parameters -- */
  memset( TotQQ, 0, neqn*sizeof(double) ) ;

  /* -- Read Loads for this Load_Case -- */
  for ( i = 0 ; i < neqn ; i++ ) fscanf( fp_input, "%lf", &Q[i] ) ;

  /* -- Perform the Requested Structural Analysis --  */
  for ( loadstep = 1 ; loadstep <= numloadsteps ; loadstep++ )
  {
    for ( cycle = 1 ; cycle <= MAXCYCLE ; cycle++ )
    {
      /* --- Assemble EACH: Second Order Analysis --- */
      if ( !fo_flag && ss_comps )
        result = assemble_stiffness_matrix( neqn, nmem, nsupports,
                                            bc, Maxa, ss, element, km ) ;

      /* -- Assemble the Load Vector -- */
      result = assemble_load_vector( neqn, cycle, Q, QQ, Residual,
                                     element, nmem, load_case ) ;

      /* -- Solve the System of Equations -- */
      result = solve( ss, QQ, Maxa, neqn, load_case ) ;

      if ( result == NO_ERROR )
      {
        /* -- Update Displacement Vector -- */
        for ( i = 0 ; i < neqn; i++ ) TotQQ[i] += QQ[i] ;

        /* -- Calculate Internal Member Reactions -- */
        member_end_forces( loadstep, load_case, nmem, neqn, cycle,
                           nsupports, element, km, SumGJF, TotQQ, numloadsteps )
;

        /* -- Check if Equilibrium is Satified -- */
        stop = calc_residual( Residual, Q, QQ, SumGJF, loadstep, neqn,
                              nsupports, bc, cycle, deltaQQ) ;
        if ( stop )
        {
          cycle = MAXCYCLE ;
          if ( ( loadstep == numloadsteps )  )
          member_end_forces( loadstep, load_case, nmem, neqn, cycle,
                             nsupports, element, km, SumGJF, TotQQ, numloadsteps
) ;
        }

      }
      else
      {
        /* -- Print Error Message -- */
```

```
          printf("\r Error Encountered...[SS] NOT POSITIVE DEFINITE:") ;
          status = ERROR ;
        }

      } /* <<<<<< end loop on cycle >>>>>> */

    } /* <<<<<< end loop on loadstep >>>>>> */

    /* -- Print Displacements and Reactions -- */
    print_displacements( load_case, njoints, TotQQ ) ;
    support_reactions( load_case, nsupports, SumGJF, bc ) ;

  } /* <<<<<< end loop on load_case >>>>>> */

  /* -- Clean - Up -- */
  fcloseall() ;              /* Close Input & Output Files */
  forget ( element ) ;       /* Structural Model Pointer */
  forget ( Kht ) ;           /* Column Height Vector */
  forget ( Maxa ) ;          /* Main Diagonal Vector */
  forget ( Q ) ;             /* Load Vector */
  forget ( QQ ) ;            /* Incremental Displacements */
  forget ( TotQQ ) ;         /* Total Displacment Vector */
  forget ( deltaQQ ) ;       /* Total Incremental Displacments */
  forget ( Residual ) ;      /* Residual Force Vector */
  forget ( SumGJF ) ;        /* Global Joint Force Vector */
  forget ( ss ) ;            /* Structure Stiffness Vector */

 } /* <<<<<< end fourth set of status checks >>>>>> */

 /* Terminate PDELTA */
 if ( !status ) pause() ;
 return ( ((status) ? NO_ERROR : ERROR) ) ;

}
/*******************************************************************************/
/* Function member_properties                                                  */
/*******************************************************************************/
int member_properties( ELEMENT *element )
{
double theta ;

 /* -- Read: BF, TF, HC, TW, Da, Db -- */
 fscanf( fp_input, "%lf %lf %lf %lf %lf %lf ", &element->BF, &element->TF,
                                               &element->HC, &element->TW,
                                               &element->Da, &element->Db ) ;
 /* -- Read: LENGTH, theta -- */
 fscanf( fp_input, "%lf %lf", &element->LENGTH, &theta ) ;

 /* -- Read: MCODE -- */
 fscanf( fp_input; "%d %d %d %d %d %d", &element->MCODE[0], &element->MCODE[1],
                                        &element->MCODE[2], &element->MCODE[3],
                                        &element->MCODE[4], &element->MCODE[5]) ;

 /* -- Calculate and Store Required Section Properties -- */
 element->Ag = 2.0*(element->BF)*(element->TF)+(element->HC)*(element->TW) ;
 element->Ix = (element->BF)*pow((element->TF),3.0)/6.0 +
               0.5*(element->BF)*(element->TF)*pow(((element->HC)+(element->TF)),2.0)
+
               (element->TW)*pow((element->HC),3.0)/12.0 ;
 element->SW = (element->Ag)*(element->LENGTH)*DENSITY ;
 element->COST = cos(theta) ;
 element->SINT = sin(theta) ;

 /* -- Initialize Axial Force -- */
 element->P  = 0.0 ;

 /* Return to Main() */
 return ( NO_ERROR ) ;
```

```
}
/*****************************************************************************/
/* Function support_conditions                                              */
/*****************************************************************************/
/* nsupports ->  Number of Boundary Conditions  */
/* bc[nsup]  ->  Number of Restrained GDOF      */
/*************************************************/
int support_conditions( int *nsupports, int *bc )
{
int nsup ;

  /* Number of support conditions */
  fscanf( fp_input, "%d", nsupports ) ;

  /* Read the Boundary Conditions */
  for ( nsup = 0 ; nsup < *nsupports ; nsup++ )
    fscanf( fp_input, "%d", &bc[nsup] ) ;

  /* Return to Main() */
  return( NO_ERROR ) ;
}

/*****************************************************************************/
/* Function skyline                                                         */
/*****************************************************************************/
int skyline( int nmem, int neqn, ELEMENT *element, int *Kht, int *Maxa )
{
/* Local Variables */
register int i, j ;
int ndof, mem, dof[6] ;

  /* For all members of the structure... */
  for ( mem = 0 ; mem < nmem ; mem++ )
  {
    /* -- Extract Current Element MCODE -- */
    for ( ndof = 0 ; ndof < 6 ; ndof++ ) dof[ndof] = element[mem].MCODE[ndof] ;

    /* -- Parse MCODE for DOF Differentials: Active Column Heights -- */
    for( j = 0 ; j < 6 ; j++ )
      for( i = 0 ; i < 6 ; i++ )
        Kht[dof[j]] = min(dof[j], max(Kht[dof[j]],dof[j]-dof[i] )) ;
  }

  /* -- Map the Main Diagonal Element Locations -- */
  Maxa[0] = 0 ;
  for ( i = 0 ; i < neqn ; i++ ) Maxa[i+1] = Maxa[i] + Kht[i] + 1 ;

  /* Return to Main() */
  return ( NO_ERROR ) ;
}
/*****************************************************************************/
/* Function assemble_stiffness_matrix                                       */
/*****************************************************************************/
int assemble_stiffness_matrix( int neqn, int nmem, int nsupports, int *bc,
                               int *Maxa, double *ss, ELEMENT *element,
                               double (*km)[6] )
{
int i, j, k, m, dof[6] ;

  /* Initialize Structure Stiffness Matrix */
  memset( ss, 0, Maxa[neqn]*sizeof(double) ) ;

  /* Assemble For All Elements */
  for ( m = 0 ; m < nmem ; m++ )
  {

    /* -- Retrieve Element MCODE for Mapping to ss[]-- */
    for ( i = 0 ; i < 6 ; i++ ) dof[i] = element[m].MCODE[i] ;
```

```
    /* -- Assemble Element->System Tangent Stiffness Matrix -- */
    for ( j = 0 ; j < 6 ; j++ )
      for ( k = 0 ; k < 6 ; k++ )
      {
        if ( dof[k] > dof[j] ) continue ;
        i = Maxa[dof[j]] + dof[j] - dof[k] ;
        ss[i] += km[j][k] ;
      }
  }

  /* Invoke Boundary Conditions by PENALTY Method */
  for ( i = 0 ; i < nsupports ; i++ ) ss[Maxa[bc[i]]] += 1.0e50 ;

  /* Return to Main() */
  return( NO_ERROR ) ;
}
/*****************************************************************************/
/* Function member_stiffness [TANGENT STIFFNESS]                            */
/*****************************************************************************/
void member_stiffness( ELEMENT *element, double (*km)[6] )
{
/* Local Variables */
double L, alpha, beta, cost, sint, GD[6], GF[6], LF[6],
       g1, g2, g3, g4, g5, g6, g7, phi, phic, phit, c1, c2, c3,
       cos_2t, sin_2t ;
int i, j, axial_ind ;

  /* -- Common Factors for Stiffness Coefficients -- */
  L      = element->LENGTH ;
  alpha  = E*(element->Ix)/L/L/L ;
  beta   = (element->Ag)*L*L/(element->Ix) ;

  /* -- Factor Rotational Stiffness for P-Delta Analysis -- */
  if ( fabs(element->P) <= ZERO_STIFFNESS ) axial_ind = 0 ;
  else if ( element->P > 0.0 )              axial_ind = 1 ;
  else                                      axial_ind = 2 ;
  switch ( axial_ind )
  {
    case 0:            /* No Axial Force, First Order */
      {
        c1 = 4.0 ;
        c2 = 2.0 ;
        c3 = 6.0 ;
      }
      break ;
    case 1:            /* Axial Compression, Second-Order */
      {
        phi  = L*sqrt(element->P/(E*element->Ix)) ;
        phic = 2.0*(1.0-cos(phi))-phi*sin(phi)    ;
        c1   = phi*(sin(phi)-phi*cos(phi))/phic   ;
        c2   = phi*(phi-sin(phi))/phic            ;
        c3   = c1 + c2                            ;
      }
      break ;
    case 2:            /* Axial Tension, Second-Order */
      {
        phi  = L*sqrt(-element->P/(E*element->Ix)) ;
        phit = 2.0*(1.0-cosh(phi))+phi*sinh(phi)   ;
        c1   = phi*(phi*cosh(phi)-sinh(phi))/phit  ;
        c2   = phi*(sinh(phi)-phi)/phit            ;
        c3   = c1 + c2                             ;
      }
  }

  /* Element Direction Cosines */
  cost = element->COST ;
  sint = element->SINT ;
  cos_2t = cost*cost ;
```

```c
  sin_2t = sint*sint ;

  /* Tangent Stiffness Elements Computations */
  /* -- References                           */
  /* -- Holzer.......................p. 126 */
  /* -- Oran.........................p. 973 */
  g1 =  alpha * ( beta*cos_2t + 2.0*sin_2t*c3 ) ;
  g2 =  alpha * cost*sint * ( beta - 2.0*c3 )   ;
  g3 =  alpha * ( beta*sin_2t + 2.0*cos_2t*c3 ) ;
  g4 = -alpha * c3*L*sint                       ;
  g5 =  alpha * c3*L*cost                       ;
  g6 =  alpha * c1*L*L                          ;
  g7 =  alpha * c2*L*L                          ;

  /* - Upper Triangular Elements - */
  /* -- Row 1 -- */
  km[0][0] =  g1 ; km[0][1] =  g2 ; km[0][2] =  g4 ;
  km[0][3] = -g1 ; km[0][4] = -g2 ; km[0][5] =  g4 ;
  /* -- Row 2 -- */
  km[1][1] =  g3 ; km[1][2] =  g5 ; km[1][3] = -g2 ;
  km[1][4] = -g3 ; km[1][5] =  g5 ;
  /* -- Row 3 -- */
  km[2][2] =  g6 ; km[2][3] = -g4 ; km[2][4] = -g5 ;
  km[2][5] =  g7 ;
  /* -- Row 4 -- */
  km[3][3] =  g1 ; km[3][4] =  g2 ; km[3][5] = -g4 ;
  /* -- Row 5 -- */
  km[4][4] =  g3 ; km[4][5] = -g5 ;
  /* -- Row 6 -- */
  km[5][5] =  g6 ;

  /* Exploit Symmetry of [km] */
  for ( i = 0 ; i < 5 ; i++ )
    for ( j = (i+1) ; j < 6 ; j++ ) km[j][i] = km[i][j] ;

  /* Return to Assemble_Stiffness Matrix() */
  return ;
}
/******************************************************************************/
/* Function assemble_load_vector                                              */
/******************************************************************************/
/* QQ[row] ->  Value of Load at GDOF (row)     */
/* or, when cycle != 1  the Residual load      */
/* Q - F is reapplied to the structure.        */
/***********************************************/
int assemble_load_vector( int neqn, int cycle, double *Q, double *QQ,
                          double *Residual, ELEMENT *element, int nmem,
                          int load_case  )
{
int row, mem ;

  if ( cycle == 1 )
  {
    double inc, load_fact, node_fact ;

    /* Calculate Incremental Load Vector */
    inc = 1.0/((double)numloadsteps) ;

    /* -- Externally Applied Load -- */
    for ( row = 0 ; row < neqn ; row++ ) QQ[row] = inc*Q[row] ;

    /* -- Selfweight Nodal Loads -- */
    switch ( load_case )
    {
      case 1:
      case 2:
      {
        load_fact = -1.0 ;
```

```
      }
      break ;
      case 3:
      case 4:
      case 5:
      case 6:
      case 7:
      case 8:
      case 9:
      case 10:
      case 11:
      case 12:
      case 13:
      case 14:
      case 15:
      case 16:
      {
        load_fact = -0.75 ;
      }
      break ;
      default:
      {
        load_fact = -1.2 ;
      }
    }
    node_fact = 0.5*inc*load_fact ;

    for ( mem = 0 ; mem < nmem ; mem++ )
    {
      QQ[element[mem].MCODE[1]] += node_fact*element[mem].SW ;
      QQ[element[mem].MCODE[4]] += node_fact*element[mem].SW ;
    }
  }

  else
  {
    /* -- Residual Load Vector -- */
    for ( row = 0 ; row < neqn ; row++ ) QQ[row] = Residual[row] ;
  }

  /* Return to Main() */
  return ( NO_ERROR ) ;
}
/*****************************************************************************/
/* Function solve                                                            */
/*****************************************************************************/
int solve( double *ss, double *QQ, int *Maxa, int neqn, int load_case )
{
/* Local Variables */
int status ;

  if ( fo_flag )                                          // FOEA
  {
    if ( load_case == 1 )
      status = ( factor(ss, Maxa, neqn ) == NOTPOSITIVEDEF ) ?
                 NOTPOSITIVEDEF : NO_ERROR ;
    else /* 1 < load_case < Active_Load_Cases */
      status = NO_ERROR ;
  }
  else                                                    // SOEA
    status = ( factor(ss, Maxa, neqn ) == NOTPOSITIVEDEF ) ?
               NOTPOSITIVEDEF : NO_ERROR ;

  if ( status == NO_ERROR )
  {
    forsub(ss, QQ, Maxa, neqn) ;
    bacsub(ss, QQ, Maxa, neqn) ;
  }
```

```
  /* Return to Main() */
  return( status ) ;
}
/******************************************************************************/
/* Function factor                                                            */
/******************************************************************************/
int factor( double *ss, int *Maxa, int neqn )
{
/* Local Variables */
register int counter1, counter2, counter3;
int ic, k, kh, ki, kk, kl, klt, kn, ku, nd ;
double b, c ;

  for ( counter1 = 0 ; counter1 < neqn ; counter1++ )
  {
    kn = Maxa[counter1] ;
    kl = kn + 1 ;
    ku = Maxa[counter1 + 1] - 1 ;
    kh = ku - kl ;

    if (kh < 0)
    {
       if ( fabs(ss[kn]) <= ZERO_STIFFNESS )
       {
         return(NOTPOSITIVEDEF) ;
       }
    }
    else if ( !kh )
    {
      goto Zero ;
    }

    k = counter1 - kh ;
    ic = 0;
    klt = ku;
    for ( counter2 = 0 ; counter2 < kh ; counter2++ )
    {
      ic++ ;
      klt-- ;
      ki = Maxa[k] ;
      nd = Maxa[k + 1] - ki - 1 ;
      if ( nd > 0 )
      {
        kk = (ic < nd ? ic : nd) ;
        c = 0.0 ;
        for ( counter3 = 1 ; counter3 <= kk ; counter3++ )
          c += ss[ki+counter3] * ( ss[klt+counter3] ) ;
        ss[klt] -= c ;
      }
      k++ ;
    }

    Zero:
    k = counter1 ;
    b = 0.0 ;
    for ( kk = kl ; kk <= ku ; kk++ )
    {
      k-- ;
      ki = Maxa[k] ;
      c = ss[kk] / ss[ki] ;
      b += c * ss[kk] ;
      ss[kk] = c ;
    }
    ss[kn] -= b ;

  }

  /* Return to solve() */
```

```
  return( NO_ERROR ) ;
}

/*****************************************************************************/
/* Function forsub                                                           */
/*****************************************************************************/
void forsub(double *ss, double *QQ, int *Maxa, int neqn)
{
/* Local Variables */
double c ;
int count, kl, ku, kh, kk, k, i ;

  for ( count = 0 ; count < neqn ; count++ )
  {
    kl = Maxa[count] + 1 ;
    ku = Maxa[count + 1] - 1 ;
    kh = ku - kl ;
    if ( kh >= 0 )
    {
      k = count ;
      c = 0.0 ;
      for ( kk = kl ; kk <= ku ; kk++ )
      {
        k--;
        c += ss[kk] * QQ[k] ;
      }
      QQ[count] -= c ;
    }
  }

  /* Return to solve() */
  return ;
}
/*****************************************************************************/
/* Function bacsub                                                           */
/*****************************************************************************/
void bacsub( double *ss, double *QQ, int *Maxa, int neqn )
{
/* Local Variables */
int count1, count2, k, kl, ku, kk, kh, i ;

  for ( count1 = 0 ; count1 < neqn ; count1++ )
  {
    k = Maxa[count1] ;
    QQ[count1] /= ss[k] ;
  }

  if ( neqn == 1 ) return ;

  count1 = neqn - 1 ;

  for ( count2 = 1 ; count2 < neqn ; count2++ )
  {
    kl = Maxa[count1] + 1 ;
    ku = Maxa[count1 + 1] - 1 ;
    kh = ku - kl ;
    if ( kh >= 0 )
    {
      k = count1 ;
      for ( kk = kl ; kk <= ku ; kk++ )
      {
        k-- ;
        QQ[k] -= ss[kk] * QQ[count1] ;
      }
    }
    count1-- ;
  }
```

```
  /* Return to solve */
  return ;
}
/*****************************************************************************/
/* Function print_displacements                                              */
/*****************************************************************************/
void print_displacements( int load_case, int njoints, double *TotQQ )
{
/* Local Variables */
int jt, dof ;

  /* -- Adjust "Zero" Displacements -- */
  for ( dof = 0 ; dof < 3*njoints ; dof++ )
  {
    if ( fabs(TotQQ[dof]) < 1.0e-6 ) TotQQ[dof] = 0.0 ;
  }

  /* Print to Appropriate File */
  if ( load_case <= 16 )
  {
    fprintf( fp_asd_dis, "\n ASD Load Case No. %d\n", load_case ) ;
    fprintf( fp_asd_dis, "\n   Jt       Global Displacements") ;
    fprintf( fp_asd_dis, "\n    #     X[in]     Y[in]     Z[rad]") ;
      for ( jt = 0 ; jt < njoints ; jt++ )
      {
        fprintf( fp_asd_dis, "\n %4d", jt+1 ) ;
        for ( dof = 0 ; dof < 3 ; dof++ )
          fprintf( fp_asd_dis, "% 10.6lf", *(TotQQ+dof+3*jt) ) ;
      }
    fprintf( fp_asd_dis, "\n" ) ;
  }
  else
  {
    fprintf( fp_lrfd_dis, "\n LRFD Load Case No. %d\n", (load_case-16) ) ;
    fprintf( fp_lrfd_dis, "\n   Jt       Global Displacements") ;
    fprintf( fp_lrfd_dis, "\n    #     X[in]     Y[in]     Z[rad]") ;
      for ( jt = 0 ; jt < njoints ; jt++ )
      {
        fprintf( fp_lrfd_dis, "\n %4d", jt+1 ) ;
        for ( dof = 0 ; dof < 3 ; dof++ )
          fprintf( fp_lrfd_dis, "% 10.6lf", *(TotQQ+dof+3*jt) ) ;
      }
    fprintf( fp_lrfd_dis, "\n" ) ;
  }

  /* Return to Main() */
  return ;
}
/*****************************************************************************/
/* Function support_reactions                                                */
/*****************************************************************************/
void support_reactions( int load_case, int nsupports, double *SumGJF, int *bc )
{
int row ;

  /* -- ASD Load Cases -- */
  if ( load_case <= 16 )
  {
    fprintf( fp_asd_dis, "\n   DOF     Support Reactions") ;
    fprintf( fp_asd_dis, "\n    #") ;
    fprintf( fp_asd_dis, "\n ... ") ;
    for ( row = 0 ; row < nsupports ; row++ )
      fprintf( fp_asd_dis, "\n %4d    % 10.4lf", *(bc+row)+1, SumGJF[bc[row]] ) ;
    fprintf( fp_asd_dis, "\n\n" ) ;
  }
  /* -- LRFD Load Cases -- */
  else
  {
    fprintf( fp_lrfd_dis, "\n   DOF     Support Reactions") ;
```

```
    fprintf( fp_lrfd_dis, "\n    #") ;
    for ( row = 0 ; row < nsupports ; row++ )
      fprintf( fp_lrfd_dis, "\n %4d    % 10.4lf", *(bc+row)+1, SumGJF[bc[row]] ) ;
    fprintf( fp_lrfd_dis, "\n\n" ) ;
  }

  /* Return to Main() */
  return ;
}
/****************************************************************************/
/* Function member_end_forces [SECANT STIFFNESS]                            */
/****************************************************************************/
void member_end_forces( int loadstep, int load_case, int nmem,
                        int neqn, int cycle, int nsupports,
                        ELEMENT *element, double (*km)[6],
                        double *SumGJF, double *TotQQ,
                        int numloadsteps )
{
/* Local Variables */
double L, alpha, beta, cost, sint, GD[6], GF[6], LF[6],
       g1, g2, g3, g4, g5, g6, g7, phi, phic, phit,
       phi1, phi2, phi3, phi4, load_fact, factor, node_fact,
       cos_2t, sin_2t, half_self_weight, sign ;
int mem, i, j, axial_ind ;

  /* -- Initialize Joint Forces -- */
  memset( SumGJF, 0, neqn*sizeof(double) ) ;

  /* -- Load Factors for Structure Selfweight -- */
  factor = ((double)loadstep)/((double)numloadsteps) ;
  switch ( load_case )
  {
    case 1:                                   // ASD Load Cases +0 inc.
    case 2:
    {
      load_fact = -1.0 ;
    }
    break ;
    case 3:                                   // ASD Load Cases +1/3 inc.
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
    case 11:
    case 12:
    case 13:
    case 14:
    case 15:
    case 16:
    {
      load_fact = -0.75 ;
    }
    break ;
    default:                                  // LRFD Load Cases
    {
      load_fact = -1.2 ;
    }
  }
  node_fact = 0.5*factor*load_fact ;

  /* -- Loop For All Members -- */
  for ( mem = 0 ; mem < nmem ; mem++ )
  {
    /* -- Common Factors for Stiffness Coefficients -- */
    L     = element[mem].LENGTH ;
```

```
alpha = E*(element[mem].Ix)/L/L/L ;
beta = (element[mem].Ag)*L*L/(element[mem].Ix) ;
half_self_weight = node_fact*element[mem].SW ;

/* -- Factor Rotational Stiffness for P-Delta Analysis -- */
if ( fabs(element[mem].P) <= ZERO_STIFFNESS ) axial_ind = 0 ;
else if ( element[mem].P > 0.0 )              axial_ind = 1 ;
else                                          axial_ind = 2 ;
switch ( axial_ind )
{
  case 0:              /* No Axial Force, First Order */
    {
      phi1 = phi2 = phi3 = phi4 = 1.0 ;
    }
    break ;
  case 1:              /* Axial Compression, Second-Order */
    {
      phi  = L*sqrt((element[mem].P)/(E*element[mem].Ix)) ;
      phic = 2.0*(1.0-cos(phi))-phi*sin(phi)              ;
      phi1 = phi*phi*phi*sin(phi)/(12.0*phic)             ;
      phi2 = phi*phi*(1.0-cos(phi))/(6.0*phic)            ;
      phi3 = phi*(sin(phi)-phi*cos(phi))/(4.0*phic)       ;
      phi4 = phi*(phi-sin(phi))/(2.0*phic)                ;
    }
    break ;
  case 2:              /* Axial Tension, Second-Order */
    {
      phi  = L*sqrt((-element[mem].P)/(E*element[mem].Ix)) ;
      phit = 2.0*(1.0-cosh(phi))+phi*sinh(phi)             ;
      phi1 = phi*phi*phi*sinh(phi)/(12.0*phit)             ;
      phi2 = phi*phi*(cosh(phi)-1.0)/(6.0*phit)            ;
      phi3 = phi*(phi*cosh(phi)-sinh(phi))/(4.0*phit)      ;
      phi4 = phi*(sinh(phi)-phi)/(2.0*phit)                ;
    }
}

/* Element Direction Cosines */
cost = element[mem].COST ;
sint = element[mem].SINT ;
cos_2t = cost*cost ;
sin_2t = sint*sint ;

/* Secant Stiffness Elements Computations */
/* -- References                          */
/* -- Holzer.......................p. 126 */
/* -- Chen & Lui...................p. 257 */
g1 =  alpha * ( beta*cos_2t + 12.0*sin_2t*phi1 ) ;
g2 =  alpha * cost*sint * ( beta - 12.0*phi1 )   ;
g3 =  alpha * ( beta*sin_2t + 12.0*cos_2t*phi1 ) ;
g4 = -alpha * 6.0*L*sint*phi2                    ;
g5 =  alpha * 6.0*L*cost*phi2                    ;
g6 =  alpha * 4.0*L*L*phi3                       ;
g7 =  alpha * 2.0*L*L*phi4                       ;

/* Upper Triangular Elements */
/* -- Row 1 -- */
km[0][0] =  g1 ; km[0][1] =  g2 ; km[0][2] =  g4 ;
km[0][3] = -g1 ; km[0][4] = -g2 ; km[0][5] =  g4 ;
/* -- Row 2 -- */
km[1][1] =  g3 ; km[1][2] =  g5 ; km[1][3] = -g2 ;
km[1][4] = -g3 ; km[1][5] =  g5 ;
/* -- Row 3 -- */
km[2][2] =  g6 ; km[2][3] = -g4 ; km[2][4] = -g5 ;
km[2][5] =  g7 ;
/* -- Row 4 -- */
km[3][3] =  g1 ; km[3][4] =  g2 ; km[3][5] = -g4 ;
/* -- Row 5 -- */
km[4][4] =  g3 ; km[4][5] = -g5 ;
```

```
    /* -- Row 6 -- */
    km[5][5] =  g6 ;

    /* Exploit Symmetry of [km] */
    for ( i = 0 ; i < 5 ; i++ )
      for ( j = (i+1) ; j < 6 ; j++ ) km[j][i] = km[i][j] ;

    /* Extract Global_Displacements @ Active Joints */
    for ( i = 0 ; i < 6 ; i++ ) GD[i] = TotQQ[element[mem].MCODE[i]] ;

    /* Calculate Global Member End_Forces */
    for ( i = 0 ; i < 6 ; i++ )
    {
      GF[i] = 0.0 ;
      for( j = 0 ; j < 6 ; j++ ) GF[i] += km[i][j]*GD[j] ;
    }

    /* -- Global Joint Forces -> ( {f+fp} = [k]{d}  ) -- */
    for ( i = 0 ; i < 6 ; i++ ) SumGJF[ element[mem].MCODE[i] ] += GF[i] ;

    /* -- Global Joint Forces -> ( {f} = [k]{d} - {fp} ) -- */
    /* -<- Selfweight, applied Vertically at End Nodes ->- */
    SumGJF[element[mem].MCODE[1]] -= half_self_weight ;
    SumGJF[element[mem].MCODE[4]] -= half_self_weight ;

    /* Calculate Axial Force for next SOA Cycle: {f} = [k]{d} - {fp} */
    if ( !ss_comps ) element[mem].P = 0.0 ;
    else
    {
      double P1, P2 ;
      P1 =  cost*GF[0] + sint*(GF[1]-half_self_weight) ;
      P2 =  cost*GF[3] + sint*(GF[4]-half_self_weight) ;
      if ( P1 > 0 ) sign =  1.0 ;
      else          sign = -1.0 ;
      element[mem].P = sign * max( (fabs(P1)) , (fabs(P2)) ) ;
    }

    /* Save Forces to Files on Completion */
    if ( cycle == MAXCYCLE && loadstep == numloadsteps )
    {
      int index ;
      FILE *fp_tmp ;

      /* -- Local Forces -> {f} = [k]{d} - {fp} -- */
      LF[0] =  cost*GF[0] + sint*(GF[1]-half_self_weight) ;
      LF[1] = -sint*GF[0] + cost*(GF[1]-half_self_weight) ;
      LF[2] =  GF[2] ;
      LF[3] =  cost*GF[3] + sint*(GF[4]-half_self_weight) ;
      LF[4] = -sint*GF[3] + cost*(GF[4]-half_self_weight) ;
      LF[5] =  GF[5] ;

      /* -- Store -- */
      fp_tmp = ( load_case <= 16 ) ? fp_asd_tmp : fp_lrfd_tmp ;
      for ( index = 0 ; index < 6 ; index++ )
      {
        if ( fabs(LF[index]) < ZERO_STIFFNESS ) LF[index] = 0.0 ;
        fprintf( fp_tmp, " %.4lf", LF[index] ) ;
      }

    }

  } /* <<<<<< end loop on mem >>>>>> */

  /* Return to Main() */
  return ;
}

/***************************************************************************************/
```

```
/* Function element_props                                                     */
/******************************************************************************/
void element_props( int fo_flag, int nmem, ELEMENT *element )
{
double bf, tf, tw, hca, hcb, Aga, Agb, Sxa, Sxb, Zxa, Zxb, Ixa, Ixb, Iy ;
int mem ;

  /* -- Calculations of Section Properties for Specification Checking -- */
  /* -- Store Properties at A-End & B-End for Proper Interaction Sums -- */
  for ( mem = 0 ; mem < nmem ; mem++ )
  {
    bf = element[mem].BF ;
    tf = element[mem].TF ;
    tw = element[mem].TW ;
    hca = element[mem].Da - 2.0*tf ;
    hcb = element[mem].Db - 2.0*tf ;
    Aga = 2.0*bf*tf+hca*tw ;
    Agb = 2.0*bf*tf+hcb*tw ;
    Ixa = (bf*pow(tf,3.0)/6.0)+(bf*tf*(hca+tf)*(hca+tf)/2.0)+(tw*pow(hca,3.0)/12.0);
    Ixb = (bf*pow(tf,3.0)/6.0)+(bf*tf*(hcb+tf)*(hcb+tf)/2.0)+(tw*pow(hcb,3.0)/12.0);
    Sxa = 2.0*Ixa/element[mem].Da ;
    Sxb = 2.0*Ixb/element[mem].Db ;
    Iy  = tf*pow(bf,3.0)/6.0 ;

    /*** -- Store in ASD temporary file -- ***/
    fprintf( fp_asd_tmp, " %.4lf %.4lf %.4lf", bf, tf, tw ) ;
    fprintf( fp_asd_tmp, " %.4lf %.3lf %.3lf %.3lf", hca, Aga, Ixa, Sxa ) ;
    fprintf( fp_asd_tmp, " %.4lf %.3lf %.3lf %.3lf", hcb, Agb, Ixb, Sxb ) ;
    fprintf( fp_asd_tmp, " %.3lf", Iy ) ;

    if ( !fo_flag )
    /*** -- Store in LRFD temporary file -- ***/
    {
      Zxa = bf*tf*(hca+tf) + hca*hca*tw/4.0 ;
      Zxb = bf*tf*(hcb+tf) + hcb*hcb*tw/4.0 ;
      fprintf(fp_lrfd_tmp," %.4lf %.4lf %.4lf", bf, tf, tw ) ;
      fprintf(fp_lrfd_tmp," %.4lf %.3lf %.3lf %.3lf %.3lf",hca,Aga,Ixa,Sxa,Zxa) ;
      fprintf(fp_lrfd_tmp," %.4lf %.3lf %.3lf %.3lf %.3lf",hcb,Agb,Ixb,Sxb,Zxb) ;
      fprintf(fp_lrfd_tmp," %.3lf", Iy ) ;
    }

  } /** next [mem]ber **/

  /* -- Return to main() -- */
  return ;

} /*** end of element_props ***/
/******************************************************************************/
/* Function calc_residual                                                     */
/******************************************************************************/
int calc_residual( double *Residual, double *Q, double *QQ,
                   double *SumGJF, int loadstep, int neqn, int nsupports,
                   int *bc, int cycle, double *deltaQQ )
{
/* Local Variables */
double factor, num_d, den_d, disp_check, force_check ;
int k, flag ;

  /* Second Order Analysis - Calculate Convergence Parameters */
  if ( ss_comps )
  {
    /* Initialize Steps - Sum Increment */
    if ( cycle == 1 )
      memcpy( deltaQQ, QQ, neqn*sizeof(double) ) ;
    else
      for ( k = 0 ; k < neqn ; k++ ) deltaQQ[k] += QQ[k] ;
```

```
    /* Displacement Convergence Parameters */
    qsort(    QQ   , neqn, sizeof(double), sort_descending ) ;
    qsort( deltaQQ, neqn, sizeof(double), sort_descending ) ;
    num_d = fabs(*QQ) ;                       // Maximum Increment this Cycle
    den_d = fabs(*deltaQQ) ;                  // Maximum Displacement this Step
    den_d = max(den_d, CPDB) ;                // Trap Division by Zero
    disp_check = (num_d/den_d/CPDB) ;         // Normalized Displacement Check

    /* -- Calculate External Load, Form the Residual -- */
    factor = ((double)loadstep)/((double)numloadsteps) ;
    for ( k = 0 ; k < neqn ; k++ )
    {
      QQ[k] = factor * Q[k] ;                 // Q
      Residual[k] = QQ[k] - SumGJF[k] ;       // Q - F
    }

    /* Absorb Residuals at Boundaries */
    for ( k = 0 ; k < nsupports ; k++ ) Residual[bc[k]] = 0.0 ;

    /* Force Convergence Parameters */
    memcpy( SumGJF, Residual, neqn*sizeof(double) ) ;
    qsort( SumGJF, neqn, sizeof(double), sort_descending) ;
    force_check = fabs(*SumGJF)/CPFB ;        // Normalized Force Check

  }
  else /* First Order Analysis Was Requested */
  {
    disp_check = force_check = 1.0 ;
  }

  /* -- Equilibrium Check -- */
  flag = ( (disp_check <= 1.0) && (force_check <= 1.0) ) ? TRUE : FALSE ;

  /* Return to Main() */
  return( flag ) ;
}
/*****************************************************************************/
/* Function sort_descending (called by qsort in calc_residual)              */
/*****************************************************************************/
int sort_descending( double *value1, double *value2 )
{
int result ;

  /* Define qsort Ordering Rules */
  if ( fabs(*value1) > fabs(*value2) )        result = -1 ;     // Larger
  else if ( fabs(*value1) < fabs(*value2) )   result =  1 ;     // Smaller
  else                                        result =  0 ;     // Equal

  return(result) ;
}
/*****************************************************************************/
/* End PDELTA Program                                                        */
/*****************************************************************************/
```

ASDSPEC.c

```
/*****************************************************************************/
/*
* ASDSPEC.h      Header File for ASDSPEC Program
*/
/********************************** CONSTANTS **********************************/
#define E       29000.0
#define PI      3.141592653589793
#define TRUE    1
#define FALSE   0
/********************************** MACROS ************************************/
#define forget(x)                    { if(x) free((void *)x) ; x = NULL ; }
/********************************* STRUCTURES *********************************/
typedef struct ELEMENT
{
  double BF,                              /* Flange Width                   */
         TF,                              /* Flange Thickness               */
         TW,                              /* Web Thickness                  */
         HCA,                             /* Web Height, A-End              */
         AGA,                             /* Area of Element A-End          */
         IXA,                             /* X-X Moment of Inertia A-End    */
         SXA,                             /* Elastic Section Modulus A-End  */
         HCB,                             /* Web Height, B-End              */
         AGB,                             /* Area of Element B-End          */
         IXB,                             /* X-X Moment of Inertia B-End    */
         SXB,                             /* Elastic Section Modulus B-End  */
         IY;                              /* Y-Y Moment of Inertia          */
} ELEMENT ;
/********************************* FUNCTIONS **********************************/
extern int main(void);
extern void read_input(int i);
extern void code_check(int i);
extern double axial_capacity(int i);
extern double shear_capacity(void);
extern double moment_capacity(int i);
extern void combined_stress(double Pa,double Va,double Ma,int i);
extern void critical_ratios(int i,double stability,double capacity,double shear);
extern int member_properties(ELEMENT *element);
extern void pause(void);
/**************************** Character Strings ******************************/
/*
*  Load Combination Titles for ASD Analysis\Design
*/
static char *str[] = { "1.00 * ( D + SL + SR )" ,
                       "1.00 * ( D + SL + SR/2 )" ,
                       "0.75 * ( D + WLL1 + WLR1 )" ,
                       "0.75 * ( D + WLL2 + WLR2 )" ,
                       "0.75 * ( D + SL + SR + WLL1/2 + WLR1/2 )" ,
                       "0.75 * ( D + SL + SR + WLL2/2 + WLR2/2 )" ,
                       "0.75 * ( D + SL + SR/2 + WLL1/2 + WLR1/2 )" ,
                       "0.75 * ( D + SL + SR/2 + WLL2/2 + WLR2/2 )" ,
                       "0.75 * ( D + SL + SR/2 + WRL1/2 + WRR1/2 )" ,
                       "0.75 * ( D + SL + SR/2 + WRL2/2 + WRR2/2 )" ,
                       "0.75 * ( D + SL/2 + SR/2 + WLL1 + WLR1 )" ,
                       "0.75 * ( D + SL/2 + SR/2 + WLL2 + WLR2 )" ,
                       "0.75 * ( D + SL/2 + SR/4 + WLL1 + WLR1 )" ,
                       "0.75 * ( D + SL/2 + SR/4 + WLL2 + WLR2 )" ,
                       "0.75 * ( D + SL/2 + SR/4 + WRL1 + WRR1 )" ,
                       "0.75 * ( D + SL/2 + SR/4 + WRL2 + WRR2 )" } ;
```

```
/******************************************************************************/
/*                                                                            */
/*  ASDSPEC.c                                                                 */
/*                                                                            */
/*  Copyright 1990, PC Structural Engineers                                   */
/*                  Blacksburg, Virginia 24060                                */
/*                                                                            */
/*  Research Team:                                                            */
/*  Eric J. Wishart.........................Research Investigator             */
/*  Thomas M. Murray, PhD...................Principal Investigator            */
/*                             Of The                                         */
/*       Charles E. Via, Jr. Department of Civil Engineering                  */
/*        Virginia Polytechnic Institute & State University                   */
/*                    Blacksburg, Virginia 24061                              */
/*                                                                            */
/*  Sponsor: NUCOR Metal Building Products                                    */
/*           Waterloo, Indiana                                                */
/*                                                                            */
/*  Performs Code Checking of Beams, Columns, and                             */
/*  Beam-Columns according to the AISC 9th Edition ASD Specification.         */
/*  Second Order Analysis Results NOT REQUIRED.                               */
/*                                                                            */
/*  Only error recognized at this point, is the calculation of Cb.  By        */
/*  definition, it is calculated using the moments at the ends of the         */
/*  unbraced length (unless larger in-between).  This program assumes         */
/*  the "end" moments are located at each discretized element "end".         */
/*  The error in the calculation will be a conservative one.                  */
/*                                                                            */
/*  InputFile:          ASD.tmp                                               */
/*  OutputFile:         ASD.chk                                               */
/*                                                                            */
/******************************************************************************/
/* INCLUDE files */
#include <stdio.h>                                  /* Microsoft Library */
#include <math.h>
#include <fcntl.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#include <conio.h>
#include "ASDSPEC.h"                                /* ASDSPEC header file */

/* DEFINE constants */
#define ERROR    0
#define NO_ERROR 1

/* Global Variables */
double P,V,M,M1,M2,Sx,bf,tf,hc,tw,Ix,Iy,Ag,
       KLx[200],KLy[200],Fy,Fa,Fv,Fb,rx,ry,KL_rx,KL_ry,KLmax,FS,
       kc,lambda_FLB,lambda_WLB,lp_FLB,lr_FLB,lp_WLB,lr_WLB,
       Sx, Ix, Ag, *SR, *CR, *VR, Q, FORCE[6] ;
int nmem,i,load_case, fo_flag, *Stab_Case, *Cap_Case, *Shear_Case ;
ELEMENT *element ;
FILE *in, *in2, *out, *max_ratios ;

/* Main Program */
main()
{
int status = NO_ERROR ;

  /* Open Output File for Writing */
  out = fopen("ASD.chk","w") ;
  max_ratios = fopen("ASD.sum","w") ;
  if ( out == NULL || max_ratios == NULL )
  {
    printf("\n CANNOT OPEN FILES ASD.chk or ASD.sum") ;
    status = ERROR ;
    pause() ;
```

```
  }

  /* Open Input Files for Reading */
  in = fopen("ASD.tmp","rb") ;
  in2 = fopen("KL.fct","r") ;
  if ( in == NULL || in2 == NULL )
  {
    printf("\n CANNOT OPEN FILES, ASD.tmp or KL.tmp") ;
    status = ERROR ;
    pause() ;
  }

  if ( status )
  {
    /* Report and Progress With Solution  */
    printf("\n ASD 9th Edition Specification Check In Progress...\n\n") ;
    fscanf(in, "%d %lf %d", &nmem, &Fy, &fo_flag );

    /* Dynamic Memory Allocation */
    element    =  (ELEMENT *) calloc( nmem, sizeof(ELEMENT) ) ;
    SR         =  (double *)  calloc( nmem, sizeof(double)  ) ;
    CR         =  (double *)  calloc( nmem, sizeof(double)  ) ;
    VR         =  (double *)  calloc( nmem, sizeof(double)  ) ;
    Stab_Case  =  (int *)     calloc( nmem,    sizeof(int)  ) ;
    Cap_Case   =  (int *)     calloc( nmem,    sizeof(int)  ) ;
    Shear_Case =  (int *)     calloc( nmem,    sizeof(int)  ) ;

    /* -- Check Dynamic Memory Allocations -- */
    if ( (element == (ELEMENT *) NULL)  )
      status = ERROR ;
    if ( (SR == NULL) || (CR == NULL) || (VR == NULL) )
      status = ERROR ;
    if ( (Stab_Case == NULL) || (Cap_Case == NULL) || (VR == NULL) )
      status = ERROR ;
    if ( !status )
    {
      fcloseall() ;
      return( ERROR ) ;
    }

    /* Read in Member Properties & Bracing Factors */
    for ( i = 0 ; i < nmem ; i++ )
    {
      status = member_properties( &element[i] ) ;
      fscanf(in2, "%lf %lf", &KLx[i], &KLy[i] ) ;
    }
    fclose( in2 ) ;

    /* Process All Load Combinations */
    for ( load_case = 0 ; load_case < 16 ; load_case++ )
    {
      /* -- Report Progress -- */
      printf("\r ASD Load Combination No. %d", load_case+1 ) ;

      /* -- Title to OutputFile -- */
      fprintf(out, "\n ASD Load Combination No. %d", load_case+1 ) ;
      fprintf(out, "\n ==========================\n\n" ) ;
      fprintf(out,"\t%s\n", str[load_case] ) ;
      fprintf(out, "\n Mem      P        V        M       Pa       Va       Ma
Stability Capacity" );
      fprintf(out, "\n  #      [k]      [k]     [kft]     [k]      [k]     [kft]
[Unity]   [Unity]" );
      fprintf(out, "\n ----  -------  -------  --------  -------  -------  --------
--------  ---------");

      /* -- Process All Members -- */
      for ( i = 0 ; i < nmem ; i++ )
      {
```

```
        read_input(i) ;

        /** -- Process a-end -- **/
        fprintf(out,"\n %3da",i+1) ;
        hc = element[i].HCA ;
        Ag = element[i].AGA ;
        Sx = element[i].SXA ;
        Ix = element[i].IXA ;                        //    (+)          (+)
        P = FORCE[0] ;                               //  P --> ------- <-- P
        V = FORCE[1] ;
        M = FORCE[2] ;
        code_check(i) ;                              // Check A-End

        /** -- Process b-end -- **/
        fprintf(out,"\n %3db",i+1) ;
        hc = element[i].HCB ;
        Ag = element[i].AGB ;
        Sx = element[i].SXB ;
        Ix = element[i].IXB ;                        //    (+)          (+)
        P = -FORCE[3] ;                              //  P --> ------- <-- P
        V = FORCE[4] ;
        M = FORCE[5] ;
        code_check(i) ;                              // Check B-End
      }
      fprintf(out, "\n" ) ;
    }

    /* -- Save Summary File -- */
    fprintf(max_ratios,"\n\t Summary of 9th Edition ASD Specification Checking\n") ;
    fprintf(max_ratios,"\n\t Table of Maximum Design Ratio Locations\n") ;
    fprintf(max_ratios,"\n\t Member  Stability  Stability  Capacity  Capacity  Shear
Shear") ;
    fprintf(max_ratios,"\n\t   No.     Case#      Ratio      Case#     Ratio   Case#
Ratio") ;
    fprintf(max_ratios,"\n\t ------  ---------  ---------  --------  --------  -----
-----") ;
    for ( i = 0 ; i < nmem ; i++ )
      fprintf(max_ratios,"\n\t %6d  %9d  %9.4lf  %8d  %8.4lf  %5d  %5.4lf", (i+1),
(Stab_Case[i]+1),        SR[i],(Cap_Case[i]+1), CR[i],(Shear_Case[i]+1),VR[i] ) ;

    /* -- Close i/o Files -- */
    fcloseall() ;

  } /*** end status check ***/

  /* Completed */
  printf("\n") ;
  return( status ) ;

} /*** end main() ***/

void read_input(int i)
{
int index ;

  /* -- Read Member Reactions -- */
  for ( index = 0 ; index < 6 ; index++ )
    fscanf(in, "%lf", &FORCE[index] ) ;

  /* -- Assign Element Properties -- */
  bf  = element[i].BF ; tf = element[i].TF ; tw = element[i].TW ;
  Iy = element[i].IY ;

  /* -- Rotational Moments for Cb Computation -- */
  M1 = ( fabs(FORCE[2]) <= fabs(FORCE[5]) ) ? FORCE[2] : FORCE[5] ;
  M2 = ( fabs(FORCE[2]) >= fabs(FORCE[5]) ) ? FORCE[2] : FORCE[5] ;

  /* Return to Main() */
```

```
  return ;

} /*** end read_input ***/



/* Code Check Subprogram */
void code_check(int i)
{
double Pa, Va, Ma ;

  Pa = axial_capacity(i) ;
  Va = shear_capacity() ;
  Ma = moment_capacity(i) ;

  fprintf(out, " %+7.2lf %+7.2lf %+9.2lf %+7.2lf %7.2lf %9.2lf", P, V, M/12.0, Pa, Va,
Ma/12.0 ) ;
  combined_stress(Pa,Va,Ma,i) ;

  /* Return to Main() */
  return ;

}

double axial_capacity(int i)
{
double Pa, Qs, Qa, f, f_new, beff, C_c, Ca, inc ;
int convergence ;

  /* Initialize Parameters */
  rx = sqrt(Ix/Ag) ;
  ry = sqrt(Iy/Ag) ;
  KL_rx = KLx[i]/rx ;                          // In-Plane Slenderness
  KL_ry = KLy[i]/ry ;                          // Out-Of-Plane Slenderness
  KLmax = max( KL_rx, KL_ry ) ;                // Maximum Slenderness Ratio
  Pa = 0.0 ;                                   // Allowable Axial Load
  Q = 1.0 ;                                    // Section Efficiency
  convergence = FALSE ;                        // Used with Slender Sections
  inc = ( load_case <= 1 ) ? 1.0 : 0.75 ;      // 1/3 Stress Increase

  /* Local Buckling Parameters */
  lambda_FLB = bf/(2.0*tf) ;                   // Flange Slenderness
  lambda_WLB = hc/tw ;                         // Web Slenderness

  /* -- 9th Edition Update -- */
  /** Reflects Research by Butler Manufacturing Co.  **/
  /** on Local Flange/Web Buckling interaction. "kc" **/
  /** is a plate buckling coefficient for h/t > 70   **/
  kc = ( (hc/tw) > 70.0 ) ? 4.05/pow(lambda_WLB, 0.46) : 1.0 ;

  /* Tension or Compression? */
  /** Analysis Passes Axial forces as +C and -T **/
  if ( P <= 0.0 )
  {
    /* Allowable Tensile Stress: Gross Section */
    Fa = -0.60*Fy ;
    Pa = Ag*Fa ;
  } /*** end tensile allowable force ***/

  else
  {
    /* Compression Capacity */
    C_c = sqrt(2.0*PI*PI*E/Fy) ;

    /* Local Buckling Limits, Table B5.1 */
    lr_FLB = 95.0/sqrt(Fy/kc) ;                // Flange Noncompact Limit
    lr_WLB = 253.0/sqrt(Fy) ;                  // Web Noncompact Limit
```

```
if ( lambda_FLB <= lr_FLB && lambda_WLB <= lr_WLB )
/* Section Composed of (Non)Compact Element(s) */
/** Chapter E **/
{
  if ( KLmax <= C_c )
  {
    Ca = KLmax/C_c ;
    FS = 5.0/3.0 + 0.375*Ca - 0.125*Ca*Ca*Ca ;
    /* Allowable Compressive Stress: Compact, Inelastic */
    Fa = ( 1.0 - 0.5*Ca*Ca )*Fy / FS ;
    Pa = Ag*Fa ;
  }
  else if ( KLmax > C_c )
  {
    FS = 23.0/12.0 ;
    /* Allowable Compressive Stress: Compact, Elastic */
    Fa = PI*PI*E/(KLmax*KLmax) / FS ;
    Pa = Ag*Fa ;
  }
}

else
/* Section Composed of Slender Element(s) */
/** Section Composed of Slender Element(s) **/
{
  if ( lambda_FLB > lr_FLB )              // Unstiffened Elements
  {
    /* Flanges Slender */
    if ( lambda_FLB < (195.0/sqrt(Fy/kc)) )
      Qs = 1.293 - 0.00309*lambda_FLB*sqrt(Fy/kc) ;
    else
      Qs = 26200.0*kc/(Fy*lambda_FLB*lambda_FLB) ;
  }
  else
  {
    Qs = 1.0 ;
  }
  Qs = min ( Qs, 1.0 ) ;                   // Error Check


ITERATE:

  if (lambda_WLB > lr_WLB)                 // Stiffened Elements
  {
    /* Web Slender */
    Pa = ( Pa ) ? Pa : 0.60*Fy*Ag ;
    f = sqrt(inc*Pa/Ag) ;
    if ( f )                               // Effective Width of Web?
    {
      beff = (253.0*tw/f)*(1.0-44.3/(lambda_WLB*f)) ;
      if ( beff >= hc ) beff = hc ;
      if ( (beff/tw) < lr_WLB ) beff = min( (lr_WLB*tw), hc ) ;
      Qa = (Ag-(hc-beff)*tw)/Ag ;
    }
    else                                   // Reduction Insignificant
    {
      Qa = 1.0 ;
    }
  }
  else                                     // (Non)compact
  {
    Qa = 1.0 ;
  }

  /* -- Q = Efficiency Factor, or Reduced Section Factor -- */
  /** -- Theoretically, the ratio of Fcr/Fy -- **/
  Q = Qs*Qa ;
  C_c = sqrt(2.0*PI*PI*E/Fy/Q) ;           // Elastic Limit ( C'c )
```

```
      if (KLmax < C_c)                          // Inelastic Cutoff at C'c
      {
        Ca = KLmax/C_c ;
        FS = 5.0/3.0 + 0.375*Ca - 0.125*Ca*Ca*Ca ;
        /* Allowable Compressive Stress: Slender, Inelastic */
        Fa = Q*( 1.0 - 0.5*Ca*Ca )*Fy / FS ;
        Pa = Ag*Fa ;
      }
      else
      {
        FS = 23.0/12.0 ;
        /* Allowable Compressive Stress: Slender, Elastic */
        Fa = PI*PI*E/(KLmax*KLmax) / FS ;
        Pa = Ag*Fa ;
      }

      /* Check for convergence of slender compression allowable */
      f_new = sqrt(inc*Pa/Ag) ;
      convergence = ( fabs(f_new-f) <= 0.10 ) ? TRUE : FALSE ;
      if ( !convergence ) goto ITERATE ;
    }

  } /*** end compressive allowable force ***/

  /* Return to Code_check() */
  return(Pa) ;

} /*** end axial_capacity ***/

double shear_capacity()
{
double Va, kv = 5.34, Cv, Aw ;

  /* Parameters */
  Aw = (2.0*tf+hc)*tw ;

  if ( lambda_WLB <= 380.0/sqrt(Fy) )
    Fv = 0.40*Fy ;
  else
  {
    Cv = 45000.0*kv/(Fy*lambda_WLB*lambda_WLB) ;
    Cv = ( Cv >= 0.8 )  ? Cv : (190.0/lambda_WLB)*sqrt(kv/Fy) ;
    Fv = (Fy/2.89)*Cv ;
    Fv = ( Fv <= 0.40*Fy ) ? Fv : 0.40*Fy ;
  }

  /* Allowable Shear Force */
  Va = Aw*Fv ;

  /* Return to Code_check() */
  return(Va) ;

}

double moment_capacity(int i)
{
double Ma, Cb, Io, AT, rT, Lb, Lc, Lu, Lr, Fb1, Fb2, Fb_old, Qs, Rpg ;
int convergence ;

  /* Initialize Parameters */
  /** Cb is usually calculated from the moments b/w supports. **/
  /** Here, it is calculated at the ends b/w each element end **/
  if ( fabs(M2) < 1.0e-3) Cb = 1.75 ;
  else
  {
    Cb = 1.75+1.05*(M1/M2)+0.3*(M1/M2)*(M1/M2) ;
    Cb = (Cb <= 2.3 ? Cb : 2.3) ;
  }
```

```
Io = tf*bf*bf*bf/12.0 ;
AT = bf*tf+hc*tw/6.0 ;
rT = sqrt(Io/AT) ;
Lb = KLy[i] ;                                  // Unbraced Length = KLy


/* Local Buckling Limits of Beams & Beam-Columns */
/** -- Flanges -- **/
lp_FLB = 65.0/sqrt(Fy) ;                       // Flange Compact Limit
lr_FLB = 95.0/sqrt(Fy/kc) ;                    // Flange Noncompact Limit
/** -- Webs -- **/
if ( P > 0.0 )
{
  lp_WLB = (P/Ag/Fy < 0.16) ?                  // Web Compact Limit
           640.0/sqrt(Fy)*(1.0-3.74*P/Ag/Fy) : 257.0/sqrt(Fy) ;
}
else
{
  lp_WLB = 640.0/sqrt(Fy) ;                    // Web Compact Limit
}
lr_WLB = 970.0/sqrt(Fy) ;                      // Web Noncompact Limit

/* Vertical Buckling Check */
if ( lambda_WLB > (14000.0/sqrt(Fy*(Fy+16.5))) || lambda_WLB > 260.0 )
  fprintf(out,"\n Bend Buckling...Ma incorrect") ;

/* Slender Flange Check: Qs */
if ( lambda_FLB > lr_FLB )
{
  /* Flanges Slender */
  if ( lambda_FLB < (195.0/sqrt(Fy/kc)) )
    Qs = 1.293 - 0.00309*lambda_FLB*sqrt(Fy/kc) ;
  else
    Qs = 26200.0*kc / (Fy*lambda_FLB*lambda_FLB) ;
}
else
{
  /* Flanges (Non)Compact */
  Qs = 1.0 ;
}
Qs = min ( Qs, 1.0 ) ;                         // Error Check

/* Compact, Inelastic Length Limit */
Lc = min( 20000.0/((hc+2.0*tf)/(bf*tf))/Fy, 76.0*bf/sqrt(Fy) ) ;

/* Inelastic LTB Limit */
Lu = rT*sqrt(102000.0*Cb/Fy) ;

/* Elastic LTB Limit */
Lr = rT*sqrt(510000.0*Cb/Fy) ;

/* Evaluate Moment Capacity Based on Section Properties */
/* -- Chapter F, Section 1 -- */
if ( (lambda_FLB <= lp_FLB) && (lambda_WLB <= lp_WLB) && (Lb <= Lu) )
{
  if ( Lb <= Lc )
  {
    /* Fully Compact, Plastic */
    Fb = 0.66*Fy ;
    Ma = Sx*Fb ;
  }
  else /* Lb <= Lu */
  {
    /* Partially Compact, Inelastic */
    Fb = 0.60*Fy ;
    Ma = Sx*Fb ;
  }
```

```
    return( Ma ) ;
  }

  /* -- Chapter F, Section 2 -- */
  if ( (lambda_FLB <= lr_FLB) && (lambda_WLB <= lr_WLB) && (Lb <= Lc) )
  {
    /* Flanges Partially Compact, Web (Non)Compact, Lb Compact */
    Fb = Fy*(0.79-0.002*lambda_FLB*sqrt(Fy/kc)) ;
    Ma = Sx*Fb ;
    return(Ma) ;
  }

  /**** Noncompact, Slender or Plate Girder Provisions ****/
  Fb_old = Fb = 0.60*Fy ;                          // Assume Fb Value
  convergence = TRUE ;                             // Initialize Value TRUE
  lr_WLB = 760.0/sqrt(Fb) ;

ITERATE:

  /* -- Chapter F, Section 3 -- */
  if ( (lambda_FLB <= lr_FLB) && (lambda_WLB <= lr_WLB) )
  {
    if ( Lb <= Lu )
      /* Flanges & Web Noncompact, Lb Compact */
      Fb = 0.60*Fy ;
    else if ( (Lb > Lu) && (Lb <= Lr) )
    {
      /* Flanges & Web Noncompact, Lb Noncompact */
      double Fb2a, Fb2b ;
      Fb2a = ( 2.0/3.0 - Fy*pow((Lb/rT),2.0)/(1530000.0*Cb) )*Fy ;
      Fb2b = 12000.0*Cb/(Lb*(hc+2.0*tf)/(bf*tf)) ;
      Fb = max( Fb2a, Fb2b ) ;                     // Select Maximum
    }
    else /* ( Lb > Lr ) */
    {
      /* Flanges & Web Noncompact, Lb Elastic */
      double Fb2a, Fb2b ;
      Fb2a = 170000.0*Cb/pow((Lb/rT),2.0) ;
      Fb2b = 12000.0*Cb/(Lb*(hc+2.0*tf)/(bf*tf)) ;
      Fb = max( Fb2a , Fb2b ) ;                    // Select Maximum
    }
    /* -- Select Minimum -- */
    Fb = ( Fb <= 0.60*Fy ) ? Fb : 0.60*Fy ;
    lr_WLB = 760.0/sqrt(Fb) ;
  }

  /* Chapter G */
  if ( (lambda_FLB > lr_FLB) || (lambda_WLB > lr_WLB) )
  {
    /* Plate Girder BASIC Allowable Bending */
    /* Stress From Chapter F, Section 3 */
    if ( Lb <= Lu )
    {
      Fb = 0.60*Fy ;
    }
    else if ( Lb <= Lr )
    {
      Fb1 = ( 2.0/3.0 - Fy*pow((Lb/rT),2.0)/(1530000.0*Cb) )*Fy ;
      Fb2 = 12000.0*Cb/(Lb*(hc+2.0*tf)/(bf*tf)) ;
      Fb = max( Fb1, Fb2 ) ;                       /* Select Maximum */
    }
    else /* ( Lb > Lr ) */
    {
      Fb1 = 170000.0*Cb/pow((Lb/rT),2.0) ;
      Fb2 = 12000.0*Cb/(Lb*(hc+2.0*tf)/(bf*tf)) ;
      Fb = max( Fb1 , Fb2 ) ;
    }
```

```
    /* -- Reduced Plate Girder Bending Stress -- */
    Fb = ( Fb <= 0.60*Fy ) ? Fb : 0.60*Fy ;
    lr_WLB = 760.0/sqrt(Fb) ;
    Rpg = 1.0 - 0.0005*(hc*tw)/(bf*tf)*(lambda_WLB - lr_WLB) ;
    Rpg = ( Rpg < 1.0 ) ? Rpg : 1.0 ;
    Fb *= Rpg ;                                /* Chapter G */
  }

  /* Check for Convergence */
  convergence = ( fabs(Fb_old-Fb) <= 0.5 ) ? TRUE : FALSE ;
  if ( convergence )
  {
    /* Compare with Appendix B, if Applicable */
    if ( lambda_FLB > lr_FLB ) Fb = min( Fb, 0.60*Fy*Qs ) ;
    Ma = Sx*Fb ;
    return( Ma ) ;
  }
  else
  {
    Fb_old = Fb ;
    goto ITERATE ;
  }

}

void combined_stress( double Pa, double Va, double Ma, int i )
{
double a_r, m_r, Cm, fa, Fe, AmpFact, stability, capacity, shear ;

  /* P-Delta Amplification (First Order ONLY!) */
  if ( (P > 0.0) && fo_flag )
  {
      Cm = 0.85 ;
      fa = P/Ag ;                                      // Axial Stress
      Fe = 149331.0 / KL_rx / KL_rx ;                  // Euler Stress
      Fe = ( Fe > fa ) ? Fe : 1.1*fa ;                 // Error Trap
      AmpFact = Cm / ( 1.0 - fa / Fe ) ;
  }
  else          /* Tension and/or Second Order Chosen */
    AmpFact = 1.0 ;

  /* ASD Eqn. H1-1: Stability Interaction Equation */
  if ( P > 0.0 )
  {
    a_r = P/Pa ;                               // Axial Term
    m_r = AmpFact*fabs(M/Sx/Fb) ;              // Moment Term
    stability = a_r + m_r ;                    // Stability Ratio
  }
  else
  {
    stability = 0.0 ;                          // Tension or Pure Bending
  }
  fprintf(out," %9.3lf", stability ) ;

  /* ASD Eqn. H1-2: Capacity Equation */
  /** Q Parameter now recognized for Combined Stresses **/
  /** when Slender elements are part of the X-Section  **/
  a_r = fabs(P/Ag/0.6/Fy/Q) ;                  // Axial Term
  m_r /= AmpFact ;                             // Moment Term
  capacity = a_r + m_r ;                       // Capacity Ratio
  fprintf(out," %9.3lf", capacity ) ;

  /* ASD Chapter F: Shear */
  shear = fabs(V/Va) ;                         // Shear Ratio

  /* Check for Maximums */
  critical_ratios( i, stability, capacity, shear ) ;
```

```
  /* Return to code_check() */
  return ;

}

void critical_ratios( int mem, double stability, double capacity, double shear )
{
  if ( stability > SR[mem] )                      // Extract Stability Ratio
  {
    SR[mem] = stability ;
    Stab_Case[mem] = load_case ;
  }
  if ( capacity > CR[mem] )                       // Extract Capacity Ratio
  {
    CR[mem] = capacity ;
    Cap_Case[mem] = load_case ;
  }
  if ( shear > VR[mem] )                          // Extract Shear Ratio
  {
    VR[mem] = shear ;
    Shear_Case[mem] = load_case ;
  }

  /* Next... */
  return ;

} /*** end critical_ratios() ***/

int member_properties( ELEMENT *element )
{
  /* -- Read: BF, TF, TW -- */
  fscanf( in, "%lf %lf %lf", &element->BF, &element->TF, &element->TW ) ;

  /* -- Read: HCA, AGA, IXA, SXA -- */
  fscanf( in, "%lf %lf %lf %lf", &element->HCA, &element->AGA,
                                 &element->IXA, &element->SXA ) ;

  /* -- Read: HCB, AGB, IXB, SXB -- */
  fscanf( in, "%lf %lf %lf %lf", &element->HCB, &element->AGB,
                                 &element->IXB, &element->SXB ) ;

  /* -- Read: Iy -- */
  fscanf( in, "%lf", &element->IY ) ;

  /* Return to Main() */
  return ( NO_ERROR ) ;

} /*** end member_properties() ***/

void pause(void)
{
  printf("\n\n Strike a key when ready...") ;
  while ( !kbhit() ) ;
  getch() ;
  return ;
}
/*****************************************************************************/
/* End of ASDSPEC Program                                                    */
/*****************************************************************************/
```

LRFDSPEC.c

```
/******************************************************************************/
/*
* LRFDSPEC.h     Header File for LRFDSPEC Program
*/
/********************************** CONSTANTS *********************************/
#define E       29000.0
#define G       11154.0
#define PI      3.141592653589793
#define TRUE    1
#define FALSE   0
/********************************** MACROS ************************************/
#define forget(x)                   { if(x) free((void *)x) ; x = NULL ; }
/********************************** STRUCTURES ********************************/
typedef struct ELEMENT
{
  double BF,                        /* Flange Width                        */
         TF,                        /* Flange Thickness                    */
         TW,                        /* Web Thickness                       */
         HCA,                       /* Web Height, A-End                   */
         AGA,                       /* Area of Element A-End               */
         IXA,                       /* X-X Moment of Inertia A-End         */
         SXA,                       /* Elastic Section Modulus A-End       */
         ZXA,                       /* Plastic Section Modulus A-End       */
         HCB,                       /* Web Height, B-End                   */
         AGB,                       /* Area of Element B-End               */
         IXB,                       /* X-X Moment of Inertia B-End         */
         SXB,                       /* Elastic Section Modulus B-End       */
         ZXB,                       /* Plastic Section Modulus B-End       */
         IY;                        /* Y-Y Moment of Inertia               */
} ELEMENT ;
/********************************** FUNCTIONS *********************************/
extern int main(void);
extern void read_input(int i);
extern void code_check(int i);
extern double axial_capacity(int i);
extern double shear_capacity(void);
extern double moment_capacity(int i);
extern void combined_stress(double oPn,double oVn,double oMn,int i);
extern void critical_ratios(int i,double capacity,double shear);
extern int member_properties(ELEMENT *element);
void pause(void);
/****************************** Character Strings *****************************/
/*
*  Load Combination Titles for LRFD Analysis\Design Output
*/
static char *str[] = { "1.2D + 1.6SL + 1.6SR" ,
                       "1.2D + 1.6SL + 0.5SR" ,
                       "1.2D + 1.3WLL1 + 1.3WLR1" ,
                       "1.2D + 1.3WLL2 + 1.3WLR2" ,
                       "1.2D + 1.6SL + 1.6SR + 0.8WLL1 + 0.8WLR1" ,
                       "1.2D + 1.6SL + 1.6SR + 0.8WLL2 + 0.8WLR2" ,
                       "1.2D + 1.6SL + 0.5SR + 0.8WLL1 + 0.8WLR1" ,
                       "1.2D + 1.6SL + 0.5SR + 0.8WLL2 + 0.8WLR2" ,
                       "1.2D + 1.6SL + 0.5SR + 0.8WRL1 + 0.8WRR1" ,
                       "1.2D + 1.6SL + 0.5SR + 0.8WRL2 + 0.8WRR2" ,
                       "1.2D + 1.3WLL1 + 1.3WLR1 + 0.5SL + 0.5SR" ,
                       "1.2D + 1.3WLL2 + 1.3WLR2 + 0.5SL + 0.5SR" ,
                       "1.2D + 1.3WLL1 + 1.3WLR1 + 0.5SL + 0.25SR" ,
                       "1.2D + 1.3WLL2 + 1.3WLR2 + 0.5SL + 0.25SR" ,
                       "1.2D + 1.3WRL1 + 1.3WRR1 + 0.5SL + 0.25SR" ,
                       "1.2D + 1.3WRL2 + 1.3WRR2 + 0.5SL + 0.25SR" } ;
```

```
/*****************************************************************************/
/*                                                                           */
/*  LRFDSPEC.c                                                               */
/*                                                                           */
/*  Copyright 1990, PC Structural Engineers                                  */
/*                  Blacksburg, Virginia 24060                               */
/*                                                                           */
/*  Research Team:                                                           */
/*  Eric J. Wishart......................Research Investigator               */
/*  Thomas M. Murray, PhD................Principal Investigator              */
/*                            Of The                                         */
/*      Charles E. Via, Jr. Department of Civil Engineering                  */
/*       Virginia Polytechnic Institute & State University                   */
/*                  Blacksburg, Virginia 24061                               */
/*                                                                           */
/*  Sponsor: NUCOR Metal Building Products                                   */
/*           Waterloo, Indiana                                               */
/*                                                                           */
/*  Performs Code Checking of Beams, Columns, and                            */
/*  Beam-Columns according to the AISC 1st Edition LRFD Specification.       */
/*  Second Order Analysis Results REQUIRED.                                  */
/*                                                                           */
/*  Only error recognized at this point, is the calculation of Cb.  By       */
/*  definition, it is calculated using the moments at the ends of the        */
/*  unbraced length (unless larger in-between).  This program assumes        */
/*  the "end" moments are located at each discretized element "end".        */
/*  The error in the calculation will be a conservative one.                 */
/*                                                                           */
/*  Inputfile:        LRFD.tmp                                               */
/*  Outputfile:       LRFD.chk                                               */
/*                                                                           */
/*****************************************************************************/
/* INCLUDE Files */
#include <stdio.h>                              /* Microsoft Library */
#include <math.h>
#include <fcntl.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#include <conio.h>
#include "LRFDSPEC.h"                           /* LRFDSPEC header file */

/* DEFINE constants */
#define ERROR    0
#define NO_ERROR 1

/* Global Variables */
double Pu,Vu,Mu,M1,M2,Zx,Sx,bf,tf,hc,tw,Ix,Iy,Ag,KLx[200],KLy[200],Fy,Fr,
       rx,ry,KL_rx,KL_ry,KLmax,lambda_FLB,lambda_WLB,lambda_LTB,lambda_C,
       lp_FLB,lp_WLB,lr_FLB,lr_WLB, *CR, *VR, FORCE[6] ;
int process,nmem,i,load_case, *Cap_Case, *Shear_Case ;
ELEMENT *element ;
FILE *in, *in2, *out, *max_ratios ;

/* Main Program */
main()
{
int status = NO_ERROR ;

  /* Open Output File for writing */
  out = fopen("LRFD.chk","w") ;
  max_ratios = fopen("LRFD.sum","w") ;
  if ( out == NULL || max_ratios == NULL )
  {
    printf("\n CANNOT OPEN FILES, LRFD.chk or LRFD.sum") ;
    status = ERROR ;
    pause() ;
  }
```

```
  /* Open Input Files for reading */
  in = fopen("LRFD.tmp","rb") ;
  in2 = fopen("KL.fct","r") ;
  if ( in == NULL || in2 == NULL )
  {
    printf("\n Cannot open Input Files, LRFD.tmp & KL.tmp") ;
    status = ERROR ;
    pause() ;
  }

  if ( status )
  {
    /* Print Progress Report */
    printf("\n LRFD 1st Edition Specification Check In Progress...\n\n") ;
    fscanf(in,"%d %d %lf", &nmem, &process, &Fy);

    /* Dynamic Memory Allocation */
    element    =  (ELEMENT *) calloc( nmem, sizeof(ELEMENT) );
    CR         =  (double *)  calloc( nmem,  sizeof(double) ) ;
    VR         =  (double *)  calloc( nmem,  sizeof(double) ) ;
    Cap_Case   =  (int *)     calloc( nmem,     sizeof(int) ) ;
    Shear_Case =  (int *)     calloc( nmem,     sizeof(int) ) ;

    /* -- Check Dynamic Memory Allocations -- */
    if ( (element == (ELEMENT *) NULL)  )
      status = ERROR ;
    if ( (CR == NULL) || (VR == NULL) )
      status = ERROR ;
    if ( (Cap_Case == NULL) || (Shear_Case == NULL) )
      status = ERROR ;
    if ( !status )
    {
      fcloseall() ;
      return( ERROR ) ;
    }

    /* Assume Magnitudes of Residual Stresses */
    /* Process = 0: Rolled, Process = 1: Welded */
    Fr = ( process ) ? 16.5 : 10.0 ;

    /* Read in Member Properties & Bracing Factors */
    for ( i = 0 ; i < nmem ; i++ )
    {
      status = member_properties( &element[i] ) ;
      fscanf(in2, "%lf %lf", &KLx[i], &KLy[i] ) ;
    }
    fclose( in2 ) ;

    /* Process All Load Combinations */
    for ( load_case = 0 ; load_case < 16 ; load_case++ )
    {
      /* -- Report Progress -- */
      printf("\r LRFD Load Combination No. %d", load_case+1 ) ;

      /* -- Title to OutputFile -- */
      fprintf(out, "\n LRFD Load Combination No. %d", load_case+1 ) ;
      fprintf(out, "\n ===========================\n\n" ) ;
      fprintf(out,"\t%s\n", str[load_case] ) ;
      fprintf(out,"\n Mem     Pu        Vu        Mu        oPn       oVn       oMn
Capacity   Shear" );
      fprintf(out,"\n  #      [k]       [k]       [kft]     [k]       [k]       [kft]
[Unity]    [Unity]");
      fprintf(out,"\n ----  -------  -------  ---------  -------  -------  ---------
--------- --------");

      /* -- Process All Members -- */
      for ( i = 0 ; i < nmem ; i++ )
      {
```

```
        read_input(i) ;

        /** -- Process a-end -- **/
        fprintf(out,"\n %3da",i+1) ;
        hc = element[i].HCA ;
        Ag = element[i].AGA ;
        Sx = element[i].SXA ;
        Zx = element[i].ZXA ;
        Ix = element[i].IXA ;                       //    (+)          (+)
        Pu = FORCE[0] ;                             //  P --> ------- <-- P
        Vu = FORCE[1] ;
        Mu = FORCE[2] ;
        code_check(i) ;                             // Check A-End

        /** -- Process b-end -- **/
        fprintf(out,"\n %3db",i+1) ;
        hc = element[i].HCB ;
        Ag = element[i].AGB ;
        Sx = element[i].SXB ;
        Zx = element[i].ZXB ;
        Ix = element[i].IXB ;                       //    (+)          (+)
        Pu = -FORCE[3] ;                            //  P --> ------á <-- P
        Vu = FORCE[4] ;
        Mu = FORCE[5] ;
        code_check(i) ;                             // Check B-End
      }
      fprintf(out, "\n" ) ;
    }

    /* -- Save Summary File -- */
    fprintf(max_ratios,"\n\t Summary of 1st Edition LRFD Specification Checking\n")
;
    fprintf(max_ratios,"\n\t Table of Maximum Design Ratio Locations\n") ;
    fprintf(max_ratios,"\n\t Member  Capacity  Capacity  Shear  Shear") ;
    fprintf(max_ratios,"\n\t   No.    Case#     Ratio    Case#  Ratio") ;
    fprintf(max_ratios,"\n\t ------  --------  --------  -----  -----") ;
    for ( i = 0 ; i < nmem ; i++ )
      fprintf(max_ratios,"\n\t %6d  %8d  %8.4lf  %5d  %5.4lf", (i+1), (Cap_Case[i]+1),
        CR[i],(Shear_Case[i]+1),VR[i] ) ;

    /* -- Close i/o Files -- */
    fcloseall() ;

  } /*** end status check ***/

  /* Completed */
  printf("\n") ;
  return( status ) ;

} /*** end main() ***/

void read_input(int i)
{
int index ;

  /* Read Member Reactions */
  for ( index = 0 ; index < 6 ; index++ )
    fscanf(in, "%lf", &FORCE[index] ) ;

  /* Extract Element Properties */
  bf  = element[i].BF ; tf = element[i].TF ; tw = element[i].TW ;
  Iy = element[i].IY ;

  /* -- Rotational Moments for Cb Computation -- */
  M1 = ( fabs(FORCE[2]) <= fabs(FORCE[5]) ) ? FORCE[2] : FORCE[5] ;
  M2 = ( fabs(FORCE[2]) >= fabs(FORCE[5]) ) ? FORCE[2] : FORCE[5] ;

  /* Return to Main() */
```

```
  return ;

} /*** end read_input ***/

/* Code Check Subprogram */
void code_check(int i)
{
double oPn, oVn, oMn ;

  oPn = axial_capacity(i) ;
  oVn = shear_capacity() ;
  oMn = moment_capacity(i) ;
  fprintf(out," %+7.2lf %+7.2lf %+9.2lf %+7.2lf %7.2lf %10.2lf", Pu,Vu,Mu/12.0,
oPn,oVn,oMn/12.0 );
  combined_stress(oPn,oVn,oMn,i) ;

  /* Return to Code_check */
  return ;

} /*** end code_check() ***/

double axial_capacity(int i)
{
double oPn, Qs, Qa, Q, f, f_new, beff ;
int convergence ;

  /* Initialize Parameters */
  rx = sqrt(Ix/Ag) ;
  ry = sqrt(Iy/Ag) ;
  KL_rx = KLx[i]/rx  ;                        // In-Plane Slenderness
  KL_ry = KLy[i]/ry ;                         // Out-of-Plane Slenderness
  KLmax = max( KL_rx , KL_ry ) ;              // Maximum Slenderness
  oPn = 0.0 ;                                 // Axial Capacity
  convergence = 0 ;                           // Used for Slender Elements

  /* Slenderness Ratios for Columns */
  lambda_FLB = bf/(2.0*tf) ;                  // Flange Slenderness
  lambda_WLB = hc/tw ;                        // Web Slenderness
  lambda_C = (KLmax/PI)*sqrt(Fy/E) ;          // Column Slenderness

  /* Tension or Compression? */
  /** Program stores axial forces as +C, -T **/
  if (Pu <= 0.0)
  {
    /* Tension Capacity: Gross Section; LRFD Ref. D1-1 */
    oPn = -0.90*Ag*Fy ;
  }  /*** end tension nominal resistance ***/

  else
  {
    /* Compressive Capacity */
    lr_FLB = 95.0/sqrt(Fy) ;                  // Flange Noncompact Limit
    lr_WLB   = 253.0/sqrt(Fy) ;               // Web Noncompact Limit

    /* -- Chapter E -- */
    if (lambda_FLB <= lr_FLB && lambda_WLB <= lr_WLB)
    {
      if (lambda_C <= 1.5)
      {
        /* Compact, Inelastic; LRFD Ref. E2-1 & E2-2 */
        oPn = 0.85*(pow(0.658,lambda_C*lambda_C))*Ag*Fy ;
      }
      if (lambda_C > 1.5)
      {
        /* Compact, Elastic; LRFD Ref. E2-1 & E2-3 */
        oPn = 0.85*(0.877/(lambda_C*lambda_C))*Ag*Fy ;
      }
    }
```

```
    else
    /** Section Composed of Slender Element(s) **/
    {
      if (lambda_FLB > lr_FLB)                    // Unstiffened Elements
      {
        /* Flanges Slender */
        if ( lambda_FLB <= (176.0/sqrt(Fy)) )
          Qs = 1.415 - 0.00437*lambda_FLB*sqrt(Fy) ;
        else
          Qs = 20000.0/(Fy*lambda_FLB*lambda_FLB) ;
      }
      else
      {
        Qs = 1.0 ;
      }
      Qs = min ( Qs, 1.0 ) ;                      // Error Check

ITERATE:

      if (lambda_WLB > lr_WLB)                    // Stiffened Elements
      {
        /* Web Slender */
        oPn = ( oPn ) ? oPn : 0.85*Fy*Ag ;
        f = sqrt(oPn/Ag) ;
        if ( f )                                  // Effective Width of Web?
        {
          beff = (326.0*tw/f)*(1.0-57.2/(lambda_WLB*f)) ;
          if ( beff >= hc ) beff = hc ;
          if ( (beff/tw) < lr_WLB ) beff = min( (lr_WLB*tw), hc ) ;
          Qa = (Ag-(hc-beff)*tw)/Ag ;
        }
        else                                       // Reduction Insignificant
        {
          Qa = 1.0 ;
        }
      }
      else                                         // (Non)compact
      {
        Qa = 1.0 ;
      }

      /* -- Q = Efficiency Factor, or Reduced Section Factor -- */
      /** -- Theoretically, the ratio of Fcr/Fy -- **/
      Q = Qs*Qa ;

      if (lambda_C*sqrt(Q) < 1.5)                 // Inelastic Cutoff @1.5
      {
        /* Inelastic; LRFD Ref. A-B5-11 */
        oPn = 0.85*Q*pow(0.658,Q*lambda_C*lambda_C)*Ag*Fy ;
      }
      else
      {
        /* Elastic; LRFD Ref. A-B5-13 */
        oPn = 0.85*(0.877/(lambda_C*lambda_C))*Ag*Fy ;
      }

      /* Check for convergence of slender compression resistance */
      f_new = sqrt(oPn/Ag) ;
      convergence = ( fabs(f_new-f) <= 0.10 ) ? TRUE : FALSE ;
      if ( !convergence ) goto ITERATE ;
    }

  } /*** end compression nominal resistance ****/

  /* Return to Code_check */
  return(oPn) ;

} /*** end axial_capacity ***/
```

```
double shear_capacity()
{
double oVn, Aw , sq_rt , web_ratio ;

  /* Parameters */
  Aw = (hc+2.0*tf)*tw ;
  sq_rt = sqrt(5.0/Fy) ;
  web_ratio = hc/tw ;

  if ( web_ratio <= 187.0*sq_rt )
    oVn = 0.9*0.6*Fy*Aw ;
  else if ( web_ratio <= 234.0*sq_rt )
    oVn = 0.9*0.6*Fy*Aw*187*sq_rt/(hc/tw) ;
  else
    oVn = 0.9*0.6*Aw*44000.0*5.0*tw*tw/hc/hc ;

  /* Return to Code_check */
  return(oVn) ;

} /*** end shear_capacity ***/

double moment_capacity(int i)
{
double X1,X2,Cw,J,Cb,Lr,Mp,Mr,Mcr,Mn,oPy,Lp,Lb ;
double oMn,oMn1,oMn2,oMn3,oMn_LTB,oMn_FLB,Cpg,Rpg,Fcr,Sxc,rT,Re,Qs ;
double lambda, lambda_p, lambda_r ;

  /* Initialize Parameters */
  Mp = Fy*Zx ;
  Mr = (Fy-Fr)*Sx ;
  Cw = (1.0/24.0)*pow(bf,3.0)*tf*pow((hc+tf),2.0) ;
  J  = (1.0/3.0)*(hc*pow(tw,3.0)+2.0*bf*pow(tf,3.0)) ;
  /** Cb is usually calculated from the moments b/w supports. **/
  /** Here, it is calculated at the ends b/w each element end **/
  if ( fabs(M2) < 1.0e-3) Cb = 1.75 ;
  else
  {
    Cb = 1.75+1.05*(M1/M2)+0.3*(M1/M2)*(M1/M2) ;
    Cb = (Cb <= 2.3 ? Cb : 2.3) ;
  }
  X1 = (PI/Sx)*sqrt(E*G*J*Ag/2.0) ;
  X2 = (4.0*Cw/Iy)*pow((Sx/(G*J)),2.0) ;
  Lr = (ry*X1/(Fy-Fr))*sqrt(1.0+sqrt(1.0+X2*(Fy-Fr)*(Fy-Fr))) ;
  Lp = 300.0*ry/sqrt(Fy) ;
  Lb = KLy[i] ;                              // Unbraced Length = KLy
  lambda_LTB = Lb/ry ;

  /* Local Buckling Limits for Beams & Beam-Columns */
  if (Fr == 10.0)
  {
    /* Rolled Sections: Lambda_r */
    lr_FLB = 141.0/sqrt(Fy-Fr) ;
    lr_WLB = 970.0/sqrt(Fy) ;
  }
  else
  {
    /* Welded Sections: Lambda_r */
    lr_FLB = 106.0/sqrt(Fy-Fr) ;
    lr_WLB = 970.0/sqrt(Fy) ;
  }

  /* Welded or Rolled Sections: Lambda p */
  lp_FLB = 65.0/sqrt(Fy) ;
  if (Pu <= 0.0)
  {
    /* Tensile Axial Force */
    lp_WLB = 640.0/sqrt(Fy) ;
  }
```

```
else
{
  /* Compressive Axial Force */
  oPy = 0.9*Fy*Ag ;
  if (Pu/oPy <= 0.125)
    lp_WLB = (640.0/sqrt(Fy))*(1.0-2.75*Pu/oPy) ;
  else
  {
    lp_WLB = (191.0/sqrt(Fy))*(2.33-Pu/oPy) ;
    if (lp_WLB < 253.0/sqrt(Fy)) lp_WLB = 253.0/sqrt(Fy) ;
  }
}

/* Check for Slender Flanges under Appendix B5 */
if ( lambda_FLB > lr_FLB )
{
   /* Flanges Slender */
   if (lambda_FLB <= (176.0/sqrt(Fy)) )
     Qs = 1.415 - 0.00437*lambda_FLB*sqrt(Fy) ;
   else
     Qs = 20000.0/(Fy*lambda_FLB*lambda_FLB) ;
}
else
{
  /* Flanges (Non)Compact */
  Qs = 1.0 ;
}
Qs = min ( Qs, 1.0 ) ;                          // Error Check

/* Evaluate Moment Capacity based on Section Properties */
if ( (lambda_FLB <= lp_FLB) && (lambda_WLB <= lp_WLB) )
{
  /* Compact, Plastic or Inelastic Transition Zone */
  if (Lb <= Lr)
  {
    /* LRFD Ref. F1-3 */
    oMn = 0.90*Cb*(Mp-(Mp-Mr)*(Lb-Lp)/(Lr-Lp)) ;
    if (oMn > 0.90*Mp) oMn = 0.90*Mp ;
    return ( oMn ) ;
  }
  else /* Compact, Lb > Lr */
  {
    /* LRFD Ref. F1-12 */
    Mcr = Cb*(PI/Lb)*sqrt(E*Iy*G*J+pow((PI*E/Lb),2.0)*Iy*Cw) ;
    if (Mcr > Cb*Mr)
    {
      oMn = 0.90*Mcr ;
    }
    else
    {
      oMn = 0.90*Cb*Mr ;
    }
    return( oMn ) ;
  }
}

/* -- Appendix F -- */
if ( (lambda_FLB <= lr_FLB) && (lambda_WLB <= lr_WLB) )
{
  if (lambda_LTB <= Lp/ry )
  {
    /* Compact Unbraced Length */
    oMn1 = 0.90*Mp ;
  }
  else
  {
    /* Lateral Torsional Buckling */
    oMn1 = 0.90*Cb*(Mp-(Mp-Mr)*(lambda_LTB-Lp/ry)/(Lr/ry-Lp/ry)) ;
```

```
    oMn1 = min( oMn1 , 0.90*Mp ) ;
  }

  /* Flange Local Buckling */
  Mr = (Fy-Fr)*Sx ;
  oMn2 = 0.90*(Mp-(Mp-Mr)*(lambda_FLB-lp_FLB)/(lr_FLB-lp_FLB)) ;

  /* Web Local Buckling */
  Mr = Fy*Sx ;
  oMn3 = 0.90*(Mp-(Mp-Mr)*(lambda_WLB-lp_WLB)/(lr_WLB-lp_WLB)) ;

  /* Select lowest value of LTB, FLB, & WLB */
  oMn = min( oMn1 , oMn2 ) ;          /* oMn = AF1-1 or A-F1-2 */
  oMn = min( oMn , oMn3  ) ;          /* oMn = oMn or A-F1-3  */

  return ( oMn ) ;
}

/* -- Appendix G, Plate Girders -- */
/*** -- << Slender Elements >> -- ***/
if ( lambda_WLB <= 260.0 || lambda_WLB <= (14000/sqrt(Fy*sqrt(Fy+Fr))) )
{
  /* Capacity as Plate Girder, Slender Web limit: hc/tw <= 260.0   */
  /* oMn = oSxcRpgReFcr                                             */
  /* where:                                                         */
  /*       o   = 0.9, Resistance factor for flexure                 */
  /*       Sxc = Section modulus to compression flange              */
  /*       Rpg = 1.0 - 0.0005ar( hc/tw - 970/sqrt(Fcr) ) <= 1.0     */
  /*       Re  = 1.0 (Not hybrid)                                   */
  /*       Fcr = Fy                          : l < lp               */
  /*       Fcr = CbFy[1-1/2(l-lp)/(lr-lp)] <= Fy   : lp < l < lr    */
  /*       Fcr = Cpg/l^2                     : l > lr               */
  /*       Cpg = 286,000Cb                   : LTB                  */
  /*       Cb  = 1.75+1.05(M1/M2)+0.3(M1/M2)^2    :                 */
  /*       Cpg = 11,200                      : FLB                  */
  /*       Cb  = 1.0                         :                      */
  /*                                                                */
  /* Calculate Independent Quantities */
  Sxc = Sx ;
  Re  = 1.0 ;

  /* Lateral Torsional Buckling Nominal Resistance */
  rT = sqrt((tf*bf*bf*bf/12.0)/(bf*tf+hc*tw/6.0)) ;
  lambda = Lb/rT ;
  lambda_p = 300/sqrt(Fy) ;
  lambda_r = 756.0/sqrt(Fy) ;
  Cpg = 286000.0*Cb ;
  if ( lambda <= lambda_p )
  {
    Fcr = Fy ;
  }
  else if ( lambda <= lambda_r )
  {
    Fcr = Cb*Fy*(1.0-0.5*(lambda-lambda_p)/(lambda_r-lambda_p)) ;
    Fcr = min( Fcr , Fy ) ;
  }
  else
  {
    Fcr = Cpg/(lambda*lambda) ;
  }
  Rpg = 1.0-0.0005*(hc*tw)/(bf*tf)*(lambda_WLB-970.0/sqrt(Fcr)) ;
  oMn_LTB = 0.9*Sxc*Rpg*Re*Fcr ;

  /* Flange Local Buckling Nominal Resistance */
  lambda = lambda_FLB ;
  lambda_p = lp_FLB ;
  lambda_r = 150.0/sqrt(Fy) ;
  Cpg = 11200.0 ;
```

```
    Cb = 1.0 ;
    if ( lambda <= lambda_p )
    {
      Fcr = Fy ;
    }
    else if ( lambda <= lambda_r )
    {
      Fcr = Cb*Fy*(1.0-0.5*(lambda-lambda_p)/(lambda_r-lambda_p)) ;
      Fcr = min( Fcr , Fy ) ;
    }
    else
    {
      Fcr = Cpg/(lambda*lambda) ;
    }
    Rpg = 1.0-0.0005*(hc*tw)/(bf*tf)*(lambda_WLB-970.0/sqrt(Fcr)) ;
    oMn_FLB = 0.9*Sxc*Rpg*Re*Fcr ;

    /* -- Select oMn as minium of LTB and FLB -- */
    oMn = min( oMn_LTB , oMn_FLB ) ;

    return ( oMn ) ;
  }

  /* Very Slender Element(s) and Cases In Error; */
  if ( (lambda_WLB > 260.0) || lambda_WLB <= (14000/sqrt(Fy*sqrt(Fy+Fr))) )
  fprintf(out,"\n\t hc/tw Limit Exceeded...") ;0
  /* Default Moment Capacity */
  fprintf(out,"\n\t Moment may be incorrect: based on A-B5:FLB") ;
  oMn = Fy*Qs*Sx ;
  return ( oMn ) ;

} /*** end moment_capacity ***/

void combined_stress( double oPn, double oVn, double oMn, int i )
{
double a_r, m_r, a, m, capacity, shear ;

  /* Initialize Axial Ratio & Moment Ratio */
  a_r = Pu/oPn ;                                  // Axial Term
  m_r = fabs(Mu/oMn) ;                            // Moment Term
  if ( a_r > 0.2 )
  {
    /* LRFD Eqn.: H1-1a */
    a = a_r ; m = 8.0/9.0*m_r ;
  }
  else
  {
    /* LRFD Eqn.: H1-1b */
    a = a_r/2.0 ; m = m_r ;
  }

  /* Print Interaction Formulae */
  capacity = a + m ;
  shear = fabs(Vu/oVn) ;                          // Shear Ratio
  fprintf(out, " %9.3lf", capacity ) ;
  fprintf(out, " %9.3lf", shear ) ;

  /* Check for Maximums */
  critical_ratios( i, capacity, shear ) ;

  /* Return to Code_check */
  return ;

} /*** end combined_stress() ***/

void critical_ratios( int mem, double capacity, double shear )
{
  if ( capacity > CR[mem] )                       // Extract Capacity Ratio
```

```
  {
    CR[mem] = capacity ;
    Cap_Case[mem] = load_case ;
  }
  if ( shear > VR[mem] )                          // Extract Shear Ratio
  {
    VR[mem] = shear ;
    Shear_Case[mem] = load_case ;
  }
  return ;
} /*** end critical_ratios() ***/

int member_properties( ELEMENT *element )
{

  /* -- Read: BF, TF, TW -- */
  fscanf( in, "%lf %lf %lf", &element->BF, &element->TF, &element->TW ) ;

  /* -- Read: HCA, AGA, IXA, SXA, ZXA -- */
  fscanf( in, "%lf %lf %lf %lf %lf", &element->HCA, &element->AGA,
                                      &element->IXA, &element->SXA,
                                      &element->ZXA ) ;

  /* -- Read: HCB, AGB, IXB, SXB, ZXB -- */
  fscanf( in, "%lf %lf %lf %lf %lf", &element->HCB, &element->AGB,
                                      &element->IXB, &element->SXB,
                                      &element->ZXB ) ;

  /* -- Read: Iy -- */
  fscanf( in, "%lf", &element->IY ) ;

  /* Return to Main() */
  return ( NO_ERROR ) ;

} /*** end member_properties ***/

void pause(void)
{
  printf("\n\n Strike a key when ready...") ;
  while ( !kbhit() ) ;
  getch() ;
  return ;
}
/*****************************************************************************/
/* End of LRFDSPEC Program                                                   */
/*****************************************************************************/
```

VITA

Eric J. Wishart was born on February 28, 1963 in Providence, RI. After changing majors twice, including two years of *Music Production & Engineering* at Berklee College of Music, he finally received a Bachelor of Science in *Civil & Environmental Engineering* from the University of Rhode Island in May, 1989. He is currently completing requirements for a Master of Science degree in *Civil Engineering* from Virginia Polytechnic Institute & State University. He plans to return to New England upon graduation.