2009-07-02

Optimized Distribution of Strength in Buckling-Restrained Brace Frames in Tall Buildings

Graham Thomas Oxborrow
Brigham Young University - Provo

OPTIMIZED DISTRIBUTION OF STRENGTH IN BUCKLING-RESTRAINED BRACE FRAMES IN TALL BUILDINGS

by

Graham T. Oxborrow

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Civil and Environmental Engineering

Brigham Young University

August 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Graham T. Oxborrow

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Paul W. Richards, Chair
Date	Richard J. Balling
Date	Fernando S. Fonseca

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Graham T. Oxborrow in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Paul W. Richards
Chair, Graduate Committee

Accepted for the Department

E. James Nelson
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

OPTIMIZED DISTRIBUTION OF STRENGTH IN BUCKLING-RESTRAINED BRACE FRAMES IN TALL BUILDINGS

Graham T. Oxborrow

Department of Civil and Environmental Engineering

Master of Science

Nonlinear time history analysis is increasingly being used in the design of tall steel structures, but member sizes still must be determined by a designer before an analysis can be performed. Often the distribution of story strength is still based on an assumed first mode response as determined from the Equivalent Lateral Force (ELF) procedure. For tall buckling restrained braced frames (BRBFs), two questions remain unanswered: what brace distribution will minimize total brace area, while satisfying story drift and ductility limits, and is the ELF procedure an effective approximation of that distribution? In order to investigate these issues, an optimization algorithm was incorporated into the OpenSees dynamic analysis platform. The resulting program uses a genetic algorithm to determine optimum designs that satisfy prescribed drift/ductility limits during nonlinear time history analyses. The computer program was used to

investigate the optimized distribution of brace strength in BRBFs with different heights. The results of the study provide insight into efficient design of tall buildings in high seismic areas and evaluate the effectiveness of the ELF procedure.

ACKNOWLEDGMENTS

I wish to thank Professor Paul W. Richards for his help and support in working through the formulation of this thesis. I also wish to thank Professor Richard J. Balling and Professor Fernando S. Fonseca for all the knowledge and help that they have given me. Each of these people has taught me valuable lessons while attending Brigham Young University. Finally, I would like to thank my wonderful wife, Heather, for the support and strength she has provided me throughout the compilation of this research.

TABLE OF CONTENTS

LIST OF TABLES ..... **ix**

LIST OF FIGURES ..... **xi**

1 Introduction **1**

2 Previous Research **3**

2.1 BRBF Experimental Studies 3

2.2 BRBF Analytical Studies 4

2.3 Optimization 6

3 Equivalent Lateral Force Procedure **11**

4 Modeling Techniques **15**

5 Optimization Procedure **21**

5.1 Summary of Optimization Procedure 21

5.2 Optimization Input and Output Parameters 25

6 Results of Optimization **27**

7 Conclusions **39**

7.1 Possible Future Work 40

References **43**

Appendix A Source Code **47**

A.1 Main 47

A.2 Random_Selection_Crossover_Mutation 51

A.3 Maximum_Fitness 56

A.4 DisplayPlane_DisplayModel2D 57
A.5 EqScaling 60
A.6 AppInitialize.v6.1.6 63
A.7 AppAnalysis.v5.2 67
A.8 Gravity_Analysis.v4.1 70
A.9 Dynamic_Analysis.v5 72
A.10 Input.v7 76
A.11 Monotonically Decreasing Version 89
A.11.1 Main_Mono 90
A.11.2 AppInitialize.Mono 94
A.11.3 AppAnalysis.Mono 98
A.11.4 Input.Mono 100
Appendix B Equivalent Lateral Force Procedure 113
B.1 Construct Design Response Spectra 113
B.2 Determine Fundamental Period of Structure 116
B.3 Determine Base Shear 116
B.4 Distribute Story Forces 117
B.5 Determine Brace Core Areas 117

LIST OF TABLES

Table 3-1. Average Brace Areas from ELF Procedure (in^2).....14

Table 5-1. Optimization Parameter Definitions.....23

Table 5-2. Optimization Input Parameters.....26

Table 6-1. Average Brace Areas from Optimization Procedure (in^2).....30

Table 6-2. Time to Complete Optimization (hr).....36

LIST OF FIGURES

Figure 3-1. 3-story ELF normalized brace area 12
Figure 3-2. 6-story ELF normalized brace area 12
Figure 3-3. 9-story ELF normalized brace area 13
Figure 3-4. 12-story ELF normalized brace area 13
Figure 3-5. 18-story ELF normalized brace area 14
Figure 4-1. BRBF elevation view 17
Figure 4-2. Spectra for earthquake records used for 3-, 6-, and 9-story frames 18
Figure 4-3. Spectra for earthquake records used for 12- and 18-story frames 19
Figure 5-1. Flowchart for optimization procedure 23
Figure 6-1. 3-story optimal normalized brace area 27
Figure 6-2. 6-story optimal normalized brace area 28
Figure 6-3. 9-story optimal normalized brace area 28
Figure 6-4. 12-story optimal normalized brace area 29
Figure 6-5. 18-story optimal normalized brace area 29
Figure 6-6. Ductility results for optimal 3-story design 31
Figure 6-7. Ductility results for optimal 6-story design 31
Figure 6-8. Ductility results for optimal 9-story design 32
Figure 6-9. Ductility results for optimal 12-story design 32
Figure 6-10. Ductility results for optimal 18-story design 33
Figure 6-11. 9-story optimal normalized brace area with similar columns 34
Figure 6-12. Monotonically decreasing 9-story optimal normalized brace area 35

Figure 6-13. Total brace area of optimal design during optimization process 37

Figure B-1. Response spectra 115

1 Introduction

Typical structural design can be separated into two distinct sections: gravity load design and lateral load design. Some examples of lateral loads are wind and earthquake forces. To compensate for these forces, designers will incorporate lateral force resisting systems into the structure, which absorb and transfer these loads to the foundation of the structure. In steel structures, these systems are composed of frames which are allowed to yield or, in other words, leave the elastic region of the steel material. The yielding of the frames should control damage to the rest of the structure and prevent the loss of life. This study focuses on designs composed of Buckling-Restrained Brace Frames (BRBF).

The use of BRBFs has increased in the United States over the last couple of decades. BRBFs are different than typical ordinary braced frames in their design and behavior. Braces in BRBFs are composed of a steel core encased in concrete and a surrounding steel tube. The steel core is coated in a material to prevent bonding problems between the core and the concrete. These braces exhibit similar behavior in compression and tension since the brace is not allowed to buckle by the concrete-filled steel tube.

As the use of BRBFs increases, designing cost effective structures with BRBFs has become more important to the engineer and to the client. Since the cross-sectional area of the braces are directly related to the cost of the building, designers prefer to minimize the total cross-sectional area of the braces in the building. Currently, one of the common

methods for determining this brace area is to use the Equivalent Lateral Force (ELF) procedure (ASCE 2005) in conjunction with BRBF design principles from AISC 341 (2005). However, little has been done in applying advanced optimization methods to BRBF structural design or to evaluate the effectiveness of the ELF procedure with respect to BRBFs.

This thesis focuses on finding an optimal distribution of brace strength for tall BRBF structures, namely, 3-, 6-, 9-, 12-, and 18-story buildings, under seismic loads and comparing those results with the distributions from the ELF procedure. This thesis begins by discussing the previous research that has been performed in the analysis of BRBFs and the optimization of steel frames. Next, the ELF procedure is discussed. An overview of the optimization procedure, with the input and output parameters defined, is included. Finally, the results of the optimization and the conclusions are presented.

2 Previous Research

2.1 BRBF Experimental Studies

For the past few decades, research into the material properties and structural response of buckling restrained braces (BRBs) has been increasing. One of the earliest reports on the characteristics of BRBs was done by Watanabe, A., et al. (1988). Watanabe indicated that a steel core member encase in a concrete filled tube produced a ductile brace with a desirable hysteretic behavior. Later, Watanabe and Nakamura (1992) proved that a material with a much lower yield stress could be used in the fabrication of such a steel core and still produce desired results.

One of the first large-scale tests of BRBFs in the United States was performed by Aiken, I., et al. (2002). They noted that BRBFs performed well in single-diagonal configurations as well as chevron. Also, they found that the axial and flexural demands on the brace did not affect the hysteretic and elongational behavior of the brace.

Black, C., et al. (2004) tested BRB specimens, as part of a comprehensive component testing program, to determine how closely the predicted values were to actual performance values. They determined that the plastic torsional buckling mode was the most critical to the brace's stability. They recommended some restrictions on the width-thickness ratio of brace cores, as well. Overall, they reported that BRBs are an

appropriate alternative to conventional framing systems since they provide ductile, stable and repeatable hysteretic behavior.

Roeder, et al. (2006) performed large scale tests on BRBF specimens. The results of the tests indicated that large bending moments were transferred from the gussets plates to the beams and columns. They observed that tapered gusset plates, viewed as less rigid connections, provided better response than rectangular ones. Overall, they recommend that BRBF connections be designed as smaller, more compact and more flexible.

Fahnestock, L., et al. (2007) performed a large-scale test on a four story prototype BRBF building. They concluded that properly detailed BRBFs will resist ductility demands and large drifts. They noted that the first and second story frames dissipated the most energy under the earthquake loading.

Takeuchi, et al. (2008) addressed the fact that BRBs in Japan are being used as elastoplastic dampers to dissipate the largest amount of seismic energy possible. They concluded that the energy adsorption capacity of BRBs can be estimated through the hardening effect of the steel. They also recommended a method for estimating the cumulative deformation capacity of BRBs by breaking down the hysteresis loop into the skeleton part and the Bauschinger part.

2.2 BRBF Analytical Studies

Many researchers have used finite element modeling (FEM) techniques to investigate the structural response of BRBFS. Sabelli, R. Mahin, S. and Chang, C. (2003) examined the seismic response of BRBFs using nonlinear dynamic time history analysis. They concluded that, although the braces exhibited adequate hysteretic behavior, they are

likely to develop significant bending and shear forces in the overall frame system and the effect that these forces would have is unclear. They recommended future analysis into this area of BRBF system behavior.

Kim, J. and Choi, H. (2004) investigated the energy dissipation capacity and earthquake response of BRBFs. They used 5-, 10-, and 20- story computer models to examine these effects. As a result, they determined that the maximum displacements of the structures decrease with an increase in BRB stiffness. Most importantly, they indicate that the most practical way to design BRBs to attain a desirable behavior is to vary the cross-sectional area of the brace compared to varying steel strength. This method is currently employed by designers for BRBs.

Fahnestock, L., et al. (2007) used finite element models to assess this overall system response and performance of BRBF systems through nonlinear dynamic time history analysis. Fiber elements were employed for the beam and column cross sections while inelastic truss elements were used to models the BRBs. They reported that BRBFs designed with the ELF procedure were adequate for required performance levels under the maximum considered earthquake. They did recommend reconsideration of the overstrength factor in the seismic design provisions.

Asgarian, B. and Amirhesari, N. (2008) investigated the differences in performance between Ordinary Brace Frames (OBF) and BRBFs. They concluded, through nonlinear dynamic analysis, that BRBFs are superior in performance and behavior than OBFs. The OBFs are limited in performance and behavior, since they experience bracing member buckling, while BRBFs do not buckle but instead show stable hysteretic behavior.

For a complete review of the current design specifications for BRBFs, the reader is referred to the *AISC 341, Seismic Specifications for Structural Steel Buildings* (2005), and recommendations by Sabelli, R. and Pottebaum, W. (2005). In these publications, specifications and limits were introduced as per the detailing of BRBF members as well as their connections to other members in the framing system.

2.3 Optimization

In the field of structural optimization, little research has been done in relation to the strength distribution of BRBs in structures; however, other steel frame systems have been designed using genetic algorithms. Different analysis procedures have been used during optimization from static pushover analysis to nonlinear time history analysis while the elements in each procedure were either elastic or inelastic.

Pezeshk, et al. (2000) minimized the total weight of the frame while complying with the strength and stability requirements from the AISC-LRFD specifications. In the study, they examined the differences between three different design procedures under elastic static pushover analysis: linear excluding P-Δ effects, linear including P-Δ effects, and geometric nonlinear analysis. They determined that using group, or tournament, selection and an adaptive crossover operator worked well in the design of plane frames. They also determined that using geometric nonlinear analysis produces more efficient designs than linear analysis, with or without P-Δ effects.

The conclusions by Pezeshk, et al. are supported by further research by Hayalioglu, M. and Degertekin, S. (2004) during an elastic static analysis. They determined that increasing the fitness scaling of infeasible designs and the probability of crossover

increased the speed to convergence. In addition, they concluded that reducing the connection stiffness improved the optimal design and the sway of the frames during the static pushover analysis. These improvements occurred from the increased flexibility would impact the drift of the frame so larger shapes were assigned to the optimal design. Since the shapes came from a discrete set, major changes to frame behavior were observed.

While the aforementioned studies minimized the weight of the structure, Kameshki, E. and Saka, M. (2001) expanded an elastic static pushover analysis to evaluate different framing configurations. They analyzed and optimized V- (chevron), two story X-, and Z-bracing configurations. They concluded that two story X-bracing configurations were the lowest weight design by using strength and serviceability limit states. One significant difference in this study was that back-to-back double equal-leg angles were used as the bracing members.

Liu, et al. (2003) observed that many optimization procedures focus on minimizing the weight a structure while ignoring the lifecycle costs. In light of this observation, Lui, et al. employed a nonlinear static pushover analysis to investigate the minimum weight and lifecycle cost objectives. An interesting aspect in their study was the high rate of mutation that was used to encourage exploration of the design space. They concluded that using the minimum weight objective is an adequate design approach as long as other design alternatives are considered such as lifetime seismic design costs and detailing/erection complexity costs.

Using a different analysis and optimization approach, Memari, A., and Madhkhan, M. (1999) developed a program with pseudo-dynamic modal analysis and gradient-based

optimization. For optimization, they used cross-sectional areas as the design variable with the minimization of total structure weight as the objective. They confirmed that changing the cross-sectional areas to minimize total structure weight and, in turn, find the most efficient design, serves well for the structural optimization of steel moment frames.

All of the previous optimization studies have used some form of equivalent method to predict the response of the structure. However, as computing speeds have increased, the need for equivalent procedures has decreased. Lagraos, et al. (2006) investigated the influence that these various methods had on the performance of real-scale steel buildings. They determined that genetic nonlinear time history techniques can be considered adequate and efficient for the design of real-world frames. These techniques will produce designs with substantially less material weight. Another observation was that artificially created ground motions produce heavier structures than naturally recorded motions.

In another study, Ohsaki, et al. (2007) used nonlinear time history analysis with multiobjective optimization for steel frames. They concluded that the introduction of a sharing, or crossover, function improved the accuracy and diversity of the Pareto, or optimal, solutions.

In the field of BRBF optimization, Balling, R., et al. (2009) applied genetic optimization methods with nonlinear time history analysis to 1-, 3-, and 5-story buildings to determine the optimum brace areas using nonlinear time history analysis. They presented design charts for 1-, 3-, and 5-story structures based on their optimum designs. They also noted that brace areas varied linearly from story to story which is not the case for the ELF procedure. They finally recommended that the ELF procedure be revised for BRBFs, since it was determined to be either insufficient or excessively conservative.

Many of the previous nonlinear time history studies selected earthquakes carefully, usually based on some site-specific characteristics, and then scaled ground motions accordingly. However, Iervolino, I. and Cornell, C. (2005) suggested that carefully selecting the earthquakes, and then scaling them, might not be necessary. They determined that randomly selected earthquakes, which were then scaled, did not induce bias on the nonlinear response. Nevertheless, more research to validate these conclusions is needed as the available set of earthquake records increases, which can provide a clearer picture of nonlinear structural response.

3 Equivalent Lateral Force Procedure

This study focuses on comparing the optimal results of tall multistory BRBF frames through optimization techniques with those designed using the Equivalent Lateral Force (ELF) procedure. The ELF procedure, due to its simplicity, is typically used in lieu of dynamic analysis to estimate the lateral loads that act on each floor of the building. The braces are then designed with these lateral loads. The steps of this procedure are

1. Construct the design response spectra;
2. Determine the fundamental period of the structure;
3. Determine the seismic base shear;
4. Convert base shear into story forces;
5. Determine the brace core areas.

A more detailed explanation of this procedure can be found in Appendix B.

For this particular study, the site soil conditions were assumed to be Site Class D and the structures were located in a high seismic area. The overall design parameters, including height and width, are summarized in a following section. Figure 3-1 through Figure 3-5 contain the results of the ELF procedure for the 3-, 6-, 9-, 12-, and 18-story BRBF structures with the normalized brace area on the abscissa and the associating story on the ordinate. Each brace area is normalized by the average brace area of the structure.

Also, Table 3-1 contains the average brace areas associated with each structure. Discrete brace sizes were used during this procedure which led to similar sizes at lower stories.

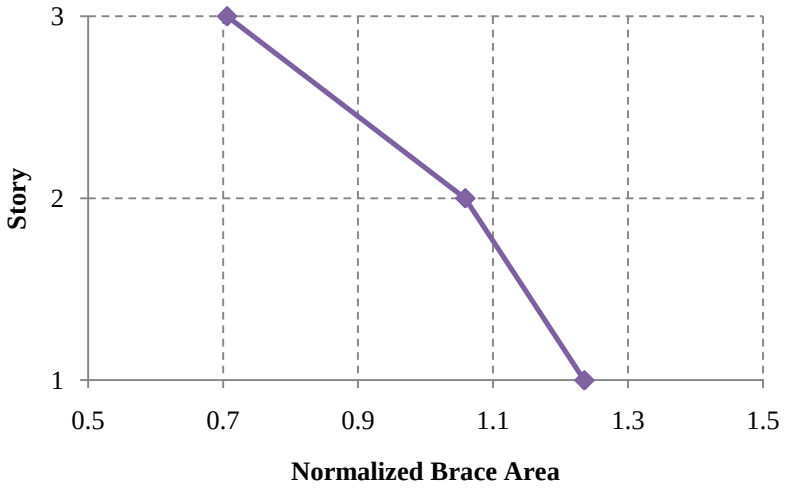

Figure 3-1. 3-story ELF normalized brace area

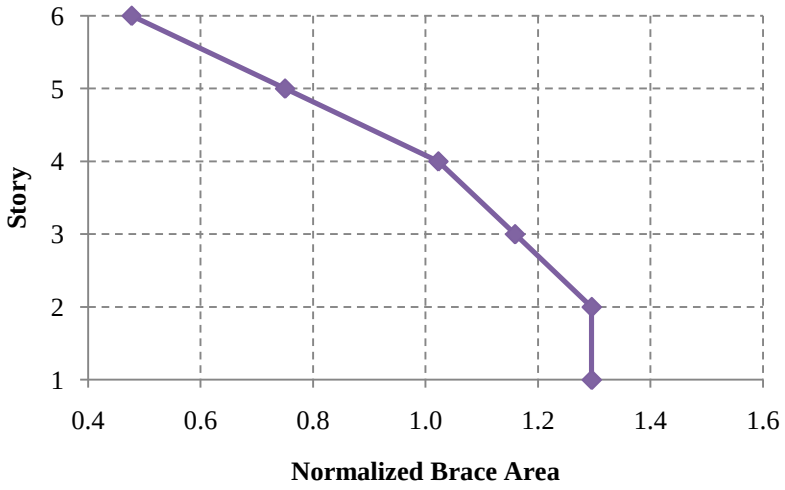

Figure 3-2. 6-story ELF normalized brace area

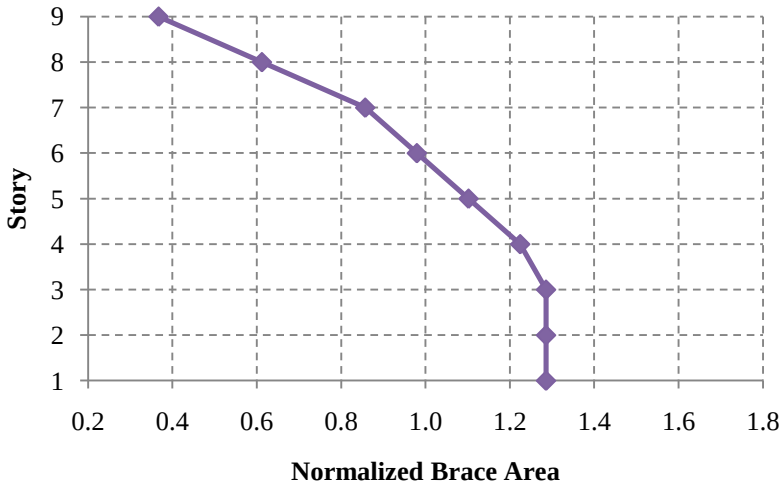

Figure 3-3. 9-story ELF normalized brace area

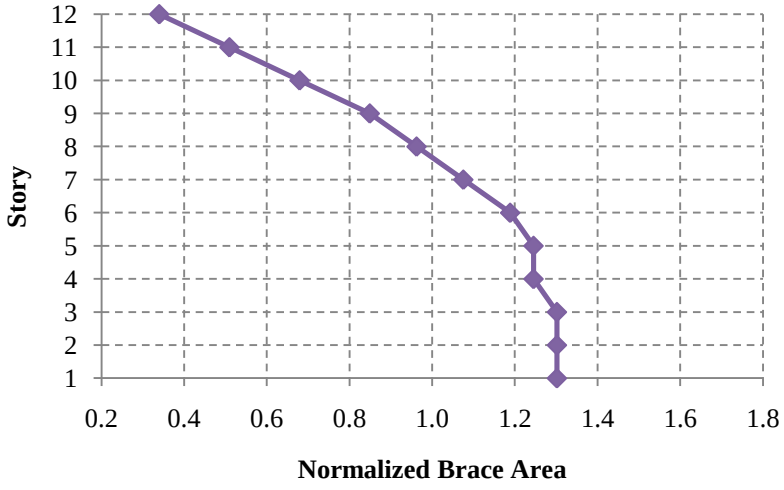

Figure 3-4. 12-story ELF normalized brace area

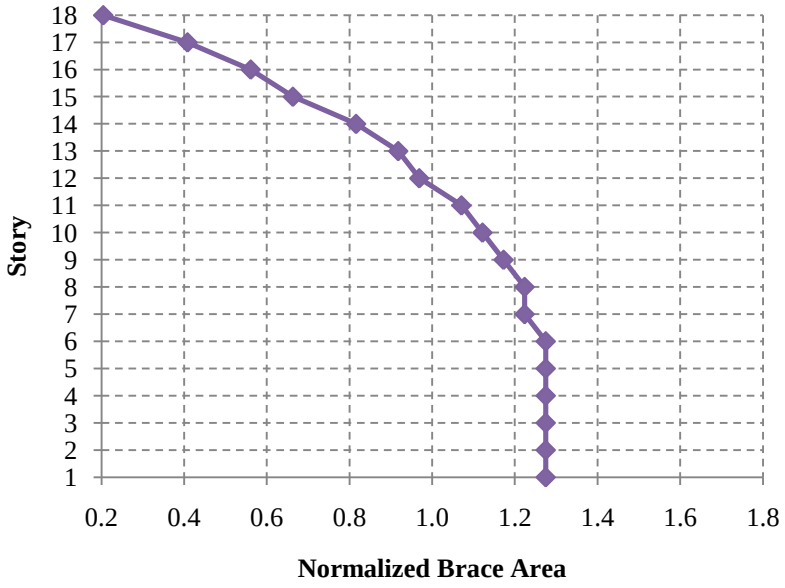

Figure 3-5. 18-story ELF normalized brace area

Table 3-1. Average Brace Areas from ELF Procedure (in^2)

Number of Stories	Average Brace Area
3	5.67
6	7.33
9	8.17
12	8.83
18	9.81

4 Modeling Techniques

Two-dimensional models were created in the nonlinear dynamic analysis platform known as the Open System for Earthquake Engineering Simulation (OpenSees). The models represented single bay study frames of varying heights. Story heights were 15 ft and bay widths were 25 ft in each direction for all cases. Story masses corresponded to a four bay by four bay building with 90 psf dead load. The only parameter that varied for the different frames was the number of stories.

The beams and columns were modeled with fiber sections and force-based nonlinear beam-column elements. A steel material using the Guiffre, Pinto, and Mengetto steel model (Steel02) was used so material hardening properties could be defined along with a previously defined yield stress and modulus of elasticity. A paper by Calado, et al. (2002) provides an adequate explanation of the different parameters, such as hysteretic properties, required for this particular steel model and values recommended by Mazzoni, et al. (2006) were used. The assumed yield stress of the beams and columns was 50 ksi.

The buckling restrained braces were modeled using corotational truss elements with the cross-sectional area as the design variable. “The corotational formulation adopts a set of corotational axes which rotate with the element, thus taking into account an exact geometric transformation between local and global frames of reference” (Mazzoni, et al. 2006). Brace sections were defined to account only for axial forces and deformations;

moments at the end of the braces were assumed to be negligible. The material Steel02 was used for the construction of the brace elements and was calibrated to produce hysteretic behavior consistent with BRBF test results, following Coy (2007). The assumed nominal yield stress of the braces was 45 ksi.

During the finite element modeling, a constant area is applied to the total length of the BRB. However, BRBs are composed of smaller cross-sectional areas along the middle portion of the brace and this area is expected to yield. This portion is then bounded by larger cross-sectional areas that should not yield. To compensate for the brace composition, a typical modulus of elasticity, *E*, of 29,000 ksi was not used but instead an effective modulus of elasticity, E_{eff}, was calculated using Equations 4–1 through 4–3.

$$E_{eff} = E * \frac{L_1}{L_{BRB\ ,Yield}} \tag{4-1}$$

$$\mathrm{L_{BRB\ ,Yield}} = 0.85(\mathrm{L_1} - 2\mathrm{L_2}) \tag{4-2}$$

$$L_2 = \frac{d_{beam}}{\sin\theta} + 24" \tag{4-3}$$

In the preceding equations, L_1 is the total brace length from centerline to centerline. The angle, θ, indicates the slope of the brace across the bay and L_2 is non-yielding length of the BRB. In Equation 4-3, 24” attempts to account for connection irregularities between different buildings. Also, $L_{BRB,Yield}$ was reduced by 15% to allow for a transition between non-yielding and yielding sections. A graphical representation of these variables is shown in Figure 4-1.

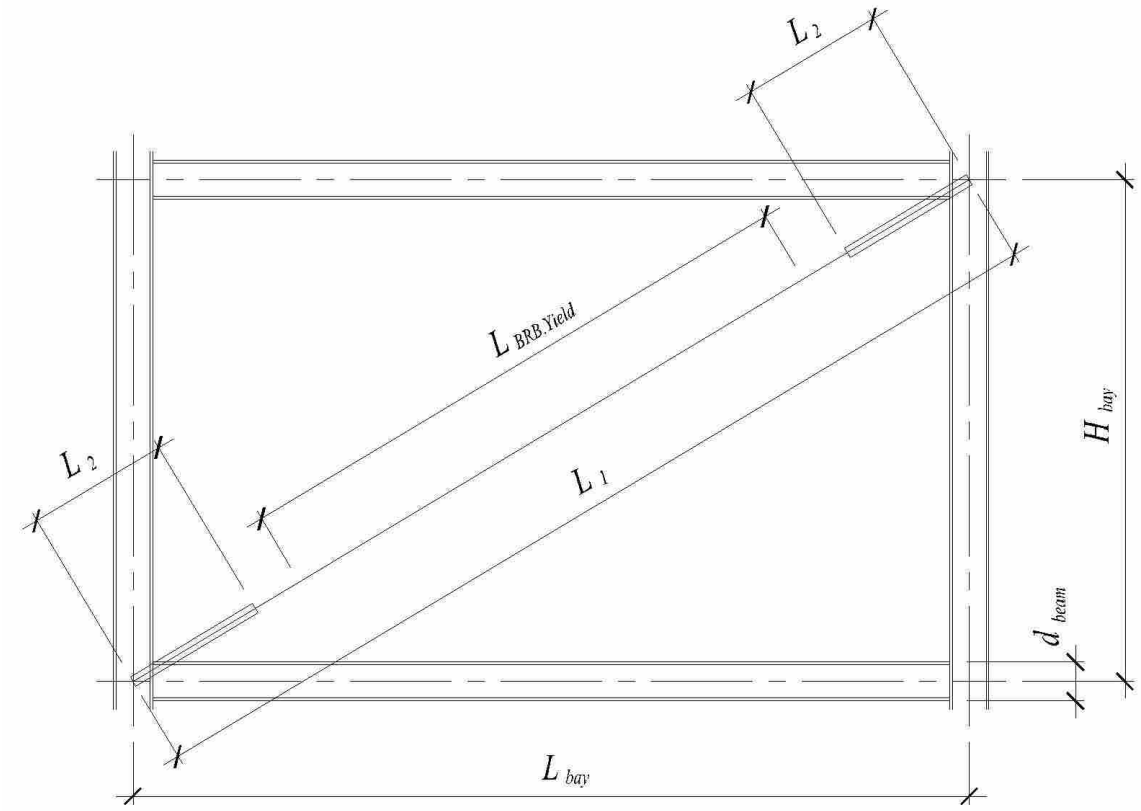

Figure 4-1. BRBF elevation view

During the optimization procedure described in the next section, different frames were generated and a nonlinear time history analysis was performed on each frame. A site was chosen with a design spectrum defined by S_{DS} equal to 1.2 and S_{D1} equal to 1.0. This site design spectrum was used for scaling the earthquakes to represent design level events. For each frame, earthquake records were scaled so the earthquake's spectral acceleration matched the design spectral acceleration at the frame's fundamental period.

Two different suites of ten earthquake records were used in this study. One of the suites was used for 3- to 9-story models while the other suite was used for 12- to 18-story models. Different suites were used to avoid excessive scaling when making the spectral acceleration of the records match the design spectrum as described above. Plots of the

earthquake response spectra and design response spectrum can be seen in Figure 4-2 and Figure 4-3. Raleigh damping with 2% for modes one and three was used in the analyses. For each frame, average brace deformations were calculated at each story from results from the ten earthquakes. These average brace deformations were then used by the optimization algorithm, as will be described in next section.

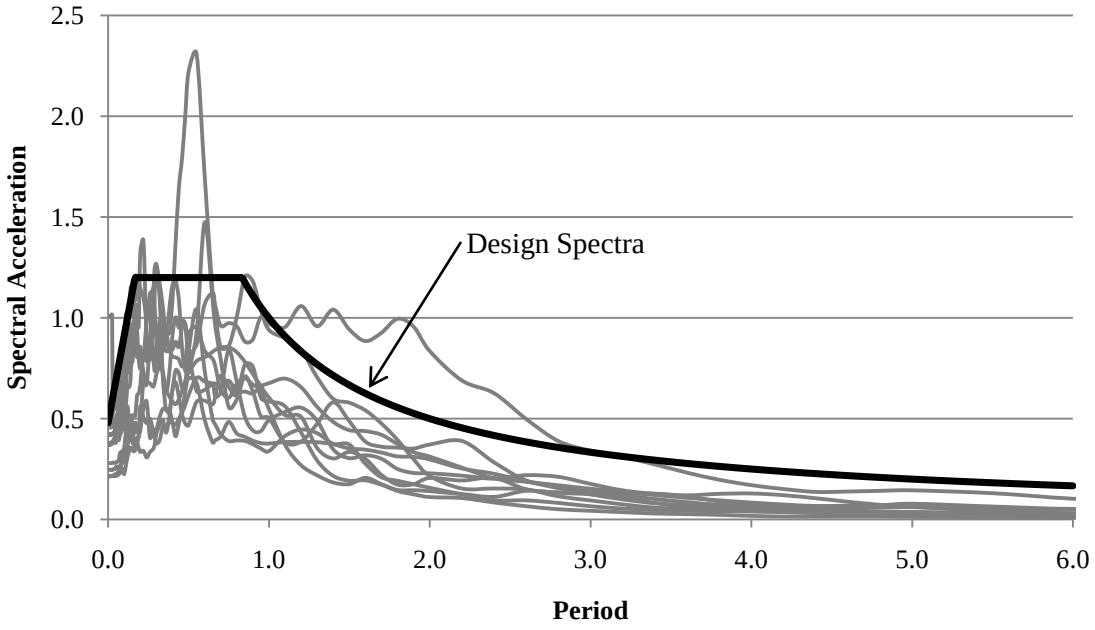

Figure 4-2. Spectra for earthquake records used for 3-, 6-, and 9-story frames

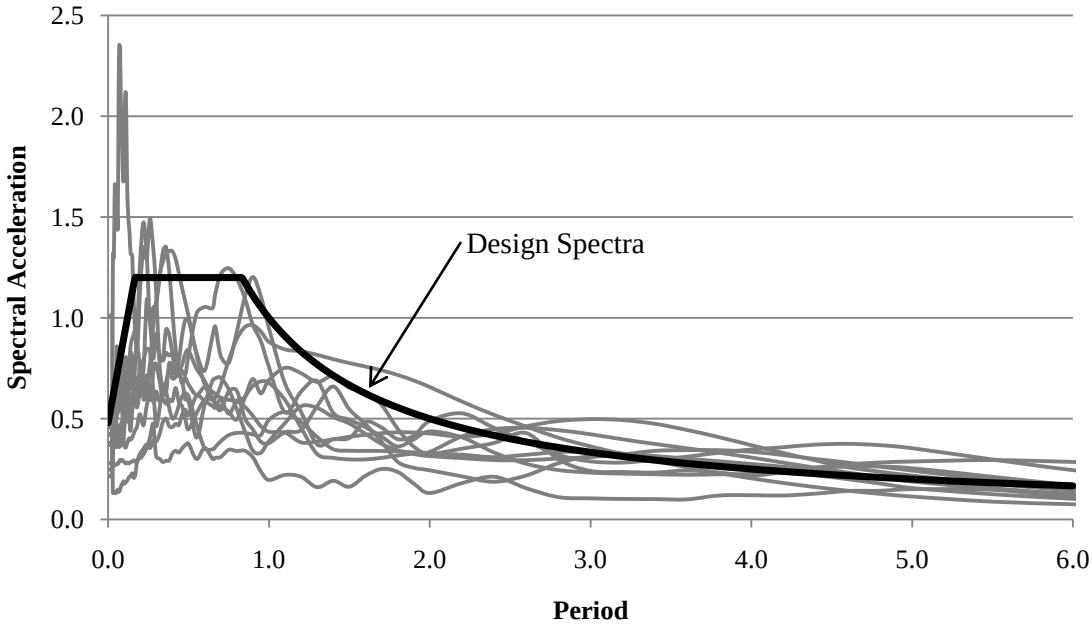

Figure 4-3. Spectra for earthquake records used for 12- and 18-story frames

5 Optimization Procedure

The optimization approach of this study involved the use of a genetic algorithm to determine the most efficient distribution of strength in tall multistory BRBF structures. Genetic algorithms mimic the evolutionary aspects of nature and, in essence, are based on the principle of "survival of the fittest." The genetic algorithm that was implemented followed that described in Balling, R. (2006). As in any genetic algorithm, constraints need be satisfied and an objective function evaluated to determine the efficiency of each design.

5.1 Summary of Optimization Procedure

For this study, drift and ductility were the constraints and total minimum brace weight was the objective. A summary of the BRB design constraints and objective is presented below.

Constraints:

$$\frac{\Delta_{xi} - allowable\ \ drift}{allowable\ \ drift} \leq 0 \qquad (5\text{-}1)$$

$$\frac{\mu_{xi} - allowable\ \ ductility}{allowable\ \ ductility} \leq 0 \qquad (5\text{-}2)$$

Objective:

$$Minimize \sum_{1}^{number\ \ of\ stories} a_i \quad (5\text{-}3)$$

In the previous equations, Δ_{xi} is the drift at *I* floor, μ_{xi} is the ductility of the brace at *I* floor and a_i is the brace area at *I* floor.

In the optimization procedure, the ductility at each story is determined based on the average brace deformation and compared to the ductility limit. Ductility is the ratio of total brace deformation, both elastic and inelastic, to the elastic deformation based on material properties. The elastic deformation was computed by Equation 5–4.

$$\Delta_e = {F_{y,brace}\, L_{brace}}\Big/{E_{brace}} \quad (5\text{-}4)$$

Next, the deformation is converted to a corresponding drift and compared to the building code-based limit.

Each constraint was normalized to their allowable values so that they could be compared during the fitness evaluation. If either constraint is violated, the design is ruled infeasible and penalized from use in future generations. The objective function is used to determine the fitness of all designs; fitter designs have a high probability of being used in the creation of future designs.

As a preface to the explanation of the exact procedure that is used, common parameters need to be defined; they are seen in Table 5-1. The last three parameters are variables that impact the selection, crossover, and mutation steps which occur after the first generation.

Table 5-1. Optimization Parameter Definitions

Parameter	Definition
nsize	Number designs created and analyzed per generation
ngener	Number of total generations or loops in the optimization process
ntourn	Number of designs randomly chosen during tournament selection
probcross	Crossover probability
probmutate	Mutation probability

An overview of the steps in the optimization procedure is given in Figure 5-1. As shown, the first set of designs is randomly generated, within specified limits, and then evolutionary procedures are used to create all subsequent designs.

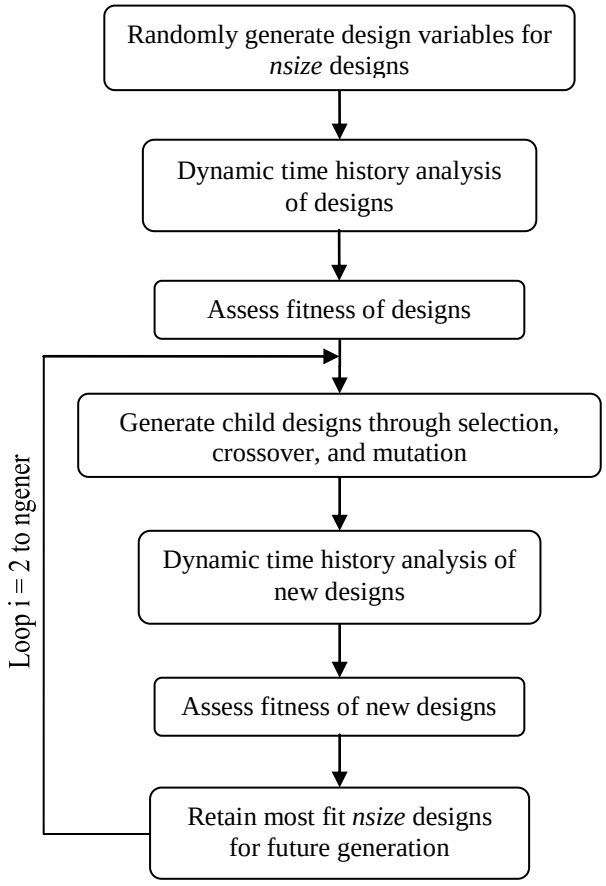

Figure 5-1. Flowchart for optimization procedure

The evolutionary methods of selection, crossover, and mutation are used in genetic algorithms. Selection involves randomly choosing *ntourn* designs and assigning the most fit design to be a parent design. During crossover, two parents are combined if a randomly generated number is greater that the crossover probability assigned at the beginning of optimization. After selection and crossover, mutation will occur if another randomly generated number is less than the mutation probability. For this study, a type of crossover called blend crossover was used, where the design variables are blended together to create the children; and dynamic mutation was used such that mutation becomes less likely in succeeding generations.

Each cycle of this loop is performed with the intention of converging on the optimal design with the lowest weight while still meeting constraint requirements. This genetic optimization method was chosen for its speed and ease in finding an optimal design in favor of gradient-based optimization methods or brute-force methods. As previously mentioned, this study investigates the optimal designs of 3-, 6-, 9-, 12-, and 18-story buildings with BRBFs and the use of genetic optimization will be a benefit since the number of design variables is increasing from 3 to 18.

In addition to the original program, a modified set of procedures was created with an additional constraint which can be seen in Equation 5–5. According to the ELF procedure, the lateral loads decrease up the structure leading to progressively smaller brace areas. The additional constraint required that the brace areas monotonically decrease, or become smaller, as one proceeds up the structure. If this assumption of decreasing brace area were true, then a more optimal design should be achieved in the

same number of generations than under the original program since the search area is smaller.

$$a_i \leq a_{i-1} \tag{5-5}$$

where *a* is the brace area at story *i* or story *i*-1.

5.2 Optimization Input and Output Parameters

Consistent parameters were used to assess the impact of the number of stories on steel distribution. A list of the optimization input parameters can be seen in Table 5-2. *Ngener* was varied based on the height of the building to allow sufficient time to search the design space for the optimal design. To increase the speed of this convergence, only a few earthquakes, chosen randomly from the suite of ten, were used until 80% of the generations had been completed. Limiting the earthquakes used permitted preliminary convergence and avoided unnecessary analysis of infeasible designs. During the final generations, the initial *nsize* designs were analyzed under the complete earthquake suite and each newly created design was fully analyzed unless it was deemed infeasible after three earthquakes. For the feasibility check, the allowable ductility ratio was 3.5 and the allowable story drift was 1.5%.

The optimization/analysis program outputs were story brace areas, fundamental period, average maximum drift at each story and story ductility. At the end of optimization, the output parameters were included in the final results for the fittest *nsize* designs for future review. Each brace area was normalized by the average brace area of the structure and the results are presented in the next section.

Table 5-2. Optimization Input Parameters

Parameter	3-	6-	9-	12-	18-
nsize	50	50	50	50	50
ngener	300	500	700	800	858
ntourn	6	6	6	6	6
probcross	0.6	0.6	0.6	0.6	0.6
probmutate	0.4	0.4	0.4	0.4	0.4

6 Results of Optimization

The modeling techniques and optimization algorithm, explained in the previous chapters, were applied successfully and produced a set of optimal designs. Figure 6-1 through Figure 6-5 contain the optimal designs for the 3-, 6-, 9-, 12-, and 18-story BRBF structures with the normalized brace area on the abscissa and the associating story on the ordinate. The results of the ELF procedure are also presented. Each brace area was normalized by the average brace area of the structure under the particular procedure. These charts only present a comparison of brace distribution not of brace area since the average brace areas of each procedure were different. In addition to the normalized charts, Table 6-1 presents the average brace areas used for normalization.

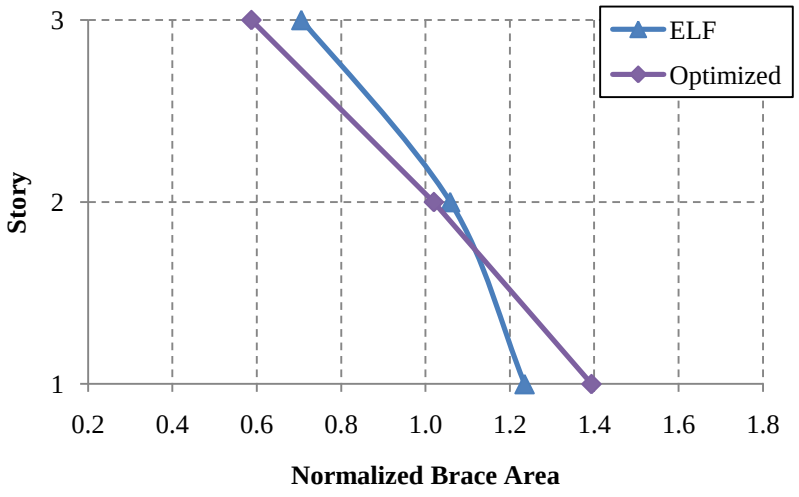

Figure 6-1. 3-story optimal normalized brace area

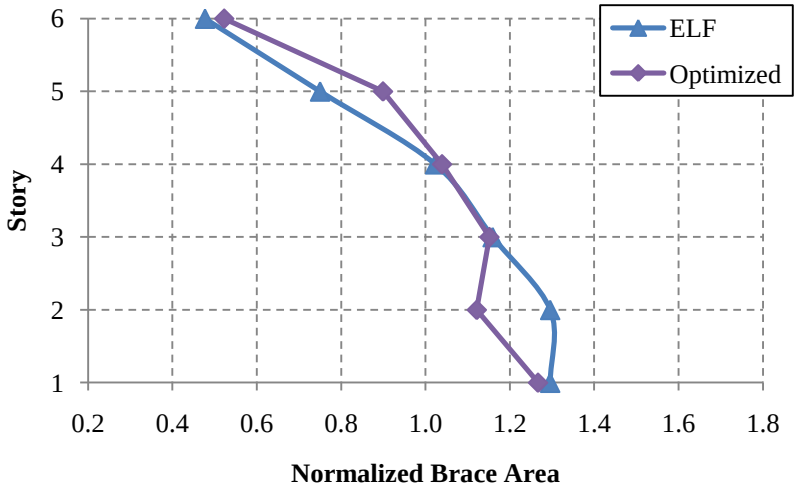

Figure 6-2. 6-story optimal normalized brace area

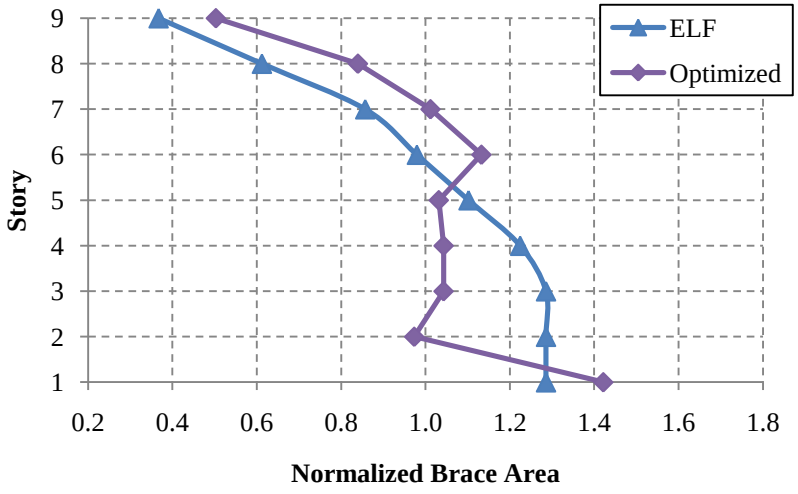

Figure 6-3. 9-story optimal normalized brace area

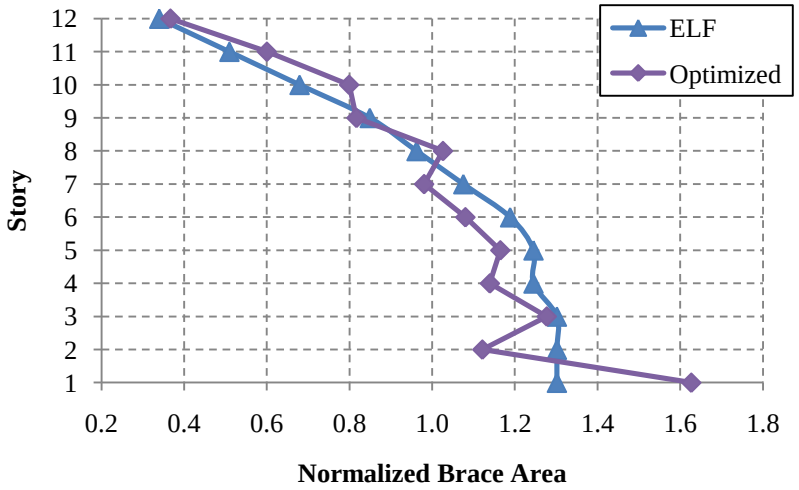

Figure 6-4. 12-story optimal normalized brace area

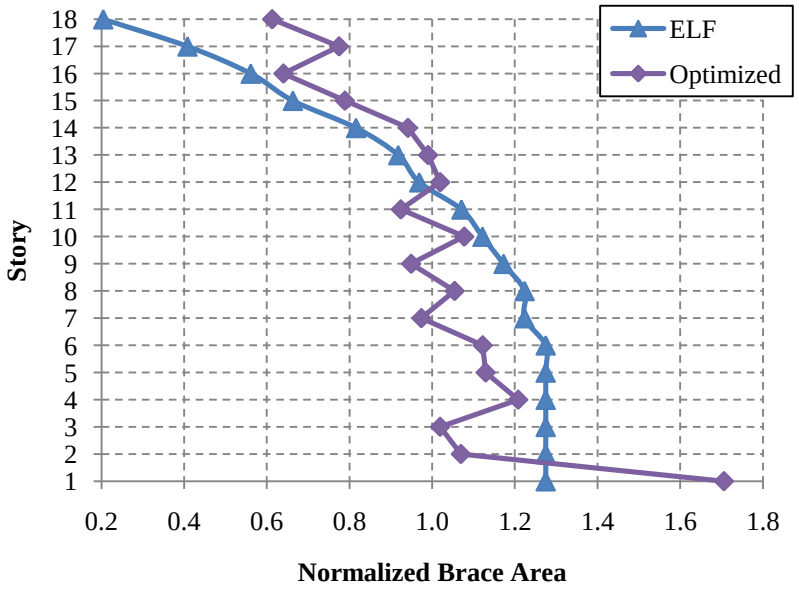

Figure 6-5. 18-story optimal normalized brace area

Table 6-1. Average Brace Areas from Optimization Procedure (in^2)

Number of Stories	Average Optimized Brace Area	Average ELF Brace Area
3	9.06	5.67
6	10.26	7.33
9	10.60	8.17
12	8.64	8.83
18	9.55	9.81

The preceding brace distributions followed a similar trend as those from the ELF procedure. For general trends of the optimal designs, the brace areas remained a similar size at the intermediate stories then become much smaller at the upper floors. Also, the brace size on the first floor was much larger than subsequent stories. The ELF procedure results did not show the larger brace at the bottom story but did produce the tapered brace sizes at the upper floors. Previous research by Balling, et al. (2009) suggested that the brace strength distribution was linear; however, this conclusion appears limited to low-rise frames. This limitation may be due to the influence of more complex modes of vibration as structures become taller.

The average brace areas between the two methods differ significantly below 9-stories since the optimized designs were larger than the results from the ELF procedure. This observation correlates with the fact that the results of the ELF procedure do not meet the ductility requirements when subjected to the suite of earthquakes used as will be seen later. If a larger suite was used, the optimized results might return an average brace area closer to the ELF procedure but this possibility would need to be evaluated through further research.

Before addressing possible causes for the different brace distributions between the two methods, graphs of the governing constraints in the optimized procedure will be

presented in Figure 6-6 through Figure 6-10 to aid in the discussion as well as the ductility results for the ELF designs. In all cases examined, ductility was the governing constraint with a limit of 3.5.

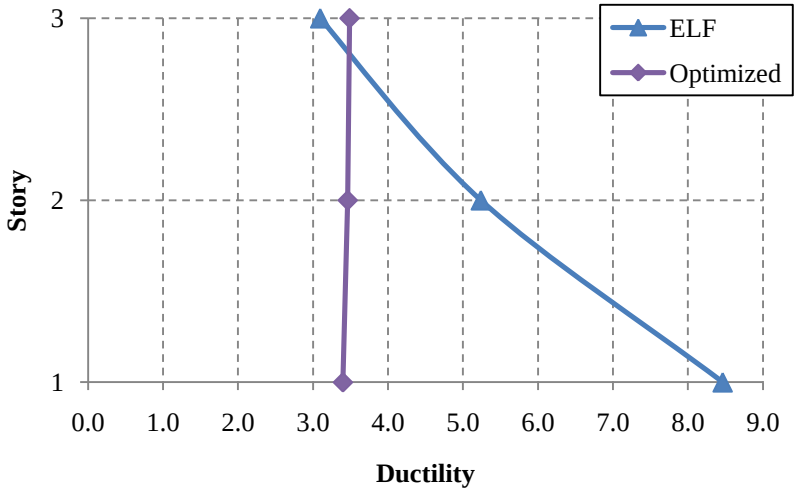

Figure 6-6. Ductility results for optimal 3-story design

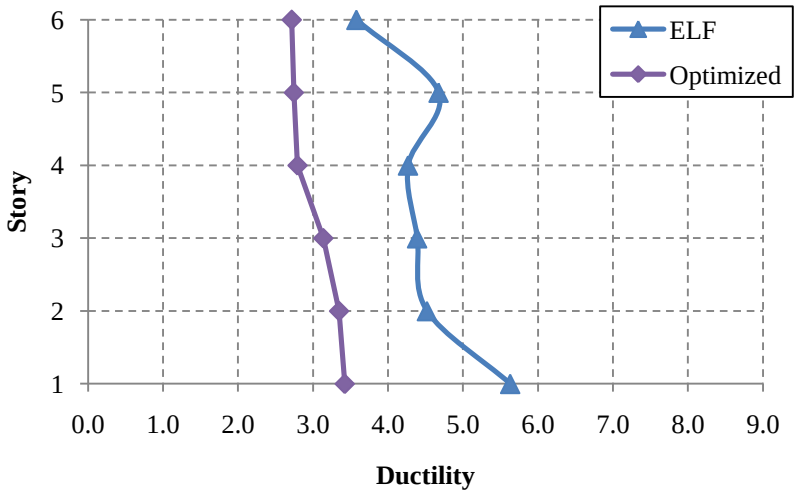

Figure 6-7. Ductility results for optimal 6-story design

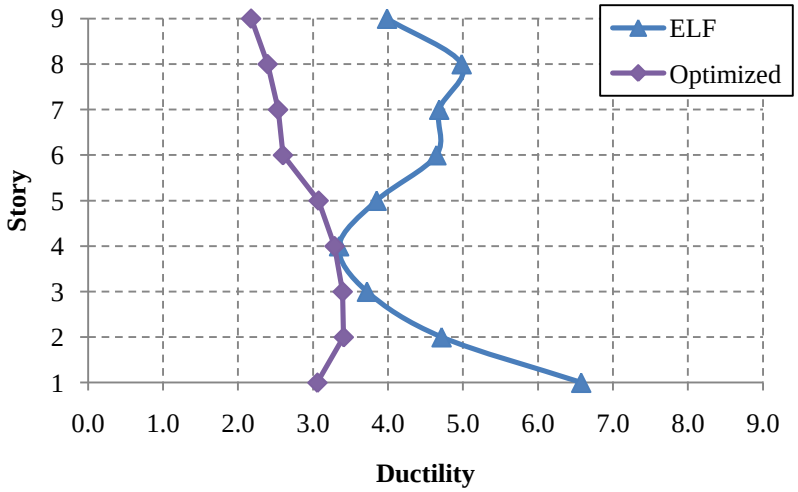

Figure 6-8. Ductility results for optimal 9-story design

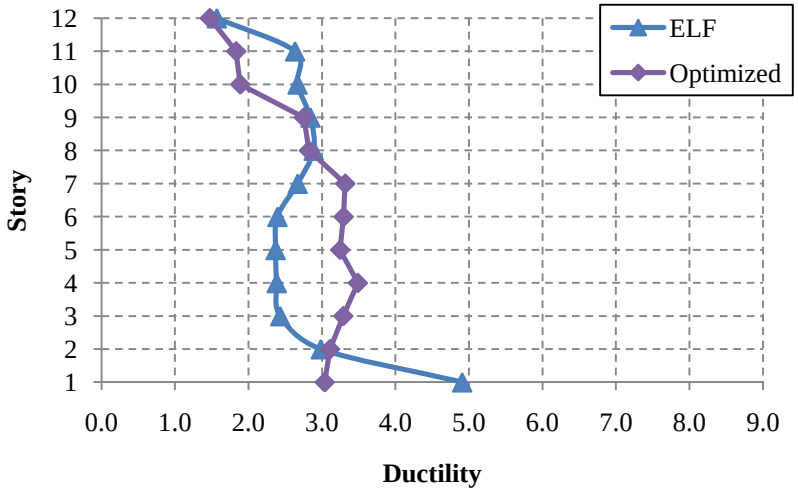

Figure 6-9. Ductility results for optimal 12-story design

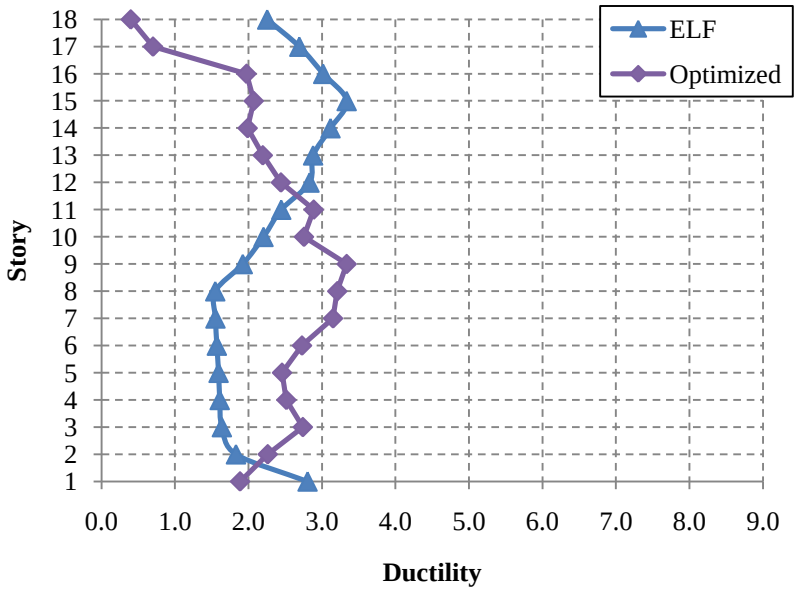

Figure 6-10. Ductility results for optimal 18-story design

As previously mentioned, the optimal designs exhibited trends where the intermediate stories had sizes similar to the average brace area. Meanwhile, the brace sizes at the upper stories decreased more rapidly and the first floor brace was much larger than those in the rest of the building. The lower story structures approached their governing constraints more consistently than their taller counterparts. In fact, the 18-story optimal design was relatively close to the limit at only one story. If the ductility constraint was reached, that particular member was assumed to be "fully stressed." The ductility plots show that none of the optimal designs, above three stories, were fully stressed.

A possible explanation for these trends at upper and lower floors may relate to this lack of a fully stressed optimal design in the taller structures. If the optimal designs were to become more fully stressed, the brace areas might demonstrate a more linear pattern that decreases up the structure as is seen with the three story design. However, fully

stressed designs might not exist due to the complex nature of the nonlinear dynamic time history analysis and the sample earthquake records which were used.

The impact of the beams and columns used in this study was not completely evaluated. The beams were the same at every floor but the columns did change at every odd floor. This change in columns could be assumed to explain the oscillation of brace size along the height of the building. However, a 9-story building model was optimized where the column sizes were the same throughout the model. This model produced a similar brace distribution, as seen in Figure 6-11, as the brace distribution with varying columns sizes, seen in Figure 6-3, signifying that the beam and column choice did not have a drastic impact on the results.

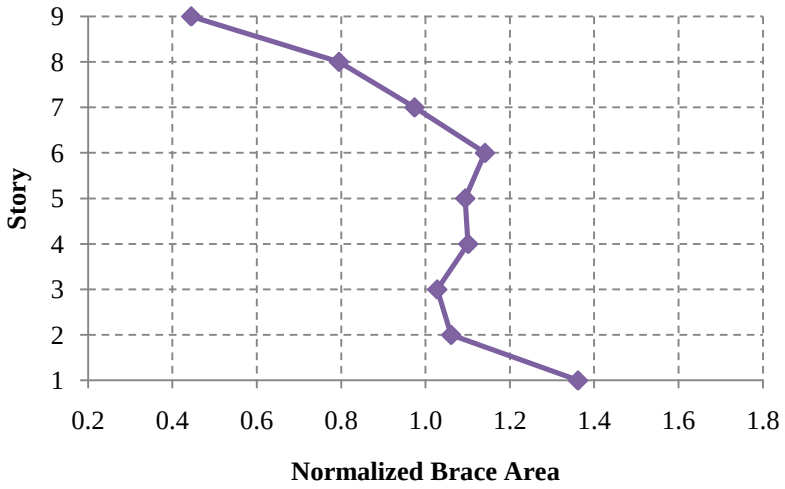

Figure 6-11. 9-story optimal normalized brace area with similar columns

As mentioned in the previous chapter, an alternate program was also created with a monotonically decreasing constraint. The 9-story model was implemented into the program and the results can be seen in Figure 6-12. The total area of the optimal design was found to be larger under the monotonic constraint than the optimal design when

allowing the braces areas to oscillate. In fact, the average brace area was 11.57 in^2 compared to 10.60 in^2 under the original conditions with a difference in total brace area of about 9 in^2 between the two designs. Therefore, the ELF assumption that the brace forces decrease monotonically on upper floors might not be correct; however, further research is needed to support this conclusion.

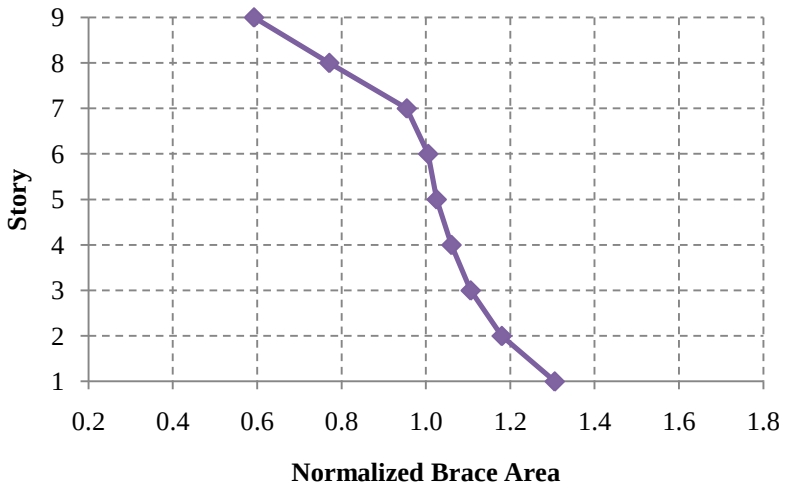

Figure 6-12. Monotonically decreasing 9-story optimal normalized brace area

Observations during the optimization process noted that the constraint limits were reached at the intermediate stories first and then the top and bottom stories later. This occurrence might indicate that the intermediate stories are the limiting factor during optimization and the most likely to violate the constraints. In other words, greater material savings were seen in making the intermediate stories fully stressed than other floors. Once the intermediate stories' constraints were met, the optimization process focused on the outer stories. As the convergence process continued, the optimal design seemed to approach a fully stressed design as much as possible, but in many cases that design was far from fully stressed.

One of the main considerations and limiting factors in nonlinear dynamic time history analysis combined with genetic optimization is the amount of time required to produce adequate results. A summary of the time required for each height building is shown in Table 6-2. As the building height increased, *ngener* increased to allow for adequate time for convergence. However, as *ngener* and building height increased, the time was lengthened dramatically. Eventually, this time constraint limited the amount of generations permitted on the taller structures. Parallel processing was not used but is available within the OpenSees framework and may be used for future studies.

Table 6-2. Time to Complete Optimization (hr)

Number of Stories	*ngener*	Completion Time
3	300	35
6	500	52
9	700	120
12	800	105
18	858	222

To ensure that convergence occurred during optimization, the braces areas of each design were recorded at each generation for the 9-story frame. The total area of the optimum design at each generation can be seen in Figure 6-13. The total brace area appears to level off after only a few initial generations. At the same location, where the number of earthquake records increases from 3 to full suite of 10, the total brace area increases due to the effect that the additional earthquakes had on the feasibility of the design set. Since the initial generations produced an optimal design rather rapidly, increasing the number of earthquakes sooner might reduce the number of required generations and, in turn, reduce the time for convergence.

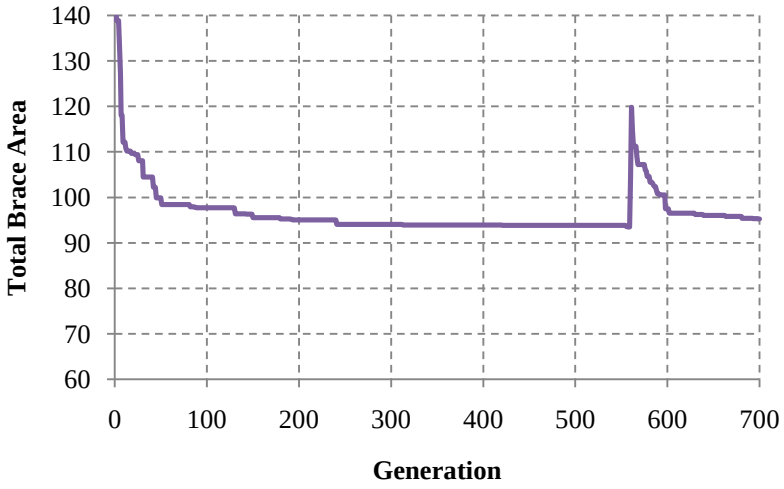

Figure 6-13. Total brace area of optimal design during optimization process

7 Conclusions

The ELF procedure and nonlinear dynamic time history analysis with optimization were used to obtain designs for 3-, 6-, 9-, 12-, and 18-story BRBF structures. Charts were presented indicating the optimal designs and the associating response of those designs in terms of ductility.

The brace distributions from the optimal designs and the ELF procedure were determined to be similar with a significant difference at the first story. In the taller structures, the brace areas decreased gradually along the intermediate stories while more rapidly increasing and decreasing at the first and upper stories, respectively. Overall, the designs were linear at the lower and intermediate stories and decreased more rapidly at the top. The much larger first floor brace area did not appear in the ELF procedure results. The higher average brace areas from the optimized results on the smaller structures indicate that the designs from the ELF procedures are inadequate since they do not meet ductility requirements.

The variation from linearity at the top and bottom of the optimized results may be due to the 1) lack of a fully stressed design at those stories, 2) an inadequate convergence to the true optimal design, or 3) the sensitivity of results to fixed sizes of beams and columns. Upon inspection, the sensitivity to fixed sizes of beams and columns did not appear to be an issue but further research is required. The ELF assumption of

monotonically decreasing forces and brace areas up the structure does not appear to be correct but further investigation is also required. The constraint limits that were used, indicating whether or not the member was fully stressed, were reached first at the intermediate stories and later on the upper and lower floors indicating that greater material savings can be found by forcing the intermediate stories to be fully stressed.

7.1 Possible Future Work

This research focused on a typical structure with an evenly distributed mass. The following is a list of areas that can be explored beyond the basic framework presented in this study and areas that require further research.

1. Investigate the complete sensitivity of fixed sizes of beams and columns on the overall brace distribution
2. Investigate the ELF assumption of monotonically decreasing lateral loads for taller structures
3. Incorporate building and loading irregularities
4. Extend to 3D spatial frames and allow for topology optimization techniques to eliminate possible braces
5. Evaluate the effects on the beams and columns (i.e. reactions and deformations) and optimize them from a standard shape set
6. Incorporate parallel processing to decrease total time for convergence and allow expansion to more complex studies

In the area of overall optimization research, Pavlovcic, L., Krajnc, A. and Beg, D. (2004) created a new objective function for the optimization procedure that incorporates

the self-manufacturing costs with the materials costs. They found that their new objective function produced results that were more efficient than the classical design approach but were similar to the volume optimization approach. They recommend that the more detailed cost function be used in topology optimization, complicated truss designs, or when deciding whether to use built up members or hot-rolled shapes since each of these cases would require a large amount of self-manufacturing costs. As research is expanded to include other aspects of BRBF design, the creation of a more complex objective function might prove beneficial.

References

Aiken, I. D., S. A. Mahin, and P. Uriz. "Large-scale testing of buckling-restrained braced frames." *Japan Passive Control Symposium.* Tokyo, Japan, 2002.

AISC. *AISC 341, Seismic provisions for structural steel buildings.* Chicago, IL: American Institute of Steel Construction Inc., 2005.

ASCE. *ASCE 7-05, Minimum design loads for buildings and other structures.* Reston, VA: American Society of Civil Engineering, 2005.

Asgarian, B., and N. Amirhesari. "A comparison of dynamic nonlinear behavior of ordinary and buckling restrained braced frames subjected to strong ground motion." *The Structural Design of Tall and Special Buildings* 17, no. 2 (2008): 367-386.

Balling, L. J. "Design of buckling-restrained braced frames using nonlinear time history analysis and optimization." *Master's Thesis.* Provo, UT: Brigham Young University, 2007.

Balling, R. J. *Computer Analysis and Optimization of Structures.* Provo, UT: BYU Academic Publishing, 2006.

Balling, R. J., L. J. Balling, and P. W. Richards. "Design of buckling restrained braced frames using nonlinear time history analysis and optimization." *Journal of Structural Engineering* 135, no. 5 (May 2009): 461-468.

Black, C. J., N. Makris, and I. D. Aiken. "Component testing, seismic evaluation and characterization of buckling-restrained braces." *Journal of Structural Engineering* 130, no. 6 (2004): 880-894.

Calado, L., and A. Brito. "Stress-Strain Relationship for Steel under Uniaxial Cyclic Loadings." *Advances in Structural Engineering* 5, no. 3 (2002): 145-151.

Coy, B. B. "Buckling-restrained brace connection design and testing." *Master's Thesis.* Provo, UT: Brigham Young University, 2007.

Fahnestock, L. A., J. M. Ricles, and R. Sause. "Experimental Evaluation of a Large-Scale Buckling-Restrained Braced Frame." *Journal of Structural Engineering* 133, no. 9 (2007): 1205-1214.

Fahnestock, L. A., R. Sause, and J. M. Ricles. "Seismic response and performance of buckling-restrained braced frames." *Journal of Structural Engineering* 133, no. 9 (2007): 1195-1204.

FEMA. *NEHRP recommended provisions for seismic regulations for new buildings and other structures (FEMA 450).* Washington, DC: Federal Emergency Management Agency, 2003.

Hayalioglu, M. S., and S. O. Degertekin. "Design of non-linear steel frames for stress and displacement constraints with semi-rigid connections via genetic optimization." *Structural and Multidisciplinary Optimization* 27 (2004): 259-271.

Iervolino, I., and C. A. Cornell. "Record selection for nonlinear seismic analysis of structures." *Earthquake Spectra* 21, no. 3 (2005): 685-713.

Kameshki, E. S., and M. P. Saka. "Genetic algorithm based optimum bracing design of non-swaying tall plane frames." *Journal of Constructional Steel Research* 57 (2001): 1081-1097.

Kim, J., and H. Choi. "Behavior and design of structures with buckling-restrained braces." *Engineering Structures* 26 (2004): 693-706.

Lagraos, N., M. Fragiadakis, M. Papadrakakis, and Y. Tsompanakis. "Structural optimization: a tool for evaluating seismic design procedures." *Engineering Structures* 28 (2006): 1623-1633.

Liu, M., S. A. Burns, and Y. K. Wen. "Optimal seismic design of steel frame buildings based on life cycle cost considerations." *earthquake Engineering and Structural Dynamics*, no. 32 (2003): 1313-1332.

Mazzoni, S., F. McKenna, M. H. Scott, and G. L. Fenves. *Open System for Earthquake Engineering Simulation User Command-Language Manual.* University of California, Berkeley: Pacific Engineering Research Center, 2006.

Memari, A. M., and M. Madhkhan. "Optimal design of steel frames subject to gravity and seismic codes' prescribed lateral forces." *Stuctural Optimization* 18 (1999): 56-66.

Ohsaki, M., T. Kinoshita, and P. Pan. "Multiobjective heuristic approaches to seismic design of steel frames with standard sections." *Earthquake Engineering and Structural Dynamics* 36 (2007): 1481-1495.

Pavlovcic, L., A. Krajnc, and D. Beg. "Cost function analysis in the structural optimization of steel frames." *Structural and Multidisciplinary Optimzation* 28 (2004): 286-295.

Pezeshk, S., C. V. Camp, and D. Chen. "Design of nonlinear framed structures using genetic optimization." *Journal of Structural Engineering* 126, no. 3 (2000): 382-388.

Roeder, C. W., D. E. Lehman, and A. Christopulos. "Seismic performance of special concentrically braced frames with buckling restrained braces." *Proceedings of the 8th U.S. National Conference on Earthquake Engineering.* San Fransisco, California, 2006.

Sabelli, R., and W. Pottebaum. "Design of a buckling-restrained braced frame utilizing 2005 seismic standards." *Proceedings of the Structures Congress and Exposition, Metropolis and Beyond - Proceedings of the 2005 Structures Congress and the 2005 Forensic Engineering Symposium.* 2005. 1807-1818.

Sabelli, R., S. Mahin, and C. Chang. "Seismic demands on steel braced frame buildings with buckling-restrained braces." *Engineering Structures* 25 (2003): 655-666.

Takeuchi, T., M. Ida, S. Yamada, and K. Suzuki. "Estimation of cumulative deformation capacity of buckling restrained braces." *Journal of Structural Engineering* 134, no. 5 (2008): 822-831.

Tsai, K. C., Y. Weng, M. Lin, C. Chen, J. Lai, and P. Hsiao. "Psuedo Dynamic Tests of a Full-Scale CFT/BRB Composite Frame: Displacement Based Seismic Design and Response Evaluations." *Proceedings of the International Workshop on Steel and Concrete Composite Construction (IWSCCC-2003).* Taipei, Taiwan, 2003. 165-176.

Watanabe, A., and H. Nakamura. "Study of the behavior of buildings using steel with low yield point." *Earthquake Engineering, Tenth World Conference.* Balkema, Rotterdam, 1992. 4465-4468.

Watanabe, A., Y. Hitomi, E. Saeki, A. Wada, and M. Fujimoto. "Properties of brace encased in buckling-restraining concrete and steel tube." *Proceedings of Ninth World Conference on Earthquake Engineering.* Tokyo-Kyoto, Japan, 1988. 719-724.

Appendix A Source Code

The following pages contain the source code for the optimization procedure and the input for OpenSees. Each section relates to a different .tcl file, which contains a set of procedures that form the program. The script may be copied directly into Notepad and saved as a .tcl file using the heading name (i.e. Main.tcl). When the Main.tcl file is run in OpenSees, all the other files will be accessed at the correct time.

A.1 Main

```
# Main algorithm

global nddv ncdv nobj nsize ngener ndesign dmin dmax cmin cmax designtype
global ddv cdv index obj feasible fitness dalpha calpha DBL_MAX DBL_MIN
global dataDir NBay NumStories dup FileName AvgDrift Ductility MassNode period

source Input.v7.tcl
source AppInitialize.v6.1.6.tcl
source Random_Selection_Crossover_Mutation.tcl
source Maximum_Fitness.tcl
source AppAnalysis.v5.2.tcl
source Gravity_Analysis.v4.1.tcl
source Dynamic_Analysis.v5.tcl
source EqScaling.tcl

Input

file mkdir Results/
set day [clock format [clock seconds] -format "%b%d %H%M"]
if [catch {open "Results/$FileName $day $designtype.out" w 0666} results] {
    puts stderr "Cannot open $results"
} else {
    puts $results "$FileName $day $designtype"
    set time [clock format [clock seconds] -format "%D %T%p %Z"]
    puts $results "Start_time: $time";     # prints the current time
    close $results
}

set DesignAvePrev [expr 1.0e-10]
set l 0; # counter for consecutive converging generations
```

```
for {set i 1} {$i <= $ngener} {incr i} {

    puts ""
    puts "Generation $i is running"

    set dalpha [expr pow(1-($i-1.0)/$ngener,$dmutpar)]
    set calpha [expr pow(1-($i-1.0)/$ngener,$cmutpar)]
    puts "$calpha $i $ngener $cmutpar"

    # Starting generation

    if {$i == 1} {
        if {$designtype == "ELF"} {
            ELF $ii; # go to this procedure to inout single case brace values
                # this can be used to analyze any design once
        } else {
            for {set j 1} {$j <= $nsize} {incr j} {
                for {set k 1} {$k <= $nddv} {incr k} {
                    set random [randint $dmin($k) $dmax($k)]
                    set ddv($j,$k) $random
                }
                for {set k 1} {$k <= $ncdv} {incr k} {
                    set random [randreal $cmin($k) $cmax($k)]
                    set cdv($j,$k) $random
                }
            }
        }
        for {set j 1} {$j <= [expr 2*$nsize]} {incr j} {
            set index($j) $j
        }
        set ndesign $nsize
        set firstnew 1
        EqScaling

    # Child generation

    } else {
        set k $nsize
        for {set j 1} {$j <= [expr $nsize/2]} {incr j} {
            set child1 $index([expr $k+1])
            set child2 $index([expr $k+2])
            GASelection $child1
            GASelection $child2
            GACrossover $child1 $child2
            GAMutation $child1
            GAMutation $child2
            incr k 2
        }
        set ndesign $k
        set firstnew [expr ($nsize+1)]
    }

    # Analyze Generation

        # Analyze the 50 best deisigns from the first 75% of ngener under all the earthquakes

        if {$i == [expr 0.80*$ngener]} {
            for {set j 1} {$j <= $nsize} {incr j} {
                puts "  Generation $i Design $j"
                set ii $index($j)
                for {set level 1} {$level <= $NumStories} {incr level} {
                    puts "     $level    $cdv($ii,$level)"
                }
                AppInitialize $ii
```

```
                AppAnalysis $ii $i
            }
        }

        for {set j $firstnew} {$j <= $ndesign} {incr j} {
            puts "  Generation $i Design $j"
            GADupAnalysis $j
            if {$dup == 0.0} {
                set ii $index($j)
                for {set level 1} {$level <= $NumStories} {incr level} {
                    puts "     $level    $cdv($ii,$level)"
                }
                AppInitialize $ii
                AppAnalysis $ii $i
            }
        }

    # Evaluate Fitness and Sort

        puts ""
        puts "Evaluating Fitness and Sorting of Generation $i"
        GAMaximumFitness
        GAEliteSort

    # Check Convergence Criteria - may not be correct - not used
        # set ConvLimit 0.01;       # percentage limit for convergence between designs
        # set NumGen4Conv 4;   # number of consecutive generations to determine convergence
        set DesignTotal 0.0
        for {set j 1} {$j <= $nsize} {incr j} {
            set ii $index($j)
            set DesignTotal [expr $DesignTotal+$obj($ii,1)]
        }
        set DesignAve [expr $DesignTotal/$nsize]
        set Change [expr abs($DesignAve/$DesignAvePrev-1)]
        set Change [format "%1.6f" $Change]
        puts "     Change from last generation is [expr $Change*100]%"
        #if {$Change <= $ConvLimit && $i >= [expr $ngener*0.8]} {
        #    incr l
        #} else {
        #    set l 0
        #}
        #if {$l >= $NumGen4Conv && $i >= [expr $ngener*0.8]} {break}
        set DesignAvePrev $DesignAve
        #puts "     $l Consecutive Generations with a change of less than [expr $ConvLimit*100]%"

        # Output file for design information (recomposed at each generation)

        if [catch {open "Results/$FileName.out" w 0666} results] {
            puts stderr "Cannot open $results"
        } else {
            set time [clock format [clock seconds] -format "%T%p %Z"]
            puts $results "Time: $time";                        # prints the current time
            puts $results "Generation $i Tot_Generations $ngener Mass: [expr 2*$MassNode]"
            puts $results ""
            for {set l 1} {$l <= $nsize} {incr l} {
                set ii $index($l);              # this references the right values and output them in the right order
                if {$nddv > 0} {puts -nonewline $results "output ddv values"}
                if {$ncdv > 0} {
                    puts $results "Design_Rank Fitness Feasibility Total_Brace_Area Period_Mode_1
Constraints_Files"
                    for {set k 1} {$k <= $nobj} {incr k} {
                        puts $results "$l $fitness($ii) $feasible($ii) $obj($ii,$k) $period($ii) $dataDir/$ii";
                            # this should output the design number, fitness, feasible,obj value
                            # and the location of the files used to determine feasibility
```

```
                    }
                    puts $results "Ground_Files: $GMfile"
                    puts $results "Scaling_Factor: $GMfact($ii)"
                    puts $results ""
                    puts $results "Level Brace_Area Max_Avg_Drift Ductility"
                    for {set level 1} {$level <= $NumStories} {incr level} {
                        puts $results "$level $cdv($ii,$level) $AvgDrift($ii,$level) $Ductility($ii,$level)";

                            #this section outputs to size of the brace on that level
                    }
                puts $results ""
                }
            }
            close $results
        }

        # Output Area information at each generation (Amended througout the optimization)

        if {$i == 1} {
            if [catch {open "Results/Areas $FileName.out" w 0666} results] {
                puts stderr "Cannot open $results"
            }
        }
        if [catch {open "Results/Areas $FileName.out" a 0666} results] {
            puts stderr "Cannot open $results"
        } else {
            if {$i == 1} {
                puts -nonewline $results "Generation ";
                for {set l 1} {$l <= $nsize} {incr l} {
                    puts -nonewline $results "Design_$l "
                }
                puts $results "Tot_Area Average_Total_Area Average_Story_Area_Opt_Design"
            }
            puts -nonewline $results "$i "
            for {set l 1} {$l <= $nsize} {incr l} {
                set ii $index($l)
                puts -nonewline $results "$obj($ii,1) "
            }
            set ii $index(1)
            set Area $obj($ii,1)
            set AvgArea [expr $Area/$ncdv]
            puts $results "$DesignTotal $DesignAve $AvgArea"
            close $results
        }
}

# Postprocess

puts "Postprocessing..."

if [catch {open "Results/$FileName $day $designtype.out" a 0666} results] {
    puts stderr "Cannot open $results"
} else {
    set time [clock format [clock seconds] -format "%D %T%p %Z"]
    puts $results "End_time: $time";                    # prints the current time
    puts $results ""
    puts $results "Stories Bays Generations_Run Generations_Total Generation_Size Tournament_Size
Mass_Floor_(k-s^2/in) Bay_Height Bay_Width Bays_In Bays_Along";
    puts $results "$NumStories $NBay [expr $i-1] $ngener $nsize $ntourn [expr 2*$MassNode] $BayH
$BayW $BaysIn $BaysAlong"
    puts $results "Crossover_Probability Mutation_Probability Cont_Cross_Parameter Cont_Mut_Parameter"
    puts $results "$probcross $probmutate $ccrosspar $cmutpar"
    puts $results ""
    for {set i 1} {$i <= $nsize} {incr i} {
```

```
        set ii $index($i);              # this references the right values and output them in the right order
        if {$nddv > 0} {puts -nonewline $results "output ddv values"}
        if {$ncdv > 0} {
            puts $results "Design_Rank Fitness Feasibility Total_Brace_Area Period_Mode_1
Constraints_Files"
            for {set k 1} {$k <= $nobj} {incr k} {
                puts $results "$i $fitness($ii) $feasible($ii) $obj($ii,$k) $period($ii) $dataDir/$ii";
                    # this should output the design number, fitness, feasible,obj value
                    # and the location of the files used to determine feasibility
            }
            puts $results "Ground_Files: $GMfile"
            puts $results "Scaling_Factor: $GMfact($ii)"
            puts $results ""
            puts $results "Level Brace_Area Max_Avg_Drift Ductility"
            for {set level 1} {$level <= $NumStories} {incr level} {
                puts $results "$level $cdv($ii,$level) $AvgDrift($ii,$level) $Ductility($ii,$level)";
                    #this section outputs to size of the brace on that level
            }
        puts $results ""
        }
    }
    close $results
}

puts "DONE WITH OPTIMIZATION!"
```

A.2 Random_Selection_Crossover_Mutation

```
# Random Integer procedure - returns a random integer
# check if that will return any number in the series
proc randint {min max} {
    set range [expr $max-$min]
    set randint [expr int(rand()*$range+$min)]

}

#-------------------------------------------------------------------------------------------------
# Random Real Number - returns a random number between a range
proc randreal {min max} {
    set range [expr $max-$min]
    set randreal [expr (rand()*$range)+$min]
}

#-------------------------------------------------------------------------------------------------
# Tournament selection
# ntourn = tournament size; nsize = generation size; nddv = number of discrete design variable
# ncdv = number of continuous design variables; fitness = array of the fitnesses of each design
# child = design number of the child
# index = index number for each design
# ddv = array of each discrete design variable in each design; cdv = same as ddv but continous

proc GASelection {child} {

    global nddv ncdv nsize ntourn ddv cdv index fitness DBL_MAX

    set minfit $DBL_MAX

    for {set i 1} {$i <= $ntourn} {incr i} {
        set randNum [randint 1.0 $nsize];    # deterimines a random integer
        # puts $randNum; # check to ensure that random integer generator is working
        set candidate $index($randNum);          # picks the design that is to be the parent
        if {$fitness($candidate) <= $minfit} {;     # checks to see if the candidate is more fit
```

```
            set minfit $fitness($candidate)
            set parent $candidate
        }
    }
    for {set i 1} {$i <= $nddv} {incr i} {    ;    # makes the child's ddv same as the parent
        set ddv($child,$i) $ddv($parent,$i)
    }
    for {set i 1} {$i <= $ncdv} {incr i} {    ;    # makes the child's ddv same as the parent
        set cdv($child,$i) $cdv($parent,$i)
    }
}

#-------------------------------------------------------------------------------------------------
# Uniform and Blend Crossover
# probcross = crossover probability; dcrosspar = discrete crossover probability;
# ccrosspar = continuous crossover probabilty; nddv = number of discrete design variables
# ncdv = number of continuous design variables; child1 = design number of first child
# child2 = design number of second child; ddv and cdv are same as in tournament selection

proc GACrossover {child1 child2} {

    global nddv ncdv probcross dcrosspar ccrosspar ddv cdv

    if {[expr rand()] < $probcross} {
        # puts "    Crossover occuring..."
        for {set i 1} {$i <= $nddv} {incr i} {
            set r [expr rand()]
            if {$r <= 0.50} {
                set a 0.0
                if {$dcrosspar>0.0} {
                    set a [expr (pow((2*$r),(1/$dcrosspar))/2)]
                }
            } else {
                set a 1
                if {$dcrosspar>0.0} {
                    set a [expr 1-(pow((2-(2*$r)),(1/$dcrosspar))/2)]
                }
            }
            set x1 $ddv($child1,$i)
            set x2 $ddv($child2,$i)
            if {$x2 >= $x1} {
                set x1 [expr $x1+0.00001]
                set x2 [expr $x2+0.99999]
            } else {
                set x2 [expr $x2+0.00001]
                set x1 [expr $x1+0.99999]
            }
            set ddv($child1,$i) [expr $a*$x1+(1.0-$a)*$x2]
            set ddv($child2,$i) [expr (1.0-$a)*$x1+$a*$x2]
        }
        for {set i 1} {$i <= $ncdv} {incr i} {
            set r [expr rand()]
            if {$r <= 0.5} {
                set a 0.0
                if {$ccrosspar>0.0} {set a [expr (pow((2.0*$r),(1.0/$ccrosspar))/2.0)]}
            } else {
                set a 1.0
                if {$ccrosspar>0.0} {set a [expr 1.0-(pow((2.0-2.0*$r),(1.0/$ccrosspar))/2.0)]}
            }
            set x1 $cdv($child1,$i)
            set x2 $cdv($child2,$i)
            # puts "$x1 $x2"
            set cdv($child1,$i) [expr $a*$x1+(1.0-$a)*$x2]
            set cdv($child2,$i) [expr (1.0-$a)*$x1+$a*$x2]
```

```
            # puts "$cdv($child1,$i) $cdv($child2,$i)"
        }
    }
}

#-------------------------------------------------------------------------------------------------
# Uniform and Dynamic Mutation
# probmutate = mutation probability; dalpha = discrete uniformity exponent; calpha = continuous uniformity
exponent
# nddv = number of discrete design variables; dmin[nddv] & dmax[nddv] = minimum and maximum value for
each discrete design variable
# ncdv = number of continuous design variables; cmin[nddv] & cmax[max] = minimum and maximum value
for each continuous design variable
# child = design number for child

proc GAMutation {child} {

    global nddv ncdv probmutate dmutpar cmutpar dmin dmax cmin cmax ddv cdv dalpha calpha

    for {set i 1} {$i <= $nddv} {incr i} {
        if {[expr rand()] < $probmutate} {
            set x $ddv($child,$i)
            set ymin [expr $dmin($i) - 0.49999]
            set ymax [expr $dmax($i) + 0.49999]
            set r [randreal $ymin $ymax]
            if {$r <= $x} {
                set ddv($child,$i) [expr $dmin($i)+pow(($r-$dmin($i)+0.5),$dalpha)*pow(($x-
$dmin($i)+0.5),(1-$dalpha))]
            } else {
                set ddv($child,$i) [expr $dmax($i)+1-pow(($dmax($i)+0.5-$r),$dalpha)*pow(($dmax($i)+0.5-
$x),(1-$dalpha))]
            }
        }
    }
    for {set i 1} {$i <= $ncdv} {incr i} {
        if {[expr rand()] < $probmutate} {
            # puts "     Mutating..."
            set x $cdv($child,$i)
            set r [randreal $cmin($i) $cmax($i)]
            if {$r <= $x} {
                set cdv($child,$i) [expr $cmin($i)+pow(($r-$cmin($i)),$calpha)*pow(($x-$cmin($i)),(1-
$calpha))]
            } else {
                set cdv($child,$i) [expr $cmax($i)-pow(($cmax($i)-$r),$calpha)*pow(($cmax($i)-$x),(1-
$calpha))]
            }
        }
    }
}

#-------------------------------------------------------------------------------------------------------------------------------
-------
# Analysis of Duplicate Design
# i = design number; index[ndesign] = index number of each design
# ncdv = number of continuous design variables; cdv[ndesign][ncdv] = value of each continuous variable in
each design
# nobj = number of objective functions
# updated obj[ndesign][nobj] = value of each objective function in each design; updated feasible[ndesign] =
value of feasibility of each design
# returned dup = flag indicating design is a duplicate

proc GADupAnalysis {i} {

    global nobj index obj feasible fitness NumStories dup
```

```
    set ii $index($i)
    set dup 0
    set j 0
    while {$dup == 0 && $j<[expr $i-1.0]} {
        incr j 1
        set jj $index($j)
        GADupValue $ii $jj
        if {$dup == 1} {puts "    Same as design $j"}
    }
    if {$dup == 1} {
        for {set k 1} {$k <= $nobj} {incr k} {
            set obj($ii,$k) $obj($jj,$k)
        }
        set feasible($ii) $feasible($jj)
        DupResults $ii $jj
    }
}

#-------------------------------------------------------------------------------------------------------------------------------------------
-----
# Duplicate results for duplicate design
# i = design number; j = previously analyzed duplicate design number

proc DupResults {i j} {

    global NumStories Ductility AvgDrift GMfact period

    set period($i) $period($j)
    for {set level 1} {$level <= $NumStories} {incr level} {  ; # makes the Drift and Ductility values the same
for this design
        set AvgDrift($i,$level) $AvgDrift($j,$level)
        set Ductility($i,$level) $Ductility($j,$level)
        set GMfact($i) $GMfact($j)
    }
}

#-------------------------------------------------------------------------------------------------------------------------------------------
------
# Check if Design i and Design j have Duplicate Values
# i,j = design index numbers; nddv = number of discrete design variables; ddv[ndesign][nddv] = value of
each discrete variable in each design
# ncdv = number of continuous design variables; cdv[ndesign][ncdv] = value of each continuous variable in
each design; dup = flag indicating design is a duplicate

proc GADupValue {i j} {

    global nddv ncdv ddv cdv dup

    set dup 1
    set k 0
    while {$dup == 1 && $k < $nddv} {
        incr k 1
        if {$ddv($i,$k) != $ddv($j,$k)} {set dup 0}
    }
    set k 0
    while {$dup == 1 && $k < $ncdv} {
        incr k 1
        if {[format "%1.1f" $cdv($i,$k)] != [format "%1.1f" $cdv($j,$k)]} {set dup 0}
    }
}

#-------------------------------------------------------------------------------------------------------------------------------------------
-------
```

```
# rotSpring2D.tcl
# SETS A MULTIPOINT CONSTRAINT ON THE TRANSLATIONAL DEGREES OF FREEDOM,
# SO DO NOT USE THIS PROCEDURE IF THERE ARE TRANSLATIONAL ZEROLENGTH
# ELEMENTS ALSO BEING USED BETWEEN THESE TWO NODES
#
# Written: MHS
# Date: Jan 2000
#
# Formal arguments
    # eleID - unique element ID for this zero length rotational spring
    # nodeR - node ID which will be retained by the multi-point constraint
    # nodeC - node ID which will be constrained by the multi-point constraint
    # matID - material ID which represents the moment-rotation relationship
    # for the spring

proc rotSpring2D {eleID nodeR nodeC matID} {
    # Create the zero length element
        element zeroLength $eleID $nodeR $nodeC -mat $matID -dir 6

    # Constrain the translational DOF with a multi-point constraint
    # retained constrained DOF_1 DOF_2 ... DOF_n
        equalDOF $nodeR $nodeC 1 2
}

#-----------------------------------------------------------------------------------------------------------------------------------
---------

# Wsection.tcl: tcl procedure for creating a wide flange steel fiber section
# written: Remo M. de Souza
# date: 06/99
# modified: 08/99 (according to the new general modelbuilder)
# input parameters
# secID - section ID number
# matID - material ID number
# d = nominal depth
# tw = web thickness
# bf = flange width
# tf = flange thickness
# nfdw = number of fibers along web depth
# nftw = number of fibers along web thickness
# nfbf = number of fibers along flange width
# nftf = number of fibers along flange thickness

proc Wsection { secID matID d tw bf tf nfdw nftw nfbf nftf} {
    set dw [expr $d - 2 * $tf]
    set y1 [expr -$d/2]
    set y2 [expr -$dw/2]
    set y3 [expr $dw/2]
    set y4 [expr $d/2]
    set z1 [expr -$bf/2]
    set z2 [expr -$tw/2]
    set z3 [expr $tw/2]
    set z4 [expr $bf/2]

    section fiberSec $secID {
        # nflJ nfJK yI zI yJ zJ yK zK yL zL
        patch quadr $matID $nfbf $nftf $y1 $z4 $y1 $z1 $y2 $z1 $y2 $z4
        patch quadr $matID $nftw $nfdw $y2 $z3 $y2 $z2 $y3 $z2 $y3 $z3
        patch quadr $matID $nfbf $nftf $y3 $z4 $y3 $z1 $y4 $z1 $y4 $z4
    }
}
```

A.3 Maximum_Fitness

```
# Segregated Maximum Fitness
# ndesign = number of designs; nobj = number of objective functions; obj[ndesign][nobj] = value of objective function in each design
# feasible[ndesign] = value of feasibility of each design; returned fitness[ndesign] = value of fitness of each design

proc GAMaximumFitness {} {

    global nobj ndesign obj feasible fitness DBL_MAX

    set maxfit 0.0
    for {set i 1} {$i <= $ndesign} {incr i} {
        if {$feasible($i) == 0.0} {
            set maxvalue -$DBL_MAX
            for {set j 1} {$j <= $ndesign} {incr j 1} {
                if {$feasible($j) == 0.0 && $j != $i} {
                    set minvalue $DBL_MAX
                    for {set k 1} {$k <= $nobj} {incr k} {
                        set value [expr ($obj($i,$k)-$obj($j,$k))]
                        if {$value < $minvalue} {set minvalue $value}
                    }
                    if {$minvalue > $maxvalue} {set maxvalue $minvalue}
                }
            }
            set fitness($i) $maxvalue
            if {$maxvalue > $maxfit} {set maxfit $maxvalue}
        }
    }
    for {set i 1} {$i <= $ndesign} {incr i} {
        if {$feasible($i) > 0.0} {set fitness($i) [expr $maxfit + $feasible($i)]}
    }
}

#-------------------------------------------------------------------------------------------------------------------------------------------
---------
# Elitism Sort
# ndesign = number of designs; nsize = generation size; fitness[ndesign] = value of fitness for each design; index[ndesign] = index number of each design

proc GAEliteSort {} {

    global nsize ndesign index fitness DBL_MAX

    for {set i 1} {$i <= $nsize} {incr i} {
        set minfit $DBL_MAX
        for {set j 1} {[expr $ndesign-($j-1)] >= $i} {incr j 1} {
            set num [expr $ndesign-($j-1)]
            set jj $index($num)
            if {$fitness($jj) <= $minfit} {
                set minfit $fitness($jj)
                set k $num
            }
        }
        if {$k != $i} {
            set ii $index($i)
            set index($i) $index($k)
            set index($k) $ii
        }
    }
}
```

A.4 DisplayPlane_DisplayModel2D

```
proc DisplayPlane {ShapeType dAmp viewPlane {nEigen 0}  {quadrant 0}} {
    ##############################################################################################
###
    ## DisplayPlane $ShapeType $dAmp $viewPlane $nEigen $quadrant
    ##############################################################################################
###
    ## setup display parameters for specified viewPlane and display
    ##            Silvia Mazzoni & Frank McKenna, 2006
    ##
    ##  ShapeType :   type of shape to display. # options: ModeShape , NodeNumbers , DeformedShape
    ##  dAmp :        relative amplification factor for deformations
    ##  viewPlane :   set local xy axes in global coordinates (XY,YX,XZ,ZX,YZ,ZY)
    ##  nEigen :      if nEigen not=0, show mode shape for nEigen eigenvalue
    ##  quadrant:     quadrant where to show this figure (0=full figure)
    ##
    ##############################################################################################
###

    set Xmin [lindex [nodeBounds] 0];    # view bounds in global coords -  will add padding on the sides
    set Ymin [lindex [nodeBounds] 1];
    set Zmin [lindex [nodeBounds] 2];
    set Xmax [lindex [nodeBounds] 3];
    set Ymax [lindex [nodeBounds] 4];
    set Zmax [lindex [nodeBounds] 5];

    set Xo 0;   # center of local viewing system
    set Yo 0;
    set Zo 0;

    set uLocal [string index $viewPlane 0];  # viewPlane local-x axis in global coordinates
    set vLocal [string index $viewPlane 1];  # viewPlane local-y axis in global coordinates


    if  {$viewPlane =="3D" } {
        set uMin $Zmin+$Xmin
        set uMax $Zmax+$Xmax
        set vMin $Ymin
        set vMax $Ymax
        set wMin -10000
        set wMax 10000
        vup 0 1 0; # dirn defining up direction of view plane
    } else {
        set keyAxisMin "X $Xmin Y $Ymin Z $Zmin"
        set keyAxisMax "X $Xmax Y $Ymax Z $Zmax"
        set axisU [string index $viewPlane 0];
        set axisV [string index $viewPlane 1];
        set uMin [string map $keyAxisMin $axisU]
        set uMax [string map $keyAxisMax $axisU]
        set vMin [string map $keyAxisMin $axisV]
        set vMax [string map $keyAxisMax $axisV]
        if {$viewPlane =="YZ" || $viewPlane =="ZY" } {
            set wMin $Xmin
            set wMax $Xmax
        } elseif  {$viewPlane =="XY" || $viewPlane =="YX" } {
            set wMin $Zmin
            set wMax $Zmax
        } elseif  {$viewPlane =="XZ" || $viewPlane =="ZX" } {
            set wMin $Ymin
            set wMax $Ymax
        } else {
        return -1
```

```
        }
    }

    set epsilon 1e-3;  # make windows width or height not zero when the Max and Min values of a
coordinate are the same

    set uWide [expr $uMax - $uMin+$epsilon];
    set vWide [expr $vMax - $vMin+$epsilon];
    set uSide [expr 0.25*$uWide];
    set vSide [expr 0.25*$vWide];
    set uMin [expr $uMin - $uSide];
    set uMax [expr $uMax + $uSide];
    set vMin [expr $vMin - $vSide];
    set vMax [expr $vMax + 2*$vSide];  # pad a little more on top, because of window title
    set uWide [expr $uMax - $uMin+$epsilon];
    set vWide [expr $vMax - $vMin+$epsilon];
    set uMid [expr ($uMin+$uMax)/2];
    set vMid [expr ($vMin+$vMax)/2];

    # keep the following general, as change the X and Y and Z for each view plane
    # next three commmands define viewing system, all values in global coords
    vrp $Xo $Yo $Zo;    # point on the view plane in global coord, center of local viewing system
    if {$vLocal == "X"} {
        vup 1 0 0; # dirn defining up direction of view plane
    } elseif {$vLocal == "Y"} {
        vup 0 1 0; # dirn defining up direction of view plane
    } elseif {$vLocal == "Z"} {
        vup 0 0 1; # dirn defining up direction of view plane
    }
    if {$viewPlane =="YZ" } {
        vpn 1 0 0; # direction of outward normal to view plane
        prp 10000. $uMid $vMid ; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="ZY" } {
        vpn -1 0 0; # direction of outward normal to view plane
        prp -10000. $vMid $uMid ; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="XY"  } {
        vpn 0 0 1; # direction of outward normal to view plane
        prp $uMid $vMid 10000; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="YX" } {
        vpn 0 0 -1; # direction of outward normal to view plane
        prp $uMid $vMid -10000; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="XZ" } {
        vpn 0 -1 0; # direction of outward normal to view plane
        prp $uMid -10000 $vMid ; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="ZX" } {
        vpn 0 1 0; # direction of outward normal to view plane
        prp $uMid 10000 $vMid ; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    } elseif  {$viewPlane =="3D" } {
        vpn 1 0.25 1.25; # direction of outward normal to view plane
        prp -100 $vMid 10000; # eye location in local coord sys defined by viewing system
        plane 10000 -10000; # distance to front and back clipping planes from eye
    }  else {
        return -1
    }
    # next three commands define view, all values in local coord system
    if  {$viewPlane =="3D" } {
        viewWindow [expr $uMin-$uWide/4] [expr $uMax/2] [expr $vMin-0.25*$vWide] [expr $vMax]
    } else {
```

```
        viewWindow $uMin $uMax $vMin $vMax
    }
    projection 1;   # projection mode, 0:prespective, 1: parallel
    fill 1;         # fill mode; needed only for solid elements

    if {$quadrant == 0} {
        port -1 1 -1 1  # area of window that will be drawn into (uMin,uMax,vMin,vMax);
    } elseif {$quadrant == 1} {
        port 0 1 0 1    # area of window that will be drawn into (uMin,uMax,vMin,vMax);
    } elseif {$quadrant == 2} {
        port -1 0 0 1   # area of window that will be drawn into (uMin,uMax,vMin,vMax);
    } elseif {$quadrant == 3} {
        port -1 0 -1 0  # area of window that will be drawn into (uMin,uMax,vMin,vMax);
    } elseif {$quadrant == 4} {
        port 0 1 -1 0   # area of window that will be drawn into (uMin,uMax,vMin,vMax);
    }

    if {$ShapeType ==  "ModeShape" } {
        display -$nEigen 0  [expr 5.*$dAmp];    # display mode shape for mode $nEigen
    } elseif  {$ShapeType ==  "NodeNumbers" } {
        display 1 -1 0  ;         # display node numbers
    } elseif  {$ShapeType ==  "DeformedShape" }  {
        display 1 2 $dAmp;        # display deformed shape  the 2 makes the nodes small
    }
};
                                                                                                    #
##########################################################################################################

proc DisplayModel2D { {ShapeType nill} {dAmp 5}  {xLoc 10} {yLoc 10} {xPixels 512} {yPixels 384} {nEigen
1} } {
    ##########################################################################################################
###
    ## DisplayModel2D $ShapeType $dAmp $xLoc $yLoc $xPixels $yPixels $nEigen
    ##########################################################################################################
###
    ## display Node Numbers, Deformed or Mode Shape in 2D problem
    ##            Silvia Mazzoni & Frank McKenna, 2006
    ##
    ##  ShapeType :   type of shape to display. # options: ModeShape , NodeNumbers , DeformedShape
    ##  dAmp :        relative amplification factor for deformations
    ##  xLoc,yLoc  :    horizontal & vertical location in pixels of graphical window (0,0=upper left-most corner)
    ##  xPixels,yPixels :   width & height of graphical window in pixels
    ##  nEigen :      if nEigen not=0, show mode shape for nEigen eigenvalue
    ##
    ##########################################################################################################
####
    global TunitTXT;                    # load time-unit text
    global ScreenResolutionX ScreenResolutionY; # read global values for screen resolution

    if {  [info exists TunitTXT] != 1} {set TunitTXT ""};       # set blank if it has not been defined previously.

    if {  [info exists ScreenResolutionX] != 1} {set ScreenResolutionX 1024};      # set default if it has not
been defined previously.
    if {  [info exists ScreenResolutionY] != 1} {set ScreenResolutionY 768};       # set default if it has not
been defined previously.

    if {$xPixels == 0} {
        set xPixels [expr int($ScreenResolutionX/2)];
        set yPixels [expr int($ScreenResolutionY/2)]
        set xLoc 10
        set yLoc 10
    }
    if {$ShapeType == "nill"} {
        puts ""; puts ""; puts "------------------"
```

```
        puts "View the Model? (N)odes, (D)eformedShape, anyMode(1),(2),(#). Press enter for NO."
        gets stdin answer
        if {[llength $answer]>0 } {
            if {$answer != "N" & $answer != "n"} {
                puts "Modify View Scaling Factor=$dAmp? Type factor, or press enter for NO."
                gets stdin answerdAmp
                if {[llength $answerdAmp]>0 } {
                    set dAmp $answerdAmp
                }
            }
            if {[string index $answer 0] == "N" || [string index $answer 0] == "n"} {
                set ShapeType NodeNumbers
            } elseif {[string index $answer 0] == "D" ||[string index $answer 0] == "d" } {
                set ShapeType DeformedShape
            } else {
                set ShapeType ModeShape
                set nEigen $answer
            }
        } else {
            return
        }
    }

    if {$ShapeType ==  "ModeShape" } {
        set lambdaN [eigen $nEigen];          # perform eigenvalue analysis for ModeShape
        set lambda [lindex $lambdaN [expr $nEigen-1]];
        set omega [expr pow($lambda,0.5)]
        set PI  [expr 2*asin(1.0)];      # define constant
        set Tperiod [expr 2*$PI/$omega];          # period (sec.)
        set fmt1 "Mode Shape, Mode=%.1i Period=%.3f %s  "
        set windowTitle [format $fmt1 $nEigen $Tperiod  $TunitTXT]
    } elseif  {$ShapeType ==  "NodeNumbers" } {
        set windowTitle "Node Numbers"
    } elseif  {$ShapeType ==  "DeformedShape" } {
        set windowTitle "Deformed Shape"
    }

    set viewPlane XY
    recorder display $windowTitle $xLoc $yLoc $xPixels $yPixels  -wipe ; # display recorder
    DisplayPlane $ShapeType $dAmp $viewPlane $nEigen 0
}
```

A.5 EqScaling

```
proc EqScaling {} {

    # This procedure assembles the design spectra and the spectra for
    # each specific earthquake
    #
    # Written: GTO
    # Date: October 2008
    #

    global Designfile GMfile GMdir

    # puts "Scaling EQ"
    set file $GMdir/$Designfile.txt
    ReadDesignSpectraFile $file;     # assembles array of design spectra

    set number 1
    foreach GM $GMfile {
        # puts $GM
```

```
        set file "$GMdir/$GM"
        append file "_050.txt"
        # puts $file
        ReadSpectraFile $file $number; # assembles array of earthquake spectra
        incr number
    }
    # puts "EQ Scaled"

}

# ------------------------------------------------------------------------------

proc Scale {per j l} {

    # This procedure looks at the response spectra for each earthquake
    # and scales that earthquake to the specific building design
    #
    # Written: GTO
    # Date: October 2008
    #
    # Formal arguments
    #    per -- period of design
    #    j -- list number for GMfact
    #    i -- design number


    global GMfact spectra spectranum designspectra designspectranum

    for {set i 1} {$i <= $spectranum($j)} {incr i} {
        # Find where the actual period value lands on spectra
        set x $spectra($j,$i)
        if {[lindex [split $x] 1] < $per} {break}
    }
    for {set k 1} {$k <= $designspectranum} {incr k} {
        # Find where actual period value lands on the design spectra
        set x $designspectra($k)
        if {[lindex [split $x] 1] > $per} {break}
    }
    # puts "$i $k"
    set design [expr ([lindex $designspectra($k) 0]+[lindex $designspectra([expr $k-1]) 0])/2]
    set actual [expr ([lindex $spectra($j,[expr $i-1]) 0]+[lindex $spectra($j,$i) 0])/2]
    lappend GMfact($l) [expr $design/$actual]
    # puts "$design $actual $GMfact($l)"
}

# ------------------------------------------------------------------------------------

proc ReadDesignSpectraFile {inFilename} {

    # A procedure which converts the response spectra from the PEER
    # strong motion database to a specific array.
    #
    # Written: GTO
    # Date: October 2008
    #
    # Formal arguments
    #    inFilename -- file which contains PEER strong motion record
    #    outArrayname -- file to be written in format .txt can read

    global designspectra designspectranum

    # Open the input file and catch the error if it can't be read
    if {[catch {open $inFilename r} inFileID]} {
        puts stderr "Cannot open $inFilename for reading"
```

```
    } else {

        # Flag indicating spectra values are found and that ground motion
        # values should be read -- ASSUMES spectra values are at end of file
        set flag 0
        set i 0
        # Look at each line in the file
        foreach line [split [read $inFileID] \n] {
            if {[llength $line] == 0} {
                # Blank line --> do nothing
                continue
            } else {
                split $line
                # Find the first line of data values and set the flag
                if {[string match [lindex $line 0] "0"] == 1} {set flag 1}
                if {$flag == 1} {
                    incr i
                    set designspectra($i) [lindex $line 1];          # first value is the acc
                    lappend designspectra($i) [lindex $line 0];  # second value is period
                }

            }
            # puts "$i $flag"
        }
        close $inFileID;    # Close the input file
        set designspectranum $i
    }
}

# ----------------------------------------------------------------------------------------------

proc ReadSpectraFile {inFilename j} {

    # A procedure which converts the response spectra from the PEER
    # strong motion database to an array containing the period and acc values.
    #
    # Written: GTO
    # Date: October 2008
    #
    # Formal arguments
    #    inFilename -- file which contains PEER strong motion record
    #    j -- earthquake number
    #
    # Assumptions
    #    The header in the PEER record is, e.g., formatted as follows:
    #     PACIFIC ENGINEERING AND ANALYSIS STRONG-MOTION DATA
    #      IMPERIAL VALLEY 10/15/79 2319, EL CENTRO ARRAY 6, 230
    #      ACCELERATION TIME HISTORY IN UNITS OF G
    #      NPTS=  3930, DT= .00500 SEC

    global spectra spectranum

    # Open the input file and catch the error if it can't be read
    if {[catch {open $inFilename r} inFileID]} {
        puts stderr "Cannot open $inFilename for reading"
    } else {

        # Flag indicating spectra values is found and that ground motion
        # values should be read -- ASSUMES spectra values after last line
        # of header!!!
        set flag 0
        set i 0
        # Look at each line in the file
        foreach line [split [read $inFileID] \n] {
```

```
            if {[llength $line] == 0} {
                # Blank line --> do nothing
                continue
            } else {
                split $line
                if {$flag == 1} {
                    incr i
                    set spectra($j,$i) [lindex $line 6];# first one is acc
                    lappend spectra($j,$i) [lindex $line 8];# second one is period
                    # puts $spectra($j,$i)
                }
                # Find the first line of data values and set the flag
                if {[string match [lindex $line 0] "1"] == 1} {set flag 1}

            }
            # puts "$i $flag"
        }
        close $inFileID;    # Close the input file
        set spectranum($j) $i
    }
}
```

A.6 AppInitialize.v6.1.6

```
# Initialize the program

proc AppInitialize {ii} {

    global nddv ncdv nobj nsize ngener ntourn probcross dcrosspar ccrosspar probmutate dmutpar cmutpar
dmin dmax cmin cmax
    global ddv cdv index obj feasible fitness dalpha calpha FileName dataDir GMdir BraceMat
    global in kip sec LuniTXT FuniTXT TuniTXT ft ksi psi lbf pcf in2 in4 cm PI g DBL_MAX DBL_MIN LCol
LBeam LBrace1 LBRBYield
    global IDCtrlNode IDCtrlDOF iSupportNode LBuilding Fy Es nu Gs AgCol IzCol AgBeam IzBeam
IzBrace EABeam EACol
    global QBeam QdlBeam QdlCol WeightCol WeightBeam BaysIn BaysAlong iFloorWeight WeightTotal
sumWiHi MassTotal
    global NBay BayH BayW NumStories N0col N0beam N0brace ColNode Fybrace Esbrace nubrace
Gsbrace k1 MassNode
    global dcol twcol bfcol tfcol dbeam twbeam bfbeam tfbeam nfdw nftw nfbf nftf ChangeFloors

    # SET UP ---------------------------------------------------------------------------
        wipe                        # clear memory of all past model definitions
        model basic -ndm 2 -ndf 3;    # Define the model builder, ndm=#dimension, ndf=#dofs
        set dataDir Data;             # set up name for data directory
        file mkdir $dataDir/$FileName/$ii/ ;   # create data directory
        source DisplayPlane_DisplayModel2D.tcl ;# procedure for displaying a plane in model and a
                            # procedure for displaying 2D perspective of model


    # define GEOMETRY -------------------------------------------------------------

        # This is formed at each "level"

        # |----------------------|
        # |                    X|
        # |                  X  |
        # |                X    |
        # |              X      |
        # |            X        |
        # |          X          |
        # |        X            |
        # |      X              |
```

```
        # |       X           |
        # |     X             |
        # |   X               |
        # |  X                |
        # |X                  |


        # nodal coordinates:

            set NumNodes 0
            set ColNode 40000

            for {set level 1} {$level <= [expr $NumStories+1]} {incr level} {
                set Y [expr ($level-1)*$LCol];
                # puts $Y
                for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier} {
                    set X [expr ($pier-1)*$LBeam]
                    set nodeID [expr $ColNode+$level*100+$pier*10];      # number for bottom node
                    node $nodeID $X $Y;                     # actually define bottom node
                    # puts "$nodeID"
                    incr NumNodes 1
                }
            }

            # puts "Number of Nodes: $NumNodes"

        # Single point constraints -- Boundary Conditions
            fixY 0.0 1 1 0;      # pin all Y=0.0 nodes

        # determine support nodes where ground motions are input, for multiple-support excitation
            set iSupportNode ""
            set level 1
            for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier} {
                set nodeID [expr $ColNode+$level*100+$pier*10]
                lappend iSupportNode $nodeID
            }

        # Set up parameters that are particular to the model for displacement control
            set IDctrlNode [expr $ColNode+$NumStories*100+1*10]; # node where displacement is read if
displacement control is used
            set IDctrlDOF 1;                    # degree of freedom of displacement read for displacement
control
            set LBuilding [expr $NumStories*$LCol];             # total building height

    # define UNIAXIAL materials - Structural-Steel properties-----------------------------------------------------------
-------------------------------------------------------------

        # Material properties, column and beam sections all defined in Input.tcl

            set BCMat 10
            set BraceMat 20
            set b_BC 0.01

            # $R0, $cR1, $cR2 control the transition from elastic to plastic branches.
            # Recommended values:
            # $R0=between 10 and 20, $cR1=0.925, $cR2=0.15
            set R0_BC 20
            set cR1_BC 0.925
            set cR2_BC 0.15

            # Beam and Column Materials
            uniaxialMaterial Steel02 $BCMat $Fy $Es $b_BC $R0_BC $cR1_BC $cR2_BC

            # Brace Materials
```

```
            set b_Brace 0.025
            # Parameters used in the Giuffré-Menegotto-Pinto equations
                set R0_Brace 1.95;    # exponent that controls the transition between elastic and hardening
branch
                set cR1_Brace 0.001;      # parameter for the change of R with cyclic loading history
                set cR2_Brace 0.001;      # parameter for the change of R with cyclic loading history

            uniaxialMaterial Steel02 $BraceMat [expr $Fybrace*1.65] [expr $k1] $b_Brace $R0_Brace
$cR1_Brace $cR2_Brace

            # puts "Materials Defined"

    # define SECTIONS-------------------------------------------------------------------------------------------------------------
----------------------

        #Beam and Column Sections

            set ColSecTag 1
            set BeamSecTag 2
            set listnum 0
            for {set level 1} {$level <= $NumStories} {incr level} {
                Wsection $ColSecTag$level $BCMat [lindex $dcol $listnum] [lindex $twcol $listnum] [lindex
$bfcol $listnum] [lindex $tfcol $listnum] $nfdw $nftw $nfbf $nftf
                # puts "$level $listnum [lindex $EACol $listnum] [lindex $dcol $listnum] [lindex $twcol
$listnum] [lindex $bfcol $listnum] [lindex $tfcol $listnum]"
                if {[expr $level % $ChangeFloors] == 0} {incr listnum 1};
            }
            Wsection $BeamSecTag $BCMat $dbeam $twbeam $bfbeam $tfbeam $nfdw $nftw $nfbf $nftf
            # puts "Sections Defined"

    # define ELEMENTS-----------------------------------------------------------------------------

        # set up geometric transformations of element
        #   separate columns and beams, in case of P-Delta analysis for columns
            set IDColTransf 1;      # all columns
            set IDBeamTransf 2;  # all beams
            set IDBraceTransf 3;  # all braces
            set ColTransfType Corotational;            # options, Linear PDelta Corotational
            geomTransf $ColTransfType $IDColTransf
            geomTransf Corotational $IDBeamTransf
            geomTransf Corotational $IDBraceTransf

        # Define Beam-Column Elements
            set np 3;    # number of Gauss integration points for nonlinear curvature distribution
            set NumElem 0
            set NumCol 0
            set NumBeam 0
            set NumBrace 0

            # columns

            set N0col 1000;    # column element numbers
            set listnum 0
            set level 0
            for {set level 1} {$level <= $NumStories} {incr level 1} {
                for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier 1} {
                    set elemID [expr $N0col+$level*10+$pier]
                    set nodeI [expr $ColNode+$level*100+$pier*10]
                    set nodeJ [expr $ColNode+($level+1)*100+$pier*10]
                    element nonlinearBeamColumn $elemID $nodeI $nodeJ $np $ColSecTag$level
$IDColTransf; # columns
                    incr NumElem
                    incr NumCol
```

```
            }
        }

        # puts "Columns Constructed"

        # beams

        set N0beam 2000;     # beam element numbers
        for {set level 1} {$level <= $NumStories} {incr level 1} {
            for {set bay 1} {$bay <= $NBay} {incr bay 1} {
                set elemID [expr $N0beam + $level*10 +$bay]
                set nodeI [expr $ColNode + ($level+1)*100 + $bay*10]
                set nodeJ [expr $ColNode + ($level+1)*100 + ($bay+1)*10]
                element nonlinearBeamColumn $elemID $nodeI $nodeJ $np $BeamSecTag
$IDBeamTransf;  # beams
                incr NumElem
                incr NumBeam
            }
        }

        # puts "Beams Constructed"


    # Define corotTruss Elements - braces

        set N0brace 3000;     # brace element numbers
        for {set level 1} {$level <= $NumStories} {incr level 1} {
            for {set pier 1} {$pier <= 1} {incr pier} {
                set elemID [expr $N0brace+$level*10+$pier]
                set nodeI [expr $ColNode+$level*100+$pier*10]
                set nodeJ [expr $ColNode+($level+1)*100+($pier+1)*10]
                element corotTruss $elemID $nodeI $nodeJ [expr $cdv($ii,$level)*pow($in,2)] $BraceMat
                # if {$level == 1} {element corotTruss $elemID $nodeI $nodeJ [expr 5.2*pow($in,2)]
$BraceMat}
                # if {$level == 2} {element corotTruss $elemID $nodeI $nodeJ [expr 3.5*pow($in,2)]
$BraceMat}
                # if {$level == 3} {element corotTruss $elemID $nodeI $nodeJ [expr 2.0*pow($in,2)]
$BraceMat}
                # element corotTruss $elemID $nodeI $nodeJ [expr 25.0*pow($in,2)] $BraceMat; # test
brace
                incr NumElem
                incr NumBrace
            }
        }

    # puts "Number of Elements: $NumElem"
    # puts "Number of Column Elements: $NumCol"
    # puts "Number of Beam Elements: $NumBeam"
    # puts "Number of Brace Elements: $NumBrace"


# Define GRAVITY LOADS, weight and masses

    # calculate dead load of frame
    # distributed weight along the beam length defined in Input procedure

    # assign masses to the nodes that the columns are connected to
    # each connection takes mass from the distributed weight of the beam

    set iFloorWeight ""
    set WeightTotal 0.0
    set sumWiHi 0.0;       # sum of story weight times height, for lateral-load distribution
    set listnum 0
    for {set level 2} {$level <= [expr $NumStories+1]} {incr level 1} { ;
```

```
            set FloorWeight 0.0
            if {$level == [expr $NumStories+1]} {
                set ColWeightFact 1;       # one column in top story
            } else {
                set ColWeightFact 2;       # two columns elsewhere
            }
            for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier 1} {;
                set weight [lindex $WeightBeam [expr $level-2]]
                set weight2 [lindex $WeightCol $listnum]
                # puts "Weight from Col:$weight2 Weight from floor:$weight"

                set WeightNode [expr $ColWeightFact*$weight2+$weight/2]
                set MassNode [expr $WeightNode/$g];
                set nodeID [expr $ColNode+$level*100+$pier*10]; # master node on all other floors is bottom
of column
                # puts "    Mass @ Node $nodeID: $MassNode $WeightNode"
                mass $nodeID $MassNode 0.0 0.0;       # define mass - units: kip-s^2/in

                set FloorWeight [expr $FloorWeight+$WeightNode];
                if {[expr $level % $ChangeFloors] == 1 && $level != 2} {incr listnum}
            }
            lappend iFloorWeight $FloorWeight
            set WeightTotal [expr $WeightTotal+ $FloorWeight]
            set sumWiHi [expr $sumWiHi+$FloorWeight*($level-1)*$LCol];    # sum of story weight times
height, for lateral-load distribution
        }
        set MassTotal [expr $WeightTotal/$g];
        # puts "    Total Building Mass: $MassTotal"



    # Define DISPLAY ------------------------------------------------------------
        DisplayModel2D NodeNumbers

    # puts "Done with AppInitialize"
}
```

A.7 AppAnalysis.v5.2

```
# AppAnalysis procedure

proc AppAnalysis {ii i} {

    global nobj cdv obj feasible FileName dataDir GMfile GMfact DBL_MAX NBay NumStories LCol LBeam
LBrace1 ngener
    global YieldDrift ElasticDrift AllowDuct AllowDrift AvgDrift Ductility PI period IDLoadTagGrav
IDLoadTagDyn
    global designtype

    set IDLoadTagGrav 500
    set IDLoadTagDyn 400;  # for uniformSupport excitation

    # Calculate the building period
        set nEigenI 1;      # mode 1
        set nEigenJ 3;      # mode 3
        set nEigenK $NumStories;    # mode for number of stories
        set lambdaN [eigen [expr $nEigenK]];              # eigenvalue analysis for nEigenJ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j
        set omegaI [expr pow($lambdaI,0.5)];
        set omegaJ [expr pow($lambdaJ,0.5)];
        set period($ii) [expr 2*$PI/$omegaI];             # 1st mode period
```

```
        set period3($ii) [expr 2*$PI/$omegaJ];              # 3rd mode period
        puts "     Period:   Mode 1: $period($ii)"
        puts "               Mode 3: $period3($ii)"

    set j 1
    set GMfact($ii) "";
    # puts $GMfact($ii);              # ground-motion scaling factor reset
    foreach GM $GMfile {
        Scale $period($ii) $j $ii;   # scale all the earthquakes to this design
        incr j
    }

    set j 0
    for {set level 1} {$level <= $NumStories} {incr level} {
        set TotDrift($level) 0.0
        set TotDeformation($level) 0.0
        # puts $TotDrift($level) $TotDeformation($level)
    }

    # puts $GMfact($ii)
    foreach GM $GMfile GMfac $GMfact($ii) {
        set feasible($ii) 0.0
        puts "    Earthquake #[expr $j+1]"
        reset
        wipeAnalysis
        remove recorders
        puts "     $GM $GMfac"
        GravityAnalysis $ii $j
        DynamicAnalysis $ii $GM $GMfac $j
        # print "$dataDir/$FileName/$ii/ModelParameters Eq[expr $j+1].out"
        remove loadPattern [expr $IDLoadTagGrav+$j]
        remove loadPattern [expr $IDLoadTagDyn+$j]

        # Calculate ductility from brace deformation
            puts "     Calculating ductility..."
            for {set level 1} {$level <= $NumStories} {incr level} {
                set Ductility($ii,$level) -$DBL_MAX
            }

            # Method from Graham Oxborrow on 4/2/09

            for {set level 1} {$level <= $NumStories} {incr level} {
                set Deformation($ii,$level,[expr $j+1]) -$DBL_MAX
                if {[catch {open "$dataDir/$FileName/$ii/BraceDeform$level Eq[expr $j+1].out" r 0666} fileID]}
{
                    puts "Cannot open $dataDir/$FileName/$ii/BraceDeform$level Eq[expr $j+1].out"
                } else {
                    foreach line [split [read $fileID] \r\n] {
                        set bracedeform [split $line]
                        set x [lindex $bracedeform 1]
                        if {$x != ""} {
                            set brdf [expr abs($x)]
                            if {$brdf > $Deformation($ii,$level,[expr $j+1])} {
                                set Deformation($ii,$level,[expr $j+1]) $brdf;
                                # this will turn out to be the max deformation on that level
                            }
                        }
                    }
                }
                # puts "Level $level Deformation: $Deformation($ii,$level,[expr $j+1])"
                close $fileID
            }
            for {set level 1} {$level <= $NumStories} {incr level} {
```

```
                set TotDeformation($level) [expr $TotDeformation($level)+$Deformation($ii,$level,[expr
$j+1])]
                set AvgDeformation($ii,$level) [expr $TotDeformation($level)/($j+1)]
                # puts "$Deformation($ii,$level,[expr $j+1]) $TotDeformation($level) [expr $j+1]
$AvgDeformation($ii,$level)"
            }
            for {set level 1} {$level <= $NumStories} {incr level} {
                set Ductility($ii,$level) [expr ($AvgDeformation($ii,$level))/$ElasticDrift]
                # puts "$Ductility($ii,$level) $ElasticDrift $AvgDeformation($ii,$level)"
            }

        # Calculate drift from brace deformation
            for {set level 1} {$level <= $NumStories} {incr level} {
                set LBraceNew [expr $LBrace1+$AvgDeformation($ii,$level)]
                set AngleH [expr acos((pow($LBeam,2)+pow($LBraceNew,2)-
pow($LCol,2))/(2*$LBeam*$LBraceNew))]
                set AngleL [expr acos((pow($LCol,2)+pow($LBraceNew,2)-
pow($LBeam,2))/(2*$LCol*$LBraceNew))]
                set DriftAngle [expr $PI/2-$AngleH-$AngleL]
                set AvgDrift($ii,$level) [expr sin($DriftAngle)]
                # puts "$DriftAngle $AngleH $AngleL $AvgDrift($ii,$level)"
            }

        # Determine feasibility value

            for {set level 1} {$level <= $NumStories} {incr level} {

                # Normalize constraints so that they can be compared
                    set drift [expr ($AvgDrift($ii,$level)-$AllowDrift)/$AllowDrift]
                    set duct [expr ($Ductility($ii,$level)-$AllowDuct)/$AllowDuct]
                    # puts "$drift $duct"

                # Determine feasibility
                    if {$drift > $duct && $drift > 0.0} {
                        set feasible($ii) [expr $feasible($ii)+$drift]
                        # puts $drift
                        # puts "      Feasible $feasible($ii)"
                    } elseif {$duct > 0.0} {
                        set feasible($ii) [expr $feasible($ii)+$duct]
                        # puts $duct
                        # puts "      Feasible $feasible($ii)"
                    }
            }
            puts "     Feasiblity $feasible($ii)"
        if {$feasible($ii) > 0.0 && [expr $j+1] >= 3 && $designtype == "Optimize"} {puts "         !!!Infeasible
design!!!"; break}
        incr j
        if {$i < [expr 0.8*$ngener] && $j == 3} {break}
    }

    set TotNum $j
    # puts "Number of Earthquakes: $TotNum"

    # Calculating the objective - minimum cost
        puts "    Calculating the objective..."
        for {set k 1} {$k <= $nobj} {incr k} {
            set obj($ii,$k) 0.0
            for {set level 1} {$level <= $NumStories} {incr level} {
                set obj($ii,$k) [expr $obj($ii,$k) + $cdv($ii,$level)];  #the first cdv is the area of the brace
                # puts $obj($ii,$k)
            }
        }
}
```

A.8 Gravity_Analysis.v4.1

```
# Gravity Analysis

proc GravityAnalysis {ii num} {

    global FileName dataDir QdlBeam QdlCol ColNode Tol NumStories NBay N0col N0beam N0brace
DBL_MAX BraceMat IDLoadTagGrav LCol LBeam ChangeFloors

    # RECORDERS

        # Drift recorders
            for {set level 1} {$level <= $NumStories} {incr level 1} {
                set pier 1
                set inodeID [expr $ColNode+$level*100+$pier*10]
                set jnodeID [expr $ColNode+($level+1)*100+($pier+1)*10]
                set elemID [expr $N0brace+$level*10+$pier]
                # recorder Drift -file "$dataDir/$FileName/$ii/DrLevel$level Eq[expr $num+1].out" -time -iNode
$inodeID  -jNode $jnodeID  -dof 1 -perpDirn 2; # lateral drift
                # recorder Drift -file "$dataDir/$FileName/$ii/DrLevel$level Eq[expr $num+1]_x.out" -time -
iNode $inodeID  -jNode $jnodeID  -dof 1 -perpDirn 2;  # lateral drift
                # recorder Drift -file "$dataDir/$FileName/$ii/DrLevel$level Eq[expr $num+1]_y.out" -time -
iNode $inodeID  -jNode $jnodeID  -dof 2 -perpDirn 1;  # vertical drift
                recorder Element -file "$dataDir/$FileName/$ii/BraceDeform$level Eq[expr $num+1].out" -time
-ele $elemID deformations;                  # strain in brace
            }

            # for {set level 1} {$level == 1} {incr level 1} {
                # set pier 1
                # set elemID [expr $N0brace+$level*10+$pier]
                # recorder Element -file "$dataDir/$FileName/$ii/ElemForce$level Eq[expr $num+1].out" -time
-ele $elemID axialForce;
                # recorder Element -file "$dataDir/$FileName/$ii/ElemMatStress$level Eq[expr $num+1].out" -
time -ele $elemID material stress;
                # recorder Element -file "$dataDir/$FileName/$ii/ElemMatStrain$level Eq[expr $num+1].out" -
time -ele $elemID material strain;
                # recorder Element -file "$dataDir/$FileName/$ii/ElemMatStressStrain$level Eq[expr
$num+1].out" -time ele $elemID section stressStrain
                # puts $BraceMat
                # puts "Force recorder defined.."
                # recorder Node -file $dataDir/DFree.out -time -node $inodeID  -dof 1 2 3 disp;       #
displacements of free node
            # }


        # Determine if other recorders are necessary

    # GRAVITY LOADS
    # define gravity load applied to beams and columns -- eleLoad applies loads in local coordinate axis


        pattern Plain [expr $IDLoadTagGrav+$num] Linear { ; # does not work for the corotational
transformation with nonlinear beam-columns in current OpenSees version
                                    # check the calculations steps for applying the loads
            #set listnum 0
            #for {set level 1} {$level <= $NumStories} {incr level 1} {
            #    for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier 1} {
            #        set elemID [expr $N0col+$level*10+$pier]
            #        if {[expr $level % $ChangeFloors] > $listnum} {incr listnum}
            #        set Point [expr -0.20*$LCol*[lindex $QdlCol $listnum]]
            #        eleLoad -ele $elemID -type -beamPoint 0.0 0.0 [expr $Point/2];    # COLUMNS (at Start
Node)
```

```
            #       eleLoad -ele $elemID -type -beamPoint 0.0 0.2 $Point;       # COLUMNS (at 0.2L on
Column)
            #       eleLoad -ele $elemID -type -beamPoint 0.0 0.4 $Point;       # COLUMNS (at 0.4L on
Column)
            #       eleLoad -ele $elemID -type -beamPoint 0.0 0.6 $Point;       # COLUMNS (at 0.6L on
Column)
            #       eleLoad -ele $elemID -type -beamPoint 0.0 0.8 $Point;       # COLUMNS (at 0.8L on
Column)
            #       eleLoad -ele $elemID -type -beamPoint 0.0 1.0 [expr $Point/2];    # COLUMNS (at 1.0L on
Column)
            #       puts $Point
            #   }
            #}
            #set listnum 0
            #for {set level 1} {$level <= $NumStories} {incr level 1} {
            #   for {set bay 1} {$bay <= $NBay} {incr bay 1} {
            #       set elemID [expr $N0beam+$level*10+$bay]
            #       set Point [expr -0.20*$LBeam*[lindex $QdlBeam $listnum]]
            #       eleLoad -ele $elemID -type -beamPoint $Point 0.0;     # BEAMS (at Start Node)
            #       eleLoad -ele $elemID -type -beamPoint $Point 0.2;     # BEAMS (at 0.2L on Beam)
            #       eleLoad -ele $elemID -type -beamPoint $Point 0.4;     # BEAMS (at 0.4L on Beam)
            #       eleLoad -ele $elemID -type -beamPoint $Point 0.6;     # BEAMS (at 0.6L on Beam)
            #       eleLoad -ele $elemID -type -beamPoint $Point 0.8;     # BEAMS (at 0.8L on Beam)
            #       eleLoad -ele $elemID -type -beamPoint $Point 1.0;     # BEAMS (at 1.0L on Beam)
            #       puts $Point
            #   }
            #   incr listnum
            #}
        }

        set nEigenI 1
        set lambdaN [eigen -fullGenLapack 1];           # eigenvalue analysis for first mode
        # puts $lambdaN
        # DisplayModel2D DeformedShape

        # Gravity-analysis parameters -- load-controlled static analysis
            set Tol 1.0e-7;             # convergence tolerance for test
            # constraints Plain;                # might be better suited for this analysis since on homogenous
single point constraints are used
            constraints Penalty $DBL_MAX $DBL_MAX;    # how it handles boundary conditions
            numberer RCM;                       # renumber dof's to minimize band-width (optimization), if you
want to
            system BandGeneral;                 # how to store and solve the system of equations in the
analysis
            test NormDispIncr $Tol 6 ;          # determine if convergence has been achieved at the end of
an iteration step
            algorithm Newton;                   # use Newton's solution algorithm: updates tangent stiffness at
every iteration
            set NstepGravity 10;                # apply gravity in 10 steps
            set DGravity [expr 1./$NstepGravity];       # first load increment;
            integrator LoadControl $DGravity;       # determine the next time step for an analysis
            analysis Static;                # define type of analysis static or transient
            analyze $NstepGravity;              # apply gravity

        # ------------ maintain constant gravity loads and reset time to zero-----------------
            loadConst -time 0.0

    # Conclude Model Construction
        puts "     Model Built"
}
```

A.9 Dynamic_Analysis.v5

```
proc DynamicAnalysis {i GM GMfac num} {

    global dataDir GMdir GMfact sec g DBL_MIN nu Gs Tol WeightCol WeightBeam iFloorWeight
WeightTotal sumWiHi MassTotal
    global NBay NumStories ColNode LCol PI period ncdv IDLoadTagDyn xDamp TmaxAnalysis
    global constraintsTypeDynamic numbererTypeDynamic systemTypeDynamic TolDynamic
maxNumIterDynamic printFlagDynamic
    global testTypeDynamic maxNumIterConvergeDynamic printFlagConvergeDynamic
algorithmTypeDynamic NewmarkGamma NewmarkBeta
    global integratorTypeDynamic analysisTypeDynamic

    # Uniform Earthquake ground motion (uniform acceleration input at all support nodes)
        set GMdirection 1;                  # ground-motion direction

    # display deformed shape:
        set ViewScale 5;                    # amplify display of deformed shape
        # DisplayModel2D DeformedShape $ViewScale ;             # display deformed shape, the scaling
factor needs to be adjusted for each model
        # recorder plot $dataDir/DFree.out Displ 10 700 400 400 -columns 1 2;   # a window to plot the nodal
displacements versus time

    # set up ground-motion-analysis parameters
        set DtAnalysis[expr 0.02*$sec];       # time-step Dt for lateral analysis

    # ----------- set up analysis parameters
        LibAnalysisDynamicParameters ;    # constraintsHandler,DOFnumberer,system-
ofequations,convergenceTest,solutionAlgorithm,integrator

    # ------------ define & apply damping
    # RAYLEIGH damping parameters, Where to put M/K-prop damping, switches
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/1099.htm)
    #         D=$alphaM*M + $betaKcurr*Kcurrent + $betaKcomm*KlastCommit + $beatKinit*$Kinitial

        set MpropSwitch 1.0;
        set KcurrSwitch 0.0;
        set KcommSwitch 1.0;
        set KinitSwitch 0.0;
        set nEigenI 1;      # mode 1
        set nEigenJ 3;      # mode 3
        set nEigenK $ncdv;
        set lambdaN [eigen [expr $nEigenK]];              # eigenvalue analysis for nEigenJ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j
        set omegaI [expr pow($lambdaI,0.5)];
        set omegaJ [expr pow($lambdaJ,0.5)];
        set alphaM [expr $MpropSwitch*$xDamp*(2*$omegaI*$omegaJ)/($omegaI+$omegaJ)];     # M-prop.
damping; D = alphaM*M
        set betaKcurr [expr $KcurrSwitch*2.*$xDamp/($omegaI+$omegaJ)];                # current-K;
+beatKcurr*KCurrent
        set betaKcomm [expr $KcommSwitch*2.*$xDamp/($omegaI+$omegaJ)];          # last-committed K;
+betaKcomm*KlastCommitt
        set betaKinit [expr $KinitSwitch*2.*$xDamp/($omegaI+$omegaJ)];                # initial-K;
+beatKinit*Kini

    # define damping
        rayleigh $alphaM $betaKcurr $betaKinit $betaKcomm;      # RAYLEIGH damping

    #  --------------------------------    perform Dynamic Ground-Motion Analysis
    # the following commands are unique to the Uniform Earthquake excitation

    # Uniform EXCITATION: acceleration input
```

```
        set inFile $GMdir/$GM.AT2
        set outFile $GMdir/$GM.g3;      # set variable holding new filename (PEER files have .at2/dt2
extension)
        ReadSMDFile $inFile $outFile dt;    # call procedure to convert the ground-motion file to proper
format
        set GMfatt [expr $g*$GMfac];        # data in input file is in g Units -- ACCELERATION TH
        set AccelSeries "Series -dt $dt -filePath $outFile -factor  $GMfatt";    # time series information
        pattern UniformExcitation  [expr $IDLoadTagDyn+$num]  $GMdirection -accel  $AccelSeries  ;
    # create Uniform excitation

        set Nsteps [expr int($TmaxAnalysis/$DtAnalysis)];
        # puts $TmaxAnalysis
        # puts $DtAnalysis
        # puts $Nsteps

        set ok [analyze $Nsteps $DtAnalysis];            # actually perform analysis; return ok=0 if analysis
was successful
        # puts $ok
        if {$ok != 0} {     ;                    # analysis was not successful.
            # ------------------------------------------------------------------------------------------------
            # change some analysis parameters to achieve convergence
            # performance is slower inside this loop
            #    Time-controlled analysis

            set ok 0;
            set controlTime [getTime];
            while {$controlTime < $TmaxAnalysis && $ok == 0} {
                set controlTime [getTime]
                set ok [analyze 1 $DtAnalysis]
                if {$ok != 0} {
                    puts "Trying Newton with Initial Tangent .."
                    test NormDispIncr   $Tol 1000  0
                    algorithm Newton -initial
                    set ok [analyze 1 $DtAnalysis]
                    test $testTypeDynamic $TolDynamic $maxNumIterDynamic  0
                    algorithm $algorithmTypeDynamic
                }
                if {$ok != 0} {
                    puts "Trying Broyden .."
                    algorithm Broyden 8
                    set ok [analyze 1 $DtAnalysis]
                    algorithm $algorithmTypeDynamic
                }
                if {$ok != 0} {
                    puts "Trying NewtonWithLineSearch .."
                    algorithm NewtonLineSearch .8
                    set ok [analyze 1 $DtAnalysis]
                    algorithm $algorithmTypeDynamic
                }
            }
        };     # end if ok !0

        puts "     Ground Motion Done. End Time: [getTime]"
}

#-------------------------------------------------------------------------------------------------

proc LibAnalysisDynamicParameters {} {

# ------------------------------------------------------------------------------------------------
# dynamic-analysis parameters
# I am setting all these variables as global variables (using variable rather than set command)
#    so that these variables can be uploaded by a procedure
#                            Silvia Mazzoni & Frank McKenna, 2006
```

```
# Set up Analysis Parameters -----------------------------------------------
# CONSTRAINTS handler -- Determines how the constraint equations are enforced in the analysis
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/617.htm)
#         Plain Constraints -- Removes constrained degrees of freedom from the system of equations
#         Lagrange Multipliers -- Uses the method of Lagrange multipliers to enforce constraints
#         Penalty Method -- Uses penalty numbers to enforce constraints
#         Transformation Method -- Performs a condensation of constrained degrees of freedom
variable constraintsTypeDynamic Transformation;
constraints $constraintsTypeDynamic ;

# DOF NUMBERER (number the degrees of freedom in the domain):
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/366.htm)
#   determines the mapping between equation numbers and degrees-of-freedom
#         Plain -- Uses the numbering provided by the user
#         RCM -- Renumbers the DOF to minimize the matrix band-width using the Reverse Cuthill-McKee
algorithm
variable numbererTypeDynamic RCM
numberer $numbererTypeDynamic

# SYSTEM (http://opensees.berkeley.edu/OpenSees/manuals/usermanual/371.htm)
#   Linear Equation Solvers (how to store and solve the system of equations in the analysis)
#   -- provide the solution of the linear system of equations Ku = P. Each solver is tailored to a specific matrix
topology.
#         ProfileSPD -- Direct profile solver for symmetric positive definite matrices
#         BandGeneral -- Direct solver for banded unsymmetric matrices
#         BandSPD -- Direct solver for banded symmetric positive definite matrices
#         SparseGeneral -- Direct solver for unsymmetric sparse matrices (-piv option)
#         SparseSPD -- Direct solver for symmetric sparse matrices
#         UmfPack -- Direct UmfPack solver for unsymmetric matrices
variable systemTypeDynamic BandGeneral;     # try UmfPack for large problems
system $systemTypeDynamic

# TEST: # convergence test to
# Convergence TEST (http://opensees.berkeley.edu/OpenSees/manuals/usermanual/360.htm)
#   -- Accept the current state of the domain as being on the converged solution path
#   -- determine if convergence has been achieved at the end of an iteration step
#         NormUnbalance -- Specifies a tolerance on the norm of the unbalanced load at the current iteration
#         NormDispIncr -- Specifies a tolerance on the norm of the displacement increments at the current
iteration
#         EnergyIncr-- Specifies a tolerance on the inner product of the unbalanced load and displacement
increments at the current iteration
#         RelativeNormUnbalance --
#         RelativeNormDispIncr --
#         RelativeEnergyIncr --
variable TolDynamic 1.e-8;                      # Convergence Test: tolerance
variable maxNumIterDynamic 10;                  # Convergence Test: maximum number of iterations that will be
performed before "failure to converge" is returned
variable printFlagDynamic 0;                # Convergence Test: flag used to print information on convergence
(optional)        # 1: print information on each step;
variable testTypeDynamic EnergyIncr;   # Convergence-test type
test $testTypeDynamic $TolDynamic $maxNumIterDynamic $printFlagDynamic;
# for improved-convergence procedure:
    variable maxNumIterConvergeDynamic 2000;
    variable printFlagConvergeDynamic 0;

# Solution ALGORITHM: -- Iterate from the last time step to the current
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/682.htm)
#         Linear -- Uses the solution at the first iteration and continues
#         Newton -- Uses the tangent at the current iteration to iterate to convergence
#         ModifiedNewton -- Uses the tangent at the first iteration to iterate to convergence
#         NewtonLineSearch --
#         KrylovNewton --
```

```
#         BFGS --
#         Broyden --
variable algorithmTypeDynamic Newton
algorithm $algorithmTypeDynamic;

# Static INTEGRATOR: -- determine the next time step for an analysis
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/689.htm)
#         LoadControl -- Specifies the incremental load factor to be applied to the loads in the domain
#         DisplacementControl -- Specifies the incremental displacement at a specified DOF in the domain
#         Minimum Unbalanced Displacement Norm -- Specifies the incremental load factor such that the
residual displacement norm in minimized
#         Arc Length -- Specifies the incremental arc-length of the load-displacement path
# Transient INTEGRATOR: -- determine the next time step for an analysis including inertial effects
#         Newmark -- The two parameter time-stepping method developed by Newmark
#         HHT -- The three parameter Hilbert-Hughes-Taylor time-stepping method
#         Central Difference -- Approximates velocity and acceleration by centered finite differences of
displacement
variable NewmarkGamma 0.5;    # Newmark-integrator gamma parameter (also HHT)
variable NewmarkBeta 0.25; # Newmark-integrator beta parameter
variable integratorTypeDynamic Newmark;
integrator $integratorTypeDynamic $NewmarkGamma $NewmarkBeta

# ANALYSIS  -- defines what type of analysis is to be performed
(http://opensees.berkeley.edu/OpenSees/manuals/usermanual/324.htm)
#         Static Analysis -- solves the KU=R problem, without the mass or damping matrices.
#         Transient Analysis -- solves the time-dependent analysis. The time step in this type of analysis is
constant. The time step in the output is also constant.
#         variableTransient Analysis -- performs the same analysis type as the Transient Analysis object. The
time step, however, is variable. This method is used when
#               there are convergence problems with the Transient Analysis object at a peak or when the time
step is too small. The time step in the output is also variable.
variable analysisTypeDynamic Transient
analysis $analysisTypeDynamic

}

#-------------------------------------------------------------------------------------------------------------------

proc ReadSMDFile {inFilename outFilename dt} {
    ############################################################
    # ReadSMDFile $inFilename $outFilename $dt
    ############################################################
    # read gm input format and output to opensees file
    #
    # Written: MHS
    # Date: July 2000
    #
    # A procedure which parses a ground motion record from the PEER
    # strong motion database by finding dt in the record header, then
    # echoing data values to the output file.
    #
    # Formal arguments
    #   inFilename -- file which contains PEER strong motion record
    #   outFilename -- file to be written in format G3 can read
    #   dt -- time step determined from file header
    #
    # Assumptions
    #   The header in the PEER record is, e.g., formatted as follows:
    #    PACIFIC ENGINEERING AND ANALYSIS STRONG-MOTION DATA
    #     IMPERIAL VALLEY 10/15/79 2319, EL CENTRO ARRAY 6, 230
    #     ACCELERATION TIME HISTORY IN UNITS OF G
    #     NPTS=  3930, DT= .00500 SEC

    upvar $dt DT;       # Pass dt by reference
```

```
    # Open the input file and catch the error if it can't be read
    if {[catch {open $inFilename r} inFileID]} {
        puts stderr "Cannot open $inFilename for reading"
    } else {
        # Open output file for writing
        set outFileID [open $outFilename w]

        # Flag indicating dt is found and that ground motion
        # values should be read -- ASSUMES dt is on last line
        # of header!!!
        set flag 0
        # Look at each line in the file
        foreach line [split [read $inFileID] \n] {
            if {[llength $line] == 0} {
                # Blank line --> do nothing
                continue
            } elseif {$flag == 1} {
                # Echo ground motion values to output file
                puts $outFileID $line
            } else {
                # Search header lines for dt
                foreach word [split $line] {
                    # Read in the time step
                    if {$flag == 1} {
                        set DT $word
                        break
                    }
                    # Find the desired token and set the flag
                    if {[string match $word "DT="] == 1} {set flag 1}
                }
            }
        }
        close $outFileID;  # Close the output file
        close $inFileID;    # Close the input file
    }
}
```

A.10 Input.v7

```
# Input File

proc Input {} {

    #Access the global variables

    global FileName nddv ncdv nobj nsize ngener ntourn probcross dcrosspar ccrosspar probmutate
    global dmutpar cmutpar dmin dmax cmin cmax Designfile GMdir GMfile NBay BayH BayW
    global NumStories in kip sec LuniTXT FuniTXT TuniTXT ft ksi psi lbf psf pcf in2 in4
    global cm PI g DBL_MAX DBL_MIN Fy Es nu Gs Fybrace Esbrace nubrace Gsbrace designtype
    global nfdw nftw nfbf nftf QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol QBeam AgBeam
    global IzBeam EABeam LCol LBeam LBrace1 LBRBYield k1 BaysIn BaysAlong WeightCol WeightBeam
QdlBeam
    global YieldDrift ElasticDrift AllowDuct AllowDrift dbeam twbeam bfbeam tfbeam xDamp ChangeFloors
TmaxAnalysis

    # define OPTIMIZATION parameters

    # Single Variables
        set FileName "Nine_Stories" ;
```

```
        # Choices: Three_Stories, Six_Stories, Nine_Stories, Twelve_Stories, Eighteen_Stories,
Nine_Stories_SameCol
        # set designtype "Optimize"; # Can be used to define what type of analysis mind "Optimize" "ELF"
        set designtype "ELF";   # If the designtype is ELF then the brace areas need to be set in the Main
procedure
        set nddv 0
        set ncdv 9;          # this corresponds to the number of stories in the building
        set nobj 1;      # this objective is to minimize the area
        set nsize 50;        # this number needs to be even
        set ngener 500;         # number of generations in the optimization
        set ntourn 6;        # number of parents selected to generate the children
        set probcross 0.6;# probability crossover will occur during creation of child designs
        set dcrosspar 0.0
        set ccrosspar 1.0;# 0.0 = uniform crossover; 1.0 = blend crossover
        set probmutate 0.4;   # probability that mutation in the design will occur
        set dmutpar 0.0
        set cmutpar 1.0;   # 0.0 = uniform mutation; >0.0 = dynamic mutation

    # Set variables for ELF procedure

        if {$designtype == "ELF"} {
            set nsize 1
            set ngener 1
        }

    # Set minimum and maximum value for the design variables
        # Discrete design variables
        #   if {$nddv != 0} {
        #       for {$nddv != 0} {$v <= $ncdv} {incr v} {
        #           set dmin($v) 0
        #           set dmax($v) 10
        #       }
        #   }

        # Continuous design variables
            if {$ncdv != 0} {
                for {set v 1} {$v <= $ncdv} {incr v} {
                    set cmin($v) 1.0
                    set cmax($v) 30.0
                }
            }

    # define UNITS -----------------------------------------------------------------------------
        set in 1.;                  # define basic units -- output units
        set kip 1.;                 # define basic units -- output units
        set sec 1.;                     # define basic units -- output units
        set LunitTXT "inch";            # define basic-unit text for output
        set FunitTXT "kip";             # define basic-unit text for output
        set TunitTXT "sec";             # define basic-unit text for output
        set ft [expr 12.*$in];          # define engineering units
        set ksi [expr $kip/pow($in,2)];         # kips per square inch
        set psi [expr $ksi/1000.];      # pounds per square inch
        set lbf [expr $psi*$in*$in];        # pounds force
        set psf [expr $lbf/(pow($ft,2))];   # pounds per square foot
        set pcf [expr $lbf/pow($ft,3)];     # pounds per cubic foot
        set in2 [expr $in*$in];         # inch^2
        set in4 [expr $in*$in*$in*$in];     # inch^4
        set cm [expr $in/2.54];             # centimeter
```

```
        set PI [expr 2*asin(1.0)];       # define constants
        set g [expr 32.2*$ft/pow($sec,2)];    # gravitational acceleration
        set DBL_MAX 1.e10;                # a really large number
        set DBL_MIN [expr 1.0/$DBL_MAX];          # a really small number


    # Earthquakes that are to be used in the analysis ------------------------------------------
    # this will be a list that will be created to cycle through on each design
        set Designfile $FileName;     # true if the design spectra file has same name as results files will have
        set GMdir "GMfiles"   ;    # ground-motion file directory
        if {$ncdv < 12} {
            # set Suite [list "CNP196"]
                # Single earthquake test
            set Suite [list "CNP196" "MUL279" "WIL180" "RO3090" "HDA255" "HCH180" "G03090" "G04090"
"B-PTS225" "B-PTS315"];
                # Initial 3- to 9-story earthquake suite
            # set Suite [list "ORR090" "WBA090" "JAB220" "TUJ262" "CYC195" "HVR090" "FRE090"
"AND270" "B-PLS135" "B-IVW360"];
                # Secondary 3- to 9-story earthquake suite
        } else {
            # set Suite [list "DZC270"]
                # Single earthquake test
            set Suite [list "DZC270" "WPI046" "TCU050-N" "CHY101-N" "YPT060" "YPT330" "LCN275" "H-
E04230" "H-E05230" "H-E06230"];
                # Intial 12- to 18-story earthquake suite
        }
        set GMfile $Suite ;               # ground-motion filenames
        set TmaxAnalysis [expr 20.*$sec];   # maximum duration of ground-motion analysis
        set xDamp 0.02;                   # damping ratio

        # Suite 1 (3- to 9-stories)
            # 1994 Northridge (90053 Canoga Park - Topanga Can) CNP196
            # 1994 Northridge (90013 Beverly Hills - 14145 Mulhol) MUL279
            # 1994 Northridge (90018 Hollywood - Willoughby Ave) WIL180
            # 1994 Northridge (90006 Sun Valley) RO3090
            # 1989 Loma Prieta (1656 Hollister Diff. Array) HDA255
            # 1989 Loma Prieta (1028 Hollister City Hall) HCH180
            # 1989 Loma Prieta (Gilroy Array #3) G03090
            # 1989 Loma Prieta (Gilroy Array #4) G04090
            # 1987 Superstition Hills (5051 Parachute Test Site) B-PTS225
            # 1987 Superstition Hills (5051 Parachute Test Site) B-PTS315

        # Suite 2 (3- to 9-stories)
            # 1994 Northridge (24278 Castaic - Old Ridge Route) ORR090
            # 1994 Northridge (90088 Anaheim - W Ball Rd) WBA090
            # 1994 Northridge (90094 Bell Gardens - Jaboneria) JAB220
            # 1994 Northridge (90061 Big Tujunga, Angeles Nat F) TUJ262
            # 1989 Loma Prieta (57217 Coyote Lake Dam (SW Abut)) CYC195
            # 1989 Loma Prieta (57191 Halls Valley) HVR090
            # 1989 Loma Prieta (57064 Fremont - Mission San Jose) FRE090
            # 1989 Loma Prieta (1652 Anderson Dam (Downstream)) AND270
            # 1987 Superstition Hills (5052 Plaster City) B-PLS135
            # 1987 Superstition Hills (5210 Wildlife Liquef. Array) B-IVW360

        # Suite 3 (12- to 18-stories)
            # 1999 Duzce, Turkey (Duzce) DZC270
            # 1999 Chi-Chi, Taiwan (TCU050) TCU050-N
            # 1999 Chi-Chi, Taiwan (CHY101) CHY101-N
```

```
        # 1999 Kocaeli, Turkey (Yarimca) YPT060
        # 1999 Kocaeli, Turkey (Yarimca) YPT330
        # 1994 Northridge (90056 Newhall - W. Pico Canyon Rd.) WPI046
        # 1992 Landers (24 Lucerne) LCN275
        # 1979 Imperial Valley (El Centro Array #4) H-E04230
        # 1979 Imperial Valley (El Centro Array #5) H-E05230
        # 1979 Imperial Valley (El Centro Array #6) H-E06230


# Input Parameters for building model ----------------------------------------
    set NBay 1;         # Number of bays
    set BayH 15.;       # bay height - standard input parameter (ft)
    set BayW 25.;       # bay width - standard input parameter (ft)
    set BaysIn 4. ;     # number of bays perpendicular to brace
    set BaysAlong 4. ;     # number of bays in line of brace
    set NumStories $ncdv;    # number of stories in building
    puts ""
    puts "Number of Stories: $NumStories Number of Bays: $NBay"

# Structural-Steel material properties ---------------------------------

    # beam and column materials
        set Fy [expr 50.0*$ksi];         # Yield stress
        set Es [expr 29000.*$ksi];          # Steel Young's Modulus
        set nu 0.3;                # Poisson's ratio
        set Gs [expr $Es/2./[expr 1+$nu]];    # Torsional stiffness Modulus

    # brace materials
        set Fybrace [expr 45.0*$ksi];      # Yield stress
        set Esbrace [expr 29000.*$ksi];       # Steel Young's Modulus
        set nubrace 0.3;             # Poisson's ratio
        set Gsbrace [expr $Esbrace/2./[expr 1+$nubrace]]; # Torsional stiffness modulus

    # column sections

        set QdlCol "";  set dcol ""
        set AgCol "";   set twcol ""
        set IzCol "";    set bfcol ""
        set EACol "";  set tfcol ""

        set nfdw 4;     # number of fibers along web depth
        set nftw 2; # number of fibers along web thickness
        set nfbf 4;  # number of fibers along flange width
        set nftf 2;   # number of fibers along flange thickness

        set ChangeFloors 2;  # column properties change every this number of floors

        Column_$FileName;  # procedure that inputs the column geometric properties

    # beam sections: W14x48

        set QBeam [expr 48.0*$lbf/$ft];             # W-section weight per length
        set AgBeam [expr 14.1*pow($in,2)];      # cross-sectional area
        set IzBeam [expr 484.0*pow($in,4)];     # moment of Inertia
        set EABeam [expr $Es*$AgBeam];            # EA, for axial-force-strain relationship

        set dbeam [expr 13.8*$in];    # nominal depth
        set twbeam [expr 0.340*$in];# web thickness
```

```
        set bfbeam [expr 8.03*$in];   # flange width
        set tfbeam [expr 0.595*$in];  # flange thickness

    # define GEOMETRY -------------------------------------------------------------
        set LCol [expr $BayH*$ft];                  # column length
        set LBeam [expr $BayW*$ft];                     # beam length
        set LBrace1 [expr sqrt(pow($LCol,2)+pow($LBeam,2))];    # brace overall length
        set LBrace2 [expr $dbeam/($LCol/$LBrace1)+24*$in]
        set LBRBYield [expr ($LBrace1-2*$LBrace2)*0.85]
        set k1 [expr ($Esbrace*$LBrace1/$LBRBYield)]
        # puts "$LBrace1 $LBrace2 $LBRBYield"

    # Define GRAVITY LOADS, weight and masses -------------------------------------

        # calculate distributed weight along the beam length

        # This is a sample of the floor layout
        # As BayIn and BaysAlong increase the bays size of the building changes
        # It is assumed that there is only one brace in the center of each exterior wall

        #   Bay 1   Bay 2   Bay 3  etc.
        # |-------|-------|-------|
        # |       |       |       |
        # |       |       |       | Bay 1
        # |       |       |       |
        # |-------|-------|-------|
        # |       |       |       |
        # |       |       |       | Bay 2   --- Floor Area - for mass associated with brace
        # |       |       |       |         |
        # |-------|-------|-------|         |
        # |       |       |       |         v
        # |       |       |       | Bay 3
        # |       |       |       |
        # |-------|-------|-------| etc.


        set FloorWeight [expr 90.0*$psf];               # floor dead load in psf
        set RoofWeight [expr 90.0*$psf];          # roof dead load in psf
        set FloorArea [expr $BaysIn/2*$BaysAlong*pow($LBeam,2)];# the tributary area associated with one
bay

        set WeightCol ""
        set WeightBeam ""
        set QdlBeam ""
        set listnum 0
        for {set level 1} {$level <= $NumStories} {incr level} {
            lappend WeightCol [expr $LCol*[lindex $QdlCol $listnum]];  # total Column weight
            if {$level == $NumStories} {
                set weight $RoofWeight
            } else {
                set weight $FloorWeight
            }
            lappend QdlBeam [expr $QBeam+$weight*($LBeam/2)];       # distributed gravity load on the
beam
            lappend WeightBeam [expr $QBeam*$LBeam+$FloorArea*$weight];     # total seismic weight
applied to beam
            # puts "$BaysIn $BaysAlong $LBeam $FloorArea $weight [expr $FloorArea*$weight]"
            if {[expr $level % $ChangeFloors] == 0} {incr listnum 1}
```

```
        }
        # puts "Beam Distributed Load: $QdlBeam Column Weight: $WeightCol Beam Weight:
$WeightBeam"


    # define CONSTRAINTS for OPTIMIZATION -------------------------------------------

        # calculate the drift at yield to later determine if ductility demand is less than ductility capacity

            set YieldDrift [expr ($Fybrace*$LBrace1)/(($LBeam/$LBrace1)*$Esbrace)]
            set ElasticDrift [expr ($Fybrace*$LBrace1)/$Esbrace]
            set AllowDuct [expr 7.0/2.0]; # This value is determined from ASCE 7-05 (R/overstrength factor)
            set AllowDrift [expr 0.015]
            # puts "Frame Drift: $YieldDrift Brace Elongation: $ElasticDrift
            # puts "Fy: $Fybrace Lbr: $LBrace1 Lbm: $LBeam Ebr: $Esbrace"
            # puts "Ductility Limit: $AllowDuct Drift Limit: $AllowDrift"

    # Other input parameters may exist -- find out what they are and include them

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Three_Stories {} {;

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W12x96

    set QdlCol [expr 96.0*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 28.2*pow($in,2)];    # cross-sectional area
    set IzCol [expr 833.*pow($in,4)]; # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 12.7*$in];   # nominal depth
    set twcol [expr 0.550*$in];    # web thickness
    set bfcol [expr 12.2*$in];  # flange width
    set tfcol [expr 0.90*$in];   # flange thickness

    # top floors: W12x45

    lappend QdlCol [expr 45.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 13.1*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 348.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 12.1*$in]; # nominal depth
    lappend twcol [expr 0.335*$in];   # web thickness
    lappend bfcol [expr 8.05*$in];    # flange width
    lappend tfcol [expr 0.575*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Six_Stories {} {;
```

```
    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x176

    set QdlCol [expr 176.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 51.8*pow($in,2)];    # cross-sectional area
    set IzCol [expr 2140.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 15.2*$in];  # nominal depth
    set twcol [expr 0.830*$in];    # web thickness
    set bfcol [expr 15.7*$in];  # flange width
    set tfcol [expr 1.31*$in];  # flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];  # web thickness
    lappend bfcol [expr 14.7*$in];    # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x68

    lappend QdlCol [expr 68.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 20.0*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 722*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.0*$in];# nominal depth
    lappend twcol [expr 0.415*$in];  # web thickness
    lappend bfcol [expr 10.0*$in];    # flange width
    lappend tfcol [expr 0.720*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Nine_Stories {} {

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x283

    set QdlCol [expr 283.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 83.3*pow($in,2)];    # cross-sectional area
    set IzCol [expr 3840.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 16.7*$in];  # nominal depth
    set twcol [expr 1.29*$in]; # web thickness
```

```
    set bfcol [expr 16.1*$in];  # flange width
    set tfcol [expr 2.07*$in];   # flange thickness

    # next set of floors up: W14x193

    lappend QdlCol [expr 193.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 56.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 2400.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 15.5*$in];# nominal depth
    lappend twcol [expr 0.890*$in];   # web thickness
    lappend bfcol [expr 15.7*$in];     # flange width
    lappend tfcol [expr 1.44*$in];# flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];   # web thickness
    lappend bfcol [expr 14.7*$in];     # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # next set of floors up: W14x74

    lappend QdlCol [expr 74.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 21.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 795.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.2*$in];# nominal depth
    lappend twcol [expr 0.450*$in];   # web thickness
    lappend bfcol [expr 10.1*$in];     # flange width
    lappend tfcol [expr 0.785*$in];    # flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];   # web thickness
    lappend bfcol [expr 8.03*$in];     # flange width
    lappend tfcol [expr 0.595*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Twelve_Stories {} {
```

```
global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

# bottom floors: W14x398

set QdlCol [expr 398.*$lbf/$ft];    # W-section weight per length
set AgCol [expr 117*pow($in,2)];# cross-sectional area
set IzCol [expr 6000.*pow($in,4)];    # moment of Inertia
set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

set dcol [expr 18.3*$in];   # nominal depth
set twcol [expr 1.77*$in]; # web thickness
set bfcol [expr 16.6*$in];  # flange width
set tfcol [expr 2.85*$in];   # flange thickness

# next set of floors up: W14x311

lappend QdlCol [expr 311.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 91.4*pow($in,2)];  # cross-sectional area
lappend IzCol [expr 4330.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

lappend dcol [expr 17.1*$in];# nominal depth
lappend twcol [expr 1.41*$in];    # web thickness
lappend bfcol [expr 16.2*$in];    # flange width
lappend tfcol [expr 2.26*$in];# flange thickness

# next set of floors up: W14x233

lappend QdlCol [expr 233.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 68.5*pow($in,2)];  # cross-sectional area
lappend IzCol [expr 3010.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

lappend dcol [expr 16.0*$in];# nominal depth
lappend twcol [expr 1.07*$in];    # web thickness
lappend bfcol [expr 15.9*$in];    # flange width
lappend tfcol [expr 1.72*$in];# flange thickness

# next set of floors up: W14x145

lappend QdlCol [expr 145.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 42.7*pow($in,2)];  # cross-sectional area
lappend IzCol [expr 1710.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 3]];         # EA, for axial-force-strain relationship

lappend dcol [expr 14.8*$in];# nominal depth
lappend twcol [expr 0.680*$in];   # web thickness
lappend bfcol [expr 15.5*$in];    # flange width
lappend tfcol [expr 1.09*$in];# flange thickness

# next set of floors up: W14x132

lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 4]];         # EA, for axial-force-strain relationship
```

```
    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];  # web thickness
    lappend bfcol [expr 14.7*$in];    # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];   # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];   # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 5]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];  # web thickness
    lappend bfcol [expr 8.03*$in];    # flange width
    lappend tfcol [expr 0.595*$in];   # flange thickness

    # puts $dcol


}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Eighteen_Stories {} {; # not ready to run yet

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x665

    set QdlCol [expr 665.*$lbf/$ft];   # W-section weight per length
    set AgCol [expr 196*pow($in,2)];# cross-sectional area
    set IzCol [expr 12400.*pow($in,4)];  # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 21.6*$in];  # nominal depth
    set twcol [expr 2.83*$in]; # web thickness
    set bfcol [expr 17.7*$in];  # flange width
    set tfcol [expr 4.52*$in];  # flange thickness

    # next set of floors up: W14x550

    lappend QdlCol [expr 550.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 162.*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 9430.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 20.2*$in];# nominal depth
    lappend twcol [expr 2.38*$in];    # web thickness
    lappend bfcol [expr 17.2*$in];    # flange width
    lappend tfcol [expr 3.82*$in];# flange thickness

    # next set of floors up: W14x455

    lappend QdlCol [expr 455.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 134.0*pow($in,2)];# cross-sectional area
    lappend IzCol [expr 7190.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship
```

```
lappend dcol [expr 19.0*$in];# nominal depth
lappend twcol [expr 2.02*$in];    # web thickness
lappend bfcol [expr 16.8*$in];    # flange width
lappend tfcol [expr 3.21*$in];# flange thickness

# next set of floors up: W14x370

lappend QdlCol [expr 370.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 109*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 5440.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 3]];         # EA, for axial-force-strain relationship

lappend dcol [expr 17.9*$in];# nominal depth
lappend twcol [expr 1.66*$in];    # web thickness
lappend bfcol [expr 16.5*$in];    # flange width
lappend tfcol [expr 2.66*$in];# flange thickness

# next set of floors up: W14x283

lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 4]];# EA, for axial-force-strain relationship

lappend dcol [expr 16.7*$in];# nominal depth
lappend twcol [expr 1.29*$in];    # web thickness
lappend bfcol [expr 16.1*$in];    # flange width
lappend tfcol [expr 2.07*$in];# flange thickness

# next set of floors up: W14x211

lappend QdlCol [expr 211.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 62.0*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 2660.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 5]];# EA, for axial-force-strain relationship

lappend dcol [expr 15.7*$in];# nominal depth
lappend twcol [expr 0.980*$in];   # web thickness
lappend bfcol [expr 15.8*$in];    # flange width
lappend tfcol [expr 1.56*$in];# flange thickness

# next set of floors up: W14x132

lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 38.8*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 6]];         # EA, for axial-force-strain relationship

lappend dcol [expr 14.7*$in];# nominal depth
lappend twcol [expr 0.645*$in];   # web thickness
lappend bfcol [expr 14.7*$in];    # flange width
lappend tfcol [expr 1.03*$in];# flange thickness

# next set of floors up: W14x132

lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 38.8*pow($in,2)];   # cross-sectional area
```

```
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 7]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];   # web thickness
    lappend bfcol [expr 14.7*$in];     # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 8]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];   # web thickness
    lappend bfcol [expr 8.03*$in];     # flange width
    lappend tfcol [expr 0.595*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------------

proc Column_Nine_Stories_SameCol {} {

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x283

    set QdlCol [expr 283.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 83.3*pow($in,2)];     # cross-sectional area
    set IzCol [expr 3840.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 16.7*$in];   # nominal depth
    set twcol [expr 1.29*$in]; # web thickness
    set bfcol [expr 16.1*$in];  # flange width
    set tfcol [expr 2.07*$in];   # flange thickness

    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];     # web thickness
    lappend bfcol [expr 16.1*$in];     # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
```

```
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];    # web thickness
    lappend bfcol [expr 16.1*$in];    # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];    # web thickness
    lappend bfcol [expr 16.1*$in];    # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # top floors: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];    # web thickness
    lappend bfcol [expr 16.1*$in];    # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------

proc ELF {ii} {

    global cdv ncdv

    # These variables can be set to specific numbers and then analyzed

    if {$ncdv == 3} {
        set cdv($ii,1) 7.0
        set cdv($ii,2) 6.0
        set cdv($ii,3) 4.0
    } elseif {$ncdv == 6} {
        set cdv($ii,1) 9.5
        set cdv($ii,2) 9.5
        set cdv($ii,3) 8.5
        set cdv($ii,4) 7.5
        set cdv($ii,5) 5.5
        set cdv($ii,6) 3.5
    } elseif {$ncdv == 9} {
        set cdv($ii,1) 10.5
        set cdv($ii,2) 10.5
```

```
        set cdv($ii,3) 10.5
        set cdv($ii,4) 10.0
        set cdv($ii,5) 9.0
        set cdv($ii,6) 8.0
        set cdv($ii,7) 7.0
        set cdv($ii,8) 5.0
        set cdv($ii,9) 3.0
    } elseif {$ncdv == 12} {
        set cdv($ii,1) 11.5
        set cdv($ii,2) 11.5
        set cdv($ii,3) 11.5
        set cdv($ii,4) 11.0
        set cdv($ii,5) 11.0
        set cdv($ii,6) 10.5
        set cdv($ii,7) 9.5
        set cdv($ii,8) 8.5
        set cdv($ii,9) 7.5
        set cdv($ii,10) 6.0
        set cdv($ii,11) 4.5
        set cdv($ii,12) 3.0
    } elseif {$ncdv == 18} {
        set cdv($ii,1) 12.5
        set cdv($ii,2) 12.5
        set cdv($ii,3) 12.5
        set cdv($ii,4) 12.5
        set cdv($ii,5) 12.5
        set cdv($ii,6) 12.5
        set cdv($ii,7) 12.0
        set cdv($ii,8) 12.0
        set cdv($ii,9) 11.5
        set cdv($ii,10) 11.0
        set cdv($ii,11) 10.5
        set cdv($ii,12) 9.5
        set cdv($ii,13) 9.0
        set cdv($ii,14) 8.0
        set cdv($ii,15) 6.5
        set cdv($ii,16) 5.5
        set cdv($ii,17) 4.0
        set cdv($ii,18) 2.0
    }
}
```

A.11 Monotonically Decreasing Version

The following procedures can be used to replace their counterparts in the original version previously presented. The monotonically decreasing constraint is implemented by randomly choosing the size of the top brace and increasing the value of subsequent braces by a small amount as seen in Equation A–1.

$$a_i = a_k + \sum_{j=i}^{k} \Delta_j \quad \text{(A-1)}$$

where a is the area of the brace, k is the top story of the structure, and Δ is the amount of increase for each brace down the structure. The following sections can be copied and run under the same conditions explained at the beginning of the chapter except Main_Mono.tcl is the file to be run to access all other procedures.

A.11.1 Main_Mono

```
# Main algorithm for monotonically decreasing constraint

global nddv ncdv nobj nsize ngener ndesign dmin dmax cmin cmax designtype
global ddv cdv index obj feasible fitness dalpha calpha DBL_MAX DBL_MIN
global dataDir NBay NumStories dup FileName AvgDrift Ductility MassNode period

source Input.Mono.tcl
source AppInitialize.Mono.tcl
source Random_Selection_Crossover_Mutation.tcl
source Maximum_Fitness.tcl
source AppAnalysis.Mono.tcl
source Gravity_Analysis.v4.1.tcl
source Dynamic_Analysis.v5.tcl
source EqScaling.tcl

Input

file mkdir Results/Monotonic/
set day [clock format [clock seconds] -format "%b%d %H%M"]
if [catch {open "Results/Monotonic/$FileName $day $designtype.out" w 0666} results] {
    puts stderr "Cannot open $results"
} else {
    puts $results "$FileName $day $designtype"
    set time [clock format [clock seconds] -format "%D %T%p %Z"]
    puts $results "Start_time: $time";     # prints the current time
    close $results
}

set DesignAvePrev [expr 1.0e-10]
set l 0; # counter for consecutive converging generations
for {set i 1} {$i <= $ngener} {incr i} {

    puts ""
    puts "Generation $i is running"

    set dalpha [expr pow(1-($i-1.0)/$ngener,$dmutpar)]
    set calpha [expr pow(1-($i-1.0)/$ngener,$cmutpar)]
    puts "$calpha $i $ngener $cmutpar"

    # Starting generation

    if {$i == 1} {
        if {$designtype == "ELF"} {
```

```
        ELF $i; # go to this procedure to inout single case brace values
            # this can be used to analyze any design once
    } else {
        for {set j 1} {$j <= $nsize} {incr j} {
            for {set k 1} {$k <= $nddv} {incr k} {
                set random [randint $dmin($k) $dmax($k)]
                set ddv($j,$k) $random
            }
            for {set k 1} {$k <= $ncdv} {incr k} {
                set random [randreal $cmin($k) $cmax($k)]
                set cdv($j,$k) $random
            }
        }
    }
    for {set j 1} {$j <= [expr 2*$nsize]} {incr j} {
        set index($j) $j
    }
    set ndesign $nsize
    set firstnew 1
    EqScaling

# Child generation

} else {
    set k $nsize
    for {set j 1} {$j <= [expr $nsize/2]} {incr j} {
        set child1 $index([expr $k+1])
        set child2 $index([expr $k+2])
        GASelection $child1
        GASelection $child2
        GACrossover $child1 $child2
        GAMutation $child1
        GAMutation $child2
        incr k 2
    }
    set ndesign $k
    set firstnew [expr ($nsize+1)]
}

# Analyze Generation

    # Analyze the 50 best deisigns from the first 75% of ngener under all the earthquakes

    if {$i == [expr 0.80*$ngener]} {
        for {set j 1} {$j <= $nsize} {incr j} {
            puts "  Generation $i Design $j"
            set ii $index($j)
            for {set level 1} {$level <= $NumStories} {incr level} {
                puts "     $level    $cdv($ii,$level)"
            }
            AppInitialize $ii
            AppAnalysis $ii $i
        }
    }

    for {set j $firstnew} {$j <= $ndesign} {incr j} {
        puts "  Generation $i Design $j"
        GADupAnalysis $j
        if {$dup == 0.0} {
            set ii $index($j)
            for {set level 1} {$level <= $NumStories} {incr level} {
                puts "     $level    $cdv($ii,$level)"
            }
            AppInitialize $ii
```

```
                AppAnalysis $ii $i
            }
        }

    # Evaluate Fitness and Sort

        puts ""
        puts "Evaluating Fitness and Sorting of Generation $i"
        GAMaximumFitness
        GAEliteSort

    # Check Convergence Criteria - may not be correct - not used
        # set ConvLimit 0.01;      # percentage limit for convergence between designs
        # set NumGen4Conv 4;   # number of consecutive generations to determine convergence
        set DesignTotal 0.0
        for {set j 1} {$j <= $nsize} {incr j} {
            set ii $index($j)
            set DesignTotal [expr $DesignTotal+$obj($ii,1)]
        }
        set DesignAve [expr $DesignTotal/$nsize]
        set Change [expr abs($DesignAve/$DesignAvePrev-1)]
        set Change [format "%1.6f" $Change]
        puts "     Change from last generation is [expr $Change*100]%"
        #if {$Change <= $ConvLimit && $i >= [expr $ngener*0.8]} {
        #    incr l
        #} else {
        #    set l 0
        #}
        #if {$l >= $NumGen4Conv && $i >= [expr $ngener*0.8]} {break}
        set DesignAvePrev $DesignAve
        #puts "     $l Consecutive Generations with a change of less than [expr $ConvLimit*100]%"

        # Output file for design information (recomposed at each generation)

        if [catch {open "Results/$FileName.out" w 0666} results] {
            puts stderr "Cannot open $results"
        } else {
            set time [clock format [clock seconds] -format "%T%p %Z"]
            puts $results "Time: $time";                        # prints the current time
            puts $results "Generation $i Tot_Generations $ngener Mass: [expr 2*$MassNode]"
            puts $results ""
            for {set l 1} {$l <= $nsize} {incr l} {
                set ii $index($l);              # this references the right values and output them in the right order
                if {$nddv > 0} {puts -nonewline $results "output ddv values"}
                if {$ncdv > 0} {
                    puts $results "Design_Rank Fitness Feasibility Total_Brace_Area Period_Mode_1
Constraints_Files"
                    for {set k 1} {$k <= $nobj} {incr k} {
                        puts $results "$l $fitness($ii) $feasible($ii) $obj($ii,$k) $period($ii) $dataDir/$ii";
                            # this should output the design number, fitness, feasible,obj value
                            # and the location of the files used to determine feasibility
                    }
                    puts $results "Ground_Files: $GMfile"
                    puts $results "Scaling_Factor: $GMfact($ii)"
                    puts $results ""
                    puts $results "Level Brace_Area Max_Avg_Drift Ductility"
                    for {set level 1} {$level <= $NumStories} {incr level} {
                        puts $results "$level $cdv($ii,$level) $AvgDrift($ii,$level) $Ductility($ii,$level)";

                            #this section outputs to size of the brace on that level
                    }
                puts $results ""
                }
            }
```

```
            close $results
        }

        # Output Area information at each generation (Amended througout the optimization)

        if {$i == 1} {
            if [catch {open "Results/Areas $FileName.out" w 0666} results] {
                puts stderr "Cannot open $results"
            }
        }
        if [catch {open "Results/Areas $FileName.out" a 0666} results] {
            puts stderr "Cannot open $results"
        } else {
            if {$i == 1} {
                puts -nonewline $results "Generation ";
                for {set l 1} {$l <= $nsize} {incr l} {
                    puts -nonewline $results "Design_$l "
                }
                puts $results "Tot_Area Average_Total_Area Average_Story_Area_Opt_Design"
            }
            puts -nonewline $results "$i "
            for {set l 1} {$l <= $nsize} {incr l} {
                set ii $index($l)
                puts -nonewline $results "$obj($ii,1) "
            }
            set ii $index(1)
            set Area $obj($ii,1)
            set AvgArea [expr $Area/$ncdv]
            puts $results "$DesignTotal $DesignAve $AvgArea"
            close $results
        }
}

# Postprocess

puts "Postprocessing..."

if [catch {open "Results/$FileName $day $designtype.out" a 0666} results] {
    puts stderr "Cannot open $results"
} else {
    set time [clock format [clock seconds] -format "%D %T%p %Z"]
    puts $results "End_time: $time";                    # prints the current time
    puts $results ""
    puts $results "Stories Bays Generations_Run Generations_Total Generation_Size Tournament_Size
Mass_Floor_(k-s^2/in) Bay_Height Bay_Width Bays_In Bays_Along";
    puts $results "$NumStories $NBay [expr $i-1] $ngener $nsize $ntourn [expr 2*$MassNode] $BayH
$BayW $BaysIn $BaysAlong"
    puts $results "Crossover_Probability Mutation_Probability Cont_Cross_Parameter Cont_Mut_Parameter"
    puts $results "$probcross $probmutate $ccrosspar $cmutpar"
    puts $results ""
    for {set i 1} {$i <= $nsize} {incr i} {
        set ii $index($i);          # this references the right values and output them in the right order
        if {$nddv > 0} {puts -nonewline $results "output ddv values"}
        if {$ncdv > 0} {
            puts $results "Design_Rank Fitness Feasibility Total_Brace_Area Period_Mode_1
Constraints_Files"
            for {set k 1} {$k <= $nobj} {incr k} {
                puts $results "$i $fitness($ii) $feasible($ii) $obj($ii,$k) $period($ii) $dataDir/$ii";
                    # this should output the design number, fitness, feasible,obj value
                    # and the location of the files used to determine feasibility
            }
            puts $results "Ground_Files: $GMfile"
            puts $results "Scaling_Factor: $GMfact($ii)"
            puts $results ""
```

```
            puts $results "Level Brace_Area Max_Avg_Drift Ductility"
            for {set level 1} {$level <= $NumStories} {incr level} {
                puts $results "$level $cdv($ii,$level) $AvgDrift($ii,$level) $Ductility($ii,$level)";
                    #this section outputs to size of the brace on that level
            }
        puts $results ""
        }
    }
    close $results
}

puts "DONE WITH OPTIMIZATION!"
```

A.11.2 AppInitialize.Mono

```
# Initialize the program

proc AppInitialize {ii} {

    global nddv ncdv nobj nsize ngener ntourn probcross dcrosspar ccrosspar probmutate dmutpar cmutpar
dmin dmax cmin cmax
    global ddv cdv index obj feasible fitness dalpha calpha FileName dataDir GMdir BraceMat
    global in kip sec LuniTXT FuniTXT TuniTXT ft ksi psi lbf pcf in2 in4 cm PI g DBL_MAX DBL_MIN LCol
LBeam LBrace1 LBRBYield
    global IDCtrlNode IDCtrlDOF iSupportNode LBuilding Fy Es nu Gs AgCol IzCol AgBeam IzBeam
IzBrace EABeam EACol
    global QBeam QdlBeam QdlCol WeightCol WeightBeam BaysIn BaysAlong iFloorWeight WeightTotal
sumWiHi MassTotal
    global NBay BayH BayW NumStories N0col N0beam N0brace ColNode Fybrace Esbrace nubrace
Gsbrace k1 MassNode
    global dcol twcol bfcol tfcol dbeam twbeam bfbeam tfbeam nfdw nftw nfbf nftf ChangeFloors

    # SET UP -----------------------------------------------------------------------------
        wipe                        # clear memory of all past model definitions
        model basic -ndm 2 -ndf 3;    # Define the model builder, ndm=#dimension, ndf=#dofs
        set dataDir Data;            # set up name for data directory
        file mkdir $dataDir/$FileName/$ii/ ;  # create data directory
        source DisplayPlane_DisplayModel2D.tcl ;# procedure for displaying a plane in model and a
                            # procedure for displaying 2D perspective of model


    # define GEOMETRY -------------------------------------------------------------

        # This is formed at each "level"

        # |------------------------|
        # |                      X|
        # |                    X  |
        # |                  X    |
        # |                X      |
        # |              X        |
        # |            X          |
        # |          X            |
        # |        X              |
        # |      X                |
        # |    X                  |
        # |  X                    |
        # |X                      |


        # nodal coordinates:
```

```
        set NumNodes 0
        set ColNode 40000

        for {set level 1} {$level <= [expr $NumStories+1]} {incr level} {
            set Y [expr ($level-1)*$LCol];
            # puts $Y
            for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier} {
                set X [expr ($pier-1)*$LBeam]
                set nodeID [expr $ColNode+$level*100+$pier*10];      # number for bottom node
                node $nodeID $X $Y;                    # actually define bottom node
                # puts "$nodeID"
                incr NumNodes 1
            }
        }

        # puts "Number of Nodes: $NumNodes"

    # Single point constraints -- Boundary Conditions
        fixY 0.0 1 1 0;      # pin all Y=0.0 nodes

    # determine support nodes where ground motions are input, for multiple-support excitation
        set iSupportNode ""
        set level 1
        for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier} {
            set nodeID [expr $ColNode+$level*100+$pier*10]
            lappend iSupportNode $nodeID
        }

    # Set up parameters that are particular to the model for displacement control
        set IDctrlNode [expr $ColNode+$NumStories*100+1*10]; # node where displacement is read if
displacement control is used
        set IDctrlDOF 1;                    # degree of freedom of displacement read for displacement
control
        set LBuilding [expr $NumStories*$LCol];            # total building height

    # define UNIAXIAL materials - Structural-Steel properties----------------------------------------------------------------
----------------------------------------------------------

    # Material properties, column and beam sections all defined in Input.tcl

        set BCMat 10
        set BraceMat 20
        set b_BC 0.01

        # $R0, $cR1, $cR2 control the transition from elastic to plastic branches.
        # Recommended values:
        # $R0=between 10 and 20, $cR1=0.925, $cR2=0.15
        set R0_BC 20
        set cR1_BC 0.925
        set cR2_BC 0.15

        # Beam and Column Materials
        uniaxialMaterial Steel02 $BCMat $Fy $Es $b_BC $R0_BC $cR1_BC $cR2_BC

        # Brace Materials

        set b_Brace 0.025
        # Parameters used in the Giuffré-Menegotto-Pinto equations
            set R0_Brace 1.95;    # exponent that controls the transition between elastic and hardening
branch
            set cR1_Brace 0.001;     # parameter for the change of R with cyclic loading history
            set cR2_Brace 0.001;     # parameter for the change of R with cyclic loading history
```

```
        uniaxialMaterial Steel02 $BraceMat [expr $Fybrace*1.65] [expr $k1] $b_Brace $R0_Brace
$cR1_Brace $cR2_Brace

        # puts "Materials Defined"

    # define SECTIONS-------------------------------------------------------------------------------------------------------------
----------------------

        #Beam and Column Sections

            set ColSecTag 1
            set BeamSecTag 2
            set listnum 0
            for {set level 1} {$level <= $NumStories} {incr level} {
                Wsection $ColSecTag$level $BCMat [lindex $dcol $listnum] [lindex $twcol $listnum] [lindex
$bfcol $listnum] [lindex $tfcol $listnum] $nfdw $nftw $nfbf $nftf
                # puts "$level $listnum [lindex $EACol $listnum] [lindex $dcol $listnum] [lindex $twcol
$listnum] [lindex $bfcol $listnum] [lindex $tfcol $listnum]"
                if {[expr $level % $ChangeFloors] == 0} {incr listnum 1};
            }
            Wsection $BeamSecTag $BCMat $dbeam $twbeam $bfbeam $tfbeam $nfdw $nftw $nfbf $nftf
            # puts "Sections Defined"

    # define ELEMENTS-----------------------------------------------------------------------------

        # set up geometric transformations of element
        #   separate columns and beams, in case of P-Delta analysis for columns
            set IDColTransf 1;     # all columns
            set IDBeamTransf 2;  # all beams
            set IDBraceTransf 3;  # all braces
            set ColTransfType Corotational;           # options, Linear PDelta Corotational
            geomTransf $ColTransfType $IDColTransf
            geomTransf Corotational $IDBeamTransf
            geomTransf Corotational $IDBraceTransf

        # Define Beam-Column Elements
            set np 3;    # number of Gauss integration points for nonlinear curvature distribution
            set NumElem 0
            set NumCol 0
            set NumBeam 0
            set NumBrace 0

            # columns

            set N0col 1000;    # column element numbers
            set listnum 0
            set level 0
            for {set level 1} {$level <= $NumStories} {incr level 1} {
                for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier 1} {
                    set elemID [expr $N0col+$level*10+$pier]
                    set nodeI [expr $ColNode+$level*100+$pier*10]
                    set nodeJ [expr $ColNode+($level+1)*100+$pier*10]
                    element nonlinearBeamColumn $elemID $nodeI $nodeJ $np $ColSecTag$level
$IDColTransf; # columns
                    incr NumElem
                    incr NumCol
                }
            }

            # puts "Columns Constructed"

            # beams

            set N0beam 2000;     # beam element numbers
```

```
            for {set level 1} {$level <= $NumStories} {incr level 1} {
                for {set bay 1} {$bay <= $NBay} {incr bay 1} {
                    set elemID [expr $N0beam + $level*10 +$bay]
                    set nodeI [expr $ColNode + ($level+1)*100 + $bay*10]
                    set nodeJ [expr $ColNode + ($level+1)*100 + ($bay+1)*10]
                    element nonlinearBeamColumn $elemID $nodeI $nodeJ $np $BeamSecTag
$IDBeamTransf;  # beams
                    incr NumElem
                    incr NumBeam
                }
            }

            # puts "Beams Constructed"


        # Define corotTruss Elements - braces

            set N0brace 3000;     # brace element numbers
            for {set level $NumStories} {$level >= 1} {incr level -1} {
                for {set pier 1} {$pier <= 1} {incr pier} {
                    set elemID [expr $N0brace+$level*10+$pier]
                    set nodeI [expr $ColNode+$level*100+$pier*10]
                    set nodeJ [expr $ColNode+($level+1)*100+($pier+1)*10]
                    if {$level == $NumStories} {
                        set elem $cdv($ii,$level)
                    } else {
                        set elem [expr $elem+$cdv($ii,$level)]
                    }
                    # puts "$level $elem $cdv($ii,$level)"
                    element corotTruss $elemID $nodeI $nodeJ [expr $elem*pow($in,2)] $BraceMat
                    # element corotTruss $elemID $nodeI $nodeJ [expr 25.0*pow($in,2)] $BraceMat; # test
brace
                    incr NumElem
                    incr NumBrace
                }
            }

        # puts "Number of Elements: $NumElem"
        # puts "Number of Column Elements: $NumCol"
        # puts "Number of Beam Elements: $NumBeam"
        # puts "Number of Brace Elements: $NumBrace"


    # Define GRAVITY LOADS, weight and masses

        # calculate dead load of frame
        # distributed weight along the beam length defined in Input procedure

        # assign masses to the nodes that the columns are connected to
        # each connection takes mass from the distributed weight of the beam

        set iFloorWeight ""
        set WeightTotal 0.0
        set sumWiHi 0.0;       # sum of story weight times height, for lateral-load distribution
        set listnum 0
        for {set level 2} {$level <= [expr $NumStories+1]} {incr level 1} { ;
            set FloorWeight 0.0
            if {$level == [expr $NumStories+1]} {
                set ColWeightFact 1;       # one column in top story
            } else {
                set ColWeightFact 2;       # two columns elsewhere
            }
            for {set pier 1} {$pier <= [expr $NBay+1]} {incr pier 1} {;
                set weight [lindex $WeightBeam [expr $level-2]]
```

```
                set weight2 [lindex $WeightCol $listnum]
                # puts "Weight from Col:$weight2 Weight from floor:$weight"

                set WeightNode [expr $ColWeightFact*$weight2+$weight/2]
                set MassNode [expr $WeightNode/$g];
                set nodeID [expr $ColNode+$level*100+$pier*10]; # master node on all other floors is bottom
of column
                # puts "    Mass @ Node $nodeID: $MassNode $WeightNode"
                mass $nodeID $MassNode 0.0 0.0;       # define mass - units: kip-s^2/in

                set FloorWeight [expr $FloorWeight+$WeightNode];
                if {[expr $level % $ChangeFloors] == 1 && $level != 2} {incr listnum}
            }
            lappend iFloorWeight $FloorWeight
            set WeightTotal [expr $WeightTotal+ $FloorWeight]
            set sumWiHi [expr $sumWiHi+$FloorWeight*($level-1)*$LCol];    # sum of story weight times
height, for lateral-load distribution
        }
        set MassTotal [expr $WeightTotal/$g];
        # puts "    Total Building Mass: $MassTotal"



    # Define DISPLAY ------------------------------------------------------------
        # DisplayModel2D NodeNumbers

    # puts "Done with AppInitialize"
}
```

A.11.3 AppAnalysis.Mono

```
# AppAnalysis procedure

proc AppAnalysis {ii i} {

    global nobj cdv obj feasible FileName dataDir GMfile GMfact DBL_MAX NBay NumStories LCol LBeam
LBrace1 ngener
    global YieldDrift ElasticDrift AllowDuct AllowDrift AvgDrift Ductility PI period IDLoadTagGrav
IDLoadTagDyn
    global designtype

    set IDLoadTagGrav 500
    set IDLoadTagDyn 400;  # for uniformSupport excitation

    # Calculate the building period
        set nEigenI 1;      # mode 1
        set nEigenJ 3;      # mode 3
        set nEigenK $NumStories;    # mode for number of stories
        set lambdaN [eigen [expr $nEigenK]];              # eigenvalue analysis for nEigenJ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j
        set omegaI [expr pow($lambdaI,0.5)];
        set omegaJ [expr pow($lambdaJ,0.5)];
        set period($ii) [expr 2*$PI/$omegaI];             # 1st mode period
        set period3($ii) [expr 2*$PI/$omegaJ];            # 3rd mode period
        puts "     Period:   Mode 1: $period($ii)"
        puts "               Mode 3: $period3($ii)"

    set j 1
    set GMfact($ii) "";
    # puts $GMfact($ii);            # ground-motion scaling factor reset
    foreach GM $GMfile {
```

```
        Scale $period($ii) $j $ii;   # scale all the earthquakes to this design
        incr j
    }

    set j 0
    for {set level 1} {$level <= $NumStories} {incr level} {
        set TotDrift($level) 0.0
        set TotDeformation($level) 0.0
        # puts $TotDrift($level) $TotDeformation($level)
    }

    # puts $GMfact($ii)
    foreach GM $GMfile GMfac $GMfact($ii) {
        set feasible($ii) 0.0
        puts "    Earthquake #[expr $j+1]"
        reset
        wipeAnalysis
        remove recorders
        puts "     $GM $GMfac"
        GravityAnalysis $ii $j
        DynamicAnalysis $ii $GM $GMfac $j
        # print "$dataDir/$FileName/$ii/ModelParameters Eq[expr $j+1].out"
        remove loadPattern [expr $IDLoadTagGrav+$j]
        remove loadPattern [expr $IDLoadTagDyn+$j]

        # Calculate ductility from brace deformation
            puts "     Calculating ductility..."
            for {set level 1} {$level <= $NumStories} {incr level} {
                set Ductility($ii,$level) -$DBL_MAX
            }

            # Method from Graham Oxborrow on 4/2/09

            for {set level 1} {$level <= $NumStories} {incr level} {
                set Deformation($ii,$level,[expr $j+1]) -$DBL_MAX
                if {[catch {open "$dataDir/$FileName/$ii/BraceDeform$level Eq[expr $j+1].out" r 0666} fileID]}
{
                    puts "Cannot open $dataDir/$FileName/$ii/BraceDeform$level Eq[expr $j+1].out"
                } else {
                    foreach line [split [read $fileID] \r\n] {
                        set bracedeform [split $line]
                        set x [lindex $bracedeform 1]
                        if {$x != ""} {
                            set brdf [expr abs($x)]
                            if {$brdf > $Deformation($ii,$level,[expr $j+1])} {
                                set Deformation($ii,$level,[expr $j+1]) $brdf;
                                # this will turn out to be the max deformation on that level
                            }
                        }
                    }
                }
                # puts "Level $level Deformation: $Deformation($ii,$level,[expr $j+1])"
                close $fileID
            }
            for {set level 1} {$level <= $NumStories} {incr level} {
                set TotDeformation($level) [expr $TotDeformation($level)+$Deformation($ii,$level,[expr
$j+1])]
                set AvgDeformation($ii,$level) [expr $TotDeformation($level)/($j+1)]
                # puts "$Deformation($ii,$level,[expr $j+1]) $TotDeformation($level) [expr $j+1]
$AvgDeformation($ii,$level)"
            }
            for {set level 1} {$level <= $NumStories} {incr level} {
                set Ductility($ii,$level) [expr ($AvgDeformation($ii,$level))/$ElasticDrift]
                # puts "$Ductility($ii,$level) $ElasticDrift $AvgDeformation($ii,$level)"
```

```
            }

        # Calculate drift from brace deformation
            for {set level 1} {$level <= $NumStories} {incr level} {
                set LBraceNew [expr $LBrace1+$AvgDeformation($ii,$level)]
                set AngleH [expr acos((pow($LBeam,2)+pow($LBraceNew,2)-
pow($LCol,2))/(2*$LBeam*$LBraceNew))]
                set AngleL [expr acos((pow($LCol,2)+pow($LBraceNew,2)-
pow($LBeam,2))/(2*$LCol*$LBraceNew))]
                set DriftAngle [expr $PI/2-$AngleH-$AngleL]
                set AvgDrift($ii,$level) [expr sin($DriftAngle)]
                # puts "$DriftAngle $AngleH $AngleL $AvgDrift($ii,$level)"
            }

        # Determine feasibility value

            for {set level 1} {$level <= $NumStories} {incr level} {

                # Normalize constraints so that they can be compared
                    set drift [expr ($AvgDrift($ii,$level)-$AllowDrift)/$AllowDrift]
                    set duct [expr ($Ductility($ii,$level)-$AllowDuct)/$AllowDuct]
                    # puts "$drift $duct"

                # Determine feasibility
                    if {$drift > $duct && $drift > 0.0} {
                        set feasible($ii) [expr $feasible($ii)+$drift]
                        # puts $drift
                        # puts "      Feasible $feasible($ii)"
                    } elseif {$duct > 0.0} {
                        set feasible($ii) [expr $feasible($ii)+$duct]
                        # puts $duct
                        # puts "      Feasible $feasible($ii)"
                    }
            }
            puts "     Feasiblity $feasible($ii)"
        if {$feasible($ii) > 0.0 && [expr $j+1] >= 3 && $designtype == "Optimize"} {puts "         !!!Infeasible
design!!!"; break}
        incr j
        if {$i < [expr 0.8*$ngener] && $j == 3} {break}
    }

    set TotNum $j
    # puts "Number of Earthquakes: $TotNum"

    # Calculating the objective - minimum cost
        puts "    Calculating the objective..."
        for {set k 1} {$k <= $nobj} {incr k} {
            set obj($ii,$k) 0.0
            set elem 0.0
            for {set level $NumStories} {$level >= 1} {incr level -1} {
                set elem [expr $elem+$cdv($ii,$level)]
                set obj($ii,$k) [expr $obj($ii,$k)+$elem] ;     #the first cdv is the area of the brace
                # puts "$obj($ii,$k) $elem $cdv($ii,$level)"
            }
        }
}
```

A.11.4 Input.Mono

```
# Input File

proc Input {} {
```

```
    #Access the global variables

    global FileName nddv ncdv nobj nsize ngener ntourn probcross dcrosspar ccrosspar probmutate
    global dmutpar cmutpar dmin dmax cmin cmax Designfile GMdir GMfile NBay BayH BayW
    global NumStories in kip sec LuniTXT FuniTXT TuniTXT ft ksi psi lbf psf pcf in2 in4
    global cm PI g DBL_MAX DBL_MIN Fy Es nu Gs Fybrace Esbrace nubrace Gsbrace designtype
    global nfdw nftw nfbf nftf QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol QBeam AgBeam
    global IzBeam EABeam LCol LBeam LBrace1 LBRBYield k1 BaysIn BaysAlong WeightCol WeightBeam
QdlBeam
    global YieldDrift ElasticDrift AllowDuct AllowDrift dbeam twbeam bfbeam tfbeam xDamp ChangeFloors
TmaxAnalysis

    # define OPTIMIZATION parameters

    # Single Variables
        set FileName "Nine_Stories" ;
        # Choices: Three_Stories, Six_Stories, Nine_Stories, Twelve_Stories, Eighteen_Stories,
Nine_Stories_SameCol
        set designtype "Optimize"; # Can be used to define what type of analysis mind "Optimize" "ELF"
        # set designtype "ELF";     # If the designtype is ELF then the brace areas need to be set in the Main
procedure
        set nddv 0
        set ncdv 9;          # this corresponds to the number of stories in the building
        set nobj 1;      # this objective is to minimize the area
        set nsize 50;        # this number needs to be even
        set ngener 700;         # number of generations in the optimization
        set ntourn 6;        # number of parents selected to generate the children
        set probcross 0.6;# probability crossover will occur during creation of child designs
        set dcrosspar 0.0
        set ccrosspar 1.0;# 0.0 = uniform crossover; 1.0 = blend crossover
        set probmutate 0.4;    # probability that mutation in the design will occur
        set dmutpar 0.0
        set cmutpar 1.0;   # 0.0 = uniform mutation; >0.0 = dynamic mutation

    # Set variables for ELF procedure

        if {$designtype == "ELF"} {
            set nsize 1
            set ngener 1
        }

    # Set minimum and maximum value for the design variables
        # Discrete design variables
        #    if {$nddv != 0} {
        #        for {$nddv != 0} {$v <= $ncdv} {incr v} {
        #            set dmin($v) 0
        #            set dmax($v) 10
        #        }
        #    }

        # Continuous design variables
            if {$ncdv != 0} {
                set cmin($ncdv) 0.0
                set cmax($ncdv) 30.0
                for {set v 1} {$v <= [expr $ncdv-1]} {incr v} {
                    set cmin($v) 0.0
                    set cmax($v) 5.0
                }
            }

    # define UNITS ----------------------------------------------------------------------------
        set in 1.;                  # define basic units -- output units
        set kip 1.;                 # define basic units -- output units
```

```
        set sec 1.;                     # define basic units -- output units
        set LunitTXT "inch";            # define basic-unit text for output
        set FunitTXT "kip";             # define basic-unit text for output
        set TunitTXT "sec";             # define basic-unit text for output
        set ft [expr 12.*$in];          # define engineering units
        set ksi [expr $kip/pow($in,2)];         # kips per square inch
        set psi [expr $ksi/1000.];      # pounds per square inch
        set lbf [expr $psi*$in*$in];        # pounds force
        set psf [expr $lbf/(pow($ft,2))];   # pounds per square foot
        set pcf [expr $lbf/pow($ft,3)];     # pounds per cubic foot
        set in2 [expr $in*$in];         # inch^2
        set in4 [expr $in*$in*$in*$in];     # inch^4
        set cm [expr $in/2.54];             # centimeter
        set PI [expr 2*asin(1.0)];      # define constants
        set g [expr 32.2*$ft/pow($sec,2)];    # gravitational acceleration
        set DBL_MAX 1.e10;              # a really large number
        set DBL_MIN [expr 1.0/$DBL_MAX];        # a really small number


    # Earthquakes that are to be used in the analysis ------------------------------------------
    # this will be a list that will be created to cycle through on each design
        set Designfile $FileName;     # true if the design spectra file has same name as results files will have
        set GMdir "GMfiles"   ;     # ground-motion file directory
        if {$ncdv < 12} {
            # set Suite [list "CNP196"]
                # Single earthquake test
            set Suite [list "CNP196" "MUL279" "WIL180" "RO3090" "HDA255" "HCH180" "G03090" "G04090"
"B-PTS225" "B-PTS315"];
                # Initial 3- to 9-story earthquake suite
            # set Suite [list "ORR090" "WBA090" "JAB220" "TUJ262" "CYC195" "HVR090" "FRE090"
"AND270" "B-PLS135" "B-IVW360"];
                # Secondary 3- to 9-story earthquake suite
        } else {
            # set Suite [list "DZC270"]
                # Single earthquake test
            set Suite [list "DZC270" "WPI046" "TCU050-N" "CHY101-N" "YPT060" "YPT330" "LCN275" "H-
E04230" "H-E05230" "H-E06230"];
                # Intial 12- to 18-story earthquake suite
        }
        set GMfile $Suite ;             # ground-motion filenames
        set TmaxAnalysis [expr 20.*$sec];   # maximum duration of ground-motion analysis
        set xDamp 0.02;                 # damping ratio

        # Suite 1 (3- to 9-stories)
            # 1994 Northridge (90053 Canoga Park - Topanga Can) CNP196
            # 1994 Northridge (90013 Beverly Hills - 14145 Mulhol) MUL279
            # 1994 Northridge (90018 Hollywood - Willoughby Ave) WIL180
            # 1994 Northridge (90006 Sun Valley) RO3090
            # 1989 Loma Prieta (1656 Hollister Diff. Array) HDA255
            # 1989 Loma Prieta (1028 Hollister City Hall) HCH180
            # 1989 Loma Prieta (Gilroy Array #3) G03090
            # 1989 Loma Prieta (Gilroy Array #4) G04090
            # 1987 Superstition Hills (5051 Parachute Test Site) B-PTS225
            # 1987 Superstition Hills (5051 Parachute Test Site) B-PTS315

        # Suite 2 (3- to 9-stories)
            # 1994 Northridge (24278 Castaic - Old Ridge Route) ORR090
            # 1994 Northridge (90088 Anaheim - W Ball Rd) WBA090
            # 1994 Northridge (90094 Bell Gardens - Jaboneria) JAB220
            # 1994 Northridge (90061 Big Tujunga, Angeles Nat F) TUJ262
            # 1989 Loma Prieta (57217 Coyote Lake Dam (SW Abut)) CYC195
            # 1989 Loma Prieta (57191 Halls Valley) HVR090
            # 1989 Loma Prieta (57064 Fremont - Mission San Jose) FRE090
            # 1989 Loma Prieta (1652 Anderson Dam (Downstream)) AND270
```

```
        # 1987 Superstition Hills (5052 Plaster City) B-PLS135
        # 1987 Superstition Hills (5210 Wildlife Liquef. Array) B-IVW360

    # Suite 3 (12- to 18-stories)
        # 1999 Duzce, Turkey (Duzce) DZC270
        # 1999 Chi-Chi, Taiwan (TCU050) TCU050-N
        # 1999 Chi-Chi, Taiwan (CHY101) CHY101-N
        # 1999 Kocaeli, Turkey (Yarimca) YPT060
        # 1999 Kocaeli, Turkey (Yarimca) YPT330
        # 1994 Northridge (90056 Newhall - W. Pico Canyon Rd.) WPI046
        # 1992 Landers (24 Lucerne) LCN275
        # 1979 Imperial Valley (El Centro Array #4) H-E04230
        # 1979 Imperial Valley (El Centro Array #5) H-E05230
        # 1979 Imperial Valley (El Centro Array #6) H-E06230


# Input Parameters for building model ----------------------------------------
    set NBay 1;         # Number of bays
    set BayH 15.;       # bay height - standard input parameter (ft)
    set BayW 25.;       # bay width - standard input parameter (ft)
    set BaysIn 4. ;     # number of bays perpendicular to brace
    set BaysAlong 4. ;     # number of bays in line of brace
    set NumStories $ncdv;    # number of stories in building
    puts ""
    puts "Number of Stories: $NumStories Number of Bays: $NBay"

# Structural-Steel material properties ---------------------------------

    # beam and column materials
        set Fy [expr 50.0*$ksi];        # Yield stress
        set Es [expr 29000.*$ksi];          # Steel Young's Modulus
        set nu 0.3;                 # Poisson's ratio
        set Gs [expr $Es/2./[expr 1+$nu]];    # Torsional stiffness Modulus

    # brace materials
        set Fybrace [expr 45.0*$ksi];      # Yield stress
        set Esbrace [expr 29000.*$ksi];       # Steel Young's Modulus
        set nubrace 0.3;              # Poisson's ratio
        set Gsbrace [expr $Esbrace/2./[expr 1+$nubrace]]; # Torsional stiffness modulus

    # column sections

        set QdlCol "";  set dcol ""
        set AgCol "";   set twcol ""
        set IzCol "";    set bfcol ""
        set EACol "";   set tfcol ""

        set nfdw 4;     # number of fibers along web depth
        set nftw 2; # number of fibers along web thickness
        set nfbf 4;  # number of fibers along flange width
        set nftf 2;  # number of fibers along flange thickness

        set ChangeFloors 2;  # column properties change every this number of floors

        Column_$FileName;  # procedure that inputs the column geometric properties

    # beam sections: W14x48

        set QBeam [expr 48.0*$lbf/$ft];             # W-section weight per length
        set AgBeam [expr 14.1*pow($in,2)];      # cross-sectional area
        set IzBeam [expr 484.0*pow($in,4)];     # moment of Inertia
        set EABeam [expr $Es*$AgBeam];              # EA, for axial-force-strain relationship

        set dbeam [expr 13.8*$in];    # nominal depth
```

```
        set twbeam [expr 0.340*$in];# web thickness
        set bfbeam [expr 8.03*$in];   # flange width
        set tfbeam [expr 0.595*$in];  # flange thickness

    # define GEOMETRY -------------------------------------------------------------
        set LCol [expr $BayH*$ft];                  # column length
        set LBeam [expr $BayW*$ft];                     # beam length
        set LBrace1 [expr sqrt(pow($LCol,2)+pow($LBeam,2))];    # brace overall length
        set LBrace2 [expr $dbeam/($LCol/$LBrace1)+24*$in]
        set LBRBYield [expr ($LBrace1-2*$LBrace2)*0.85]
        set k1 [expr ($Esbrace*$LBrace1/$LBRBYield)]
        # puts "$LBrace1 $LBrace2 $LBRBYield"

    # Define GRAVITY LOADS, weight and masses -------------------------------------

        # calculate distributed weight along the beam length

        # This is a sample of the floor layout
        # As BayIn and BaysAlong increase the bays size of the building changes
        # It is assumed that there is only one brace in the center of each exterior wall

        #   Bay 1   Bay 2   Bay 3  etc.
        # |-------|-------|-------|
        # |       |       |       |
        # |       |       |       | Bay 1
        # |       |       |       |
        # |-------|-------|-------|
        # |       |       |       |
        # |       |       |       | Bay 2   --- Floor Area - for mass associated with brace
        # |       |       |       |          |
        # |-------|-------|-------|          |
        # |       |       |       |          v
        # |       |       |       | Bay 3
        # |       |       |       |
        # |-------|-------|-------| etc.


        set FloorWeight [expr 90.0*$psf];               # floor dead load in psf
        set RoofWeight [expr 90.0*$psf];          # roof dead load in psf
        set FloorArea [expr $BaysIn/2*$BaysAlong*pow($LBeam,2)];# the tributary area associated with one
bay

        set WeightCol ""
        set WeightBeam ""
        set QdlBeam ""
        set listnum 0
        for {set level 1} {$level <= $NumStories} {incr level} {
            lappend WeightCol [expr $LCol*[lindex $QdlCol $listnum]];  # total Column weight
            if {$level == $NumStories} {
                set weight $RoofWeight
            } else {
                set weight $FloorWeight
            }
            lappend QdlBeam [expr $QBeam+$weight*($LBeam/2)];       # distributed gravity load on the
beam
            lappend WeightBeam [expr $QBeam*$LBeam+$FloorArea*$weight];     # total seismic weight
applied to beam
            # puts "$BaysIn $BaysAlong $LBeam $FloorArea $weight [expr $FloorArea*$weight]"
            if {[expr $level % $ChangeFloors] == 0} {incr listnum 1}
        }
        # puts "Beam Distributed Load: $QdlBeam Column Weight: $WeightCol Beam Weight:
$WeightBeam"
```

```
    # define CONSTRAINTS for OPTIMIZATION -------------------------------------------

        # calculate the drift at yield to later determine if ductility demand is less than ductility capacity

            set YieldDrift [expr ($Fybrace*$LBrace1)/(($LBeam/$LBrace1)*$Esbrace)]
            set ElasticDrift [expr ($Fybrace*$LBrace1)/$Esbrace]
            set AllowDuct [expr 7.0/2.0]; # This value is determined from ASCE 7-05 (R/overstrength factor)
            set AllowDrift [expr 0.015]
            # puts "Frame Drift: $YieldDrift Brace Elongation: $ElasticDrift
            # puts "Fy: $Fybrace Lbr: $LBrace1 Lbm: $LBeam Ebr: $Esbrace"
            # puts "Ductility Limit: $AllowDuct Drift Limit: $AllowDrift"

    # Other input parameters may exist -- find out what they are and include them

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Three_Stories {} {;

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W12x96

    set QdlCol [expr 96.0*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 28.2*pow($in,2)];    # cross-sectional area
    set IzCol [expr 833.*pow($in,4)]; # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 12.7*$in];   # nominal depth
    set twcol [expr 0.550*$in];    # web thickness
    set bfcol [expr 12.2*$in];  # flange width
    set tfcol [expr 0.90*$in];   # flange thickness

    # top floors: W12x45

    lappend QdlCol [expr 45.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 13.1*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 348.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 12.1*$in]; # nominal depth
    lappend twcol [expr 0.335*$in];   # web thickness
    lappend bfcol [expr 8.05*$in];    # flange width
    lappend tfcol [expr 0.575*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Six_Stories {} {;

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x176

    set QdlCol [expr 176.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 51.8*pow($in,2)];    # cross-sectional area
    set IzCol [expr 2140.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 15.2*$in];   # nominal depth
```

```
    set twcol [expr 0.830*$in];     # web thickness
    set bfcol [expr 15.7*$in];  # flange width
    set tfcol [expr 1.31*$in];   # flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];   # web thickness
    lappend bfcol [expr 14.7*$in];     # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x68

    lappend QdlCol [expr 68.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 20.0*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 722*pow($in,4)];     # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];          # EA, for axial-force-strain relationship

    lappend dcol [expr 14.0*$in];# nominal depth
    lappend twcol [expr 0.415*$in];   # web thickness
    lappend bfcol [expr 10.0*$in];     # flange width
    lappend tfcol [expr 0.720*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------------

proc Column_Nine_Stories {} {

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x283

    set QdlCol [expr 283.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 83.3*pow($in,2)];     # cross-sectional area
    set IzCol [expr 3840.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 16.7*$in];   # nominal depth
    set twcol [expr 1.29*$in]; # web thickness
    set bfcol [expr 16.1*$in];  # flange width
    set tfcol [expr 2.07*$in];   # flange thickness

    # next set of floors up: W14x193

    lappend QdlCol [expr 193.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 56.8*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 2400.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 15.5*$in];# nominal depth
    lappend twcol [expr 0.890*$in];   # web thickness
    lappend bfcol [expr 15.7*$in];     # flange width
    lappend tfcol [expr 1.44*$in];# flange thickness

    # next set of floors up: W14x132
```

```
    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];  # web thickness
    lappend bfcol [expr 14.7*$in];     # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # next set of floors up: W14x74

    lappend QdlCol [expr 74.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 21.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 795.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.2*$in];# nominal depth
    lappend twcol [expr 0.450*$in];  # web thickness
    lappend bfcol [expr 10.1*$in];     # flange width
    lappend tfcol [expr 0.785*$in];    # flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];  # web thickness
    lappend bfcol [expr 8.03*$in];     # flange width
    lappend tfcol [expr 0.595*$in];    # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------------------

proc Column_Twelve_Stories {} {

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x398

    set QdlCol [expr 398.*$lbf/$ft];    # W-section weight per length
    set AgCol [expr 117*pow($in,2)];# cross-sectional area
    set IzCol [expr 6000.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 18.3*$in];  # nominal depth
    set twcol [expr 1.77*$in]; # web thickness
    set bfcol [expr 16.6*$in];  # flange width
    set tfcol [expr 2.85*$in];  # flange thickness

    # next set of floors up: W14x311

    lappend QdlCol [expr 311.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 91.4*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 4330.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 17.1*$in];# nominal depth
```

```
    lappend twcol [expr 1.41*$in];     # web thickness
    lappend bfcol [expr 16.2*$in];     # flange width
    lappend tfcol [expr 2.26*$in];# flange thickness

    # next set of floors up: W14x233

    lappend QdlCol [expr 233.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 68.5*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 3010.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 2]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 16.0*$in];# nominal depth
    lappend twcol [expr 1.07*$in];     # web thickness
    lappend bfcol [expr 15.9*$in];     # flange width
    lappend tfcol [expr 1.72*$in];# flange thickness

    # next set of floors up: W14x145

    lappend QdlCol [expr 145.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 42.7*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1710.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 3]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.8*$in];# nominal depth
    lappend twcol [expr 0.680*$in];   # web thickness
    lappend bfcol [expr 15.5*$in];     # flange width
    lappend tfcol [expr 1.09*$in];# flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 4]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];   # web thickness
    lappend bfcol [expr 14.7*$in];     # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];    # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];    # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 5]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];   # web thickness
    lappend bfcol [expr 8.03*$in];     # flange width
    lappend tfcol [expr 0.595*$in];    # flange thickness

    # puts $dcol


}

# ------------------------------------------------------------------------------------------------------------------------

proc Column_Eighteen_Stories {} {; # not ready to run yet

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x665
```

```
set QdlCol [expr 665.*$lbf/$ft];    # W-section weight per length
set AgCol [expr 196*pow($in,2)];# cross-sectional area
set IzCol [expr 12400.*pow($in,4)];  # moment of Inertia
set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

set dcol [expr 21.6*$in];   # nominal depth
set twcol [expr 2.83*$in]; # web thickness
set bfcol [expr 17.7*$in];  # flange width
set tfcol [expr 4.52*$in];   # flange thickness

# next set of floors up: W14x550

lappend QdlCol [expr 550.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 162.*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 9430.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

lappend dcol [expr 20.2*$in];# nominal depth
lappend twcol [expr 2.38*$in];     # web thickness
lappend bfcol [expr 17.2*$in];     # flange width
lappend tfcol [expr 3.82*$in];# flange thickness

# next set of floors up: W14x455

lappend QdlCol [expr 455.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 134.0*pow($in,2)];# cross-sectional area
lappend IzCol [expr 7190.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 2]];          # EA, for axial-force-strain relationship

lappend dcol [expr 19.0*$in];# nominal depth
lappend twcol [expr 2.02*$in];     # web thickness
lappend bfcol [expr 16.8*$in];     # flange width
lappend tfcol [expr 3.21*$in];# flange thickness

# next set of floors up: W14x370

lappend QdlCol [expr 370.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 109*pow($in,2)];    # cross-sectional area
lappend IzCol [expr 5440.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 3]];          # EA, for axial-force-strain relationship

lappend dcol [expr 17.9*$in];# nominal depth
lappend twcol [expr 1.66*$in];     # web thickness
lappend bfcol [expr 16.5*$in];     # flange width
lappend tfcol [expr 2.66*$in];# flange thickness

# next set of floors up: W14x283

lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
lappend EACol [expr $Es*[lindex $AgCol 4]];# EA, for axial-force-strain relationship

lappend dcol [expr 16.7*$in];# nominal depth
lappend twcol [expr 1.29*$in];     # web thickness
lappend bfcol [expr 16.1*$in];     # flange width
lappend tfcol [expr 2.07*$in];# flange thickness

# next set of floors up: W14x211

lappend QdlCol [expr 211.*$lbf/$ft];  # W-section weight per length
lappend AgCol [expr 62.0*pow($in,2)];   # cross-sectional area
lappend IzCol [expr 2660.*pow($in,4)];  # moment of Inertia
```

```
    lappend EACol [expr $Es*[lindex $AgCol 5]];# EA, for axial-force-strain relationship

    lappend dcol [expr 15.7*$in];# nominal depth
    lappend twcol [expr 0.980*$in];  # web thickness
    lappend bfcol [expr 15.8*$in];   # flange width
    lappend tfcol [expr 1.56*$in];# flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 6]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];  # web thickness
    lappend bfcol [expr 14.7*$in];   # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # next set of floors up: W14x132

    lappend QdlCol [expr 132.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 38.8*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 1530.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 7]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 14.7*$in];# nominal depth
    lappend twcol [expr 0.645*$in];  # web thickness
    lappend bfcol [expr 14.7*$in];   # flange width
    lappend tfcol [expr 1.03*$in];# flange thickness

    # top floors: W14x48

    lappend QdlCol [expr 48.*$lbf/$ft];   # W-section weight per length
    lappend AgCol [expr 14.1*pow($in,2)];  # cross-sectional area
    lappend IzCol [expr 484.*pow($in,4)];   # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 8]];         # EA, for axial-force-strain relationship

    lappend dcol [expr 13.8*$in];# nominal depth
    lappend twcol [expr 0.340*$in];  # web thickness
    lappend bfcol [expr 8.03*$in];   # flange width
    lappend tfcol [expr 0.595*$in];   # flange thickness

    # puts $dcol

}

# ------------------------------------------------------------------------------------------------------------------------------

proc Column_Nine_Stories_SameCol {} {

    global QdlCol AgCol IzCol EACol dcol twcol bfcol tfcol lbf ft in Es

    # bottom floors: W14x283

    set QdlCol [expr 283.*$lbf/$ft];   # W-section weight per length
    set AgCol [expr 83.3*pow($in,2)];    # cross-sectional area
    set IzCol [expr 3840.*pow($in,4)];    # moment of Inertia
    set EACol [expr $Es*$AgCol];# EA, for axial-force-strain relationship

    set dcol [expr 16.7*$in];  # nominal depth
    set twcol [expr 1.29*$in]; # web thickness
    set bfcol [expr 16.1*$in];  # flange width
    set tfcol [expr 2.07*$in];  # flange thickness
```

```
    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];     # web thickness
    lappend bfcol [expr 16.1*$in];     # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];     # web thickness
    lappend bfcol [expr 16.1*$in];     # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # next set of floors up: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];     # web thickness
    lappend bfcol [expr 16.1*$in];     # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # top floors: W14x283

    lappend QdlCol [expr 283.*$lbf/$ft];  # W-section weight per length
    lappend AgCol [expr 83.3*pow($in,2)];   # cross-sectional area
    lappend IzCol [expr 3840.*pow($in,4)];  # moment of Inertia
    lappend EACol [expr $Es*[lindex $AgCol 1]];# EA, for axial-force-strain relationship

    lappend dcol [expr 16.7*$in];# nominal depth
    lappend twcol [expr 1.29*$in];     # web thickness
    lappend bfcol [expr 16.1*$in];     # flange width
    lappend tfcol [expr 2.07*$in];# flange thickness

    # puts $dcol

}

# -------------------------------------------------------------------------------------------------------------------------

proc ELF {ii} {

    global cdv ncdv

    # These variables can be set to specific numbers and then analyzed

    if {$ncdv == 3} {
        set cdv($ii,1) 7.0
        set cdv($ii,2) 6.0
```

```
        set cdv($ii,3) 4.0
    } elseif {$ncdv == 6} {
        set cdv($ii,1) 9.5
        set cdv($ii,2) 9.5
        set cdv($ii,3) 8.5
        set cdv($ii,4) 7.5
        set cdv($ii,5) 5.5
        set cdv($ii,6) 3.5
    } elseif {$ncdv == 9} {
        set cdv($ii,1) 10.5
        set cdv($ii,2) 10.5
        set cdv($ii,3) 10.5
        set cdv($ii,4) 10.0
        set cdv($ii,5) 9.0
        set cdv($ii,6) 8.0
        set cdv($ii,7) 7.0
        set cdv($ii,8) 5.0
        set cdv($ii,9) 3.0
    } elseif {$ncdv == 12} {
        set cdv($ii,1) 11.5
        set cdv($ii,2) 11.5
        set cdv($ii,3) 11.5
        set cdv($ii,4) 11.0
        set cdv($ii,5) 11.0
        set cdv($ii,6) 10.5
        set cdv($ii,7) 9.5
        set cdv($ii,8) 8.5
        set cdv($ii,9) 7.5
        set cdv($ii,10) 6.0
        set cdv($ii,11) 4.5
        set cdv($ii,12) 3.0
    } elseif {$ncdv == 18} {
        set cdv($ii,1) 12.5
        set cdv($ii,2) 12.5
        set cdv($ii,3) 12.5
        set cdv($ii,4) 12.5
        set cdv($ii,5) 12.5
        set cdv($ii,6) 12.5
        set cdv($ii,7) 12.0
        set cdv($ii,8) 12.0
        set cdv($ii,9) 11.5
        set cdv($ii,10) 11.0
        set cdv($ii,11) 10.5
        set cdv($ii,12) 9.5
        set cdv($ii,13) 9.0
        set cdv($ii,14) 8.0
        set cdv($ii,15) 6.5
        set cdv($ii,16) 5.5
        set cdv($ii,17) 4.0
        set cdv($ii,18) 2.0
    }
}
```

Appendix B Equivalent Lateral Force Procedure

The following pages contain a more detailed explanation of the steps taken to determine the results of the ELF procedure. Different portions of these steps can be found in ASCE 7 (2005).

B.1 Construct Design Response Spectra

The first step in the ELF Procedure requires the assembly of the design response spectra in order to approximate the forces on the structure in the event of an earthquake. The response spectrum combines the specific geologic conditions of the site with the general characteristics of the region.

Seven specific parameters are needed to construct the spectrum: S_S, S_1, T_0, T_S, T_L, F_a, and F_v. The first two parameters, S_S and S_1, are the 0.2 s and the 1.0 s spectral response accelerations under a 5% damping ratio, respectively. The following two parameters, T_0 and T_S, are period values that will be calculated during the analysis and correspond to changes in the calculation of the response spectra. The next parameter, T_L, defines the location where the long period transition in the response spectrum commences. The final two parameters, F_a and F_v, are site coefficients that correspond to the soil properties of the site. Each of these parameters is applied to different equations to define the total response spectrum.

To begin, the mapped spectral responses for 0.2 s and 1.0s needs to be determined by Equations B–1 and B–2.

$$S_{MS} = F_a S_S \quad \text{(B-1)}$$

$$S_{M1} = F_v S_1 \quad \text{(B-2)}$$

These mapped spectral accelerations demonstrate the response spectrum that would be anticipated considering the maximum earthquake expected at the site. However, in the ELF Procedure, these spectral accelerations are reduced by 2/3 as can be seen in Equations B–3 and B–4.

$$S_{DS} = \frac{2}{3} S_{MS} \quad \text{(B-3)}$$

$$S_{D1} = \frac{2}{3} S_{M1} \quad \text{(B-4)}$$

After computing S_{DS} and S_{D1}, the periods T_0 and T_S are calculated from Equations B–5 and B–6.

$$T_0 = 0.2 \frac{S_{D1}}{S_{DS}} \quad \text{(B-5)}$$

$$T_S = \frac{S_{D1}}{S_{DS}} \quad \text{(B-6)}$$

With the basic parameters defined, the design response spectrum can be constructed and an example can be seen in Figure B-1. To completely assemble this spectrum a series of equations needs to be calculated using the design spectral acceleration values and

periods. The first equation, Equation B–7, computes the sharp positive sloping line for periods less than T_0.

$$S_a = S_{DS}\left(0.4 + 0.6\frac{T}{T_0}\right) \quad \text{(B-7)}$$

Next, for the periods greater than T_0 but less than T_S, the response acceleration is to be set equivalent to the value of S_{DS}. After the period increases past T_S, the design spectral acceleration is given by Equation B–8 while the period is less than T_L.

$$S_a = \frac{S_{D1}}{T} \quad \text{(B-8)}$$

Finally, for periods greater than T_L, a more gradual slope is attained for the spectrum using Equation B–9.

$$S_a = \frac{S_{D1}T_L}{T^2} \quad \text{(B-9)}$$

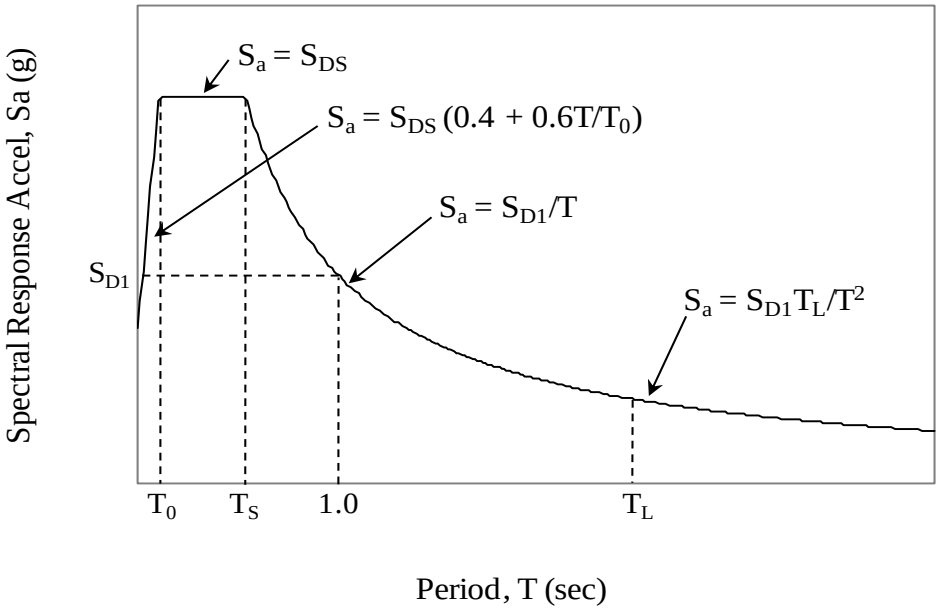

Figure B-1. Response spectra

B.2 Determine Fundamental Period of Structure

The design response spectrum predicts the acceleration of all structures at the site based on their fundamental period. Therefore, for each particular building it is necessary to calculate the fundamental period of the structure to determine the corresponding acceleration. FEMA 450 indicates that the approximate fundamental period of the structure is to be determined from Equation B–10.

$$T_a = C_r h_n^{\,x} \tag{B-10}$$

where h_n is the height in ft above the base to the highest level of the structure and C_r and x are parameters provided in FEMA 450.

Once the approximate period is determined, it needs to be multiplied by the upper limit coefficient C_u, which is dependent on the seismicity of the region. The fundamental period can then be used to obtain the specific spectral acceleration for the structure.

B.3 Determine Base Shear

Using the spectral acceleration, the base shear can be attained. FEMA 450 provides three specific equations, Equations B–11 through B–13, to determine the base shear.

$$V = C_s W \tag{B-11}$$

$$C_s = \frac{S_a/g}{(R/I)} \tag{B-12}$$

for $T \leq T_s$ and

$$C_s = \frac{S_{D1}}{T}\left(\frac{1}{R/I}\right) \quad \text{(B-13)}$$

for $T \geq T_s$.

In the preceding equations, W is the total seismic weight of the building and C_s is a calculated parameter.

B.4 Distribute Story Forces

In the ELF Procedure, the total base shear needs to be distributed to each story of the building. The distribution of forces are placed in such a fashion that the building will be pushed into its first mode shape. To distribute the base shear, use Equation B–14.

$$F_x = C_{vx} V \quad \text{(B-14)}$$

where the term C_{vx} is defined through an equation related to the height of the story from the base of the structure.

B.5 Determine Brace Core Areas

With the forces distributed to each story, the brace core areas can be determined. The braces need to remain elastic under these story shear forces. Each brace area is individually calculated and then rounded to the nearest reasonable increment for production. After determining the brace area of each floor, all the areas were summed and the percentage of distribution attributed to each floor was calculated for presentation in this study.