Topology Optimization of Building Bracing Schemes

by

Zhen John Goo

B.S., University of Minnesota, 2011

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

June 2013

Signature of Author: _____
Department of Civil and Environmental Engineering
May 10th, 2013

Certified by: _____
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by: _____
Rory Clune
Massachusetts Institute of Technology
Thesis Reader

Accepted by: _____
Heidi Nepf
Chair, Departmental Committee on Graduate Students

Topology Optimization of Building Bracing Schemes

by

Zhen John Goo

Submitted to the Department of Civil and Environmental Engineering in June, 2013, in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Civil and Environmental Engineering

Abstract

The thesis presents a technique for producing economical solutions for conventional structural steel frames using topology optimization on the lateral bracing scheme. The study focuses mainly on minimizing the amount of material used and optimizing the placement of bracing elements in typical steel frame structures to achieve economical and realistic solutions. Linear structural analysis is performed on steel frame structures while considering static gravity and wind loading. The optimization scheme uses a "multi-level design" approach with two distinct optimization loops. The optimal beam and column sizes in a structural steel frame system are generated in the first optimization loop and a bracing removal criterion is derived in the second loop to optimize the lateral bracing topology. A space constraint is imposed on the steel frame structure to enable designers to specify large empty spaces. A performance index is proposed to compare the cost between structural steel frames designed using conventional approaches, which rely on engineering experience and trial-and-error, and the approach specified in this study, which uses a multi-step optimization scheme. Two case studies are made, comparing steel frame structures designed using the proposed method with one designed using the traditional method.

Keywords: Topology optimization, steel frame optimization, space-constrained optimization, multilevel optimization

Thesis supervisor: Jerome J. Connor

Title: Professor of Civil and Environmental Engineering

Thesis reader: Rory Clune

Acknowledgements

I would like to express my deepest appreciation to all those who helped and guided me for my time at MIT. These people made my thesis possible through support, suggestions and encouragement. A special gratitude to my professor, Dr. Connor, whose contribution in providing structural engineering knowledge and general wisdom, helped me to be a better engineer and person.

Furthermore, I would also like to acknowledge my thesis advisor, Dr. Rory Clune, who provided endless guidance with patience throughout the scope of my thesis, especially in report writing. Special thanks goes my classmate, Zahraa Saiyed, who helped in editing my report. Last but not least, I would like to thank all my fellow classmates in the Master of Engineering program for being encouraging and thoughtful for the past year.

Table of Contents

Table of Figures

1.0 Chapter 1 – Introduction

1.1 Outline

This chapter introduces the design of steel frame structures with and without optimization. The chapter will discuss reasons that drive this study and briefly introduce the content of the study. The last part of this chapter gives an outline of the scope of this study.

1.2 Introduction

Steel has many desirable characteristics as one of the most common construction materials in civil structure (McCormac 2011). Some of the properties of steel that make it desirable for engineering are its malleability as well as high strength to mass ratio. Steel is used abundantly as a construction material in many low- to medium-rise buildings. Short steel frame structures are most commonly used as warehouses, factories, and housings. These buildings mainly provide shelter and personal space for individuals. Most of these buildings are built in large quantities, which make efficiency of erection a very critical component of design. Experienced designers normally generate conceptual designs for these structures from their experience. At times, this knowledge may not be applicable to new projects and at other times, designers may leave out important design factors. Most of the steel frame structures end up being designed using trial-and-error, with the help of previous experience. The derivation of economical lateral bracing schemes can be challenging due to a large number of possibilities for the arrangement of bracings. A computational optimization scheme can be used to derive economical conceptual designs of the steel frame structure is investigated in this study.

The usage of optimization schemes in civil structure design helps designers to save time, material and cost. Using optimization schemes, designers can avoid the troublesome process of deriving designs through trial-and-error. Optimal solutions should strive to be material and cost efficient. Application of optimization schemes to civil structures is a tedious process as a large amount of iteration is required to be computed before the solution converges. This is because realistic civil structures are complex problems that are highly nonlinear (Connor 2003). Every optimization

iteration requires the structural response to be computed to progress to the following iteration. When realistic civil structure models are used in optimization processes, the computation of structural response takes a long time and hence the whole optimization process becomes very time consuming. In the study, the structural analysis part of the optimization scheme is done using a computational structural analysis tool, SAP2000. In SAP2000, linear static frame analysis, one of the simplest structural analysis methods, is used in the study to save computation time (Asif 2013).

In this study, short multistory steel frame structures are investigated. The optimization scheme developed in this study adopts an approach that uses "multi-level design" from the study done by Liang (2000). The approach involves two optimization loops that optimize the gravity system of steel frame structures in the first run and optimizes the topology the lateral bracing elements in the second run. The process of deriving the conceptual design by having disintegrated gravity and lateral systems is commonly used in conventional steel frame design (McCormac 2011). The technique provides reasonable results and in some cases it gives a conservative design.

In the first part of the optimization scheme, the gravity system of the steel frame is designed assuming continuous columns and simply-supported beams. Fixed-end beams are rarely used in design due to the inefficient use of materials and costly connection design (McCormac 2011). This study investigates the effect of the placement of bracing elements, for which the objective cannot be achieved if moment frames are used. All the bracing elements in steel frame structures are essentially truss members that make up the building lateral system by acting in the axial direction only. While designing these axially loaded bracing elements, only static wind loading is taken into consideration. The assumptions for studying only static wind loading are that seismic load is controlled by dampers and dynamic wind loading does not control the design of short structures (Connor 2003). These assumptions simplify the structural analysis method used in the study. Since the process of analyzing structures takes up the largest portion of the time used for optimizing the design, reducing the effective time of each structural analysis will drastically reduce the total time of optimization.

An additional component is introduced to the optimization scheme. The new component is a space constraint that defines the coordinates of null spaces in the steel frame structure. These null spaces do not have any bracing elements running through them. The null spaces can be defined as 2D planes or as 3D void spaces. The 2D planes can only be vertical or horizontal and the 3D spaces can only be in rectangular boxes. However, designers can combine unlimited amounts of null spaces to get desired space constraints. The purpose of these null spaces is to accommodate large usable spaces or a more efficient usage of natural lighting. If a 2D null space is placed on the exterior surface of the structure (same as the case studies in this research), the structure can have unblocked window spaces to maximize the usage of natural lighting. If a 3D null space is placed in the structure, the null area will be free of all bracing elements and large usable spaces can be built.

One important feature of the study is that the steel frame structure is investigated in 3D space. This allows the investigation to capture more realistic structural responses for designing the lateral system. A 2D model cannot investigate the third dimension force effect, the coupling effect of forces in asymmetrically loaded structures (Connor 2003). When idealizing structures as a 2D model, the center of twist of a structure is assumed to be aligned with the force. This is only true for a structure with a symmetrical plan. When a structure does not have a symmetrical plan, the center of twist will be shifted according to the placement of the lateral system in the structure. A force coupling effect will be produced when asymmetrical structures are loaded laterally. The effect cannot be captured in a 2D model.

1.3 **Summary**

The main objective of this research is to derive an optimization scheme to help designers to generate economical steel frame structures in the conceptual design phase. This optimization scheme reduces the effort designers need for preliminary design. The "multi-level design" approach adopted from Liang's paper (2000) is re-created in this paper with some modifications. The approach uses two optimization loops in which the first loop generates the steel frame beam and column structure under strength constraints. The second optimization loop generates optimal lateral bracing systems in steel frame structures using a derived bracing removal criterion.

The removal criterion removes lateral bracing elements that are underutilized. The final outcomes of the optimization scheme are considered more economical as compared to designs made using conventional methods. This is proven using a performance index derived while studying the results of the study.

1.4 **Thesis Outline**

The thesis is divided into five chapters. Chapter Two covers the literature review of optimization techniques used in structural engineering. Chapter Three records the methodology used in the research in two sections. The first part describes two computational tools used in the research, MATLAB and SAP2000. The functions of each of the computational tools used in the study are explained. The second part of Chapter Three explains in detail the optimization scheme involved in the study, including the first and second optimization loop.

In Chapter Four, a performance index is developed and two case studies are conducted. In both case studies, the performance index is used as a comparison tool between the designs derived from the optimization scheme proposed in the study and the designs made using conventional methods. Finally, Chapter Five presents the findings of the study and proposes possible future work.

2.0 Chapter 2 – Literature Review

2.1 Outline

This chapter reviews the origins of optimization in structural engineering. The type of optimization done in the field of structural engineering is explored here, and the process of developing the optimization scheme in the study is discussed.

2.2 Optimization Techniques in Structural Engineering

The use of numerical optimization techniques to design in the field of structural engineering has been recorded since 1956. One of the seminal studies of optimization techniques for frame structures was done by Heyman in 1956 (Smith 2002). Optimization techniques shorten the time consumed by designers in creating better conceptual structures. Conventional methods of designing lateral bracing systems are based on trial-and-error methods with the aid of previous experience (Liang 2000). This process of designing requires a deep understanding of the nature of structures, in which case intuition from experience can sometimes be incorrect. Humans can make mistakes, and designs developed using the conventional method must be tested thoroughly to ensure safety. Much of the mentioned tedious processes involved in the conventional method of design can be mitigated using an optimization scheme.

In the field of structural engineering, optimization techniques have been used to solve many different problems, especially in design. These problems cover every aspect of structural engineering, ranging from the smallest structural elements such as a bolt up to the whole structure. These optimization techniques can be generally grouped into three distinct categories, which are cross-sectional optimization, topology optimization and geometry optimization (Smith 2002). The cross-sectional optimization focuses on sizing structural elements by assuming fixed topology and geometry. The sizing of the structural elements in this manner is approached using methods such as performance-based design or strength-based design (Connor 2003). The topology optimization studies the placement of structural elements in a design. One of the common techniques used in topology optimization is the element removal technique based on stress limits (Smith 2002). Geometry optimization is a technique that combines the usage of both the cross-sectional optimization and the topology

optimization. The "multi-level design" technique used in the study is an optimization scheme that is adopted from Liang's previous research (2000), which is categorized as geometry optimization.

Many building optimization techniques have been developed in the past years by various researches. In the past, the usage of optimization techniques in the field of structural engineering is done using alternative techniques. They change based on the objective of optimization desired by designers. In the research paper by Adeli and Karim (1997), a neural network model is used to optimize the shapes and sizes of cold-formed steel against its respective compliances. The neural dynamic model developed by Adeli and Park (used the Lyapunov function to help prove stability of the optimization process and finds the local minimum of the structural optimization problem. In another paper, Baker (1992) used energy methods to measure the efficiency of structural elements and Zhou and Rozvany (1992) improved the efficiency of optimization schemes on sizing large scale structural systems using discretized optimality criteria (Liang 2000).

All the mentioned techniques of optimization are used in the cross-sectional optimization category in the field of structural engineering. Those techniques only work if the topology of the structural system is fixed. In all the design problems, the placements of structural elements in the building design are to be determined by the designers, especially when designing the lateral system of structures. The optimization of the topology of a lateral bracing system is difficult due to the highly non-linear nature of structures and the large number of possibilities for the arrangement of bracing elements (Connor 2003). Pure rigid frames are not economical to resist lateral loading compared to diagonal bracing truss elements (McCormac 2011), but the placement of the diagonal members are crucial to make the lateral resisting system economical.

Many papers write about different methods of topology optimization on 2D structural lateral bracing schemes to achieve the same final result, for example, a paper by Liang (2011), a paper by Baldock (2006) and a paper by Kicinger (2004). A result obtained from these papers is illustrated in Figure 1. These papers mostly utilize performance

base optimization that achieves the minimum compliance in a structure under lateral loading using the minimum amount of materials. This is because the design of a multistory steel frame structure is usually controlled by performance rather than strength (Connor 2003). This proves the consistency of the different optimization techniques used in those papers. Some of these papers use only 2D models while others further develop their optimization technique to accommodate 3D space. 2D models are only able to idealize symmetrical buildings with symmetrical loads, 3D models are able to capture more realistic structural responses, such as force coupling effects from unsymmetrical loadings (Connor 2003).

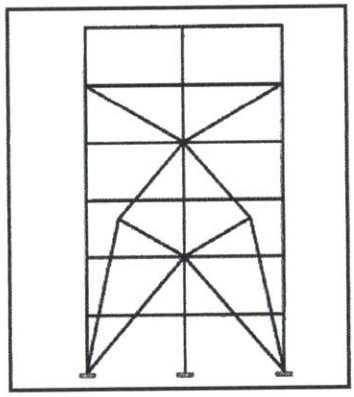

Figure 1: Optimized topology layout of a 2D steel frame structure. (Liang 2000)

In the study, only linear static wind load is investigated. The dynamic effects from wind and seismic are neglected. Dynamic loading is not being investigated in this study mainly because dynamic wind loading does not control design of short structures. This can be seen in paper by Mijar (1998), where the steel frame structures are optimized according to static compliance minimization and eigenvalue optimization. The eigenvalue optimization includes the effect of dynamic loading in the structural optimization, and the study yields the same result as Liang's paper (2000). These assumptions allow the computational time of the optimization process to be much faster.

The "multi-level design" optimization scheme in this study has been adopted from Liang's paper (2000). In Liang's paper, the optimization scheme involves two loops for steel frame structures in a linear time path. The first optimization loop designs the gravity system of steel frame structures based on strength constraints. The sized steel

frame structure from the first optimization loop is used as the starting point in the second optimization loop. The second optimization loop generates the optimal topology of a lateral bracing system for multistory steel frame structures by using a frame removal criterion. The frame removal criterion gradually removes inefficient materials with the lowest strain energy from the continuum design domain. A similar frame removal criterion is used in another paper by Querin (1999). Querin uses a more robust bi-directional evolutionary structural optimization which combines frame removal in addition to optimizing the topology of bracing systems in steel frame structures.

All papers mentioned previously state that topology optimization techniques mainly focus on deriving optimized structural bracing schemes that satisfy the building design code limits. A more adaptable optimization scheme is proposed in this paper. A space constraint is introduced in the second loop of the optimization scheme. The similar type of constraint can be found in a paper by Smith (2002). Smith proposed two different types of obstacle constraints, which are one-sided planar and spherical constraints. The one-sided planar constraint is used to keep all structural elements on one side of a planar surface defined. The spherical constraint is used to define a spherical null space with structural elements. The application of both space constraints is shown in Figure 2. These space constraints allow the designer to specify null spaces in the structure. In our study, no bracing elements will be placed in the null spaces. These null spaces allow for large areas in the structure to provide for large windows for natural lighting.

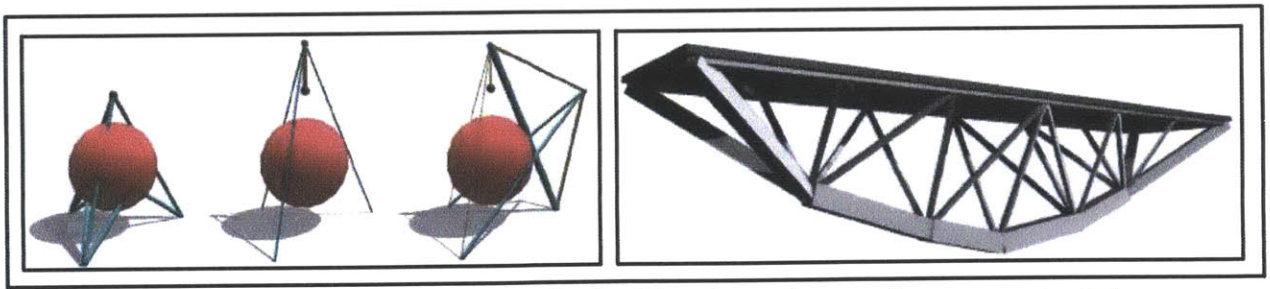

Figure 2: Left: Tripod designed using limited by the sphere volume constraint.
Right: A bridge created limiting all trusses beneath the deck. (Smith 2002)

In conclusion, many optimization research papers have been written for design of steel structures. Each of these papers has its own interpretation and implementation scheme. The variations are in terms of the optimization schemes, types of materials, types of loadings, methods of evaluating compliance, and constraints. In this study, precedent studies are incorporated into the investigation process and a logical optimization scheme is adopted. *The major contribution of this study is the addition of a space constraint that differentiates this research from previous precedents.*

3.1 Chapter 3 – Methodology

3.2 Outline

This chapter discusses the computational tools involved in the study. It begins by describing the overall flow of processes involved in optimizing the conceptual models of conventional steel frame structures for more economical design solutions. The process consists of two optimization loops, which have been adopted from a previous research paper (Liang et al., 2000). The first optimization loop generates optimal multistory unbraced frame structures. The second optimization loop gives economical solutions of structural bracing system topologies. A spatial constraint is embedded into the second optimization loop and will be further described in this chapter.

3.3 Computational Tool

The entirety of this study is conducted using two computational tools, MATLAB and SAP2000, to automate processes of iteration. MATLAB stores and runs the algorithms for all the optimization processes involved (First Optimization Loop and Second Optimization Loop). SAP2000 generates and analyzes the structural models using the inputs from MATLAB. An intercommunication platform between the two computation tools is developed using the documented SAP2000's Application Programming Interface (API). The API can be found in the installation file of SAP2000. The details of each function for both computational tools used in the study are described in the following section.

3.3.1 Optimization Algorithms – MATLAB

In the study, the first function of MATLAB is to store and execute codes to convert parametric design variables into arrays of numerical data. These arrays of numerical data are used to define, generate and modify the structural models in SAP2000. The second function of MATLAB is to store and execute user-defined algorithms to iterate the optimization variables whenever needed. These algorithms were then used to remove under-utilized bracing elements and generate adequate initial guesses for optimization. The third function of MATLAB used in the study is the inbuilt optimization toolbox for both the first optimization loop and the second optimization

loop. The MATLAB codes constructed for this study were recorded in the Appendix. The codes consist of two optimization loops with two inbuilt function each.

Within the optimization toolbox, the "fmincon" algorithm has been utilized as an optimization algorithm that finds the minimum outcome of a constrained nonlinear multivariable objective function. The *"fmincon"* algorithm in the optimization toolbox has a limitation that only accepts objective functions that produce one scalar quantity each. In the study, the objectives of both of the optimization loops were to minimize the total weight of the structure. While using the *"fmincon"* algorithm, equality or inequality constraints, which can be linear or nonlinear, have to be defined to set the limits for the optimization solutions. For the same reason, upper bounds and lower bounds of the solutions can also be specified for the *"fmincon"* algorithm. Although the structural analysis used is linear, the geometry of structures can be nonlinear (Connor 2003), the strength constraints in the first optimization loop and the compliance constraints in the second optimization loop are both nonlinear inequalities.

One criterion while using the *"fmincon"* algorithm is that designers are required to use engineering judgment to guess for the initial inputs of the optimization variables. The *"fmincon"* algorithm tends to converge to a solution that is the local minimum of a function. By having a "good" initial guess of the optimization, in which case the variables are close to the optimal solution, the convergence rate will be much higher. However, a "good" initial guess is not straightforward, and skills for appropriate guesses are developed through many years of engineering experience.

3.3.2 **Structural Analysis Software - SAP2000**

In the study, the main function of SAP2000 computational structural analysis tool is to carry out structural analysis on the steel frame structure using static frame analysis. The simplest structural analysis approach has been chosen to shorten the total amount of time required to optimize the topology of the bracing scheme. The main structural response that is needed for the optimization are the total weight of the structure, the compliance of the structure under defined load cases and the force and moment diagrams in each member of the structure. Once all the optimization processes were completed, SAP2000 is used to generate graphic structural model.

3.4 **Optimization Scheme**

This study investigates an optimization scheme to generate economical and realistic structural bracing schemes under static lateral loadings. The optimization scheme consists of two optimization loops, aiming to quickly create novel and physically realistic steel frame structures. The first optimization loop is created to generate optimal beam and column sizes in a steel frame structure. In the second optimization loop, braces will be generated by joining all the points that made up the vertical planes and an algorithm will be used to gradually remove the underutilized braces.

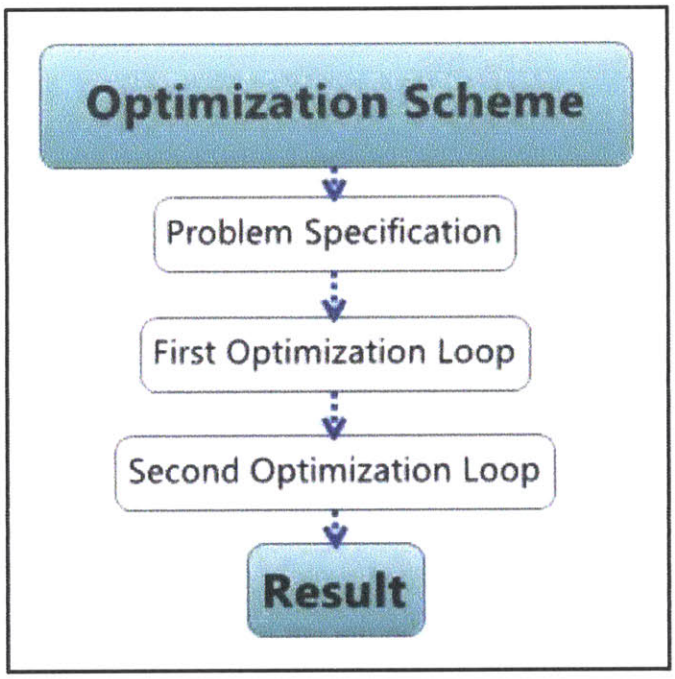

Figure 3: Flowchart of general processes involved in the optimization scheme (detailed process of first optimization loop and second optimization loop are shown in Figures 3 and 5).

The general processes involved in the optimization scheme are shown in Figure 2. More detailed flowcharts for the first optimization loop and the second optimization loop will be shown in Figure 4 and Figure 6. To use the optimization scheme, designers are required to input all the ten design parameters to specify the problem. These design variables have no bounds, but the designers have input physically realistic parameters to obtain a feasible structure. These inputs should also be scaled to the units specified in the scheme accordingly. The design parameters and their respective units used by the scheme are as follows:

1) Number of stories.
2) Number of frontal bay.
3) Number of side bay.
4) Height of one of the floor. [ft]
5) Width of one of the frontal bay. [ft]
6) Width of one of the side bay. [ft]
7) Number of floors that are grouped together to use a same steel member size for beams and columns
8) External loading – Superimposed Dead Load, Live Load and Wind Load magnitude. [k/ft]
9) Thickness of concrete slabs. [ft]
10) Space constraints parameters.

The first six parameters define the overall geometry of the multistory steel frame structure. The seventh parameter is an option provided to generate more realistic results. This is because in a typical structural design process, designers group structural elements that experience similar stress to save time and cost for both design and construction (McCormac, 2011). During the design phase, sizing every structural element is very time consuming and not economical. Alternatively, the typical method to design is to group similar elements together and size the controlling element. In the construction phase, if elements are sized with great variation, a lot of mistakes will occur while pieces of elements are being moved around on the construction site. The seventh parameter also needs to be within a reasonable range, for example, refrain from having group of ten floors. Within each group, the beams and columns are further broken down into smaller groups, interior/exterior beams and interior/exterior columns. The exterior elements take on twice as much load as their interior counterpart.

The eighth parameter defines the external loading applied on the structure. The live load magnitude depends on the purpose of the structure. The superimposed dead load is mainly created by the concrete slabs. The wind load will be a function of location and height of the structure. The ninth parameter is the thickness of the concrete slab throughout the whole structure. The thickness of the concrete slab mainly controls the

lateral stiffness provided by the slab to the structure. The last user input parameter is the space constraint parameter. The space constraint parameter is defined as coordinates that bound a space that will be free of bracings.

3.5 First Optimization Loop

Once all the parameters are input into MATLAB, the first optimization loop begins. A complete process flowchart of the first optimization loop is recorded in Figure 4. The first optimization loop starts out as MATLAB sets up the intercommunication platform with SAP2000 using the documented API codes. MATLAB passes data to SAP2000 as a one-dimensional array. Once the platform is set up, MATLAB will prompt SAP2000 application to begin and generate a blank new space for modeling. The default units used in SAP2000 are inches for length and kips-force for force. Similarly, the scale of magnitude of forces defined in civil structural problems is in kips-force, but the unit used to define these problems is usually in feet. For the ease for unity of units used to define data points, the default units in SAP2000 are reset to kips-force for force and feet for length. In this study, the only material used in the study is ASTM A572-50 structural steel, with minimum yield strength of 50ksi. A square cross section of 6 inches by 6 inches is defined as the default frame section. The choice of a square cross section is to simplify calculations in the study that will be discussed later.

Once the intercommunication platform between MATLAB and SAP2000 is built and the modeling space is defined accordingly, the constructed MATLAB sequence will automatically generate points and frame elements using the Cartesian coordinate system. The first six parameters input by the users are used to generate arrays of data that define the location of the points and frame elements on a Cartesian coordinate system. The points generated in SAP2000 help to define the location where each frame element intersects and are also used to reference points whenever needed. To reiterate, the frame elements are the beams and columns in the steel frame structure. Once the frame elements are generated, another set of algorithms will group these frame elements into smaller groups according to the nature of loads they are sustaining. The whole structure will be stratified into layers of groups of floors according to the seventh parameter input by designers. In each of these stratified layers, the frame elements are further broken-down into four groups, which are the interior or exterior

beams and the interior or exterior columns. The grouping of frame elements will be helpful when applying load or section modifiers to the frame elements.

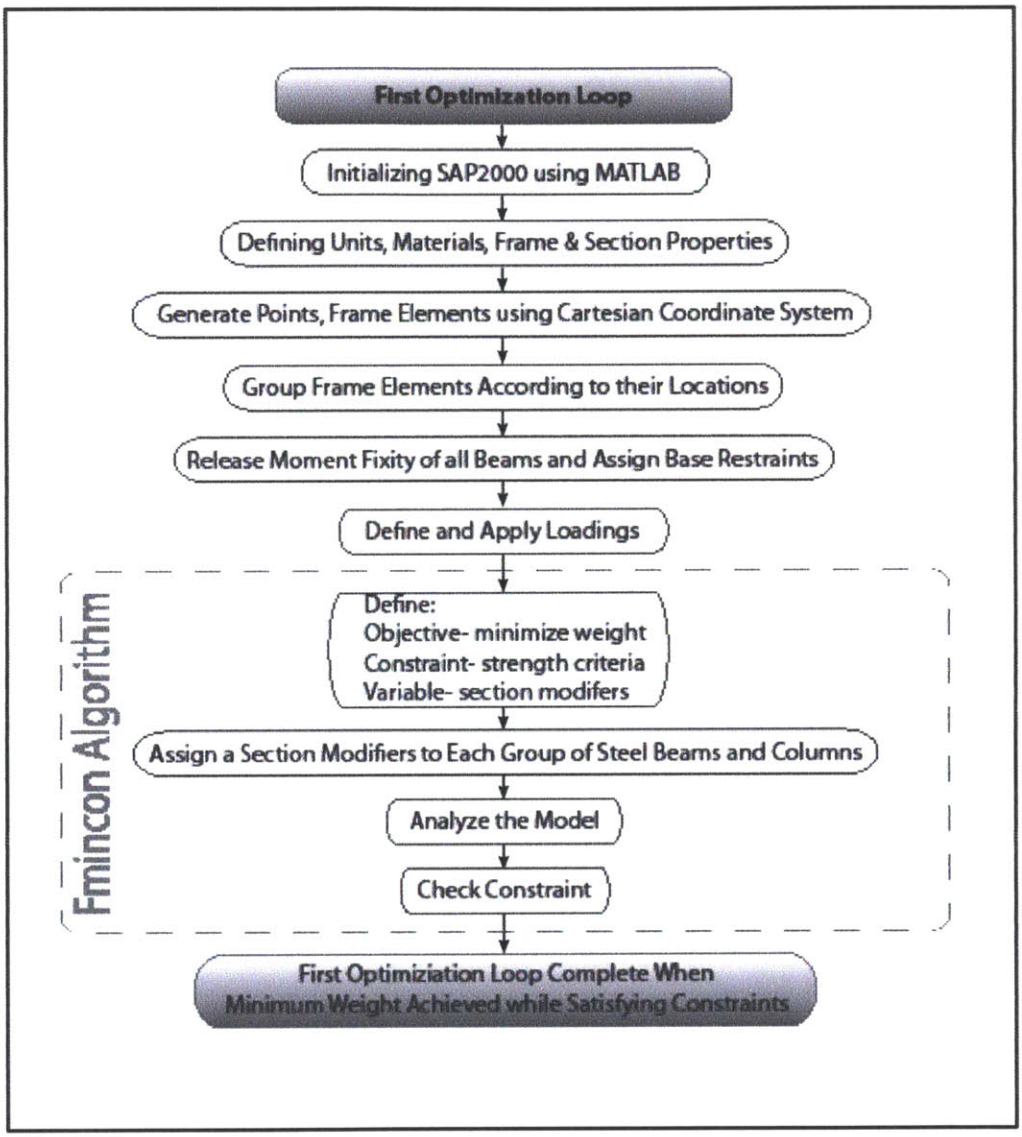

Figure 4: The first optimization loop process flowchart.

In the study, the beams in the steel frame structure are all modeled as simply-supported members. To model a simply-supported beam in SAP2000, moment is released from both ends of the beam and torsional fixity is released only at one end of the beam for stability. With this configuration, the beam becomes less effective in the lateral support system. Using the points defined, every point located on the ground level is restrained to idealize the pinned support condition. The final step before using the "*fmincon*" algorithm on the structure is applying loads to the structure. The load

magnitudes are parameters that are provided by the designers. The live load should be computed according to the functionality of the building. The superimposed dead loads are mostly contributed from the concrete slab. The self-weight of all the steel members in the structure are computed by SAP2000. As for the wind load, a linear wind profile is assumed, varying from zero at the base of the structure to the magnitude specified by designers at the top of the structure.

Once all the geometry, loads and material properties have been defined into SAP2000, the parameters in *"fmincon"* algorithm are to be specified next. In the *"fmincon"* algorithm, the objective is set to minimize the weight of the structure. The constraint is set to ensure that each frame element satisfies the axial force and moment strength criteria, and the variable to be optimized is the section modifier.

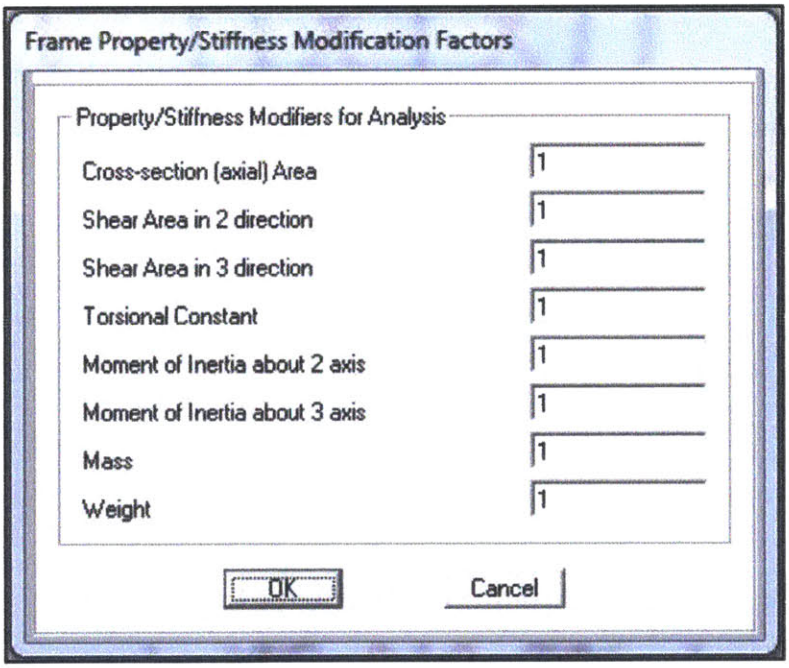

Figure 5: Section modifier user interface in SAP2000 required eight variables input.

The section modifier is chosen to be the variable in this optimization algorithm for two reasons. First, it is much easier to apply a section modifier on structural elements in SAP2000 as compared to the process of redefining new sections. Section modifiers are easily defined in SAP2000 modeling space; while the process of redefining new sections requires them to be defined and old sections to be replaced. When new sections are defined, the old sections do not get deleted but instead pile up in the

sections bank when the optimization process iterates. The second reason that the section modifier is chosen is because they are able to modify all the frame element section properties at once. For the first optimization loop, the weight of structures (objective of optimization) is only influenced by the cross sectional area of the steel members used since the material and geometry of the beams and columns are fixed attributes defined by the designers. The satisfaction of strength criteria (constraint of optimization) in each steel member is directly influenced by the area and moment of inertia of a section. All these factors that influence the objective and the constraint of the *"fmincon"* algorithm are directly correlated to the section modifier.

In this study, section modifiers are consistent within groups of steel members to simulate realistic design. The user interface for section modifiers in SAP2000 required eight variable inputs as shown in Figure 5. For simplification of computation, the study uses a square cross sectional area with the assumption that the depth and width of the steel section vary at the same rate. The section modifiers influence each member directly as follow:

$$X_{group} = section\ modifier\ of\ a\ group \tag{1}$$

If all the sections are square,

$$base = width = L \tag{2}$$

$$rate\ of\ change\ in\ base\ = rate\ of\ change\ in\ width = X_{group} \tag{3}$$

Then the section modifiers for each group of elements are as follow,

$$Cross\ section\ (axial)\ area_{group} = (X_{group})^2 \tag{4}$$

$$Shear\ area\ in\ 2\ direction_{group} = (X_{group})^2 \tag{5}$$

$$Shear\ area\ in\ 3\ direction_{group} = (X_{group})^2 \tag{6}$$

$$Torsional\ constant_{group} = (X_{group})^4 \tag{7}$$

$$Moment\ of\ inertia\ about\ 2\ axis_{group} = (X_{group})^4 \tag{8}$$

$$Moment\ of\ inertia\ about\ 3\ axis_{group} = (X_{group})^4 \tag{9}$$

$$Mass_{group} = (X_{group})^2 \tag{10}$$

$$Weight_{group} = (X_{group})^2 \tag{11}$$

The following parameters to be defined in the *"fmincon"* algorithm are the initial guesses for the variables, which give the lower and upper bound of the variables. The initial guess for the variable of the *"fmincon"* algorithm is set up according expected load path in the steel frame structure, frame elements supporting larger tributary area are sized larger. The beams and columns are grouped together based on their respective heights in the structure, a few floors are grouped together and the sizes decrease towards the top of the structure. Within this height determined group of frame elements, members are further distinguished in size based on their proximity to the exterior of the structure. This is because beams or columns that are located at the exterior bound of the structure only support half the tributary area compared to their interior counterparts. The lower and upper bound of the variables are not necessary within this study because the variables are bounded by the strength criteria constraint.

Once all the *"fmincon"* algorithm parameters are defined, the algorithm begins to execute. First, *"fmincon"* algorithm will apply the initial guess of section modifiers to their respective groups of frame elements and perform static frame analysis on the structure. The objective of the *"fmincon"* algorithm, which is the weight of the frame skeletal structure, is computed by SAP2000 by considering only unfactored self-weight of every frame element. The program stores the value of the weight computed and checks the constraint on each of the frame elements to ensure that they are all within their elastic strength region. For all the beams in the steel frame structure, the maximum moments are computed using SAP2000 and compared to the moment capacity in each beam. The moment capacity for each beam is calculated using the following equation (AISC 2011):

$$Moment\ capacity = \emptyset \cdot F_y \cdot Z_x \tag{12}$$

$$Z_x = \frac{width \times depth^2}{6} \tag{13}$$

where, $\emptyset$ = safety factor in steel moment capacity, 0.90

F_y = steel elastic yielding strength, 50ksi

Z_x = plastic section modulus for square cross section

As for the columns, the axial stress in each column is computed using SAP2000 and compared to the axial load capacity. Since all the columns are in compression, the steel strength of each column is determined as the minimum value between Euler's buckling strength and the steel elastic yielding strength (AISC 2011). The axial force load capacity in each column is calculated using the following formula:

$$F = minimum\left(\frac{\pi^2 \cdot E \cdot I}{(k \cdot L)^2}, F_y\right) \tag{14}$$

$$Axial\ load\ capacity = \emptyset \cdot A_s \cdot F \tag{15}$$

where, E = elastic modulus of steel, 29000ksi

I = moment of inertia

k = column effective length factor

L = length of column

$\emptyset$ = safety factor for axial load capacity in steel, 0.90

A_s = cross sectional area of steel

Using the MATLAB inbuilt *"fmincon"* algorithm, the initial guess of section modifiers are required to yield a result that satisfies the constraint. Once the first calculation is completed, the *"fmincon"* algorithm uses the *Active-Set* algorithm to iterate the next set of variables. The *Active-Set* algorithm has been chosen because the gradient used in linear static frame analysis is not provided in the objective function. The *Active-Set* algorithm computes the next iteration of variable using a "quasi-Newton approximation to the Hessian of the Lagrangian" (The Math Works, Inc. 2013). This allows the *Active-Set* algorithm to take large steps between iterations, allowing for faster convergence of results. The *Active-Set* algorithm computes the Lagrange multipliers of the objective function and stores them. All subsets of Lagrange multipliers with infeasible constraints are removed. The more detailed process of the *Active-Set* algorithm can be found in the book, "Numerical Optimization", by Jorge Nocedal (2006) and the paper, "A fast algorithm for nonlinearly constrained optimization calculations" by Michael Powell. Once the minimum objective of the first optimization loop has been achieved, the second optimization loop begins.

3.6 **Second Optimization Loop**

The output from the first optimization loop, which is the set of the section modifiers, applied on all the beams and columns for the gravity system of the steel frame. Similar to the first optimization loop, a flow chart of the processes involved in the second optimization loop are developed and illustrated in Figure 6.

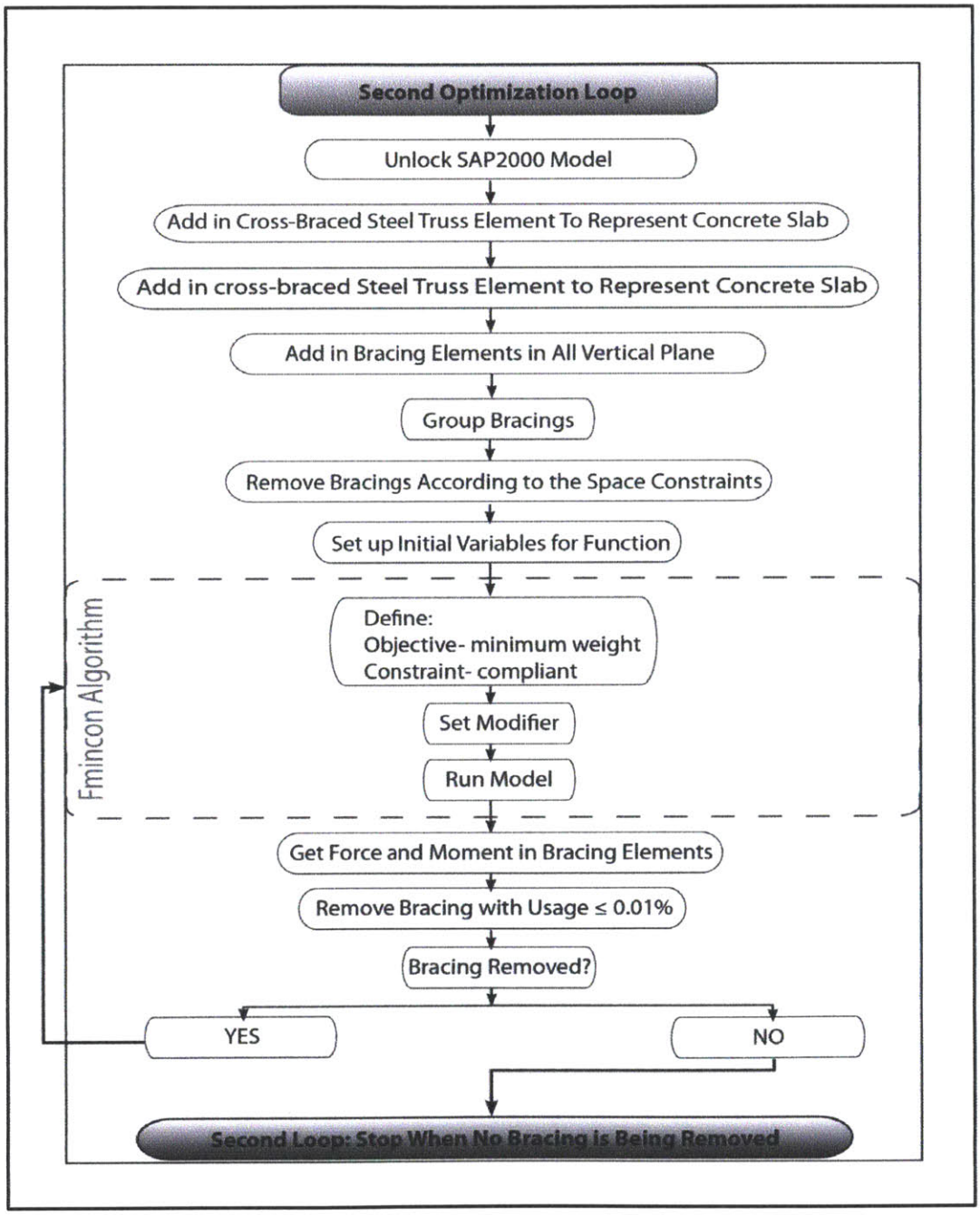

Figure 6: The second optimization loop process flowchart.

The second optimization loop is developed to generate economical topologies of the building bracing scheme for the lateral support system. While generating the economical topologies, some assumptions are made to simplify the structural design process. In this study, only linear static wind loading is considered. All the steel bracing elements are considered to be truss elements, which do not resist any shearing or twisting action. Before placing all the bracings in the steel frame structure, the SAP2000 model has to be unlocked; this is a safety feature in SAP2000 to ensure that the structural model is unaffected when the analysis is running. The first addition to the structural steel frame in the second optimization loop is to place cross-braced steel frames into the structure to substitute for the stiffness provided by reinforced concrete slabs. The method used to model these cross-braced steel trusses is recorded in the following section.

3.6.1 **Equivalent Frame Analysis for Reinforced Concrete Slab**

The goal to maintain high physical accuracy in the structural modeling encounters a problem when concrete slabs are to be included in the structural model for lateral loading resistance. Concrete slabs in a multistory steel frame structure contribute a significant portion of lateral stiffness to the structure and cannot be ignored while modeling a realistic structure (Corley 1961). Typically in a computational structural modeling space, concrete slabs are modeled as thin shells or plates with the equivalent stiffness (Asif 2013). The structural response of thin shells or plates cannot be computed using the static frame analysis. A more sophisticated finite element analysis is required to be used and the analysis will consume considerably more time. In order to keep the structural analysis process fast and simple, the concrete slabs cannot be modeled as thin shells or plates.

A method of idealizing concrete slabs as simple steel frame elements was proposed in the paper, "The Equivalent Frame Analysis for Reinforced Concrete Slabs", by Corley in 1961. The method measures the lateral stiffness provided by concrete slabs in a multistory steel frame structure and replaces all the concrete slabs using cross-braced steel truss elements. The method is illustrated in Figure 7. The method is adopted in our study.

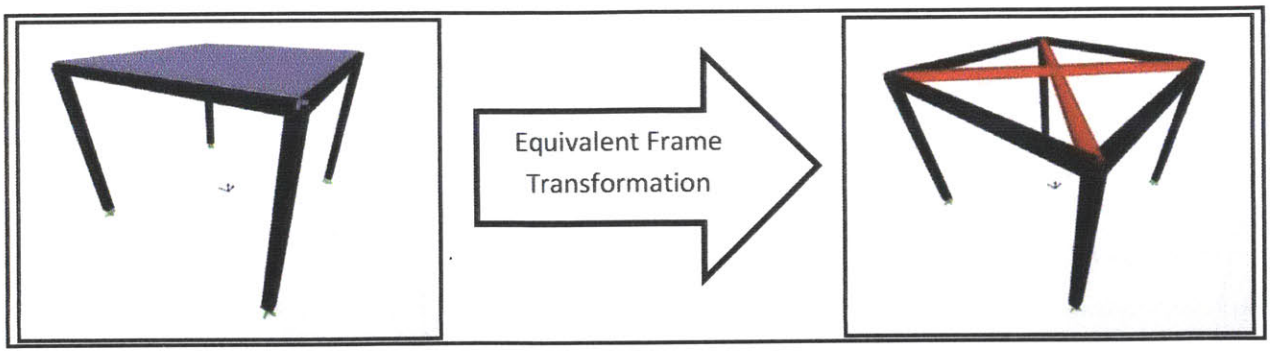

Figure 7: The equivalent frame transformation that convert the stiffness in a concrete slab to two cross-braced steel truss elements.

First, the torsional stiffness of the concrete slabs that contributes to the lateral stiffness is calculated using the follow formulas (AISC 318-05):

$$K_{torsional} = \frac{9 \cdot E_c \cdot C}{L_2 \cdot \left(1 - {}^{c_2}/_{L_2}\right)^3} \tag{16}$$

$$C = \left(1 - \frac{0.63 \cdot x}{y}\right) \cdot \left(\frac{x^3 \cdot y}{3}\right) \tag{17}$$

where:
- E_c = Young's modulus of elasticity for concrete
- C = torsional constant
- L_2 = width of the concrete slab
- c_2 = width of columns supporting the concrete slab
- x = shorter side of the rectangular concrete cross section
- y = longer side of the rectangular concrete cross section

In the study, the concrete slabs are assumed to be normal-weight concrete with a specified 28-day compressive strength of 4ksi. The Young's modulus of elasticity of the concrete in psi was calculated using the following formula:

$$E_c = 57{,}000\sqrt{f_c'} \tag{18}$$

where:
- f_c' = specified 28-day compressive strength of concrete [psi]

In the denominator portion of the calculation for the torsional stiffness, the equation can be further simplified by assuming that the optimization scheme always minimizes the width of the columns supporting the concrete slab, but limited by the strength

constraints. Therefore, c_2 will have considerably lower value than L_2. Hence, the ratio of $^{c_2}/_{L_2}$ can be neglected in the $K_{torsional}$ equation.

The end result from equation 16 is the lateral stiffness provided by the concrete beam at a direction that is parallel to the direction of L_2. The stiffness, $K_{torsional}$, does not translate directly as the stiffness of the cross-braced truss elements because of the difference in direction. A geometry transformation equation can be applied to $K_{torsional}$ to get the required stiffness in each cross-braced truss element, which is directly converted to the required steel cross sectional area. The area required in each cross-braced truss element can be derived as follow (Connor 2003):

$$V = 2 \cdot F \cdot \cos\theta \tag{19}$$

$$V = K_{torsional} \cdot U_{global} \tag{20}$$

$$F = \frac{A_{truss\_required} \cdot E_{steel}}{L_{truss}} \cdot u_{local} \tag{21}$$

$$U_{global} \cdot cos\theta = u_{local} \tag{22}$$

$$A_{truss\_required} = \frac{K_{torsional} \cdot L_{truss}}{2 \cdot A_{truss\_required} \cdot E_{steel} \cdot cos^2\theta} \tag{23}$$

where: V = applied force

F = axial force in the truss element

θ = angle between the truss element and the direction of L_2

U_{global} = global displacement

$A_{truss\_required}$ = required area in each truss element

E_{steel} = Young's modulus of elasticity for steel (29,000ksi)

L_{truss} = length of truss element

Using this entire equivalent frame modeling method, the computational time can be greatly reduced.

3.6.2 **Second Optimization Loop (continued)**

Once all the cross-braced steel truss elements are placed into the steel frame structure, the steel frame structure will be flooded by braces. All these bracing elements are first placed between two points that are not in the line of any beam or

column. These pairs of points are ones that have at least two different coordinates in terms of x, y or z. Then, all the joints that are different in all x, y and z coordinates are removed from the system as the optimization algorithm is meant to generate realistic steel frames where no braces should run across open room spaces. At the same time, all braces that are in the horizontal plane are also removed because there should not be braces running across the slab system in a real steel frame structure. The last modification to the bracing elements is to remove all the bracings according to space constraints. The space constraint is defined in a vector form with six elements as follow:

$$space\ constraint = \begin{pmatrix} bottom\ floor\ number \\ top\ floor\ number \\ frontal\ wall\ starting\ number \\ frontal\ wall\ ending\ number \\ side\ wall\ starting\ number \\ side\ wall\ ending\ number \end{pmatrix} \tag{24}$$

These six elements bound a space that is free of bracings elements. There is no limitation to the number of space constraints that can be defined in the algorithm, but having too many null spaces with no bracings might make the structure unstable. The whole process of defining and gradual removal of all the bracing elements in the steel frame structure is illustrated in Figure 8. The process of defining all the steel braces is embedded in the second optimization loop. Once all the steel braces are in place, all the braces in the structure will be redefined as truss elements by releasing their moment fixity. After the removal of the bracing elements due to impracticability and space constraints, there still exist many bracing elements leftover in the frame structure. If one variable is assigned to each leftover bracing element, the optimization algorithm will take a very long time to be executed. Therefore, those leftover bracing elements are grouped together according to their location, length and nature of force applied on them. The bracing elements are located higher in the structure are smaller than those that are nearer to the base. The longer the bracing elements, the larger its cross sectional area. Since steel frame structures are not always symmetrical from the front and the side view, the bracing elements should be sized accordingly.

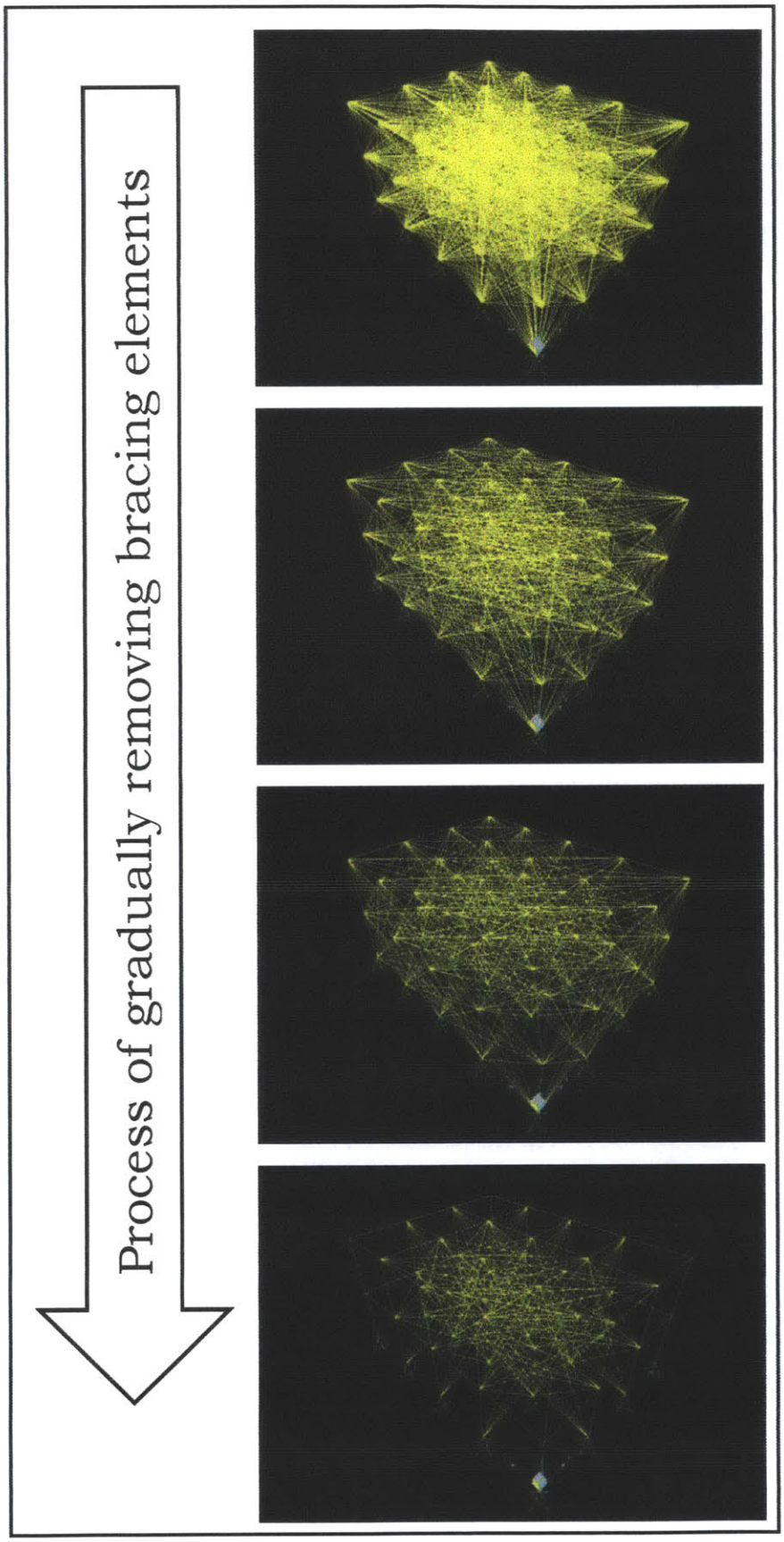

Figure 8: Process of gradually removing bracing elements. First, bracing removal to obtain realistic bracing placement and then to satisfy space constraint.

Once the structural elements that are required for the lateral system analysis are modeled in SAP2000, the second *"fmincon"* algorithm in MATLAB will be set up. In the second optimization loop, the objective of the *"fmincon"* algorithm is similar to the objective in the first *"fmincon"* algorithm, which is to minimize the total weight of the whole structure. Since the weight of the structure is directly related to the amount of material required to build the structure, the solutions from the second *"fmincon"* algorithm can be considered economical in terms of material usage. As for the constraint of the *"fmincon"* algorithm, compliance in the lateral movement of the structure is investigated. The variables of the second *"fmincon"* algorithm are set to be section modifiers, similar to the first *"fmincon"* algorithm, but the section modifiers in the second *"fmincon"* algorithm are applied on the groups of bracings instead of all the beams and columns. Once these three parameters of the second *"fmincon"* algorithm are defined, the initial guesses, upper bound and lower bound of the variable will be inputted. The initial guesses for the second *"fmincon"* algorithm is made by the designer while taking into consideration the mentioned sizing criterion while grouping the bracing elements. The same rule on the upper and lower bound applied on the first *"fmincon"* algorithm is adopted in the second *"fmincon"* algorithm. The bounds are not critical as the variables are actually limited by the constraint.

The second *"fmincon"* algorithm starts to execute once all the required input is defined. Initially, the second *"fmincon"* algorithm applies the initial guess for section modifiers on the respective groups of bracings. Then, SAP2000 will be used to run static frame analysis on the frame structure. The overall weight of the structure will be computed by using the load case that only considers the self-weight of every structural element. The lateral compliance of the structure is extracted from the result computed by SAP2000. The constraint of the second *"fmincon"* algorithm is set up to compare the inter-story drift of every joint in the structure to an inter-story drift limit of $\frac{h}{300}$ (AISC 2011). The variable h in the equation refers to the story height. Similar to the first *"fmincon"* algorithm, the result from the initial guesses in the second *"fmincon"* algorithm are required to satisfy the constraint. If the constraint is not satisfied, the initial guesses are required to be rescaled. The same optimization algorithm, Active-Set algorithm is used in the second *"fmincon"* algorithm. Using the initial result, the

second *"fmincon"* algorithm generates the next set of variables using the same approach explained in the first *"fmincon"* algorithm.

Once the second *"fmincon"* algorithm satisfies the objective of optimization while being constrained, the resulting frame structure will be checked against a frame removing criteria. The frame removing criteria used in this study compares the axial forces in each bracing element to their respective axial load capacity. If the usage of a bracing element is less than 0.01% of its axial load capacity, the algorithm will remove the bracing from the structure. While using this frame removing criteria, the designers have to be especially careful while grouping bracing elements. This is because only one steel size will be assigned to each group of bracing elements based on the bracing element that requires the largest steel size in each group. If bracing elements within each group experiences large disparities in member forces, then those bracing elements with low member forces will be removed unintentionally due to inefficient material usage. This limitation of the frame removal criteria is hard to avoid but can be minimized by further breaking down the grouping of bracing elements. The most optimal situation is that no bracing elements are grouped together, but the time taken for the optimization process will be inefficiently long. Since there is no way to ensure that the frame removal criteria are perfect, the final solution of the study can only be considered "economical" instead of "most optimized" solution.

Once all the underutilized bracings are removed from the structure, a constructed code will regenerate new variables for a rerun of the *"fmincon"* algorithm. This code makes sure that the new variables satisfy the constraint of the *"fmincon"* algorithm by gradually increasing the magnitude of the variables. The new generated variables will be used to initiate the second *"fmincon"* algorithm on the modified structure. After some of the bracing elements in the steel frame structure are removed, the computation time of the optimization is shown to improve drastically. This whole process goes on until the frame removing criteria no longer removes any bracing elements.

In the second optimization loop, some modifications were done to the *"fmincon"* algorithm to speed up the rate of convergence. The termination tolerance of the

objective and the compliance were adjusted to be 0.01, instead of the default value of 1e-6. This is acceptable because an error of 5% is considered negligible in structural design. Safety factors placed in pertinent design codes have already considered and precluded these negligible errors (ASIC 2011).

4.0 Chapter 4 – Results

4.1 Outline

Four case studies have been investigated using the developed optimization scheme. A performance index is developed to compare each result from all four cases to steel frame designed using a conventional method.

4.2 Performance Index

The ultimate objective of this study is to generate economical conceptual design of steel frame structures. The measurement of the objective of the study can be done by comparing costs from the results generated using the optimization scheme or a conventional method. The total cost of a structure ties together with many factors, including the complexity of design (design time), efficiency of design (amount of material) and constructability of design (construction time). A performance index is developed to measure the value of a design considering factors mentioned above.

The complexity of design is a subjective issue that varies in everyone's opinion. The comparison of the complexity of design is not included in the performance index However, the optimization scheme developed in this study only requires the designer to decide on the parametric design variables. These parametric design variables are usually provided by the building owner. The rest of the steps involved in steel frame structure design are all tackled in the proposed optimization scheme.
Sizing the beams and columns is a tedious process but it is done in the first optimization loop in the scheme. Designers usually use trial-and-error methods with their engineering experience to find a solution for the placement of the lateral bracing element in a steel frame structure. The process can take up a long time and the result can be feasible but with room for improvement. The second optimization loop is able to derive economical solutions of steel frame structures in a much shorter time.

In terms of efficient of design, total cost of a structure is determined by its type and amount of materials used. In this study, the material investigated is ASTM572-50 structure steel with a mass of $490 lb/ft^3$ (McCormac 2011). Since the type of material is fixed, the performance index developed takes into account of only the amount of

material used. The amount of material used is determined from the total weight of the structure.

The same gravity system will be utilized in the design by the optimization scheme and the design through conventional methods. The constructability of the structure is measured by numbers of lateral bracing elements. As the number of bracing elements increase, the effort required to build the structure increases. The effort can be quantified by the difficulty to place each bracing elements, but difficulty is a subject matter that is hard to be used to compare the designs. In the performance index, the constructability of the steel frame structure is quantified by the number of connections required to be built. Connections are costly and have a direct influence on the final cost of a design (McCormac 2011). Each bracing element requires two connections to be installed on into the steel frame structure. The total number of connections is calculated based on doubling the amount of bracing elements.

The two factors used to derive the performance index measure different quantities. Equivalent cost values of each element measured in the two factors is calculated by using the 2013 market value. For the cost of the material, a ton of steel costs about 700 US$ (Steelonthenet.com 2013). This is equivalent to about 320 US$ for a kip of steel. For the connections, the cost of a connection is a function of the force provided by the connection. For simplicity of calculation, all the connections are assumed to be the same and have a cost of 50 US$/connection. The result of the performance index will calculated as follow:

$$Performance\ Index = \left(total\ weight[kips] \times 320\left[\frac{US\$}{kips}\right]\right) + \left(\#\ connections\ \times 50\left[\frac{US\$}{connection}\right]\right) \quad (25)$$

4.3 **Results**

In this thesis, two case studies were examined. In each case study, a similar steel frame structure is made using the conventional design methods. Since the optimization scheme takes into account the space constraints, frames designed using the conventional method are also limited by the same constraints. The space constraints specified 2D null spaces at all the exterior walls of the structure. The space constraints should be consistent when doing the comparison because the study

examines the optimal placement of the bracing elements and placing bracing elements furthest away from the center of rigidity will be the most optimal placement of bracing elements (Connor 2003). Therefore, no bracing elements should be placed in the null spaces in using both design method. While designing the steel frame structure using the conventional method, the placement of the bracing elements is determined using trial-and-error. Steel frame structures designed using the conventional method initiate steel frame models derived from the first optimization loop. Instead of going through the second optimization loop, all the bays in the steel frame structure that are not under the space constraints are filled with cross-braced steel trusses. The compliance of the steel frame structure is computed using SAP2000 and compared to the limited compliance from building design code. Then the trusses in bracing the steel frame structure are removed one by one until the compliance of the structure no longer fulfill the design code limit. The frame removal starts from the center of the structure as those trusses provide the least amount of the lateral stiffness to the structure.

The final steel frame structures designed using the conventional method are compared to the steel frame derived from the optimization scheme. The weight of steel frame structures, number of the bracing elements and compliance of both steel frame structures are recorded for each case study. The performance index for all the steel frame structures are computed and compared to their counterpart in each case study graphically.

Case Study I

In the first case study, the steel frame structure is two stories tall with three frontal bays and two side bays. All the bay widths are set to be 10 feet apart and the story height is also 10 feet tall. Since the steel frame structure has only two stories, the gravity system in the first floor has the same members sizing as the second floor. 4 inches thick of concrete slab is used for the flooring in for the whole structure. The loading used in case study I is tabulated in Table 1. Using the thickness of the concrete slab, equivalent cross-braced steel trusses are placed as the floor in the structure. These cross-braced elements are hidden in the final display picture of the steel frame structures to allow for a clearer view of the lateral bracing elements. The final steel frame structures produced using the two different methods are shown in

Figure 10. The black frame elements are the beams and columns and the red frame elements are the bracing elements. The visual comparison of the two methods clearly shows that the optimization scheme produces a steel frame structure that uses less bracing elements. The sizes of each structural element group are recorded in Table 2 and the results from case study I are tabulated in Table 3. Table 3 shows a slight increase in weight from the conventional method to the optimized solution. The number of connections used in the design made using conventional method is twice the number of the connections used in the optimized solution. The cost index derived previously is also tabulated in Table 3.

Loadings	Magnitude [kips/ft]	
Live Load	Interior beams	2
	Exterior beams	1
Dead Load	Program calculated	
Superimposed Dead Load	Interior beams	2.5
	Exterior beams	1.25
Wind Load	Linearly varies from 1 kips/ft at the top of the structure to 0 at the base of the structure	

Table 1: Loading used in case study I

Structural Elements	Section Modifiers	Equivalent cross-sectional area $[in^2]$
Interior columns	0.772	2.316
Exterior columns	0.458	1.374
Interior beams	0.732	2.196
Exterior beams	0.716	2.148
Diagonal members	0.158	0.474

Table 2: Sizes of structural elements after optimization

	Weight [kips]	Number of connections
Optimization Scheme	22.71	28
Conventional Method	24.27	56
Cost Index	320 [US$/kips]	50 [US$/connection]

Table 3: Results of case study I

In Figure 9, the cost of each element in the performance index is calculated in terms of US$. The total cost of the whole structure is computed and shown in the figure. The steel frame structure designed using the conventional method is more costly in both criteria examined in the performance index.

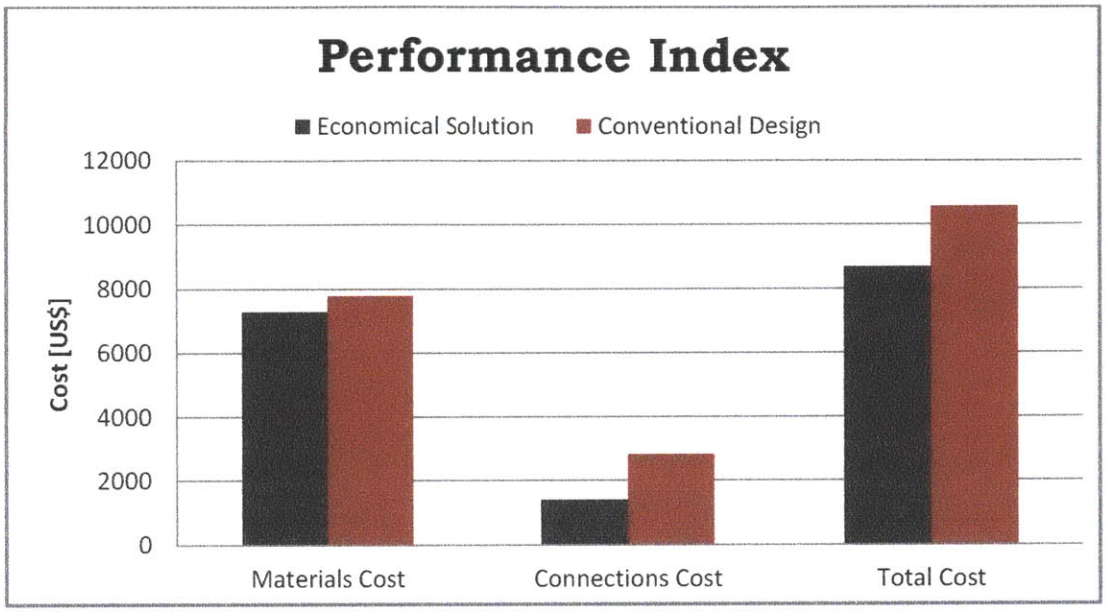

Figure 9: Performance Index comparison in Case Study I.

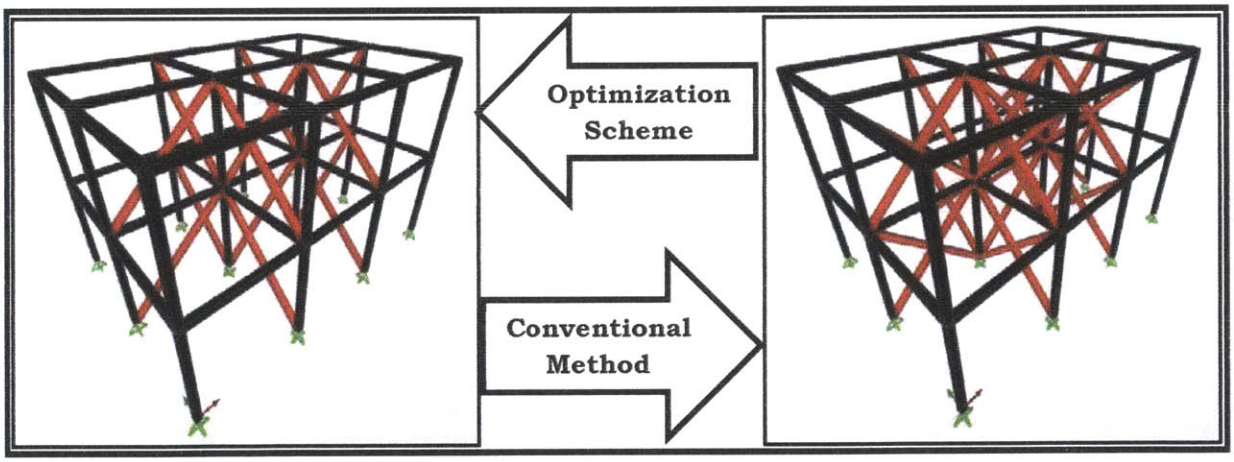

Figure 10: Steel frame structure of two different methods in Case Study I.

4.3.1 Case Study II

In the second case study, the steel frame structure is four stories tall with three frontal bays and three side bays. All the bay widths are set to be 10 feet apart and the story height is also 10 feet tall. The four story structure is divided into two distinct groups for beams and columns sizing. The first and second floors have same frame

member sizes and the same for the third and fourth floors. The loading used in case study II is tabulated in Table 4. Similar to the first case study, 4 inches thick of concrete slab is used for the flooring in for the whole structure and the equivalent cross-braced steel trusses are placed as the floor in the structure. These cross-braced elements are also hidden in all the final display pictures of the steel frame structures to allow for a clearer view of the lateral bracing elements. The final steel frame structures produced using the two different methods are shown in Figure 12. The black frame elements are the beams and columns and the red frame elements are the bracing elements. The denser red elements in the steel frame designed using the conventional method shows more bracings elements placed in the structure. The sizes of each structural element group are recorded in Table 5 and the results from case study I are tabulated in Table 6. Table 6 shows a slight increase in weight and number of connections used from the conventional method to the optimized solution. The cost index derived previously is also tabulated in Table 6. In Figure 11, the cost of each element in the performance index is calculated in terms of US$. The total cost of the whole structure is computed and shown in the figure. The steel frame structure designed using the conventional method is more costly in both criteria examined in the performance index.

Loadings	Magnitude [kips/ft]	
Live Load	Interior beams	2
	Exterior beams	1
Dead Load	Program calculated	
Superimposed Dead Load	Interior beams	2.5
	Exterior beams	1.25
Wind Load	Linearly varies from 1 kips/ft at the top of the structure to 0 at the base of the structure	

Table 4: Loading used in case study II.

Structural Elements	Section Modifiers	Equivalent cross-sectional area $[in^2]$
Interior columns (Floor 1 to 2)	0.772	2.316
Exterior columns (Floor 1 to 2)	0.458	1.374
Interior beams (Floor 1 to 2)	0.732	2.196
Exterior beams (Floor 1 to 2)	0.716	2.148
Interior columns (Floor 3 to 4)	0.778	2.334
Exterior columns (Floor 3 to 4)	0.65	1.950
Interior beams (Floor 3 to 4)	0.744	2.232
Exterior beams (Floor 3 to 4)	0.698	2.094
Diagonal members 1 bay (Floor 1 to 2)	0.0528	0.158
Diagonal members 2 bay (Floor 1 to 2)	0.0754	0.226
Diagonal members 3 bay (Floor 1 to 2)	0.1006	0.302
Diagonal members 1 bay (Floor 3 to 4)	0.1176	0.353
Diagonal members 2 bay (Floor 3 to 4)	0.0502	0.151
Diagonal members 3 bay (Floor 3 to 4)	0.09	0.270

Table 5: Sizes of structural elements after optimization.

	Weight [kips]	Number of connections
Optimization Scheme	60.22	50
Conventional Method	64.41	64
Cost Index	320 [US$/kips]	50 [US$/connection]

Table 6: Results of case study II.

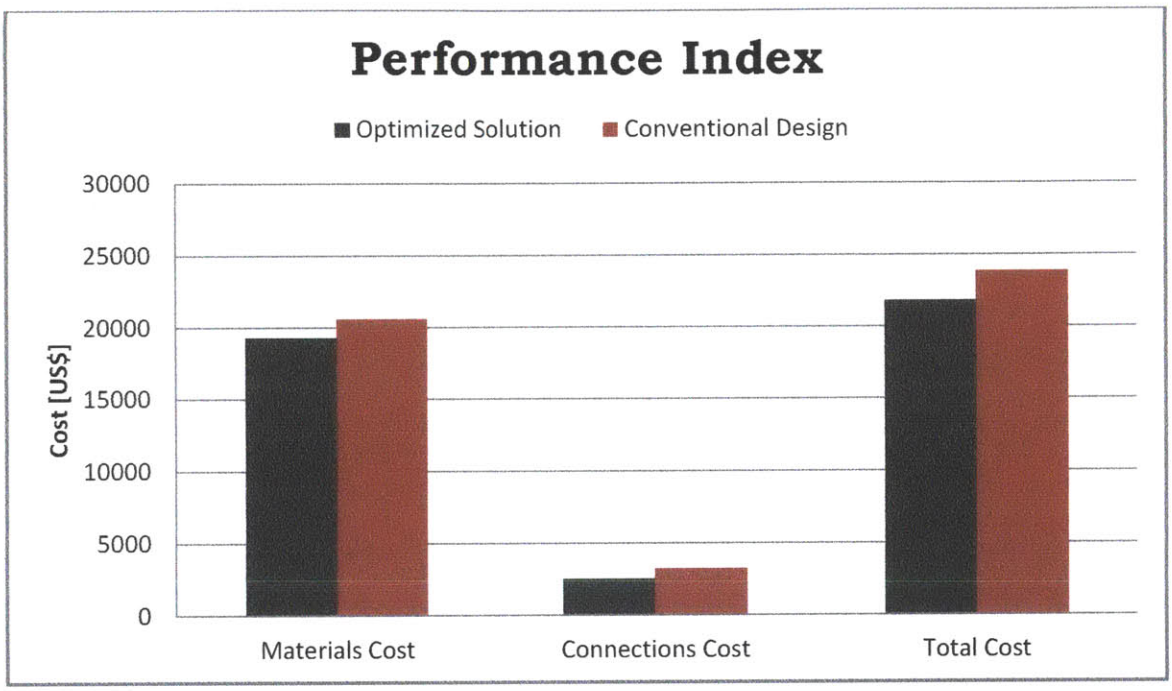

Figure 11: Performance Index comparison in Case Study II

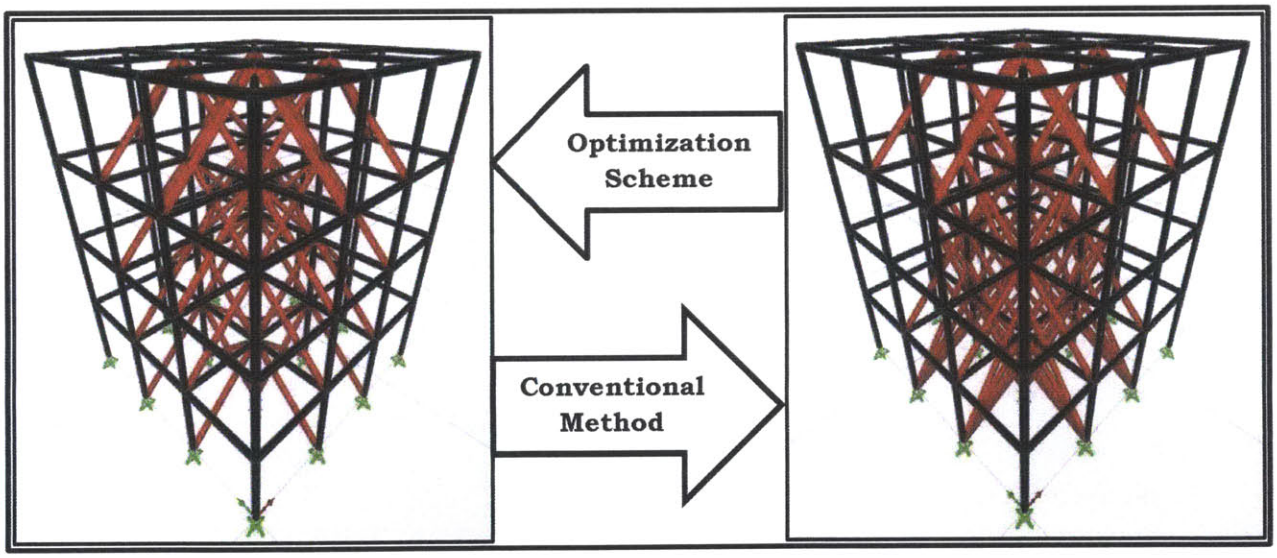

Figure 12: Steel frame structure of two different methods in Case Study II

5.0 Chapter 5 – Discussion

5.1 Outline

This chapter analyzes the results from the case studies. The future application and the adaptability of the optimization scheme proposed in the study are explored.

5.2 Discussion

Both case studies show a similar trend of results. The steel frame structure designed using the optimization scheme developed in the study produces more economical structures. The designs made from the optimization scheme in both case studies use less material in terms of weight and number of connections, and they are easier to construct as well as being less expensive. Also, interpreting the design step of the steel frame structure, the optimization scheme can be considered more robust because work conducted by the designers is considerably cut. The designers are only required to input the parametric design variables into the developed MATLAB optimization scheme and the rest of the job will be done by the computer. As for the conventional method, the designers are required to spend time and use previous engineering experience to run different structural models using computational structural analysis tools to find a "good" design. This process takes time and experience, which makes the optimization scheme a more economical method to derive conceptual designs of the steel frame structure.

Both designs in all the two case studies are checked to satisfy the compliance limit given in the AISC (2011) design code. The design derived using different methods yields different results. In a realistic structural design case, they all satisfy the code limit and are feasible structures. However, the design derived using the conventional method deflects less. This is because the conventional method uses more structural bracing elements. However, in designing civil structures, all the additional material used is wasted as the design codes have already been made to ensure safety. The comparison can be further studied by increasing the deflection limit on the steel frame structure designed using the optimization scheme. This further investigation of the topic will help to determine the reliability of the optimization scheme.

The main objective of the research is to derive an optimization scheme to help designers to generate economical steel frame structures in the conceptual design phase. Using the performance index developed in this study, the results generated using the optimization scheme are more economical than steel frame structures designed using the conventional method. Also, in both case studies, the space constraints successfully restrain the structures from have bracing elements on all the external walls. This concludes that the optimization scheme derived in this study helps to generate economical results with open spaces.

5.3 **Future Work**

There are some limitations to the optimization scheme proposed in the study. Many assumptions are made in the study and if any of the assumptions are not compatible with the real situation, the design is considered unfeasible. For example, the study is only done on linear static lateral loadings, if the structure is located in a seismically active area, the optimization scheme does not yield realistic results. However, modifications can be made to the optimization scheme to cover a wider range of steel frame engineering. The current optimization scheme is concluded to be able to produce economical results in a small scale structure. However, if the number of variables in the study increases, the computation time will increase exponentially and the optimization tool becomes infeasible.

There are several modifications that can be done on the proposed optimization scheme. For example, the frame removal criteria used in this study can only generate economical lateral bracing schemes compared to conventional designs. The frame removal criteria can be improved to ensure that optimal bracing schemes can be found. A good method is to use the bi-directional evolutionary structural optimization that allows the frame removal criteria to remove and add bracing elements. Another option to explore would be one that makes the optimization scheme more robust by finding ways to reduce the time consumed in optimization. With these additions, improvements can be made in reducing the structural response computation time, reducing the amount of variables or using a better optimization algorithm.

6.0 References

- Adeli, Hojjat, and Asim Karim. "Neural network model for optimization of cold-formed steel beams." Journal of Structural Engineering 123, no. 11 (1997): 1535-1543.
- Adeli, Hojjat, and Hyo Seon Park. "A neural dynamics model for structural optimization—theory." Computers & structures 57, no. 3 (1995): 383-390.
- Asif, Abell, Mike, "Technical Knowledge Base." CSi Knowledge Base. Last modified Jan 16 , 2013. Accessed May 7, 2013. https://wiki.csiberkeleyhttp://www.chicagomanualofstyle.org/tools_citationguide.html.com/x/u5AS
- Baker, William F. "Energy-based design of lateral systems." Structural Engineering International 2.2 (1992): 99-102.
- Baldock, Robert, and Kristina Shea. "Structural topology optimization of braced steel frameworks using genetic programming." In Intelligent Computing in Engineering and Architecture, pp. 54-61. Springer Berlin Heidelberg, 2006.
- Bertsekas, Dimitri P., and John N. Tsitsiklis. "Neuro-dynamic programming: An overview." In Decision and Control, 1995., Proceedings of the 34th IEEE Conference on, vol. 1, pp. 560-564. IEEE, 1995.
- Connor, Jerome, Introduction to Structural Motion Control, (Pearson Education, Inc., 2003)
- Corley, William Gene, M. A. Sozen, and C. P. Siess. "The equivalent frame analysis for reinforced concrete slabs." PhD diss., University of Illinois at Urbana-Champaign, 1961.
- Gaynor, Andrew T., James K. Guest, and Cristopher D. Moen. "Reinforced concrete force visualization and design using bilinear truss-continuum topology optimization." Journal of Structural Engineering (2012).
- Hoenderkamp, Johan CD, Hubertus H. Snijder, and Herm Hofmeyer. "Push-pull interface connections in steel frames with precast concrete infill panels." Open Constr. Build. Technol. J 6 (2012): 63-73.
- Kicinger, Rafal, Tomasz Arciszewski, and Kenneth De Jong. "Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings." In Evolutionary Computation, 2004. CEC2004. Congress on, vol. 1, pp. 411-418. IEEE, 2004.
- Kim, C. K., H. S. Kim, J. S. Hwang, and S. M. Hong. "Stiffness-based optimal design of tall steel frameworks subject to lateral loading." Structural optimization 15, no. 3-4 (1998): 180-186.
- Lee, S., S. Bobby, S. M. J. Spence, A. Tovar, and A. Kareem. "Shape and Topology Sculpting of Tall Buildings under Aerodynamic Loads." In 20th Analysis and Computation Specialty Conference, pp. 323-334. ASCE, 2012.
- Liang, Qing Quan, and Grant P. Steven. "A performance-based optimization method for topology design of continuum structures with mean compliance

constraints." Computer methods in applied mechanics and Engineering 191, no. 13 (2002): 1471-1489.
- Liang, Qing Quan, Yi Min Xie, and Grant P. Steven. "Optimal topology design of bracing systems for multistory steel frames." Journal of Structural Engineering 126, no. 7 (2000): 823-829.
- Manual, Steel Construction. "American Institute of Steel Construction." AISC, USA (2011).
- McCormac, Jack C., Stephen F. Csernak, and Manojkumar V. Chitawadagi. Structural steel design. Intext Educational Publishers, 2011.
- Mijar, Anand R., Colby C. Swan, Jasbir S. Arora, and Iku Kosaka. "Continuum topology optimization for concept design of frame bracing systems." Journal of Structural Engineering 124, no. 5 (1998): 541-550.
- Murty, Katta G. Linear complementarity, linear and nonlinear programming. Berlin: Heldermann, 1988.
- Nocedal, Jorge, and Stephen J. Wright. Numerical optimization. Springer Science+ Business Media, 2006.
- Powell, Michael JD. "A fast algorithm for nonlinearly constrained optimization calculations." In Numerical analysis, pp. 144-157. Springer Berlin Heidelberg, 1978.
- Querin, O. M., G. P. Steven, and Y. M. Xie. "Evolutionary structural optimization (ESO) using a bidirectional algorithm." Engineering Computations 15, no. 8 (1998): 1031-1048.
- Querin, O. M., V. Young, G. P. Steven, and Y. M. Xie. "Computational efficiency and validation of bi-directional evolutionary structural optimisation." Computer methods in applied mechanics and engineering 189, no. 2 (2000): 559-573.
- Rozvany, G. I. N., M. Zhou, and T. Birker. "Generalized shape optimization without homogenization." Structural Optimization 4, no. 3-4 (1992): 250-252.
- Steelonthenet.com, "Steel prices for years 2012 and 2013." Last modified 2013. Accessed May 12, 2013. http://www.steelonthenet.com/steel-prices.html.
- Smith, Jeffrey, Jessica Hodgins, Irving Oppenheim, and Andrew Witkin. "Creating models of truss structures with optimization." In ACM Transactions on Graphics (TOG), vol. 21, no. 3, pp. 295-301. ACM, 2002.
- Stromberg, Lauren L., Alessandro Beghini, William F. Baker, and Glaucio H. Paulino. "Application of layout and topology optimization using pattern gradation for the conceptual design of buildings." Structural and Multidisciplinary Optimization 43, no. 2 (2011): 165-180.
- Tenek, Lazarus H., and Ichiro Hagiwara. "Static and vibrational shape and topology optimization using homogenization and mathematical programming." Computer methods in applied mechanics and engineering 109, no. 1 (1993): 143-154.
- The Math Works, Inc., "Documentation of fmincon function (Active-Set-Algorithm)." Last modified 2013. Accessed May 9, 2013. http://www.mathworks.com/help/optim/ug/fmincon.html.

Appendix

MATLAB Codes

```
%% Clean up the workspace of MATLAB
        clear;
        clc;
        format long;
%% Initiate the SAP2000 API, constructing the platform that links SAP2000 with MATLAB↙
as described in the chapter 3.3 (Computation Tool)
for n1 = 1;
    % pass data to Sap2000 as one-dimensional arrays
        feature('COM_SafeArraySingleDim', 1);
    % pass non-scalar arrays to Sap2000 API by reference
        feature('COM_PassSafeArrayByRef', 1);
    % create Sap2000 object, any ERROR, please exit from SAP2000 fully
        SapObject = actxserver('sap2000.SapObject');
    % create SapModel object
        SapModel = SapObject.SapModel;
    % start Sap2000 application
        SapObject.ApplicationStart;
    % initialize model
        ret = SapModel.InitializeNewModel;
    % create new blank model
        ret = SapModel.File.NewBlank;
end
%% Set the units used in SAP2000 as kips  for force and ft for length
for n1 = 1;
        ret = SapModel.SetPresentUnits(4);
end
%% Input parametric design (Specifying problem) - described the First Optimization Loop↙
Chapter under the Methodology
        number_of_story = 7;
        number_of_frontal_bay = 4;
        number_of_side_bay = 4;
        height_of_story = 10; %ft
        width_of_frontal_bay = 10; %ft
        width_of_side_bay = 10; %ft
        group_of_story = 3; % number of story which the sizes of steel sections are↙
grouped together
        %assume that cross section is a square
        width_of_cross_section = 0.5; %[ft]
        A = (width_of_cross_section)^2; %cross sectional area of initial section [ft^2]
        thickness_of_concrete_slab = 4/12; %4 inches concrete slab[ft]
        f_c = 4; %[ksi]
        E_c = 57*sqrt(f_c*1000); %[ksi]
        SUPERDEAD = 1; %[kip/ft]
        LIVELOAD = 1; %[kip/ft]
        WINDLOAD = 5; %[kip/ft]
        number_of_group_of_floor = fix((number_of_story-1)/group_of_story)+1;
        total_beams_and_columns = ((number_of_story)*(number_of_frontal_bay+1)*↙
(number_of_side_bay+1))+((number_of_story)*(number_of_frontal_bay)*↙
(number_of_side_bay+1))+...
                                  ((number_of_story)*(number_of_side_bay)*↙
(number_of_frontal_bay+1));          %total number of beams and columns
        % space constraints input variables;
        % input format of wall with no bracings = [bottomfloor#  topfloor# FVwallstart#↙
FVwallend# SVwallstart# SVwallend#];
        % 1 <= bottomfloor# <= topfloor# <= number_of_story
```

```
        % 1 <= FVwallstart# <= FVwallend# <= (number_of_frontal_bay + 1)
        % 1 <= SVwallstart# <= SVwallend# <= (number_of_side_bay + 1)
        % input should not remove all the bracings in any floors
        removal(1,:) = [1 7 1 5 1 1];
        removal(2,:) = [1 7 1 5 5 5];
        removal(3,:) = [1 7 1 1 1 5];
        removal(4,:) = [1 7 5 5 1 5];
        % current setting take out all exterior bracings

%% Defining materials of structure as Steel
for n1 = 1;
        ret = SapModel.PropMaterial.AddQuick('A572Gr50',1,5,0,0,0,0,0,'A572Gr50');
end
%% DEFINE square frame section property
for n1 = 1;
        ret = SapModel.PropFrame.SetRectangle('.5ftx.5ft_rectangle','A572Gr50',↙
width_of_cross_section,width_of_cross_section);
end
%% ADD point object by coordinates generated from the parametric design variables by↙
designers
for n1 = 1;
        for n=1:((number_of_frontal_bay+1)*(number_of_side_bay+1)*(number_of_story+1));
            ret = SapModel.PointObj.AddCartesian((mod(n-1,number_of_frontal_bay+1))*↙
(width_of_frontal_bay), ...
            mod((fix((n-1)./(number_of_frontal_bay+1))),number_of_side_bay+1)*↙
(width_of_side_bay), ...
            (fix((n-1)./((number_of_frontal_bay+1)*(number_of_side_bay+1))))*↙
(height_of_story),'','');
        end
        total_points = ((number_of_frontal_bay+1)*(number_of_side_bay+1)*↙
(number_of_story+1));
        total_points_in_one_floor = ((number_of_frontal_bay+1)*(number_of_side_bay+1));
end
%% ADD frame object by coordinates generated from the parametric design variables by↙
designers(beams and columns)
for n1 = 1;
       for n=1:((number_of_story)*(number_of_frontal_bay+1)*(number_of_side_bay+1));
            ret = SapModel.FrameObj.AddByCoord((mod(n-1,number_of_frontal_bay+1))*↙
(width_of_frontal_bay), ...
            mod((fix((n-1)./(number_of_frontal_bay+1))),number_of_side_bay+1)*↙
(width_of_side_bay), ...
            (fix((n-1)./((number_of_frontal_bay+1)*(number_of_side_bay+1))))*↙
(height_of_story), ...
            (mod(n-1,number_of_frontal_bay+1))*(width_of_frontal_bay), ...
            mod((fix((n-1)./(number_of_frontal_bay+1))),number_of_side_bay+1)*↙
(width_of_side_bay), ...
            (fix((n-1)./((number_of_frontal_bay+1)*(number_of_side_bay+1))))+1)*↙
(height_of_story), ...
            '', '1ftx1ft_rectangle', sprintf('Column %d',n), 'GLOBAL'); %vertical frame↙
elememts
       end

        for n=1:((number_of_story)*(number_of_frontal_bay)*(number_of_side_bay+1));
            [ret, a] = SapModel.FrameObj.AddByCoord((mod(n-1,number_of_frontal_bay))*↙
(width_of_frontal_bay), ...
```

```
            mod((fix((n-1)./(number_of_frontal_bay))),number_of_side_bay+1)*↙
(width_of_side_bay), ...
            (fix((n-1)./((number_of_frontal_bay)*(number_of_side_bay+1)))+1)*↙
(height_of_story), ...
            (mod(n-1,number_of_frontal_bay)+1)*(width_of_frontal_bay), ...
            (mod((fix((n-1)./(number_of_frontal_bay))),number_of_side_bay+1))*↙
(width_of_side_bay), ...
            (fix((n-1)./((number_of_frontal_bay)*(number_of_side_bay+1)))+1)*↙
(height_of_story), ...
            '', '1ftx1ft_rectangle', sprintf('FV Beam %d',n), 'GLOBAL'); %front view↙
horizontal frame elememts
        end

       for n=1:((number_of_story)*(number_of_side_bay)*(number_of_frontal_bay+1));
            [ret, a] = SapModel.FrameObj.AddByCoord ((mod((fix((n-1)./↙
(number_of_side_bay))),number_of_frontal_bay+1))*(width_of_frontal_bay), ...
            ((mod(n-1,number_of_side_bay)))*(width_of_side_bay), ...
            (fix((n-1)./((number_of_side_bay)*(number_of_frontal_bay+1)))+1)*↙
(height_of_story), ...
            (mod((fix((n-1)./(number_of_side_bay))),number_of_frontal_bay+1))*↙
(width_of_frontal_bay), ...
            ((mod(n-1,number_of_side_bay))+1)*(width_of_side_bay), ...
            (fix((n-1)./((number_of_side_bay)*(number_of_frontal_bay+1)))+1)*↙
(height_of_story), ...
            '', '1ftx1ft_rectangle', sprintf('SV Beam %d',n), 'GLOBAL'); %side view↙
horizontal frame elememts
       end
end
%% ADD and assign group for all beams and columns to allow the optimization scheme to↙
generate relistic structure - described in page 29 of thesis
for n1 = 1;
        ret = SapModel.GroupDef.SetGroup('Interior Column');
        ret = SapModel.GroupDef.SetGroup('Exterior Column');
        ret = SapModel.GroupDef.SetGroup('Exterior Column Front');
        ret = SapModel.GroupDef.SetGroup('Exterior Column Back');
        ret = SapModel.GroupDef.SetGroup('Exterior Column Left');
        ret = SapModel.GroupDef.SetGroup('Exterior Column Right');
        ret = SapModel.GroupDef.SetGroup('Interior Beam');
        ret = SapModel.GroupDef.SetGroup('Exterior Beam');

         for n=1:((number_of_story)*(number_of_frontal_bay+1)*(number_of_side_bay+1));
         ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n),'Interior↙
Column',false(),0);
         end

         for m=1:number_of_story;
            for n=1:number_of_frontal_bay+1;
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',(n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1))),'Exterior Column Front',false(),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',(n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1))),'Interior Column',true(),0);
            end
         end

         for m=1:number_of_story;
```

```
            for n=((number_of_frontal_bay+1)*(number_of_side_bay)+1):↙
((number_of_frontal_bay+1)*(number_of_side_bay+1));
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Exterior Column Back',false(),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Interior Column',true(),0);
            end
         end

         for m=1:number_of_story;
            for n=1:number_of_frontal_bay+1:((number_of_side_bay+1)*↙
(number_of_frontal_bay+1));
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Exterior Column Left',false(),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Interior Column',true(),0);
            end
         end

         for m=1:number_of_story;
            for n=(number_of_frontal_bay+1):number_of_frontal_bay+1:↙
((number_of_side_bay+1)*(number_of_frontal_bay+1));
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Exterior Column Right',false(),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n+(m-1)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1)),'Interior Column',true(),0);
            end
         end

         ret = SapModel.FrameObj.SetGroupAssign('Exterior Column Right','Exterior↙
Column',false(),1);
         ret = SapModel.FrameObj.SetGroupAssign('Exterior Column Left','Exterior↙
Column',false(),1);
         ret = SapModel.FrameObj.SetGroupAssign('Exterior Column Front','Exterior↙
Column',false(),1);
         ret = SapModel.FrameObj.SetGroupAssign('Exterior Column Back','Exterior↙
Column',false(),1);

         for m=1:number_of_story;
            for n=1:number_of_frontal_bay;
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n+(m-1)↙
*number_of_frontal_bay*(number_of_side_bay+1)),'Exterior Beam',false(),0);
            end
            for n=1+number_of_frontal_bay*number_of_side_bay:number_of_frontal_bay*↙
(number_of_side_bay+1);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n+(m-1)↙
*number_of_frontal_bay*(number_of_side_bay+1)),'Exterior Beam',false(),0);
            end
            for n=1:number_of_side_bay;
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n+(m-1)*↙
(number_of_frontal_bay+1)*number_of_side_bay),'Exterior Beam',false(),0);
            end
            for n=1+number_of_frontal_bay*number_of_side_bay:(number_of_frontal_bay+1)↙
*number_of_side_bay;
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n+(m-1)*↙
```

```
(number_of_frontal_bay+1)*number_of_side_bay),'Exterior Beam',false(),0);
            end
         end

           for n=1:number_of_frontal_bay*(number_of_side_bay+1)*number_of_story;
           ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n),'Interior↙
Beam',false(),0);
           end

           for n=1:(number_of_frontal_bay+1)*number_of_side_bay*number_of_story;
           ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n),'Interior↙
Beam',false(),0);
           end
           ret = SapModel.FrameObj.SetGroupAssign('Exterior Beam','Interior Beam',↙
true(),1);
end
%% Continue grouping of beams and columns
for n1 = 1;
total_number_of_columns_in_one_floor=(number_of_frontal_bay+1)*(number_of_side_bay+1);
        total_number_of_FV_beams_in_one_floor=(number_of_frontal_bay)*↙
(number_of_side_bay+1);
        total_number_of_SV_beams_in_one_floor=(number_of_frontal_bay+1)*↙
(number_of_side_bay);
        for h=1:group_of_story:number_of_story;
        if h<number_of_story-group_of_story+1;
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to %d Interior Column',h,↙
h+group_of_story-1));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to %d Exterior Column',h,↙
h+group_of_story-1));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to %d Interior Beam',h,↙
h+group_of_story-1));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to %d Exterior Beam',h,↙
h+group_of_story-1));
        for n=(h-1)*total_number_of_columns_in_one_floor+1:(h-1+group_of_story)↙
*total_number_of_columns_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n),sprintf('Floor %d↙
to %d Interior Column',h,h+group_of_story-1),false(),0);
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n),sprintf('Floor %d↙
to %d Exterior Column',h,h+group_of_story-1),false(),0);
        end
        ret = SapModel.FrameObj.SetGroupAssign('Exterior Column',sprintf('Floor %d to %↙
d Interior Column',h,h+group_of_story-1),true(),1);
        ret = SapModel.FrameObj.SetGroupAssign('Interior Column',sprintf('Floor %d to %↙
d Exterior Column',h,h+group_of_story-1),true(),1);
        for n=(h-1)*total_number_of_FV_beams_in_one_floor+1:(h-1+group_of_story)↙
*total_number_of_FV_beams_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n),sprintf('Floor %↙
d to %d Interior Beam',h,h+group_of_story-1),false(),0);
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n),sprintf('Floor %↙
d to %d Exterior Beam',h,h+group_of_story-1),false(),0);
        end
        for n=(h-1)*total_number_of_SV_beams_in_one_floor+1:(h-1+group_of_story)↙
*total_number_of_SV_beams_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n),sprintf('Floor %↙
d to %d Interior Beam',h,h+group_of_story-1),false(),0);
```

```
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n),sprintf('Floor %↙
d to %d Exterior Beam',h,h+group_of_story-1),false(),0);
        end
        ret = SapModel.FrameObj.SetGroupAssign('Exterior Beam',sprintf('Floor %d to %d↙
Interior Beam',h,h+group_of_story-1),true(),1);
        ret = SapModel.FrameObj.SetGroupAssign('Interior Beam',sprintf('Floor %d to %d↙
Exterior Beam',h,h+group_of_story-1),true(),1);
        elseif h>=number_of_story-group_of_story+1;
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to roof Interior Column',↙
h));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to roof Exterior Column',↙
h));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to roof Interior Beam',h));
        ret = SapModel.GroupDef.SetGroup(sprintf('Floor %d to roof Exterior Beam',h));
        for n=(h-1)*total_number_of_columns_in_one_floor+1:↙
number_of_story*total_number_of_columns_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n),sprintf('Floor %d↙
to roof Interior Column',h),false(),0);
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('Column %d',n),sprintf('Floor %d↙
to roof Exterior Column',h),false(),0);
        end
        ret = SapModel.FrameObj.SetGroupAssign('Exterior Column',sprintf('Floor %d to↙
roof Interior Column',h),true(),1);
        ret = SapModel.FrameObj.SetGroupAssign('Interior Column',sprintf('Floor %d to↙
roof Exterior Column',h),true(),1);
        for n=(h-1)*total_number_of_FV_beams_in_one_floor+1:(h-1+group_of_story)↙
*total_number_of_FV_beams_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n),sprintf('Floor %↙
d to roof Interior Beam',h),false(),0);
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('FV Beam %d',n),sprintf('Floor %↙
d to roof Exterior Beam',h),false(),0);
        end
        for n=(h-1)*total_number_of_SV_beams_in_one_floor+1:↙
number_of_story*total_number_of_SV_beams_in_one_floor;
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n),sprintf('Floor %↙
d to roof Interior Beam',h),false(),0);
        ret = SapModel.FrameObj.SetGroupAssign(sprintf('SV Beam %d',n),sprintf('Floor %↙
d to roof Exterior Beam',h),false(),0);
        end
        ret = SapModel.FrameObj.SetGroupAssign('Exterior Beam',sprintf('Floor %d to↙
roof Interior Beam',h),true(),1);
        ret = SapModel.FrameObj.SetGroupAssign('Interior Beam',sprintf('Floor %d to↙
roof Exterior Beam',h),true(),1);
        end
        end
end
%% SET release at all beams
for n1 = 1;
        null = (zeros(6,1,'double'));
        strongaxismoment = logical([0;0;0;0;0;1]);
        ret = SapModel.FrameObj.SetReleases('Interior Beam', strongaxismoment,↙
strongaxismoment, null,null,1);
        ret = SapModel.FrameObj.SetReleases('Exterior Beam', strongaxismoment,↙
strongaxismoment, null,null,1);
end
```

```
%% ASSIGN point object restraint at base
for n1 = 1;
        fixed_base = logical(ones(6,1));
        pinned = logical([1;1;1;0;0;0]);
        roller = logical([0;0;1;0;0;0]);
        ret = SapModel.PointObj.DeleteRestraint('ALL',1);   % delete all existing↙
restraints
        for n=1:((number_of_frontal_bay+1)*(number_of_side_bay+1));
            ret = SapModel.PointObj.SetRestraint(num2str(n),pinned);
        end
end
%% ADD load patterns
for n1 = 1;
        ret = SapModel.LoadPatterns.Add('SUPERDEAD',2,0, true());
        ret = SapModel.LoadPatterns.Add('LIVE',3,0, true());
        ret = SapModel.LoadPatterns.Add('WIND',6,0, true());
        ret = SapModel.LoadPatterns.AutoWind.SetASCE705('WIND',1, 0, 0.8, 0.5, 5, 0.15,↙
0.15, false(), 0, 0, 70, 1, 1, 1.1, 0.85, 0.85);
end
%% ASSIGN loading for load pattern
for n1 = 1;
        ret = SapModel.FrameObj.SetLoadDistributed('Interior Beam', 'SUPERDEAD', 1, 10,↙
0, 1, SUPERDEAD, SUPERDEAD,'GLOBAL',true(),false(),1);
        ret = SapModel.FrameObj.SetLoadDistributed('Interior Beam', 'LIVE', 1, 10, 0,↙
1, LIVELOAD, LIVELOAD,'GLOBAL',true(),false(),1);
        ret = SapModel.FrameObj.SetLoadDistributed('Exterior Beam', 'SUPERDEAD', 1, 10,↙
0, 1, SUPERDEAD, SUPERDEAD,'GLOBAL',true(),false(),1);
        ret = SapModel.FrameObj.SetLoadDistributed('Exterior Beam', 'LIVE', 1, 10, 0,↙
1, LIVELOAD, LIVELOAD,'GLOBAL',true(),false(),1);
        for m=1:number_of_story;
            for n=1:number_of_frontal_bay;
            ret = SapModel.FrameObj.SetLoadDistributed(sprintf('FV Beam %d',n+((m-1)↙
*total_number_of_FV_beams_in_one_floor)), ...
            'WIND', 1, 8, 0, 1, m/number_of_story*WINDLOAD,↙
m/number_of_story*WINDLOAD,'GLOBAL',true(),false(),0);
            end
             for n=1:number_of_side_bay;
            ret = SapModel.FrameObj.SetLoadDistributed(sprintf('SV Beam %d',n+((m-1)↙
*total_number_of_SV_beams_in_one_floor)), ...
            'WIND', 1, 7, 0, 1, m/number_of_story*WINDLOAD,↙
m/number_of_story*WINDLOAD,'GLOBAL',true(),false(),0);
            end
        end
end
%% ADD load combinations
for n1 = 1;
        ret = SapModel.RespCombo.AddDesignDefaultCombos(true(), false(), false(),↙
false()); %add steel frame design default combo
        % delete load case 1,3,4,6,7,8
        % load case 2 for gravity design
        % load case 5 for lateral design
        ret = SapModel.RespCombo.Delete('UDSTL1');
        ret = SapModel.RespCombo.Delete('UDSTL3');
        ret = SapModel.RespCombo.Delete('UDSTL4');
        ret = SapModel.RespCombo.Delete('UDSTL6');
```

```
        ret = SapModel.RespCombo.Delete('UDSTL7');
        ret = SapModel.RespCombo.Delete('UDSTL8');
end
%% REFRESH view
for n1 = 1;
    ret = SapModel.View.RefreshView(0, false());
end
%% The first "fmincon" optimization algorithm by MATLAB - described further in page 30↙
of thesis
for n1 = 1;
%making x0 as [n+1 ; n ; n+1 ; n ; ..... ; 2 ; 1 ; 2; 1]
        number_of_group_of_floor = fix((number_of_story-1)/group_of_story)+1;
        x0=ones(number_of_group_of_floor*4,1);
        for n = 2 : number_of_group_of_floor;
        x0(1:4*(n-1),1)=x0(1:4*(n-1),1)+ones(4*(n-1),1);
        end
        for n = 1 : number_of_group_of_floor;
        x0(1+((n-1)*4):4+((n-1)*4),1)=x0(1+((n-1)*4):4+((n-1)*4),1)+[1;0;1;0];
        end
        lb=zeros(number_of_group_of_floor*4,1);
        ub=max(x0)*ones(number_of_group_of_floor*4,1);
        options = optimoptions('fmincon');
        options = optimoptions(options,'DiffMinChange', 0.5);
        options = optimoptions(options,'Display', 'iter');
        options = optimoptions(options,'TolCon', 1e-2);
        options = optimoptions(options,'TolFun', 1e-2);
        options = optimoptions(options,'TolX', 1e-2);
        x = fmincon(@BeamColumnSizing2,x0,[],[],[],[],lb,ub,@StrengthConstraints2,↙
options)
        [Weight] = BeamColumnSizing2(x);[c,ceq]=StrengthConstraints2(x);
end
%% UNLOCK model
for n1 = 1;
        ret = SapModel.SetModelIsLocked(false());
end
%% The beginning of the second optimization loop.
% ADD in all cross braced frame elements to represent concrete slab
for n1 = 1;
            % read all the coordinates of the existing points
                    X=zeros(1,1,'double');Y=zeros(1,1,'double');Z=zeros(1,1,'double');↙
a='';
                    matrix = zeros(total_points,3);
                    for n=1:total_points;
                        [ret, matrix(n,1), matrix(n,2), matrix(n,3)] = SapModel.↙
PointObj.GetCoordCartesian(sprintf('%d',n), X, Y, Z);
                    end
            % calculation equivalent frame element area
            torsional_constant_frontal = (1-0.63↙
*thickness_of_concrete_slab/width_of_frontal_bay)*↙
(thickness_of_concrete_slab^3*width_of_frontal_bay/3);
            torsional_constant_side = (1-0.63↙
*thickness_of_concrete_slab/width_of_side_bay)*↙
(thickness_of_concrete_slab^3*width_of_side_bay/3);
            K_t_frontal = (9*E_c*torsional_constant_frontal)/width_of_frontal_bay;
            K_t_side = (9*E_c*torsional_constant_side)/width_of_side_bay;
```

```
            theta = atan(width_of_side_bay/width_of_frontal_bay);
            length_of_cross_braces = width_of_side_bay/cos(theta);
            area_of_cross_braces = K_t_frontal*length_of_cross_braces/(2*29000*(cos↙
(theta))^2);
            area_modifiers_for_cross_braces = area_of_cross_braces/A;
            % setting modifiers
            ModValue = zeros(1,1,'double');
               ModValue(1,1) = (area_modifiers_for_cross_braces);  %cross sectional↙
area modifier
               ModValue(2,1) = (area_modifiers_for_cross_braces);  %shear area in local↙
2 direction modifier
               ModValue(3,1) = (area_modifiers_for_cross_braces);  %shear area in local↙
3 direction modifier
               ModValue(4,1) = (area_modifiers_for_cross_braces)^2;  %Torsional↙
constant modifier
               ModValue(5,1) = (area_modifiers_for_cross_braces)^2;  %Moment of inertia↙
about local 2 axis modifier
               ModValue(6,1) = (area_modifiers_for_cross_braces)^2;  %Moment of inertia↙
about local 3 axis modifier
               ModValue(7,1) = 0;  %Mass modifier
               ModValue(8,1) = 0;  %Weight modifier
            %set up group for cross braces
            ret = SapModel.GroupDef.SetGroup('Cross braces');
            % generating frame elements using coordinates
            iteration = 1;
            for h = 2 : number_of_story+1;
            for m = 1 : number_of_side_bay;
            for n = 1 : number_of_frontal_bay;
            mm = n+((m-1)*(number_of_frontal_bay+1))+((h-1)*↙
(total_points_in_one_floor));
            nn = n+1+((m)*(number_of_frontal_bay+1))+((h-1)*↙
(total_points_in_one_floor));
            ret = SapModel.FrameObj.AddByCoord(matrix(mm,1), matrix(mm,2),matrix(mm,↙
3),...
                                               matrix(nn,1), matrix(nn,2),matrix(nn,↙
3), ...
                                               '', '1ftx1ft_rectangle', sprintf('Cross↙
brace %d',iteration), 'GLOBAL');
            ret = SapModel.FrameObj.SetModifiers(sprintf('Cross brace %d',iteration),↙
ModValue(1:8,1),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Cross brace %d',↙
iteration),'Cross braces',false(),0);
            iteration = iteration + 1;
            mm = n+1+((m-1)*(number_of_frontal_bay+1))+((h-1)*↙
(total_points_in_one_floor));
            nn = n+((m)*(number_of_frontal_bay+1))+((h-1)*(total_points_in_one_floor));
            ret = SapModel.FrameObj.AddByCoord(matrix(mm,1), matrix(mm,2),matrix(mm,↙
3),...
                                               matrix(nn,1), matrix(nn,2),matrix(nn,↙
3), ...
                                               '', '1ftx1ft_rectangle', sprintf('Cross↙
brace %d',iteration), 'GLOBAL');
            ret = SapModel.FrameObj.SetModifiers(sprintf('Cross brace %d',iteration),↙
ModValue(1:8,1),0);
            ret = SapModel.FrameObj.SetGroupAssign(sprintf('Cross brace %d',↙
```

```
iteration),'Cross braces',false(),0);
            iteration = iteration + 1;
            end
            end
            end
            ret = SapModel.FrameObj.SetReleases('Cross braces', strongaxismoment,↙
strongaxismoment, null,null,1);
            total_number_of_cross_braces = iteration - 1;
            total_beams_and_columns_and_cross_braces = ((number_of_story)*↙
(number_of_frontal_bay+1)*(number_of_side_bay+1))+((number_of_story)*↙
(number_of_frontal_bay)*(number_of_side_bay+1))+...
                                    ((number_of_story)*(number_of_side_bay)*↙
(number_of_frontal_bay+1))+total_number_of_cross_braces;          %total number of↙
beams and columns
end
%% beginning of the frame removal criteria (user defined algorithm)
% ADD in all possible bracings in the vertical planes within the floor group and group↙
them according to their length
for n1 = 1;
% add groups for bracing according to their number of bays (equivalent to length)
       for h = 1:group_of_story:number_of_story;
       for n = 1:max([number_of_frontal_bay,number_of_side_bay,number_of_story]);
       ret = SapModel.GroupDef.SetGroup(sprintf('Bracings Group %d Above Floor %d',n,↙
h));
       end
       end
% add a frame bracing element to every two point elements that are different in at↙
least two axis of coordinate
% only braces in vertical are generated
       item = 1;
       for h=1:group_of_story:number_of_story;
           h1 = (h-1)*height_of_story;
           h2 = (h+group_of_story-1)*height_of_story;
       for n = 1:total_points-1;
       for m = n+1:total_points;
       if matrix(n,1)==matrix(m,1) && matrix(n,2)~=matrix(m,2) && matrix(n,3)~=matrix↙
(m,3) && h1<=matrix(n,3) && matrix(n,3)<=h2 && h1<=matrix(m,3) && matrix(m,3)<=h2||...
          matrix(n,1)~=matrix(m,1) && matrix(n,2)==matrix(m,2) && matrix(n,3)~=matrix↙
(m,3) && h1<=matrix(n,3) && matrix(n,3)<=h2 && h1<=matrix(m,3) && matrix(m,3)<=h2 ;
       ret = SapModel.FrameObj.AddByCoord(matrix(n,1), matrix(n,2),matrix(n,3),matrix↙
(m,1),matrix(m,2),matrix(m,3),...
            '', '1ftx1ft_rectangle', sprintf('Bracings %d',item), 'GLOBAL');
       if matrix(n,1)==matrix(m,1)
           nn=max([(abs(matrix(n,2)-matrix(m,2))/width_of_frontal_bay),(abs(matrix(n,3)↙
-matrix(m,3))/height_of_story)]);
       elseif    matrix(n,2)==matrix(m,2)
           nn=max([(abs(matrix(n,1)-matrix(m,1))/width_of_side_bay),(abs(matrix(n,3)-↙
matrix(m,3))/height_of_story)]);
       end
       ret = SapModel.FrameObj.SetGroupAssign(sprintf('Bracings %d',item),sprintf↙
('Bracings Group %d Above Floor %d',nn,h),false(),0);
        item = item+1;
       end
       end
       end
```

```
        end
        item = item-1;
        for n = 1:item;
        ret = SapModel.FrameObj.SetReleases( sprintf('Bracings %d',n), strongaxismoment,↙
strongaxismoment, null,null,0);
        end
end
%% REMOVE bracings accordings space constraints given by designers
for n1 = 1;
         for n = 1:size(removal(:,1));
         z11 = (removal(n,1)-1)*height_of_story;
         z22 = removal(n,2)*height_of_story;
         x11 = (removal(n,3)-1)*width_of_frontal_bay;
         x22 = (removal(n,4)-1)*width_of_frontal_bay;
         y11 = (removal(n,5)-1)*width_of_side_bay;
         y22 = (removal(n,6)-1)*width_of_side_bay;
         ret = SapModel.SelectObj.ClearSelection;
         ret = SapModel.SelectObj.CoordinateRange(x11*12, x22*12,y11*12,y22*12,z11*12,↙
z22*12,false(),'GLOBAL',false(),false(),true());
         ret = SapModel.SelectObj.Group('Interior Beam',true());
         ret = SapModel.SelectObj.Group('Interior Column',true());
         ret = SapModel.SelectObj.Group('Exterior Beam',true());
         ret = SapModel.SelectObj.Group('Exterior Column',true());
         ret = SapModel.SelectObj.Group('Cross braces',true());
         ret = SapModel.FrameObj.Delete('',2);
         end
         ret = SapModel.SelectObj.ClearSelection;
end
%% REFRESH view
for n1 = 1;
    ret = SapModel.View.RefreshView(0, false());
end
%% Second "fmincon" algorithm that keeps running until no further frame removal is done↙
- decribed in page 37 and 38 of thesis
for n1 = 1;
NumberofFrames = 1; NumberofFrames1 = 0;
while NumberofFrames ~= NumberofFrames1;

% set up y0 as [1,2,3,4,....,2,3,4,5,.....3,4,5,6,.....]
        number_of_group_of_bracings_in_each_floor_group=max([number_of_frontal_bay,↙
number_of_side_bay,number_of_story]);
        y0=ones↙
(number_of_group_of_floor*number_of_group_of_bracings_in_each_floor_group,1);
        for n = 2 : number_of_group_of_floor;
        y0(1:number_of_group_of_bracings_in_each_floor_group*(n-1),1)=y0(1:↙
number_of_group_of_bracings_in_each_floor_group*(n-1),1)+ones↙
(number_of_group_of_bracings_in_each_floor_group*(n-1),1);
        end
        for n = 1 : number_of_group_of_floor;
        for m = 1 : number_of_group_of_bracings_in_each_floor_group;
        y0((m)+(n-1)*(number_of_group_of_bracings_in_each_floor_group),1)=y0((m)+(n-1)*↙
(number_of_group_of_bracings_in_each_floor_group),1)+(m-1);
        end
        end
```

```
%making sure that constraints are always feasible
       [Weight] = BracingTopology_RemoveMember_MinForce4(y0);[c,ceq] =↙
ComplianceConstraints3(y0);
       while max(c)>0;
           y0 = y0+ones(size(y0,1),1)
       [Weight] = BracingTopology_RemoveMember_MinForce4(y0);[c,ceq] =↙
ComplianceConstraints3(y0);
       end
       y0

%set upper and lower bound of y0
        lb=0*ones(size(y0,1),1,'double');
        ub=2*max(y0)*ones(size(y0,1),1,'double');

% optimization
        options = optimoptions('fmincon');
        options = optimoptions(options,'DiffMinChange', 0.5);
        options = optimoptions(options,'Display', 'iter');
        options = optimoptions(options,'TolCon', 1e-2);
        options = optimoptions(options,'TolFun', 1e-2);
        options = optimoptions(options,'TolX', 1e-2);
        y = fmincon(@BracingTopology_RemoveMember_MinForce4,y0,[],[],[],[],lb,ub,↙
@ComplianceConstraints3,options)
% % % y =...
% % %    [0.041136296873365;...
% % %    0.054848395831153;...
% % %    0.068560494788942;...
% % %    0.082272593746730;...
% % %    7.000000000000220;...
% % %    7.999999999999996;...
% % %    9.000000000000309;...
% % %    0.033633087907873;...
% % %    0.041136296873365;...
% % %    0.054848395831153;...
% % %    0.068560494788942;...
% % %    6.000000000000229;...
% % %    7.000000000000031;...
% % %    8.000000000000025;...
% % %    0.112681369504044;...
% % %    0.027424197915577;...
% % %    0.041136296873365;...
% % %    0.073807927370027;...
% % %    5.000000000000286;...
% % %    6.000000000000012;...
% % %    7.000000000000100];
   [Weight] = BracingTopology_RemoveMember_MinForce4(y);[c,ceq] =↙
ComplianceConstraints3(y);

% get force and moment in bracings
        Name = cellstr('');
        [ret,NumberofFrames,FrameNames] = SapModel.FrameObj.GetNameList(0, Name);
        NumberofBracings = NumberofFrames - total_beams_and_columns_and_cross_braces;
        MaxBracingAxialForce = zeros(NumberofBracings,1);

%extract current existing bracings labels
```

```
        bracing_number = cellstr('');
        for n = 1:NumberofBracings
        bracing_number(n,1) = regexprep(FrameNames↙
(n+total_beams_and_columns_and_cross_braces),'\D','');
        end
        brace = str2double(bracing_number); %all the leftover bracing's number as↙
double

%Reading the remaining bracings axial forces
        ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput;
        ret = SapModel.Results.Setup.SetComboSelectedForOutput(sprintf('UDSTL%d',5));
        for m = 1 : NumberofBracings;
        NumberResults = 0;Obj = cellstr(' ');ObjSta = zeros(1,1,'double');Elm = cellstr↙
(' ');ElmSta = zeros(1,1,'double');LoadCase = cellstr(' ');StepType = cellstr(' ');↙
StepNum = zeros(1,1,'double');
        P = zeros(1,1,'double');V2 = zeros(1,1,'double');V3 = zeros(1,1,'double');T =↙
zeros(1,1,'double');M2 = zeros(1,1,'double');M3 = zeros(1,1,'double');
        [ret,NumberResults, Obj, ObjSta, Elm, ElmSta, LoadCase, StepType, StepNum, P,↙
V2, V3, T, M2, M3] = ...
        SapModel.Results.FrameForce(sprintf('Bracings %d',brace(m)),0, NumberResults,↙
Obj, ObjSta, Elm, ElmSta, LoadCase, StepType, StepNum, P, V2, V3, T, M2, M3);
        MaxBracingAxialForce(m,1) = (max(P));
        end

% unlock model
        ret = SapModel.SetModelIsLocked(false());

% remove frame members if less than 0.01% capacity is in use
        number_of_frame_removed = 0;
        for m = 1:NumberofBracings;
        Modifiers = zeros(8,1,'double');
        [ret,modifier] = SapModel.FrameObj.GetModifiers(sprintf('Bracings %d',brace↙
(m)), Modifiers);
        if MaxBracingAxialForce(m,1)<=0.05*0.9*50*144*modifier(1)*A  && modifier(1)<=1↙
|| modifier(1)<=0.0001;
        ret = SapModel.FrameObj.Delete(sprintf('Bracings %d',brace(m)),0);
        number_of_frame_removed = number_of_frame_removed + 1;
        end
        end
        number_of_frames_removed = sprintf('%d frames',number_of_frame_removed)
        [ret,NumberofFrames1,FrameNames1] = SapModel.FrameObj.GetNameList(0, Name);

% refresh view
        ret = SapModel.View.RefreshView(0, false());
end
       [Weight] = BracingTopology_RemoveMember_MinForce4(y);[c,ceq] =↙
ComplianceConstraints3(y);

end
```