AUTOMATED TOPOLOGY DESIGN OF HIGH-RISE BUILDINGS BY GENETIC ALGORITHM OPTIMIZATION AND SAP2000 OAPI CODES

Talha ORHAN

MASTER'S THESIS

Civil Engineering Department, Structural Engineering Division

Advisor: Asst. Prof. Kıvanç TAŞKIN

Eskişehir

Anadolu University

Graduate School of Sciences

August, 2018

FINAL APPROVAL FOR THESIS

This thesis titled “Automated Topology Design of High-Rise Steel Buildings by Genetic Algorithm Optimization and SAP2000 OAPI Codes” has been prepared and submitted by Talha Orhan in partial fullfillment of the requirements in “Anadolu University Directive on Graduate Education and Examination” for the Degree of Master of Science in Civil Engineering Department has been examined and approved on 02/08/2018.

Committee Members	**Title Name Surname**	**Signature**
Member (Supervisor)	: Asst. Prof. Kıvanç TAŞKIN	.....
Member	: Asst. Prof. Ahmet Ozan ÇELİK	.....
Member	: Asst. Prof. Ahmet Anıl DİNDAR	.....

Prof.Dr. Ersin YÜCEL
Director of Graduate School of Sciences

ABSTRACT

AUTOMATED TOPOLOGY DESIGN OF HIGH-RISE BUILDINGS BY GENETIC ALGORITHM OPTIMIZATION AND SAP2000 OAPI CODES

Talha ORHAN

Department of Civil Engineering

Programme in Structural Division

Anadolu University, Graduate School of Sciences, August 2018

Supervisor: Asst. Prof. Kıvanç TAŞKIN

In this study, in order to determine the outer shell geometry of high-rise diagrid structures by means of structural material usage simplified strength and stiffness based design methods are encoded in Matlab with the help of SAP2000 OAPI codes and Genetic Algorithm. Five different building cores are modeled in SAP2000 which consist of 30, 48, 60, 72, and 90 stories, and based on these core models the diagrid shell geometries are analyzed for both uniform and varying angle geometries. For the evaluation of optimum varying angle topology Genetic Algorithm is used. Dead, live and wind loads are evaluated and applied to the models based on minimum design loads according to ASCE 7-10. On the basis of resultant module diagonal cross-sectional areas an algorithm was developed and encoded in order to determine minimum required pipe section sizes according to buckling design of compression members as stated in AISC 360-10.

The results of the analysis show that, optimized varying angle models are more efficient than uniform models by means of structural material usage.

Keywords: Steel structure, High-rise buildings, Optimization, Genetic algorithm, Diagrid structures

ÖZET

YÜKSEK BİNALARDA GENETİK ALGORİTMA OPTİMİZASYONU VE SAP2000 OAPI KODLARIYLA ŞEKİL TASARIM OTOMASYONU

Talha ORHAN

İnşaat Mühendisliği Anabilim Dalı

Yapı Bilim Dalı

Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Ağustos 2018

Danışman: Dr. Öğr. Üyesi Kıvanç TAŞKIN

Bu çalışmada, yüksek diyagrid yapılarda kullanılan yapı malzemesi miktarı açısından optimum dış kabuk geometrisinin belirlenebilmesi için kuvvet dayanımı ve yatay rijitlik bazlı basitleştirilmiş ön tasarım metotları kullanılarak, Matlab'da SAP2000 OAPI kodları ve Genetik Algoritma yardımıyla bir kod oluşturulmuştur. SAP2000'de çekirdek kısmı modellenmiş 30, 48, 60, 72 ve 90 katlı beş farklı binanın analizleri, tanımlanmış çekirdek modelleri baz alınarak hem sabit, hem de değişken açılı diyagrid kabuk geometriler için yapı malzemesi akma dayanımı ve göreli modül ötelemesi limitleri altında gerçekleştirilmiştir. Değişken açılı modellerin geometrisinin belirlenmesinde ve optimum sonucun bulunmasında Genetik Algoritma kullanılmıştır. Yapılara uygulanan ölü, hareketli ve rüzgar yükleri ASCE7-10'da belirtilen minimum tasarım yüklerine göre belirlenmiştir. Bulunan modül diyagonal kesit alanlarına göre AISC 360-10'da belirtilen basınç elemanlarının burkulma hesaplarından faydalanılarak Matlab ortamında boru kesitler için bir boyut belirleme algoritması oluşturulmuştur.

Analiz sonuçları bina yükseklik-genişlik oranı arttıkça optimize edilmiş değişken açılı modellerde sabit açılı modellere göre toplam kullanılan yapı malzemesi miktarı açısından verimin arttığını göstermektedir.

Anahtar Sözcükler: Çelik yapı, Yüksek binalar, Optimizasyon, Genetik algoritma, Diyagrid yapılar

ACKNOWLEDGEMENT

This study is dedicated to my mother, who held my hand and stood me up whenever I stumbled on the road of life. I sincerely appreciate her and all my family members for their life-long efforts to give me courage and wisdom whenever I needed.

In addition, I want to express my sincere thanks to Asst. Prof. Kıvanç TAŞKIN for his patience and timely advice throughout this study. Thanks are extended to all faculty members of the Civil Engineering department at Anadolu University.

Talha ORHAN

02/08/2018

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Talha ORHAN

TABLE OF CONTENTS

Page No.

TITLE PAGE ..... **i**

FINAL APPROVAL FOR THESIS ..... **ii**

ABSTRACT ..... **iii**

ÖZET ..... **iv**

ACKNOWLEDGEMENT ..... **v**

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES ..... **vi**

TABLE OF CONTENTS ..... **vii**

LIST OF TABLES ..... **viii**

LIST OF FIGURES ..... **x**

NOMENCLATURE ..... **xv**

1. INTRODUCTION **1**

2. METHODOLOGY **4**

2.1. Structural Design **4**

2.2. Load Definitions **5**

2.3. Analysis Process **6**

2.3.1. Strength analysis **9**

2.3.2. Stiffness analysis **13**

2.3.3. Optimization **17**

3. RESULTS **19**

4. CONCLUSION & REMARKS **55**

REFERENCES **57**

APPENDIX

RESUME

LIST OF TABLES

Page No.

Table 2.1. Load combination factors 6

Table 3.1. Weight and profit percentage results for optimum uniform and optimized models 24

Table 3.2. Optimum module diagonal angles of 30-story optimized model Type 0 46

Table 3.3. Optimum module diagonal angles of 48-story optimized model Type 0 46

Table 3.4. Optimum module diagonal angles of 60-story optimized model Type 0 47

Table 3.5. Optimum module diagonal angles of 72-story optimized model Type 0 47

Table 3.6. Optimum module diagonal angles of 90-story optimized model Type 0 48

Table 3.7. Optimum module diagonal angles of 30-story optimized model Type 1 48

Table 3.8. Optimum module diagonal angles of 48-story optimized model Type 1 49

Table 3.9. Optimum module diagonal angles of 60-story optimized model Type 1 49

Table 3.10. Optimum module diagonal angles of 72-story optimized model Type 1 50

Table 3.11. Optimum module diagonal angles of 90-story optimized model Type 1 50

Table 3.12. Resultant diagonal pipe sections of 30-story optimized models 51

Table 3.13. Resultant diagonal pipe sections of 48-story optimized models 51

Page No.

Table 3.14. Resultant diagonal pipe sections of 60-story optimized models …… 52

Table 3.15. Resultant diagonal pipe sections of 72-story optimized models …… 53

Table 3.16. Resultant diagonal pipe sections of 90-story optimized models …… 53

LIST OF FIGURES

Page No.

Figure 1.1. The trussed façade of Chocolate Factory Menier at Noisel ...……. 1

Figure 1.2. (a) 30 St. Mary Axe, (b) Capital Gate, and
(c) Mode Gakuen Spiral Towers ...……. 2

Figure 2.1. Diagonal geometry types due to base module shape,
(a) Type 1, (b) Type 0 …… 4

Figure 2.2. Module types considered in (a) previous studies,
(b) this study ……... 4

Figure 2.3 Plan view of core building model (Angelucci et al., 2017) ...…….. 5

Figure 2.4. The superposition of core model and diagrid shell
(Moon et al., 2007) ..…….. 6

Figure 2.5. Optimization of Varying Angle Diagrids Flowchart ...……... 7

Figure 2.6. (a) Loading scheme for m-th module, under (b) gravity loads,
(c) lateral loads, and (d) moment ……... 9

Figure 2.7. Loading area to be applied on diagonals ……. 10

Figure 2.8. Shear distribution differences according to (a) Type 0,
and (b) Type 1 …… 11

Figure 2.9. Moment distribution differences according to (a) Type 0,
and (b) Type 1 …… 12

Figure 2.10. (a) Shear and (b) bending deformations under wind load ……. 13

Figure 2.11. Diagonal geometry definition stages in code …… 17

Figure 2.12. Diagonal geometry progress on (a) 1st iteration,
(b) 50th iteration, (c) 200th iteration, and (d) 2000th iteration ……. 18

Figure 3.1. Weight results due to diagonal angle for 30-story Type 0 …….. 19

Figure 3.2. Weight results due to diagonal angle for 48-story Type 0 …….. 19

Figure 3.3. Weight results due to diagonal angle for 60-story Type 0 …….. 20

Figure 3.4. Weight results due to diagonal angle for 72-story Type 0 …….. 20

Page No.

Figure 3.5. Weight results due to diagonal angle for 90-story Type 0 …….. 20

Figure 3.6. Weight results due to diagonal angle for 30-story Type 1 …….. 21

Figure 3.7. Weight results due to diagonal angle for 48-story Type 1 …….. 21

Figure 3.8. Weight results due to diagonal angle for 60-story Type 1 …….. 21

Figure 3.9. Weight results due to diagonal angle for 72-story Type 1 …….. 22

Figure 3.10. Weight results due to diagonal angle for 90-story Type 1 …… 22

Figure 3.11. The change of optimum diagonal angle by aspect ratio of uniform models ……..…….. 23

Figure 3.12. Weight comparison between optimum uniform and optimized models for Type 0 ……..……. 23

Figure 3.13. Weight comparison between optimum uniform and optimized models for Type 1 ….…….. 24

Figure 3.14. Diagonal area based on strength and stiffness design for 30-story optimum uniform model Type 0 ….……. 25

Figure 3.15. Diagonal area based on strength and stiffness design for 30-story optimized model Type 0 ..……. 26

Figure 3.16. Diagonal area based on strength and stiffness design for 48-story optimum uniform model Type 0 ……. 26

Figure 3.17. Diagonal area based on strength and stiffness design for 48-story optimized model Type 0 …..……. 27

Figure 3.18. Diagonal area based on strength and stiffness design for 60-story optimum uniform model Type 0 ..……. 27

Figure 3.19. Diagonal area based on strength and stiffness design for 60-story optimized model Type 0 ..……. 28

Figure 3.20. Diagonal area based on strength and stiffness design for 72-story optimum uniform model Type 0 ..……. 28

Page No.

Figure 3.21. Diagonal area based on strength and stiffness design for 72-story optimized model Type 0 29

Figure 3.22. Diagonal area based on strength and stiffness design for 90-story optimum uniform model Type 0 29

Figure 3.23. Diagonal area based on strength and stiffness design for 90-story optimized model Type 0 30

Figure 3.24. Diagonal area based on strength and stiffness design for 30-story optimum uniform model Type 1 30

Figure 3.25. Diagonal area based on strength and stiffness design for 30-story optimized model Type 1 31

Figure 3.26. Diagonal area based on strength and stiffness design for 48-story optimum uniform model Type 1 31

Figure 3.27. Diagonal area based on strength and stiffness design for 48-story optimized model Type 1 32

Figure 3.28. Diagonal area based on strength and stiffness design for 60-story optimum uniform model Type 1 32

Figure 3.29. Diagonal area based on strength and stiffness design for 60-story optimized model Type 1 33

Figure 3.30. Diagonal area based on strength and stiffness design for 72-story optimum uniform model Type 1 33

Figure 3.31. Diagonal area based on strength and stiffness design for 72-story optimized model Type 1 34

Figure 3.32. Diagonal area based on strength and stiffness design for 90-story optimum uniform model Type 1 34

Figure 3.33. Diagonal area based on strength and stiffness design for 90-story optimized model Type 1 35

Page No.

Figure 3.34. Module lateral displacements of 30-story optimum uniform model Type 0 …….. 36

Figure 3.35. Module lateral displacements of 30-story optimized model Type 0 …..…….. 36

Figure 3.36. Module lateral displacements of 48-story optimum uniform model Type 0 …….. 37

Figure 3.37. Module lateral displacements of 48-story optimized model Type 0 …..…….. 37

Figure 3.38. Module lateral displacements of 60-story optimum uniform model Type 0 …….. 38

Figure 3.39. Module lateral displacements of 60-story optimized model Type 0 …..…….. 38

Figure 3.40. Module lateral displacements of 72-story optimum uniform model Type 0 …….. 39

Figure 3.41. Module lateral displacements of 72-story optimized model Type 0 …..…….. 39

Figure 3.42. Module lateral displacements of 90-story optimum uniform model Type 0 …….. 40

Figure 3.43. Module lateral displacements of 90-story optimized model Type 0 …..…….. 40

Figure 3.44. Module lateral displacements of 30-story optimum uniform model Type 1 …….. 41

Figure 3.45. Module lateral displacements of 30-story optimized model Type 1 …..…….. 41

Figure 3.46. Module lateral displacements of 48-story optimum uniform model Type 1 …….. 42

Page No.

Figure 3.47. Module lateral displacements of 48-story optimized model Type 1 ….…….. 42

Figure 3.48. Module lateral displacements of 60-story optimum uniform model Type 1 …….. 43

Figure 3.49. Module lateral displacements of 60-story optimized model Type 1 ….…….. 43

Figure 3.50. Module lateral displacements of 72-story optimum uniform model Type 1 …….. 44

Figure 3.51. Module lateral displacements of 72-story optimized model Type 1 ….…….. 44

Figure 3.52. Module lateral displacements of 90-story optimum uniform model Type 1 …….. 45

Figure 3.53. Module lateral displacements of 90-story optimized model Type 1 ….…….. 45

NOMENCLATURE

SAP : Structural Analysis Program

OAPI : Open Application Programming Interface

RP : Reference points (diagonal starting points at base level)

LRFD : Load Resistance Factor Design

ASCE : American Society of Civil Engineers

AISC : American Institute of Steel Construction

n_{RP} : Number of reference points at base

Q_m : Total gravity load on m-th module

Q_D : Gravity load on m-th module due to dead load

Q_L : Gravity load on m-th module due to live load

H : Total height of structure

H_m : Height difference between m-th module top and ground level

f_{LC} : Load combination factor

$f_{LC,D}$: Load combination factor for dead load

$f_{LC,L}$: Load combination factor for live load

$f_{LC,W}$: Load combination factor for wind load

r_g : Ratio of gravity loads on diagonals / full slab

$h_{b.m}$: Height difference between m-th module base and ground level

h_m : Module height

h_s : Story height

A_s : Slab area

$F_{m.D}$: Gravity load applied on diagonals on m-th module due to dead load

$F_{m.L}$: Gravity load applied on diagonals on m-th module due to live load

$N_{m.k.D}$: Axial force on k-th diagonal on m-th module due to dead load

$N_{m.k.L}$: Axial force on k-th diagonal on m-th module due to live load

$N_{m.k.V}$: Axial force on k-th diagonal on m-th module due to shear force

$N_{m.k.M}$: Axial force on k-th diagonal on m-th module due to moment

$N_{m.k}$: Resultant axial force on k-th diagonal on m-th module

V_m : Shear force on m-th module due to wind load

M_m : Overturning moment on m-th module

P_w : Wind pressure

B_f : Flange width of structure

B_w : Web width of structure

b_w : Bay width of a diagonal

n_d : Number of diagonals on a module

$n_{d,w}$: Number of diagonals on one web façade of a module

$n_{d,f}$: Number of diagonals on one flange façade of a module

σ_y : Yielding stress of structural steel

$A_{d,m}$: Resultant cross-sectional area of one diagonal on m-th module

$A_{d,m,str}$: Diagonal cross-sectional area of one diagonal on m-th module by strength based design

$A_{d,m,stiff}$: Diagonal cross-sectional area of one diagonal on m-th module by stiffness based design

$\varepsilon_{w,k}$: Strain due to shear on k-th diagonal on web façade of m-th module

$\varepsilon_{f,k}$: Strain due to overturning moment on k-th diagonal on flange façade of m-th module

Δv_m : Module displacement due to shear

Δr_m : Module displacement due to rotation

γ_m : Module shearing strain

χ_m : Curvature of m-th module

θ_m : Diagonal angle on m-th module

$\Delta\beta_m$: Rotation angle on m-th module due to overturning moment

$F_{w,k}$: Axial force on k-th diagonal on web façade of m-th module due to shearing strain

$F_{f,k}$: Axial force on k-th diagonal on flange façade of m-th module due to curvature

δ_w : Web diagonal contribution on resisting moment

δ_f : Flange diagonal contribution on resisting shear

u_m : Inter-module drift

α : Inter-module drift ratio limit

P_n : Nominal compressive strength

F_{cr} : Critical stress

A_g : Gross area of the member

F_e : Buckling stress

L_d : Diagonal length

r : Radius of gyration

r_1 : Inner radius of pipe section

r_2 : Outer radius of pipe section

E : Modulus of elasticity

P_d : Design compressive strength

1. INTRODUCTION

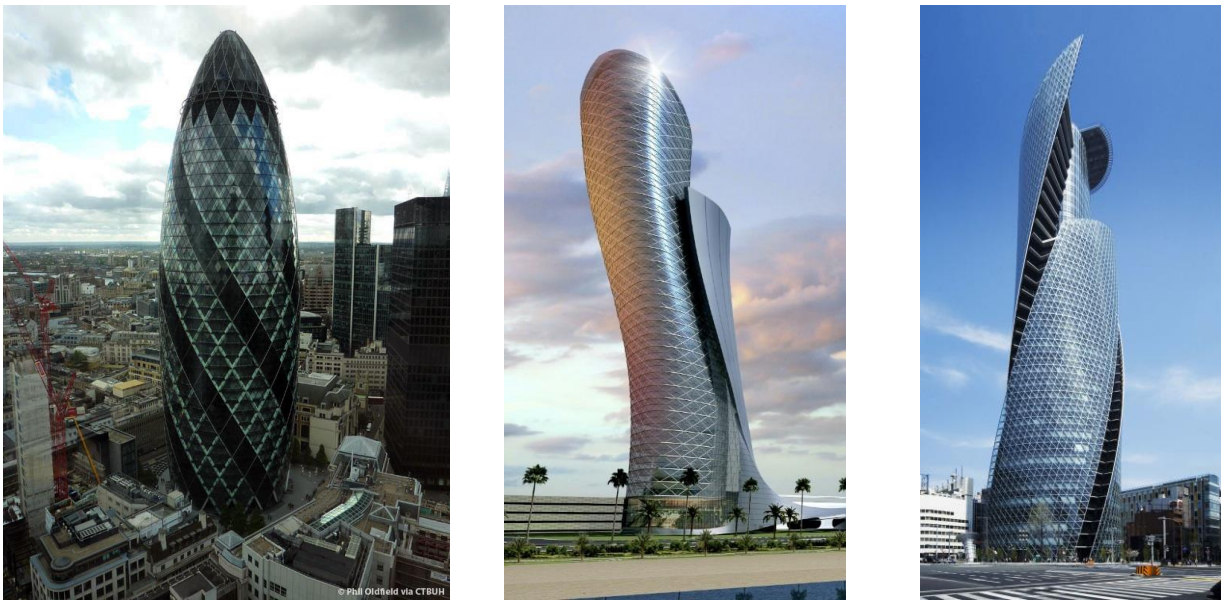
Diagrid structures consist of diagonal columns on the perimeter of the structure, that supports gravity and lateral loads as axial loads on diagonals. The unique triangular lattice design of diagrids have attracted architectures' and engineers' attention for last decades, and gave rise to a trend while designing modern tall buildings.

In time, these structures proved to be economic on material usage, as well as being structurally efficient due to their geometric configuration. As stated by the authors in their research about a comparative structural analysis between some recent tall buildings; since diagonals do not have shear or bending moment as internal force, they can be much more efficient by means of strength and stiffness capacity while being economic on material usage [1].

According to [2], the diagrid concept for steel structures is quite old that a building in France with trussed façade designed in 1873 can be given as an example (see Figure 1.1).

Figure 1.1. *The trussed façade of Chocolate Factory Menier at Noisel*

Some modern high-rise buildings that are built in latest years can be given as an example to diagrid structures, such as 30 St. Mary Axe, built in 2003 (180m) in England, Mode Gakuen Spiral Towers (170m), built in 2008 in Japan, and Capital Gate (160m), built in 2011 in United Arab Emirates (see Figure 1.2).



(a) (b) (c)

Figure 1.2. *(a) 30 St. Mary Axe, (b) Capital Gate, (c) Mode Gakuen Spiral Towers*

In scientific literature, some studies, although very few, have been developed on structural behavior, design criteria and performance assessment on mostly uniform angle designed diagrids. The whole design consists of repetitions of a unit diagonal module geometry.

With his published paper in 2007 [3], Moon proposed a stiffness based methodology to determine preliminary cross-sectional areas of diagonals on diagrid structures. In this study, steel diagrid buildings having various heights and various uniform diagonal slopes, were analyzed and compared in order to find optimum angle for uniform sloped models.

In [4], a 100-story high-rise diagrid structure is examined, by modeling the diagrid shell with various uniform diagonal angles, according to both strength and stiffness based designs in order to observe the results by means of structural performance and total weight.

Differently from previous studies, in the paper published by Zhang et al. in 2012 [5], four models with different aspect (height to width) ratio were analyzed by strength and stiffness based design, while having gradually varying diagonal angles from base to the roof of the building. The process was applied for various top and bottom angles in order to evaluate optimum geometry for minimum weight of the structure.

Furthermore, in the paper published in 2017 by Mele [6], a 90-story steel high-rise building having various uniform diagonal angles was examined by means of strength and stiffness based design with and without secondary bracing system (at core perimeter), in order to evaluate whether it is needed to neutralize inter-module flexibility effects.

In addition, Angelucci G. and Mollaioli F. (2017) [7], analyzed a 90-story steel tall building, having 69° optimal diagonal angle versus 82° non-optimal diagonal angle and also with some non-uniform diagonal geometries, by stiffness based design. The second order analysis (P-Δ) in linear elastic region for models were also applied in order to determine the optimum model by steel weight within lateral drift and member strength limits.

Automation of optimization in each area of science and engineering have become highly demanded, rather than trial and error method manually since it is more time consuming. For this purpose, many mathematical algorithms inspired by nature have been developed, such as; Artificial Bee Colony, Particle Swarm Optimization, Firefly Algorithm, etc. Within these algorithms Genetic Algorithm (GA) is widely used, since it was introduced in 1970s by John Holland and his colleagues [8].

In this paper, in order to optimize the diagonal geometry of high-rise diagrid structures with a simplified preliminary analysis, two Matlab codes with the help of SAP2000 OAPI commands are developed for the sake of automation of the process. One of the codes is designed to apply strength and stiffness based designs on predetermined uniform diagonal geometries, for the purpose of comparison, to determine diagrid shell weight in terms of minimum required member sizes. The other algorithm, while having the same design algorithms, has Genetic Algorithm additionally implemented in order to find the optimum diagonal geometry with an automated iterative way.

The purposes of this research are; the automation of this optimization process, to compare uniform geometry and optimized models, and evaluate the differences between them in terms of material usage and feasibility.

2. METHODOLOGY

2.1. Structural Design

Five building core models, which consist of 30-, 48-, 60-, 72- and 90-stories, each having two different base module diagonal geometries defined as “Type 0” and “Type 1” (see Figure 2.1) are studied in order to observe optimum diagonal angle and optimum weight differences according to aspect ratio (H/B) and diagonal geometry types.

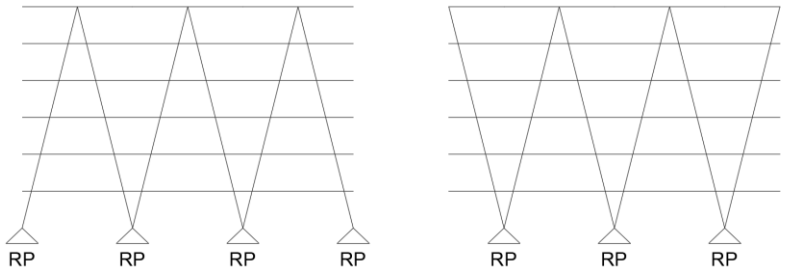


Figure 2.1. *Diagonal geometry types due to base module shape, (a) Type 1, (b) Type 0*

Unlike previous studies, which have x-bracing type of modules (see Figure 2.2a), in this study k-bracing type of modules are considered (see Figure 2.2b) for the purpose of more flexibility of the diagonal angles along the height of the structure, hence possibility for more optimum results.

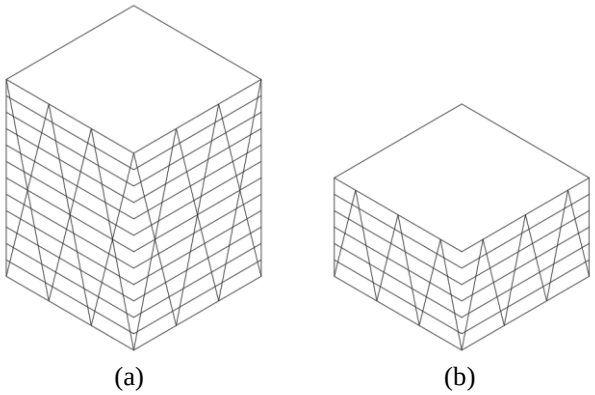


Figure 2.2. *Module types considered in (a) previous studies, (b) this study*

The dimensions for plan view (see Figure 2.3) and gravity core area are, 36m x 36m and 18m x 18m, respectively. Story heights are defined constantly as 4m. Each core model is designed according to these parameters for the sake of comparison, which is similar to the model mentioned in the research by Moon et al. [3].

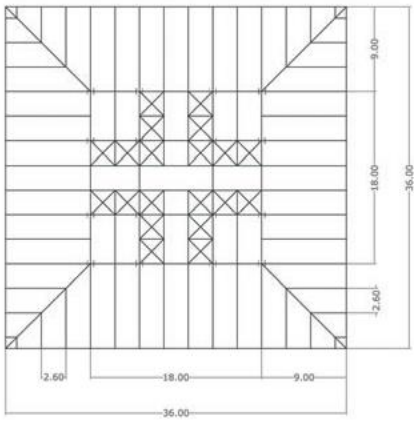


Figure 2.3 *Plan view of core building model (Angelucci et al., 2017)*

The exoskeletons of structures are modeled as steel diagonals without any columns, which create a triangular pattern, with moment-released ends to be able to carry only axial loads. Diagrid shell supports are defined as pin supports, whereas the gravity core columns have fixed supports.

For five building core models, and two diagonal geometry types; various uniform angle and optimized shaped diagrids are analyzed and compared according to weight efficiency. n_{RP} value is defined as "3" for Type 0 models, and "4" for Type 1 models, in order to have six diagonals on each façade of the building. Also as an input, a linear material model with S275 steel material type is defined for all models. And, inter-story drift ratio is defined as 1/500.

2.2. Load Definitions

Models are designed as office buildings hence, 2.4 kN/m^2 live load is applied to slabs [2]. Including frame members' weights, an average 7 kN/m^2 is applied as dead load. And for the lateral force, an average wind pressure of 2.86 kN/m^2 is applied to each floor diaphragms which is equivalent pressure for a basic wind speed of 110 mph (approximately 50 m/s), category D, according to ASCE 7-10 [9].

Load combination factors are defined by LRFD according to ASCE 7-10, which can be seen on Table 2.1.

Table 2.1. *Load combination factors*

Load Combination Name	Load Patterns		
	Dead Load	Live Load	Wind Load
D	1.4	0	0
DL	1.2	1.6	0
DW1	1.2	0	0.5
DW2	0.9	0	1
DLW	1.2	1	1

2.3. Analysis Process

Since tall buildings must ensure three design criteria, which are serviceability, strength and stability [10], the analysis procedure consists of stiffness analysis by a predefined drift limit, and strength analysis by the factors of safety according to AISC 360-10 [11] to meet those criteria. These analysis procedures are applied to all models in order to evaluate the required minimum diagonal cross-section area.

One of the assumptions while defining geometry and cross-sections of diagonals is, core and diagrid shell is superposed (see Figure 2.4). Gravity core is assumed to carry most of gravity loads, whereas diagrid shell is assumed to carry all of the lateral loads and a portion of gravity loads [3]. Additionally, each story nodes are modeled as diaphragms for each floor level, to have same lateral displacements.

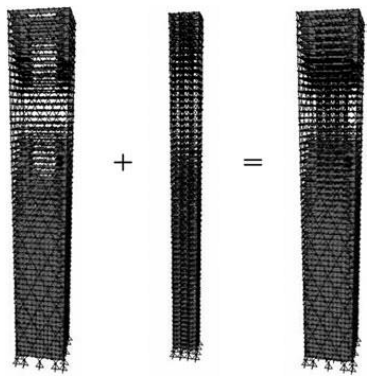


Figure 2.4. *The superposition of core model and diagrid shell (Moon et al., 2007)*

As stated previously, this study aims to automatize the determination of the diagonal geometry of high-rise diagrid structures by means of optimum material usage

while satisfying strength and serviceability limits. Therefore, two types of computer codes, for uniform angle (see A-1) and varying angle (see A-2) models, were developed using MATLAB software with the help of SAP2000 OAPI codes when there is a need for the code to interact with SAP2000. The flowchart of the analysis by Matlab is shown in Figure 2.5.

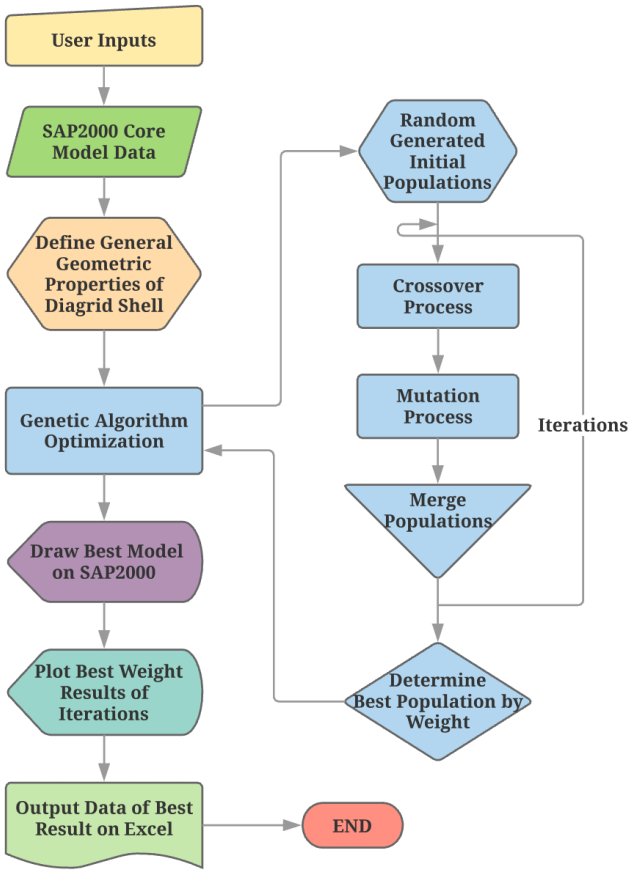


Figure 2.5. *Optimization of Varying Angle Diagrids Flowchart*

Before running the code, there must be a core model already designed in SAP2000 software with already defined material, load patterns (dead load, live load and wind load) and load combinations. In addition, the code is designed for square or rectangular plans, therefore for core models, which have plan views with other geometric shapes, the algorithm will not work correctly. User input stage of the code includes parameters that are diagonal geometry type (0 or 1 for Type 0 or Type 1, respectively), number of reference points (n_{RP}), drift ratio limit, material name (ex. S275); dead load (kN/m^2), live load (kN/m^2) and wind pressure (kN/m^2) values, file path of the SAP2000 core model.

These inputs are sufficient for the predefined uniform slope models. However, for the optimization process of varying angle models there are two more input parameters needed by the Genetic Algorithm optimization that are numbers of iterations and populations. Except this stage, all processes are handled by the code automatically.

After inputs are defined by user, program opens SAP2000 file of the core model to be able to access all data about the core model geometry and definitions. In this stage, the node coordinates, load patterns, load combinations, and material properties are transferred and stored to corresponding parameters (see A-3).

By the help of node coordinates list, n_{RP} and diagonal geometry type, general geometry parameters of diagrid shell are defined (see A-4, A-5) such as; story height, total building height, building web and flange width, number of diagonals on each facade, etc. Load patterns and multipliers of load combinations are used in strength analysis, which is explained in more detail in corresponding section (see Section 2.3.2).

The next stage differs according to the diagrid model design whether it's a uniform slope or a varying angle. If the model is uniform sloped, according to the predefined diagonal angle, all diagonal end-point coordinates, thus diagonal lengths, are calculated for each module. Strength analysis and stiffness analysis (see A-6) are applied, to be able to find required diagonal cross-section area within stress and drift limits. As a result, total weight of the diagrid shell is calculated. However, if the model will be optimized with varying angle diagonals, Genetic Algorithm is applied for this stage. As shown in Figure 2.5, there are five stages in this process, which are,

- Random generated initial populations,
- Crossover process (see A-8 - A-11),
- Mutation process (see A-12),
- Merging & truncating populations,
- Determination of the best population by weight.

First three stages of this process repeats the processes mentioned for the uniform sloped model with one difference, that is the population position determination. Population positions indicate the angle of diagonals for each module, for uniform sloped models this is a predetermined parameter as mentioned before. On the other hand, for varying angle models this data is a variable that changes for each population in each iteration. After each iteration, both the populations that were produced by crossover and

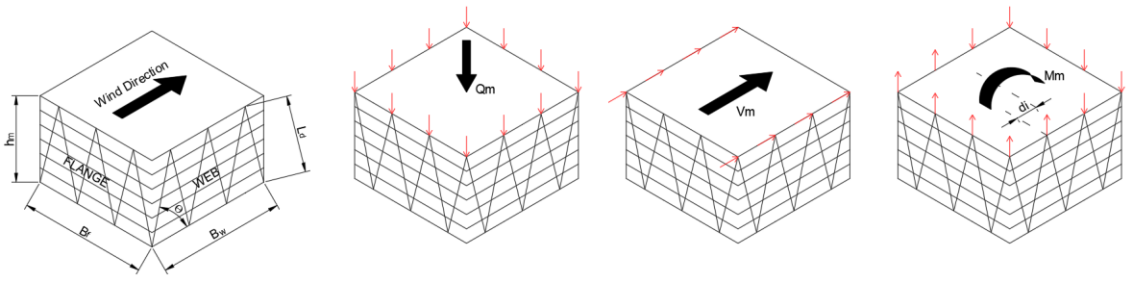
mutation are merged and sorted according to the calculated total weight of each population. At the end of the iterations, best population is decided by minimum total weight of the diagrid shell that has been found in the iterative process. This optimization process is explained in more detail in section 2.3.3.

Afterwards, a new SAP2000 file is created in order to draw the model of diagrid shell according to best (or predetermined) positions for the sake of display (see A-13).

Finally as the last phase of the code, all the parameters of the best (or predefined) model are exported to an excel file, such as module diagonal angles, module diagonal areas, inter-module drifts, etc.

2.3.1. Strength analysis

In this phase of the analysis, diagonal cross-section areas are defined according to simplified formulas as mentioned in Montuori et al. [4] under given assumptions before. Additionally, all load types are calculated considering the base level of each module. The loading distribution on diagonal intersection nodes according to different type of loads of a generic module are shown on Figure 2.6.



(a) (b) (c) (d)

Figure 2.6. *(a) Loading scheme for m-th module, under (b) gravity loads, (c) lateral loads, and (d) moment.*

The gravity load on each module Q_m, consists of dead and live loads. These load patterns can have different load combination factors (f_{LC}); hence they are considered separately. In addition, the ratio of gravity loads (r_g) that will be applied on diagonals are calculated according to the node coordinate list which is transferred from SAP2000 core model of the structure. Since there are no columns on perimeter of the model, core columns' base point coordinates are used to calculate core area, and beam members' end-

point coordinates are used to calculate full slab area. The hatched area shown on Figure 2.7 indicates the portion of the gravity loads for each story.

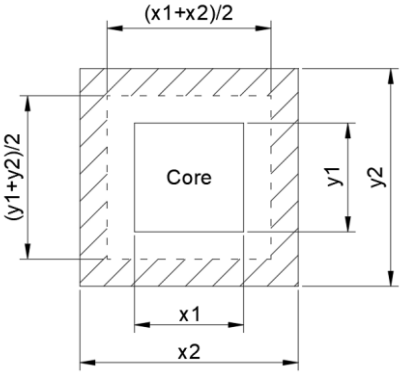


Figure 2.7. *Loading area to be applied on diagonals*

Hence, r_g can be defined as,

$$r_g = \frac{(x_2 y_2) - \frac{(x_1+x_2)(y_1+y_2)}{4}}{x_2 y_2} \tag{2.1}$$

The cumulative dead and live loads on module are calculated based on; dead (Q_D) and live (Q_L) area load inputs, total height of the structure (H), module base height ($h_{b,m}$) from ground level, slab area (A_s) and story height (h_s). Therefore, the gravity loads on m-th module $F_{m,D}$ and $F_{m,L}$ can be defined as,

$$F_{m,D} = r_g \frac{(H - h_{b,m})}{h_s} Q_D A_s \tag{2.2}$$

$$F_{m,L} = r_g \frac{(H - h_{b,m})}{h_s} Q_L A_s \tag{2.3}$$

Under these gravity loads on diagonal intersections, k-th diagonal force on m-th module ($N_{m,k,D}$ and $N_{m,k,L}$) can be written as,

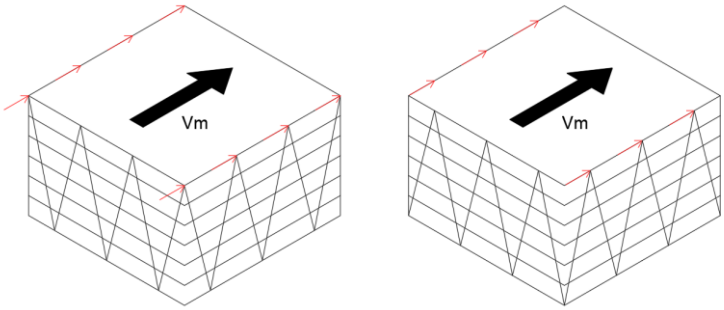
$$N_{m,k,D} = \frac{F_{m,D}}{n_d \sin\theta} \tag{2.4}$$

$$N_{m,k,L} = \frac{F_{m,L}}{n_d \sin\theta} \tag{2.5}$$

The wind load causes both shear and overturning moment on modules. The module shear (V_m) is calculated based on wind pressure (P_w) input, cumulatively. Thus, it can be written as in the following equation (see Eq. 2.6),

$$V_m = P_w\, B_f\, (H - h_{b,m}) \tag{2.6}$$

Shear force on module is carried by web diagonals. However, the load distribution by shear on diagonal intersections depends on which geometry type (see Figure 2.8.) the module is.



(a) (b)

Figure 2.8. *Shear distribution differences according to (a) Type 0, and (b) Type 1*

If the module geometry is Type 0, in addition to web diagonals, corner flange diagonals also help to carry shear. On windward façade, corner diagonals have tension, and on leeward façade they have compression. Additionally, both web and flange corner diagonals have half the axial load of other web diagonals. On the other hand, if the module geometry is Type 1, only web diagonals help to carry shear equally. Therefore, if module geometry is Type 0; for flange and web diagonals on corners Eq. 2.7, and for the remaining web diagonals Eq. 2.8 is used; otherwise, if module geometry is Type 1, Eq. 2.8 is used to determine the axial loads on all web diagonals by shear.

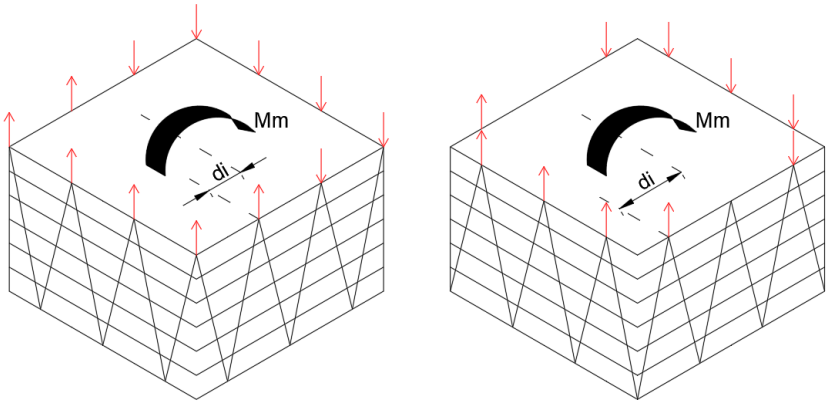
$$N_{m,k,V} = \frac{\pm V_m}{4\, n_{d,w} \cos\theta} \tag{2.7}$$

$$N_{m,k,V} = \frac{\pm V_m}{2\, n_{d,w} \cos\theta} \tag{2.8}$$

The overturning moment (M_m) on module is defined as,

$$M_m = \frac{P_w\, B_f\, (H - h_{b,m})^2}{2} \tag{2.9}$$

The overturning moment caused by wind load is carried by diagonals by means of the orthogonal distance (d_i) of the diagonal intersection points from the centroid axis of the slab which is orthogonal to the wind direction. Also, the module geometry affects the load distribution on diagonals similarly as mentioned under shear force (see Figure 2.9).



(a) (b)

Figure 2.9. *Moment distribution differences according to (a) Type 0, and (b) Type 1*

The vertical loads ($F_{m,i,M}$) on i-th diagonal intersection point of m-th module by moment can be defined as,

$$F_{m,i,M} = \frac{M_m d_i}{\sum_{i=1}^{n_k} d_i^2} \tag{2.10}$$

Hence, the axial force ($N_{m,k,M}$) on k-th diagonal by moment is given by,

$$N_{m,k,M} = \pm \frac{F_{m,i,M}}{2 \sin\theta} \tag{2.11}$$

After all the load types are calculated and stored in corresponding parameters in the code, the resulting axial force with load combination factors is evaluated as,

$$N_{m,k} = f_{LC,D}\, N_{m,k,D} + f_{LC,L}\, N_{m,k,L} + f_{LC,W}\, N_{m,k,V} + f_{LC,W}\, N_{m,k,M} \tag{2.12}$$

Finally, in order to define minimum required cross-sectional area ($A_{d,m}$) of diagonals on a module, each resulting diagonal force is stored and maximum tension and compression values for each module are evaluated. Afterwards, $A_{d,m}$ is evaluated according to AISC 360-10 [4], to have more reliable results in order to be used on actual model design. As a result, if the maximum force on module is tension,

$$A_{d,m,str} = \frac{\max(|N_{m,k}|)}{0.9\, \sigma_y} \tag{2.13}$$

Otherwise,

$$A_{d,m,str} = \frac{\max(|N_{m,k}|)}{0.6\, \sigma_y} \tag{2.14}$$

2.3.2. Stiffness analysis

In this stage of the analysis, shear and overturning moment effects on lateral displacement by wind load is considered, in order to evaluate module diagonal cross-sectional areas under drift ratio limits. For this purpose, the equations mentioned in [5] are used based on Figure 2.10. for simplicity.

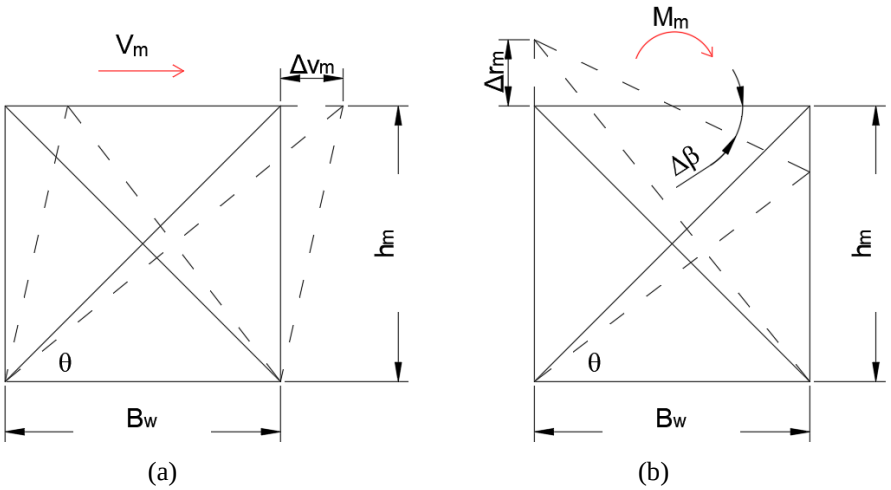


Figure 2.10. *(a) Shear and (b) bending deformations under wind load*

As shown in Figure 2.10.(a), the strain due to shear on k-th diagonal on web façade of m-th module, $\varepsilon_{w,k}$, can be evaluated by means of module drift Δv_m, as,

$$\varepsilon_{w,k} = \frac{\Delta v_m \cos\theta_m}{\frac{h_m}{\sin\theta_m}} = \frac{\Delta v_m \sin\theta_m \cos\theta_m}{h_m} = \gamma_m \sin\theta_m \cos\theta_m \tag{2.15}$$

Here, γ_m indicates module shearing strain. Then, the axial force of k-th diagonal on web of m-th module ($F_{w,k}$) can be defined as,

$$F_{w,k} = A_{d,m}\, E\, \varepsilon_{w,k} = A_{d,m}\, E\, \gamma_m \sin\theta_m \cos\theta_m \tag{2.16}$$

Considering the fact that there must be equilibrium between internal and external forces, the following equation is derived (see Eq. 2.17).

$$V_m = 2\sum_{k=1}^{n_w}\left(F_{w,k}\cos\theta_m\right) = 2\, A_{d,m}\, E\, \gamma_m \sum_{k=1}^{n_w}(\sin\theta_m \cos^2\theta_m) \tag{2.17}$$

On the other hand, the strain on k-th diagonal on flange façade of m-th module caused by overturning moment (see Figure 2.10b), $\varepsilon_{f,k}$, is evaluated using module rotation $\Delta\beta_m$, as,

$$\varepsilon_{f,k} = \frac{\Delta r_m \ \sin\theta_m}{\frac{h_m}{\sin\theta_m}} = \frac{\Delta r_m \ \sin^2\theta_m}{h_m} = \frac{B_w \ \Delta\beta_m}{2 \ h_m} \ \sin^2\theta_m = \frac{B_w}{2} \ \chi_m \ \sin^2\theta_m \qquad (2.18)$$

Here, χ_m indicates m-th module curvature caused by overturning moment. Thus, axial force on k-th diagonal on flange façade of m-th module ($F_{f,k}$) can be written as,

$$F_{f,k} = A_{d,m} \ E \ \varepsilon_{f,k} = A_{d,m} \ E \ \frac{B_w}{2} \ \chi_m \ \sin^2\theta_m \qquad (2.19)$$

Considering equilibrium between internal and external forces, it can be derived as,

$$M_m = B_w \sum_{k=1}^{n_f} \left(F_{f,k} \ \sin\theta_m\right) = \frac{1}{2} \ A_{d,m} \ E \ B_w^2 \ \chi_m \ \sum_{k=1}^{n_f} \sin^3\theta_m \qquad (2.20)$$

As mentioned in Section 2.3.1; while resisting shear some flange diagonals, and while resisting moment some web diagonals can help due to distribution of the loads on diagonal intersections and module geometry. Hence, two parameters; δ_w for web diagonal contribution on resisting moment, and δ_f for flange diagonal contribution on resisting shear, is defined in order to include these effects in calculations. Basically, these parameters indicate an additional value to n_w for shear resisting, and to n_f for moment resisting. δ_w and δ_f is evaluated by the help of strength analysis output data, since the web and flange diagonal axial forces are evaluated differently for each load pattern in the code (see A-6). While $F_{w,m,V}$ and $F_{f,m,V}$ indicate the total axial forces of web and flange diagonals on m-th module, respectively, due to shear; and $F_{w,m,M}$ and $F_{f,m,M}$ indicate the resultant axial loads on web and flange diagonals on m-th module, respectively, due to overturning moment, δ_w and δ_f can be defined as,

$$\delta_w = \frac{F_{w,m,M}}{F_{f,m,M}} \qquad (2.21)$$

$$\delta_f = \frac{F_{f,m,V}}{F_{w,m,V}} \qquad (2.22)$$

Hence, from Eq. 2.17 and 2.20, χ_m/γ_m ratio on m-th module can be derived as,

$$\frac{\chi_m}{\gamma_m} = \frac{4 \ M_m \ (n_w + \delta_f)(\sin\theta_m \ \cos^2\theta_m)}{V_m \ B_w^2 \ (n_f + \delta_w)(\sin^3\theta_m)} \qquad (2.23)$$

In general, while designing tall buildings, maximum roof deflection - which consists of bending and shear deformations - is considered as a stiffness parameter since most of the building codes indicate only roof drift limit [12]. However, the deformed

shapes under moment and shear differs prominently. Thus, in this paper inter-module drift is considered as stiffness design parameter.

The lateral displacement of m-th module, $u(H_m)$, can be approximately derived by means of γ_m and χ_m as,

$$u(H_m) = \sum_{j=1}^{m} \gamma_j \, h_j + \sum_{j=1}^{m} \chi_j h_j (H_m - H_{j-1}) \tag{2.24}$$

The same equation is derived for (m-1)-th module as,

$$u(H_{m-1}) = \sum_{j=1}^{m-1} \gamma_j \, h_j + \sum_{j=1}^{m-1} \chi_j h_j (H_{m-1} - H_{j-1}) \tag{2.25}$$

While $H_{m-1} - H_m = h_m$, from Eq. 2.24 and 2.25, the inter-module drift ratio can be obtained as,

$$\frac{u_m}{h_m} = \gamma_m + \chi_m h_m + \sum_{j=1}^{m-1} \chi_j h_j \leq \alpha \tag{2.26}$$

And for m = 1, which indicates base module, it can be written as,

$$\gamma_m + \chi_m h_m = \alpha \tag{2.27}$$

Here, α indicates inter-module drift ratio limit which is an input parameter in the code. In this study, as most of the building codes permit, α value is defined as 0.002 [12].

From Eq. 2.23 and 2.27, γ_1 and χ_1 values are found for base module. After that, consecutively solving Eq. 2.23 and 2.26 for other modules give γ_m and χ_m values for each module. While all γ_m and χ_m are evaluated in this process, these values are used in either Eq. 2.28 or Eq. 2.29 given below, which is derived from Eq. 2.17 and 2.20, in order to evaluate the required minimum area according to stiffness design.

$$A_{d,m,stiff} = \frac{V_m}{2 \, E \, \gamma_m \, (n_w + \delta_f)(\sin\theta_m \, \cos^2\theta_m)} \tag{2.28}$$

$$A_{d,m,stiff} = \frac{2 \, M_m}{E \, B_w^2 \, \chi_m \, (n_f + \delta_w)(\sin^3\theta_m)} \tag{2.29}$$

After having diagonal cross-section area values known as a result of both strength and stiffness analysis for each module, the maximum of these values for each module is selected as final result by the code. In the case of $A_{d,m,str} > A_{d,m,stiff}$, the code calculates the corresponding inter-module drift ratio accordingly (see A-6).

The diagonals are designed as pipe sections. In order to evaluate the preliminary diagonal pipe section sizes of modules under compression, an algorithm is also developed and added to the code to define them according to AISC 360-10 [4] LRFD limitations (see A-6).

As stated in [11], nominal compressive strength (P_n), is determined based on the critical stress (F_{cr}) and gross area (A_g) of the member section (Eq. 2.30).

$$P_n = F_{cr}\, A_g \tag{2.30}$$

Here, F_{cr} is evaluated according to Eq. 2.31 or 2.32, as,

$$F_{cr} = \left(0.658^{\sigma_y/F_e}\right) F_y \text{ , when } \frac{\sigma_y}{F_e} \leq 2.25 \tag{2.31}$$

$$F_{cr} = 0.877\, F_e \text{ , when } \frac{\sigma_y}{F_e} \geq 2.25 \tag{2.32}$$

Elastic buckling stress, F_e, is determined according to Eq. 2.33, as,

$$F_e = \frac{\pi^2 E}{\left(\frac{K\, L_d}{r}\right)^2} \tag{2.33}$$

K = 1 for truss members, and radius of gyration, r, of a pipe section can be defined as,

$$r = \sqrt{\frac{r_1^2 + r_2^2}{4}} \tag{2.34}$$

where r_1 is inner, and r_2 is outer radius of the pipe. Hence, F_e can be defined by,

$$F_e = \frac{\pi^2 E\, (r_1^2 + r_2^2)}{4\, L_d^2} \tag{2.35}$$

In addition, for a pipe section gross area, it can be written as,

$$A_g = \pi(\, r_2^2 - r_1^2) \tag{2.36}$$

For $\sigma_y/F_e = 1$, using Eq. 2.35 and 2.36, r_1 and r_2 of the pipe section can be evaluated as,

$$r_1 = \frac{2\, F_y\, L_d^2}{\pi^2\, E} - \frac{A_g}{2\,\pi} \text{ , and } r_2 = \frac{2\, F_y\, L_d^2}{\pi^2\, E} + \frac{A_g}{2\,\pi} \tag{2.37}$$

Since the design compression force, P_d, is defined as,

$$P_d \geq 0.9\, P_n \tag{2.38}$$

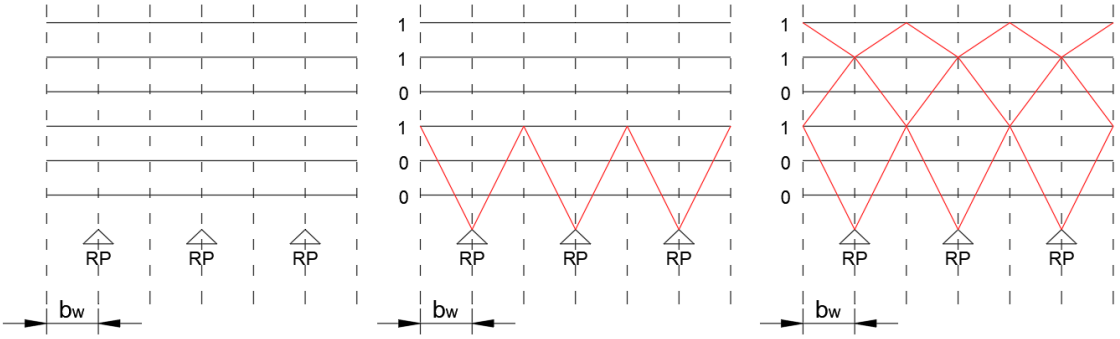
using Eq. 2.30, 2.31 and 2.38, it can be concluded as in Eq. 2.39, which has the same factor value used in strength analysis for compression member design in Eq. 2.14.

$$P_d = 0.5922\, \sigma_y\, A_g \approx 0.6\, \sigma_y\, A_g \tag{2.39}$$

Therefore, the pipe section sizes for each module are defined according to Eq. 2.37, after some rounding applications are made to the results in order to have reasonable values.

2.3.3. Optimization

The diagonal geometry definition procedure proposed in this study is modeled based on a binary system. Therefore, a binary encoded Genetic Algorithm is utilized in order to have consistency. To be able to encode the diagonal geometry the following steps are carried out (see A-5).



(a) (b) (c)

Figure 2.11. *Diagonal geometry definition stages in code*

According to input $n_{RP,}$ B_w, and B_f, the bay width (b_w) for a diagonal line is evaluated which indicates the horizontal region edges for each diagonal on a façade of a module (see Figure 2.11a).

The story number of the structure is considered as the length of the position vector of the populations which is constant for a predefined building core model. Hence, population position vectors are filled with ones and zeros. By multiplying the index number of the "1" values in the position vector with story height, the z-axis coordinates of the diagonal end points are defined. First for module diagonals (see 2.11b), then for all the modules to the top (see 2.11c), this process is repeated.

For the optimization process by Genetic Algorithm, first initial population positions are created randomly as binary numbers. As mentioned above, these positions are used to define diagonal geometry (member end-coordinates and member length). According to these positions fitness, which indicates weight of the diagrid shell in this study, is evaluated and populations are sorted in terms of their fitness.

Afterwards, main loop phase, which consists of crossover and mutation processes, takes place. In crossover process (see. A-8), two parents are selected within the population by tournament selection method (see A-7). To these parent populations crossover process is applied, which can be explained as data exchange between parents' position vector, and two new populations are created as a result. After fitness evaluation of crossover populations, a parent is selected within population for mutation process, which is data changes in position vector. Then, fitness of this mutated population is also evaluated.

After the completion of crossover and mutation processes, which is the end of each iteration, all populations are merged and sorted according to their fitness value. The excess populations that are last at fitness ranking are truncated and next iteration starts.

This process is repeated until a defined convergence ratio is provided or a defined number of iterations were completed. And the population with best fitness rank is selected as best solution. An example geometry progress throughout the iterations for 60-story model is shown on Figure 2.12.

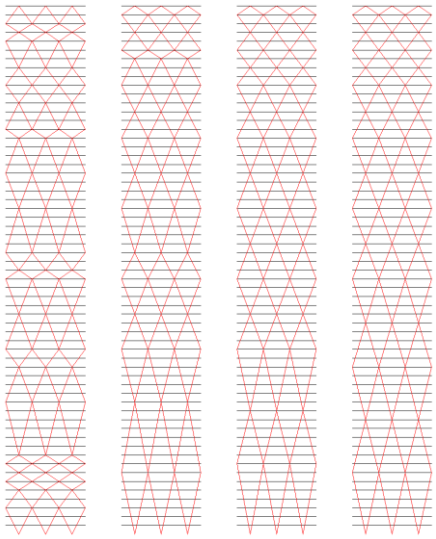


Figure 2.12. *Diagonal geometry progress on (a) 1st iteration, (b) 50th iteration, (c) 200th iteration, and (d) 2000th iteration*

3. RESULTS

According to strength and stiffness based designs and optimization method mentioned in this study, uniform and varying angle models are analyzed by Matlab codes.

The weight results of uniform models with different diagonal angles show that; while diagonal slope increases, first a decrement then an increment occur on weight values as shown on Figure 3.1-3.10.

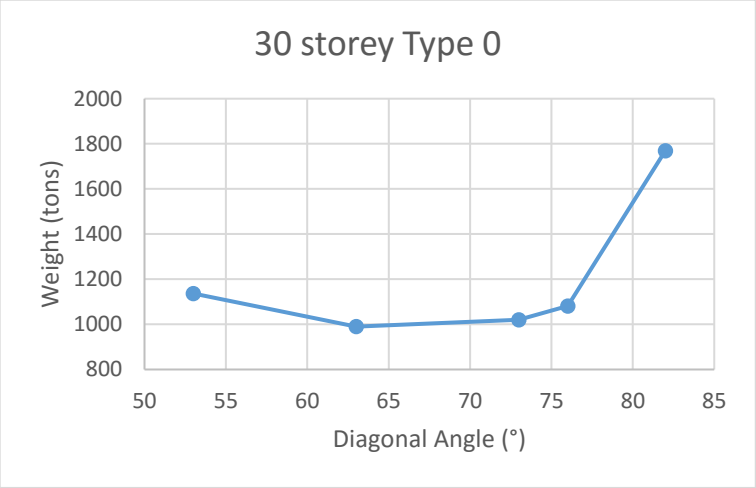


Figure 3.1. *Weight results due to diagonal angle for 30-story Type 0*

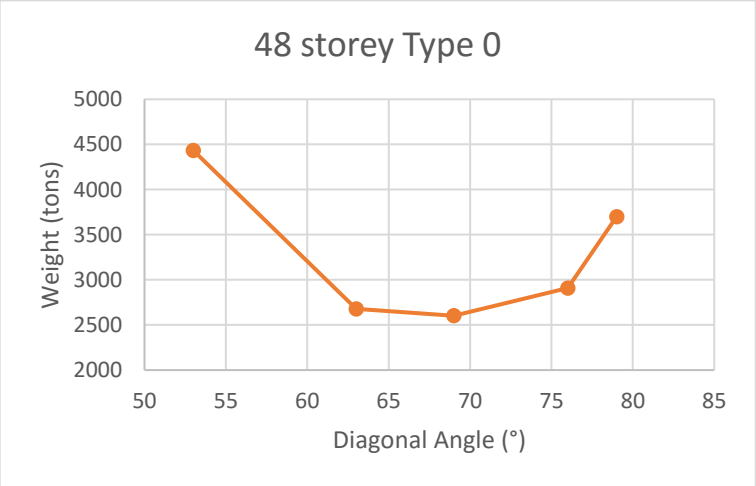


Figure 3.2. *Weight results due to diagonal angle for 48-story Type 0*

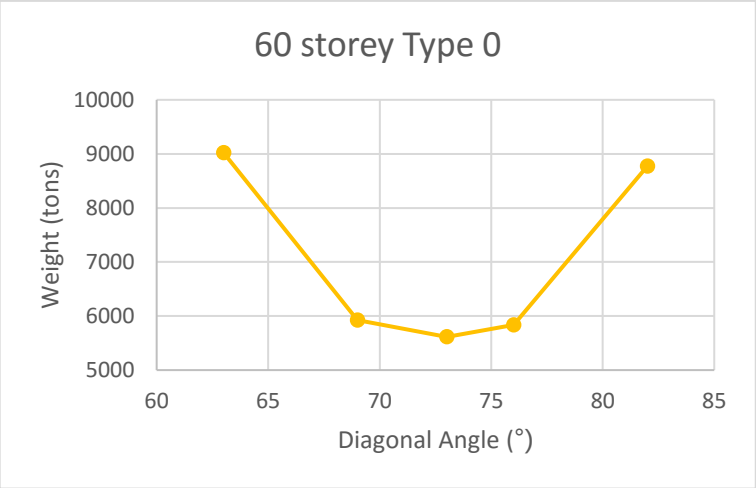


Figure 3.3. *Weight results due to diagonal angle for 60-story Type 0*

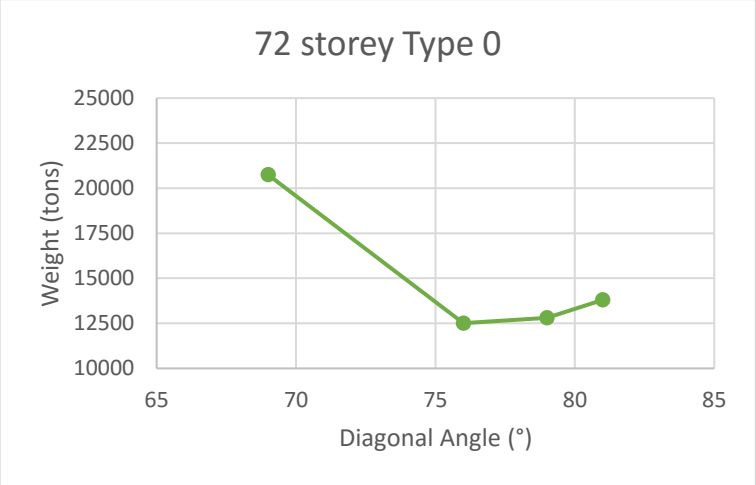


Figure 3.4. *Weight results due to diagonal angle for 72-story Type 0*

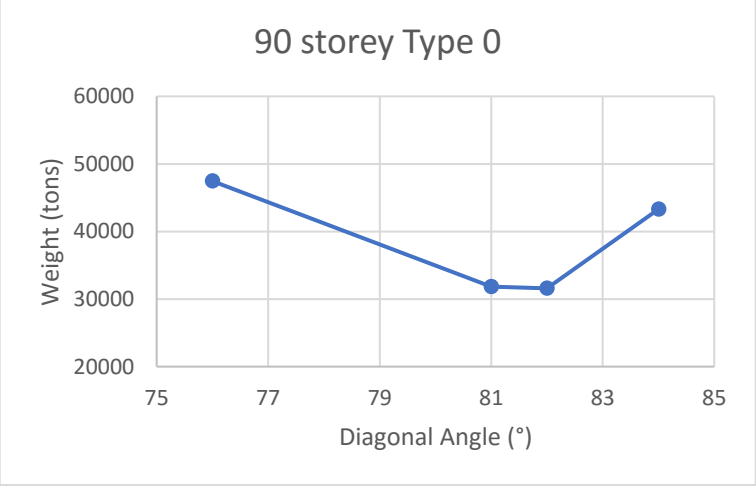


Figure 3.5. *Weight results due to diagonal angle for 90-story Type 0*

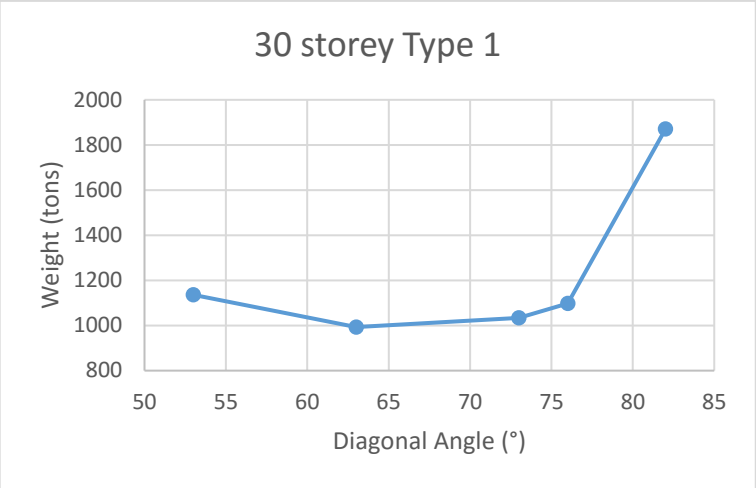


Figure 3.6. *Weight results due to diagonal angle for 30-story Type 1*

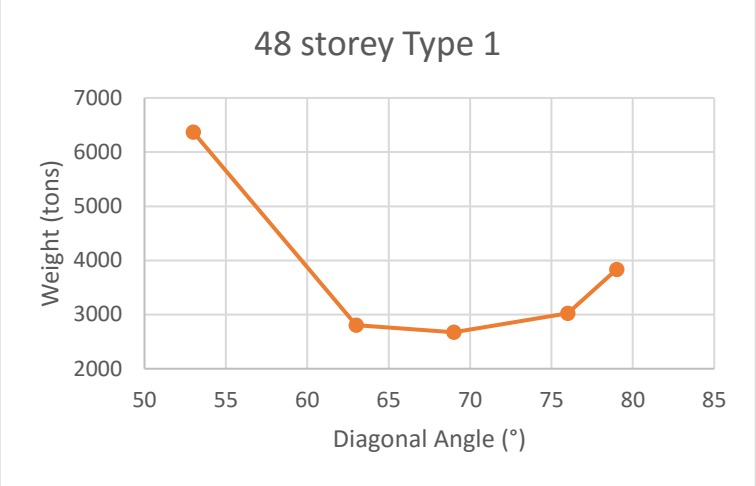


Figure 3.7. *Weight results due to diagonal angle for 48-story Type 1*

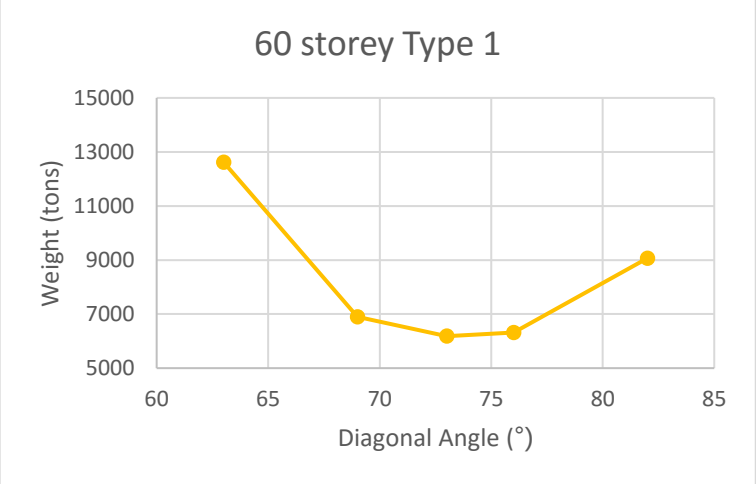


Figure 3.8. *Weight results due to diagonal angle for 60-story Type 1*

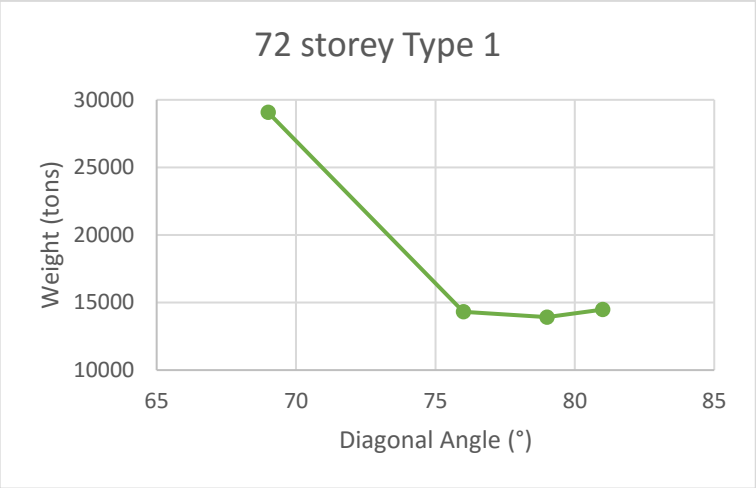


Figure 3.9. *Weight results due to diagonal angle for 72-story Type 1*

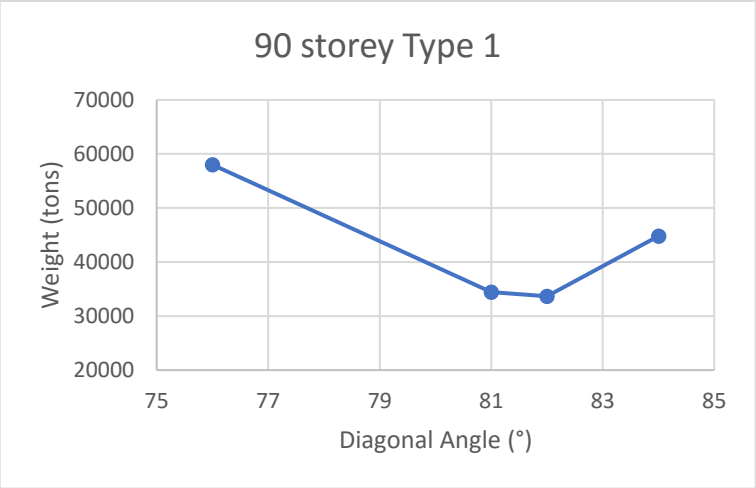


Figure 3.10. *Weight results due to diagonal angle for 90-story Type 1*

These figures also show that optimum diagonal angles for uniform models are independent in terms of Type 0 and Type 1.

The optimum diagonal angles of uniform models according to these plots are considered in terms of aspect ratio, and the results indicate that optimum diagonal angle increases as the aspect ratio increases (see Figure 3.11). The resultant optimum angles were found as 63°, 69°, 73°, 79° and 82° for 30, 48, 60, 72 and 90-story models, respectively.

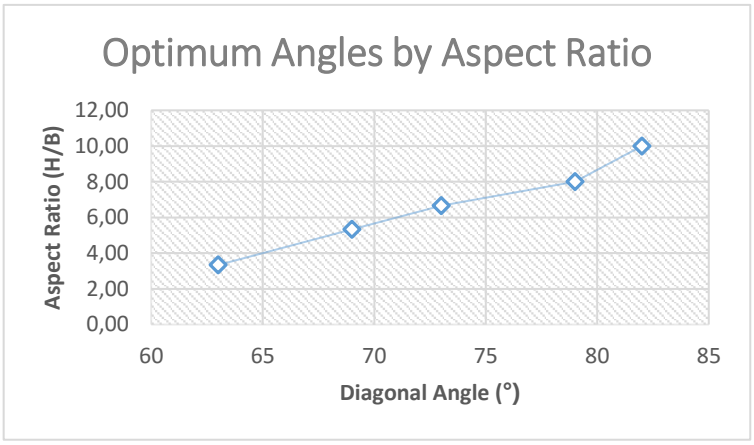


Figure 3.11. *The change of optimum diagonal angle by aspect ratio of uniform models*

In addition, taking into consideration the weights of optimized models and optimum uniform models, the results are shown on Figure 3.12, 3.13 for the sake of comparison. From these figures, it can be remarked that, a considerable weight difference occurs between optimum uniform and optimized model weights while model height, hence aspect ratio, increases. The details of these results are shown in Table 3.1.

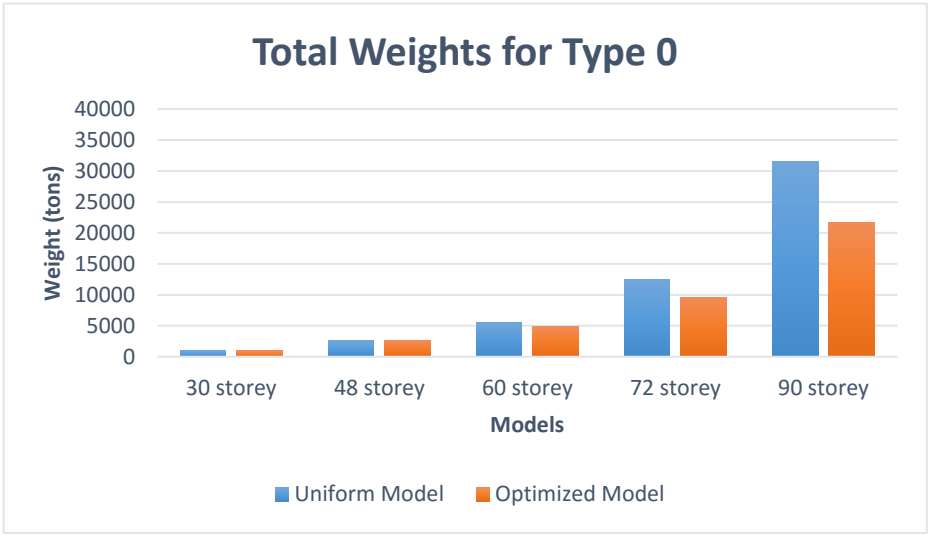


Figure 3.12. *Weight comparison between optimum uniform and optimized models for Type 0*

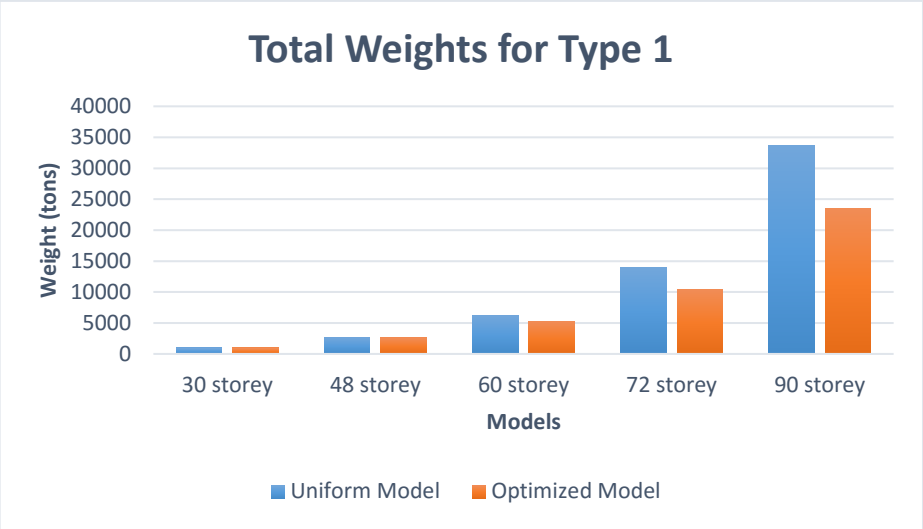


Figure 3.13. *Weight comparison between optimum uniform and optimized models for Type 1*

Table 3.1. *Weight and profit percentage results for optimum uniform and optimized models*

Nunmber of Stories	Type 0			Type 1		
	Optimum Uniform Weight (tons)	Optimized Weight (tons)	Profit (%)	Optimum Uniform Weight (tons)	Optimized Weight (tons)	Profit (%)
30	990	978	1.21	993	992	0.10
48	2602	2583	0.73	2673	2617	2.10
60	5614	4858	13.47	6183	5273	14.72
72	12510	9595	23.30	13921	10397	25.31
90	31585	21769	31.08	33668	23522	30.14

As shown on Table 3.1, on 30 and 48-story models, profit percentages are negligible which indicates that for corresponding aspect ratios of 3.33 and 5.33, optimization process is not needed. On the other hand, 60, 72 and 90-story models, which have aspect ratios of 6.67, 8, and 10, have considerable profit percentages up to 31.08%. Thus, it can be concluded that the optimization process is needed for buildings that have aspect ratios more than 6 in order to design a material economic diagrid structure. In addition, for both optimum uniform and optimized models, Type 0 models have less weight than Type 1 models which is the result of load distribution difference according to geometry. This outcome is because of the high sloped long diagonals at base module - which is the resultant topology of the optimization for optimized models and resist maximum gravity

forces - have maximum resultant cross-sectional areas, hence dominates the total weight of the diagrid shell. Since more diagonals are involved in the distribution of shear forces on Type 0, this geometry is more efficient on material usage than Type 1.

Furthermore, as a result of strength and stiffness based designs, the corresponding diagonal area results for each module is shown on Figure 3.14 – 3.33. As can be seen in Figure 3.14 - 3.17, 3.24 - 3.27 since the aspect ratio is low, the stiffness design can not dominate the resultant topology. However, for the other models that have high aspect ratio, stiffness design becomes the deciding factor for the diagrid geometry (see Figure 3.18 – 3.23, 3.28 – 3.33). Moreover, the zigzag pattern on these figures are the result of the load distribution difference between module geometry types as mentioned before.

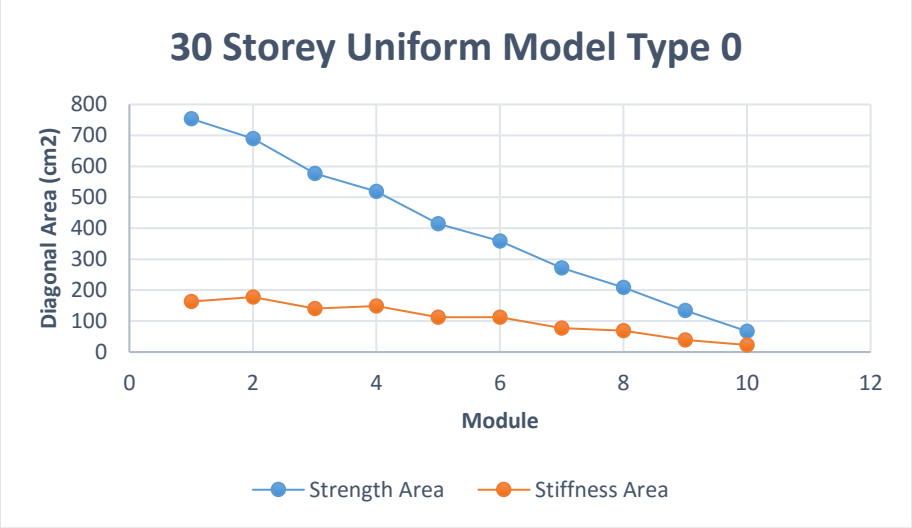


Figure 3.14. *Diagonal area based on strength and stiffness design for 30-story optimum uniform model Type 0*

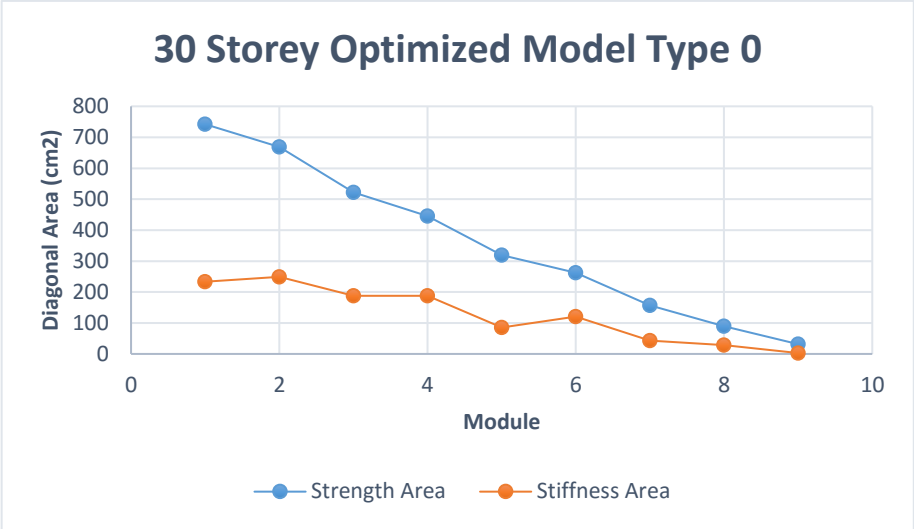


Figure 3.15. *Diagonal area based on strength and stiffness design for 30-story optimized model Type 0*

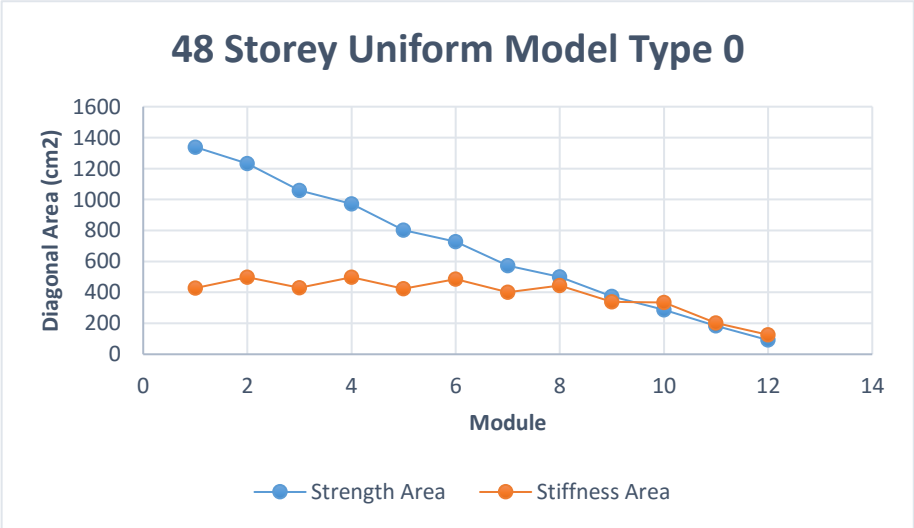


Figure 3.16. *Diagonal area based on strength and stiffness design for 48-story optimum uniform model Type 0*

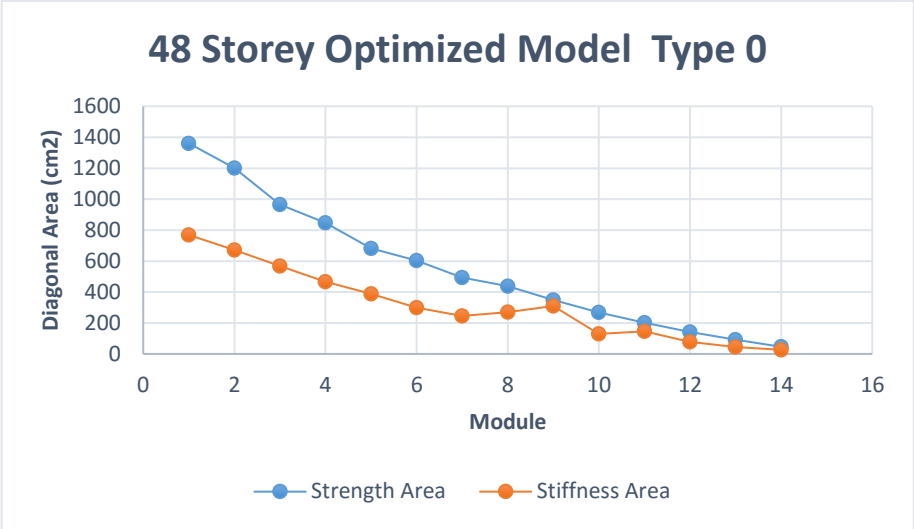


Figure 3.17. *Diagonal area based on strength and stiffness design for 48-story optimized model Type 0*

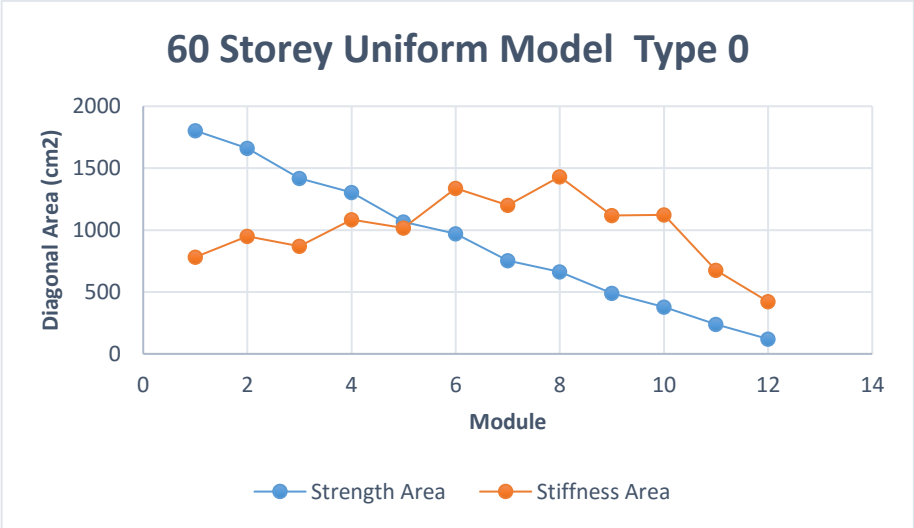


Figure 3.18. *Diagonal area based on strength and stiffness design for 60-story optimum uniform model Type 0*

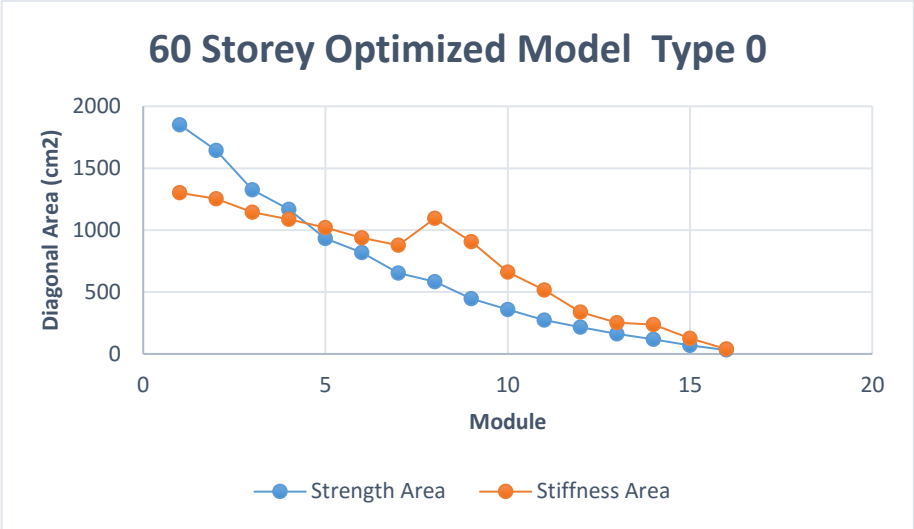


Figure 3.19. *Diagonal area based on strength and stiffness design for 60-story optimized model Type 0*

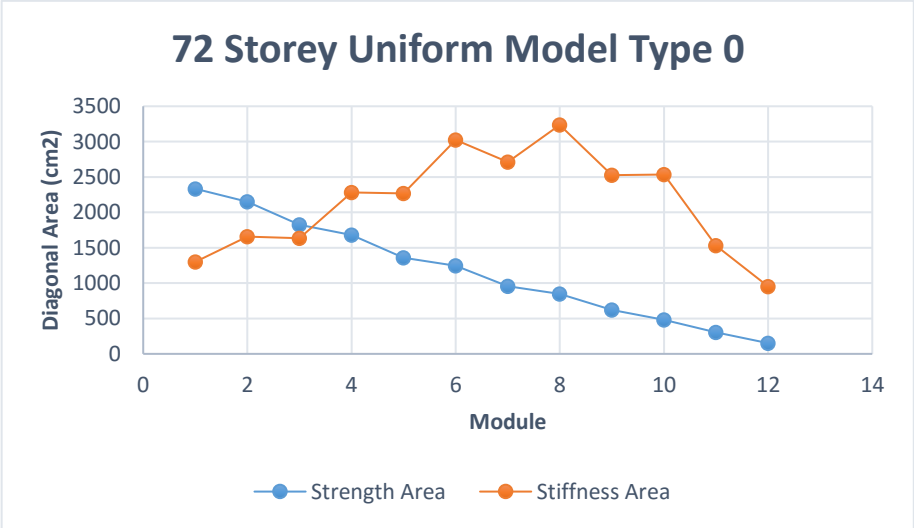


Figure 3.20. *Diagonal area based on strength and stiffness design for 72-story optimum uniform model Type 0*

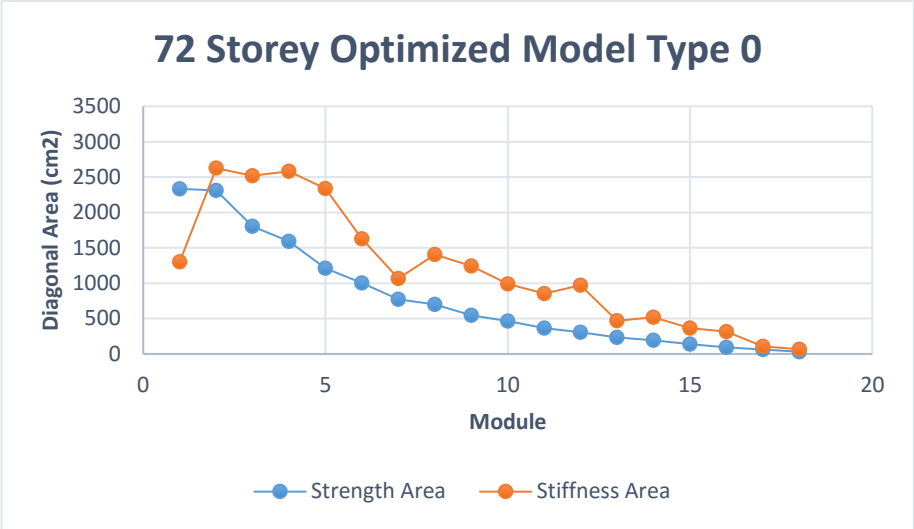


Figure 3.21. *Diagonal area based on strength and stiffness design for 72-story optimized model Type 0*

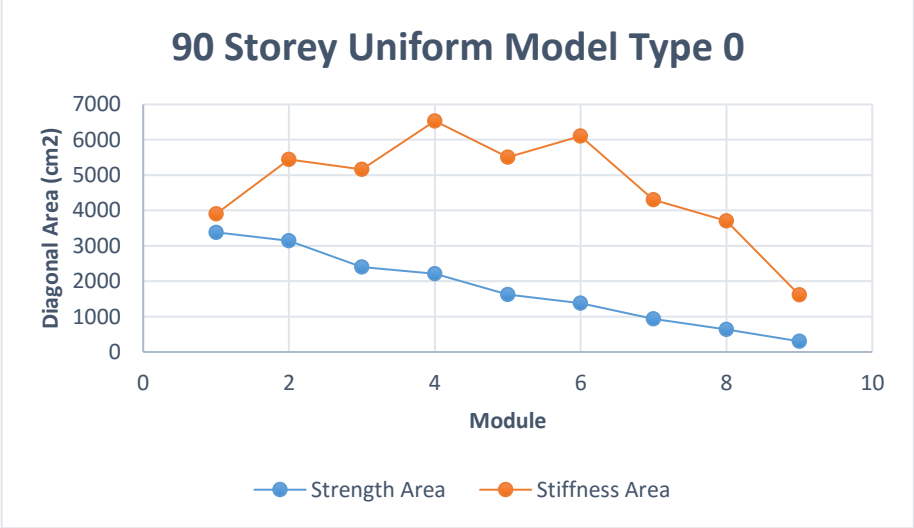


Figure 3.22. *Diagonal area based on strength and stiffness design for 90-story optimum uniform model Type 0*

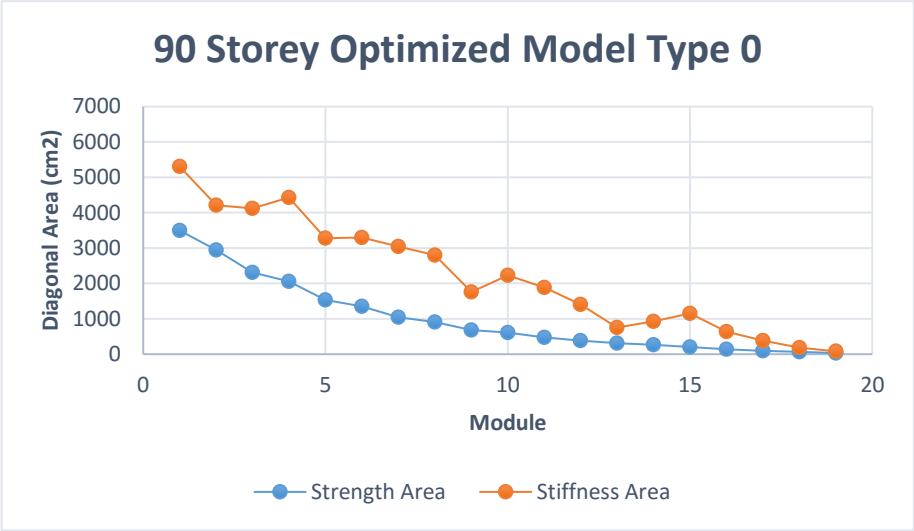


Figure 3.23. *Diagonal area based on strength and stiffness design for 90-story optimized model Type 0*

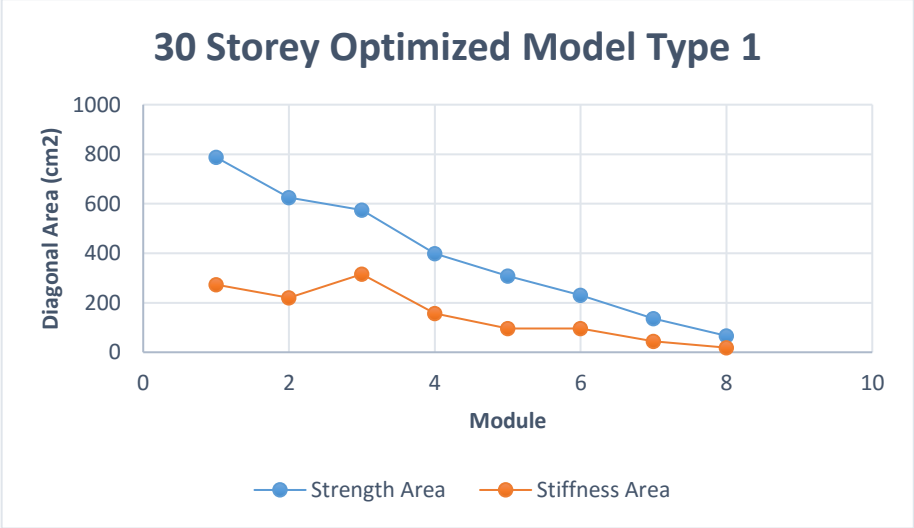


Figure 3.24. *Diagonal area based on strength and stiffness design for 30-story optimum uniform model Type 1*

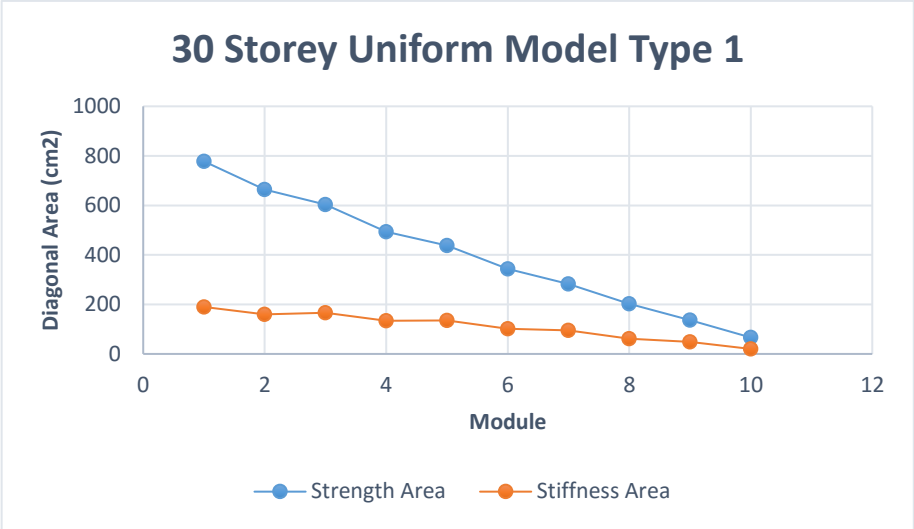


Figure 3.25. *Diagonal area based on strength and stiffness design for 30-story optimized model Type 1*

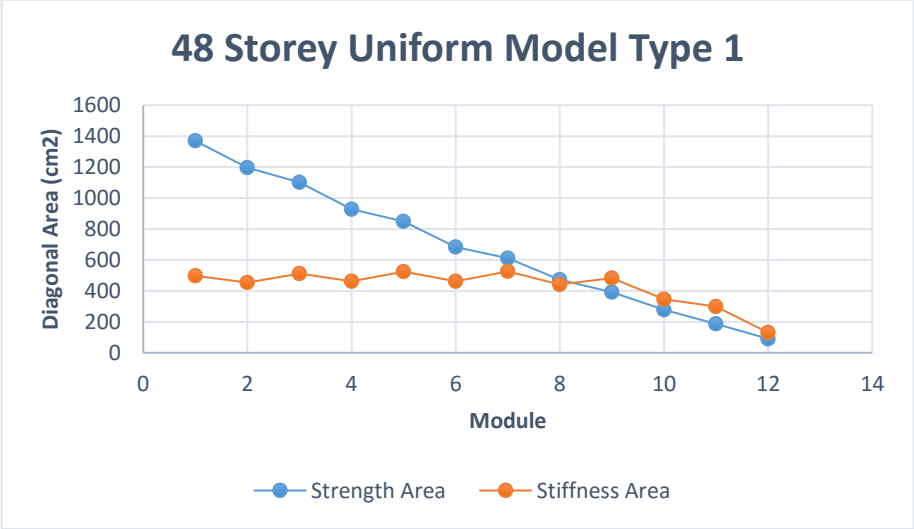


Figure 3.26. *Diagonal area based on strength and stiffness design for 48-story optimum uniform model Type 1*

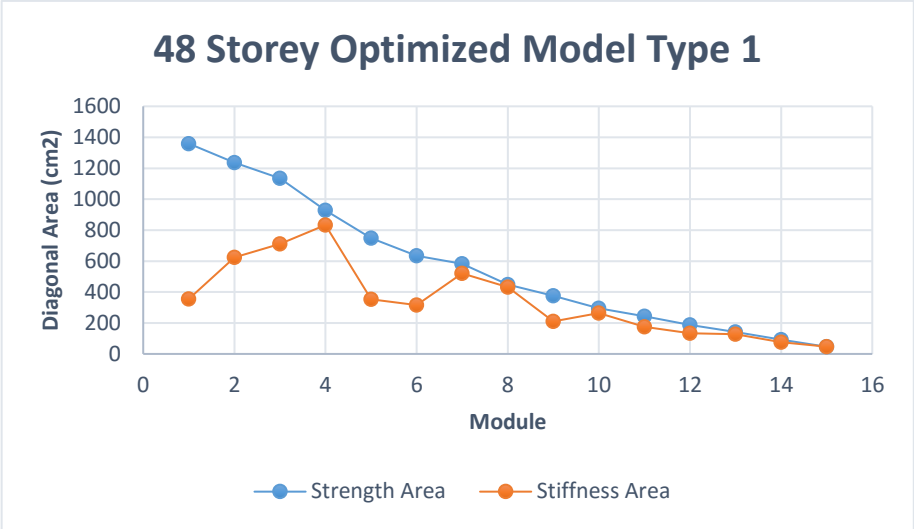


Figure 3.27. *Diagonal area based on strength and stiffness design for 48-story optimized model Type 1*

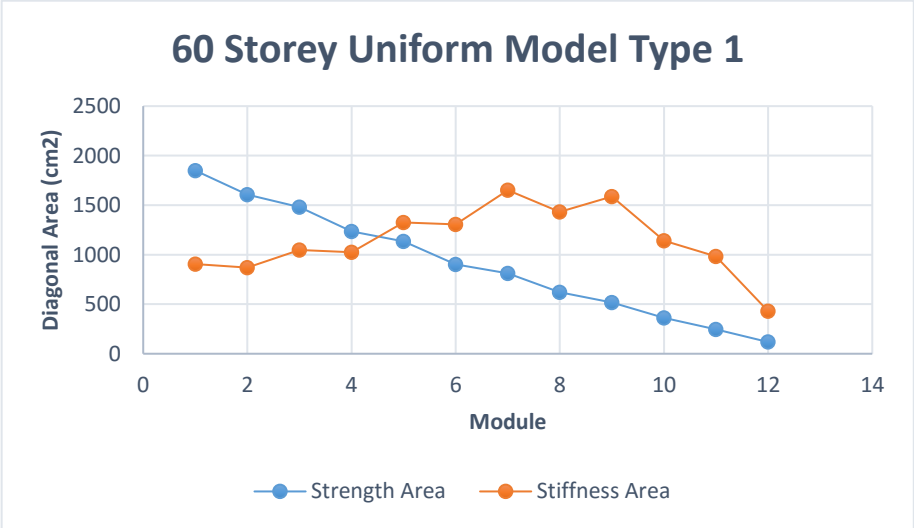


Figure 3.28. *Diagonal area based on strength and stiffness design for 60-story optimum uniform model Type 1*

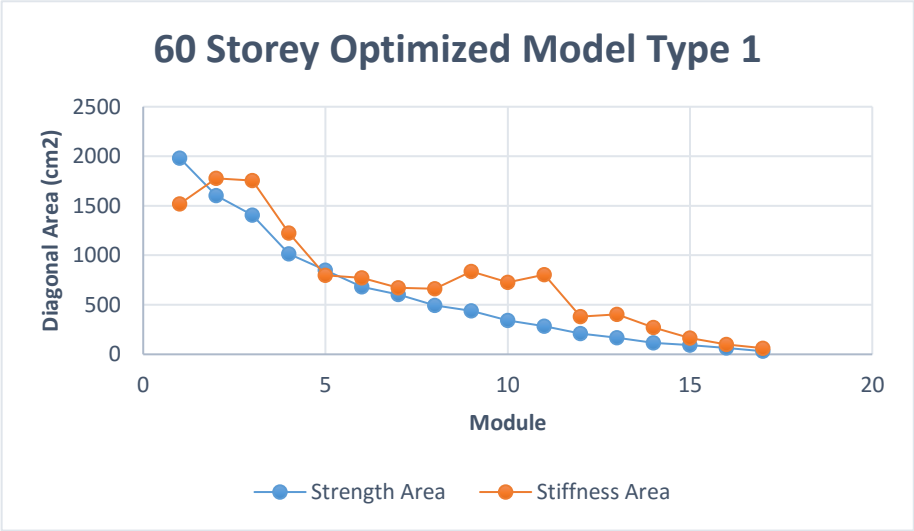


Figure 3.29. *Diagonal area based on strength and stiffness design for 60-story optimized model Type 1*

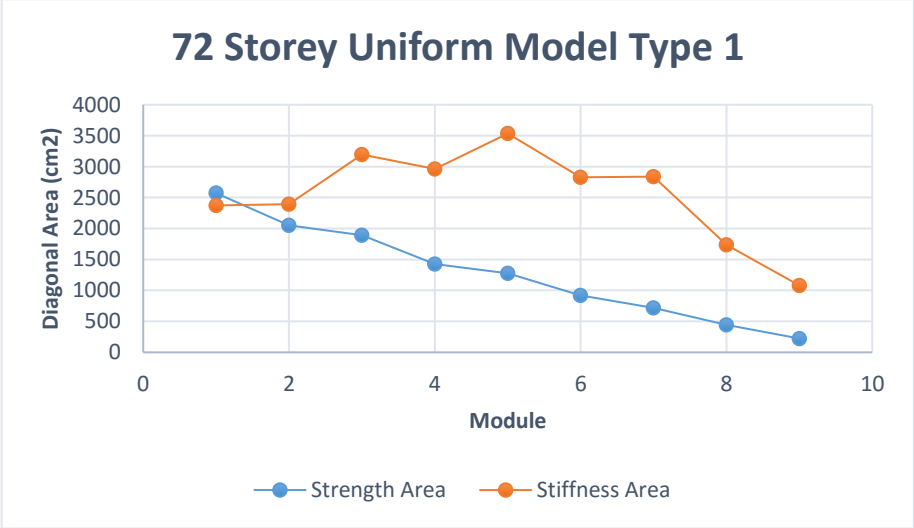


Figure 3.30. *Diagonal area based on strength and stiffness design for 72-story optimum uniform model Type 1*

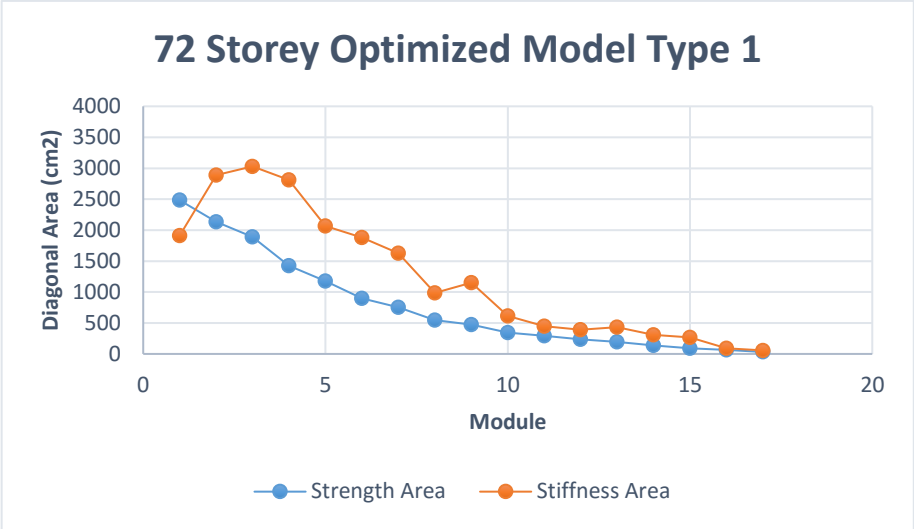


Figure 3.31. *Diagonal area based on strength and stiffness design for 72-story optimized model Type 1*

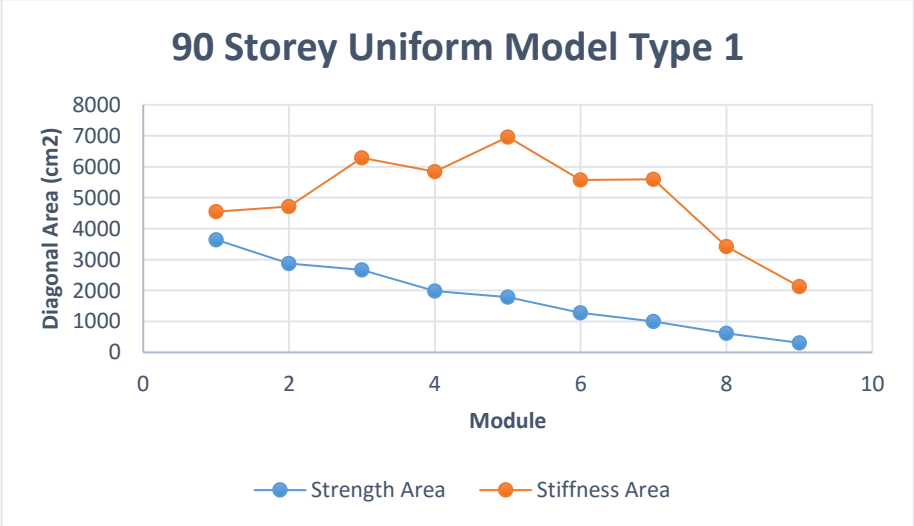


Figure 3.32. *Diagonal area based on strength and stiffness design for 90-story optimum uniform model Type 1*

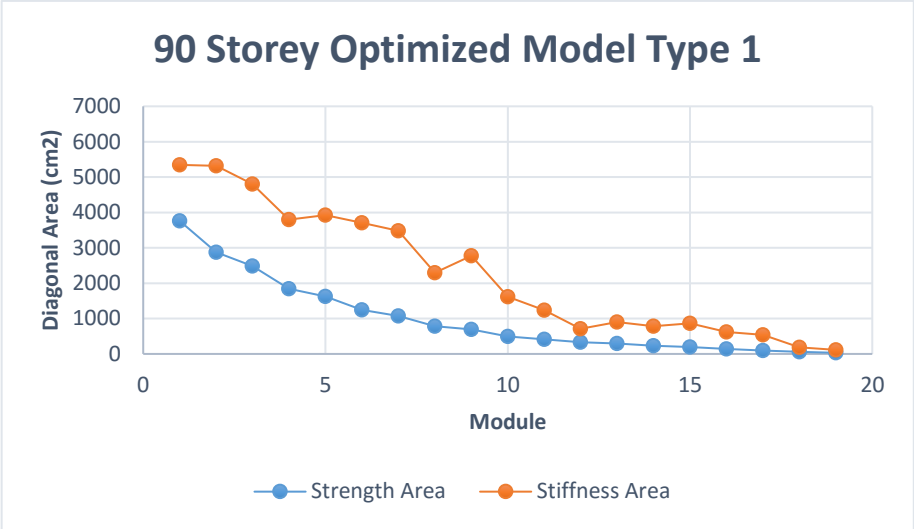


Figure 3.33. *Diagonal area based on strength and stiffness design for 90-story optimized model Type 1*

Furthermore, the resulting lateral displacements of models are shown on Figure 3.34 - 3.53. It can be seen that, on low aspect ratio models, because of the domination of strength design, module lateral displacement values cannot meet the limits (see Figure 3.34 – 3.37). Since the aim of the optimization is reaching drift limit for each module so that the module diagonal areas would be minimum, the efficiency of the optimization is low for these models. However, the lateral displacements converge to the limit as the aspect ratio increases (see Figure 3.38 – 3.53). Hence, it can be pointed out that the optimization efficiency is higher for these models.

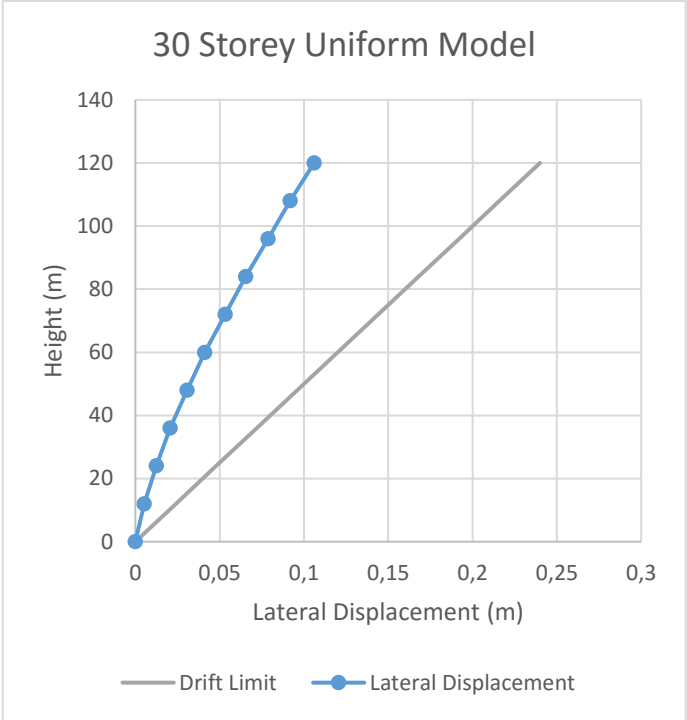


Figure 3.34. *Module lateral displacements of 30-story optimum uniform model Type 0*

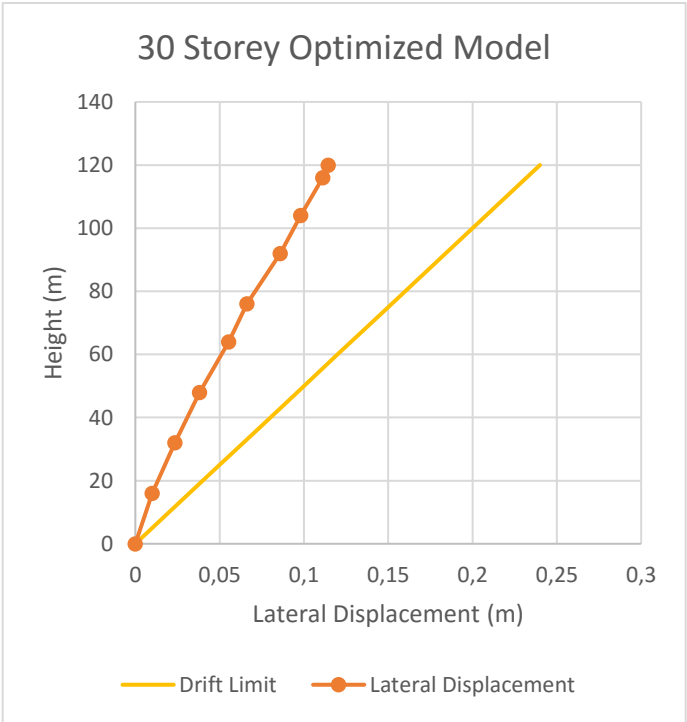


Figure 3.35. *Module lateral displacements of 30-story optimized model Type 0*

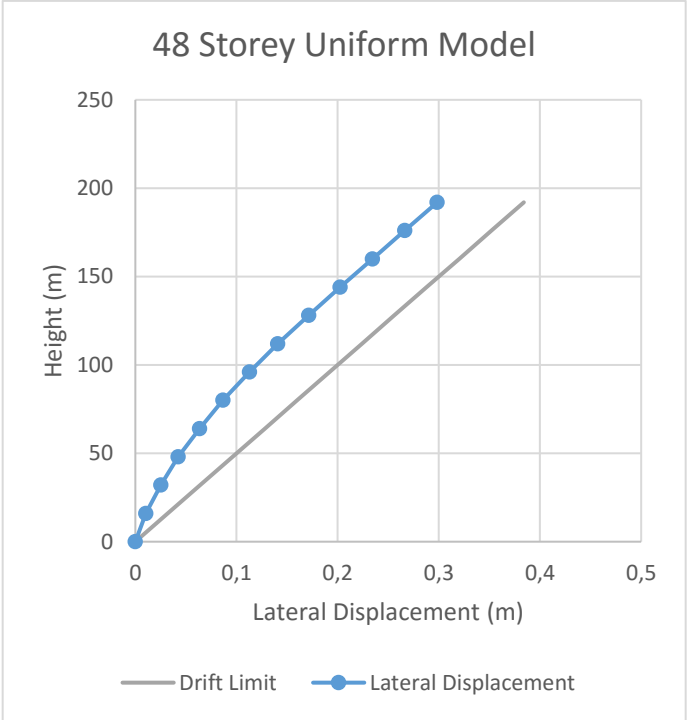


Figure 3.36. *Module lateral displacements of 48-story optimum uniform model Type 0*

Figure 3.37. *Module lateral displacements of 48-story optimized model Type 0*

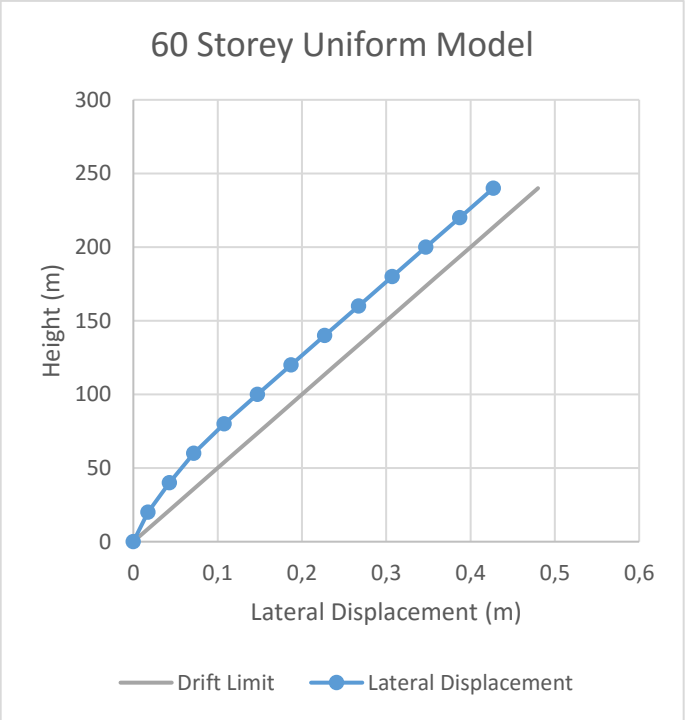


Figure 3.38. *Module lateral displacements of 60-story optimum uniform model Type 0*

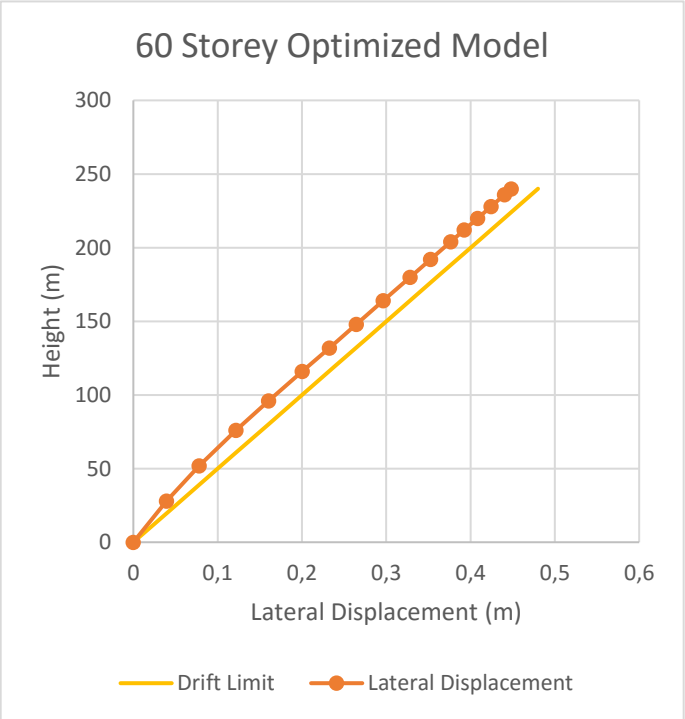


Figure 3.39. *Module lateral displacements of 60-story optimized model Type 0*

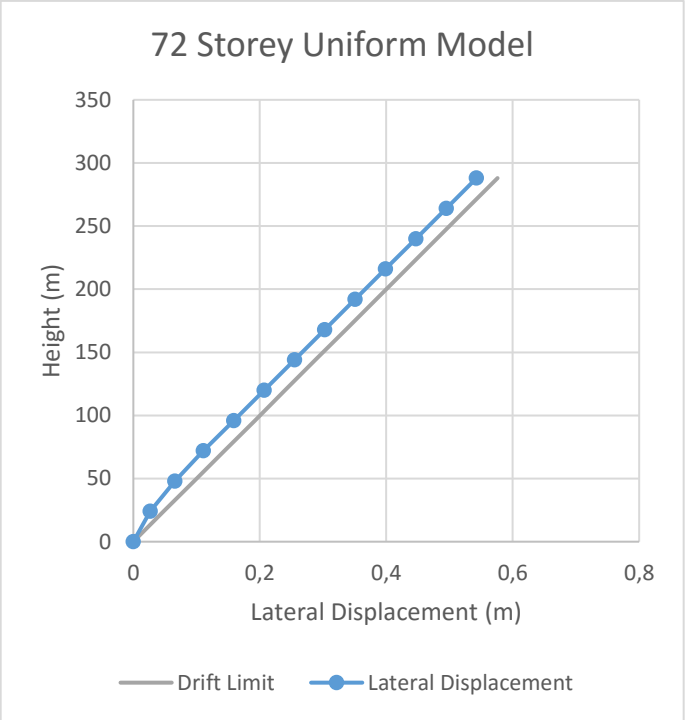


Figure 3.40. *Module lateral displacements of 72-story optimum uniform model Type 0*

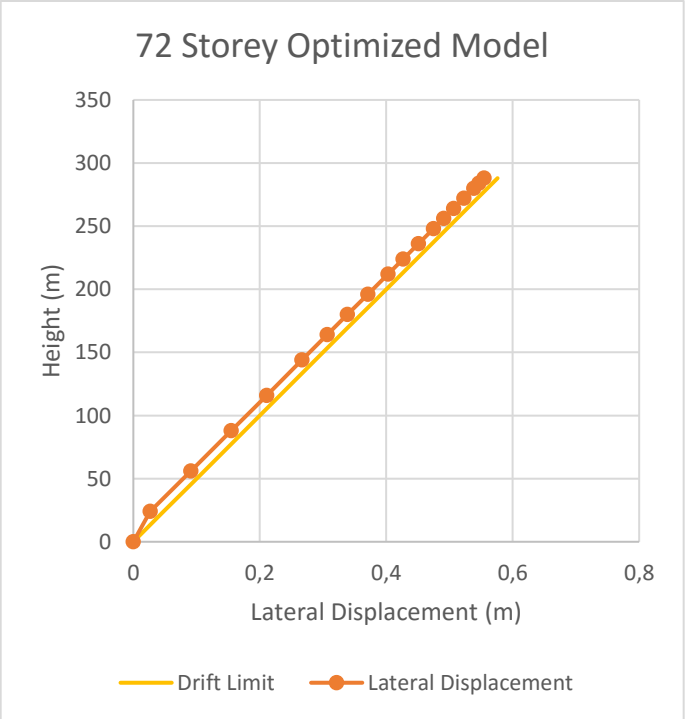


Figure 3.41. *Module lateral displacements of 72-story optimized model Type 0*

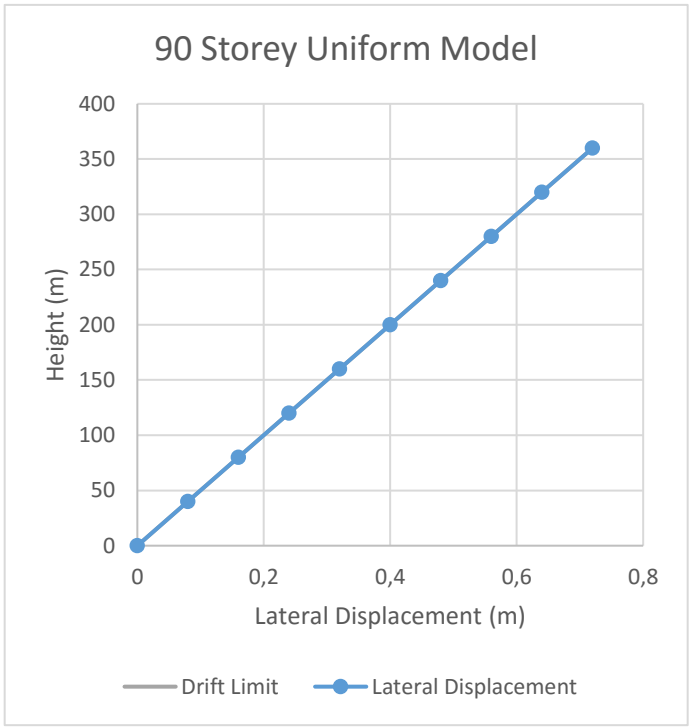


Figure 3.42. *Module lateral displacements of 90-story optimum uniform model Type 0*

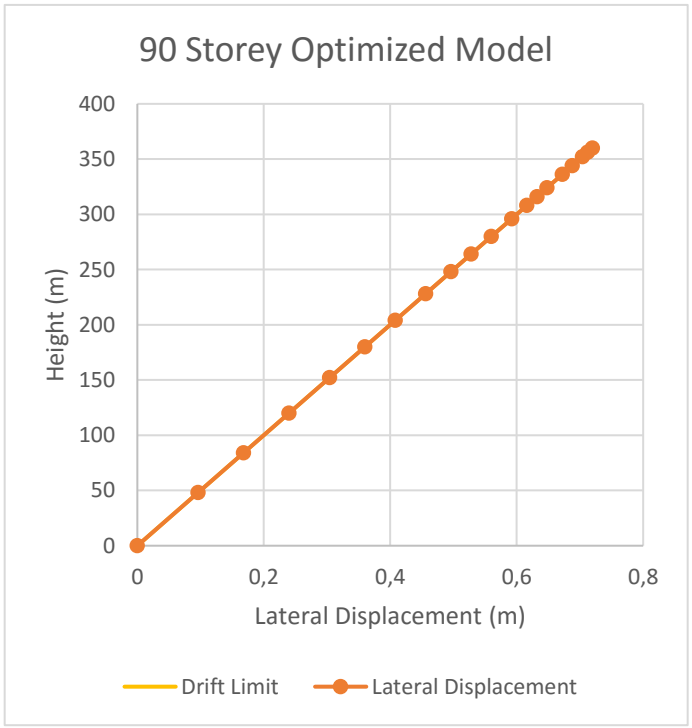


Figure 3.43. *Module lateral displacements of 90-story optimized model Type 0*

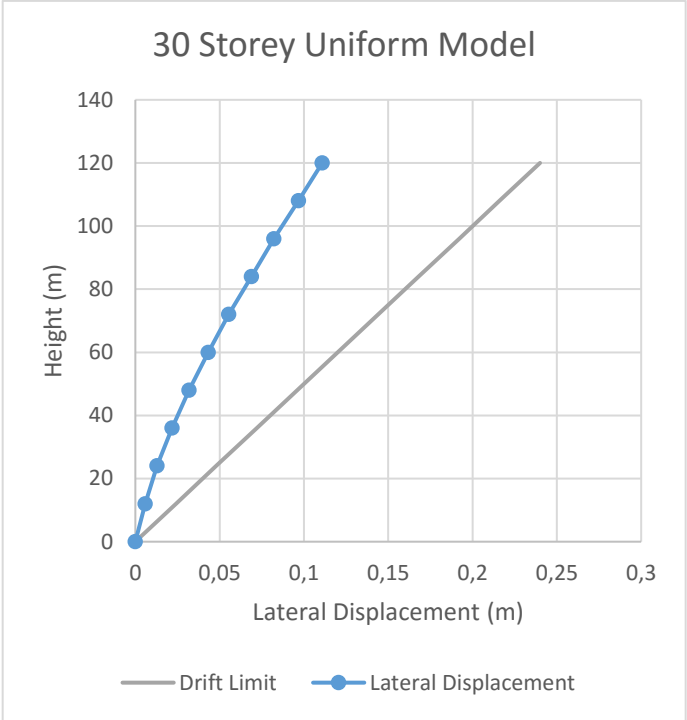


Figure 3.44. *Module lateral displacements of 30-story optimum uniform model Type 1*

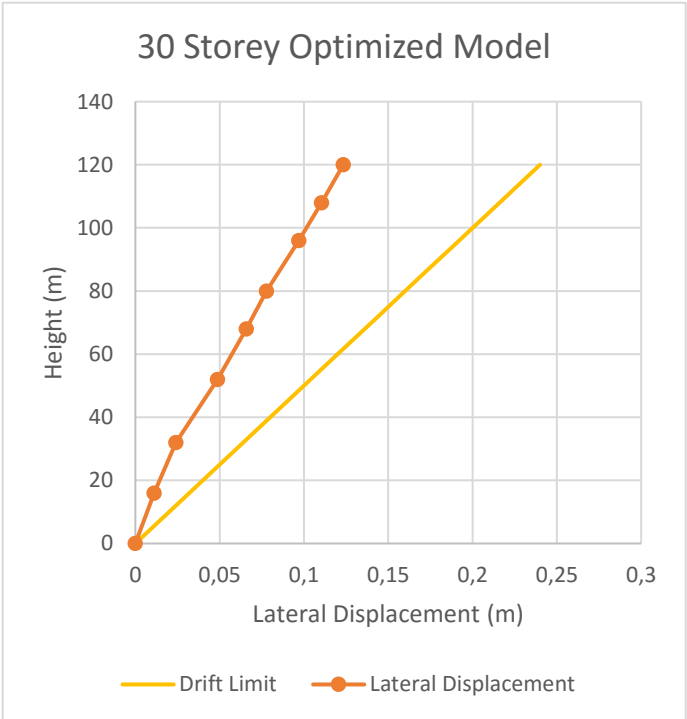


Figure 3.45. *Module lateral displacements of 30-story optimized model Type 1*

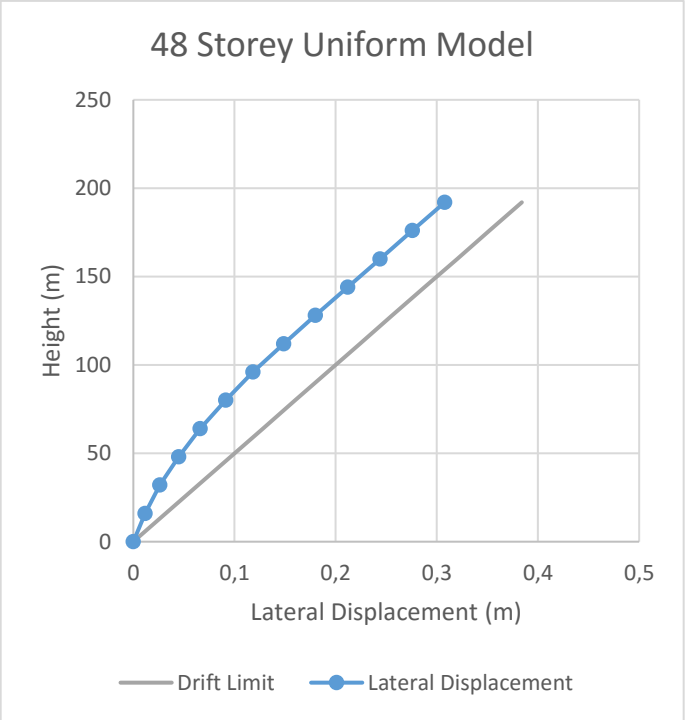


Figure 3.46. *Module lateral displacements of 48-story optimum uniform model Type 1*

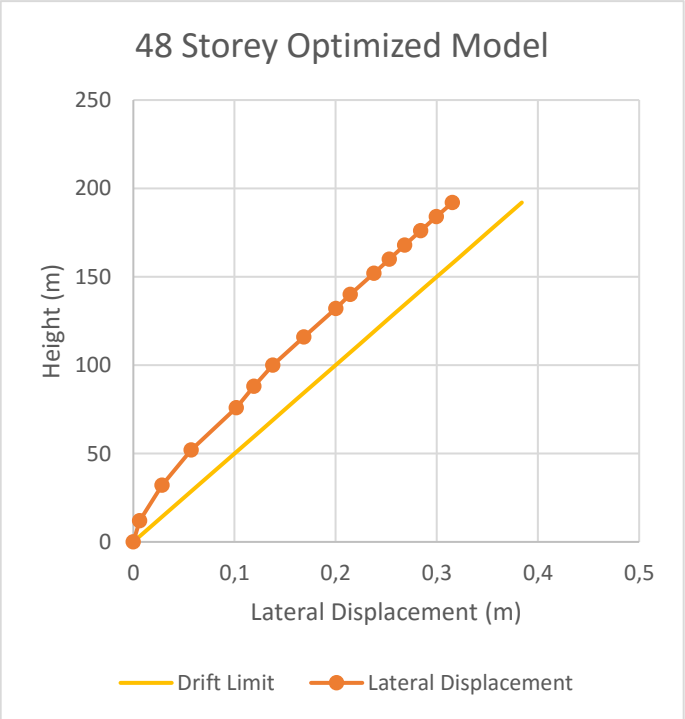


Figure 3.47. *Module lateral displacements of 48-story optimized model Type 1*

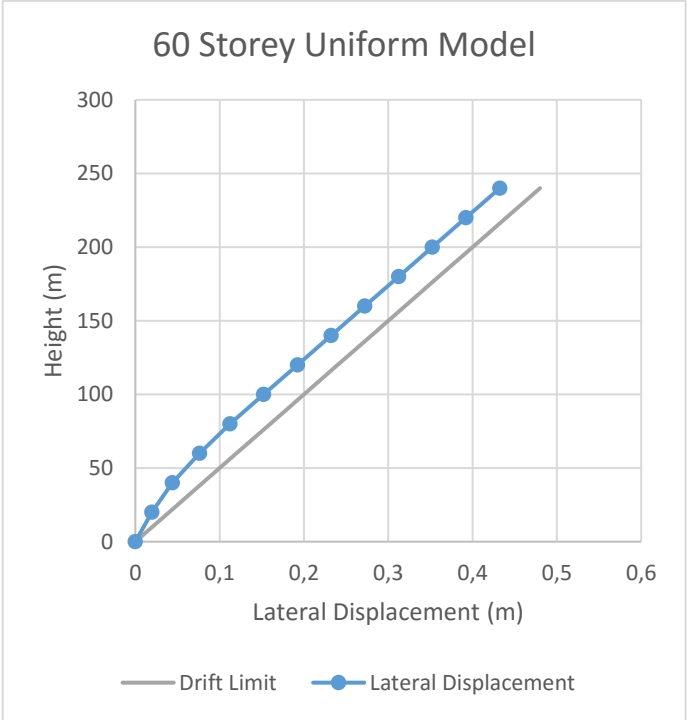


Figure 3.48. *Module lateral displacements of 60-story optimum uniform model Type 1*

Figure 3.49. *Module lateral displacements of 60-story optimized model Type 1*

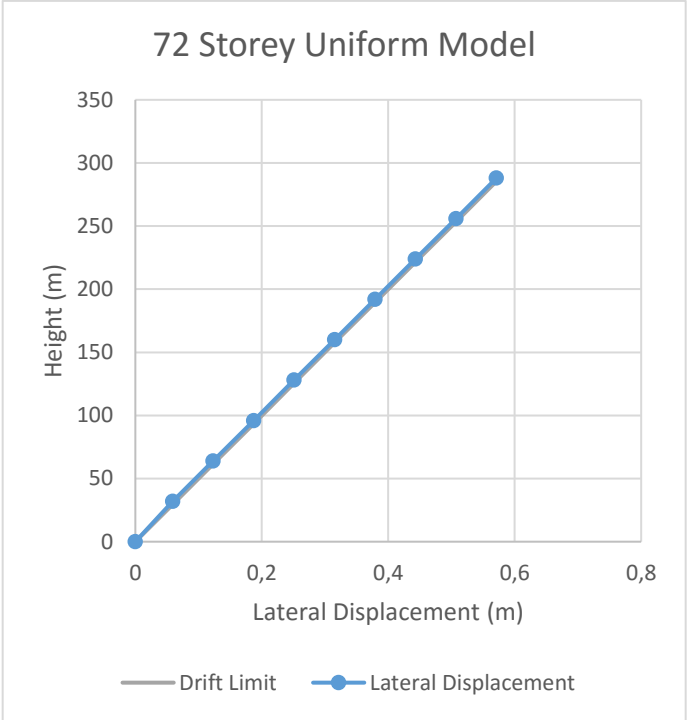


Figure 3.50. *Module lateral displacements of 72-story optimum uniform model Type 1*

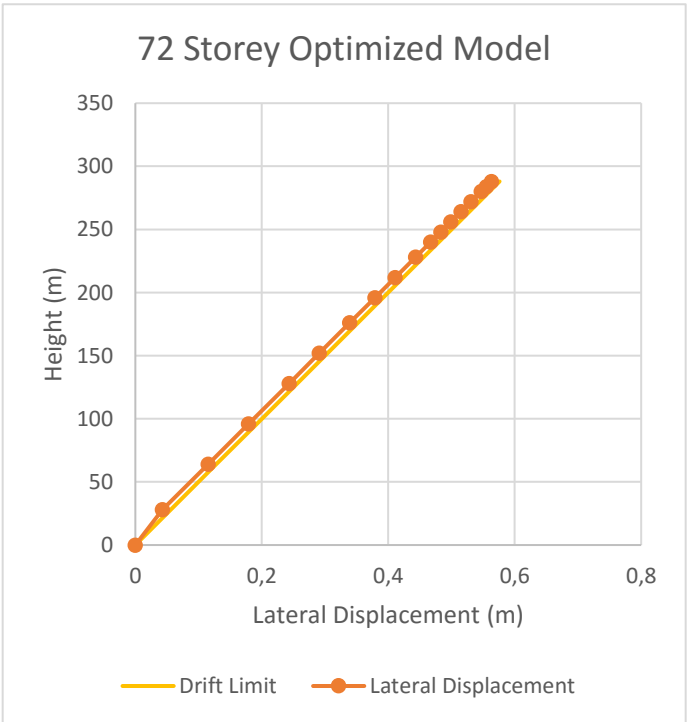


Figure 3.51. *Module lateral displacements of 72-story optimized model Type 1*

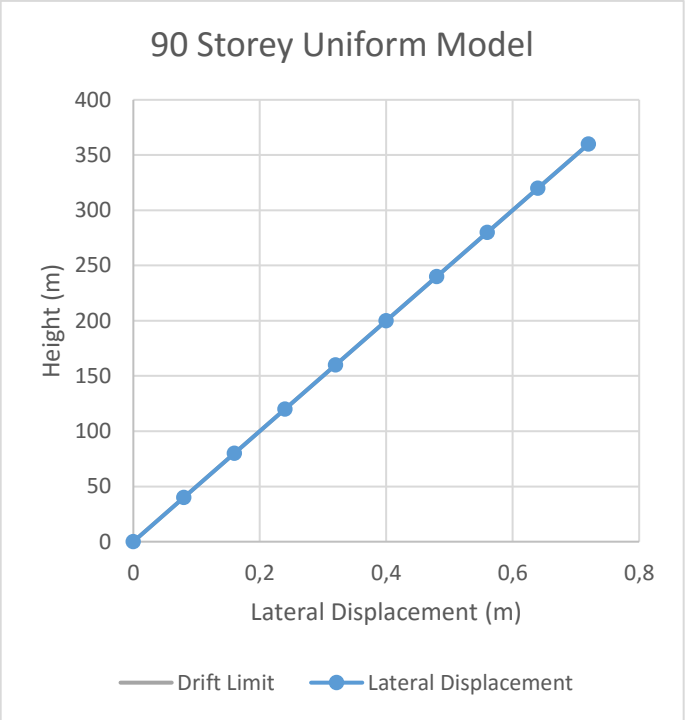


Figure 3.52. *Module lateral displacements of 90-story optimum uniform model Type 1*

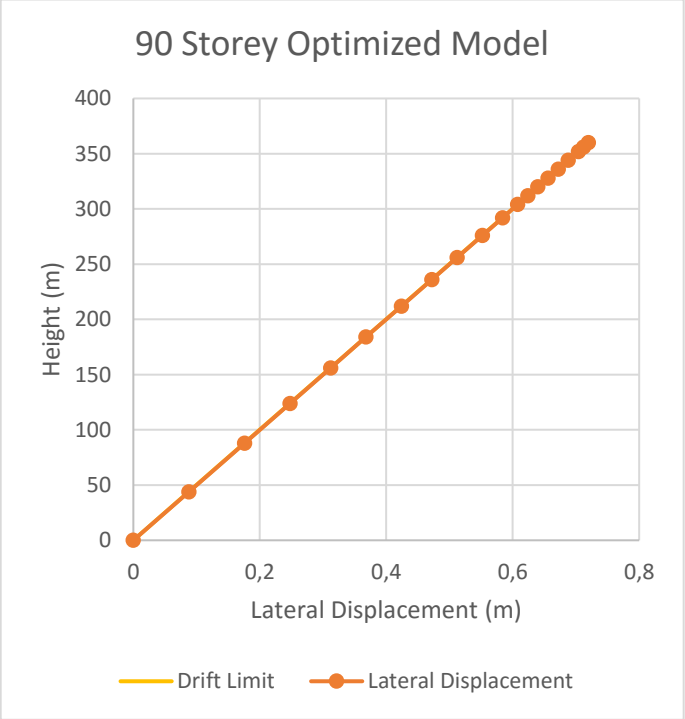


Figure 3.53. *Module lateral displacements of 90-story optimized model Type 1*

As a result of the optimization process, the optimum diagonal angles for each module are given in Table 3.2-3.11. According to these results, it can be seen that, while the aspect ratio increases, the base module diagonal angle also increases because of the gravity loads becoming more dominant than lateral loads. In addition, a gradual decrement of diagonal angles from base to top module is observed which is the result of lateral forces becoming more dominant than gravity forces while the module height from the base of the structure increases.

Table 3.2. *Optimum module diagonal angles of 30-story optimized model Type 0*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	16.00	16.00	69.44
2	16.00	32.00	69.44
3	16.00	48.00	69.44
4	16.00	64.00	69.44
5	12.00	76.00	63.43
6	16.00	92.00	69.44
7	12.00	104.00	63.43
8	12.00	116.00	63.43
9	4.00	120.00	33.69

Table 3.3. *Optimum module diagonal angles of 48-story optimized model Type 0*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	24.00	24.00	75.96
2	20.00	44.00	73.30
3	20.00	64.00	73.30
4	16.00	80.00	69.44
5	16.00	96.00	69.44
6	12.00	108.00	63.43
7	12.00	120.00	63.43
8	12.00	132.00	63.43
9	16.00	148.00	69.44
10	8.00	156.00	53.13
11	12.00	168.00	63.43
12	8.00	176.00	53.13
13	8.00	184.00	53.13
14	8.00	192.00	53.13

Table 3.4. *Optimum module diagonal angles of 60-story optimized model Type 0*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	28.00	28.00	77.91
2	24.00	52.00	75.96
3	24.00	76.00	75.96
4	20.00	96.00	73.30
5	20.00	116.00	73.30
6	16.00	132.00	69.44
7	16.00	148.00	69.44
8	16.00	164.00	69.44
9	16.00	180.00	69.44
10	12.00	192.00	63.43
11	12.00	204.00	63.43
12	8.00	212.00	53.13
13	8.00	220.00	53.13
14	8.00	228.00	53.13
15	8.00	236.00	53.13
16	4.00	240.00	33.69

Table 3.5. *Optimum module diagonal angles of 72-story optimized model Type 0*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	24.00	24.00	75.96
2	32.00	56.00	79.38
3	32.00	88.00	79.38
4	28.00	116.00	77.91
5	28.00	144.00	77.91
6	20.00	164.00	73.30
7	16.00	180.00	69.44
8	16.00	196.00	69.44
9	16.00	212.00	69.44
10	12.00	224.00	63.43
11	12.00	236.00	63.43
12	12.00	248.00	63.43
13	8.00	256.00	53.13
14	8.00	264.00	53.13
15	8.00	272.00	53.13
16	8.00	280.00	53.13
17	4.00	284.00	33.69

Table 3.5 continues as,

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
18	4.00	288.00	33.69

Table 3.6. *Optimum module diagonal angles of 90-story optimized model Type 0*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	48.00	48.00	82.87
2	36.00	84.00	80.54
3	36.00	120.00	80.54
4	32.00	152.00	79.38
5	28.00	180.00	77.91
6	24.00	204.00	75.96
7	24.00	228.00	75.96
8	20.00	248.00	73.30
9	16.00	264.00	69.44
10	16.00	280.00	69.44
11	16.00	296.00	69.44
12	12.00	308.00	63.43
13	8.00	316.00	53.13
14	8.00	324.00	53.13
15	12.00	336.00	63.43
16	8.00	344.00	53.13
17	8.00	352.00	53.13
18	4.00	356.00	33.69
19	4.00	360.00	33.69

Table 3.7. *Optimum module diagonal angles of 30-story optimized model Type 1*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	16.00	16.00	69.44
2	16.00	32.00	69.44
3	20.00	52.00	73.30
4	16.00	68.00	69.44
5	12.00	80.00	63.43
6	16.00	96.00	69.44
7	12.00	108.00	63.43
8	12.00	120.00	63.43

Table 3.8. *Optimum module diagonal angles of 48-story optimized model Type 1*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	12.00	12.00	63.43
2	20.00	32.00	73.30
3	20.00	52.00	73.30
4	24.00	76.00	75.96
5	12.00	88.00	63.43
6	12.00	100.00	63.43
7	16.00	116.00	69.44
8	16.00	132.00	69.44
9	8.00	140.00	53.13
10	12.00	152.00	63.43
11	8.00	160.00	53.13
12	8.00	168.00	53.13
13	8.00	176.00	53.13
14	8.00	184.00	53.13
15	8.00	192.00	53.13

Table 3.9. *Optimum module diagonal angles of 60-story optimized model Type 1*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	28.00	28.00	77.91
2	32.00	60.00	79.38
3	28.00	88.00	77.91
4	24.00	112.00	75.96
5	16.00	128.00	69.44
6	16.00	144.00	69.44
7	12.00	156.00	63.43
8	12.00	168.00	63.43
9	12.00	180.00	63.43
10	12.00	192.00	63.43
11	12.00	204.00	63.43
12	8.00	212.00	53.13
13	8.00	220.00	53.13
14	8.00	228.00	53.13
15	4.00	232.00	33.69
16	4.00	236.00	33.69
17	4.00	240.00	33.69

Table 3.10. *Optimum module diagonal angles of 72-story optimized model Type 1*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	28.00	28.00	77.91
2	36.00	64.00	80.54
3	32.00	96.00	79.38
4	32.00	128.00	79.38
5	24.00	152.00	75.96
6	24.00	176.00	75.96
7	20.00	196.00	73.30
8	16.00	212.00	69.44
9	16.00	228.00	69.44
10	12.00	240.00	63.43
11	8.00	248.00	53.13
12	8.00	256.00	53.13
13	8.00	264.00	53.13
14	8.00	272.00	53.13
15	8.00	280.00	53.13
16	4.00	284.00	33.69
17	4.00	288.00	33.69

Table 3.11. *Optimum module diagonal angles of 90-story optimized model Type 1*

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
1	44.00	44.00	82.23
2	44.00	88.00	82.23
3	36.00	124.00	80.54
4	32.00	156.00	79.38
5	28.00	184.00	77.91
6	28.00	212.00	77.91
7	24.00	236.00	75.96
8	20.00	256.00	73.30
9	20.00	276.00	73.30
10	16.00	292.00	69.44
11	12.00	304.00	63.43
12	8.00	312.00	53.13
13	8.00	320.00	53.13
14	8.00	328.00	53.13
15	8.00	336.00	53.13
16	8.00	344.00	53.13

Table 3.11 continues as,

Module	Module Height (m)	Cumulative Height (m)	Diagonal Angle
17	8.00	352.00	53.13
18	4.00	356.00	33.69
19	4.00	360.00	33.69

Finally, the resultant module diagonal pipe sections that are evaluated according to Section 2.3.2, are shown in Table 3.12 – 3.16. According to these results, the section sizes differ according to Type 0 and Type 1 models, since the load distribution changes, and the module diagonal length is also a defining parameter which can differ between those types of geometries.

Table 3.12. *Resultant diagonal pipe sections of 30-story optimized models*

30-story Optimized Model		
Module No.	**Diagonal Section Name**	
	Type 0	**Type 1**
1	PIPE 600x50	PIPE 610x50
2	PIPE 600x40	PIPE 600x40
3	PIPE 590x30	PIPE 710x30
4	PIPE 590x30	PIPE 580x30
5	PIPE 460x30	PIPE 460x30
6	PIPE 580x20	PIPE 570x20
7	PIPE 450x20	PIPE 450x10
8	PIPE 450x10	PIPE 450x10
9	PIPE 240x10	-

Table 3.13. *Resultant diagonal pipe sections of 48-story optimized models*

48-story Optimized Model		
Module No.	**Diagonal Section Name**	
	Type 0	**Type 1**
1	PIPE 860x60	PIPE 530x110
2	PIPE 740x60	PIPE 740x60
3	PIPE 730x50	PIPE 740x60
4	PIPE 610x50	PIPE 850x40
5	PIPE 600x40	PIPE 490x60
6	PIPE 480x50	PIPE 490x50

Table 3.13 continues as,

Module No.	Diagonal Section Name	
	Type 0	Type 1
7	PIPE 480x40	PIPE 590x40
8	PIPE 470x40	PIPE 590x30
9	PIPE 580x30	PIPE 370x40
10	PIPE 360x30	PIPE 460x30
11	PIPE 460x20	PIPE 350x30
12	PIPE 340x20	PIPE 350x20
13	PIPE 340x10	PIPE 340x20
14	PIPE 340x10	PIPE 340x10
15	-	PIPE 340x10

Table 3.14. *Resultant diagonal pipe sections of 60-story optimized models*

60-story Optimized Model		
Module No.	Diagonal Section Name	
	Type 0	Type 1
1	PIPE 1000x70	PIPE 1000x70
2	PIPE 870x70	PIPE 1120x60
3	PIPE 860x60	PIPE 1000x70
4	PIPE 740x60	PIPE 860x50
5	PIPE 730x50	PIPE 610x50
6	PIPE 610x60	PIPE 600x50
7	PIPE 610x60	PIPE 490x50
8	PIPE 620x70	PIPE 490x50
9	PIPE 610x60	PIPE 500x70
10	PIPE 490x50	PIPE 490x60
11	PIPE 480x40	PIPE 500x60
12	PIPE 360x40	PIPE 370x40
13	PIPE 350x30	PIPE 370x40
14	PIPE 350x30	PIPE 360x30
15	PIPE 340x20	PIPE 260x30
16	PIPE 250x10	PIPE 250x20
17	-	PIPE 250x10

Table 3.15. *Resultant diagonal pipe sections of 72-story optimized models*

72-story Optimized Model		
Module No.	**Diagonal Section Name**	
	Type 0	**Type 1**
1	PIPE 900x100	PIPE 1020x90
2	PIPE 1140x80	PIPE 1270x80
3	PIPE 1140x80	PIPE 1150x100
4	PIPE 1020x90	PIPE 1150x90
5	PIPE 1010x90	PIPE 890x90
6	PIPE 760x80	PIPE 880x80
7	PIPE 620x70	PIPE 760x80
8	PIPE 640x90	PIPE 620x60
9	PIPE 630x80	PIPE 620x70
10	PIPE 510x80	PIPE 480x50
11	PIPE 500x70	PIPE 370x50
12	PIPE 510x80	PIPE 370x40
13	PIPE 370x50	PIPE 370x50
14	PIPE 380x60	PIPE 360x40
15	PIPE 360x40	PIPE 360x30
16	PIPE 360x40	PIPE 250x20
17	PIPE 250x20	PIPE 250x10
18	PIPE 250x10	-

Table 3.16. *Resultant diagonal pipe sections of 90-story optimized models*

90-story Optimized Model		
Module No.	**Diagonal Section Name**	
	Type 0	**Type 1**
1	PIPE 1680x110	PIPE 1560x120
2	PIPE 1300x120	PIPE 1560x120
3	PIPE 1300x120	PIPE 1320x130
4	PIPE 1190x140	PIPE 1170x120
5	PIPE 1040x120	PIPE 1060x140
6	PIPE 930x140	PIPE 1060x130
7	PIPE 920x130	PIPE 940x140
8	PIPE 810x140	PIPE 790x110
9	PIPE 650x110	PIPE 800x140
10	PIPE 680x140	PIPE 650x100
11	PIPE 660x110	PIPE 520x100

Table 3.16 continues as,

Module No.	Diagonal Section Name	
	Type 0	**Type 1**
12	PIPE 530x110	PIPE 390x80
13	PIPE 400x80	PIPE 410x100
14	PIPE 410x100	PIPE 400x80
15	PIPE 520x90	PIPE 410x90
16	PIPE 390x70	PIPE 390x70
17	PIPE 370x40	PIPE 380x60
18	PIPE 260x30	PIPE 260x30
19	PIPE 250x20	PIPE 250x20

4. CONCLUSION & REMARKS

In this study, five different building models, which have different aspect ratios, are analyzed by strength and stiffness based designs based on previous studies in literature through a designed Matlab code, which includes both SAP2000 OAPI codes to transfer core model information to Matlab and Genetic Algorithm codes to iteratively optimize the geometry definition process, in order to automate the preliminary topology design of high-rise steel structures optimally in terms of material economy. The building core models are designed manually on SAP2000 by only defining load parameters, material type and load combinations. In addition, a different typology from previous studies for module diagonal geometry types are proposed.

According to weight results of the uniform models with different diagonal angles, it is safe to say that optimum angle value changes in terms of building height. Since the plan dimensions are same for every model, it can be concluded that the optimum angle differs by means of aspect ratio. Also, it can be observed that diagonal geometry types (Type 0 and Type 1) do not affect this result.

The results also show that, the optimized models have more efficiency in material usage. However, the efficiency ratio changes according to aspect ratio, since the stiffness design becomes more dominant with higher aspect ratio values. Up to 31% efficiency on material usage was obtained on optimized 90-story model, compared to optimum uniform 90-story model. In addition, Type 0 and Type 1 weight results differ on both uniform and optimized models, which indicates Type 1 models are slightly inefficient in terms of material usage, as per Type 0.

About the resultant module diagonal cross-sectional areas based on strength and stiffness analysis, for 30-story and 48-story models strength areas are higher, however as the aspect ratio rises, stiffness areas dominate the design of member sections. Also, it can be concluded that optimized models tend to have similar strength and stiffness area values which means the areas converge to minimum of both results as much as design limitations allow, whereas optimum uniform models can have very distinctive strength and stiffness area values.

Furthermore, by observing resultant lateral displacement graphs, it can be concluded that, as aspect ratio rises the lateral displacements converge to the limit because, stiffness areas dominate the design.

Moreover, based on the resultant module diagonal angles of optimized models, a gradual decrement from bottom to top modules stands out; however, the aspect ratio increment gives rise to lower module results. Also, there are no significant differences between Type 0 and Type 1 models in terms of resultant diagonal angles, therefore it should be mentioned that diagonal geometry type does not affect the topology design.

In addition, it is observed that the resultant pipe sections of modules differ according to diagonal geometry type of the models, since the angles and module diagonal lengths change in terms of the optimum results of these types.

In conclusion, it is advised by the authors that an optimization process is needed for structures which have aspect ratio more than 6 for the sake of economic material usage. Additionally, Type 0 models proved to have more optimum results, thus this type should be used while designing high-rise steel diagrid shells for the same purpose. The resultant optimized models have more modules, hence more diagrid connections, than uniform models. Since the design of these connections are very detailed and have high costs, the topic of whether optimized or uniform models are more economic should be studied.

REFERENCES

[1] Montuori G.M., Mele E., Brandonisio G., and De Luca A. (2014). Geometrical patterns for diagrid buildings: Exploring alternative design strategies from the structural point of view. Engineering Structures 71, 112-127.

[2] Schulitz H.C., Sobek W., and Habermann K.J. (2001). Steel construction manual. Munich, Germany: Birkhäuser.

[3] Moon K.S., Connor J.J., and Fernandez J.E. (2007). Diagrid structural systems for tall buildings: Characteristics and methodology for preliminary design. Struct. Design Tall Spec. Build. 16, 205–230.

[4] Montuori G.M., Mele E., Brandonisio G., and De Luca A. (2014). Design criteria for diagrid tall buildings: Stiffness versus strength. Struct. Design Tall Spec. Build. 23, 1294–1314.

[5] Zhang C., Zhao F., and Liu Y. (2012). Diagrid tube structures composed of straight diagonals with gradually varying angles. Struct. Design Tall Spec. Build. 21, 283–295.

[6] Mele E. (2017). Diagrid: The renaissance of steel structures for tall buildings. In: *EUROSTEEL 2017*, Copenhagen, Denmark. Sep. 13-15, 2017.

[7] Angelucci G. and Mollaioli F. (2017). Diagrid structural systems for tall buildings: Changing pattern configuration through topological assesments. Struct. Design Tall Spec. Build. 26, 1396.

[8] Goldberg D. (1989). Genetic algorithms in research, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 1-25.

[9] ASCE. (2010). Minimum design loads for buildings and other structures. ASCE/SEI Standard 7-10.

[10] Jayachandran P. (2003). International Conference on Tall Buildings. May, 2003, 1–20.

[11] ANSI/AISC 360-10. (2010). Specification for structural steel buildings. June, 22, 2010.

[12] Smith R.J. (2011). Deflection limits in tall buildings—Are they useful? In: *Structures Congress 2011*, April, 2011.

APPENDIX-1 An Example Main Code of a Uniform Model

```
%% Clean-Up the Workspace & Command Window
clear;
close all;
clc;

%% USER INPUTS
dtype = 0;                  % '0' is for only middle RP's, '1' is for RP's with corners
ndsp = 3;                   % Number of Diagonal Starting Points at Base on 1 Façade
alpha = 1/500;              % Inter-Module Drift Ratio Limit
wp = 2.86;                  % Wind Pressure (kN/m2)
DL = 7;                     % Dead Load (kN/m2)
LL = 2.4;                   % Live Load (kN/m2)

MatName = 'S275';           % Material Name (The input material must
                            % have already been defined in Core Model)

% Full Path of Core Model
CoreModelPath = 'C:\Users\Kullanıcı\OneDrive\Ms Thesis\Shape\Sap\Core90\core90.sdb';

% Shape Model Template Parent Folder Path
ShapeModelPath = 'C:\Users\Kullanıcı\OneDrive\Ms Thesis\Shape\Sap\Shape';

%% SAP2000 INPUTS

%% Program Path
ProgramPath = 'C:\Program Files\Computers and Structures\SAP2000 19\sap2000.exe';
```

```
%% Full Path to API DLL
APIDLLPath = 'C:\Program Files\Computers and Structures\SAP2000 19\SAP2000v19.dll';

%% Create OAPI Helper Object
a = NET.addAssembly(APIDLLPath);
helper = SAP2000v19.Helper;
helper = NET.explicitCast(helper,'SAP2000v19.cHelper');

%% Create SAP2000 Object
SapObject = helper.CreateObject(ProgramPath);
SapObject = NET.explicitCast(SapObject,'SAP2000v19.cOAPI');

%% Start SAP2000 Application
SapObject.ApplicationStart;

%% Create SapModel Object
Sap = NET.explicitCast(SapObject.SapModel,'SAP2000v19.cSapModel');

%% Shortcuts
File = NET.explicitCast(Sap.File,'SAP2000v19.cFile');
FrameObj = NET.explicitCast(Sap.FrameObj,'SAP2000v19.cFrameObj');
PointObj = NET.explicitCast(Sap.PointObj,'SAP2000v19.cPointObj');
View = NET.explicitCast(Sap.View,'SAP2000v19.cView');
PropMaterial = NET.explicitCast(Sap.PropMaterial,'SAP2000v19.cPropMaterial');
RespCombo = NET.explicitCast(Sap.RespCombo,'SAP2000v19.cCombo');
LoadPatterns = NET.explicitCast(Sap.LoadPatterns,'SAP2000v19.cLoadPatterns');
```

```
%% Open Core Model & Get Core Data
[pc,sigmay,Ed,uw,LC] = coreData(File,CoreModelPath,Sap,PointObj,...
                                PropMaterial,MatName,RespCombo,LoadPatterns);

%% Define Shape Model Geometry Properties
[webl,flangel,el,fn,bw,RP,ndw,ndf]= shapeGeo(pc,ndsp,dtype);

%% GA INPUTS

%% Problem Definition
nVar = fn;                          % Number of Decision Variables

%% Initialization
pop.Position = [];
pop.Weight = 0;                     % Total Weight of Structure
pop.EDCR = 0;                       % Effective DCR
pop.Diag = [];                      % All Info About Modules

% Initialize Position
temp = [0 0 0 0 0 0 0 0 0 1];       % Equivalent to 82 degrees of diagonal angle
pop.Position = repmat(temp,1,(nVar/numel(temp)));

% Find the Floor Number Which The Diagonals Change Direction
ind = find(pop.Position == 1);              % Floor Numbers of Intersection Points
h = [0 ind.*el];                            % Elevations of the Points
nmod = numel(h)-1;                          % Number of Modules
hm = diff(h).';                             % Height of Each Module
```

```
% Define Diagonal Names & Coordinates for Analysis Stage
[Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,ndw,ndf,hm,nmod,webl,flangel);

% Strength & Stiffness Analysis
[Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,hm,webl,...
                                  flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR);

% Store Total Weight of Structure
pop.Weight = WEIGHT;

% Store EDCR of Model
pop.EDCR = EDCR;                  % Effective Demand to Capacity Ratio

% Store Diag Data
RS = Diag;                        % Results Structure
fields = {'Fnamex1','Fnamex2','Fnamey1','Fnamey2','FpIx1','FpJx1','FpIx2',...
                                  'FpJx2','FpIy1','FpJy1','FpIy2','FpJy2','Fnum'};
RS = rmfield(RS,fields);          % Remove Unnecessary Fields
pop.Diag = RS;

%% Draw the SAP2000 Model of Best Result & Save in a New Folder
% Find the Floor Number Which The Diagonals Change Direction
ind = find(pop.Position == 1);          % Floor Numbers of Intersection Points
h = [0 ind.*el];                        % Elevations of the Points
nmod = numel(h)-1;                      % Number of Modules

% Best Model Folder Name
folname = 'Uniform SAP Model';
```

```
% Draw Diagonals
[~,mdest] = BestDraw(ndsp,dtype,bw,pc,h,RP,ShapeModelPath,...
                                    PointObj,folname,File,FrameObj,View);

% Save Best Model
File.Save(mdest);
clear mdest folname;

%% Store Needed Best Solution Variables to Excel Template
OUT = pop.Diag;
nrow = numel(OUT)+1;
range1 = 'A2:';
range2 = ['T' num2str(nrow)];
range = [range1 range2];
OUT = struct2table(OUT);
writetable(OUT,'Results.xlsx','WriteVariableNames',false,'Range',range);

%% Close Sap2000
SapObject.ApplicationExit(false());

%% Delete .NET Commands
clear a helper File FrameObj PointObj View PropMaterial RespCombo...
      LoadPatterns Sap SapObject;

%% Store Solution Data in Mat File
save('Shape-Data.mat');

%% Exit Matlab
% exit;
```

APPENDIX-2 An Example Main Code of an Optimized Model

```
%% Clean-Up the Workspace & Command Window
clear;
close all;
clc;

%% USER INPUTS
dtype = 0;              % '0' is for only middle RP's, '1' is for RP's with corners
ndsp = 3;               % Number of Diagonal Starting Points at Base on 1 Plane

alpha = 1/500;          % Inter-Module Drift Ratio Limit

ItNumber = 2000;        % Number of Iterations
PopNumber = 50;         % Number of Populations

wp = 2.86;              % Wind Pressure (kN/m2)
DL = 7;                 % Dead Load (kN/m2)
LL = 2.4;               % Live Load (kN/m2)

MatName = 'S275';       % Material Name (The input material must
                        % have already been defined in Core Model)

% Full Path of Core Model
CoreModelPath = 'C:\Users\User\Desktop\Talha\Shape\Sap\Core90\core90.sdb';

% Shape Model Template Parent Folder Path
ShapeModelPath = 'C:\Users\User\Desktop\Talha\Shape\Sap\Shape';
```

```
%% Start Timer
tstart = tic;

%% SAP2000 INPUTS

%% Program Path
ProgramPath = 'C:\Program Files\Computers and Structures\SAP2000 19\sap2000.exe';

%% Full Path to API DLL
APIDLLPath = 'C:\Program Files\Computers and Structures\SAP2000
    19\SAP2000v19.dll';

%% Create OAPI Helper Object
NET.addAssembly(APIDLLPath);
helper = SAP2000v19.Helper;
helper = NET.explicitCast(helper,'SAP2000v19.cHelper');

%% Create SAP2000 Object
SapObject = helper.CreateObject(ProgramPath);
SapObject = NET.explicitCast(SapObject,'SAP2000v19.cOAPI');

%% Start SAP2000 Application
SapObject.ApplicationStart;

%% Create SapModel Object
Sap = NET.explicitCast(SapObject.SapModel,'SAP2000v19.cSapModel');

%% Shortcuts
File = NET.explicitCast(Sap.File,'SAP2000v19.cFile');
```

```
FrameObj = NET.explicitCast(Sap.FrameObj,'SAP2000v19.cFrameObj');
PointObj = NET.explicitCast(Sap.PointObj,'SAP2000v19.cPointObj');
View = NET.explicitCast(Sap.View,'SAP2000v19.cView');
PropMaterial = NET.explicitCast(Sap.PropMaterial,'SAP2000v19.cPropMaterial');
RespCombo = NET.explicitCast(Sap.RespCombo,'SAP2000v19.cCombo');
LoadPatterns = NET.explicitCast(Sap.LoadPatterns,'SAP2000v19.cLoadPatterns');

%% Hide Application, Hide <=> Visible
SapObject.Hide;

%% Open Core Model & Get Core Data
[pc,sigmay,Ed,uw,LC] = coreData(File,CoreModelPath,Sap,PointObj,...
                                PropMaterial,MatName,RespCombo,LoadPatterns);

%% Define Shape Model Geometry Properties
[webl,flangel,el,fn,bw,RP,ndw,ndf]= shapeGeo(pc,ndsp,dtype);

%% GA INPUTS

%% Problem Definition
nVar = fn;                  % Number of Decision Variables

%% GA Parameters
MaxIt = ItNumber;           % Maximum Number of Iterations
nPop = PopNumber;           % Population Size

cp = 0.8;                   % Crossover Percentage
nc = 2*round(cp*nPop/2);    % Number of Offsprings (also Parnets)
```

```
pm = 0.3;                   % Mutation Percentage
nm = round(pm*nPop);        % Number of Mutants
mu = 0.02;                  % Mutation Rate

TournamentSize = 3;         % Tournament Size

%% Initialization
empty_individual.Position = [];
empty_individual.Weight = 0;            % Total Weight of Structure
empty_individual.EDCR = 0;              % Effective DCR
empty_individual.Diag = [];             % All Info About Modules

pop = repmat(empty_individual,nPop,1);

for i = 1:nPop
    % Initialize Position
    pop(i).Position = randi([0 1],1,nVar-1);
    pop(i).Position(nVar) = 1;

    % Find the Floor Number Which The Diagonals Change Direction
    ind = find(pop(i).Position == 1);       % Floor Numbers of Intersection Points
    h = [0 ind.*el];                        % Elevations of the Points
    nmod = numel(h)-1;                      % Number of Modules
    hm = diff(h).';                         % Height of Each Module

    % Define Diagonal Names & Coordinates for Analysis Stage
    [Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,ndw,ndf,hm,nmod,webl,flangel);
```

```
    % Strength & Stiffness Analysis
    [Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,hm,webl,...
                                      flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR);

    % Store Total Weight of Structure
    pop(i).Weight = WEIGHT;

    % Store EDCR of Model
    pop(i).EDCR = EDCR;

    % Store Diag Data
    RS = Diag;                        % Results Structure
    fields = {'Fnamex1','Fnamex2','Fnamey1','Fnamey2','FpIx1','FpJx1','FpIx2',...
                          'FpJx2','FpIy1','FpJy1','FpIy2','FpJy2','Fnum'};
    RS = rmfield(RS,fields);          % Remove Unnecessary Fields
    pop(i).Diag = RS;

end

% Sort Population
Weights = [pop.Weight];
[~, SortOrder] = sort(Weights);
pop = pop(SortOrder);

% Array to Hold Best Length Values
BestWeight = zeros(MaxIt,1);

%% Main Loop
WeightsAll = zeros(MaxIt,nPop);
```

```
PosAll = struct('');

for it = 1:MaxIt

    % Crossover
    popc = repmat(empty_individual,nc/2,2);

    for k = 1:nc/2

        i1 = TournamentSelection(pop,TournamentSize);
        i2 = TournamentSelection(pop,TournamentSize);

        % Select Parents
        p1 = pop(i1);
        p2 = pop(i2);

        % Perform Crossover
        [popc(k,1).Position, popc(k,2).Position] = Crossover(p1.Position,p2.Position);
        popc(k,1).Position(nVar) = 1;
        popc(k,2).Position(nVar) = 1;

        %% popc(k,1)
        % Find the Floor Number Which The Diagonals Change Direction
        ind = find(popc(k,1).Position == 1);     % Floor Numbers of Intersection Points
        h = [0 ind.*el];                         % Elevations of the Points
        nmod = numel(h)-1;                       % Number of Modules
        hm = diff(h).';                          % Height of Each Module
```

```
% Define Diagonal Names & Coordinates for Analysis Stage
[Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,ndw,ndf,hm,nmod,webl,flangel);

% Strength & Stiffness Analysis
[Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,hm,...
                                  webl,flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR);

% Store Total Weight of Structure
popc(k,1).Weight = WEIGHT;

% Store EDCR of Model
popc(k,1).EDCR = EDCR;

% Store Diag Data
RS = Diag;                          % Results Structure
fields = {'Fnamex1','Fnamex2','Fnamey1','Fnamey2','FpIx1','FpJx1','FpIx2',...
                               'FpJx2','FpIy1','FpJy1','FpIy2','FpJy2','Fnum'};
RS = rmfield(RS,fields);            % Remove Unnecessary Fields
popc(k,1).Diag = RS;

%% popc(k,2)
% Find the Floor Number Which The Diagonals Change Direction
ind = find(popc(k,2).Position == 1);    % Floor Numbers of Intersection Points
h = [0 ind.*el];                        % Elevations of the Points
nmod = numel(h)-1;                      % Number of Modules
hm = diff(h).';                         % Height of Each Module

% Define Diagonal Names & Coordinates for Analysis Stage
[Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,ndw,ndf,hm,nmod,webl,flangel);
```

```
    % Strength & Stiffness Analysis
    [Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,hm,...
                                webl,flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR);

    % Store Total Weight of Structure
    popc(k,2).Weight = WEIGHT;

    % Store EDCR of Model
    popc(k,2).EDCR = EDCR;

    % Store Diag Data
    RS = Diag;                          % Results Structure
    fields = {'Fnamex1','Fnamex2','Fnamey1','Fnamey2','FpIx1','FpJx1','FpIx2',...
                'FpJx2','FpIy1','FpJy1','FpIy2','FpJy2','Fnum'};
    RS = rmfield(RS,fields);            % Remove Unnecessary Fields
    popc(k,2).Diag = RS;

end

popc = popc(:);

%% Mutation
popm = repmat(empty_individual,nm,1);

for k = 1:nm

    % Select Parent
    i = randi([1 nPop]);
    p = pop(i);
```

```
% Perform Mutation
popm(k).Position = Mutate(p.Position,mu);
popm(k).Position(nVar) = 1;

% Find the Floor Number Which The Diagonals Change Direction

ind = find(popm(k).Position == 1);      % Floor Numbers of Intersection Points
h = [0 ind.*el];                        % Elevations of the Points
nmod = numel(h)-1;                      % Number of Modules
hm = diff(h).';                         % Height of Each Module

% Define Diagonal Names & Coordinates for Analysis Stage
[Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,ndw,ndf,hm,nmod,webl,flangel);

% Strength & Stiffness Analysis
[Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,hm,...
                                  webl,flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR);

% Store Total Weight of Structure
popm(k).Weight = WEIGHT;

% Store EDCR of Model
popm(k).EDCR = EDCR;

% Store Diag Data
RS = Diag;                              % Results Structure
fields = {'Fnamex1','Fnamex2','Fnamey1','Fnamey2','FpIx1','FpJx1','FpIx2',...
          'FpJx2','FpIy1','FpJy1','FpIy2','FpJy2','Fnum'};
RS = rmfield(RS,fields);          % Remove Unnecessary Fields
```

```
    popm(k).Diag = RS;

end

%% Create Merged Population
pop =  [pop
        popc
        popm]; %#ok<AGROW>

% Sort Population
Weights = [pop.Weight];
[~, SortOrder] = sort(Weights);
pop = pop(SortOrder);

% Truncation
pop = pop(1:nPop);

% Store All Weight Data
for i = 1:nPop
    WeightsAll(it,i) = pop(i).Weight;              % Store All Weights
    PosAll(it).Pos(i,:) = pop(i).Position;         % Store All Positions
end

% Store Best Solution Ever Found
BestSol = pop(1);

% Store Best Length Ever Found
BestWeight(it) = BestSol.Weight;
```

```
    % Show Iteration Information
    disp(['Iteration ' num2str(it) ': Best Weight = ' num2str(BestWeight(it))]);

end

%% Plot Results and Save Plot
figure;
plot(BestWeight,'LineWidth',2);
xlabel('Iteration');
ylabel('Weight (tons)');
grid on;

saveas(gcf,'Shape-Plot','png');
close;

%% Draw the SAP2000 Model of Best Result & Save in a New Folder
% Find the Floor Number Which The Diagonals Change Direction
ind = find(BestSol.Position == 1);              % Floor Numbers of Intersection Points
h = [0 ind.*el];                                % Elevations of the Points

% Best Model Folder Name
folname = 'Best Model SAP';

% Draw Diagonals
[~,mdest] = BestDraw(ndsp,dtype,bw,pc,h,RP,ShapeModelPath,...
                                PointObj,folname,File,FrameObj,View);
```

```
% Save Best Model
File.Save(mdest);

clear mdest folname;

%% Store Needed Best Solution Variables to Excel Template
OUT = BestSol.Diag;
nrow = numel(OUT)+1;
range1 = 'A2:';
range2 = ['T' num2str(nrow)];
range = [range1 range2];
OUT = struct2table(OUT);
writetable(OUT,'Results.xlsx','WriteVariableNames',false,'Range',range);

%% Stop Timer & Store Analysis Time
tstop = toc(tstart);
day = floor(tstop/86400);
hr = floor(tstop/3600) - (day*24);
min = floor(tstop/60) - (hr*60) - (day*1440);
sec = tstop - (hr*3600) - (min*60) - (day*86400);
Timer = [num2str(day) ' days, ' num2str(hr) ' hours, ' num2str(min) ' minutes, '
    num2str(sec) ' seconds.']; %#ok<NASGU>
clear day hr min sec tstart tstop;

%% Close Sap2000
SapObject.ApplicationExit(false());
```

```
%% Delete .NET Commands
clear a helper File FrameObj PointObj View PropMaterial RespCombo...
    LoadPatterns Sap SapObject;

%% Write Solution to Mat File
save('Shape-Data.mat','-v7.3');

%% Exit Matlab
exit;

%% END
```

APPENDIX-3 Function of coreData

```
function [pc,sigmay,Ed,uw,LC] = coreData(File,CoreModelPath,Sap,PointObj,...
                                    PropMaterial,MatName,RespCombo,LoadPatterns)

File.OpenFile(CoreModelPath);

% Unlock Model
Sap.SetModelIsLocked(false());

% Change Units to kN_m_C
Sap.SetPresentUnits(SAP2000v19.eUnits.kN_m_C);

% Get Point Names
np = 0;                 % Number of Points
pls = cellstr('');      % List of Points as Cell
[~,np,pls] = PointObj.GetNameList(np,pls);
np = double(np);
pls = cell(pls);

% Get Point Coordinates of Core Model
pc = zeros(np,3);
for i = 1:np
   x = 0;
   y = 0;
   z = 0;
   [~,x,y,z] = PointObj.GetCoordCartesian(char(pls(i)),x,y,z);
   % pc = x,y and z Coordinates of Points
   pc(i,1) = x;
   pc(i,2) = y;
```

```
    pc(i,3) = z;
end

% Get Yield Stress Value of Material
sigmay = 0; Fu = 0; eFy = 0; eFu = 0; SSType = 0; SSHysType = 0;
StrainAtHardening = 0; StrainAtMaxStress = 0; StrainAtRupture = 0;
FinalSlope = 0; MatProp = MatName;

[~,sigmay,~,~,~,~,~,~,~,~,~] = PropMaterial.GetOSteel_1(MatProp,sigmay,...
        Fu,eFy,eFu,SSType,SSHysType,StrainAtHardening,StrainAtMaxStress,…
        StrainAtRupture,FinalSlope);
sigmay = double(sigmay);

% Get Modulus of Elasticity ---> ONLY WORKS FOR ISOTROPIC MATERIALS!
me = 0;                 % Modulus of Elasticity
mu = 0;                 % Poisson's Ratio
ma = 0;                 % Thermal Coefficient
mg = 0;                 % Shear Modulus
[~,me,~,~,~] = PropMaterial.GetMPIsotropic(MatProp,me,mu,ma,mg);
Ed = double(me);

% Get Unit Weight of Material
uw = 0; mass = 0;
[~,uw,~] = PropMaterial.GetWeightAndMass(MatProp,uw,mass);
uw = double(uw);

% Get Number of Load Cases
nlcase = 0; lcasel = {''};
[~,nlcase,lcasel] = LoadPatterns.GetNameList(nlcase,lcasel);
```

```
nlcase = double(nlcase);
lcasel = cell(lcasel);

% Get Load Combination List
nlc = 0; lcl = {''};
[~,nlc,lcl] = RespCombo.GetNameList(nlc,lcl);
nlc = double(nlc);          % Number of Load Cases
lcl = cell(lcl);            % Load Case Name List

% Get Load Combination Scale Factors
nl = 0; CName = {''}; SF = [];
CType = NET.createArray('SAP2000v19.eCNameType',nlc);
empty_LC.Name = '';
empty_LC.CaseList = {''};
empty_LC.Factor = zeros(1,nlcase);

LC = repmat(empty_LC, [nlc 1]);

for ii = 1:nlc
   [~,nl,~,CName,SF] = RespCombo.GetCaseList(char(lcl(ii)), nl, CType, CName, SF);
   nl = double(nl);
   CName = cell(CName);
   SF = double(SF);
   LC(ii).Name = char(lcl(ii));
   LC(ii).CaseList = lcasel;
   idx = ismember(lcasel,CName);
   LC(ii).Factor(idx) = SF;
end
end
```

APPENDIX-4 Functiıon of shapeGeo

```
function [webl,flangel,el,fn,bw,RP,ndw,ndf]= shapeGeo(pc,ndsp,dtype)

webl = max(pc(:,1)) - min(pc(:,1));                % Web Length of the Structure
flangel = max(pc(:,2)) - min(pc(:,2));             % Flange Length of the Structure
bh = round((max(pc(:,3)) - min(pc(:,3))),4);       % Building Height
el = round(max(diff(sort(pc(:,3)))),4);            % Floor Elevation Height
fn = round(bh/el);                                 % Number of Floors

if dtype == 0
   nbx = 2*ndsp;                  % Number of Bays on X-Axis
   nby = nbx;                     % Number of Bays on Y-Axis
   bw = webl/nbx;                 % Bay Width for Middle Start Only Diagonals

   % Define Coordinates of Starting Points for Diagonals at Base (Reference Points for
Diagonals
   RP = [((1:2:nbx)*bw).' repmat(min(pc(:,2)),[ndsp 1]) repmat(min(pc(:,3)),[ndsp 1]);
         ((1:2:nbx)*bw).' repmat(max(pc(:,2)),[ndsp 1]) repmat(min(pc(:,3)),[ndsp 1]);
         repmat(min(pc(:,1)),[ndsp 1]) ((1:2:nby)*bw).' repmat(min(pc(:,3)),[ndsp 1]);
         repmat(max(pc(:,1)),[ndsp 1]) ((1:2:nby)*bw).' repmat(min(pc(:,3)),[ndsp 1])];

   ndw = ndsp*2;       % Number of Diagonal Elements on One Web in Each Module
   ndf = ndw;          % Number of Diagonal Elements on One Flange in Each Module
else
   nbx = 2*(ndsp-1);              % Number of Bays on X-Axis
   nby = nbx;                     % Number of Bays on Y-Axis
   bw = webl/nbx;                 % Bay Width for Diagonals Starting at Corners
```

```
    % Define Coordinates of Starting Points for Diagonals at Base (Reference Points for
Diagonals
    RP = [((0:2:nbx)*bw).' repmat(min(pc(:,2)),[ndsp 1]) repmat(min(pc(:,3)),[ndsp 1]);
          ((0:2:nbx)*bw).' repmat(max(pc(:,2)),[ndsp 1]) repmat(min(pc(:,3)),[ndsp 1]);
          repmat(min(pc(:,1)),[ndsp 1]) ((0:2:nby)*bw).' repmat(min(pc(:,3)),[ndsp 1]);
          repmat(max(pc(:,1)),[ndsp 1]) ((0:2:nby)*bw).' repmat(min(pc(:,3)),[ndsp 1])];

    ndw = (ndsp-1)*2;  % Number of Diagonal Elements on One Web in Each Module
    ndf = ndw;         % Number of Diagonal Elements on One Flange in Each Module
end

end
```

APPENDIX-5 Functiıon of shapePrep

```
function [Diag,AR] = shapePrep(ndsp,dtype,bw,pc,h,RP,…
                                ndw,ndf,hm,nmod,webl,flangel)


% Define Module Data Template
empty_Diag.Name = cellstr('');          % Module Group Name
empty_Diag.Fnamex1 = cellstr('');       % Diagonal Name List on First X-Plane
empty_Diag.Fnamex2 = cellstr('');       % Diagonal Name List on Second X-Plane
empty_Diag.Fnamey1 = cellstr('');       % Diagonal Name List on First Y-Plane
empty_Diag.Fnamey2 = cellstr('');       % Diagonal Name List on Second Y-Plane
empty_Diag.FpIx1 = [];                  % Diagonal I-End Coordinate on First X-Plane
empty_Diag.FpJx1 = [];                  % Diagonal J-End Coordinate on First X-Plane
empty_Diag.FpIx2 = [];                  % Diagonal I-End Coordinate on Second X-Plane
empty_Diag.FpJx2 = [];                  % Diagonal J-End Coordinate on Second X-Plane
empty_Diag.FpIy1 = [];                  % Diagonal I-End Coordinate on First Y-Plane
empty_Diag.FpJy1 = [];                  % Diagonal J-End Coordinate on First Y-Plane
empty_Diag.FpIy2 = [];                  % Diagonal I-End Coordinate on Second Y-Plane
empty_Diag.FpJy2 = [];                  % Diagonal J-End Coordinate on Second Y-Plane
empty_Diag.Fnum = 0;                    % Number of Elements
empty_Diag.Flength = 0;                 % Length of Frame Elements in Group
empty_Diag.Angle = 0;                   % Diagonal Angle with Horizontal Axis
empty_Diag.DeadLoad = 0;                % Total Dead Load on Module
empty_Diag.LiveLoad = 0;                % Total Live Load on Module
empty_Diag.Shear = 0;                   % Total Shear on Module
empty_Diag.Moment = 0;       % Overturning Moment due to Wind Load on Module
empty_Diag.Tension = 0;                 % Module Max Diagonal Tension Force
empty_Diag.Compression = 0;          % Module Max Diagonal Compression Force
empty_Diag.Astr = 0;           % Module Diagonal Section Area by Strength Analysis
empty_Diag.Astiff = 0;         % Module Diagonal Section Area by Stiffness Analysis
```

```
empty_Diag.Afin = 0;                % Resulting Module Diagonal Section Area
empty_Diag.Sname = {''};            % Resulting Module Diagonal Section Name
empty_Diag.Drift = 0;               % Inter-Module Drift
empty_Diag.LatDisp = 0;             % Total Lateral Displacement at Module Level
empty_Diag.Dratio = 0;              % Module Drift Ratio
empty_Diag.Mheight = 0;             % Module Height
empty_Diag.Cheight = 0;             % Module Cumulative Height
empty_Diag.DCR = 0;                 % Module Demand to Capacity Ratio (Percentage)
empty_Diag.Weight = 0;              % Total Module Weight

%%% PS : First and Second X or Y-Planes Refer to Minimum and Maximum Global
%%%     X and Y Coordinate Values of Frame Element End Points in Core Model
%%%     Respectively.

Diag = repmat(empty_Diag,[nmod 1]);

% Cumulative Height of Modules
ch = cumsum(hm);

% Initial Module Data
for ii = 1:nmod
   Diag(ii).Name = ['DIAG' num2str(ii)];
   Diag(ii).Fnum = 2*ndw + 2*ndf;
   Diag(ii).Flength = sqrt(hm(ii)^2 + bw^2);
   Diag(ii).Angle = atand(hm(ii)/bw);
   Diag(ii).Mheight = hm(ii);
   Diag(ii).Cheight = ch(ii);
end
```

```
% Define Diagonal Names & I-End and J-End Coordinates of Diagonals

for ii = 2:numel(h)

   for jj = 1:size(RP,1)

      quo1 = floor((jj-1)/ndsp);
      quo2 = quo1*ndsp;

      % Define Diagonal Name, Start&End Point Coordinate Data & Store
      if dtype == 0
         if jj <= ndsp              % For X-Axis Diagonals
            if mod(ii,2) == 0
               bp1 = RP(jj,1);
               bp2 = bp1;
               ep1 = RP(jj,1) - bw;
               ep2 = RP(jj,1) + bw;
            else
               bp1 = RP(jj,1) - bw;
               bp2 = RP(jj,1) + bw;
               ep1 = RP(jj,1);
               ep2 = ep1;
            end

            % Diagonal Names
            dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1));
            dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)));
            dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + ndw);
            dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)) + ndw);
```

```
    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamex1(2*(jj-quo2)-1) = cellstr(dg1);
    Diag(ii-1).Fnamex1(2*(jj-quo2)) = cellstr(dg2);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-1) = cellstr(dg3);
    Diag(ii-1).Fnamex2(2*(jj-quo2)) = cellstr(dg4);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIx1(2*(jj-quo2)-1,:) = [bp1 min(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx1(2*(jj-quo2)-1,:) = [ep1 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx1(2*(jj-quo2),:) = [bp2 min(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx1(2*(jj-quo2),:) = [ep2 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2)-1,:) = [bp1 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2)-1,:) = [ep1 max(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2),:) = [bp2 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2),:) = [ep2 max(pc(:,2)) h(ii)];

elseif jj > 2*ndsp && jj <= 3*ndsp      % For Y-Axis Diagonals
    if mod(ii,2) == 0
        bp1 = RP(jj,2);
        bp2 = bp1;
        ep1 = RP(jj,2) - bw;
        ep2 = RP(jj,2) + bw;
    else
        bp1 = RP(jj,2) - bw;
        bp2 = RP(jj,2) + bw;
        ep1 = RP(jj,2);
        ep2 = ep1;
    end
```

```
        % Diagonal Names
        dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw);
        dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)) + 2*ndw);
        dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw + ndf);
        dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)) + 2*ndw + ndf);

        % Name List of Diagonals on First&Second X-Plane
        Diag(ii-1).Fnamey1(2*(jj-quo2)-1) = cellstr(dg1);
        Diag(ii-1).Fnamey1(2*(jj-quo2)) = cellstr(dg2);
        Diag(ii-1).Fnamey2(2*(jj-quo2)-1) = cellstr(dg3);
        Diag(ii-1).Fnamey2(2*(jj-quo2)) = cellstr(dg4);

        % Diagonal I&J Point Coordinates on First&Second X-Plane
        Diag(ii-1).FpIy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) bp1 h(ii-1)];
        Diag(ii-1).FpJy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) ep1 h(ii)];
        Diag(ii-1).FpIy1(2*(jj-quo2),:) = [min(pc(:,1)) bp2 h(ii-1)];
        Diag(ii-1).FpJy1(2*(jj-quo2),:) = [min(pc(:,1)) ep2 h(ii)];
        Diag(ii-1).FpIy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) bp1 h(ii-1)];
        Diag(ii-1).FpJy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) ep1 h(ii)];
        Diag(ii-1).FpIy2(2*(jj-quo2),:) = [max(pc(:,1)) bp2 h(ii-1)];
        Diag(ii-1).FpJy2(2*(jj-quo2),:) = [max(pc(:,1)) ep2 h(ii)];

    end
else                % For dtype = 1
    if jj <= ndsp           % For X-Axis Diagonals
        if mod(ii,2) == 0

            % First Corner Diagonals
            if mod(jj,ndsp) == 1
```

```
    bp1 = RP(jj,1);
    ep1 = RP(jj,1) + bw;

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1));
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + ndw);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamex1(2*(jj-quo2)-1) = cellstr(dg1);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-1) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIx1(2*(jj-quo2)-1,:) = [bp1 min(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx1(2*(jj-quo2)-1,:) = [ep1 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2)-1,:) = [bp1 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2)-1,:) = [ep1 max(pc(:,2)) h(ii)];

% End Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,1);
    ep1 = RP(jj,1) - bw;

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2));
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + ndw);

    % Name List of Diagonals on First&Second X-Plane
```

```
    Diag(ii-1).Fnamex1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-2) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIx1(2*(jj-quo2)-2,:) = [bp1 min(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx1(2*(jj-quo2)-2,:) = [ep1 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2)-2,:) = [bp1 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2)-2,:) = [ep1 max(pc(:,2)) h(ii)];

% The Diagonals Between Corners
else

    bp1 = RP(jj,1);
    bp2 = bp1;
    ep1 = RP(jj,1) - bw;
    ep2 = RP(jj,1) + bw;

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2));
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1));
    dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + ndw);
    dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + ndw);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamex1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamex1(2*(jj-quo2)-1) = cellstr(dg2);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-2) = cellstr(dg3);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-1) = cellstr(dg4);
```

```
        % Diagonal I&J Point Coordinates on First&Second X-Plane
        Diag(ii-1).FpIx1(2*(jj-quo2)-2,:) = [bp1 min(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx1(2*(jj-quo2)-2,:) = [ep1 min(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx1(2*(jj-quo2)-1,:) = [bp2 min(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx1(2*(jj-quo2)-1,:) = [ep2 min(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx2(2*(jj-quo2)-2,:) = [bp1 max(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx2(2*(jj-quo2)-2,:) = [ep1 max(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx2(2*(jj-quo2)-1,:) = [bp2 max(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx2(2*(jj-quo2)-1,:) = [ep2 max(pc(:,2)) h(ii)];

    end
else
    % First Corner Diagonals
    if mod(jj,ndsp) == 1

        % For Diagonals
        bp1 = RP(jj,1) + bw;
        ep1 = RP(jj,1);

        % Diagonal Names
        dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1));
        dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + ndw);

        % Name List of Diagonals on First&Second X-Plane
        Diag(ii-1).Fnamex1(2*(jj-quo2)-1) = cellstr(dg1);
        Diag(ii-1).Fnamex2(2*(jj-quo2)-1) = cellstr(dg2);

        % Diagonal I&J Point Coordinates on First&Second X-Plane
        Diag(ii-1).FpIx1(2*(jj-quo2)-1,:) = [bp1 min(pc(:,2)) h(ii-1)];
```

```
    Diag(ii-1).FpJx1(2*(jj-quo2)-1,:) = [ep1 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2)-1,:) = [bp1 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2)-1,:) = [ep1 max(pc(:,2)) h(ii)];

% End Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,1) - bw;
    ep1 = RP(jj,1);

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2));
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + ndw);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamex1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamex2(2*(jj-quo2)-2) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIx1(2*(jj-quo2)-2,:) = [bp1 min(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx1(2*(jj-quo2)-2,:) = [ep1 min(pc(:,2)) h(ii)];
    Diag(ii-1).FpIx2(2*(jj-quo2)-2,:) = [bp1 max(pc(:,2)) h(ii-1)];
    Diag(ii-1).FpJx2(2*(jj-quo2)-2,:) = [ep1 max(pc(:,2)) h(ii)];

% The Diagonals Between Corners
else

    % For Diagonals
    bp1 = RP(jj,1) - bw;          % X-Coordinate of Start Points of Diagonals
```

```
        bp2 = RP(jj,1) + bw;
        ep1 = RP(jj,1);              % X-Coordinate of End Points of Diagonals
        ep2 = ep1;

        % Diagonal Names
        dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2));
        dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1));
        dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + ndw);
        dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + ndw);

        % Name List of Diagonals on First&Second X-Plane
        Diag(ii-1).Fnamex1(2*(jj-quo2)-2) = cellstr(dg1);
        Diag(ii-1).Fnamex1(2*(jj-quo2)-1) = cellstr(dg2);
        Diag(ii-1).Fnamex2(2*(jj-quo2)-2) = cellstr(dg3);
        Diag(ii-1).Fnamex2(2*(jj-quo2)-1) = cellstr(dg4);

        % Diagonal I&J Point Coordinates on First&Second X-Plane
        Diag(ii-1).FpIx1(2*(jj-quo2)-2,:) = [bp1 min(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx1(2*(jj-quo2)-2,:) = [ep1 min(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx1(2*(jj-quo2)-1,:) = [bp2 min(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx1(2*(jj-quo2)-1,:) = [ep2 min(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx2(2*(jj-quo2)-2,:) = [bp1 max(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx2(2*(jj-quo2)-2,:) = [ep1 max(pc(:,2)) h(ii)];
        Diag(ii-1).FpIx2(2*(jj-quo2)-1,:) = [bp2 max(pc(:,2)) h(ii-1)];
        Diag(ii-1).FpJx2(2*(jj-quo2)-1,:) = [ep2 max(pc(:,2)) h(ii)];

    end
end
```

```
elseif jj > 2*ndsp && jj <= 3*ndsp        % For Y-Axis Diagonals
   if mod(ii,2) == 0

      % First Corner Diagonals
      if mod(jj,ndsp) == 1

         bp1 = RP(jj,2);
         ep1 = RP(jj,2) + bw;

         % Diagonal Names
         dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw);
         dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw + ndf);

         % Name List of Diagonals on First&Second X-Plane
         Diag(ii-1).Fnamey1(2*(jj-quo2)-1) = cellstr(dg1);
         Diag(ii-1).Fnamey2(2*(jj-quo2)-1) = cellstr(dg2);

         % Diagonal I&J Point Coordinates on First&Second X-Plane
         Diag(ii-1).FpIy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) bp1 h(ii-1)];
         Diag(ii-1).FpJy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) ep1 h(ii)];
         Diag(ii-1).FpIy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) bp1 h(ii-1)];
         Diag(ii-1).FpJy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) ep1 h(ii)];

      % End Corner Diagonals
      elseif mod(jj,ndsp) == 0

         bp1 = RP(jj,2);
         ep1 = RP(jj,2) - bw;
```

```
    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw);
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw + ndf);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamey1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamey2(2*(jj-quo2)-2) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) ep1 h(ii)];
    Diag(ii-1).FpIy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) ep1 h(ii)];

% The Diagonals Between Corners
else

    bp1 = RP(jj,2);
    bp2 = bp1;
    ep1 = RP(jj,2) - bw;
    ep2 = RP(jj,2) + bw;

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw);
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw);
    dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw + ndf);
    dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw + ndf);

    % Name List of Diagonals on First&Second X-Plane
```

```
        Diag(ii-1).Fnamey1(2*(jj-quo2)-2) = cellstr(dg1);
        Diag(ii-1).Fnamey1(2*(jj-quo2)-1) = cellstr(dg2);
        Diag(ii-1).Fnamey2(2*(jj-quo2)-2) = cellstr(dg3);
        Diag(ii-1).Fnamey2(2*(jj-quo2)-1) = cellstr(dg4);

        % Diagonal I&J Point Coordinates on First&Second X-Plane
        Diag(ii-1).FpIy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) bp1 h(ii-1)];
        Diag(ii-1).FpJy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) ep1 h(ii)];
        Diag(ii-1).FpIy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) bp2 h(ii-1)];
        Diag(ii-1).FpJy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) ep2 h(ii)];
        Diag(ii-1).FpIy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) bp1 h(ii-1)];
        Diag(ii-1).FpJy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) ep1 h(ii)];
        Diag(ii-1).FpIy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) bp2 h(ii-1)];
        Diag(ii-1).FpJy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) ep2 h(ii)];

    end
else
    % First Corner Diagonals
    if mod(jj,ndsp) == 1

        bp1 = RP(jj,2) + bw;
        ep1 = RP(jj,2);

        % Diagonal Names
        dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw);
        dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw + ndf);

        % Name List of Diagonals on First&Second X-Plane
        Diag(ii-1).Fnamey1(2*(jj-quo2)-1) = cellstr(dg1);
```

```
    Diag(ii-1).Fnamey2(2*(jj-quo2)-1) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) ep1 h(ii)];
    Diag(ii-1).FpIy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) ep1 h(ii)];

% End Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,2) - bw;
    ep1 = RP(jj,2);

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw);
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw + ndf);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamey1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamey2(2*(jj-quo2)-2) = cellstr(dg2);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) ep1 h(ii)];
    Diag(ii-1).FpIy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) ep1 h(ii)];

% The Diagonals Between Corners
```

```
else

    % For Diagonals
    bp1 = RP(jj,2) - bw;
    bp2 = RP(jj,2) + bw;
    ep1 = RP(jj,2);
    ep2 = ep1;

    % Diagonal Names
    dg1 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw);
    dg2 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw);
    dg3 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-2) + 2*ndw + ndf);
    dg4 = num2str(((ii-2)*Diag(ii-1).Fnum) + (2*(jj-quo2)-1) + 2*ndw + ndf);

    % Name List of Diagonals on First&Second X-Plane
    Diag(ii-1).Fnamey1(2*(jj-quo2)-2) = cellstr(dg1);
    Diag(ii-1).Fnamey1(2*(jj-quo2)-1) = cellstr(dg2);
    Diag(ii-1).Fnamey2(2*(jj-quo2)-2) = cellstr(dg3);
    Diag(ii-1).Fnamey2(2*(jj-quo2)-1) = cellstr(dg4);

    % Diagonal I&J Point Coordinates on First&Second X-Plane
    Diag(ii-1).FpIy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy1(2*(jj-quo2)-2,:) = [min(pc(:,1)) ep1 h(ii)];
    Diag(ii-1).FpIy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) bp2 h(ii-1)];
    Diag(ii-1).FpJy1(2*(jj-quo2)-1,:) = [min(pc(:,1)) ep2 h(ii)];
    Diag(ii-1).FpIy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) bp1 h(ii-1)];
    Diag(ii-1).FpJy2(2*(jj-quo2)-2,:) = [max(pc(:,1)) ep1 h(ii)];
    Diag(ii-1).FpIy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) bp2 h(ii-1)];
    Diag(ii-1).FpJy2(2*(jj-quo2)-1,:) = [max(pc(:,1)) ep2 h(ii)];
```

```
                end
            end
        end
    end
  end
end

% Get Core Base Point Coordinates
pidx = pc(:,3)==0;
cpc = pc(pidx,1:2);         % Core Point Coordinates

% Define Vertical Floor Load Ratio for Diagonal Shell
delxc = max(cpc(:,1)) - min(cpc(:,1));      % Core Width on X-Axis
delyc = max(cpc(:,2)) - min(cpc(:,2));      % Core Width on Y-Axis
FloorArea = webl*flangel;
TempArea = (webl+delxc)*(flangel+delyc)/4;
AR = (FloorArea - TempArea)/FloorArea;

end
```

APPENDIX-6 Functiıon of shapeAnalyze

```
function [Diag,WEIGHT,EDCR] = shapeAnalyze(Diag,dtype,LC,nmod,h,ndw,ndf,...
                                hm,webl,flangel,el,alpha,uw,Ed,sigmay,DL,LL,wp,AR)

% Define Global Shear, Over-Turning Moment and Gravity Loads on Modules
WL = zeros(nmod,1);          % Total Shear on Module
M = zeros(nmod,1);           % Overturning Moment due to Wind Load on Module
VL1 = zeros(nmod,1);         % Dead Load on Module
VL2 = zeros(nmod,1);         % Live Load on Module

for ii = 1:nmod
   hw = max(h)-h(ii);        % Height Difference Between Roof, and Bottom of Module
   WL(ii) = wp*flangel*hw;                    % Shear on Module
   M(ii) = wp*flangel*(hw^2)/2;               % Moment on Module
   VL1(ii) = DL*AR*webl*flangel*hw/el;        % Dead Load on Module
   VL2(ii) = LL*AR*webl*flangel*hw/el;        % Live Load on Module
   Diag(ii).DeadLoad = VL1(ii);
   Diag(ii).LiveLoad = VL2(ii);
   Diag(ii).Shear = WL(ii);
   Diag(ii).Moment = M(ii);
end

theta = [Diag.Angle].';      % Module Diagonal Angle Array
ctheta = cosd(theta);        % Cosinus Theta Array
stheta = sind(theta);        % Sinus Theta Array
```

```
% ------------------------------------------------------------------------------------------ %
% --- Strength Analysis --- %

% Define Module Vertical Load Component on Diagonals
Fdl = zeros(nmod,1);          % Dead Load Component on Diagonals
Fll = zeros(nmod,1);          % Live Load Component on Diagonals

for ii = 1:nmod
   Fdl(ii) = -VL1(ii)/(2*(ndw+ndf)*stheta(ii));
   Fll(ii) = -VL2(ii)/(2*(ndw+ndf)*stheta(ii));
end

% Define Module Shear Component on Diagonals
Fsw = zeros(nmod,ndw);      % Shear Component on Web Diagonals
Fsf1 = zeros(nmod,ndf);     % Shear Component on Windward Flange Diagonals
Fsf2 = zeros(nmod,ndf);     % Shear Component on Leeward Flange Diagonals

for ii = 1:nmod
   Fs = WL(ii)/(2*ndw*ctheta(ii));
   if dtype == 0
      if mod(ii,2) == 1
         Fsf1(ii,[1 end]) = Fs/2;
         Fsf2(ii,[1 end]) = -Fs/2;
         Fsw(ii,[1 end]) = [-1 1]*Fs/2;
         Fsw(ii,2:2:ndw-2) = Fs;
         Fsw(ii,3:2:(ndw-1)) = -Fs;
      else
         Fsw(ii,1:2:(ndw-1)) = Fs;
         Fsw(ii,2:2:ndw) = -Fs;
```

```
        end
    else
        if mod(ii,2) == 1
            Fsw(ii,1:2:(ndw-1)) = Fs;
            Fsw(ii,2:2:ndw) = -Fs;
        else
            Fsf1(ii,[1 end]) = Fs/2;
            Fsf2(ii,[1 end]) = -Fs/2;
            Fsw(ii,[1 end]) = [-1 1]*Fs/2;
            Fsw(ii,2:2:ndw-2) = Fs;
            Fsw(ii,3:2:(ndw-1)) = -Fs;
        end
    end
end

% Define Module Bending Moment Component on Diagonals
Fmf1 = zeros(nmod,ndf);      % Moment Component on Windward Flange Diagonals
Fmf2 = zeros(nmod,ndf);      % Moment Component on Leeward Flange Diagonals
Fmw = zeros(nmod,ndw);       % Moment Component on Web Diagonals

for ii = 1:nmod
    % Loading Point Coordinates
    mpc = unique([Diag(ii).FpJx1; Diag(ii).FpJx2;
                  Diag(ii).FpJy1; Diag(ii).FpJy2],'rows');
    mpc(:,[2 3]) = [];
    % X-Axis Coordinate of Module Centroid
    cc = (max(mpc) + min(mpc))/2;    % Centroid Coordinate
    dc = max(mpc) - cc;              % Max Distance from Centroid
```

```
% Sum of All Loading Point Distance^2 from Centroid
d1 = cc - mpc;
sqd = sum(d1.^2);
% Web Diagonal Distance Array from Centroid
mpcw = unique(Diag(ii).FpJx1,'rows','stable');
mpcw(:,[2 3]) = [];
d2 = cc - mpcw;
nd2 = numel(d2);
% Max Moment Component (Flange Members)
Fm = M(ii)*dc/(sqd*2*stheta(ii));
% Windward Flange Diagonal Forces
Fmf1(ii,:) = Fm;
% Leeward Flange Diagonal Forces
Fmf2(ii,:) = -Fm;
% Web Plane Diagonal Forces
if dtype == 0
   if mod(ii,2) == 1
      Fmw(ii,1) = Fm;
      Fmw(ii,ndw) = -Fm;
      for jj = 2:nd2-1
         Fmw(ii,[2*jj-2 2*jj-1]) = Fm*d2(jj)/dc;
      end
   else
      for jj = 1:nd2
         Fmw(ii,[2*jj-1 2*jj]) = Fm*d2(jj)/dc;
      end
   end
else
   if mod(ii,2) == 1
```

```
            for jj = 1:nd2
                Fmw(ii,[2*jj-1 2*jj]) = Fm*d2(jj)/dc;
            end
        else
            Fmw(ii,1) = Fm;
            Fmw(ii,ndw) = -Fm;
            for jj = 2:nd2-1
                Fmw(ii,[2*jj-2 2*jj]-1) = Fm*d2(jj)/dc;
            end
        end
    end
end

% Evaluate Diagonal Axial Forces under Load Combinations
kk = ismember(LC(1).CaseList,'DEAD');
qq = ismember(LC(1).CaseList,'LIVE');
rr = ismember(LC(1).CaseList,'WX');
fdgw = zeros(1,ndw);            % Web Diagonal Forces
fdgf1 = zeros(1,ndf);           % Windward Flange Diagonal Forces
fdgf2 = zeros(1,ndf);           % Leeward Flange Diagonal Forces
Ft = zeros(nmod,numel(LC));     % Max Module Tension Forces by Load Comb.
Fc = zeros(nmod,numel(LC));     % Max Module Compression Forces by Load Comb.

for ii = 1:nmod
    for tt = 1:numel(LC)
        % Web Diagonal Forces
        for jj = 1:ndw
            fdgw(jj) = Fdl(ii)*LC(tt).Factor(kk)+Fll(ii)*LC(tt).Factor(qq)+...
                LC(tt).Factor(rr)*(Fsw(ii,jj)+Fmw(ii,jj));
```

```
        end
        % Flange Diagonal Forces
        for jj = 1:ndf
            fdgf1(jj) = Fdl(ii)*LC(tt).Factor(kk)+Fll(ii)*LC(tt).Factor(qq)+...
                LC(tt).Factor(rr)*(Fmf1(ii,jj)+Fsf1(ii,jj));
            fdgf2(jj) = Fdl(ii)*LC(tt).Factor(kk)+Fll(ii)*LC(tt).Factor(qq)+...
                LC(tt).Factor(rr)*(Fmf2(ii,jj)+Fsf2(ii,jj));
        end
        % Temporary All Module Forces under Load Comb.
        tempf = [fdgw fdgf1 fdgf2];
        % Evaluate Tension & Compression Forces
        if min(tempf)>=0
            Ft(ii,tt) = max(tempf);
            Fc(ii,tt) = 0;
        elseif max(tempf)<0
            Ft(ii,tt) = 0;
            Fc(ii,tt) = abs(min(tempf));
        else
            Ft(ii,tt) = max(tempf);
            Fc(ii,tt) = abs(min(tempf));
        end
    end
end

% Resulting Module Max Tension & Compression Axial Forces Array
Fdgt = max(Ft.').';
Fdgc = max(Fc.').';
```

```
% Required Diagonal Area Results under Tension & Compression
Adst = Fdgt/(0.9*sigmay);
Adsc = Fdgc/(0.6*sigmay);

% Resulting Module Diagonal Areas by Strength Analysis
Adstr = max(Adst,Adsc);

% Store Results
for ii = 1:nmod
   Diag(ii).Tension = Fdgt(ii);
   Diag(ii).Compression = Fdgc(ii);
   Diag(ii).Astr = Adstr(ii)*10000;       % Strength Area as cm2
end

% ------------------------------------------------------------------------------------------ %
% --- Stiffness Analysis --- %

% Evaluate Contribution Parameter for Bending
wcb = zeros(nmod,1);                % Web Contribution Against Bending
fcs = zeros(nmod,1);                % Flange Contribution Against Shear

for ii = 1:nmod
   % Bending
   Ffm = sum(Fmf1(ii,:));           % Total Diagonal Force on One Flange
   Fwm = sum(abs(Fmw(ii,:)));       % Total Diagonal Force on One Web
   wfr = Fwm/Ffm;                   % Web/Flange Force Ratio
   wcb(ii) = wfr*ndf;
   % Shear
   Ffs = sum(Fsf1(ii,:));           % Total Diagonal Force on One Flange
```

```
    Fws = sum(abs(Fsw(ii,:)));          % Total Diagonal Force on One Web
    fwr = Ffs/Fws;                      % Flange/Web Force Ratio
    fcs(ii) = fwr*ndw;
end

% Evaluate Stiffness & Final Area
gamma = zeros(nmod,1);                  % Module Shear Strain
ksi = zeros(nmod,1);                    % Module Curvature
Adstiff = zeros(nmod,1);                % Stiffness Area
kgr = zeros(nmod,1);                    % Ksi/Gamma Ratio
ksisum = zeros(nmod,1);                 % Sum of Bending Disp. from Base to
Adfin = zeros(nmod,1);                  % Module Final Area
dratio = zeros(nmod,1);                 % Inter-Module Drift Ratio
drift = zeros(nmod,1);                  % Module Drift
ldisp = zeros(nmod,1);                  % Cumulative Lateral Drift

for ii = 1:nmod
    kgr(ii) = 4*M(ii)*(ndw+fcs(ii))*(ctheta(ii)^2)/...
              (WL(ii)*(webl^2)*(ndf+wcb(ii))*(stheta(ii)^2));
    if ii == 1
        gamma(ii) = alpha/(kgr(ii)*hm(ii)+1);
        ksi(ii) = kgr(ii)*gamma(ii);
    else
        ksisum(ii) = sum(ksi(1:ii-1).*hm(1:ii-1));
        gamma(ii) = (alpha-ksisum(ii))/(kgr(ii)*hm(ii)+1);
        ksi(ii) = gamma(ii)*kgr(ii);
    end
    Adstiff(ii) = WL(ii)/(2*Ed*(ndw+fcs(ii))*gamma(ii)*stheta(ii)*(ctheta(ii)^2));
    if Adstr(ii) > Adstiff(ii)
```

```
        Adfin(ii) = Adstr(ii);
        gamma(ii) = WL(ii)/(2*Ed*Adfin(ii)*(ndw+fcs(ii))*stheta(ii)*(ctheta(ii)^2));
        ksi(ii) = 2*M(ii)/(Ed*Adfin(ii)*(webl^2)*(ndf+wcb(ii))*(stheta(ii)^3));
    else
        Adfin(ii) = Adstiff(ii);
    end
    if ii == 1
        dratio(ii) = gamma(ii) + ksi(ii)*hm(ii);
        drift(ii) = dratio(ii)*hm(ii);
        ldisp(ii) = drift(ii);
    else
        dratio(ii) = gamma(ii) + ksi(ii)*hm(ii) + sum(ksi(1:ii-1).*hm(1:ii-1));
        drift(ii) = dratio(ii)*hm(ii);
        ldisp(ii) = drift(ii) + ldisp(ii-1);
    end
end

% Store Results
for ii = 1:nmod
    Diag(ii).Afin = Adfin(ii)*10000;            % Final Diagonal Area as cm2
    Diag(ii).Astiff = Adstiff(ii)*10000;        % Stiffness Area as cm2
    Diag(ii).Dratio = dratio(ii);               % Module Drift Ratio
    Diag(ii).Drift = drift(ii);                 % Inter-Module Drift
    Diag(ii).LatDisp = ldisp(ii);               % Total Module Lateral Displacement
    Diag(ii).DCR = 100*Adstr(ii)/Adfin(ii);   % Demand to Capacity Ratio
end

% Evaluate Pipe Sections According to ASCI 360-10
r1 = zeros(nmod,1);         % Inside Radius(m)
```

```
r2 = zeros(nmod,1);         % Outside Radius(m)

for ii = 1:nmod
   r1(ii) = sqrt((2*sigmay*(Diag(ii).Flength^2)/((pi^2)*Ed))-…
                  (Diag(ii).Afin/(20000*pi)));
   r2(ii) = sqrt((2*sigmay*(Diag(ii).Flength^2)/((pi^2)*Ed))+…
                  (Diag(ii).Afin/(20000*pi)));
end

t3 = ceil(200*r2)/100;          % Resulting Outside Diameter(m) (Precision by centimeters)
tw = ceil(100*(r2-r1))/100;   % Resulting Wall Thickness(m)   (Precision by centimeters)

% Define Sections
for ii = 1:nmod
   Diag(ii).Sname = ['PIPE ' num2str(t3(ii)*1000) 'x' num2str(tw(ii)*1000)];
end

% Weight Evaluation
wgt = zeros(nmod,1);
for ii = 1:nmod
   % Total Weight of Structure (Tons)
   wgt(ii) = Diag(ii).Flength*Diag(ii).Afin*Diag(ii).Fnum*uw/100000;
   Diag(ii).Weight = wgt(ii);            % Module Weight (Tons)
end
% Evaluate Effective DCR
dcrwgt = [Diag.DCR].*[Diag.Weight];
WEIGHT = sum(wgt);
EDCR = sum(dcrwgt)/WEIGHT;
end
```

APPENDIX-7. Function of TournamentSelection

```
function i=TournamentSelection(pop,m)

    nPop=numel(pop);

    S=randsample(nPop,m);

    spop=pop(S);

    sweights=[spop.Weight];

    [~, j]=min(sweights);

    i=S(j);

end
```

APPENDIX-8. Function of Crossover

```
function [y1, y2]=Crossover(x1,x2)

    pSinglePoint=0.1;
    pDoublePoint=0.2;
    pUniform=1-pSinglePoint-pDoublePoint;

    METHOD=RouletteWheelSelection([pSinglePoint pDoublePoint pUniform]);

    switch METHOD
        case 1
            [y1, y2]=SinglePointCrossover(x1,x2);

        case 2
            [y1, y2]=DoublePointCrossover(x1,x2);

        case 3
            [y1, y2]=UniformCrossover(x1,x2);

    end

end
```

APPENDIX-9. Function of SinglePointCrossover

```
function [y1, y2]=SinglePointCrossover(x1,x2)

    nVar=numel(x1);

    c=randi([1 nVar-1]);

    y1=[x1(1:c) x2(c+1:end)];
    y2=[x2(1:c) x1(c+1:end)];

end
```

APPENDIX-10. Function of DoublePointCrossover

```
function [y1, y2]=DoublePointCrossover(x1,x2)

    nVar=numel(x1);

    cc=randsample(nVar-1,2);
    c1=min(cc);
    c2=max(cc);

    y1=[x1(1:c1) x2(c1+1:c2) x1(c2+1:end)];
    y2=[x2(1:c1) x1(c1+1:c2) x2(c2+1:end)];

end
```

APPENDIX-11. Function of UniformCrossover

```
function [y1, y2]=UniformCrossover(x1,x2)

    alpha=randi([0 1],size(x1));

    y1=alpha.*x1+(1-alpha).*x2;
    y2=alpha.*x2+(1-alpha).*x1;

end
```

APPENDIX-12. Function of Mutate

```
function y=Mutate(x,mu)

    nVar=numel(x);

    nmu=ceil(mu*nVar);

    j=randsample(nVar,nmu);

    y=x;
    y(j)=1-x(j);

end
```

APPENDIX-13. Function of BestDraw

```
function [dgl,mdest] = BestDraw(ndsp,dtype,bw,pc,h,RP,ShapeModelPath,...
   PointObj,folname,File,FrameObj,View)

% --- Create a Copy of SAP2000 Template to be Deleted After Analysis --- %
% Make a New Temporary Folder under Temp Folder
dest = [cd '\' folname];
mkdir(dest);

% Copy SAP2000 Template to Temp
copyfile(ShapeModelPath,dest);

% Open SAP2000 Model
mdest = [dest '\shape.sdb'];
File.OpenFile(mdest);

% --- Model Drawing Process --- %
% Assign Bracing and Ring Beams

dglx1 = {''};                    % Diagonal Frame Name List on First X Plane
dglx2 = {''};                    % Diagonal Frame Name List on Second X Plane
dgly1 = {''};                    % Diagonal Frame Name List on First Y Plane
dgly2 = {''};                    % Diagonal Frame Name List on Second Y Plane
dg1 = ''; dg2 = ''; dg3 = ''; dg4 = ''; rb1 = ''; rb2 = ''; rb3 = ''; rb4 = '';

for ii = 2:numel(h)
   for jj = 1:size(RP,1)
      quo1 = floor((jj-1)/ndsp);
      quo2 = quo1*ndsp;
```

```
% Defining Start & End Coordinates of Diagonals
if dtype == 0
   if jj <= ndsp                    % For X-Axis Diagonals
      if mod(ii,2) == 0

         bp1 = RP(jj,1);            % X-Coordinate of Beginning Points of Diagonals
         bp2 = bp1;
         ep1 = RP(jj,1) - bw;       % X-Coordinate of End Points of Diagonals
         ep2 = RP(jj,1) + bw;

         % Drawing Ring Beams
         [~,rb1] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),ep2,…
                                                  min(pc(:,2)),h(ii),rb1);
         [~,rb2] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),ep2,…
                                                  max(pc(:,2)),h(ii),rb2);

      else

         % For Diagonals
         bp1 = RP(jj,1) - bw;
         bp2 = RP(jj,1) + bw;
         ep1 = RP(jj,1);
         ep2 = ep1;


         % For Ring Beams
         ep3 = RP(jj,1) + 2*bw;
```

```
    % Drawing the Ring Beams
    if mod(jj,ndsp) == 1

        [~,rb1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii),ep1,…
                                                    min(pc(:,2)),h(ii),rb1);
        [~,rb2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii),ep1,…
                                                    max(pc(:,2)),h(ii),rb2);
        [~,rb3] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),ep3,…
                                                    min(pc(:,2)),h(ii),rb3);
        [~,rb4] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),ep3,…
                                                    max(pc(:,2)),h(ii),rb4);

    elseif mod(jj,ndsp) == 0

        [~,rb3] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),bp2,…
                                                    min(pc(:,2)),h(ii),rb3);
        [~,rb4] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),bp2,…
                                                    max(pc(:,2)),h(ii),rb4);

    else

        [~,rb3] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),ep3,…
                                                    min(pc(:,2)),h(ii),rb3);
        [~,rb4] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),ep3,…
                                                    max(pc(:,2)),h(ii),rb4);

    end
end
```

```
    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),ep1,...
                                                    min(pc(:,2)),h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(bp2,min(pc(:,2)),h(ii-1),ep2,…
                                                    min(pc(:,2)),h(ii),dg2);
    [~,dg3] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),ep1,…
                                                    max(pc(:,2)),h(ii),dg3);
    [~,dg4] = FrameObj.AddByCoord(bp2,max(pc(:,2)),h(ii-1),ep2,…
                                                    max(pc(:,2)),h(ii),dg4);

    dg1 = char(dg1);
    dg2 = char(dg2);
    dg3 = char(dg3);
    dg4 = char(dg4);
     % Name List of Diagonals on First X-Plane
    dglx1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
    dglx1(ii-1,2*(jj-quo2)) = cellstr(dg2);
     % Name List of Diagonals on Second X-Plane
    dglx2(ii-1,2*(jj-quo2)-1) = cellstr(dg3);
    dglx2(ii-1,2*(jj-quo2)) = cellstr(dg4);

elseif jj > 2*ndsp && jj <= 3*ndsp                % For Y-Axis Diagonals
    if mod(ii,2) == 0

       bp1 = RP(jj,2);            % Y-Coordinate of Beginning Points of Diagonals
       bp2 = bp1;
       ep1 = RP(jj,2) - bw;       % Y-Coordinate of End Points of Diagonals
       ep2 = RP(jj,2) + bw;
```

```
    % Drawing the Ring Beams
    [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                    ep2,h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    ep2,h(ii),rb2);

else

    % For Diagonals
    bp1 = RP(jj,2) - bw;
    bp2 = RP(jj,2) + bw;
    ep1 = RP(jj,2);
    ep2 = ep1;

    % For Ring Beams
    ep3 = RP(jj,2) + 2*bw;

    % Drawing the Ring Beams
    if mod(jj,ndsp) == 1

       [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii),min(pc(:,1)),…
                                                    ep1,h(ii),rb1);
       [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii),max(pc(:,1)),…
                                                    ep1,h(ii),rb2);
       [~,rb3] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                    ep3,h(ii),rb3);
       [~,rb4] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    ep3,h(ii),rb4);
```

```
    elseif mod(jj,ndsp) == 0

        [~,rb3] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                    bp2,h(ii),rb3);
        [~,rb4] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    bp2,h(ii),rb4);

    else

        [~,rb3] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                    ep3,h(ii),rb3);
        [~,rb4] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    ep3,h(ii),rb4);

    end
end

% Drawing the Diagonals on SAP Model
[~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),min(pc(:,1)),…
                                                    ep1,h(ii),dg1);
[~,dg2] = FrameObj.AddByCoord(min(pc(:,1)),bp2,h(ii-1),min(pc(:,1)),…
                                                    ep2,h(ii),dg2);
[~,dg3] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),max(pc(:,1)),…
                                                    ep1,h(ii),dg3);
[~,dg4] = FrameObj.AddByCoord(max(pc(:,1)),bp2,h(ii-1),max(pc(:,1)),…
                                                    ep2,h(ii),dg4);
dg1 = char(dg1);
dg2 = char(dg2);
dg3 = char(dg3);
```

```
        dg4 = char(dg4);
        % Name List of Diagonals on First Y-Plane
        dgly1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
        dgly1(ii-1,2*(jj-quo2)) = cellstr(dg2);
        % Name List of Diagonals on Second Y-Plane
        dgly2(ii-1,2*(jj-quo2)-1) = cellstr(dg3);
        dgly2(ii-1,2*(jj-quo2)) = cellstr(dg4);

    end
else                        % For dtype = 1
    if jj <= ndsp           % For X-Axis Diagonals
        if mod(ii,2) == 0
            % The Left Corner Diagonals
            if mod(jj,ndsp) == 1

                bp1 = RP(jj,1);
                ep1 = RP(jj,1) + bw;
                % Drawing Ring Beams
                [~,rb1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii),…
                                                ep1,min(pc(:,2)),h(ii),rb1);
                [~,rb2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii),…
                                                ep1,max(pc(:,2)),h(ii),rb2);

                % Drawing the Diagonals on SAP Model
                [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                                ep1,min(pc(:,2)),h(ii),dg1);
                [~,dg2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                                ep1,max(pc(:,2)),h(ii),dg2);
                dg1 = char(dg1);
```

```
    dg2 = char(dg2);
    dglx1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
    dglx2(ii-1,2*(jj-quo2)-1) = cellstr(dg2);

% The Right Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,1);
    ep1 = RP(jj,1) - bw;

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),…
                                  bp1,min(pc(:,2)),h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),…
                                  bp1,max(pc(:,2)),h(ii),rb2);


    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                  ep1,min(pc(:,2)),h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                  ep1,max(pc(:,2)),h(ii),dg2);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dglx1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
    dglx2(ii-1,2*(jj-quo2)-2) = cellstr(dg2);

% The Diagonals Between Corners
else
```

```
bp1 = RP(jj,1);
bp2 = bp1;
ep1 = RP(jj,1) - bw;
ep2 = RP(jj,1) + bw;

% Drawing Ring Beams
[~,rb1] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),…
                                        ep2,min(pc(:,2)),h(ii),rb1);
[~,rb2] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),…
                                        ep2,max(pc(:,2)),h(ii),rb2);

% Drawing the Diagonals on SAP Model
[~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                        ep1,min(pc(:,2)),h(ii),dg1);
[~,dg2] = FrameObj.AddByCoord(bp2,min(pc(:,2)),h(ii-1),…
                                        ep2,min(pc(:,2)),h(ii),dg2);
[~,dg3] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                        ep1,max(pc(:,2)),h(ii),dg3);
[~,dg4] = FrameObj.AddByCoord(bp2,max(pc(:,2)),h(ii-1),…
                                        ep2,max(pc(:,2)),h(ii),dg4);
dg1 = char(dg1);
dg2 = char(dg2);
dg3 = char(dg3);
dg4 = char(dg4);
% Name List of Diagonals on First X-Plane
dglx1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
dglx1(ii-1,2*(jj-quo2)-1) = cellstr(dg2);
% Name List of Diagonals on Second X-Plane
```

```
        dglx2(ii-1,2*(jj-quo2)-2) = cellstr(dg3);
        dglx2(ii-1,2*(jj-quo2)-1) = cellstr(dg4);

    end
else

    % The Left Corner Diagonals
    if mod(jj,ndsp) == 1

        % For Diagonals
        bp1 = RP(jj,1) + bw;
        ep1 = RP(jj,1);

        % For Ring Beams
        ep2 = RP(jj,1) + (2*bw);

        % Drawing Ring Beams
        [~,rb1] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),…
                                                ep2,min(pc(:,2)),h(ii),rb1);
        [~,rb2] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),…
                                                ep2,max(pc(:,2)),h(ii),rb2);

        % Drawing the Diagonals on SAP Model
        [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                                ep1,min(pc(:,2)),h(ii),dg1);
        [~,dg2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                                ep1,max(pc(:,2)),h(ii),dg2);
        dg1 = char(dg1);
        dg2 = char(dg2);
```

```
    dglx1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
    dglx2(ii-1,2*(jj-quo2)-1) = cellstr(dg2);

% The Right Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,1) - bw;
    ep1 = RP(jj,1);

    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                        ep1,min(pc(:,2)),h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                        ep1,max(pc(:,2)),h(ii),dg2);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dglx1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
    dglx2(ii-1,2*(jj-quo2)-2) = cellstr(dg2);

% The Diagonals Between Corners
else

    % For Diagonals
    bp1 = RP(jj,1) - bw;
    bp2 = RP(jj,1) + bw;
    ep1 = RP(jj,1);
    ep2 = ep1;

    % For Ring Beams
```

```
    ep3 = RP(jj,1) + (2*bw);

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(ep1,min(pc(:,2)),h(ii),…
                                  ep3,min(pc(:,2)),h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(ep1,max(pc(:,2)),h(ii),…
                                  ep3,max(pc(:,2)),h(ii),rb2);

    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(bp1,min(pc(:,2)),h(ii-1),…
                                  ep1,min(pc(:,2)),h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(bp2,min(pc(:,2)),h(ii-1),…
                                  ep2,min(pc(:,2)),h(ii),dg2);
    [~,dg3] = FrameObj.AddByCoord(bp1,max(pc(:,2)),h(ii-1),…
                                  ep1,max(pc(:,2)),h(ii),dg3);
    [~,dg4] = FrameObj.AddByCoord(bp2,max(pc(:,2)),h(ii-1),…
                                  ep2,max(pc(:,2)),h(ii),dg4);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dg3 = char(dg3);
    dg4 = char(dg4);
    % Name List of Diagonals on First X-Plane
    dglx1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
    dglx1(ii-1,2*(jj-quo2)-1) = cellstr(dg2);
    % Name List of Diagonals on Second X-Plane
    dglx2(ii-1,2*(jj-quo2)-2) = cellstr(dg3);
    dglx2(ii-1,2*(jj-quo2)-1) = cellstr(dg4);

end
```

```
    end
elseif jj > 2*ndsp && jj <= 3*ndsp        % For Y-Axis Diagonals
    if mod(ii,2) == 0
        % The Left Corner Diagonals
        if mod(jj,ndsp) == 1

            bp1 = RP(jj,2);
            ep1 = RP(jj,2) + bw;

            % Drawing Ring Beams
            [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii),min(pc(:,1)),…
                                                    ep1,h(ii),rb1);
            [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii),max(pc(:,1)),…
                                                    ep1,h(ii),rb2);
            % Drawing the Diagonals on SAP Model
            [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
                                                    min(pc(:,1)),ep1,h(ii),dg1);
            [~,dg2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                    max(pc(:,1)),ep1,h(ii),dg2);
            dg1 = char(dg1);
            dg2 = char(dg2);
            dgly1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
            dgly2(ii-1,2*(jj-quo2)-1) = cellstr(dg2);

        % The Right Corner Diagonals
        elseif mod(jj,ndsp) == 0

            bp1 = RP(jj,2);
```

```
    ep1 = RP(jj,2) - bw;

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                bp1,h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                bp1,h(ii),rb2);

    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
                                                min(pc(:,1)),ep1,h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                max(pc(:,1)),ep1,h(ii),dg2);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dgly1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
    dgly2(ii-1,2*(jj-quo2)-2) = cellstr(dg2);

% The Diagonals Between Corners
else

    bp1 = RP(jj,2);
    bp2 = bp1;
    ep1 = RP(jj,2) - bw;
    ep2 = RP(jj,2) + bw;

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                ep2,h(ii),rb1);
```

```
        [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    ep2,h(ii),rb2);

        % Drawing the Diagonals on SAP Model
        [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
                                                    min(pc(:,1)),ep1,h(ii),dg1);
        [~,dg2] = FrameObj.AddByCoord(min(pc(:,1)),bp2,h(ii-1),…
                                                    min(pc(:,1)),ep2,h(ii),dg2);
        [~,dg3] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                    max(pc(:,1)),ep1,h(ii),dg3);
        [~,dg4] = FrameObj.AddByCoord(max(pc(:,1)),bp2,h(ii-1),…
                                                    max(pc(:,1)),ep2,h(ii),dg4);
        dg1 = char(dg1);
        dg2 = char(dg2);
        dg3 = char(dg3);
        dg4 = char(dg4);
        % Name List of Diagonals on First X-Plane
        dgly1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
        dgly1(ii-1,2*(jj-quo2)-1) = cellstr(dg2);
        % Name List of Diagonals on Second X-Plane
        dgly2(ii-1,2*(jj-quo2)-2) = cellstr(dg3);
        dgly2(ii-1,2*(jj-quo2)-1) = cellstr(dg4);

    end
else
    % The Left Corner Diagonals
    if mod(jj,ndsp) == 1

        bp1 = RP(jj,2) + bw;
```

```
    ep1 = RP(jj,2);

    % For Ring Beams
    ep2 = RP(jj,2) + (2*bw);

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                    ep2,h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                    ep2,h(ii),rb2);


    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
                                                    min(pc(:,1)),ep1,h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                    max(pc(:,1)),ep1,h(ii),dg2);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dgly1(ii-1,2*(jj-quo2)-1) = cellstr(dg1);
    dgly2(ii-1,2*(jj-quo2)-1) = cellstr(dg2);

% The Right Corner Diagonals
elseif mod(jj,ndsp) == 0

    bp1 = RP(jj,2) - bw;
    ep1 = RP(jj,2);

    % Drawing the Diagonals on SAP Model
```

```
    [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
                                                  min(pc(:,1)),ep1,h(ii),dg1);
    [~,dg2] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                  max(pc(:,1)),ep1,h(ii),dg2);
    dg1 = char(dg1);
    dg2 = char(dg2);
    dgly1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
    dgly2(ii-1,2*(jj-quo2)-2) = cellstr(dg2);

% The Diagonals Between Corners
else

    % For Diagonals
    bp1 = RP(jj,2) - bw;
    bp2 = RP(jj,2) + bw;
    ep1 = RP(jj,2);
    ep2 = ep1;

    % For Ring Beams
    ep3 = RP(jj,2) + (2*bw);

    % Drawing Ring Beams
    [~,rb1] = FrameObj.AddByCoord(min(pc(:,1)),ep1,h(ii),min(pc(:,1)),…
                                                  ep3,h(ii),rb1);
    [~,rb2] = FrameObj.AddByCoord(max(pc(:,1)),ep1,h(ii),max(pc(:,1)),…
                                                  ep3,h(ii),rb2);

    % Drawing the Diagonals on SAP Model
    [~,dg1] = FrameObj.AddByCoord(min(pc(:,1)),bp1,h(ii-1),…
```

```
                                                    min(pc(:,1)),ep1,h(ii),dg1);
                [~,dg2] = FrameObj.AddByCoord(min(pc(:,1)),bp2,h(ii-1),…
                                                    min(pc(:,1)),ep2,h(ii),dg2);
                [~,dg3] = FrameObj.AddByCoord(max(pc(:,1)),bp1,h(ii-1),…
                                                    max(pc(:,1)),ep1,h(ii),dg3);
                [~,dg4] = FrameObj.AddByCoord(max(pc(:,1)),bp2,h(ii-1),…
                                                    max(pc(:,1)),ep2,h(ii),dg4);
                dg1 = char(dg1);
                dg2 = char(dg2);
                dg3 = char(dg3);
                dg4 = char(dg4);

                % Name List of Diagonals on First X-Plane
                dgly1(ii-1,2*(jj-quo2)-2) = cellstr(dg1);
                dgly1(ii-1,2*(jj-quo2)-1) = cellstr(dg2);
                % Name List of Diagonals on Second X-Plane
                dgly2(ii-1,2*(jj-quo2)-2) = cellstr(dg3);
                dgly2(ii-1,2*(jj-quo2)-1) = cellstr(dg4);

            end
        end
    end
  end
end
% Store All Diagonals Name List in a Matrix
dgl  = [dglx1 dglx2 dgly1 dgly2];

View.RefreshView;
```

```
% Get Point Names of Diagrid Shell
nps = 0;                        % Number of Points
plcs = cellstr('');             % List of Points as Cell
[~,nps,plcs] = PointObj.GetNameList(nps,plcs);
nps = double(nps);
plcs = cell(plcs);
plds = str2double(plcs);        % List of Points as Double

% Get Point Coordinates of Diagrid Shell
pcs = zeros(nps,3);
for ii = 1:nps
   x = 0;
   y = 0;
   z = 0;
   [~,x,y,z] = PointObj.GetCoordCartesian(char(plcs(ii)),x,y,z);
   % pc = x,y and z Coordinates of Points
   pcs(ii,1) = x;
   pcs(ii,2) = y;
   pcs(ii,3) = z;
end

% Assign Supports to Base Points of Diagrid
ind = pcs(:,3)==0;
supp = plds(ind);                       % Support Point Names at Base

% For Pin Supports at Diagrid Base: True=Released, False=Not Released
value = false(6,1);
value(1:3) = true;
```

```
for ii = 1:size(supp,2)
    PointObj.SetRestraint(num2str(supp(ii)), value);
end
end
```