Linnæus University

Master's Thesis in Mechanical/Structural Engineering

Parametrized Finite Element Simulation of Multi-Storey Timber Structures

Authors: *Le Kuai*

Surpervisor LNU: *Sigurdur Ormarsson*
Examinar LNU: *Krushna Mahapatra*

Course Code: *17VT - 5TS04E*

Semester: *Spring 2016, 30 credits*

Linnaeus University, Faculty of Technology

Table of Contents

Abstract

With the acceleration of global urbanization trends, more and more intentions are put on multi-storey buildings. As the world leading area of wood construction, European countries started the construction of multi-storey timber building for a decade ago. However, unlike the traditional buildings made of reinforced concrete, the design of wooden high-rise buildings would face a substantial amount of new challenges because such high-rise timber buildings are touching the limitations of the timber engineering field. In this thesis, a parameterized three-dimensional FE-model (in ABAQUS) of a multi-storey timber frame building is created. Variable geometrical parameters, connection stiffness as well as boundary connections and applied wind and gravity loads are defined in a Python script to make it possible to analyze the influence of these parameters on the global structural behavior of the studied multi-storey timber frame building. The results and analysis implied that the script successfully worked and was capable to create different complex building geometries in an easy way for the finite element analysis.

Keywords: wood, multi-storey timber frame structure, finite element analysis, ABAQUS, simulation, Python script.

Acknowledgement

I would like to thank my supervisor, Professor Sigurdur Ormarsson, who not only provide the opportunity for me to work on this multi-storey project, but also has carefully guided me through the whole process. His suggestions and guidance have been essential for the performance of this thesis.

I would also thank my family and friends who encourage me and stay with me at all time. They provide the driving forces for this thesis.

Le Kuai

Vaxjo 22th of May 2016

1. Introduction:

80% of the world's population of eight billion will live in urban situation by 2050. In the next decade some 75 million multiple family housing units will be required in China alone to accommodate the approximately 300 million people expected to migrate into urban & adjacent suburban areas.

Peter Wilson

1.1 Background

The global urbanization trends all over the world required increasing amount of buildings. Multi-storey buildings are especially needed in some countries such as China and India, which has huge population and with low land per capita. Simultaneously, tall timber buildings are more closed to real life thanks to the introduction of Cross Laminated Timber (CLT) a decade ago, (BOELLAARD, 2012). From Birdport House, the 8 storey building in London at 2010 to 14 storey building in Norway at 2015, European countries presented their interest to construct more amounts of timber structures.

There is still challenge for engineers to find new ways of utilizing wood as main load carrying material in tall buildings up to 20 stories because such high-rise timber buildings are touching the limit of what is possible in the field of timber engineering. Therefore some designs and monitoring techniques based on software simulations need to be established.

Thanks to Oden (1972), Crisfield (1991), Kleiber (1989) and many other pioneers, there have been many theories and methods developed for linear and nonlinear studies of different building structures, whereas the design of tall timber buildings would face a substantial amount of new challenges in terms of analyzing deformations and stability failures caused by horizontal wind loads. Therefore in order to prevent overturn of a building and minimize lateral deformations, horizontal and vertical stabilization systems are of significant importance in tall timber structures, (E. Frü

hwald&T. Toratti& S. Thelandersson& E. Serrano, 2007). Create good FE-models of the entire timber structure is therefore very important for the engineers to catch the potential failure or buckling problems.

1.2 Aim and Purpose

The aim of this project is to create a flexible three-dimensional finite element model of a multi-storey timber structure. The model is created with a parameterized Python script which enables effective and flexible creation of buildings with different sizes and geometries.

The purpose of this thesis is to use the finite element model to study the overall structural behaviors of different multi-storey timber frame buildings. The studies are especially concentrated on how connection stiffness has influence on global structural behaviors of the multi-storey timber buildings. This model is also a key model for so-called adaptive modelling of the timber buildings. There are the most critical connections analyzed very carefully (with a full 3D solid modelling including nonlinear contact interactions between elements) and linked to this global model through different constraint requirements.

1.3 Hypothesis and Limitations

It is expected that the methodology established in this thesis and the results acquired from the corresponding simulations based on finite element method would help engineers to perform more accurate analyses of multi-storey timber buildings. Moreover, the results obtained from the established model could be extended broadly through the parameterized modelling method. Modelling of multi-storey timber buildings is a very challenging work from many aspects.

This thesis is an important step in the effort to carefully analyze the whole multi-storey timber buildings. But because of limited time of this work an extensive researches cannot be done.

The following model limitations have been made in this thesis:

- No internal walls are able to be created in in the model.
- The slab geometry is not changeable.

- The diagonal columns are not able to be crossed.
- A stability analysis has not been performed.

2. Literature review:

This section describes former research of multi-storey timber buildings that touch upon this thesis work.

2.1 A 14-storey timber building in Norway

In the city of Bergen in Norway, there is a fourteen storey residential building called “Treet”, which means “The tree” in Norwegian. The building started to be constructed in April 2014 and became finished in the late of 2015.

The structural design looks like a cabinet which contains of drawers. The cabinet is made of 3D glulam frame truss and the ‘drawers’, are built by prefabricated residential modules of wood. The main bending stiffness of the building is given by the glulam truss which lies in the facade plane. The CLT walls are separate from the main load bearing system and they are just used for elevator shafts and staircases, (Abrahamsen RB,Malo, 2014)

For the building, the wind load was used as the main dominate load in the design process according to Eurocode. The maximum wind speed and design wind pressure were calculated to be V = 44.8 m/s and q_d = 1.26 kN/m2, (Abrahamsen RB,Malo, 2014). Although the building “Treet” is located in an earthquake zone according to the regulations in Norway, the earthquake load is not considered to be the critical load because it is smaller than the load caused by the wind. It may be noted that the frame structure of the “Treet” building is of similar type as the frame structure studied in this work.

2.2 A brief history of finite element software

Linear finite element analysis was first renowned in the middle of the 20th century thanks to the articles by Turner, Clough, Martin and Topp(1956), and the work presented by the Boeing Research Group. Shortly after that this method was expanded

to solve different types of non-linear problems. At beginning, this method was rejected by many scientists because of its 'lacks of essence'. Some of the involved articles ware for example refused by the *Journal of Applied Method* for many years, (Miao,Z& Xiaochuan,Y& Jianhui, L, 1991). However, numerous engineers were aware of the meanings and prospects of the finite element analysis since it provides possibilities to solve the real and complicated problems which were impossible to solve with normal analytical methods.

The first finite element program came from Ed Wilson in 1960s, this implicit finite element program was nameless, however, used in countless laboratories around the world, which brought a huge impact on engineering analysis. The second generation program SAP was developed at Berkeley University in America (a structural analysis program), and the first commercial finite element program named MARC was developed by Pedro V. Marcal at Brown University in United States, (Andreas Öchsner, Marco Öchsner, 2016). Around the same period, John A. Swanson developed the first non-linear finite element program ANSYS, which monopolized the market for years, (About-ANSYS, 2017).

In the beginning, most commercial finite element programs focused on static solutions and implicit dynamic solutions. The researchers in Berkeley Universities contributed the theories of these static and implicit methods.

Another offspring of finite element method is explicit programs. In 1964, the program developed by Costantino in Illinois Institute of Technology (IIT) in Chicago might be the first explicit finite element program, (Miao,Z& Xiaochuan,Y& Jianhui, L, 1991). It focuses on linear materials and small deformations. It calculated nodal forces and nodal displacements using stiffness matrices with optimal bandwidth. The milestones in the development of explicit finite element programs came from John Hallquist in Lawrence Livermore national laboratory. The key to his success was the development of interactions of contact-impact in his program, DYNA. This algorithm made significant breakthroughs for engineering simulations. In the 1980s, the DYNA was commercialized by ESI in France, named PAMCRASH, and the explicit method became another choice for engineers.

The software ABAQUS (used in this work) was developed by HKS (Hibbitt, Karlsson & Sorensen, INC) which was located in Rhode Island, USA. It is powerful simulation software based on finite element method. It consists of two main modules: ABAQUS/Standard and ABAQUS/Explicit, among ABAQUS/Standard, three more

modules are attached: ABAQUS/Aqua, ABAQUS/Design and ABAQUS/Foundation. Furthermore, it provides interface for MOLDFLOW and ADAMS/Flex, which are two simulation software for dynamic analyzes of mechanical systems. Figure 1 shows the relationship between these modules.

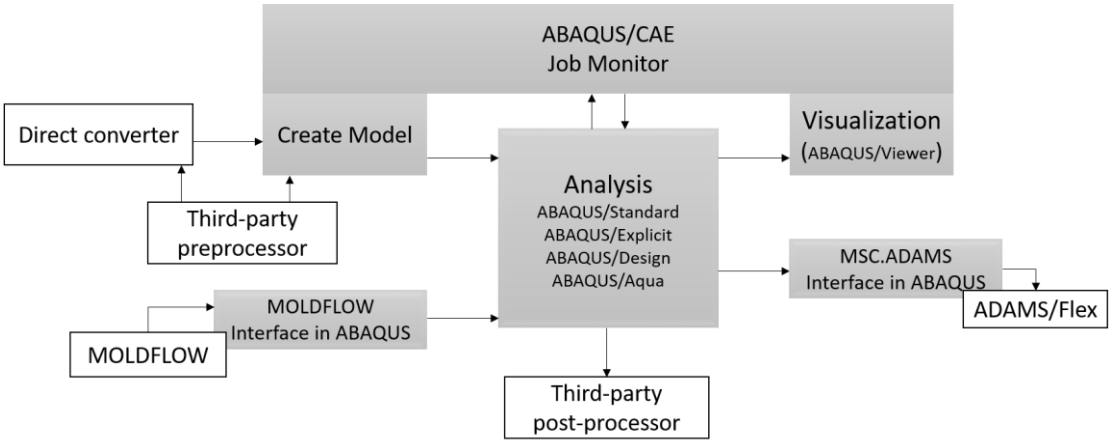

Figure 1: Different modulus used in ABAQUS.

As a general FE-tool, ABAQUS is not only solving traditional structural problems. It is also able to simulate heat conduction, mass diffusion, and acoustics, coupled thermo-electrical problems and coupled seepage-stress problems. It also provides a simplified method to create models for these complicated problems. Typical model in ABAQUS consists of several distinct modules, which jointly describe the physical problem that needs to be analyzed. Typical input parameters needed for an FE-model is: Discrete geometry description; Cross-section properties; Assigned material properties; Loads and boundary conditions; Type of analysis; Output requirements etc. ABAQUS/CAE is a complete user-friendly environment for generations of ABAQUS models. It provides a simple and concise interface for model creation, job monitoring and visualization of results. The ABAQUS/CAE is divided into several function modules, each module define one specific part/property in the model such as the geometric shape, material properties, meshes and so on. After complete creation of the model, ABAQUS/CAE creates an input file (file.inp) needed for the ABAQUS-solver. The simulated results from the ABAQUS-solver are thereafter visualized in ABAQUS/CAE for further studies.

3. Theory

3.1 Wood

3.11 Background

It is essential to have a good understanding of the used construction materials to get better understanding of the structural functionality of the constructions. This section presents the basic background of wood and explains the characteristics that might be significant in this thesis work.

From archaeological research, the first known tree is *Wattieza*, its fossils were found in New York State, year 2007, dating back to the Middle Devonian, the age nearly 385 million years ago, (Rerallack & Huang, 2011). After millions of years evolution, this species now consist of thousands different family members. The wood falls into two main groups, the softwoods and hardwoods, the former are gymnosperms known as conifers, the latter represent the angiosperms known as dicotyledons which have broad leaves, (Desch & Dinwoodie, 1996).

3.12 Chemical properties of wood

The wood mainly consists of three elements: 49% Carbon of dry weight, 6% Hydrogen of dry weight and 44% Oxygen of dry weight. Besides these three main elements, slight amount of Nitrogen and Ash were also found to be elements in wood, (Williamson, 2002).

The cell wall, consist of millions of microfibrils which can be subdivided into several different layers according to the arrangement of the microfibrils, (Desch & Dinwoodie, 1996). As shown in Figure 2, the cell wall can be simplified to primary wall and secondary wall. The primary wall is characteristically thin with random arrangement of microfibrils. The secondary wall consist of three layers, the Outer layer, Middle layer and Inner layer, denoted as S_1, S_2, and S_3.

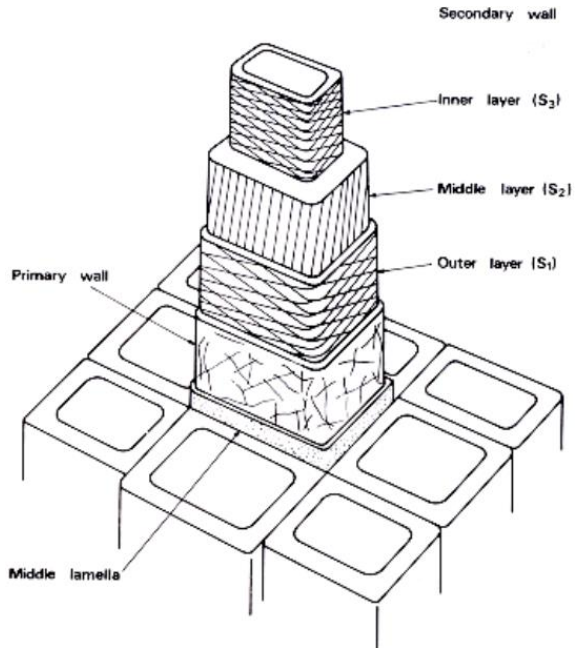

Figure 2: Schematic drawing of the microstructure of wood, (Williamson, 2002).

As an orthotropic material, the material properties of wood depend on its three main material directions, called longitudinal direction, radial direction and tangential direction. It means wood to have three symmetry planes which are perpendicular to each other at every material point inside the material. Figure 3 shows the material structure of wood including coordinate system that defines the orthotropic material directions (*l*,*r*,*t*).

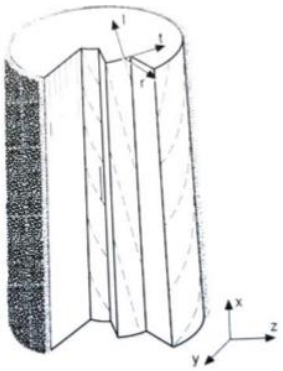

Figure 3: Local and global coordinate system of wood, (Ormarsson, 1999).

3.13 Elastic material behavior of wood

For modelling of three dimensional elastic behavior of wood the constitutive relation can be given by generalized Hooke's law, see e.g. (Ormarsson, 1999). The elastic

strain is given by

$$\bar{\varepsilon}_e = \bar{C}\bar{\sigma} \tag{3.1}$$

where $\bar{C}$ is the local compliance matrix and $\bar{\varepsilon}_e$ and $\bar{\sigma}$ represents the local elastic strains and the stress matrices that refer to the local material directions (*l*,*r*,*t*). Both the elastic strain and stress matrices are 6 x 1 matrices as shown below:

$$\bar{\sigma} = \begin{bmatrix} \sigma_l \\ \sigma_r \\ \sigma_t \\ \gamma^e_{lr} \\ \gamma^e_{lt} \\ \gamma^e_{rt} \end{bmatrix} \quad , \quad \bar{\varepsilon}_e = \begin{bmatrix} \varepsilon^e_l \\ \varepsilon^e_r \\ \varepsilon^e_t \\ \tau_{lr} \\ \tau_{lt} \\ \tau_{rt} \end{bmatrix} \tag{3.2}$$

and the compliance matrix $\bar{C}$ is given as:

$$\bar{C} = \begin{bmatrix} \frac{1}{E_l} & -\frac{\nu_{rl}}{E_r} & -\frac{\nu_{tl}}{E_t} & 0 & 0 & 0 \\ -\frac{\nu_{lr}}{E_l} & \frac{1}{E_r} & -\frac{\nu_{tr}}{E_t} & 0 & 0 & 0 \\ -\frac{\nu_{lt}}{E_l} & -\frac{\nu_{rt}}{E_r} & \frac{1}{E_t} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{lr}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{lt}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{rt}} \end{bmatrix} \tag{3.3}$$

The E_l E_r and E_t are representing the moduli of elasticity in the orthotropic directions and G_{lr}, G_{lt}, G_{rt} are the shear moduli in the respective orthotropic plane. In addition, the parameters ν_{lr}, $\nu_{rl}, \nu_{lt}, \nu_{tl}, \nu_{tr}, \nu_{rt}$ are the Poisson's ratios for the orthotropic material directions. Other material behaviors of wood, such as shrinking/swelling, mechano sorption, creep and plasticity, will not be discussed here because the modelling work presented here is only dealing with elastic behavior in multi-storey timber buildings.

3.14 Engineered wood products

Normal sawn timber can only be found up to certain dimensions, because of the

lengths and cross section dimensions of the trees and also because of the method the sawmills are treating it, (Crocetti, Johansson, & Kliger, 2011). In order to produce larger timber dimensions, the engineered wood products (EWP) were created. They are glued products that consist of sawn timber lamellae, veneers, particles and some type of adhesive, (Crocetti, Johansson, & Kliger, 2011) Figure 3 presents an approximate timeline for the invention of distinguished types of engineered wood products.

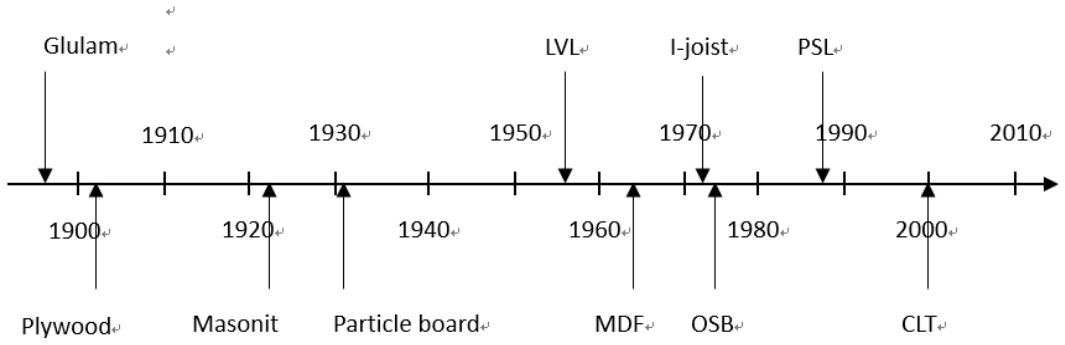

Figure 3: A timeline for the invention of distinguished types of EWP

Glued laminated timber

Glued laminated timber is one of the oldest engineering wood products. The first patent of it was processed by Otto Hetzer in 1906, (Crocetti, Johansson, & Kliger, 2011). This kind of timber is consisting of at least four laminations bonded together with adhesives. The glued laminated timber is not only significantly stronger than solid beams of same size but also lower in the variability in strength, which has been proved by tests. The most common failures for a glulam beams are bending failures and shear failures, occurred at knots or finger-joints. High tensile stresses perpendicular to the grain direction need also to be considered, especially in curved structures and beams with holes and notches, (Crocetti, Johansson, & Kliger, 2011).

Cross-laminated timber

After the production start of cross-laminated timber in the age of 2000's, this kind of engineered wood product has won a significant market share in timber construction. It consists of glued sawn timber in a layered structure where the layers are oriented perpendicular to each other. The number of layers is often an odd number and the size can be as large as 300 mm in thickness, 3000 mm wide and 24 meter long.

Laminated veneer lumber

The laminated veneer lumber, also named LVL, is another kind of EWP. It was developed between 1950s and 1960s and it is made up by gluing wood veneer sheets together, (Crocetti, Johansson, & Kliger, 2011). The maximum size of LVL reaches 3000 x 24000 mm. The veneer based products provide high stiffness and bending, shear, tension and compression strength. For some types of LVL products, e.g. floor structures, intended to provide higher stiffness cross the panel, it is produced with some layers with the grain direction oriented perpendicular to the fiber direction of the majority layers.

3.2 Finite Element Method

Finite Element Method (FEM) or Finite Element Analysis (FEA), is a numerical method to solve all kind of field problems, (Cook, Plesha, & Malkus, 1989). It comes originated from the matrix-displacement method for frame structure in 1940s to 1950s. Finite Element Method is an approximation method for solving differential equations, which established for physical or engineering problems, including control equations and boundary conditions. The analysis program has been standardized, it includes structural or regional discretization, unit analysis, holistic analysis and numerical solution, (Shouyi, 2005). From the physical point of view, the finite element method transforms the continuum problem into the discrete problem. From the mathematical point of view, the finite element method transforms the differential equation problem into the algebraic equation problem. As a result, the problem of infinite degree of freedom becomes a finite degree of freedom problem.

Equilibrium equation

Equilibrium differential equation, also referred to as equilibrium equation, describes the relationship between stresses and the external forces. As an example, the equilibrium equation for a plane problem is:

$$\left.\begin{array}{l} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + f_x = 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + f_y = 0 \end{array}\right\} \tag{3.4}$$

For a plane stress problem there are only three stress components σ_x, σ_y and τ_{xy} in the *xy*-plane and the other three stress components $\sigma_z = \tau_{zx} = \tau_{zy} = 0$. The terms f_x, f_y are the body forces acting in the x respective y directions.

Constitutive equation

A constitutive equation of a material represents the relationship between stresses and strains. For an isotropic linear elastic material, it presents:

$$\sigma = D\varepsilon \tag{3.5}$$

where the material matrix for a plane problem is given by

$$D = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \tag{3.6}$$

Finite element formulation

The first step in a finite element analysis is to discrete the structure by dividing the object into numerous numbers of small finite elements through imaginary lines or faces, see Figure 4*(a)*. In addition the elements are assigned a limited amount of nodal points (with some degrees of freedom) per element. Figure 4*(b)* shows a typical triangle element used for two dimensional stress analyses, where every nodal point has two displacement degrees of freedom.

$$a^e = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix}, \qquad a_i = \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} \qquad (i = 1,2,3) \tag{3.7}$$

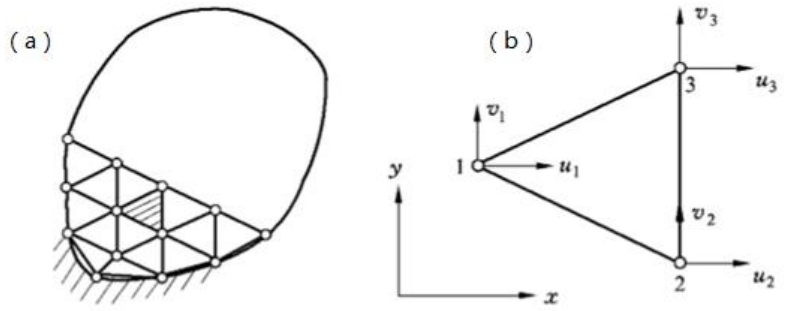

Figure 4: *(a)*: Element mesh used to discretize a two dimensional structure
(b): A linear triangle element used for 2D stress analysis.

Afterward, the stiffness matrix K^e for each finite element needs to be established. The stiffness matrix describes the relationship between the nodal forces and the nodal displacements. The external load is used to calculate the equivalent nodal forces and

the element stiffness matrix K^e can be calculated with an integral expression as

$$K^e = \int_{\Omega^e} B^T D B d\Omega \tag{3.8}$$

where D is the material matrix and B is a matrix based on the element approximations. For a triangle element with six degrees of freedom the stiffness matrix becomes a 6 x 6 matrix. The relationship between internal nodal forces and displacements could therefore be written as:

$$\begin{Bmatrix} U_1 \\ V_1 \\ U_2 \\ V_2 \\ U_3 \\ V_3 \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} \tag{3.9}$$

where k_{ij} presents the nodal force in the ith d.o.f. when an unit displacement occurs at the jth d.o.f while all the other displacement components are zero, (Shouyi, 2005).

In FE-analysis, the discrete system is based on nodes, so it is necessary to transfer the distributed load acting on the element body into equivalent nodal forces. Nodes where concentrated forces are acting are usually chosen as nodal points. Therefore the physical stress and facial stress need to be transferred to nodal forces according to static equivalent. Static equivalent means that when the element has a virtual displacement, the virtual work caused by physical stresses should be equal to the virtual work caused by its equivalent nodal forces, (Shouyi, 2005), which has:

$$\begin{aligned} \delta a^{e\,T} P_f^e &= \int_{\Omega^e} \delta u^T f \Omega \\ \delta u &= N \delta a \end{aligned} \tag{3.10}$$

where δa^e is the vector of nodal virtual displacements ; P_f^e is the equivalent nodal force vector for the element and *f* is representing the stresses in the element.

For a plane problem, the element equation for static equivalence is given by

$$P_f^e = \iint_{A^e} N^T f t \, dx \, dy \tag{3.11}$$

where A^e is area of the element, *N* is the shape function matrix and *t* is the thickness of the 2D structure.

The next step is to perform the global analysis where the basic task is to establish the global equilibrium equation consisting the global stiffness matrix and the global nodal force vector. A set of algebraic equations for nodal equilibrium conditions needs to be established for all nodal points. Figure 5 shows when one nodal point is taken from the structure and the nodal forces acting on it are equal to the internal forces coming

from every element associated with this nodal point.

The equilibrium equations can be written as:

$$\sum_e F_{ie} = P_i, \quad \text{where} \quad F_{ie} = [U_{ie} \quad V_{ie}]^T \text{ and } P_i = [X_i \quad Y_i]^T \tag{3.12}$$

where F_{ie} is an internal force vector for all elements having nodal point i and P_i is a vector for the external forces acting on the nodal point i. Combining all equilibrium equations together, the global finite element equation can be written as

$$Ka = P \tag{3.13}$$

where K is the global stiffness matrix; a is the global nodal displacement vector and P is the global nodal load vector, which are:

$$a = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_N \end{Bmatrix}, \qquad P = \begin{Bmatrix} P_1 \\ P_2 \\ P_3 \\ \dots \\ P_N \end{Bmatrix} \tag{3.14}$$

For establishment of the global stiffness matrix, the most common method is to create it by assembling the element stiffness matrices based on topology information for the whole element mesh.

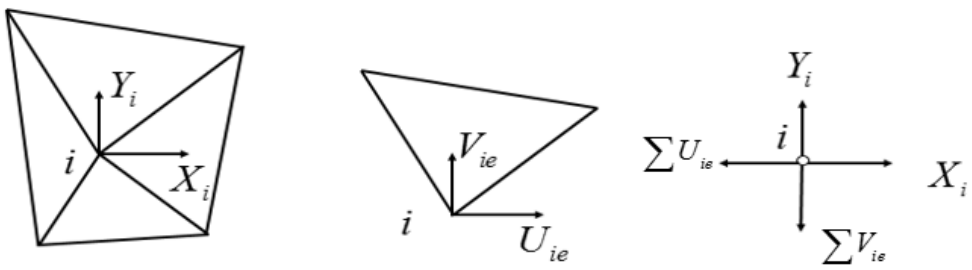

Figure 5: Illustration of force equilibrium in node i.

3.3 Joints in constructions

In structural timber design mechanical joints are of significant importance since they need to be able to transfer forces between different timber elements. One single dowel joint is only able to transfer shear forces acting perpendicular to the dowels' axis. If the connection consist of multiply dowels (dowel group) it is possible to transfer a bending moment, (Bader, Schweigler, & Hochreiner, 2016). Moment-resisting joints

can work in three different structures, see Figure 6, which *(a)* is the example of a joint of handrail which supports the beam in a pedestrian bridge, and *(b)* shows a splice joint and *(c)* present a knee joint connections in a frame, (Racher, 1995).

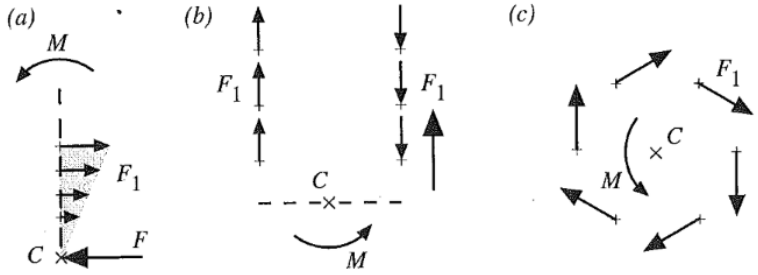

Figure 6: The three types of moment-resisting joints, (Bader, Schweigler, & Hochreiner, 2016).

According to the above connection types, there are the joints similar to type (*c*) that are of interest for design of high timber frame buildings.

For connection design, both local and global analysis needs to be considered to understand both the local force distributions in the connection and their influence on the global structural behavior of the building.

There are two main types of joints used in the timber engineering field, a splice joint which connects two parallel timber members, see Figure 7*(a)*; and a joint which connect two cross-grained timber members, see Figure 7*(b)*. The timber members are assumed as rigid in order to analyze the elastic behavior of the joints since they are stiffer than the joints. If the relative rotation of the joint members is ω, see Figure 7*(c)* the external moment can be defined as

$$\mathrm{M} = \sum_{j=1}^{n} F_{M,j}\, r_j \qquad (3.15)$$

where $F_{M,j}$ represents the internal forces acting on the fastener, and the radial distance r_j is from the rotational center C to the fasteners.

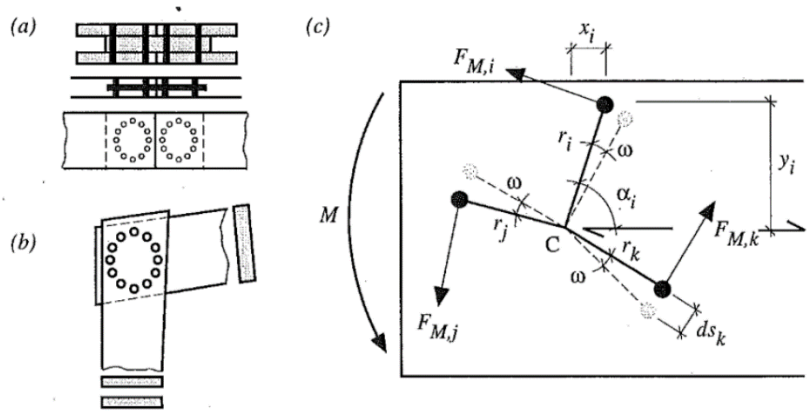

Figure 7: *(a)* A joint connecting two parallel timber members; *(b)* A joint connecting cross-grained timber members; *(c)* Illustration of force distribution caused by a relative rotation ϖ of two joint members, (Bader, Schweigler, & Hochreiner, 2016).

Since the force distribution in the joint is assumed linear elastic, the following relationship can be defined

$$ds_k = \frac{F_{M,k}}{K_{\alpha_k}} \qquad and \qquad \omega = \frac{d_{s_k}}{r_k} = \frac{F_{M,k}}{K_{\alpha_k} r_k} = \frac{F_{M,j}}{K_{\alpha_j} r_j} = \frac{F_{M,i}}{K_{\alpha_i} r_i} \tag{3.16}$$

From equations (3.15) and (3.16), the force acting on the fastener i is given by

$$F_{M,i} = \frac{K_{\alpha_i} r_i}{K_r} M \tag{3.17}$$

where the rotational stiffness of the connection is given as

$$K_r = \sum_{j=1}^{n} K_{\alpha_j} {r_j}^2 \tag{3.18}$$

3.4 Basic beam theory

A beam element is a kind of one-dimensional element which can resist axial, lateral, and twisting loads. A structure consisting of beam elements is regarded as a frame in a structural analysis. A continuous beam over two or more supports can be simulated using one beam element per span. This means when beam elements are used in statics analysis the discretization phase of modeling becomes quite trivial, (Cook, Plesha, & Malkus, 1989).

Beam elements can be divided into 2D beam elements and 3D beam elements. A 2D beam element has three degrees of freedom, namely, one rotational, one axial and one lateral translation degrees of freedom. If the shear strain is assumed to be neglected then the *Euler-Bernoulli beam theory* can be used. For high beams the transverse shear deformation is taken into account according to *Timoshenko beam theory* when beam vibration is studied, (Cook, Plesha, & Malkus, 1989).

The stiffness matrix for a 2D beam element is 6 by 6 in size. The following equation shows typical stiffness matrix for a 2D beam element:

$$[k] = \begin{bmatrix} \frac{12EI_z}{L^3} & \frac{6EI_z}{L^2} & \frac{-12EI_z}{L^3} & \frac{6EI_z}{L^2} \\ \frac{6EI_z}{L^2} & \frac{4EI_z}{L^2} & \frac{-6EI_z}{L^2} & \frac{2EI_z}{L^2} \\ \frac{-12EI_z}{L^3} & \frac{-6EI_z}{L^2} & \frac{12EI_z}{L^3} & \frac{-6EI_z}{L^2} \\ \frac{6EI_z}{L^2} & \frac{2EI_z}{L^2} & \frac{-6EI_z}{L^2} & \frac{4EI_z}{L^2} \end{bmatrix} \tag{3.19}$$

For a 3D beam element, there are six degrees of freedom located in each nodal point. For a local coordinate system, they are the displacements along the coordinate axes u_i, v_i, w_i and the rotation angles around same axes $\theta_{xi}, \theta_{yi}, \theta_{zi}$, see Figure 8. Similarly, the nodal forces are six i.e. axial force U_i, two transverse shear forces V_i and W_i, torsional moment M_{xi} and two bending moments M_{zi} and M_{yi}. Hence, the nodal displacement vector and the nodal force vector are presented as:

$$a^e = \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix}, a_i = \begin{bmatrix} u_i & v_i & w_i & \theta_{xi} & \theta_{yi} & \theta_{zi} \end{bmatrix}, (i = 1,2) \tag{3.20}$$

$$F^e = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}, F_i = \begin{bmatrix} U_i & V_i & W_i & M_{xi} & M_{yi} & M_{zi} \end{bmatrix}, (i = 1,2) \tag{3.21}$$

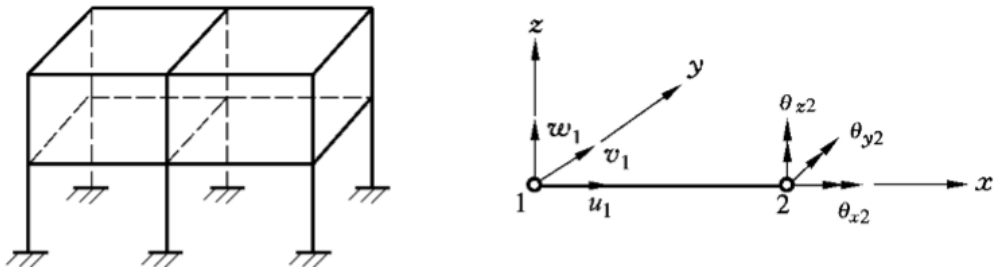

Figure 8: A frame structure analyzed with 3D beam elements with six degrees of

freedom.

According to finite element method, the stiffness matrix can be calculated as:

$$K^e = \int_{\Omega^e} B^T D B d\Omega = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}, k_{ij} = \int_{\Omega^e} B_i^T D B_j d\Omega \tag{3.22}$$

where the sub matrices are:

$$k_{11} = \begin{bmatrix} \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2} \\ 0 & 0 & \frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 \\ 0 & -\frac{6EI_y}{l^2} & 0 & \frac{GI_k}{l} & 0 & 0 \\ \frac{6EI_z}{l^2} & 0 & 0 & 0 & \frac{4EI_y}{l} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{4EI_z}{l} \end{bmatrix} \tag{3.23}$$

$$k_{22} = \begin{bmatrix} \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{l^3} & 0 & 0 & 0 & -\frac{6EI_z}{l^2} \\ 0 & 0 & \frac{12EI_y}{l^3} & 0 & \frac{6EI_y}{l^2} & 0 \\ 0 & \frac{6EI_y}{l^2} & 0 & \frac{GI_k}{l} & 0 & 0 \\ -\frac{6EI_z}{l^2} & 0 & 0 & 0 & \frac{4EI_y}{l} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{4EI_z}{l} \end{bmatrix} \tag{3.24}$$

$$k_{12} = k_{21}^T \begin{bmatrix} \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2} \\ 0 & 0 & -\frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 \\ 0 & 0 & 0 & -\frac{GI_k}{l} & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{l^2} & 0 & \frac{2EI_y}{l} & 0 \\ 0 & -\frac{6EI_z}{l^2} & 0 & 0 & 0 & \frac{2EI_z}{l} \end{bmatrix} \tag{3.25}$$

For a 3D element the global stiffness matrix can be calculated as

$$\overline{K^e} = T^T K^e \tag{3.26}$$

where the transform matrix T is:

$$T = \begin{bmatrix} \phi & 0 & 0 & 0 \\ 0 & \phi & 0 & 0 \\ 0 & 0 & \phi & 0 \\ 0 & 0 & 0 & \phi \end{bmatrix}, \phi = \begin{bmatrix} \cos(x,\bar{x}) & \cos(x,\bar{y}) & \cos(x,\bar{z}) \\ \cos(y,\bar{x}) & \cos(y,\bar{y}) & \cos(y,\bar{z}) \\ \cos(z,\bar{x}) & \cos(z,\bar{y}) & \cos(z,\bar{z}) \end{bmatrix} \quad (3.27)$$

3.5 Basic shell theory

When the size of one direction (thickness) is much smaller than the other directions in a structure, as well as the stress could also be ignored in that direction, the problems can be treated and solved as a shell problem. Two examples of shells are a water tank and an eggshell, (Cook, Plesha, & Malkus, 1989).

For shells, both the geometric shapes and their deformations can be quite complicated, which shows great hardness for constitutive equations to solve. Analytical solutions for shells can be acquired only under special circumstances. Hence, the finite element method becomes a powerful tool for the analysis of shell structures, (Shouyi, 2005).

There are three main elements used for the analysis of shell structures with the finite element method, i.e. flat shell element, curved element and degenerated shell element. This thesis is focusing on the flat shell elements so the other two element types are not discussed here.

A curved shell structure can be meshed with multiple shell elements where each element is a curved element. However, when the mesh is small enough, the curved elements are not needed and the flat shell element can approximately be used. The set of these flat elements could alternate the previous curved shell elements, (Greene, Strome, & Weikel, 1961). The most common flat shell elements are rectangular shell element and triangle shell element.

Shell structure can be divided into rectangular elements as shown in Figure 9. For the established local coordinate systems the shell element has 5 degrees of freedom in each nodal point.

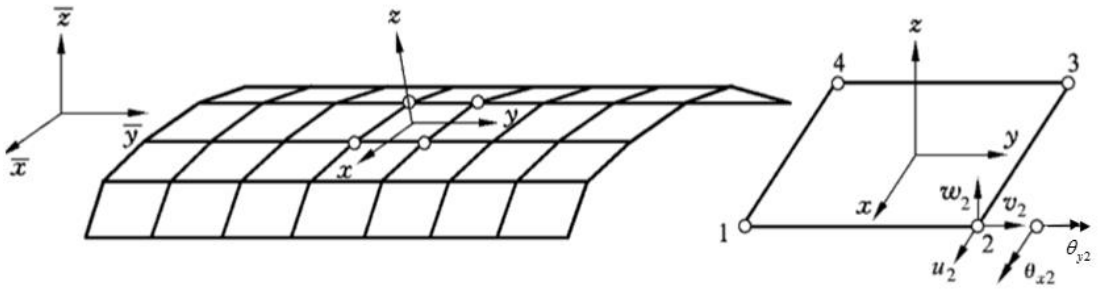

Figure 9: A shell structure meshed with rectangular shell elements with 5 degrees of freedom in each nodal point.

For a local coordinate system the equilibrium equation for the element in-plane action can be presented as:

$$F^{p} = K^{P} a^{P} \tag{3.28}$$

where the nodal force vector F^{p} and the nodal displacement vector a^{P} are:

$$F^{P} = \begin{Bmatrix} F_1^{P} \\ F_2^{P} \\ F_3^{P} \\ F_4^{P} \end{Bmatrix}, F_i^{P} = \begin{Bmatrix} U_i \\ V_i \end{Bmatrix}; a^{P} = \begin{Bmatrix} a_1^{P} \\ a_2^{P} \\ a_3^{P} \\ a_4^{P} \end{Bmatrix}, a_i^{P} = \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} \tag{3.29}$$

The element stiffness matrix can be written as

$$K^{p} = \begin{bmatrix} k_{11}^{p} & k_{12}^{p} & k_{13}^{p} & k_{14}^{p} \\ k_{21}^{p} & k_{22}^{p} & k_{23}^{p} & k_{24}^{p} \\ k_{31}^{p} & k_{32}^{p} & k_{33}^{p} & k_{34}^{p} \\ k_{41}^{p} & k_{42}^{p} & k_{43}^{p} & k_{44}^{p} \end{bmatrix} \tag{3.30}$$

Furthermore, the plate bending situation under local coordinates cannot be ignored, and the stiffness equation is:

$$F^{\mathrm{b}} = K^{b} a^{b} \tag{3.31}$$

Similarly, the nodal force vector F^{b} and the nodal displacement vector a^{b} are:

$$F^{b} = \begin{Bmatrix} F_1^{b} \\ F_2^{b} \\ F_3^{b} \\ F_4^{b} \end{Bmatrix}, F_i^{b} = \begin{Bmatrix} W_i \\ M_{xi} \\ M_{yi} \end{Bmatrix}; a^{b} = \begin{Bmatrix} a_1^{b} \\ a_2^{b} \\ a_3^{b} \\ a_4^{b} \end{Bmatrix}, a_i^{b} = \begin{Bmatrix} \omega_i \\ \theta_{xi} \\ \theta_{yi} \end{Bmatrix} \tag{3.32}$$

The element stiffness matrix can be written as

$$K^b = \begin{bmatrix} k_{11}^b & k_{12}^b & k_{13}^b & k_{14}^b \\ k_{21}^b & k_{22}^b & k_{23}^b & k_{24}^b \\ k_{31}^b & k_{32}^b & k_{33}^b & k_{34}^b \\ k_{41}^b & k_{42}^b & k_{43}^b & k_{44}^b \end{bmatrix} \tag{3.33}$$

The stiffness matrix for the plate bending is a 12 x 12 matrix and the stiffness matrix for the plane stress action is an 8 x 8 matrix. Thus the stiffness matrix of a rectangular shell element is a 20 x 20 matrix based on the combination of the above two matrices. However, in a global coordinate system, each node has 6 degrees of freedom in each element. Thus the local stiffness matrix need to be expanded considering the influence of angle displacement of θ_{zi}, (Shouyi, 2005), which expand the element stiffness matrix to a 24 x 24 matrix. Moreover, the equilibrium equation of flat shell elements is:

$$F^e = K^e a^e \tag{3.34}$$

The next step is the establishment of global stiffness matrix. The transform equation for nodal displacements for a 4-node shell element is:

$$a^e = T\overline{a^e} \tag{3.35}$$

where the transformation matrix T is:

$$T = \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix}, \lambda = \begin{bmatrix} \phi & 0 \\ 0 & \phi \end{bmatrix}, \phi = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{3.36}$$

where φ is the angle between the local z axis and the $\bar{z}$ axis in global coordinate systems.

The nodal forces are transformed similarly as the displacements

$$F^e = T\overline{F^e} \tag{3.37}$$

The global equilibrium equation for the shell element is given as

$$\overline{F^e} = \overline{K^e}\,\overline{a^e} \tag{3.38}$$

where:

$$\overline{K^e} = T^T K^e T \tag{3.39}$$

3.6 Calculation of wind load

The wind influences on structures have already been defined by Eurocode 1 published in the year 1991. This series of Eurocodes were established because of the efforts pushed by the Commission of the European Community to eliminate the technical obstacles and harmonize technical specifications, (European Committee for Standardisation, 1991).

According to Eurocode 1, the calculations of wind velocities and wind pressures works on the basis of mean wind velocity V_m and basic wind velocity V_b. The basic wind velocity V_b is calculated according to the characteristic of 10 minutes mean wind velocity, irrespective of the direction of winds and time of year, at 10 meters above ground level in open country terrain as well as low vegetation and isolated obstacles.

With the data of basic wind velocity V_b, the calculation equation of mean wind velocity V_m of distinguished height z is:

$$V_m(z) = C_r(z) \cdot C_0 \cdot V_b \tag{3.40}$$

where C_0 is orography factor, given from National Annex, and $C_r(z)$ is roughness factor, defined by the following equations:

$$C_r(z) = \begin{cases} k_r \cdot \ln\left(\frac{z_{\min}}{z_0}\right), z \le z_{\min} \\ k_r \cdot \ln\left(\frac{z}{z_0}\right), z_{\min} < z \le z_{\max} \end{cases} \tag{3.41}$$

The terrain factor k_r is defined according to the following equation:

$$k_r = 0.19\left(\frac{z_0}{z_{0,\Pi}}\right)^{0.07} \tag{3.42}$$

where the roughness height z_0 and the minimum height z_{min} are defined in Table 1. The maximum height z_{max} is taken as 200m, (European Committee for Standardisation, 1991).

Table 1: Terrain categories and terrain parameters, (Eurocode 1 1991).

Terrain category	Z_0	Z_{min}
0 Sea or coastal area exposed to the open sea	0.003	1
Ⅰ Lakes or flat and horizontal area without obstacles	0.01	1
Ⅱ Area with low vegetation such as grass and isolated obstacles with separations of at least 20 obstacle heights	0.05	2
Ⅲ Area with regular cover of vegetation or buildings or with isolated obstacles with separations of maximum 20 obstacle heights	0.3	5
Ⅳ Area in which at least 15 % of the surface is covered with buildings and their average height exceeds 15 m	1.0	10

From the equations of mean wind velocity V_m, it is obvious that the relationships between the height z and mean wind velocity V_m is an exponential function. Furthermore, having the data of mean wind velocity V_m it is possible to calculate the characteristic peak velocity pressure $q_p(z)$ by

$$q_p(z) = \left(1 + 7I_v(z)\right) \cdot 0.5\rho \cdot V_m(z)^2 \tag{3.43}$$

where ρ means the density of the air, $I_v(z)$ is the turbulence intensity, which is defined according to the height as well:

$$I_v(z) = \begin{cases} \dfrac{\sigma_v}{V_m(z_{\min})}, z \le z_{\min} \\ \dfrac{\sigma_v}{V_m(z)}, z_{\min} < z \le z_{\max} \end{cases} \tag{3.44}$$

Moreover, the standard deviation of the turbulence σ_v is calculated by

$$\sigma_v = k_r \cdot V_b \cdot k_l \tag{3.45}$$

where the terrain factor k_r is acquired in equation (3.42) and the turbulence factor k_l is given in the National Annex.

To calculate the wind pressure on buildings, two main methods are mentioned in Eurocode 1, one using wind pressure coefficients and one using wind force coefficients. By using wind pressure coefficients, the wind pressure is presented as follows:

$$F_{w.p}(z) = c_{pe} \cdot q_p(z) \tag{3.46}$$

where the characteristic peak velocity pressure is given from equation (3.39) and the total outside pressure factors c_{pe} is the sum of the windward pressure factor $c_{pe.10D}$ and leeward pressure factor $c_{pe.10E}$ as

$$c_{pe} = (c_{pe.10D} + c_{pe.10E}) \tag{3.47}$$

These wind pressure factors $c_{pe.10D}$ and $c_{pe.10E}$ are both defined in Table 2 in Eurocode 1.

Table 2: External pressure coefficients for vertical walls, Eurocode 1

Zone	A		B		C		D		E	
h/d	$c_{pe.10}$	$c_{pe.1}$	$c_{pe.10}$	$c_{pe.1}$	$c_{pe.10}$	$c_{pe.1}$	$c_{pe.10}$	$c_{pe.1}$	$c_{pe.10}$	$c_{pe.1}$
5	-1.2	-1.4	-0.8	-1.1	-0.5		0.8	1.0	-0.7	
1	-1.2	-1.4	-0.8	-1.1	-0.5		0.8	1.0	-0.5	
≤ 0.25	-1.2	-1.4	-0.8	-1.1	-0.5		0.7	1.0	-0.3	

The other method is more complicated because two more parameters need to be calculated i.e. the structural factor $c_s c_d(z_s)$ and the shape factor for forces c_f.The equation used in this method is

$$F_{w.f}(z) = c_s c_d(z_s) \cdot c_f \cdot q_p(z) \tag{3.48}$$

The calculation of the factors $c_s c_d(z_s)$ and c_f will not be mentioned here since this method is not implemented in this work.

3.7 Non-Linear

There are three sources of nonlinearity in simulations of structural mechanics: material nonlinearity, geometric nonlinearity and boundary nonlinearity. The most common one is the material nonlinearity. Most metals present good linear stress-strain relationship for small strains and nonlinear yield conditions and irreversible strains for increased strains, see Figure 10 *(a)*.

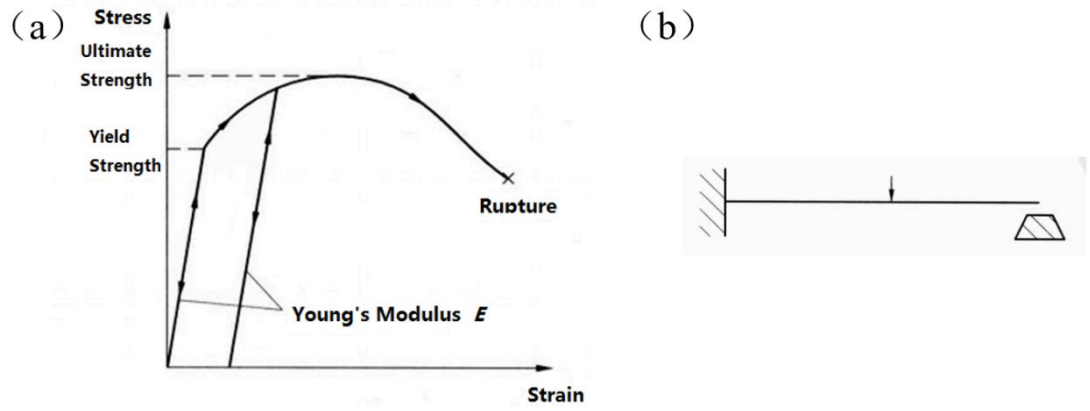

Figure 10: (*a*) Typical stress-strain curve for metals; (*b*) A beam with nonlinear boundary condition.

If the boundary condition varies during the analysis, boundary nonlinearity might occur, (Korte., V.Boel, & W.D.Corte, 2014). Figure 10*(b)* shows a beam that would curve as linear cantilever beam until the right end encounter the obstacle. The beforehand bending can be regarded as linear for small curvature but when the right end touch the obstacle the boundary condition changes and the nonlinear response would occur.

The third resource of nonlinearity is geometric nonlinearity, it occurs when thin structures are influenced by initial imperfections. Lateral buckling of long columns, frame structures or thin plates are examples of typical nonlinear stability problems.

4. Methodology

Simulation with the finite element method to study deformation and stress behaviors in multi-storey timber building is the main purpose of this thesis. Special focus is on effective modelling of mechanical connections to study how the connection stiffness has influence of the global structural behavior of the building., This work is intended to give engineers and researchers better understanding of how multi-storey timber buildings behaves during wind loading as well as be aware of the potential critical connection points in the building.

4.1 Geometry of the structural object

One primary step of this work is to create the geometric structure of the studied building that will be simulated. The geometry is based on the previous constructed multi-storey timber frame building in Bergen, Norway, (Abrahamsen RB,Malo, 2014). This is a common and widely used geometry structure for multi-storey timber buildings around in the world. To study different geometric solutions (e.g. different locations of columns and diagonals) a parametrized scripting is used to create the model geometry. More flexible geometry changes would be possible to study in further studies.

4.2 Variable input parameters

This thesis intends to study how numbers of variable input parameters have influence on overall structural behavior of a multi-storey timber frame building. The parameters in question are dimensions of the building, number of floors, number and locations of diagonals and vertical columns, material properties, cross-sectional properties, boundary conditions and the loading of the structure.

4.3 Modeling

In order to simplify the simulation process, the timber structure is modeled by three-dimensional beam and shell elements. Considering that the modeling would fulfill the reality, the vertical columns and the diagonals can be located differently by using variable input parameters. To connect the timber elements together, elastic spring elements and short beam elements (to connect the dowel groups) are used to simulate the slot-in plate connections. Each spring element consists of six local degrees of freedom, i.e. three slip and three rotational degrees of freedom. Some of the stiffness values for the spring elements are based on simple hand calculations of mechanical dowel groups. Figure 11 shows schematic picture of some spring connections used in the model.

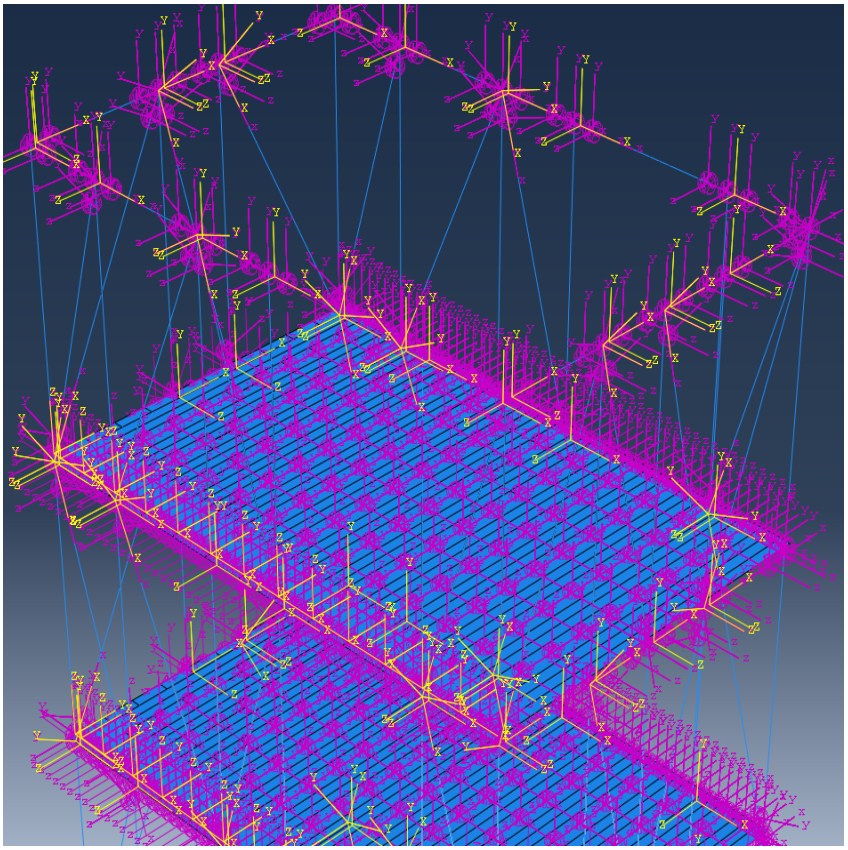

Figure 11: Illustration of spring connections used in a model

The focus of the thesis is to study the relationship between variable input parameters and the overall structural behavior of the timber frame structure. A parametrized Python scripting is introduced in order to simplify creation of model. The input parameters are e.g. all used material parameters, the dimensions of the building, and even the number and locations of the columns and diagonals. The detailed script files for the model can be found in Appendix 2.

The beam element used in the model is B32 (Timoshenko beam type) meaning that shear deformations are also taken into account. Element S4 is used as shell element also because of the shear deformations.

4.4 Simulations in ABAQUS

A primary step before your modelling starts is to define the system of units you are going to use. ABAQUS has no build-in systems for units. Therefore the users always

need to pay attention on the units so all outputs in ABAQUS become in consistent unit system. Some common unit systems are shown in Table 3.

Table 3: Different unit systems

Quality	SI Unit (m)	SI unit (mm)	US unit (ft)	US unit (inch)
Length	m	mm	Ft	in
Force	N	N	Lbf	lbf
Mass	Kg	Tonne($10^3 kg$)	Slug	$lbf \cdot s^2 / in$
Time	S	s	s	s
Stress	Pa(N/m^2)	MPa(N/mm^2)	lbf/ft^2	psi(lbf/in^2)
Energy	J	mJ($10^{-3} J$)	$ft \cdot lbf$	$in \cdot lbf$
Density	kg/m^3	$tonne/mm^3$	$slug/ft^3$	$lbf \cdot s^2 / in^4$

The unit system used in ABAQUS in this thesis is the International System of Units (SI unit).

4.41 Beam element used in ABAQUS

The beam element used in ABAQUS is assumed to have a flat plain cross-section which becomes perpendicular to the axis of the beam when it deforms. ABAQUS provides several types of cross section profiles that the user needs to define. In addition to these defined cross section shapes, the user is allowed to design an arbitrary shape of the cross section through 'ARBITRARY' option in ABAQUS, (SIMULA).

The properties of the beam element are defined through the numerical integration of the section, or, through the area, the moment of inertia and the reverse forms. The orientation of the cross-section must also be defined in ABAQUS, (Miao,Z& Xiaochuan,Y& Jianhui, L, 1991).

In ABAQUS, axial deformation, bi-axial bending and torsional deformation can occur for each beam element. In addition, Timoshenko beam element also considers the effects of transverse shear deformations.

4.42 Shell element used in ABAQUS

ABAQUS provide two types of shell elements, the normal shell and continuum-based shell. For the normal shell element the thickness is defined through the definitions of

cross section properties. Unlike normal shell elements, the continuum-based shell is similar to three-dimensional solid element, it is established through discretization of the whole object which establish its mathematical descriptions, (Miao,Z& Xiaochuan,Y& Jianhui, L, 1991).

For a simulation of contact problems, the continuum-based shell works better as it is able to provide the variable thickness, however, for a thin-shell problem, the normal shell elements works better.

4.43 The iterations balance and convergence in ABAQUS

Before discussing what the convergence and iteration balance is, some words about the modules 'step', 'load increment' and 'iteration' need to be explained previously as well as the relationships between each of these modules.

When simulating in ABAQUS the user can define loading, boundary conditions in the load module. The loads are applied on the structure during time increments defined in the step module. Since the total load is ramped (through small load increments) on the structure, ABAQUS is capable to simulate non-linear structural behavior that occurs during the step time. Moreover, after each load increment, the structure must show an approximate equilibrium state. The simulated results after each load increment are then possible to be visualized as output for so-called frames, (Miao,Z& Xiaochuan,Y& Jianhui, L, 1991).

An equilibrium iteration process is a probe to control static equilibrium of the structure in the end of each load increment, however, if the structure does not is in equilibrium state after the first equilibrium iteration, ABAQUS would process a new iteration until the structure will achieve an equilibrium state. Once the equilibrium state is achieved, one load increment is finished and then it is possible to output the numerical results for this load increment.

For any load increase ΔP, the nonlinear response looks like the diagrams in Figures 12 and 13, first ABAQUS tries to calculate the displacement correction c_a by using the initial stiffness matrix K_0 and the load increment ΔP.The initial stiffness K_0 is based on the initial displacement u_0.The new displacement u_a can be calculated by adding the displacement correction c_a and the initial displacement u_0.

Thereafter, a new stiffness matrix K_a can be calculated, and new internal forces I_b can also be acquired. ABAQUS will calculate the difference between P and I_i, until

the out of balance force R_i will be small enough to be regarded as equilibrium state. This is called that the solution convergence, otherwise ABAQUS would try to process new equilibrium iteration.

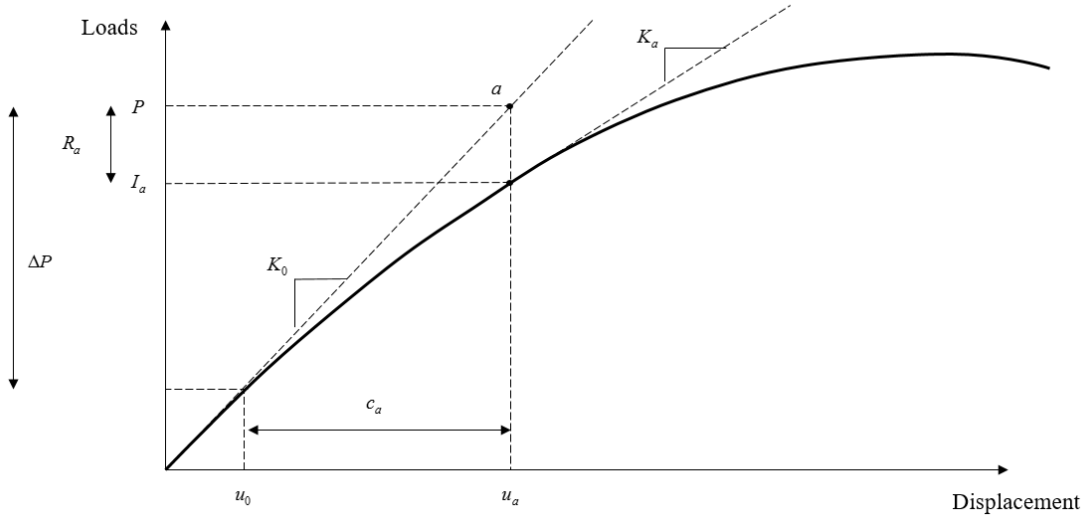

Figure 12: The first iteration in one load increment

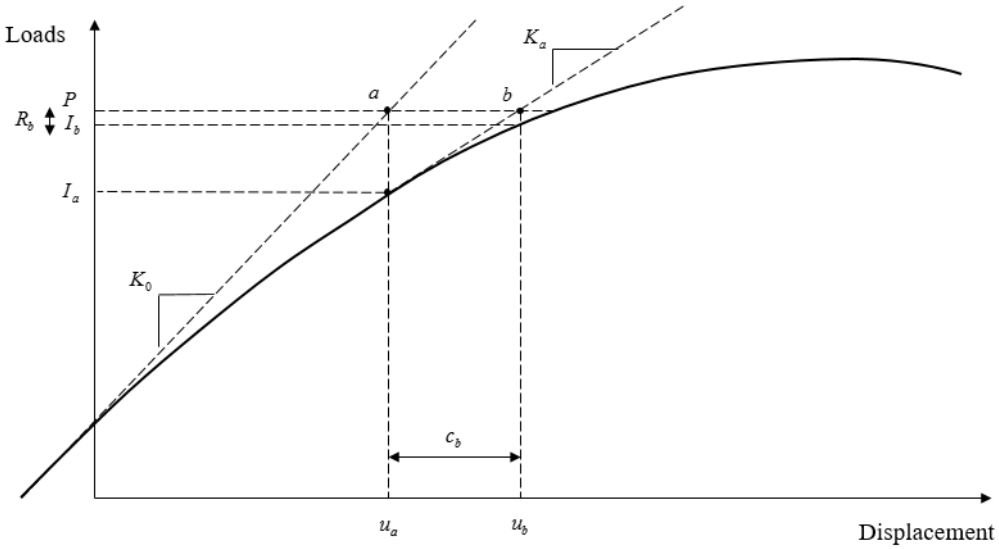

Figure 13: The second iteration

4.5 Load calculation

Horizontal wind load

As mentioned in the introduction, the thesis is not only dealing with vertical loads, such as gravity and loads from rain or snow, but also considering horizontal load from

wind. Therefore it is necessary to create the wind load that varies with the height of the building. According to Eurocode 1, the wind pressure is varying as exponential function with the height of the building. Since the size of the building is changeable in the model, the calculated exponential function also is it. Thus another script is introduced to calculate the wind load. According to Chapter 3.6 the following wind calculation is based on a 15x10x60 m building which is located on a terrain of type II and with a basic wind velocity of 24 m/s. The detailed script can be found in Appendix 2. The calculated exponential functions of the mean wind velocity $V_m(z)$ and the peak velocity pressure $q_p(z)$ is shown in Figure 14.

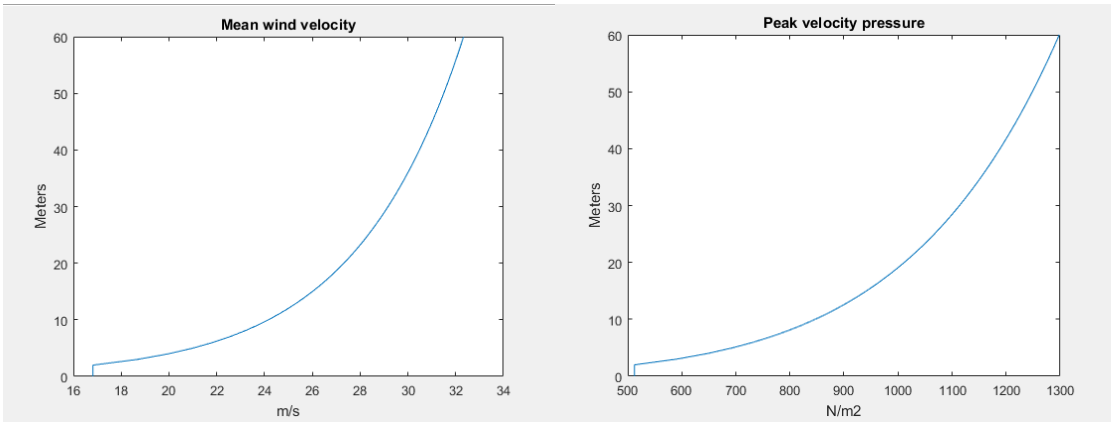

Figure 14: Height variation in mean wind velocity (left) and peak velocity pressure (right).

Based on the peak velocity pressure the wind pressure is calculated according to the equation (3.46) as

$$F_{w.p}(z) = c_{pe} \cdot q_p(z) \tag{4.5}$$

The variation in wind pressure over height is shown in Figure 15.

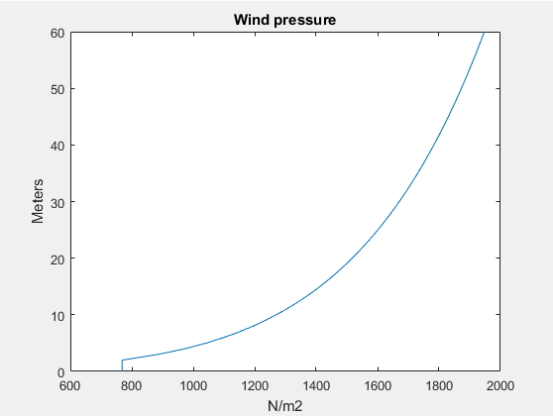

Figure 15 : Wind pressure based on a square building with dimensions 15x10x60 m.

However, the wind pressure acquired from the calculations can´t be arranged in the model directly since the used element type is a beam element which is not able to be loaded with pressure. To overcome this obstacle, the writer assumed that the wall pressure is converted to the vertical columns as a distinguished line load suitable for the beam elements. The horizontal wind load distribution used is shown in Figure 16 *(a)*.

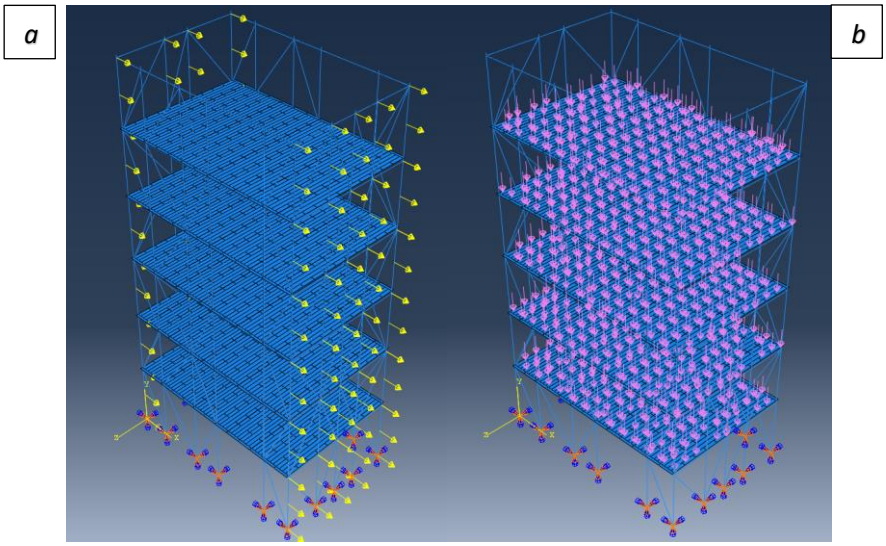

Figure 16: *(a)* Distribution of wind load on the vertical columns; *(b)* Distributions of self-weight pressure on the floor slabs.

The vertical load is caused by self-weight and imposed load. The self-weight of the structure was replaced with an external downward pressure applied on the floor slabs, see Figure 16 *(b)*. The total self-weight of the building is averagely divided with the number of floors, and thereafter the divided forces acting on each slab is transferred to

a pressure on the floor. All the calculations and assignments of the loads are automatically done by a script which is shown in detail in Appendix 2.

5. Results and analysis

This chapter presents how the parametrized Python script is used to create the numerous FE-models for multi-storey timber frame buildings based on the distinguished input parameters. Moreover the parameterized FE-models are discussed and analyzed when facing the horizontal loads of wind. Some numerical results are discussed in this chapter as well.

5.1 Parametrized scripting

As mentioned in chapter 4, the Python scripts are parameterized both regarding the model size and several other parameters. In the following numerous figures are used to present how the scripts and he corresponding FE-models work.

Figure 17 shows the input parameters used to create a FE-model of 6 storey timber frame building with a 30 meters height by setting the 'Height_of_Truss = 30' and 'Number_of_Layar = 5'. Based on these two parameters and several geometry parameters for location of the columns and diagonals the script automatically calculates the 3D coordinates for all the timber elements and assembly them together as shown in Figure 17.

```
26  Number_of_Column_in_C_plane=4
27  Number_of_Column_in_D_plane=2
28
29  Height_of_Truss=30*(10*Scale_factor)# m
30  Number_of_Layer=5
31
32  Dist_bet_colum_of_side_1=[2.4,3.6,2.4,4.8
33  Dist_bet_colum_of_side_2=[3,2,2,3,0,0,0,0
34  Dist_bet_colum_of_side_3=[2.4,3.6,2.4,4.8
35  Dist_bet_colum_of_side_4=[4,2,4,0,0,0,0,0
36  ##########################################
37  #    ATTENTION!!!!!!!: The Total Length o
38  ##########################################
39
40  #Define where to put Diagonals
41  Dia_A_L=[0,1,4,7,10,11,14,17,20,21,24,27]
42  Dia_A_R=[2,5,6,9,12,15,16,19,22,25,26,29]
43  Dia_B_L=[0,3,5,6,8,11,13,14,16,19,21,22]
44  Dia_B_R=[1,2,4,7,9,10,12,15,17,18,20,23]
45  Dia_C_L=[0,1,4,7,10,11,14,17,20,21,24,27]
46  Dia_C_R=[2,5,6,9,12,15,16,19,22,25,26,29]
47  Dia_D_L=[0,1,5,6,11,12,16,17]
48  Dia_D_R=[2,3,7,8,9,13,14,15]
49  #
```

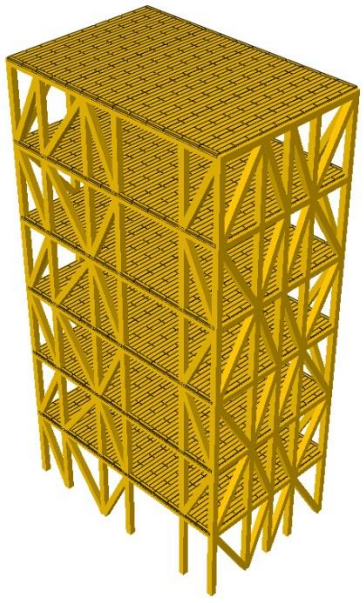

Figure 17: Input parameters used in the script to define the parametrized geometry of the studied multi-storey timber frame building.

The parameter matrices ‘Dist_bet_column_of_side_i’ and ‘Dia_j_k’ are used in the script to define distances between the columns and positions of the diagonals. It may be noted that only one diagonal can be located within each square made by two columns and two slabs. It is even possible to have squares without diagonals or no diagonals at all as shown in Figure 18. However, the lacks of diagonals would significantly reduce the global bending stiffness of the structures comparing to buildings with diagonals, which would be discussed in chapter 5, section 2.

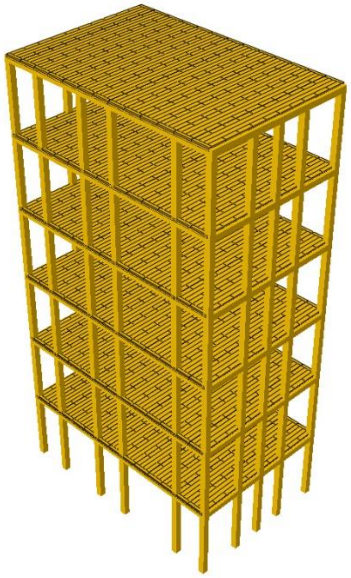

Figure 18 : Geometry of timber frame building without diagonal elements.

The parametrized scripting allows us to create arbitrary geometries of this type of multi-storey timber frame building in a simple and effective way.

5.2 Influences of spring stiffness, cross-sectional dimensions and diagonals on the global bending stiffness of the building

As mentioned in Chapter 3 and 4, the connections between the timber elements are simulated by special springs between the elements. Each spring element consists of six degrees of freedom (three slip and three rotational degrees of freedom).

5.21 Rotational spring stiffness

To study the influence of rotational spring stiffness on the global bending stiffness behavior of the structure, the symmetrical structure shown in Figure 19 was forced

(prescribed) to 0.5 meters horizontal displacement at the top of the building. This was done for two different stiffness situations, one for a semi-rigid connections with rotational stiffness and one for totally pinned connections. Afterwards, the reaction forces were selected. The reaction forces are shown in Figure 20.

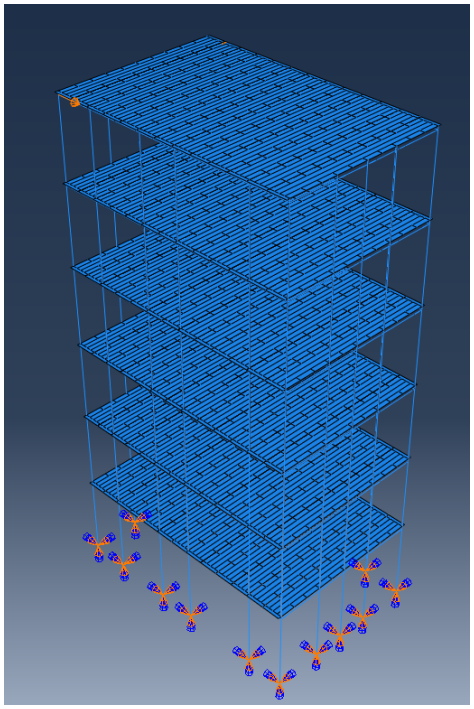

Figure 19: Symmetrical structure without diagonals.

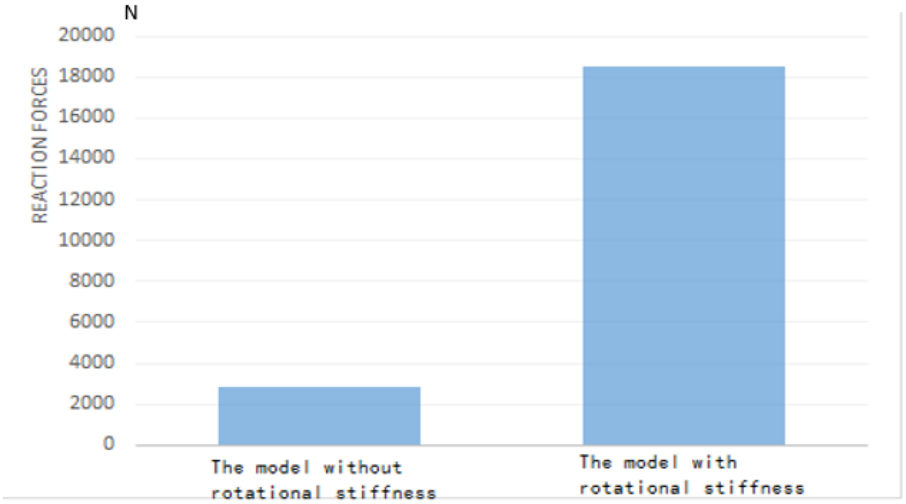

Figure 20: Reaction forces caused by prescribed horizontal displacement of 0.5 m at the top of the building.

The results show a large difference between the global stiffness of these two structures. For the structure without rotational stiffness only 2800N was required whereas the

required force was 18500N for the moment stiff structure. This shows that the rotational stiffness of the connections is significantly important in structural designs of multi-storey timber buildings.

Although the results in Figure 20 shows that the importance of the rotational stiffness of the connections, it is still not clear how would rotational stiffness influence the global bending stiffness. Therefore, numerous more analysis are made.

For the structure shown in Figure 19, numerous reactions forces caused by the horizontal top displacement were studied for different spring stiffness, Figure 21 shows the relationship between rotational spring stiffness and the horizontal reaction force at the top of the structure.

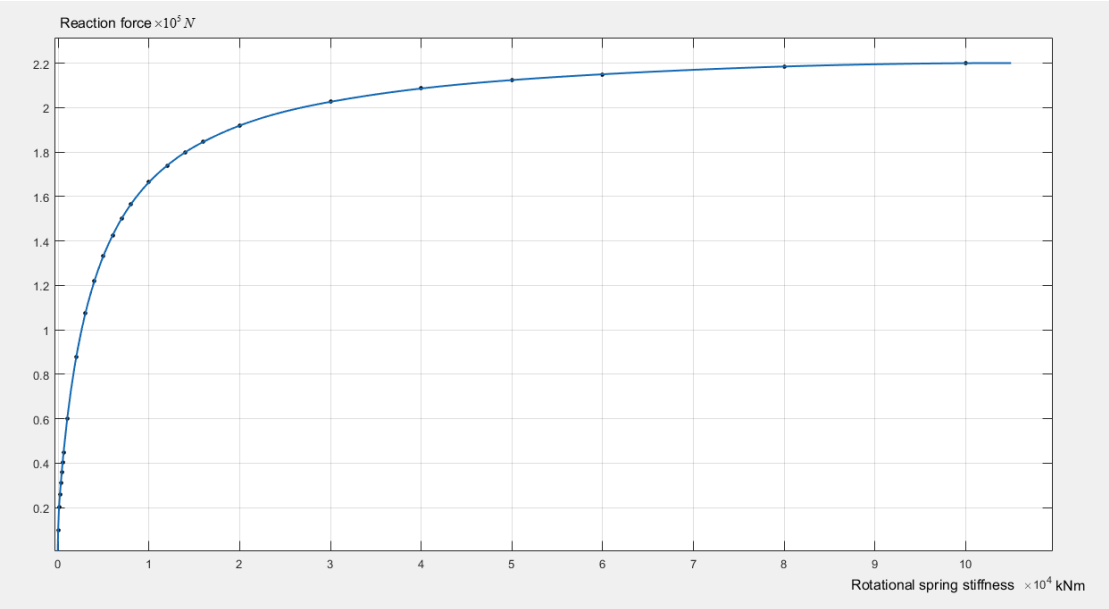

Figure 21: The relationship between (in plane) rotational spring stiffness and the horizontal reaction force at the top of the building.

The figure presents a nonlinear behavior of the global bending stiffness of the building. The plot shows the bending stiffness increase sharply up to rotational spring stiffness value of 10000 kNm. Afterwards the reaction force increase slowly until it becomes constant for large stiffness values. When it reached 60000 kNm, the slope has been very small meaning that further increase in the spring stiffness would not help to increase the global bending stiffness of the building.

5.22 Cross-sectional dimensions

From Figure 21, the relationship between rotational spring stiffness and the global

bending stiffness shows a non-linear relation. However, the convergent limit value for the reaction force would also changes by changing the cross-sectional dimensions. Figure 22 shows comparison of global bending stiffness between two buildings with two different cross-sectional dimensions.

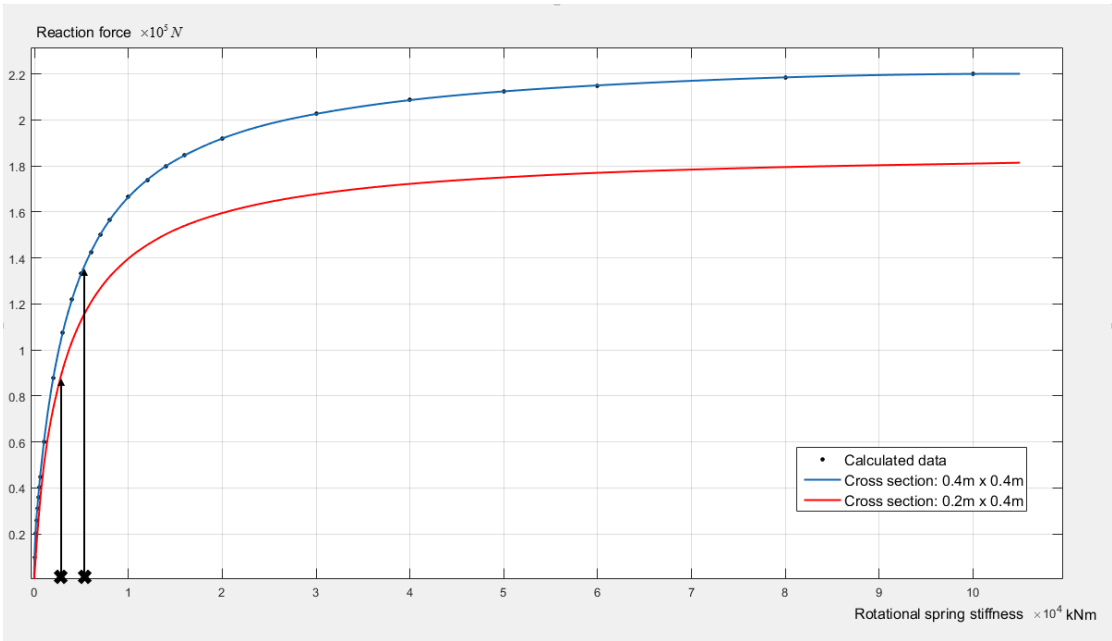

Figure 22 Comparison of global bending stiffness for two buildings with two different cross-sectional dimensions.

From Figure 22, it is obvious that the cross-sectional area has an influence on the global bending stiffness. For a structure with 0.4x0.4 m cross section area, the maximum reaction force is about 200 kN. Whereas for a cross section of 0.2x 0.4 m the maximum reaction force is 181 kN. The next question is what would be a realistic rotational spring stiffness of a typical slotted-in steel plate connection used for these sizes of cross sections. To address this, the rotational stiffness of an 18 dowel connection shown in Figure 23 has been calculated according to Eq. 3.20 and shown as located points on the curves in Figure 22.

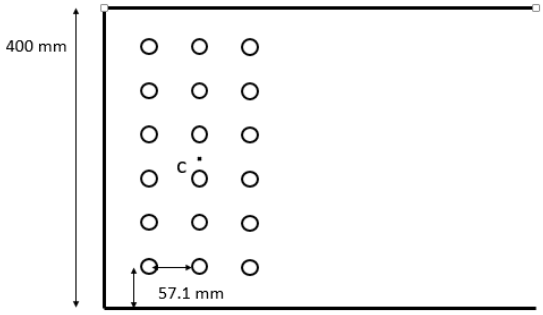

Figure 23: A geometry of an 18 dowel connection.

According to equation 3.20, the rotational stiffness of the 18 dowel connection can be calculated as

$$K_r = 2K_{ser}\sum_{j=1}^{9} {r_j}^2$$

According to Eurocode 5 the slip modulus K_{ser} is given by

$$K_{ser} = \frac{\rho^{1.5}}{23} \times d = \frac{420^{1.5}}{23} \times 12 = 4491\ N/mm$$

where r_j is the radial distance from the center point C of the dowel group to the center of each dowel, ρ is the density of the wood material and d is the dowel diameter. The geometry term for the studied connection is $\sum_{j=1}^{9} r^2$ = 105285 mm^2

For connections with two respective three slotted in steel plates the rotational stiffness of the connection are:

$$K_r = 2K_{ser}\sum_{j=1}^{9} {d_j}^2 = 2 \times 4491\frac{N}{mm} \times 105285\ mm^2 = 946kNm$$

And for the cross section of 0.4x 0.4 m, 6 cuts of the dowel connections can be assembled, it means the total rotational stiffness is 6 × 946 *kNm*= 5676 *kNm.* However, for a cross section of 0.2x 0.4 m, only 4 cuts could be assembled, which caused the total rotational stiffness became 4 × 946 *kNm*= 3784 *kNm.* All the two calculated results could be found in Figure 22.

5.23 Diagonals

After the analysis of the rotational stiffness, the influence of number of diagonal on

the global bending stiffness was studied. By adding more diagonals to the previous structure the relationship is plotted in Figure 24.

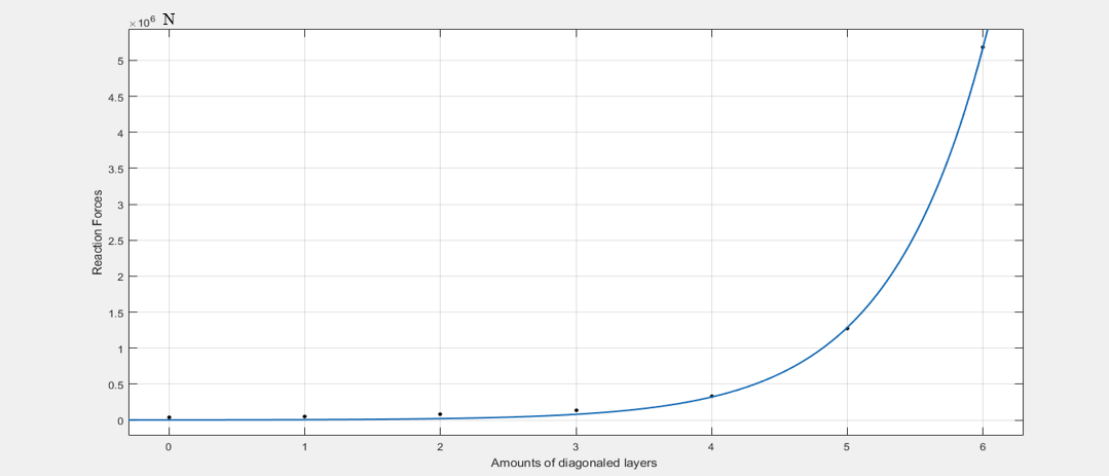

Figure 24: The relationship between reactions force and numbers of diagonalized stories

From the plot, it can be acquired that the existence of diagonals increased the reaction forces, which also means the global bending stiffness of the structure. Furthermore, the existence of the diagonals in lower part of the structure provides small increase of the global stiffness. For buildings with more than three diagonalized stories the global bending stiffness presents an exponential increases when more diagonals were added.

As previously mentioned, the existences of diagonals significantly impact the global stiffness of the structures. However, no study has been done for the real structure. In order to simulate the influences of diagonals on global bending deformation of the structures, two more simulations were performed with same sizes and material properties, however, one model had numerous diagonals whereas the other had none diagonal at all. Both the structures were loaded with same wind pressure on the left side of the building. Figure 25(*a*) shows the deformed structure with diagonals and Figure 25(*b*) shows the structure without diagonals.

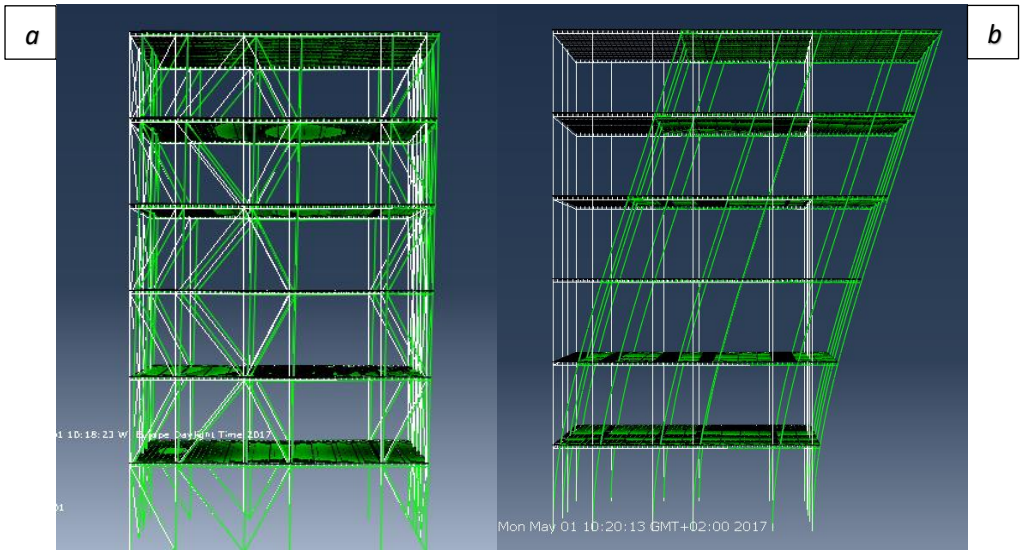

Figure 25: Deformed frame structures *(a)* with diagonals and (*b*) without diagonals.

From the deformation plots in Figure 25, it is obvious that the existence of diagonals have large impact on the global bending stiffness and deformations of the structure. Figure 26 shows how the node displacement varies along the height of the structure. All the nodes are located on the same column.

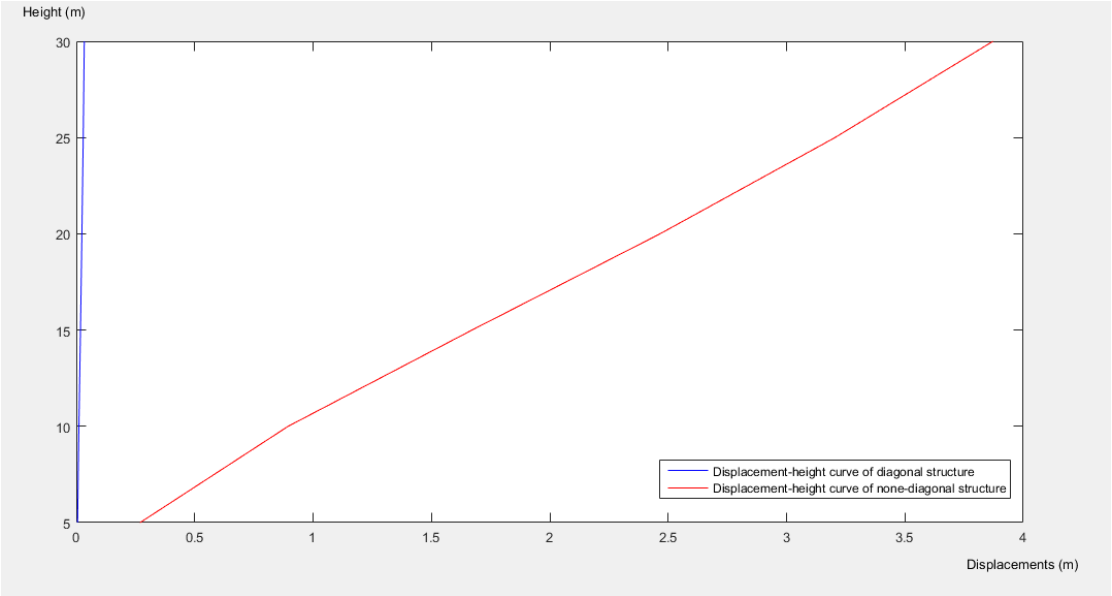

Figure 26: Variation in horizontal displacements with increase in height.

The results in Figure 26 show the nodal displacements to increase with increase in height. It presents approximately linear increase even though the wind load has an exponential growth. Moreover, the maximum node displacement for a structure with numerous diagonals is approximately 0.06 m whereas the maximum node displacement for the none-diagonal structure is 15.5 m. This indicates clearly the

importance of diagonal in such multi-storey structure.

5.3 Simulations of different frame structures using parameterized modelling

As mentioned earlier the Python scripts created in this work allows the user to create different frame structures in a simple way by using the input parameters described in section 5.1. In this section, several analyses are performed to illustrate the variety and flexibility of the parameterized modelling.

5.31 Buildings with different heights

Three different frame structures with total heights of 30, 40 and 50 m were created see Figure 27. All other input parameters are unchanged except the number of floors which increases according to the height of the structure. All the buildings have been loaded with the horizontal (and exponential distributed) wind load described in section 4.5.

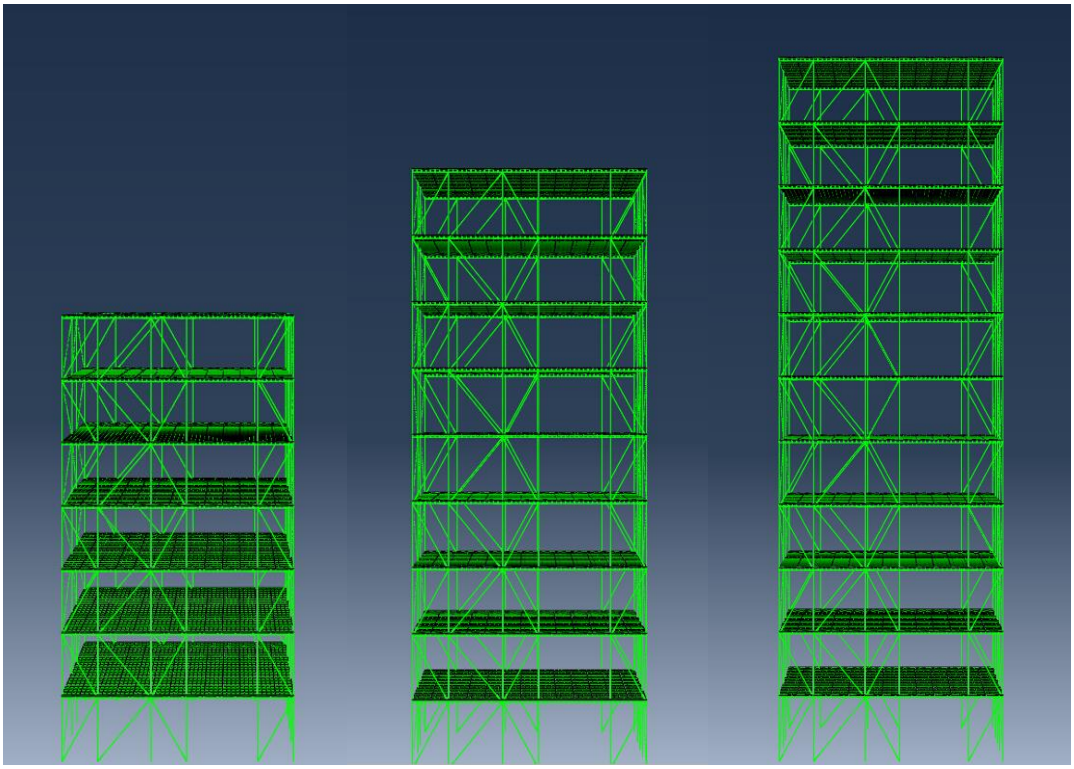

Figure 27: Three frame structures with total height of 30, 40 and 50 m

Figure 28 presents relationship between the maximum horizontal nodal displacements and the height of structures. The plot shows an exponential increase in nodal displacements with increased structural height. This means that high buildings are more sensitive for increase in height (because of the nonlinear wind loading) than lower buildings are. ,

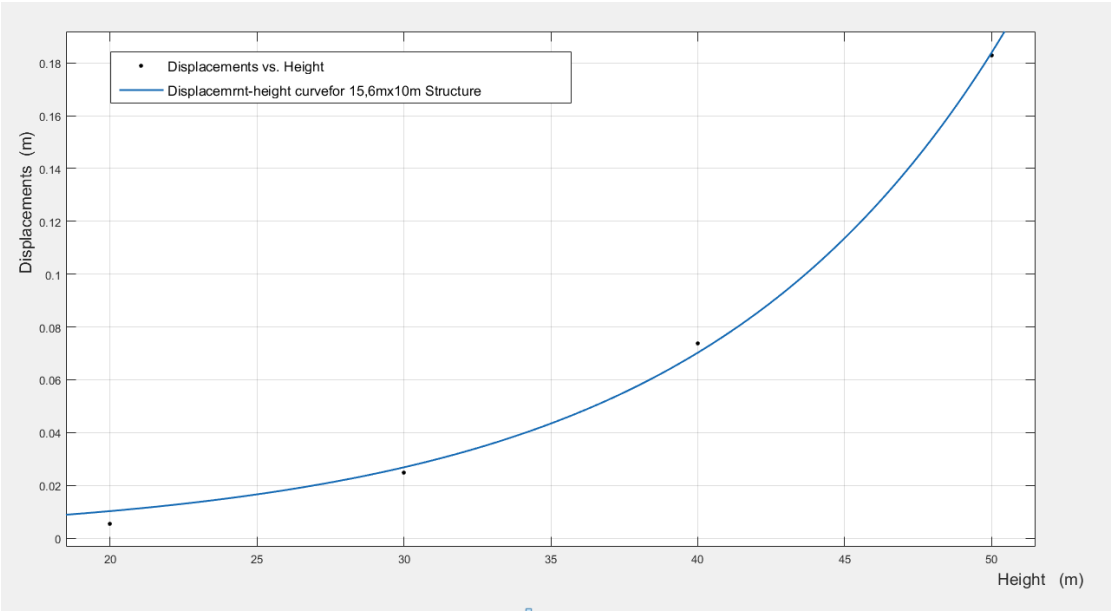

Figure 28: Maximum horizontal nodal displacement at the top of the structures when they are exposed to same wind loading.

The width and depth of the structure are also parameters in the script that easily can be changed. Figure 29 presents similar displacement-height curves as shown in the previous figure for three buildings with different width/depth ratios.

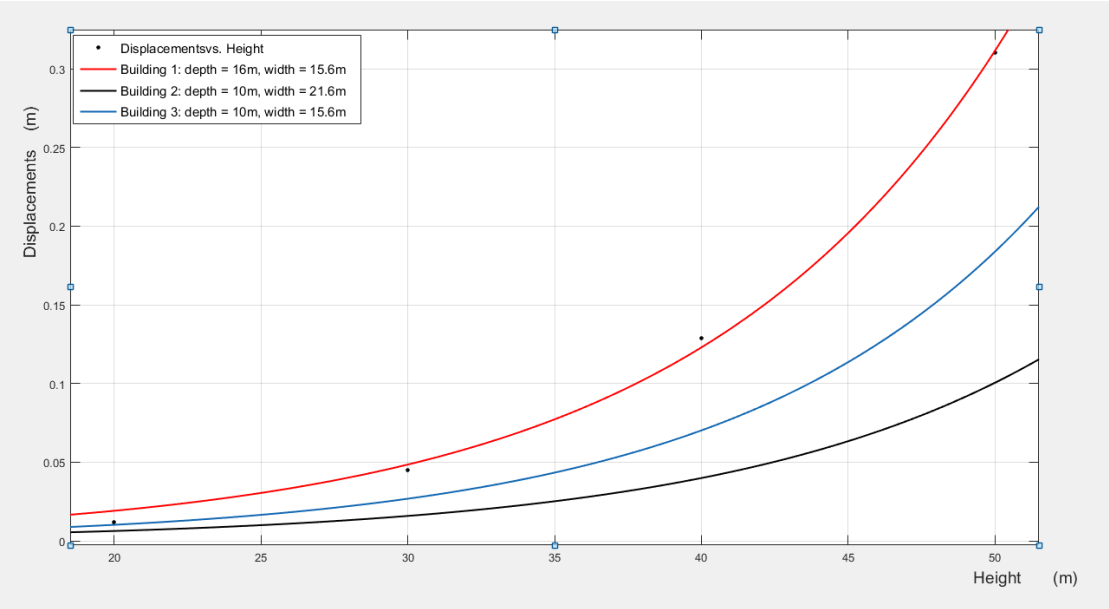

Figure 29: Displacement-height curves for three 55 meter buildings with different width/depth ratios.

Figure 29 shows a significant decrease in the maximum nodal displacement for increased width dimension. However, the efforts to increase the depth of the frame structure showed an opposite results, the maximum nodal displacement significantly increased. It means although that the increase in global stiffness is increasing slower with increased house depth than the expanded wind effect.

5.4 Comparison between symmetrical and unsymmetrical multi-storey structures

To study structural behavior of symmetrical and unsymmetrical multi-storey buildings, Figure 31 presents two frame structures, one unsymmetrical 55 m timber frame structure and one symmetrical of same length.

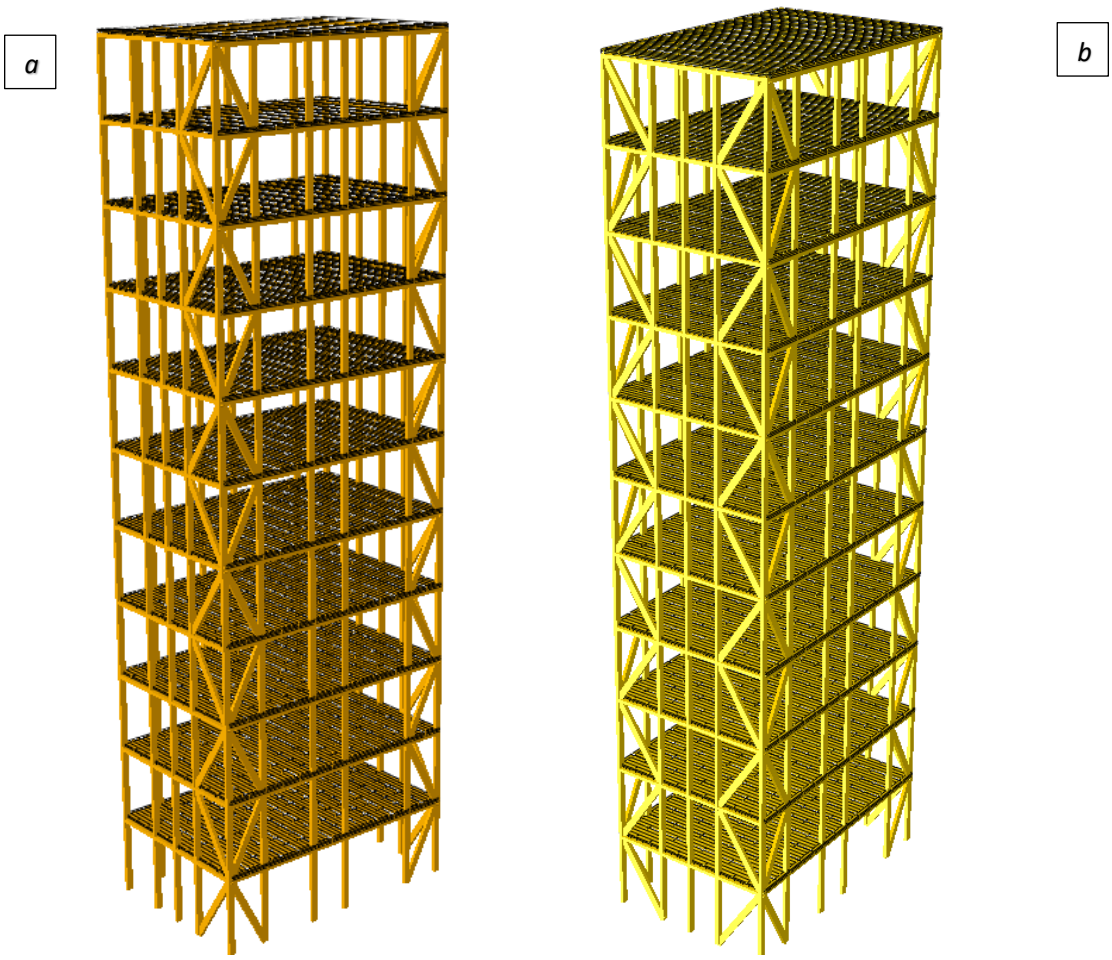

Figure 30: 55 m high timber frame structures; (*a*) unsymmetrical geometry and (*b*) Symmetrical geometry.

The same horizontal wind load is applied on both the buildings. Figures 32 and 33 show the deformations of the unsymmetrical respective the symmetrical timber structures caused by the wind load.

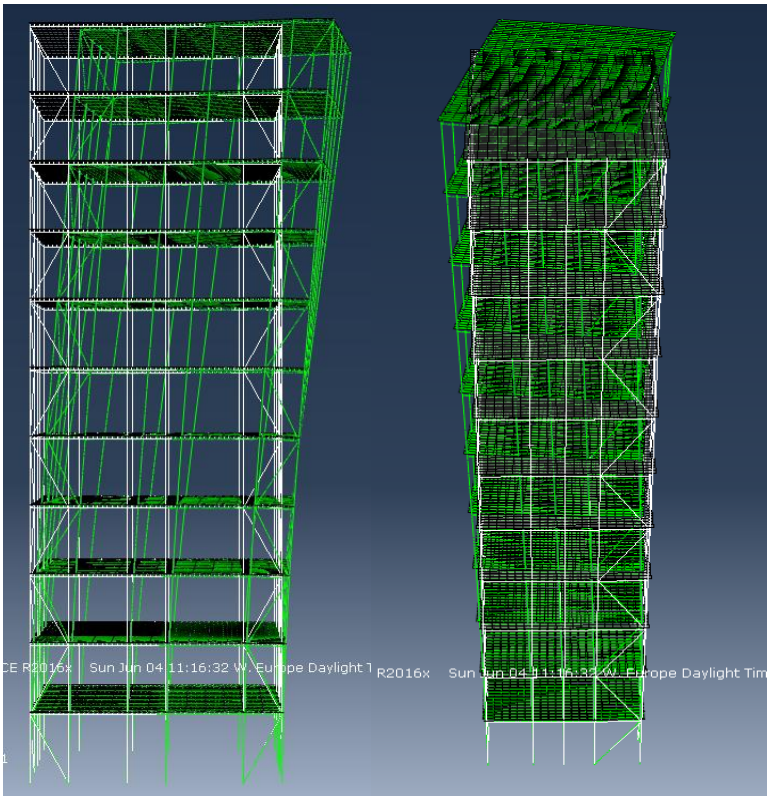

Figure 31: Deformed unsymmetrical timber frame structure when loaded with horizontal wind load.

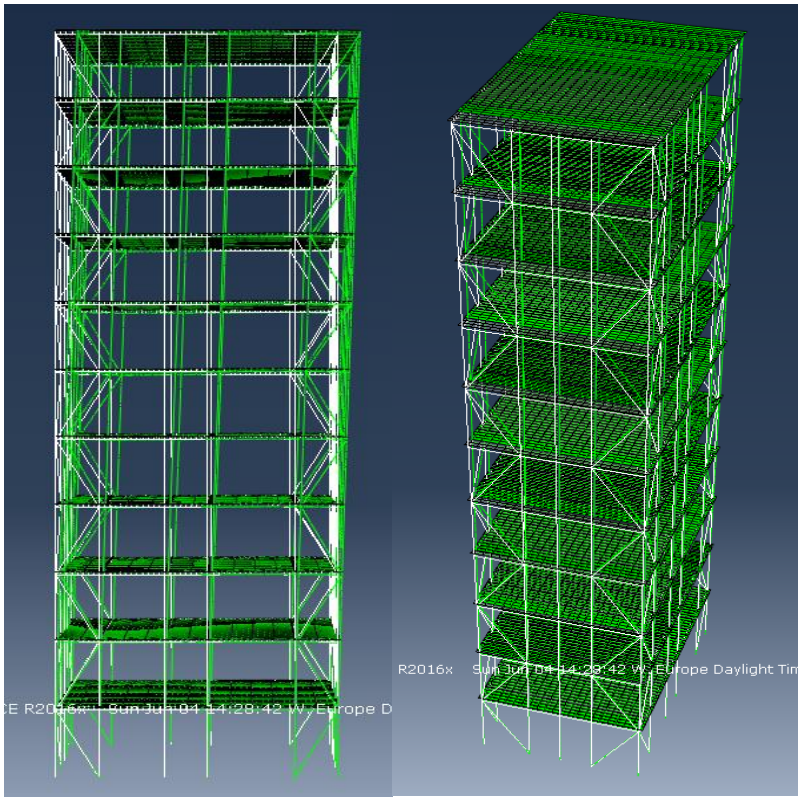

Figure 32: Deformed symmetrical timber frame structure when loaded with horizontal wing load.

The unsymmetrical structure in Figure 32 shows significant twisting deformation of the entire structure. The bending deformation becomes also bigger for this structure than for the symmetric structure shown in Figure 33. It can be predicted that the higher the timber structure is, the twisting deformations would become more dominating.

6. Discussion

The results acquired from the simulations present the feasibility of the scripts created in this thesis. The simulation methodology used has been shown to be reasonable for analyzing real multi-storey timber structures. By using parametrized Python scripts to create the input files, the users are able to create countless FE-models of different multi-storey buildings with distinguished sizes, boundary conditions, loading and material and connection properties by defining different parameter sets in the scripts. The benefit of using parametrized modelling is that it supports the engineers and designers to optimize the structural geometry and to study all possible load cases to avoid potential failures or stability problems.

Based on the results presented in Chapter 5, it is shown that the rotational stiffness of the frame connections have significant effect on the global bending stiffness of the structure, see e.g. Figure 20. Using very stiff connections, see Figures 21 and 22, corresponds to beam elements that would be glued together. However, it is not possible to create mechanical connections (of slotted-in steel plates) with such high stiffness, which often causes the connections to be the critical points in the timber frame structures. The rotational stiffness calculated by hand (according to Eurocode 5) for typical dowel based slotted-in steel plate connection also showed significant lower rotational stiffness than the maximum limit value shown in Figure 22.

Another factor that has huge impacts on the global bending stiffness of the structure is the existence of diagonals in the building.

After all, limitations of this work need to be discussed. First it should be mentioned that finite element simulation is an approximate solution of the studied problem where the accuracy is strongly linked to the mesh density, choice of elements, definition of boundary conditions etc. In this thesis, the frame structures are simulated by 3D beam elements because we are meanly interested of the overall structural behavior of the building. This means for example that the simulation of the connections is strongly simplified but it is sufficient for the global analysis of the structure. Examples of limitations in the script are that it is not possible to create internal walls in the building and the diagonals are not allowed to be crossed.

As initially mentioned in the first chapter, this work is a first step in a larger project e where different real multi-storey timber buildings will be carefully analyzed. Hence

numerous further works are required in the future the scripts created in this work can easily be further developed for more realistic and complex timber frame buildings. This will also save much time for the researchers and students that will continue on this work.

7. Conclusion

The simulation results present that the parametrized Python script significantly help the users to create large FE-models of different multi-storey timber frame structures. This method is significantly reducing the pre-processing time for the engineers and researchers. Furthermore, it also provide a method to create a flexible three-dimensional finite element model to study the influences of variable parameters such as connection stiffness, cross-sectional dimensions, number of columns and diagonal patterns, on the global structural behaviors of high-rise timber buildings.

In addition, the simulation results show that the global bending stiffness of the structure is significantly influenced by the rotational connection stiffness. The global bending stiffness increases sharply in the beginning but becomes more constant for large connection stiffness. This limitation value occurs when the connection stiffness reaches a value which corresponds to glued connections.

Furthermore, the existence and location of the diagonal elements have a major influence on the global bending behavior of the high multi-storey frame structures. The global bending stiffness of the structure shows an exponential increase when more diagonals were assembled to the structure.

From twisting results, another conclusion is that the bending deformation increases as well for an unsymmetrical structure compare to symmetric one. Also it can be predicted that the higher the timber is, the twisting deformations would become more dominating.

8. Future work

The next continuation of this work would be a stability (buckling) analysis to study the risk of local and global stability failure of the timber frame structure. An interesting thing would be to use the buckling results (eigenvalues and eigen modes) to obtain the buckling lengths (including effects from semi-rigid connections) needed for calculation of the stress criteria used for design of timber elements.

The Python script needs to be further developed regarding: changeable floor elements with holes, internal walls including doors and windows and more general diagonal pattern.

Finally adaptive modeling of the most critical timber connections need to be possible to perform in the future scripts. This means that the script will allow the user to perform a fully detailed 3D solid analysis (including contact interaction between the different parts in the connection) of some selected connection in the structure. This is very important to better understand the local connection behaviors and also to validate the simplified spring based connection models.

Reference

European Committee for Standardisation. (1991). Eurocode 1: Actions on structures — General actions .

About-ANSYS. (April 2, 2017). Source: ANSYS: http://www.ansys.com/en-GB/About-ANSYS

Abrahamsen RB,Malo. (2014). STRUCTURAL DESIGN AND ASSEMBLY OF “TREET” - A 14-STOREY TIMBER RESIDENTIAL BUILDING IN NORWAY.

Andreas Öchsner, Marco Öchsner. (2016). The Finite Element Analysis Program MSC Marc/Mentat: A First Introduction.

BaderTomasK., SchweiglerMichael, & HochreinerGeorg. (Augus 22-25, 2016). LOAD DISTRIBUTION IN MULTI-DOWEL TIMBER CONNECTIONS UNDER MOMENT LOADING-INTEGRATIVEEVALUATION OF MULTISCALE EXPERIMENTS. Wold Conference onTimer Engineering, page 1.

BOELLAARDB.J.H. (2012). Design of an outrigger structure for tall timber buildings.

CookD.Robert, PleshaE.Michael., & MalkusS.David. (1989). Concepts and Applications of Finite Element Analysis. John Wiley&Sons.

CrocettiR., JohanssonM., & KligerR. (2011). Design of timber structures. Stockholm: Swedish wood.

DeschE.H., & DinwoodieM.J. (1996). Timber Structure,Properties,Conversion and Use,7th edition. MACMILLAN PRESS LTD.

E. Frühwald&T. Toratti& S. Thelandersson& E. SerranoEmilssonA. (2007). Design of safe timber structures-how can we learn from structural failures in concrete, steel and timber?

GreeneE.B., StromeR.D., & WeikelC.R. (1961). Application of stiffness method to the analysis of shell structures. Los Angeles.

Korte.S., V.Boel, & W.D.Corte. (2014). Static and fatigue fracture mechanics properties of self-compacting concrete using three-point bending tests and wedge-splitting tests. Construction and Building Materials, 页 1-8.

Miao,Z& Xiaochuan,Y& Jianhui, L. (1991). Finite Element Analysis and Application Based on Abaqus. Tsinghua University Press.

OrmarssonSigurdur. (1999). Nmerical Analysis of Moisture-Related Distortions in Sawn Timber. Goteborg: CHALMERS UNIVERSITY OF TECHNOLOGY.

RacherP. (1995). Proceedings of Timber Engineering STEP IC.

RerallackJ.Gregory, & HuangChengmin. (January 1st, 2011). Ecology and evolution of Devonian trees in New York, USA. Palaeogeography, Palaeoclimatology, Palaeoecology, page 110-128.

ShouyiX. (2005). Finite element method. bEIJING: China Building Industry Press.
SIMULA. ABAQUS/CAE User's Manual.
WilliamsonG.Thomas. (2002). APA ENGINEERED WOOD HANDBOOK. McGraw-Hill.

Appendix 1: Created Python Script

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from abaqusConstants import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import math
#
# Definition of constants (Variable)
#
Scale_factor=0.1
# Number_of_Column_in_XY_plane=1
# Number_of_Column_in_YZ_plane=2

Number_of_Column_in_A_plane=4
Number_of_Column_in_B_plane=3
Number_of_Column_in_C_plane=4
Number_of_Column_in_D_plane=3

Height_of_Truss=30*(10*Scale_factor)# m
Number_of_Layer=5

Dist_bet_colum_of_side_1=[2.4,3.6,2.4,4.8,2.4,0,0,0,0,0,0,0,0,0,0,]
Dist_bet_colum_of_side_2=[3,2,2,3,0,0,0,0,0,0,0,0,0,0,0,]
Dist_bet_colum_of_side_3=[2.4,3.6,2.4,4.8,2.4,0,0,0,0,0,0,0,0,0,0,]
Dist_bet_colum_of_side_4=[3,2,2,3,0,0,0,0,0,0,0,0,0,0,0,]
##############################################################################
##############################################################################
##
#      ATTENTION!!!!!!!: The Total Length of Side_1 must equal to Side_3, as the same, the Length of Side 2 must equal to Side_4, because of the rectangle shape #
##############################################################################
##############################################################################
##


#Define where to put Diagonals
Dia_A_L=[0,1,4,7,10,11,14,17,20,21,24,27]#,30,31,34,37,40,41,44]
Dia_A_R=[2,5,6,9,12,15,16,19,22,25,26]#,29,32,35,36,39,42]
Dia_B_L=[0,3,5,6,8,11,13,14,16,19,21,22]#,24,27,29,30,32,35]
Dia_B_R=[1,2,4,7,9,10,12,15,17,18,20,23]#,25,26,28,31,33,34]
Dia_C_L=[0,1,4,7,10,11,14,17,20,21,24,27]#,30,31,34,37,40,41,44]
Dia_C_R=[2,5,6,9,12,15,16,19,22,25,26]#,29,32,35,36,39,42]
```

```
Dia_D_L=[0,1,5,6,11,12,16,17]#,18,23,24]
Dia_D_R=[2,3,7,8,9,13,14,15]#,20,21,25,26]
#
# Maxmum_Number_of_Column=5    #more column needs more Lx

# Cross section dimensions
Width_of_section=0.4*(10*Scale_factor)
Length_of_section=0.2*(10*Scale_factor)
Width_of_pin=0.05
Length_of_pin=0.1
#
Length_of_connection=(0.6-Width_of_section/2)*(10*Scale_factor)#m
Length_of_connection_2=(0.6-Length_of_section/2)*(10*Scale_factor)#m
Length_Hinge=Length_of_connection+(Width_of_section/2)
#define the parameters of plate
Len_Pla= 1.2
Wid_Pla= 10+Width_of_section
Length_of_plate=Len_Pla*(10*Scale_factor) #m
Width_of_plate=Wid_Pla*(10*Scale_factor)    #m
Length_of_Crossection_plate=Length_of_plate    #m
Width_of_Crossection_plate=0.2*(10*Scale_factor)    #m
Number_of_partition_in_CLT_V=3

#Define Wind Loads
Basic_wind_velocity=24 #m/s
Air_density=1.25 #
#define the type of terrain
Type_of_terrain=2
#Other definations
Turblence_factor=1
Orography_factor=1
Zo=[ (0.003,1),
        (0.01,1),
        (0.05,2),
        (0.3,5),
        (1,10)]
#
#Define the self-weight per storey
S_Weight=750000 # N

#
for i, j in enumerate(Dia_A_L):
  Number_of_Dia_A_L=i+1
for i, j in enumerate(Dia_A_R):
  Number_of_Dia_A_R=i+1
for i, j in enumerate(Dia_B_L):
  Number_of_Dia_B_L=i+1
for i, j in enumerate(Dia_B_R):
  Number_of_Dia_B_R=i+1
for i, j in enumerate(Dia_C_L):
  Number_of_Dia_C_L=i+1
for i, j in enumerate(Dia_C_R):
  Number_of_Dia_C_R=i+1
for i, j in enumerate(Dia_D_L):
  Number_of_Dia_D_L=i+1
for i, j in enumerate(Dia_D_R):
  Number_of_Dia_D_R=i+1
```

```
Length_of_Truss=Width_of_Truss=0,
sum=0
for i in Dist_bet_colum_of_side_1:
    sum+=i
    Length_of_Truss=sum
sum1=0
for i in Dist_bet_colum_of_side_2:
    sum1+=i
    Width_of_Truss=sum1


Length_of_Truss=Length_of_Truss*(10*Scale_factor) # m
Width_of_Truss=Width_of_Truss*(10*Scale_factor)    # m



Lx_A = [0]*(Number_of_Column_in_A_plane+1)
Lx_C = [0]*(Number_of_Column_in_C_plane+1)
Lz_B = [0]*(Number_of_Column_in_B_plane+1)
Lz_D = [0]*(Number_of_Column_in_D_plane+1)
# Lx_A_1=Lx_A_2=Lx_A_3=Lx_A_4=Lx_A_5=Lx_A_6=Lx_A_7=Lx_A_8=0
for i in range(Number_of_Column_in_A_plane+1):
  Lx_A[i]=Dist_bet_colum_of_side_1[i]

for i in range(Number_of_Column_in_B_plane+1):
  Lz_B[i]=Dist_bet_colum_of_side_2[i]

for i in range(Number_of_Column_in_C_plane+1):
  Lx_C[i]=Dist_bet_colum_of_side_3[i]

for i in range(Number_of_Column_in_D_plane+1):
  Lz_D[i]=Dist_bet_colum_of_side_4[i]


# Lx1=(Length_of_Truss/(Number_of_Column_in_XY_plane+1))*(10*Scale_factor)
# Lx2=Lx3=Lx4=Lx5=Lx6=Lx1

Ly1=(Height_of_Truss/(Number_of_Layer+1))*(10*Scale_factor)
Ly2=Ly3=Ly1
Ly4=Ly5=Ly6=Ly1
Ly7=Ly8=ly9=Ly1
Ly10=Ly11=Ly12=Ly1

#
# Material properties
#
E_l=210000000000
# Top and bottom chords C24
El_TopBott_k = 13700000000.0 #[Pa]
#Er_TopBott_k = El_TopBott_k
#Et_TopBott_k = El_TopBott_k
Er_TopBott_k = 800000000
Et_TopBott_k = 800000000
#Glr_TopBott_k = El_TopBott_k/16 # Orthotrop material
Glr_TopBott_k = 8500000000 # Isotrop matrial
#Glt_TopBott_k = El_TopBott_k/16 # Orthotrop material
```

```
Glt_TopBott_k = 8500000000 # Isotrop matrial
#Grt_TopBott_k = 60000000.0 #[Pa] Orthotrop material
Grt_TopBott_k = 2000000000 # Isotrop matrial
nylr_TopBott_k = 0.0
nylt_TopBott_k = 0.0
nyrt_TopBott_k = 0.0
t_Diago_k = 0.0
# #
# # Diagonals C24
# El_Diago_k = 7400000000.0 #[Pa]
# #Er_Diago_k = El_Diago_k
# #Et_Diago_k = El_Diago_k
# Er_Diago_k = 370000000.0*7400.0/11000.0
# Et_Diago_k = 370000000.0*7400.0/11000.0
# #Glr_Diago_k = El_Diago_k/16 # Orthotrop material
# Glr_Diago_k = El_Diago_k/2.0 # Isotrop matrial
# #Glt_Diago_k = El_Diago_k/16 # Orthotrop material
# Glt_Diago_k = El_Diago_k/2.0 # Isotrop matrial
# #Grt_Diago_k = 60000000.0 #[Pa] Orthotrop material
# Grt_Diago_k = El_Diago_k/2.0 # Isotrop matrial
# nylr_Diago_k = 0.0
# nylt_Diago_k = 0.0
# nyr


#
# Coordinates for the parts
#


Lx_co_A=[0]*(Number_of_Column_in_A_plane+2)
sum2=0
for i in range(Number_of_Column_in_A_plane+1):
    sum2+=Lx_A[i]
    Lx_co_A[i+1]=sum2


Lx_co_C=[0]*(Number_of_Column_in_C_plane+2)
sum3=0
for i in range(Number_of_Column_in_C_plane+1):
    sum3+=Lx_C[i]
    Lx_co_C[i+1]=sum3


Lz_co_B=[0]*(Number_of_Column_in_B_plane+2)
sum4=0
for i in range(Number_of_Column_in_B_plane+1):
    sum4+=Lz_B[i]
    Lz_co_B[i+1]=sum4


Lz_co_D=[0]*(Number_of_Column_in_D_plane+2)
sum5=0
for i in range(Number_of_Column_in_D_plane+1):
    sum5+=Lz_D[i]
    Lz_co_D[i+1]=sum5
```

```
# Truss_member_coord_x =
[(0,0)]*(4+(2*Number_of_Column_in_XY_plane+2*Number_of_Column_in_YZ_plane)+(Numbe
r_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of
_Column_in_B_plane+Number_of_Column_in_D_plane+4))
#Truss_member_coord_x
Truss_member_coord_x =
[(0,0)]*(4+(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Col
umn_in_B_plane+Number_of_Column_in_D_plane)+(Number_of_Layer+1)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=Lx_co_A[i]+(Width
_of_section/2),Lx_co_A[i+1]-(Width_of_section/2),
  for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i]=Lx_co_C[i]+(Width_of_section/2),Lx_co_C[
i+1]-(Width_of_section/2),
  for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+n]=Length_of_Truss,Length_of_Truss,
  for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(Number_of_Column_i
n_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column_in_B_plane+1)+n]
=0,0,
for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
=Lx_co_A[i],Lx_co_A[i],
for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+Number_of_Column_in_B_plane+2+i]=Lx_co_C[Number_of_C
olumn_in_C_plane+1-i],Lx_co_C[Number_of_Column_in_C_plane+1-i]
for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+1+n]=Length_of_Truss,Length_of_Truss,
for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(N
umber_of_Column_in_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column
_in_B_plane+1)+n]=0,0,
```

```
#
#
#Truss_member_coord_y
Truss_member_coord_y                                                    =
[(0,0)]*(4+(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Col
umn_in_B_plane+Number_of_Column_in_D_plane)+(Number_of_Layer+1)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=Ly1*(Number_of_L
ayer+1-m),Ly1*(Number_of_Layer+1-m),
  for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i]=Ly1*(Number_of_Layer+1-m),Ly1*(Number
_of_Layer+1-m),
  for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+n]=Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer+1-m),
  for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(Number_of_Column_i
n_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column_in_B_plane+1)+n]
=Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer+1-m),
for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
=0.0, Height_of_Truss
for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+Number_of_Column_in_B_plane+2+i]=0.0, Height_of_Truss
for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+1+n]=0.0, Height_of_Truss
for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(N
umber_of_Column_in_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column
_in_B_plane+1)+n]=0.0, Height_of_Truss
#
#
#Truss_member_coord_z
Truss_member_coord_z                                                    =
[(0,0)]*(4+(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Col
```

```
umn_in_B_plane+Number_of_Column_in_D_plane)+(Number_of_Layer+1)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=0,0,
  for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i]=-1*Width_of_Truss,-1*Width_of_Truss,
  for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+n]=-1*Lz_co_B[n]-(Length_of_section/2),-1*(Lz_co_B[n+1]-(Length_of_section/
2)),
  for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(Number_of_Column_i
n_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column_in_B_plane+1)+n]
=-1*Lz_co_D[n]-(Length_of_section/2),-1*(Lz_co_D[n+1]-(Length_of_section/2)),
for i in range(Number_of_Column_in_A_plane+1):

Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
=0,0,
for i in range(Number_of_Column_in_C_plane+1):

Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+Number_of_Column_in_B_plane+2+i]=-1*Width_of_Truss,-1*
Width_of_Truss,
for n in range(Number_of_Column_in_B_plane+1):

Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+N
umber_of_Column_in_A_plane+1+n]=-Lz_co_B[n],-Lz_co_B[n],
for n in range(Number_of_Column_in_D_plane+1):

Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(N
umber_of_Column_in_A_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column
_in_B_plane+1)+n]=-Lz_co_D[Number_of_Column_in_D_plane+1-n],-Lz_co_D[Number_of_C
olumn_in_D_plane+1-n]
#
#
#Truss_ref_coord_x


Truss_Ref_coord_x                                                    =
[(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
```

```
  for i in range(Number_of_Column_in_A_plane+1):

Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=Lx_co_A[i]+Length_of_
connection+Width_of_section/2,Lx_co_A[i+1]-Width_of_section/2-Length_of_connection,
  for i in range(Number_of_Column_in_C_plane+1):

Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i]=Lx_co_C[i]+Length_of_connection+Width_of_se
ction/2,Lx_co_C[i+1]-Width_of_section/2-Length_of_connection,
  for n in range(Number_of_Column_in_B_plane+1):

Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+n]=Length_of_Truss,Length_of_Truss,
  for n in range(Number_of_Column_in_D_plane+1):

Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(Number_of_Column_in_A
_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column_in_B_plane+1)+n]=0,0
,
#
#
#Truss_ref_coord_y

Truss_Ref_coord_y                                                                =
[(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
  for                                  i                                  in
range(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_i
n_B_plane+Number_of_Column_in_D_plane+4):

Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=Ly1*(Number_of_Layer
+1-m),Ly1*(Number_of_Layer+1-m),

#
#
#Truss_ref_coord_z
Truss_Ref_coord_z                                                                =
[(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):

Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]=0,0
  for i in range(Number_of_Column_in_C_plane+1):

Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i]=-(Width_of_Truss),-(Width_of_Truss)
  for n in range(Number_of_Column_in_B_plane+1):

Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
```

```
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+n]=-Lz_co_B[n]-Length_of_connection_2-Length_of_section/2,-((Lz_co_B[n+1])-Leng
th_of_section/2-Length_of_connection_2),
  for n in range(Number_of_Column_in_D_plane+1):

Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+(Number_of_Column_in_A
_plane+1)+(Number_of_Column_in_C_plane+1)+(Number_of_Column_in_B_plane+1)+n]=-Lz
_co_D[n]-Length_of_connection_2-Length_of_section/2,-((Lz_co_D[n+1])-Length_of_section/2-
Length_of_connection_2),




#
#Define the parameters for diagonals
#
#A
Diagonal_A_x_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
Diagonal_A_y_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
Diagonal_A_z_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
Diagonal_A_x_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
Diagonal_A_y_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
Diagonal_A_z_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):
    Diagonal_A_x_1[(m)*(Number_of_Column_in_A_plane+1)+i]= Lx_co_A[i],Lx_co_A[i+1]
    Diagonal_A_y_1[(m)*(Number_of_Column_in_A_plane+1)+i]=
Ly1*(Number_of_Layer-m),Ly1*(Number_of_Layer+1-m)
    Diagonal_A_z_1[(m)*(Number_of_Column_in_A_plane+1)+i]= 0,0
    Diagonal_A_x_2[(m)*(Number_of_Column_in_A_plane+1)+i]= Lx_co_A[i],Lx_co_A[i+1]
    Diagonal_A_y_2[(m)*(Number_of_Column_in_A_plane+1)+i]=
Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer-m)
    Diagonal_A_z_2[(m)*(Number_of_Column_in_A_plane+1)+i]= 0,0
#
#B
Diagonal_B_x_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
Diagonal_B_y_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
Diagonal_B_z_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
Diagonal_B_x_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
Diagonal_B_y_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
Diagonal_B_z_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_B_plane+1):
    Diagonal_B_x_1[(m)*(Number_of_Column_in_B_plane+1)+i]=
Length_of_Truss,Length_of_Truss
    Diagonal_B_y_1[(m)*(Number_of_Column_in_B_plane+1)+i]=
Ly1*(Number_of_Layer-m),Ly1*(Number_of_Layer+1-m)
    Diagonal_B_z_1[(m)*(Number_of_Column_in_B_plane+1)+i]=
-1*Lz_co_B[i],-1*Lz_co_B[i+1]
    Diagonal_B_x_2[(m)*(Number_of_Column_in_B_plane+1)+i]=
Length_of_Truss,Length_of_Truss
    Diagonal_B_y_2[(m)*(Number_of_Column_in_B_plane+1)+i]=
Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer-m)
    Diagonal_B_z_2[(m)*(Number_of_Column_in_B_plane+1)+i]=
```

```
-1*Lz_co_B[i],-1*Lz_co_B[i+1]
#
#C
Diagonal_C_x_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
Diagonal_C_y_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
Diagonal_C_z_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
Diagonal_C_x_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
Diagonal_C_y_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
Diagonal_C_z_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_C_plane+1):
    Diagonal_C_x_1[(m)*(Number_of_Column_in_C_plane+1)+i]= Lx_co_C[i],Lx_co_C[i+1]
    Diagonal_C_y_1[(m)*(Number_of_Column_in_C_plane+1)+i]=
Ly1*(Number_of_Layer-m),Ly1*(Number_of_Layer+1-m)
    Diagonal_C_z_1[(m)*(Number_of_Column_in_C_plane+1)+i]=
-1*Width_of_Truss,-1*Width_of_Truss
    Diagonal_C_x_2[(m)*(Number_of_Column_in_C_plane+1)+i]= Lx_co_C[i],Lx_co_C[i+1]
    Diagonal_C_y_2[(m)*(Number_of_Column_in_C_plane+1)+i]=
Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer-m)
    Diagonal_C_z_2[(m)*(Number_of_Column_in_C_plane+1)+i]=
-1*Width_of_Truss,-1*Width_of_Truss
#
#D
Diagonal_D_x_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
Diagonal_D_y_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
Diagonal_D_z_1 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
Diagonal_D_x_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
Diagonal_D_y_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
Diagonal_D_z_2 = [(0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_D_plane+1):
    Diagonal_D_x_1[(m)*(Number_of_Column_in_D_plane+1)+i]= 0,0
    Diagonal_D_y_1[(m)*(Number_of_Column_in_D_plane+1)+i]=
Ly1*(Number_of_Layer-m),Ly1*(Number_of_Layer+1-m)
    Diagonal_D_z_1[(m)*(Number_of_Column_in_D_plane+1)+i]=
-1*Lz_co_D[i],-1*Lz_co_D[i+1]
    Diagonal_D_x_2[(m)*(Number_of_Column_in_D_plane+1)+i]= 0,0
    Diagonal_D_y_2[(m)*(Number_of_Column_in_D_plane+1)+i]=
Ly1*(Number_of_Layer+1-m),Ly1*(Number_of_Layer-m)
    Diagonal_D_z_2[(m)*(Number_of_Column_in_D_plane+1)+i]=
-1*Lz_co_D[i],-1*Lz_co_D[i+1]




Pla_Diagonals_A_L=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane
+1))
Pla_Diagonals_A_R=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane
+1))
Pla_Diagonals_B_L=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane
+1))
Pla_Diagonals_B_R=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane
+1))
Pla_Diagonals_C_L=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane
```

```
+1))
Pla_Diagonals_C_R=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_C_plan
e+1))
Pla_Diagonals_D_L=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plan
e+1))
Pla_Diagonals_D_R=[(0,0,0,0,0,0,0)]*((Number_of_Layer+1)*(Number_of_Column_in_D_plan
e+1))


sequence_A=[0]*(Number_of_Column_in_A_plane+1)
sum7=0
for i in range(Number_of_Column_in_A_plane):
   sum7+=1
   sequence_A[i+1]=sum7

for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_A):

Pla_Diagonals_A_L[n*(Number_of_Column_in_A_plane+1)+i]=(j+1,j+2,n+1,1,Diagonal_A_x_
1[(n)*(Number_of_Column_in_A_plane+1)+i][0],Diagonal_A_y_1[(n)*(Number_of_Column_in
_A_plane+1)+i][0],Diagonal_A_z_1[(n)*(Number_of_Column_in_A_plane+1)+i][0],Diagonal_
A_x_1[(n)*(Number_of_Column_in_A_plane+1)+i][1],Diagonal_A_y_1[(n)*(Number_of_Colum
n_in_A_plane+1)+i][1],Diagonal_A_z_1[(n)*(Number_of_Column_in_A_plane+1)+i][1])
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_A):

Pla_Diagonals_A_R[n*(Number_of_Column_in_A_plane+1)+i]=(j+1,j+2,n+1,2,Diagonal_A_x_
2[(n)*(Number_of_Column_in_A_plane+1)+i][0],Diagonal_A_y_2[(n)*(Number_of_Column_in
_A_plane+1)+i][0],Diagonal_A_z_2[(n)*(Number_of_Column_in_A_plane+1)+i][0],Diagonal_
A_x_2[(n)*(Number_of_Column_in_A_plane+1)+i][1],Diagonal_A_y_2[(n)*(Number_of_Colum
n_in_A_plane+1)+i][1],Diagonal_A_z_2[(n)*(Number_of_Column_in_A_plane+1)+i][1])


sequence_B=[0]*(Number_of_Column_in_B_plane+1)
sum8=0
for i in range(Number_of_Column_in_B_plane):
   sum8+=1
   sequence_B[i+1]=sum8
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_B):

Pla_Diagonals_B_L[n*(Number_of_Column_in_B_plane+1)+i]=(j+1,j+2,n+1,1,Diagonal_B_x_
1[(n)*(Number_of_Column_in_B_plane+1)+i][0],Diagonal_B_y_1[(n)*(Number_of_Column_in
_B_plane+1)+i][0],Diagonal_B_z_1[(n)*(Number_of_Column_in_B_plane+1)+i][0],Diagonal_
B_x_1[(n)*(Number_of_Column_in_B_plane+1)+i][1],Diagonal_B_y_1[(n)*(Number_of_Colum
n_in_B_plane+1)+i][1],Diagonal_B_z_1[(n)*(Number_of_Column_in_B_plane+1)+i][1])
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_B):

Pla_Diagonals_B_R[n*(Number_of_Column_in_B_plane+1)+i]=(j+1,j+2,n+1,2,Diagonal_B_x_
2[(n)*(Number_of_Column_in_B_plane+1)+i][0],Diagonal_B_y_2[(n)*(Number_of_Column_in
_B_plane+1)+i][0],Diagonal_B_z_2[(n)*(Number_of_Column_in_B_plane+1)+i][0],Diagonal_
B_x_2[(n)*(Number_of_Column_in_B_plane+1)+i][1],Diagonal_B_y_2[(n)*(Number_of_Colum
n_in_B_plane+1)+i][1],Diagonal_B_z_2[(n)*(Number_of_Column_in_B_plane+1)+i][1])

sequence_C=[0]*(Number_of_Column_in_C_plane+1)
sum9=0
```

```
for i in range(Number_of_Column_in_C_plane):
   sum9+=1
   sequence_C[i+1]=sum9
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_C):

Pla_Diagonals_C_L[n*(Number_of_Column_in_C_plane+1)+i]=(j+1,j+2,n+1,1,Diagonal_C_x
_1[(n)*(Number_of_Column_in_C_plane+1)+i][0],Diagonal_C_y_1[(n)*(Number_of_Column_i
n_C_plane+1)+i][0],Diagonal_C_z_1[(n)*(Number_of_Column_in_C_plane+1)+i][0],Diagonal
_C_x_1[(n)*(Number_of_Column_in_C_plane+1)+i][1],Diagonal_C_y_1[(n)*(Number_of_Colu
mn_in_C_plane+1)+i][1],Diagonal_C_z_1[(n)*(Number_of_Column_in_C_plane+1)+i][1])
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_C):

Pla_Diagonals_C_R[n*(Number_of_Column_in_C_plane+1)+i]=(j+1,j+2,n+1,2,Diagonal_C_x
_2[(n)*(Number_of_Column_in_C_plane+1)+i][0],Diagonal_C_y_2[(n)*(Number_of_Column_i
n_C_plane+1)+i][0],Diagonal_C_z_2[(n)*(Number_of_Column_in_C_plane+1)+i][0],Diagonal
_C_x_2[(n)*(Number_of_Column_in_C_plane+1)+i][1],Diagonal_C_y_2[(n)*(Number_of_Colu
mn_in_C_plane+1)+i][1],Diagonal_C_z_2[(n)*(Number_of_Column_in_C_plane+1)+i][1])

sequence_D=[0]*(Number_of_Column_in_D_plane+1)
sum10=0
for i in range(Number_of_Column_in_D_plane):
   sum10+=1
   sequence_D[i+1]=sum10
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_D):

Pla_Diagonals_D_L[n*(Number_of_Column_in_D_plane+1)+i]=(j+1,j+2,n+1,1,Diagonal_D_x
_1[(n)*(Number_of_Column_in_D_plane+1)+i][0],Diagonal_D_y_1[(n)*(Number_of_Column_i
n_D_plane+1)+i][0],Diagonal_D_z_1[(n)*(Number_of_Column_in_D_plane+1)+i][0],Diagona
l_D_x_1[(n)*(Number_of_Column_in_D_plane+1)+i][1],Diagonal_D_y_1[(n)*(Number_of_Col
umn_in_D_plane+1)+i][1],Diagonal_D_z_1[(n)*(Number_of_Column_in_D_plane+1)+i][1])
for n in range(Number_of_Layer+1):
  for i, j in enumerate(sequence_D):

Pla_Diagonals_D_R[n*(Number_of_Column_in_D_plane+1)+i]=(j+1,j+2,n+1,2,Diagonal_D_x
_2[(n)*(Number_of_Column_in_D_plane+1)+i][0],Diagonal_D_y_2[(n)*(Number_of_Column_i
n_D_plane+1)+i][0],Diagonal_D_z_2[(n)*(Number_of_Column_in_D_plane+1)+i][0],Diagona
l_D_x_2[(n)*(Number_of_Column_in_D_plane+1)+i][1],Diagonal_D_y_2[(n)*(Number_of_Col
umn_in_D_plane+1)+i][1],Diagonal_D_z_2[(n)*(Number_of_Column_in_D_plane+1)+i][1])




Dia_A_L_x=[(0,0)]*Number_of_Dia_A_L
Dia_A_L_y=[(0,0)]*Number_of_Dia_A_L
Dia_A_L_z=[(0,0)]*Number_of_Dia_A_L
for i in range(Number_of_Dia_A_L):
   Dia_A_L_x[i] = Pla_Diagonals_A_L[Dia_A_L[i]][4],Pla_Diagonals_A_L[Dia_A_L[i]][7]
   Dia_A_L_y[i] = Pla_Diagonals_A_L[Dia_A_L[i]][5],Pla_Diagonals_A_L[Dia_A_L[i]][8]
   Dia_A_L_z[i] = Pla_Diagonals_A_L[Dia_A_L[i]][6],Pla_Diagonals_A_L[Dia_A_L[i]][9]

Dia_A_R_x=[(0,0)]*Number_of_Dia_A_R
Dia_A_R_y=[(0,0)]*Number_of_Dia_A_R
Dia_A_R_z=[(0,0)]*Number_of_Dia_A_R
```

```
for i in range(Number_of_Dia_A_R):
   Dia_A_R_x[i] = Pla_Diagonals_A_R[Dia_A_R[i]][4],Pla_Diagonals_A_R[Dia_A_R[i]][7]
   Dia_A_R_y[i] = Pla_Diagonals_A_R[Dia_A_R[i]][5],Pla_Diagonals_A_R[Dia_A_R[i]][8]
   Dia_A_R_z[i] = Pla_Diagonals_A_R[Dia_A_R[i]][6],Pla_Diagonals_A_R[Dia_A_R[i]][9]

Dia_B_L_x=[(0,0)]*Number_of_Dia_B_L
Dia_B_L_y=[(0,0)]*Number_of_Dia_B_L
Dia_B_L_z=[(0,0)]*Number_of_Dia_B_L
for i in range(Number_of_Dia_B_L):
   Dia_B_L_x[i] = Pla_Diagonals_B_L[Dia_B_L[i]][4],Pla_Diagonals_B_L[Dia_B_L[i]][7]
   Dia_B_L_y[i] = Pla_Diagonals_B_L[Dia_B_L[i]][5],Pla_Diagonals_B_L[Dia_B_L[i]][8]
   Dia_B_L_z[i] = Pla_Diagonals_B_L[Dia_B_L[i]][6],Pla_Diagonals_B_L[Dia_B_L[i]][9]

Dia_B_R_x=[(0,0)]*Number_of_Dia_B_R
Dia_B_R_y=[(0,0)]*Number_of_Dia_B_R
Dia_B_R_z=[(0,0)]*Number_of_Dia_B_R
for i in range(Number_of_Dia_B_R):
   Dia_B_R_x[i] = Pla_Diagonals_B_R[Dia_B_R[i]][4],Pla_Diagonals_B_R[Dia_B_R[i]][7]
   Dia_B_R_y[i] = Pla_Diagonals_B_R[Dia_B_R[i]][5],Pla_Diagonals_B_R[Dia_B_R[i]][8]
   Dia_B_R_z[i] = Pla_Diagonals_B_R[Dia_B_R[i]][6],Pla_Diagonals_B_R[Dia_B_R[i]][9]


Dia_C_L_x=[(0,0)]*Number_of_Dia_C_L
Dia_C_L_y=[(0,0)]*Number_of_Dia_C_L
Dia_C_L_z=[(0,0)]*Number_of_Dia_C_L
for i in range(Number_of_Dia_C_L):
   Dia_C_L_x[i] = Pla_Diagonals_C_L[Dia_C_L[i]][4],Pla_Diagonals_C_L[Dia_C_L[i]][7]
   Dia_C_L_y[i] = Pla_Diagonals_C_L[Dia_C_L[i]][5],Pla_Diagonals_C_L[Dia_C_L[i]][8]
   Dia_C_L_z[i] = Pla_Diagonals_C_L[Dia_C_L[i]][6],Pla_Diagonals_C_L[Dia_C_L[i]][9]

Dia_C_R_x=[(0,0)]*Number_of_Dia_C_R
Dia_C_R_y=[(0,0)]*Number_of_Dia_C_R
Dia_C_R_z=[(0,0)]*Number_of_Dia_C_R
for i in range(Number_of_Dia_C_R):
   Dia_C_R_x[i] = Pla_Diagonals_C_R[Dia_C_R[i]][4],Pla_Diagonals_C_R[Dia_C_R[i]][7]
   Dia_C_R_y[i] = Pla_Diagonals_C_R[Dia_C_R[i]][5],Pla_Diagonals_C_R[Dia_C_R[i]][8]
   Dia_C_R_z[i] = Pla_Diagonals_C_R[Dia_C_R[i]][6],Pla_Diagonals_C_R[Dia_C_R[i]][9]


Dia_D_L_x=[(0,0)]*Number_of_Dia_D_L
Dia_D_L_y=[(0,0)]*Number_of_Dia_D_L
Dia_D_L_z=[(0,0)]*Number_of_Dia_D_L
for i in range(Number_of_Dia_D_L):
   Dia_D_L_x[i] = Pla_Diagonals_D_L[Dia_D_L[i]][4],Pla_Diagonals_D_L[Dia_D_L[i]][7]
   Dia_D_L_y[i] = Pla_Diagonals_D_L[Dia_D_L[i]][5],Pla_Diagonals_D_L[Dia_D_L[i]][8]
   Dia_D_L_z[i] = Pla_Diagonals_D_L[Dia_D_L[i]][6],Pla_Diagonals_D_L[Dia_D_L[i]][9]

Dia_D_R_x=[(0,0)]*Number_of_Dia_D_R
Dia_D_R_y=[(0,0)]*Number_of_Dia_D_R
Dia_D_R_z=[(0,0)]*Number_of_Dia_D_R
for i in range(Number_of_Dia_D_R):
   Dia_D_R_x[i] = Pla_Diagonals_D_R[Dia_D_R[i]][4],Pla_Diagonals_D_R[Dia_D_R[i]][7]
   Dia_D_R_y[i] = Pla_Diagonals_D_R[Dia_D_R[i]][5],Pla_Diagonals_D_R[Dia_D_R[i]][8]
   Dia_D_R_z[i] = Pla_Diagonals_D_R[Dia_D_R[i]][6],Pla_Diagonals_D_R[Dia_D_R[i]][9]

Diagonal_Length_A_L=[0]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):
```

```
   Diagonal_Length_A_L[i]=
sqrt((Dia_A_L_x[i][1]-Dia_A_L_x[i][0])**2+(Dia_A_L_y[i][1]-Dia_A_L_y[i][0])**2)
Diagonal_Length_A_R=[0]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):
   Diagonal_Length_A_R[i]=
sqrt((Dia_A_R_x[i][1]-Dia_A_R_x[i][0])**2+(Dia_A_R_y[i][1]-Dia_A_R_y[i][0])**2)


Diagonal_Length_B_L=[0]*(Number_of_Dia_B_L)
for i in range(Number_of_Dia_B_L):
   Diagonal_Length_B_L[i]=
sqrt((Dia_B_L_z[i][1]-Dia_B_L_z[i][0])**2+(Dia_B_L_y[i][1]-Dia_B_L_y[i][0])**2)
Diagonal_Length_B_R=[0]*(Number_of_Dia_B_R)
for i in range(Number_of_Dia_B_R):
   Diagonal_Length_B_R[i]=
sqrt((Dia_B_R_z[i][1]-Dia_B_R_z[i][0])**2+(Dia_B_R_y[i][1]-Dia_B_R_y[i][0])**2)


Diagonal_Length_C_L=[0]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):
   Diagonal_Length_C_L[i]=
sqrt((Dia_C_L_x[i][1]-Dia_C_L_x[i][0])**2+(Dia_C_L_y[i][1]-Dia_C_L_y[i][0])**2)
Diagonal_Length_C_R=[0]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):
   Diagonal_Length_C_R[i]=
sqrt((Dia_C_R_x[i][1]-Dia_C_R_x[i][0])**2+(Dia_C_R_y[i][1]-Dia_C_R_y[i][0])**2)


Diagonal_Length_D_L=[0]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):
   Diagonal_Length_D_L[i]=
sqrt((Dia_D_L_z[i][1]-Dia_D_L_z[i][0])**2+(Dia_D_L_y[i][1]-Dia_D_L_y[i][0])**2)
Diagonal_Length_D_R=[0]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):
   Diagonal_Length_D_R[i]=
sqrt((Dia_D_R_z[i][1]-Dia_D_R_z[i][0])**2+(Dia_D_R_y[i][1]-Dia_D_R_y[i][0])**2)




Dia_Deg_A_L=[0]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):

Dia_Deg_A_L[i]=atan((Dia_A_L_y[i][1]-Dia_A_L_y[i][0])/(Dia_A_L_x[i][1]-Dia_A_L_x[i][0]
))
Dia_Deg_A_R=[0]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):

Dia_Deg_A_R[i]=atan((Dia_A_R_y[i][0]-Dia_A_R_y[i][1])/(Dia_A_R_x[i][1]-Dia_A_R_x[i][0
]))

Dia_Deg_B_L=[0]*(Number_of_Dia_B_L)
for i in range(Number_of_Dia_B_L):

Dia_Deg_B_L[i]=atan((Dia_B_L_y[i][1]-Dia_B_L_y[i][0])/(Dia_B_L_z[i][0]-Dia_B_L_z[i][1]
))
Dia_Deg_B_R=[0]*(Number_of_Dia_B_R)
```

```
for i in range(Number_of_Dia_B_R):

Dia_Deg_B_R[i]=atan((Dia_B_R_y[i][0]-Dia_B_R_y[i][1])/(Dia_B_R_z[i][0]-Dia_B_R_z[i][1]
))

Dia_Deg_C_L=[0]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):

Dia_Deg_C_L[i]=atan((Dia_C_L_y[i][1]-Dia_C_L_y[i][0])/(Dia_C_L_x[i][1]-Dia_C_L_x[i][0
]))
Dia_Deg_C_R=[0]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):

Dia_Deg_C_R[i]=atan((Dia_C_R_y[i][0]-Dia_C_R_y[i][1])/(Dia_C_R_x[i][1]-Dia_C_R_x[i][0
]))

Dia_Deg_D_L=[0]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):

Dia_Deg_D_L[i]=atan((Dia_D_L_y[i][1]-Dia_D_L_y[i][0])/(Dia_D_L_z[i][0]-Dia_D_L_z[i][1
]))
Dia_Deg_D_R=[0]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):

Dia_Deg_D_R[i]=atan((Dia_D_R_y[i][0]-Dia_D_R_y[i][1])/(Dia_D_R_z[i][0]-Dia_D_R_z[i][
1]))




Dia_A_conect_L_x=[(0,0)]*(Number_of_Dia_A_L)
Dia_A_conect_L_y=[(0,0)]*(Number_of_Dia_A_L)
Dia_A_conect_L_z=[(0,0)]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):
  Dia_A_conect_L_x[i]                                                    =
Length_Hinge*(cos(Dia_Deg_A_L[i]))+Dia_A_L_x[i][0],(Diagonal_Length_A_L[i]-Length_Hin
ge)*(cos(Dia_Deg_A_L[i]))+Dia_A_L_x[i][0]
  Dia_A_conect_L_y[i]                                                    =
Length_Hinge*(sin(Dia_Deg_A_L[i]))+Dia_A_L_y[i][0],(Diagonal_Length_A_L[i]-Length_Hin
ge)*(sin(Dia_Deg_A_L[i]))+Dia_A_L_y[i][0]
  Dia_A_conect_L_z[i] = 0,0

Dia_A_conect_R_x=[(0,0)]*(Number_of_Dia_A_R)
Dia_A_conect_R_y=[(0,0)]*(Number_of_Dia_A_R)
Dia_A_conect_R_z=[(0,0)]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):
  Dia_A_conect_R_x[i]                                                    =
Length_Hinge*(cos(Dia_Deg_A_R[i]))+Dia_A_R_x[i][0],(Diagonal_Length_A_R[i]-Length_Hin
ge)*(cos(Dia_Deg_A_R[i]))+Dia_A_R_x[i][0]
  Dia_A_conect_R_y[i]                                                    =
-Length_Hinge*(sin(Dia_Deg_A_R[i]))++Dia_A_R_y[i][0],-(Diagonal_Length_A_R[i]-Length_
Hinge)*(sin(Dia_Deg_A_R[i]))++Dia_A_R_y[i][0]
  Dia_A_conect_R_z[i] = 0,0

Dia_B_conect_L_x=[(0,0)]*(Number_of_Dia_B_L)
Dia_B_conect_L_y=[(0,0)]*(Number_of_Dia_B_L)
Dia_B_conect_L_z=[(0,0)]*(Number_of_Dia_B_L)
```

```
for i in range(Number_of_Dia_B_L):
  Dia_B_conect_L_x[i] = Length_of_Truss,Length_of_Truss
  Dia_B_conect_L_y[i] =
Length_Hinge*(sin(Dia_Deg_B_L[i]))+Dia_B_L_y[i][0],(Diagonal_Length_B_L[i]-Length_Hin
ge)*(sin(Dia_Deg_B_L[i]))+Dia_B_L_y[i][0]
  Dia_B_conect_L_z[i] =
-Length_Hinge*(cos(Dia_Deg_B_L[i]))+Dia_B_L_z[i][0],-(Diagonal_Length_B_L[i]-Length_Hi
nge)*(cos(Dia_Deg_B_L[i]))+Dia_B_L_z[i][0]

Dia_B_conect_R_x=[(0,0)]*(Number_of_Dia_B_R)
Dia_B_conect_R_y=[(0,0)]*(Number_of_Dia_B_R)
Dia_B_conect_R_z=[(0,0)]*(Number_of_Dia_B_R)
for i in range(Number_of_Dia_B_R):
  Dia_B_conect_R_x[i] = Length_of_Truss,Length_of_Truss
  Dia_B_conect_R_y[i] =
-Length_Hinge*(sin(Dia_Deg_B_R[i]))+Dia_B_R_y[i][0],-(Diagonal_Length_B_R[i]-Length_Hi
nge)*(sin(Dia_Deg_B_R[i]))+Dia_B_R_y[i][0]
  Dia_B_conect_R_z[i] =
-Length_Hinge*(cos(Dia_Deg_B_R[i]))+Dia_B_R_z[i][0],-(Diagonal_Length_B_R[i]-Length_H
inge)*(cos(Dia_Deg_B_R[i]))+Dia_B_R_z[i][0]

Dia_C_conect_L_x=[(0,0)]*(Number_of_Dia_C_L)
Dia_C_conect_L_y=[(0,0)]*(Number_of_Dia_C_L)
Dia_C_conect_L_z=[(0,0)]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):
  Dia_C_conect_L_x[i] =
Length_Hinge*(cos(Dia_Deg_C_L[i]))+Dia_C_L_x[i][0],(Diagonal_Length_C_L[i]-Length_Hin
ge)*(cos(Dia_Deg_C_L[i]))+Dia_C_L_x[i][0]
  Dia_C_conect_L_y[i] =
Length_Hinge*(sin(Dia_Deg_C_L[i]))+Dia_C_L_y[i][0],(Diagonal_Length_C_L[i]-Length_Hin
ge)*(sin(Dia_Deg_C_L[i]))+Dia_C_L_y[i][0]
  Dia_C_conect_L_z[i] = -Width_of_Truss,-Width_of_Truss
Dia_C_conect_R_x=[(0,0)]*(Number_of_Dia_C_R)
Dia_C_conect_R_y=[(0,0)]*(Number_of_Dia_C_R)
Dia_C_conect_R_z=[(0,0)]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):
  Dia_C_conect_R_x[i] =
Length_Hinge*(cos(Dia_Deg_C_R[i]))+Dia_C_R_x[i][0],(Diagonal_Length_C_R[i]-Length_Hi
nge)*(cos(Dia_Deg_C_R[i]))+Dia_C_R_x[i][0]
  Dia_C_conect_R_y[i] =
-Length_Hinge*(sin(Dia_Deg_C_R[i]))+Dia_C_R_y[i][0],-(Diagonal_Length_C_R[i]-Length_H
inge)*(sin(Dia_Deg_C_R[i]))+Dia_C_R_y[i][0]
  Dia_C_conect_R_z[i] = -Width_of_Truss,-Width_of_Truss

Dia_D_conect_L_x=[(0,0)]*(Number_of_Dia_D_L)
Dia_D_conect_L_y=[(0,0)]*(Number_of_Dia_D_L)
Dia_D_conect_L_z=[(0,0)]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):
  Dia_D_conect_L_x[i] = 0,0
  Dia_D_conect_L_y[i] =
Length_Hinge*(sin(Dia_Deg_D_L[i]))+Dia_D_L_y[i][0],(Diagonal_Length_D_L[i]-Length_Hin
ge)*(sin(Dia_Deg_D_L[i]))+Dia_D_L_y[i][0]
  Dia_D_conect_L_z[i] =
-Length_Hinge*(cos(Dia_Deg_D_L[i]))+Dia_D_L_z[i][0],-(Diagonal_Length_D_L[i]-Length_H
inge)*(cos(Dia_Deg_D_L[i]))+Dia_D_L_z[i][0]

Dia_D_conect_R_x=[(0,0)]*(Number_of_Dia_D_R)
```

```
Dia_D_conect_R_y=[(0,0)]*(Number_of_Dia_D_R)
Dia_D_conect_R_z=[(0,0)]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):
  Dia_D_conect_R_x[i] = 0,0
  Dia_D_conect_R_y[i] =
-Length_Hinge*(sin(Dia_Deg_D_R[i]))+Dia_D_R_y[i][0],-(Diagonal_Length_D_R[i]-Length_
Hinge)*(sin(Dia_Deg_D_R[i]))+Dia_D_R_y[i][0]
  Dia_D_conect_R_z[i] =
-Length_Hinge*(cos(Dia_Deg_D_R[i]))+Dia_D_R_z[i][0],-(Diagonal_Length_D_R[i]-Length_
Hinge)*(cos(Dia_Deg_D_R[i]))+Dia_D_R_z[i][0]




# Dia_A_L=[0,3,6,9,16,19]
# Dia_A_R=[1,2,8,11,18]
# Dia_B_L=[0,3,6,9,13,15]
# Dia_B_R=[1,2,5,14]
# Dia_C_L=[0,3,6,9,12,17]
# Dia_C_R=[1,2,8,11,18]
# Dia_D_L=[0,3,6,9,11]
# Dia_D_R=[1,2,8]


Dig_connect_colum_A_L=[(0,0)]*(Number_of_Dia_A_L)
for i, j in enumerate(Dia_A_L):

Dig_connect_colum_A_L[i]=(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+j%(Number_of_Column_in_A_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_pl
ane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in
_D_plane+4)+j%(Number_of_Column_in_A_plane+1)+1
Dig_connect_colum_A_R=[(0,0)]*(Number_of_Dia_A_R)
for i, j in enumerate(Dia_A_R):

Dig_connect_colum_A_R[i]=(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+j%(Number_of_Column_in_A_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_pl
ane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in
_D_plane+4)+j%(Number_of_Column_in_A_plane+1)+1

Dig_connect_colum_B_L=[(0,0)]*(Number_of_Dia_B_L)
for i, j in enumerate(Dia_B_L):

Dig_connect_colum_B_L[i]=(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+1+j%(Number_of_Column_in_B_plane+1),(Number_of_Lay
er+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column
_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+j%(Nu
mber_of_Column_in_B_plane+1)+1
Dig_connect_colum_B_R=[(0,0)]*(Number_of_Dia_B_R)
for i, j in enumerate(Dia_B_R):

Dig_connect_colum_B_R[i]=(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+1+j%(Number_of_Column_in_B_plane+1),(Number_of_Lay
```

er+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+j%(Number_of_Column_in_B_plane+1)+1*

Dig_connect_colum_C_L=[(0,0)](Number_of_Dia_C_L)*
for i, j in enumerate(Dia_C_L):

Dig_connect_colum_C_L[i]=(Number_of_Layer+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1-j%(Number_of_Column_in_C_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1-j%(Number_of_Column_in_C_plane+1)-1*
Dig_connect_colum_C_R=[(0,0)](Number_of_Dia_C_R)*
for i, j in enumerate(Dia_C_R):

Dig_connect_colum_C_R[i]=(Number_of_Layer+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1-j%(Number_of_Column_in_C_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1-j%(Number_of_Column_in_C_plane+1)-1*

Dig_connect_colum_D_L=[(0,0)](Number_of_Dia_D_L)*
for i, j in enumerate(Dia_D_L):

Dig_connect_colum_D_L[i]=(Number_of_Layer+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_D_plane+1-j%(Number_of_Column_in_D_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_D_plane+1-j%(Number_of_Column_in_D_plane+1)-1*

Dig_connect_colum_D_R=[(0,0)](Number_of_Dia_D_R)*
for i, j in enumerate(Dia_D_R):

Dig_connect_colum_D_R[i]=(Number_of_Layer+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_D_plane+1-j%(Number_of_Column_in_D_plane+1),(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_D_plane+1-j%(Number_of_Column_in_D_plane+1)-1*

for i in range(Number_of_Dia_D_L):
if Dig_connect_colum_D_L[i][0] > (Number_of_Layer+1)(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu*

```
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_
Column_in_D_plane:
    Dig_connect_colum_D_L[i] =
((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4),Dig_connect_colum_D_L[i]
[1])


for i in range(Number_of_Dia_D_R):
  if Dig_connect_colum_D_R[i][0] >
(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+Number_of_
Column_in_D_plane:
    Dig_connect_colum_D_R[i] =
((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4),Dig_connect_colum_D_R[i]
[1])
#
#Plate
#

Number_of_CLT_Plates = (Length_of_Truss/Len_Pla)
Num_Pla=int(Number_of_CLT_Plates)
Plate_member_coord_x=[(0,0)]*(Num_Pla)




Number_of_Timber=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
  Number_of_Timber[i]=int(Lx_A[i]/Len_Pla)

Total_Number_of_Timber=[0]*(Number_of_Column_in_A_plane+2)
sum11=0
for i in range(Number_of_Column_in_A_plane+1):
  sum11+=Number_of_Timber[i]
  Total_Number_of_Timber[i+1]=sum11

for i in range(Num_Pla):
    Plate_member_coord_x[i]=Length_of_plate*i,Length_of_plate*(i+1)

Plate_member_coord_y=[(0,0)]*(Number_of_Layer+1)
for i in range(Number_of_Layer+1):
    Plate_member_coord_y[i]=0,Ly1*(Number_of_Layer+1-i)+(Width_of_section/2)



Plate_member_coord_x_=[(0,0)]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    Plate_member_coord_x_[i]=[0]*(100)


for m in range(Number_of_Column_in_A_plane+1):
```

```
  for i in range(Number_of_Timber[m]):

Plate_member_coord_x_[m][i]=(Total_Number_of_Timber[m]+i)*Length_of_plate,(Total_Numb
er_of_Timber[m]+i+1)*Length_of_plate


Plate_member_coord_x_1=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    Plate_member_coord_x_1[i]=filter(lambda x: x != 0, Plate_member_coord_x_[i])




#
#Coordinates for connections to CLT and Pin
#

CLT_Pin_Coord_x=[(0)]*(Num_Pla*(Number_of_partition_in_CLT_V+1))
for i in range(Num_Pla*(Number_of_partition_in_CLT_V+1)):
    CLT_Pin_Coord_x[i]=(Length_of_plate/(Number_of_partition_in_CLT_V+1))*(i+1)




CLT_Pin_Coord_x_=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    CLT_Pin_Coord_x_[i]=[0]*(100)


for m in range(Number_of_Column_in_A_plane+1):
  for i in range(Number_of_Timber[m]*(Number_of_partition_in_CLT_V+1)-1):

CLT_Pin_Coord_x_[m][i]=(Total_Number_of_Timber[m]*4+i+1)*(Length_of_plate/(Number_of
_partition_in_CLT_V+1))

CLT_Pin_Coord_x_1=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    CLT_Pin_Coord_x_1[i]=filter(lambda x: x != 0, CLT_Pin_Coord_x_[i])

CLT_Pin_Coord_y=[(0)]*(Number_of_Layer+1)
for i in range(Number_of_Layer+1):
   CLT_Pin_Coord_y[i]=Ly1*(Number_of_Layer+1-i)

CLT_Pin_Coord_z=[(0)]*(2)
for i in range(2):
   CLT_Pin_Coord_z[i]=-Width_of_Truss*i



CLT_Pin_Coord_z_1=[0]*(Number_of_Column_in_B_plane+1)

#Wind Load Calculation
#
Z_0=Zo[Type_of_terrain]
Terrain_factor=0.19*pow((Z_0[0]/Zo[2][0]),0.07)
Standard_deviation_of_the_turbulence=Terrain_factor*Basic_wind_velocity*Turblence_factor
```

```
H_int=int(Height_of_Truss)
Roughness_factor=[0]*(H_int+1)



for i in range(Z_0[1]+1):
    Roughness_factor[i]=Terrain_factor*math.log((Z_0[1]/Z_0[0]))
for i in range(H_int-Z_0[1]+1):
    Roughness_factor[i+Z_0[1]]=Terrain_factor*math.log((i+Z_0[1])/Z_0[0])
Mean_wind_velocity=[0]*(H_int+1)
for i in range(H_int+1):
    Mean_wind_velocity[i]=Roughness_factor[i]*Orography_factor*Basic_wind_velocity
Turbulence_intensity=[0]*(H_int+1)
for i in range(Z_0[1]+1):

Turbulence_intensity[i]=Standard_deviation_of_the_turbulence/(Mean_wind_velocity[Z_0[1]])
for i in range(H_int-Z_0[1]+1):

Turbulence_intensity[i+Z_0[1]]=Standard_deviation_of_the_turbulence/(Mean_wind_velocity[i+
Z_0[1]])

Characteristic_peak_velocity_pressure=[0]*(H_int+1)
for i in range(H_int+1):

Characteristic_peak_velocity_pressure[i]=(1+7*(Turbulence_intensity[i]))*0.5*Air_density*pow
(Mean_wind_velocity[i],2)

H= Height_of_Truss*10*Scale_factor
B= Length_of_Truss*10*Scale_factor
D= Width_of_Truss*10*Scale_factor
Windward_pressure_factor=0.8
if H/D <= 0.25:
    Windward_pressure_factor=0.7
if 0.25<= H/D <= 1:
    Windward_pressure_factor=0.7+((0.8-0.7)/(1-0.25))*(H/D-0.25)

Leeward_pressure_factor=0.7
if H/D <= 0.25:
    Leeward_pressure_factor=0.3
if 0.25<= H/D <= 1:
    Leeward_pressure_factor=0.3+((0.5-0.3)/(1-0.25))*(H/D-0.25)
if 1<= H/D <= 5:
    Leeward_pressure_factor=0.5+((0.7-0.5)/(5-1))*(H/D-1)

Correlation_factor=1.0
if H/D <= 1:
    Correlation_factor=0.85
if 1<= H/D <= 5:
    Correlation_factor=0.85+((1-0.85)/(5-1))*(H/D-1)

Wind_pressure_Windward=[0]*(H_int+1)
Wind_pressure_Leeward=[0]*(H_int+1)
for i in range(H_int+1):

Wind_pressure_Windward[i]=Windward_pressure_factor*Correlation_factor*Characteristic_pea
k_velocity_pressure[i]
```

```
Wind_pressure_Leeward[i]=Leeward_pressure_factor*Correlation_factor*Characteristic_peak_velocity_pressure[i]


#
# Parameters for partition
#
Crossbeam_partition_parameter_1=[(0,0)]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):

Crossbeam_partition_parameter_1[i]=(abs(0-(Length_of_connection/((Lx_A[i]-Width_of_section)-(Length_of_connection*0)))),abs(1-(Length_of_connection/((Lx_A[i]-Width_of_section)-(Length_of_connection*1)))))

Crossbeam_partition_parameter_2=[0]*(Number_of_Column_in_B_plane+1)
for n in range(2):
  for i in range(Number_of_Column_in_B_plane+1):

Crossbeam_partition_parameter_2[i]=(abs(0-(Length_of_connection_2/((Lz_B[i]-Length_of_section)-(Length_of_connection_2*0)))),abs(1-(Length_of_connection_2/((Lz_B[i]-Length_of_section)-(Length_of_connection_2*1)))))

Crossbeam_partition_parameter_3=[0]*(Number_of_Column_in_C_plane+1)
for n in range(2):
  for i in range(Number_of_Column_in_C_plane+1):

Crossbeam_partition_parameter_3[i]=(abs(0-(Length_of_connection/((Lx_C[i]-Width_of_section)-(Length_of_connection*0)))),abs(1-(Length_of_connection/((Lx_C[i]-Width_of_section)-(Length_of_connection*1)))))

Crossbeam_partition_parameter_4=[0]*(Number_of_Column_in_D_plane+1)
for n in range(2):
  for i in range(Number_of_Column_in_D_plane+1):

Crossbeam_partition_parameter_4[i]=(abs(0-(Length_of_connection_2/((Lz_D[i]-Length_of_section)-(Length_of_connection_2*0)))),abs(1-(Length_of_connection_2/((Lz_D[i]-Length_of_section)-(Length_of_connection_2*1)))))

Crossbeam_partition_coords_x_1=[0,0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
  Crossbeam_partition_coords_x_1[i] =
[Length_of_connection,Lx_A[i]-Width_of_section-Length_of_connection]

Crossbeam_partition_coords_x_2=[0,0]*(Number_of_Column_in_B_plane+1)
for i in range(Number_of_Column_in_B_plane+1):
  Crossbeam_partition_coords_x_2[i] =
[Length_of_connection_2,Lz_B[i]-Length_of_section-Length_of_connection_2]

Crossbeam_partition_coords_x_3=[0,0]*(Number_of_Column_in_C_plane+1)
for i in range(Number_of_Column_in_C_plane+1):
  Crossbeam_partition_coords_x_3[i] =
[Length_of_connection,Lx_C[i]-Width_of_section-Length_of_connection]

Crossbeam_partition_coords_x_4=[0,0]*(Number_of_Column_in_D_plane+1)
for i in range(Number_of_Column_in_D_plane+1):
  Crossbeam_partition_coords_x_4[i] =
```

```
[Length_of_connection_2,Lz_D[i]-Length_of_section-Length_of_connection_2]

Crossbeam_partition_coords_y=[0]*(Number_of_Layer+1)
for i in range(Number_of_Layer+1):
   Crossbeam_partition_coords_y[i]=(0.0)

Diagonal_partition_parameter_A_L=[(0,0)]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):
   Diagonal_partition_parameter_A_L[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_A_L[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_A_L[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_A_R=[(0,0)]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):
   Diagonal_partition_parameter_A_R[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_A_R[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_A_R[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_B_L=[(0,0)]*(Number_of_Dia_B_L)
for i in range(Number_of_Dia_B_L):
   Diagonal_partition_parameter_B_L[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_B_L[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_B_L[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_B_R=[(0,0)]*(Number_of_Dia_B_R)
for i in range(Number_of_Dia_B_R):
   Diagonal_partition_parameter_B_R[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_B_R[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_B_R[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_C_L=[(0,0)]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):
   Diagonal_partition_parameter_C_L[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_C_L[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_C_L[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_C_R=[(0,0)]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):
   Diagonal_partition_parameter_C_R[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_C_R[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_C_R[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_D_L=[(0,0)]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):
   Diagonal_partition_parameter_D_L[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_D_L[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_D_L[i])-(Length_Hinge*1)))))

Diagonal_partition_parameter_D_R=[(0,0)]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):
   Diagonal_partition_parameter_D_R[i]=
(abs(0-(Length_Hinge/((Diagonal_Length_D_R[i])-(Length_Hinge*0)))),abs(1-(Length_Hinge/((
Diagonal_Length_D_R[i])-(Length_Hinge*1)))))

Crossbeam_partition_coords_x_A_L=[0,0]*(Number_of_Dia_A_L)
Crossbeam_partition_coords_y_A_L=[0,0]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):
```

```
  Crossbeam_partition_coords_x_A_L[i]                                    =
[Length_Hinge*(cos(Dia_Deg_A_L[i])),(Diagonal_Length_A_L[i]-Length_Hinge)*(cos(Dia_Deg
_A_L[i]))]
  Crossbeam_partition_coords_y_A_L[i]                                    =
[Length_Hinge*(sin(Dia_Deg_A_L[i])),(Diagonal_Length_A_L[i]-Length_Hinge)*(sin(Dia_Deg
_A_L[i]))]

Crossbeam_partition_coords_x_B_L=[0,0]*(Number_of_Dia_B_L)
Crossbeam_partition_coords_y_B_L=[0,0]*(Number_of_Dia_B_L)
for i in range(Number_of_Dia_B_L):
  Crossbeam_partition_coords_x_B_L[i]                                    =
[Length_Hinge*(cos(Dia_Deg_B_L[i])),(Diagonal_Length_B_L[i]-Length_Hinge)*(cos(Dia_Deg
_B_L[i]))]
  Crossbeam_partition_coords_y_B_L[i]                                    =
[Length_Hinge*(sin(Dia_Deg_B_L[i])),(Diagonal_Length_B_L[i]-Length_Hinge)*(sin(Dia_Deg
_B_L[i]))]

Crossbeam_partition_coords_x_C_L=[0,0]*(Number_of_Dia_C_L)
Crossbeam_partition_coords_y_C_L=[0,0]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):
  Crossbeam_partition_coords_x_C_L[i]                                    =
[Length_Hinge*(cos(Dia_Deg_C_L[i])),(Diagonal_Length_C_L[i]-Length_Hinge)*(cos(Dia_De
g_C_L[i]))]
  Crossbeam_partition_coords_y_C_L[i]                                    =
[Length_Hinge*(sin(Dia_Deg_C_L[i])),(Diagonal_Length_C_L[i]-Length_Hinge)*(sin(Dia_Deg
_C_L[i]))]

Crossbeam_partition_coords_x_D_L=[0,0]*(Number_of_Dia_D_L)
Crossbeam_partition_coords_y_D_L=[0,0]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):
  Crossbeam_partition_coords_x_D_L[i]                                    =
[Length_Hinge*(cos(Dia_Deg_D_L[i])),(Diagonal_Length_D_L[i]-Length_Hinge)*(cos(Dia_De
g_D_L[i]))]
  Crossbeam_partition_coords_y_D_L[i]                                    =
[Length_Hinge*(sin(Dia_Deg_D_L[i])),(Diagonal_Length_D_L[i]-Length_Hinge)*(sin(Dia_Deg
_D_L[i]))]


Crossbeam_partition_coords_x_A_R=[0,0]*(Number_of_Dia_A_R)
Crossbeam_partition_coords_y_A_R=[0,0]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):
  Crossbeam_partition_coords_x_A_R[i]                                    =
[Length_Hinge*(cos(Dia_Deg_A_R[i])),(Diagonal_Length_A_R[i]-Length_Hinge)*(cos(Dia_De
g_A_R[i]))]
  Crossbeam_partition_coords_y_A_R[i]                                    =
[-Length_Hinge*(sin(Dia_Deg_A_R[i])),-(Diagonal_Length_A_R[i]-Length_Hinge)*(sin(Dia_De
g_A_R[i]))]

Crossbeam_partition_coords_x_B_R=[0,0]*(Number_of_Dia_B_R)
Crossbeam_partition_coords_y_B_R=[0,0]*(Number_of_Dia_B_R)
for i in range(Number_of_Dia_B_R):
  Crossbeam_partition_coords_x_B_R[i]                                    =
[Length_Hinge*(cos(Dia_Deg_B_R[i])),(Diagonal_Length_B_R[i]-Length_Hinge)*(cos(Dia_De
g_B_R[i]))]
  Crossbeam_partition_coords_y_B_R[i]                                    =
[-Length_Hinge*(sin(Dia_Deg_B_R[i])),-(Diagonal_Length_B_R[i]-Length_Hinge)*(sin(Dia_De
g_B_R[i]))]
```

```
Crossbeam_partition_coords_x_C_R=[0,0]*(Number_of_Dia_C_R)
Crossbeam_partition_coords_y_C_R=[0,0]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):
  Crossbeam_partition_coords_x_C_R[i] = [Length_Hinge*(cos(Dia_Deg_C_R[i])),(Diagonal_Length_C_R[i]-Length_Hinge)*(cos(Dia_Deg_C_R[i]))]
  Crossbeam_partition_coords_y_C_R[i] = [-Length_Hinge*(sin(Dia_Deg_C_R[i])),-(Diagonal_Length_C_R[i]-Length_Hinge)*(sin(Dia_Deg_C_R[i]))]

Crossbeam_partition_coords_x_D_R=[0,0]*(Number_of_Dia_D_R)
Crossbeam_partition_coords_y_D_R=[0,0]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):
  Crossbeam_partition_coords_x_D_R[i] = [Length_Hinge*(cos(Dia_Deg_D_R[i])),(Diagonal_Length_D_R[i]-Length_Hinge)*(cos(Dia_Deg_D_R[i]))]
  Crossbeam_partition_coords_y_D_R[i] = [-Length_Hinge*(sin(Dia_Deg_D_R[i])),-(Diagonal_Length_D_R[i]-Length_Hinge)*(sin(Dia_Deg_D_R[i]))]


Vertical_frame_partition_parameter_1=[0]*(1)

Vertical_frame_partition_parameter_1=Height_of_Truss/(Height_of_Truss+(Length_of_section/2))

Vertical_frame_partition_parameter_2=[0]*(Number_of_Layer)
for i in range(Number_of_Layer):
  Vertical_frame_partition_parameter_2[i]=(Ly1/((Number_of_Layer+1-i)*Ly1))

Vertical_partition_coords_x = [0.0,Length_of_Truss,Length_of_Truss,0.0]

Vertical_partition_coords_y=[0]*(Number_of_Layer+1)
for i in range(Number_of_Layer+1):
  Vertical_partition_coords_y[i] = ((i+1)*Ly1)

#CLT Assembly

Vertical_partition_CLT_coord_x=[0]*(Number_of_partition_in_CLT_V)
for i in range(Number_of_partition_in_CLT_V):

Vertical_partition_CLT_coord_x[i]=(Length_of_plate/(Number_of_partition_in_CLT_V+1))*(i+1)

Number_of_partition_in_CLT_H=int(Wid_Pla)-1
Horizentional_partition_CLT_coord_y=[0]*(Number_of_partition_in_CLT_H)
for i in range(Number_of_partition_in_CLT_H):

Horizentional_partition_CLT_coord_y[i]=((Wid_Pla-Width_of_section)/(Number_of_partition_in_CLT_H+1))*(i+1)+(Width_of_section/2)

Number_of_Expar_CLT=2
Ex_partition_CLT_coord_y=[Width_of_section/2,Width_of_plate-(Width_of_section/2)]


#Partion parameter for assembly of CLT
```

```
CLT_Based_Partiton_parameter_=[0]*(Number_of_Column_in_A_plane+1)
CLT_Based_Partiton_coord_x_=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
  CLT_Based_Partiton_parameter_[i]=[0]*(100)
  CLT_Based_Partiton_coord_x_[i]=[0]*(100)


for m in range(Number_of_Column_in_A_plane+1):
  for i in range(Number_of_Timber[m]*(Number_of_partition_in_CLT_V+1)-1):

CLT_Based_Partiton_parameter_[m][i]=(Length_of_plate/(Number_of_partition_in_CLT_V+1))
/(Lx_A[m]-Width_of_section-((Length_of_plate/(Number_of_partition_in_CLT_V+1))-(Width_of_
section/2))-((Length_of_plate/(Number_of_partition_in_CLT_V+1))*(i-1)))

CLT_Based_Partiton_parameter_[m][0]=((Length_of_plate/(Number_of_partition_in_CLT_V+1
))-(Width_of_section/2))/(Lx_A[m]-Width_of_section)
CLT_Based_Partiton_parameter_1=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
   CLT_Based_Partiton_parameter_1[i] = filter(lambda x: x != 0,
CLT_Based_Partiton_parameter_[i])


for m in range(Number_of_Column_in_A_plane+1):
  for i in range(Number_of_Timber[m]*(Number_of_partition_in_CLT_V+1)-1):

CLT_Based_Partiton_coord_x_[m][i]=(((Length_of_plate/(Number_of_partition_in_CLT_V+1))-
(Width_of_section/2))+(Length_of_plate/(Number_of_partition_in_CLT_V+1))*i)
CLT_Based_Partiton_coord_x_1=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
   CLT_Based_Partiton_coord_x_1[i] = filter(lambda x: x != 0,
CLT_Based_Partiton_coord_x_[i])


CLT_Based_Partiton_parameter_2_=[0]*(Number_of_Column_in_D_plane+1)
CLT_Based_Partiton_coord_2_x_=[0]*(Number_of_Column_in_D_plane+1)
for i in range(Number_of_Column_in_D_plane+1):
  CLT_Based_Partiton_parameter_2_[i]=[0]*(100)
  CLT_Based_Partiton_coord_2_x_[i]=[0]*(100)

for m in range(Number_of_Column_in_D_plane+1):
  for i in range(Lz_D[m]-1):

CLT_Based_Partiton_parameter_2_[m][i]=(Wid_Pla-Width_of_section)/(Number_of_partition_i
n_CLT_H+1)/(Lz_D[m]-Width_of_section-((Wid_Pla-Width_of_section)/(Number_of_partition_in
_CLT_H+1)-(Width_of_section/2))-(((Wid_Pla-Width_of_section)/(Number_of_partition_in_CLT_
H+1))*(i-1)))

CLT_Based_Partiton_parameter_2_[m][0]=((Wid_Pla-Width_of_section)/(Number_of_partition_
in_CLT_H+1)-(Width_of_section/2))/(Lz_D[m]-Width_of_section)
CLT_Based_Partiton_parameter_2_2=[0]*(Number_of_Column_in_D_plane+1)
for i in range(Number_of_Column_in_D_plane+1):
   CLT_Based_Partiton_parameter_2_2[i] = filter(lambda x: x != 0,
CLT_Based_Partiton_parameter_2_[i])



for m in range(Number_of_Column_in_D_plane+1):
```

```
  for i in range(Lz_D[m]-1):

CLT_Based_Partiton_coord_2_x_[m][i]=(((Wid_Pla-Width_of_section)/(Number_of_partition_in
_CLT_H+1)-(Width_of_section/2))+(Wid_Pla-Width_of_section)/(Number_of_partition_in_CLT_
H+1)*i)
CLT_Based_Partiton_coord_2_x_2=[0]*(Number_of_Column_in_D_plane+1)
for i in range(Number_of_Column_in_D_plane+1):
    CLT_Based_Partiton_coord_2_x_2[i] = filter(lambda x: x != 0,
CLT_Based_Partiton_coord_2_x_[i])




#
#Extra data
#
#

Plate_connection_coord_x=[(0)]*(Num_Pla-1)
for i in range(Num_Pla-1):
    Plate_connection_coord_x[i]=Length_of_plate*(i+1)

Plate_connection_coord_z=[(0)]*(Number_of_partition_in_CLT_H+Number_of_Expar_CLT)
for i in range(Number_of_partition_in_CLT_H+Number_of_Expar_CLT):

Plate_connection_coord_z[i]=-((Wid_Pla-Width_of_section)/(Number_of_partition_in_CLT_H+1
))*(i)




#
# Create Parts
#

for i in range(Number_of_Column_in_A_plane+1):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
      (Truss_member_coord_x[i][1]-Truss_member_coord_x[i][0]), 0.0))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-A-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-A-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Column_in_B_plane+1):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(

-1*(Truss_member_coord_z[Number_of_Column_in_A_plane+1+i][1]-Truss_member_coord_z[
Number_of_Column_in_A_plane+1+i][0])
, 0.0))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-B-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-B-'+str(i+1)].BaseWire(sketch=
```

```
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Column_in_C_plane+1):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(

(Truss_member_coord_x[Number_of_Column_in_A_plane+1+Number_of_Column_in_B_plane+
1+i][1]-Truss_member_coord_x[Number_of_Column_in_A_plane+1+Number_of_Column_in_B
_plane+1+i][0])
, 0.0))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-C-'+str(i+1), type=
        DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-C-'+str(i+1)].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Column_in_D_plane+1):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(

-1*(Truss_member_coord_z[Number_of_Column_in_A_plane+1+Number_of_Column_in_B_pla
ne+1+Number_of_Column_in_C_plane+1+i][1]-Truss_member_coord_z[Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i][0])
, 0.0))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-D-'+str(i+1), type=
        DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-D-'+str(i+1)].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,0.0),
point2=(0.0,Height_of_Truss+(Length_of_section/2)))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-'+str(3), type=
        DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-'+str(3)].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
        Length_Hinge,0.0))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-'+str(4), type=
        DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-'+str(4)].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

#Dia
for i in range(Number_of_Dia_A_L):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0,                    0),
point2=(Dia_A_L_x[i][1]-Dia_A_L_x[i][0], Dia_A_L_y[i][1]-Dia_A_L_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-A1-'+str(i+1), type=
        DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].BaseWire(sketch=
```

```
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
for i in range(Number_of_Dia_A_R):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_A_R_x[i][1]-Dia_A_R_x[i][0], Dia_A_R_y[i][1]-Dia_A_R_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-A2-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Dia_B_L):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_B_L_z[i][0]-Dia_B_L_z[i][1], Dia_B_L_y[i][1]-Dia_B_L_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-B1-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
for i in range(Number_of_Dia_B_R):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_B_R_z[i][0]-Dia_B_R_z[i][1], Dia_B_R_y[i][1]-Dia_B_R_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-B2-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Dia_C_L):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_C_L_x[i][1]-Dia_C_L_x[i][0], Dia_C_L_y[i][1]-Dia_C_L_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-C1-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
for i in range(Number_of_Dia_C_R):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_C_R_x[i][1]-Dia_C_R_x[i][0], Dia_C_R_y[i][1]-Dia_C_R_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-C2-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].BaseWire(sketch=
      mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

for i in range(Number_of_Dia_D_L):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_D_L_z[i][0]-Dia_D_L_z[i][1], Dia_D_L_y[i][1]-Dia_D_L_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-D1-'+str(i+1), type=
      DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].BaseWire(sketch=
```

```
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
for i in range(Number_of_Dia_D_R):
  mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
  mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0, 0),
point2=(Dia_D_R_z[i][0]-Dia_D_R_z[i][1], Dia_D_R_y[i][1]-Dia_D_R_y[i][0]))
  mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Dia-D2-'+str(i+1), type=
        DEFORMABLE_BODY)
  mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
#Plate

mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=30.0)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0.0, 0.0),
        point2=(Length_of_plate, Width_of_plate))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Plate-1', type=
        DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-Plate-1'].BaseShell(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=30.0)
# mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-Width_of_section/2, 0.0),
        # point2=(Length_of_plate, Width_of_plate))
# mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Plate-2', type=
        # DEFORMABLE_BODY)
# mdb.models['Model-1'].parts['Part-Plate-2'].BaseShell(sketch=
        # mdb.models['Model-1'].sketches['__profile__'])
# del mdb.models['Model-1'].sketches['__profile__']
#Connections for CLT and frame
mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0*Height_of_Truss)
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
        0.0, (Width_of_section/2)))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Part-Connec-CLT', type=
        DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-Connec-CLT'].BaseWire(sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']




#
# Create material
#
mdb.models['Model-1'].Material(name='Material_C24')
mdb.models['Model-1'].materials['Material_C24'].Elastic(moduli=INSTANTANEOUS,
        table=((El_TopBott_k, Er_TopBott_k, Et_TopBott_k, nylr_TopBott_k, nylt_TopBott_k,
nyrt_TopBott_k, Glr_TopBott_k, Glt_TopBott_k, Grt_TopBott_k), ),
type=ENGINEERING_CONSTANTS)
# mdb.models['Model-1'].Material(name='Material-C18')
# mdb.models['Model-1'].materials['Material-C18'].Elastic(moduli=INSTANTANEOUS,
        # table=((El_Diago_k, Er_Diago_k, Et_Diago_k, nylr_Diago_k, nylt_Diago_k,
```

```
nyrt_Diago_k, Glr_Diago_k, Glt_Diago_k, Grt_Diago_k), ),
type=ENGINEERING_CONSTANTS)
mdb.models['Model-1'].Material(name='Material_Steel')
mdb.models['Model-1'].materials['Material_Steel'].Elastic(table=((E_l, 0.3), ))

#
# Create cross section profiles
#
mdb.models['Model-1'].RectangularProfile(a=Length_of_section, b=Width_of_section, name=
    'Profile-Beam-1')
mdb.models['Model-1'].RectangularProfile(a=Width_of_section, b=Length_of_section, name=
    'Profile-Beam-2')
mdb.models['Model-1'].RectangularProfile(a=Width_of_pin, b=Length_of_pin, name=
    'Profile-Pin-1')


#
# Create beam sections
#
mdb.models['Model-1'].BeamSection(consistentMassMatrix=False, integration=
    DURING_ANALYSIS, material='Material_C24', name='Section-Beam-1',
    poissonRatio=0.0, profile='Profile-Beam-1', temperatureVar=LINEAR)
mdb.models['Model-1'].BeamSection(consistentMassMatrix=False, integration=
    DURING_ANALYSIS, material='Material_C24', name='Section-Beam-2',
    poissonRatio=0.0, profile='Profile-Beam-2', temperatureVar=LINEAR)
mdb.models['Model-1'].BeamSection(consistentMassMatrix=False, integration=
    DURING_ANALYSIS, material='Material_Steel', name='Section-Pin-1',
    poissonRatio=0.0, profile='Profile-Pin-1', temperatureVar=LINEAR)

#
# Create Plate sections
#

mdb.models['Model-1'].HomogeneousShellSection(idealization=NO_IDEALIZATION,
    material='Material_C24', name='Section-Plate-1', poissonDefinition=DEFAULT,
    preIntegrate=ON, thickness=Width_of_Crossection_plate, thicknessField='',
thicknessModulus=None,
    thicknessType=UNIFORM, useDensity=OFF)


#
# Assign beam section orientation
#
for i in range(Number_of_Column_in_A_plane+1):
   mdb.models['Model-1'].parts['Part-A-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-A-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Column_in_B_plane+1):
   mdb.models['Model-1'].parts['Part-B-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-B-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Column_in_C_plane+1):
   mdb.models['Model-1'].parts['Part-C-'+str(i+1)].assignBeamSectionOrientation(method=
```

```
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-C-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Column_in_D_plane+1):
   mdb.models['Model-1'].parts['Part-D-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-D-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+1,Height_of_Truss+1,0)))

mdb.models['Model-1'].parts['Part-3'].assignBeamSectionOrientation(method=
  N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-3'].edges.getByBoundingBox(0,0,0,Length_of_Truss+1
,Height_of_Truss+1,0)))

mdb.models['Model-1'].parts['Part-4'].assignBeamSectionOrientation(method=
  N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-4'].edges.getByBoundingBox(0,0,0,Length_of_Truss+1
,Height_of_Truss+1,0)))
mdb.models['Model-1'].parts['Part-Connec-CLT'].assignBeamSectionOrientation(method=
  N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Connec-CLT'].edges.getByBoundingBox(0,0,0,Length_
of_Truss+1,Height_of_Truss+1,0)))

for i in range(Number_of_Dia_A_L):

mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Dia_A_R):

mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].edges.getByBoundingBox(0,-1*(He
ight_of_Truss+1),0,Length_of_Truss+1,0,0)))

for i in range(Number_of_Dia_B_L):

mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Dia_B_R):

mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].edges.getByBoundingBox(0,-1*(He
ight_of_Truss+1),0,Length_of_Truss+1,0,0)))
```

```
for i in range(Number_of_Dia_C_L):

mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Dia_C_R):

mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].edges.getByBoundingBox(0,-1*(He
ight_of_Truss+1),0,Length_of_Truss+1,0,0)))

for i in range(Number_of_Dia_D_L):

mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+1,Height_of_Truss+1,0)))
for i in range(Number_of_Dia_D_R):

mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].edges.getByBoundingBox(0,-1*(H
eight_of_Truss+1),0,Length_of_Truss+1,0,0)))

mdb.models['Model-1'].parts['Part-Plate-1'].MaterialOrientation(
    additionalRotationType=ROTATION_NONE, axis=AXIS_3, fieldName='', localCsys=
    None, orientationType=GLOBAL, region=Region(

faces=mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_pla
te+1,Width_of_plate+1,0)))




#
# Assign sections
#

for i in range(Number_of_Column_in_A_plane+1):
  mdb.models['Model-1'].parts['Part-A-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-A-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Column_in_C_plane+1):
  mdb.models['Model-1'].parts['Part-C-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
```

```
edges=mdb.models['Model-1'].parts['Part-C-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Column_in_B_plane+1):
 mdb.models['Model-1'].parts['Part-B-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-B-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-2',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Column_in_D_plane+1):
 mdb.models['Model-1'].parts['Part-D-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-D-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_
of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-2',
thicknessAssignment=
    FROM_SECTION)

mdb.models['Model-1'].parts['Part-3'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-3'].edges.getByBoundingBox(0,0,0,Length_of_Truss+
Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-1', thicknessAssignment=
    FROM_SECTION)
mdb.models['Model-1'].parts['Part-4'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-4'].edges.getByBoundingBox(0,0,0,Length_of_Truss+
Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Pin-1', thicknessAssignment=
    FROM_SECTION)

for i in range(Number_of_Dia_A_L):
 mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Dia_A_R):
 mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].edges.getByBoundingBox(0,-Heigh
t_of_Truss-1,0,Length_of_Truss+Width_of_Truss,0,0)), sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)

for i in range(Number_of_Dia_B_L):
 mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+Width_of_Truss,Height_of_Truss+1,0)), sectionName='Section-Beam-1',
```

```
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Dia_B_R):
  mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].edges.getByBoundingBox(0,-Heigh
t_of_Truss-1,0,Length_of_Truss+Width_of_Truss,0,0)),          sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)

for i in range(Number_of_Dia_C_L):
  mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+Width_of_Truss,Height_of_Truss+1,0)),          sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Dia_C_R):
  mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].edges.getByBoundingBox(0,-Heigh
t_of_Truss-1,0,Length_of_Truss+Width_of_Truss,0,0)),          sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)

for i in range(Number_of_Dia_D_L):
  mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Le
ngth_of_Truss+Width_of_Truss,Height_of_Truss+1,0)),          sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)
for i in range(Number_of_Dia_D_R):
  mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].edges.getByBoundingBox(0,-Heigh
t_of_Truss-1,0,Length_of_Truss+Width_of_Truss,0,0)),          sectionName='Section-Beam-1',
thicknessAssignment=
    FROM_SECTION)

mdb.models['Model-1'].parts['Part-Plate-1'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

faces=mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_pla
te+1,Width_of_plate+1,0)), sectionName='Section-Plate-1', thicknessAssignment=
    FROM_SECTION)

# # mdb.models['Model-1'].parts['Part-Plate-2'].SectionAssignment(offset=0.0,
    # # offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    #                                                                                #
faces=mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_p
late+1,Width_of_plate+1,0)), sectionName='Section-Plate-1', thicknessAssignment=
```

```
    # # FROM_SECTION)

mdb.models['Model-1'].parts['Part-Connec-CLT'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(

edges=mdb.models['Model-1'].parts['Part-Connec-CLT'].edges.getByBoundingBox(-Width_of_se
ction,-Width_of_section,0,Width_of_section,Width_of_section,0)),  sectionName='Section-Pin-1',
thicknessAssignment=
    FROM_SECTION)









#
# Create partition of the frame
#


mdb.models['Model-1'].parts['Part-3'].PartitionEdgeByParam(edges=mdb.models['Model-1'].par
ts['Part-3'].edges.findAt(((0.0,Height_of_Truss,0.0),)),
parameter=Vertical_frame_partition_parameter_1)

for  i  in  range(Number_of_Layer):
mdb.models['Model-1'].parts['Part-3'].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-3'].edges.findAt(((Vertical_partition_coords_x[0],Vertical_par
tition_coords_y[i],0.0),)), parameter=Vertical_frame_partition_parameter_2[i])

# for i in range(Number_of_Column_in_A_plane+1):
  # for n in range(2):
    # mdb.models['Model-1'].parts['Part-A-'+str(i+1)].PartitionEdgeByParam(edges=
    #
mdb.models['Model-1'].parts['Part-A-'+str(i+1)].edges.findAt(((Crossbeam_partition_coords_x_
1[i][n],Crossbeam_partition_coords_y[0],0.0),)),
parameter=Crossbeam_partition_parameter_1[i][n])
for i in range(Number_of_Column_in_B_plane+1):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-B-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-B-'+str(i+1)].edges.findAt(((Crossbeam_partition_coords_x_
2[i][n],Crossbeam_partition_coords_y[0],0.0),)),
parameter=Crossbeam_partition_parameter_2[i][n])
# for i in range(Number_of_Column_in_C_plane+1):
  # for n in range(2):
    # mdb.models['Model-1'].parts['Part-C-'+str(i+1)].PartitionEdgeByParam(edges=
    #
mdb.models['Model-1'].parts['Part-C-'+str(i+1)].edges.findAt(((Crossbeam_partition_coords_x_
3[i][n],Crossbeam_partition_coords_y[0],0.0),)),
parameter=Crossbeam_partition_parameter_3[i][n])
for i in range(Number_of_Column_in_D_plane+1):
  for n in range(2):
```

```
   mdb.models['Model-1'].parts['Part-D-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-D-'+str(i+1)].edges.findAt(((Crossbeam_partition_coords_x_
4[i][n],Crossbeam_partition_coords_y[0],0.0),)),
parameter=Crossbeam_partition_parameter_4[i][n])

for i in range(Number_of_Dia_A_L):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_A_L[i][n],Crossbeam_partition_coords_y_A_L[i][n],0.0),)),
parameter=Diagonal_partition_parameter_A_L[i][n])
for i in range(Number_of_Dia_A_R):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_A_R[i][n],Crossbeam_partition_coords_y_A_R[i][n],0.0),)),
parameter=Diagonal_partition_parameter_A_R[i][n])
for i in range(Number_of_Dia_B_L):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_B_L[i][n],Crossbeam_partition_coords_y_B_L[i][n],0.0),)),
parameter=Diagonal_partition_parameter_B_L[i][n])
for i in range(Number_of_Dia_B_R):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_B_R[i][n],Crossbeam_partition_coords_y_B_R[i][n],0.0),)),
parameter=Diagonal_partition_parameter_B_R[i][n])

for i in range(Number_of_Dia_C_L):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_C_L[i][n],Crossbeam_partition_coords_y_C_L[i][n],0.0),)),
parameter=Diagonal_partition_parameter_C_L[i][n])
for i in range(Number_of_Dia_C_R):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_C_R[i][n],Crossbeam_partition_coords_y_C_R[i][n],0.0),)),
parameter=Diagonal_partition_parameter_C_R[i][n])
for i in range(Number_of_Dia_D_L):
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_D_L[i][n],Crossbeam_partition_coords_y_D_L[i][n],0.0),)),
parameter=Diagonal_partition_parameter_D_L[i][n])
for i in range(Number_of_Dia_D_R):
```

```
  for n in range(2):
    mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].edges.findAt(((Crossbeam_partition_coor
ds_x_D_R[i][n],Crossbeam_partition_coords_y_D_R[i][n],0.0),)),
parameter=Diagonal_partition_parameter_D_R[i][n])

#CLT Partition
for i in range(Number_of_partition_in_CLT_V):
  mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=

mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0), point1=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[0],0),
point2=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[1],0))

for i in range(Number_of_Expar_CLT):
  mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=

mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0),                    point1=(0,Ex_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Ex_partition_CLT_coord_y[i],0))

for i in range(Number_of_partition_in_CLT_H):
  mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=

mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0),                    point1=(0,Horizentional_partition_CLT_coord_y[i],0),
point2=(Vertical_partition_CLT_coord_x[0],Horizentional_partition_CLT_coord_y[i],0))
for i in range(Number_of_partition_in_CLT_H):
  mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=

mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0),
point1=(Vertical_partition_CLT_coord_x[2],Horizentional_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Horizentional_partition_CLT_coord_y[i],0))
# #
# #for better looking =。=
# for i in range(Number_of_partition_in_CLT_V):
  # mdb.models['Model-1'].parts['Part-Plate-2'].PartitionFaceByShortestPath(faces=
      #
mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_plate+1
,Width_of_plate+1,0),
point1=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[0],0),
point2=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[1],0))
# mdb.models['Model-1'].parts['Part-Plate-2'].PartitionFaceByShortestPath(faces=
      #
mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_plate+1
,Width_of_plate+1,0),                    point1=(0,Ex_partition_CLT_coord_y[0],0),
point2=(0,Ex_partition_CLT_coord_y[1],0))

# for i in range(Number_of_Expar_CLT):
  # mdb.models['Model-1'].parts['Part-Plate-2'].PartitionFaceByShortestPath(faces=
      #
mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_plate+1
,Width_of_plate+1,0),                    point1=(0,Ex_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Ex_partition_CLT_coord_y[i],0))
```

```
# for i in range(Number_of_partition_in_CLT_H):
  # mdb.models['Model-1'].parts['Part-Plate-2'].PartitionFaceByShortestPath(faces=
      #
mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_plate+1
,Width_of_plate+1,0),          point1=(0,Horizentional_partition_CLT_coord_y[i],0),
point2=(Vertical_partition_CLT_coord_x[0],Horizentional_partition_CLT_coord_y[i],0))
# for i in range(Number_of_partition_in_CLT_H):
  # mdb.models['Model-1'].parts['Part-Plate-2'].PartitionFaceByShortestPath(faces=
      #
mdb.models['Model-1'].parts['Part-Plate-2'].faces.getByBoundingBox(-10,0,0,Length_of_plate+1
,Width_of_plate+1,0),
point1=(Vertical_partition_CLT_coord_x[2],Horizentional_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Horizentional_partition_CLT_coord_y[i],0))


# # # for i in range(Number_of_partition_in_CLT_V):
  # # # mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=
      #                                  #                                  #
mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0), point1=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[0],0),
point2=(Vertical_partition_CLT_coord_x[i],Ex_partition_CLT_coord_y[1],0))

# # # for i in range(Number_of_Expar_CLT):
  # # # mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=
      #                                  #                                  #
mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0),          point1=(0,Ex_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Ex_partition_CLT_coord_y[i],0))

# # # for i in range(Number_of_partition_in_CLT_H):
  # # # mdb.models['Model-1'].parts['Part-Plate-1'].PartitionFaceByShortestPath(faces=
      #                                  #                                  #
mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_plate+1,W
idth_of_plate+1,0),          point1=(0,Horizentional_partition_CLT_coord_y[i],0),
point2=(Length_of_plate,Horizentional_partition_CLT_coord_y[i],0))

#Partition frame again for CLT
for m in range(Number_of_Column_in_A_plane+1):
  for n in range(Number_of_Timber[m]*(Number_of_partition_in_CLT_V+1)-1):
    mdb.models['Model-1'].parts['Part-A-'+str(m+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-A-'+str(m+1)].edges.findAt(((CLT_Based_Partiton_coord_x_
1[m][n],0.0,0.0),)), parameter=CLT_Based_Partiton_parameter_1[m][n])


for m in range(Number_of_Column_in_C_plane+1):
  for n in range(Number_of_Timber[m]*(Number_of_partition_in_CLT_V+1)-1):
    mdb.models['Model-1'].parts['Part-C-'+str(m+1)].PartitionEdgeByParam(edges=

mdb.models['Model-1'].parts['Part-C-'+str(m+1)].edges.findAt(((CLT_Based_Partiton_coord_x_
1[m][n],0.0,0.0),)), parameter=CLT_Based_Partiton_parameter_1[m][n])


# for m in range(Number_of_Column_in_D_plane+1):
  # for n in range(Lz_D[m]-1):
    # mdb.models['Model-1'].parts['Part-D-'+str(m+1)].PartitionEdgeByParam(edges=
    #
```

```
mdb.models['Model-1'].parts['Part-D-'+str(m+1)].edges.findAt(((CLT_Based_Partiton_coord_2_
x_2[m][n],0.0,0.0),)), parameter=CLT_Based_Partiton_parameter_2_2[m][n])



#
# Create assembly of the parts
#


# Number_of_Column_in_A_plane
# Number_of_Column_in_B_plane
# Number_of_Column_in_C_plane
# Number_of_Column_in_D_plane


#
(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_i
n_B_plane+Number_of_Column_in_D_plane+4)



mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-A-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i+1),
        part=mdb.models['Model-1'].parts['Part-A-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-A-ASS-'+str((m)*(Number_of_Column_in_
A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Colu
mn_in_D_plane+4)+i+1)].translate(vector=(Truss_member_coord_x[(m)*(Number_of_Column_
in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Co
lumn_in_D_plane+4)+i][0],

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_C_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-C-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_
in_A_plane+Number_of_Column_in_B_plane+2+i+1),
        part=mdb.models['Model-1'].parts['Part-C-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-C-ASS-'+str((m)*(Number_of_Column_in_
A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Colu
mn_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2+i+1)
].translate(vector=

(Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_
in_A_plane+Number_of_Column_in_B_plane+2+i][0],

Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
```

```
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_B_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-B-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_
in_A_plane+1+i+1),

part=mdb.models['Model-1'].parts['Part-B-'+str(i+1)]),mdb.models['Model-1'].rootAssembly.rot
ate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
        0.0),          axisPoint=(0.0,          0.0,          0.0),
instanceList=('Part-B-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+i+1),  )),mdb.models['Model-1'].rootAssembly.translate(instanceList=('Par
t-B-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number
_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane
+1+i+1), ),

vector=(Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in
_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_
Column_in_A_plane+1+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][0],
        Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][0]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_D_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-D-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_
in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i+1),

part=mdb.models['Model-1'].parts['Part-D-'+str(i+1)]),mdb.models['Model-1'].rootAssembly.rot
ate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
        0.0),          axisPoint=(0.0,          0.0,          0.0),
instanceList=('Part-D-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1
+i+1),  )),mdb.models['Model-1'].rootAssembly.translate(instanceList=('Part-D-ASS-'+str((m)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Col
umn_in_B_plane+1+Number_of_Column_in_C_plane+1+i+1), ),

vector=(Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in
_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_
Column_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+
1+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
```

```
n_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i][0],
        Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i][
0]))

for i in
range(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_i
n_B_plane+Number_of_Column_in_D_plane+4):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Colu
mn_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i+1),
        part=mdb.models['Model-1'].parts['Part-3']),
mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Number_of
_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Num
ber_of_Column_in_D_plane+4)+i+1)].translate(vector=(Truss_member_coord_x[(Number_of_L
ayer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Colu
mn_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],

Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0],
Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-A-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),
        part=mdb.models['Model-1'].parts['Part-4']),
mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+2*i+1)].translate(vector=(Lx_co_A[i],
Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]))
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-A-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),
        part=mdb.models['Model-1'].parts['Part-4']),
mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+2*i+2)].translate(vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+i][1],

Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][1],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][1]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_C_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
```

```
name='Part-4-C-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),
      part=mdb.models['Model-1'].parts['Part-4']),
mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+2*i+1)].translate(vector=(Lx_co_C[i],

Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i][0],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i][0]))
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-C-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),
      part=mdb.models['Model-1'].parts['Part-4']),
mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+2*i+2)].translate(vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_p
lane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_i
n_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2+i][1],

Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i][1],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+Number_of_Column_in_B_plane+2+i][1]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_B_plane+1):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
90.0, axisDirection=(0.0,Height_of_Truss,
      0.0),          axisPoint=(0.0,          0.0,          0.0),
instanceList=('Part-4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),  )),mdb.m
odels['Model-1'].rootAssembly.translate(instanceList=('Part-4-B-'+str((m)*(Number_of_Column_
in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Co
lumn_in_D_plane+4)+2*i+1), ),

vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_p
lane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][0],
Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+i][0],-Lz_co_B[i]))
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
90.0, axisDirection=(0.0,Height_of_Truss,
```

```
    0.0),                axisPoint=(0.0,                0.0,                0.0),
instanceList=('Part-4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),  )),mdb.m
odels['Model-1'].rootAssembly.translate(instanceList=('Part-4-B-'+str((m)*(Number_of_Column_
in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Co
lumn_in_D_plane+4)+2*i+2), ),

vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_p
lane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][1],
Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+i][1],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+i][1]))

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_D_plane+1):
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
90.0, axisDirection=(0.0,Height_of_Truss,
    0.0),                axisPoint=(0.0,                0.0,                0.0),
instanceList=('Part-4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C
_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+1),   )),mdb.
models['Model-1'].rootAssembly.translate(instanceList=('Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*i+1), ),

vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_p
lane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i][
0],
Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+Number_of_Column_in_B_plane+1+Number_of_Column_in_C_plane+1+i][0],-(Lz_c
o_D[i])))
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
90.0, axisDirection=(0.0,Height_of_Truss,
    0.0),                axisPoint=(0.0,                0.0,                0.0),
instanceList=('Part-4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C
_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*i+2),   )),mdb.
models['Model-1'].rootAssembly.translate(instanceList=('Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*i+2), ),

vector=(Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_p
lane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+i][
1],
```

```
Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+i][1],
Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_
plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+i][1]))

for i in range(Number_of_Dia_A_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-A-L-ASS-'+str(i+1),
      part=mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-A-L-ASS-'+str(i+1)].translate(vector=(
Dia_A_L_x[i][0],
      Dia_A_L_y[i][0],Dia_A_L_z[i][0]))
for i in range(Number_of_Dia_A_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-A-L-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(Dia_Deg_A_L[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-A-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-A-L-'+str(i+1)+'-1'].translate(vector=(Dia_A_L_x[i][0],
      Dia_A_L_y[i][0],Dia_A_L_z[i][0])),
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-A-L-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(Dia_Deg_A_L[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-A-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-A-L-'+str(i+1)+'-2'].translate(vector=(Dia_A_conect_L_x[i][1],
      Dia_A_conect_L_y[i][1],Dia_A_conect_L_z[i][1])),

for i in range(Number_of_Dia_A_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-A-R-ASS-'+str(i+1),
      part=mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-A-R-ASS-'+str(i+1)].translate(vector=(
Dia_A_R_x[i][0],
      Dia_A_R_y[i][0],Dia_A_R_z[i][0]))
for i in range(Number_of_Dia_A_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-A-R-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(-Dia_Deg_A_R[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-A-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-A-R-'+str(i+1)+'-1'].translate(vector=(Dia_A_R_x[i][0],
      Dia_A_R_y[i][0],Dia_A_R_z[i][0])),
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-A-R-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(-Dia_Deg_A_R[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-A-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-A-R-'+str(i+1)+'-2'].translate(vector=(Dia_A_conect_R_x[i][1],
      Dia_A_conect_R_y[i][1],Dia_A_conect_R_z[i][1])),
```

```
for i in range(Number_of_Dia_B_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-B-L-ASS-'+str(i+1),

part=mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)]),mdb.models['Model-1'].rootAssemb
ly.rotate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
      0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('Part-Dia-B-L-ASS-'+str(i+1), )),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-B-L-ASS-'+str(i+1)].translate(vector=(
Dia_B_L_x[i][0],
      Dia_B_L_y[i][0],Dia_B_L_z[i][0]))

for i in range(Number_of_Dia_B_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-B-L-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.rotate(ang
le=(Dia_Deg_B_L[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-B-L-'+str(i+1)+'-1'].translate(vector=(Dia_B_L_x[i][0],
      Dia_B_L_y[i][0],Dia_B_L_z[i][0])),
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-B-L-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.rotate(ang
le=(Dia_Deg_B_L[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-B-L-'+str(i+1)+'-2'].translate(vector=(Dia_B_conect_L_x[i][1],
      Dia_B_conect_L_y[i][1],Dia_B_conect_L_z[i][1])),

for i in range(Number_of_Dia_B_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-B-R-ASS-'+str(i+1),

part=mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)]),mdb.models['Model-1'].rootAssemb
ly.rotate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
      0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('Part-Dia-B-R-ASS-'+str(i+1), )),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-B-R-ASS-'+str(i+1)].translate(vector=(
Dia_B_R_x[i][0],
      Dia_B_R_y[i][0],Dia_B_R_z[i][0]))

for i in range(Number_of_Dia_B_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-B-R-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.rotate(ang
le=(-Dia_Deg_B_R[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-B-R-'+str(i+1)+'-1'].translate(vector=(Dia_B_R_x[i][0],
      Dia_B_R_y[i][0],Dia_B_R_z[i][0])),
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-B-R-'+str(i+1)+'-2',
```

```
part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.rotate(ang
le=(-Dia_Deg_B_R[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-B-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-B-R-'+str(i+1)+'-2'].translate(vector=(Dia_B_conect_R_x[i][1],
    Dia_B_conect_R_y[i][1],Dia_B_conect_R_z[i][1])),

for i in range(Number_of_Dia_C_L):
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-C-L-ASS-'+str(i+1),
    part=mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-C-L-ASS-'+str(i+1)].translate(vector=(
Dia_C_L_x[i][0],
    Dia_C_L_y[i][0],Dia_C_L_z[i][0]))

for i in range(Number_of_Dia_C_L):
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-C-L-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(Dia_Deg_C_L[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-C-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-C-L-'+str(i+1)+'-1'].translate(vector=(Dia_C_L_x[i][0],
    Dia_C_L_y[i][0],Dia_C_L_z[i][0])),
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-C-L-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(Dia_Deg_C_L[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-C-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-C-L-'+str(i+1)+'-2'].translate(vector=(Dia_C_conect_L_x[i][1],
    Dia_C_conect_L_y[i][1],Dia_C_conect_L_z[i][1])),

for i in range(Number_of_Dia_C_R):
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-C-R-ASS-'+str(i+1),
    part=mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)]),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-C-R-ASS-'+str(i+1)].translate(vector=(
Dia_C_R_x[i][0],
    Dia_C_R_y[i][0],Dia_C_R_z[i][0]))
for i in range(Number_of_Dia_C_R):
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-C-R-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(-Dia_Deg_C_R[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-C-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-C-R-'+str(i+1)+'-1'].translate(vector=(Dia_C_R_x[i][0],
    Dia_C_R_y[i][0],Dia_C_R_z[i][0])),
  mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-C-R-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(-Dia_Deg_C_R[i]/pi*180), axisDirection=(0, 0, 0+1), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-C-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
```

```
'Part-4-Dia-C-R-'+str(i+1)+'-2'].translate(vector=(Dia_C_conect_R_x[i][1],
      Dia_C_conect_R_y[i][1],Dia_C_conect_R_z[i][1])),

for i in range(Number_of_Dia_D_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-D-L-ASS-'+str(i+1),

part=mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)]),mdb.models['Model-1'].rootAssemb
ly.rotate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
      0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('Part-Dia-D-L-ASS-'+str(i+1), )),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-D-L-ASS-'+str(i+1)].translate(vector=(
Dia_D_L_x[i][0],
      Dia_D_L_y[i][0],Dia_D_L_z[i][0]))
for i in range(Number_of_Dia_D_L):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-D-L-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.rotate(an
gle=(Dia_Deg_D_L[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-L-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-D-L-'+str(i+1)+'-1'].translate(vector=(Dia_D_L_x[i][0],
      Dia_D_L_y[i][0],Dia_D_L_z[i][0])),
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-D-L-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.rotate(an
gle=(Dia_Deg_D_L[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-L-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances[
'Part-4-Dia-D-L-'+str(i+1)+'-2'].translate(vector=(Dia_D_conect_L_x[i][1],
      Dia_D_conect_L_y[i][1],Dia_D_conect_L_z[i][1])),

for i in range(Number_of_Dia_D_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-Dia-D-R-ASS-'+str(i+1),

part=mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)]),mdb.models['Model-1'].rootAssemb
ly.rotate(angle=90.0, axisDirection=(0.0,Height_of_Truss,
      0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('Part-Dia-D-R-ASS-'+str(i+1), )),
mdb.models['Model-1'].rootAssembly.instances['Part-Dia-D-R-ASS-'+str(i+1)].translate(vector=(
Dia_D_R_x[i][0],
      Dia_D_R_y[i][0],Dia_D_R_z[i][0]))

for i in range(Number_of_Dia_D_R):
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-D-R-'+str(i+1)+'-1',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.rotate(an
gle=(-Dia_Deg_D_R[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-R-'+str(i+1)+'-1', )),mdb.models['Model-1'].rootAssembly.instances
['Part-4-Dia-D-R-'+str(i+1)+'-1'].translate(vector=(Dia_D_R_x[i][0],
      Dia_D_R_y[i][0],Dia_D_R_z[i][0])),
```

```
   mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-4-Dia-D-R-'+str(i+1)+'-2',

part=mdb.models['Model-1'].parts['Part-4']),mdb.models['Model-1'].rootAssembly.rotate(angle=
(90), axisDirection=(0, Height_of_Truss, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.rotate(an
gle=(-Dia_Deg_D_R[i]/pi*180), axisDirection=(1, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-4-Dia-D-R-'+str(i+1)+'-2', )),mdb.models['Model-1'].rootAssembly.instances
['Part-4-Dia-D-R-'+str(i+1)+'-2'].translate(vector=(Dia_D_conect_R_x[i][1],
      Dia_D_conect_R_y[i][1],Dia_D_conect_R_z[i][1])),

for n in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[i]):
      mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-ASS-Plate-'+str(j+1)+'-'+str(i+1)+'-'+str(n+1),

part=mdb.models['Model-1'].parts['Part-Plate-1']),mdb.models['Model-1'].rootAssembly.rotate(a
ngle=(-90), axisDirection=(1.2, 0, 0), axisPoint=(0, 0, 0),
instanceList=('Part-ASS-Plate-'+str(j+1)+'-'+str(i+1)+'-'+str(n+1), )),mdb.models['Model-1'].ro
otAssembly.instances['Part-ASS-Plate-'+str(j+1)+'-'+str(i+1)+'-'+str(n+1)].translate(vector=
     (Plate_member_coord_x_1[i][j][0], Plate_member_coord_y[n][1],(Width_of_section/2))),


# # # for m in range(2):
  # # # for n in range(Number_of_Layer):
    # # # for i in range(Num_Pla*(Number_of_partition_in_CLT_V+1)):
      #       #       #       mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(m+1),
      #                                                 #                                                 #
part=mdb.models['Model-1'].parts['Part-Connec-CLT']),mdb.models['Model-1'].rootAssembly.ins
tances['Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(m+1)].translate(vector=
     # # # (CLT_Pin_Coord_x[i], CLT_Pin_Coord_y[n],CLT_Pin_Coord_z[m])),

for m in range(2):
  for n in range(Number_of_Layer+1):
    for i in range(Number_of_Column_in_A_plane+1):
      for j in range(Number_of_Timber[i]*(Number_of_partition_in_CLT_V+1)-1):
        mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-ASS-Con-CLT-'+str(j+1)+'-'+str(i+1)+'-'+str(n+1)+'-'+str(m+1),

part=mdb.models['Model-1'].parts['Part-Connec-CLT']),mdb.models['Model-1'].rootAssembly.ins
tances['Part-ASS-Con-CLT-'+str(j+1)+'-'+str(i+1)+'-'+str(n+1)+'-'+str(m+1)].translate(vector=
      (CLT_Pin_Coord_x_1[i][j], CLT_Pin_Coord_y[n],CLT_Pin_Coord_z[m])),


for n in range(Number_of_Layer+1):
   for i in range(Number_of_Column_in_B_plane+2):
      mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(0+1),

part=mdb.models['Model-1'].parts['Part-Connec-CLT']),mdb.models['Model-1'].rootAssembly.ins
tances['Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(0+1)].translate(vector=
      (Length_of_Truss, CLT_Pin_Coord_y[n],-Lz_co_B[i])),

for n in range(Number_of_Layer+1):
   for i in range(Number_of_Column_in_D_plane+2):
```

```
    mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
name='Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(1+1),

part=mdb.models['Model-1'].parts['Part-Connec-CLT']),mdb.models['Model-1'].rootAssembly.ins
tances['Part-ASS-Con-CLT-'+str(i+1)+'-'+str(n+1)+'-'+str(1+1)].translate(vector=
    (0, CLT_Pin_Coord_y[n],(-Width_of_Truss)-(-Lz_co_D[i]))),


#
# Create local coordinate systems for the Cross and Vertical beam
#
#
(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_i
n_B_plane+Number_of_Column_in_D_plane+4)

Crossbeam_A=[0]*((Number_of_Layer+1)*(Number_of_Column_in_A_plane+1))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):

Crossbeam_A[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysType=
    CARTESIAN,
name='Coordsys-Crossbeam_'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i+1),
origin=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]), point1=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][1],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][1],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][1]), point2=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]+1,
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]))


Crossbeam_B=[0]*((Number_of_Layer+1)*(Number_of_Column_in_B_plane+1))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_B_plane+1):

Crossbeam_B[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysType=
    CARTESIAN,
name='Coordsys-Crossbeam_'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of
_Column_in_A_plane+1+i+1), origin=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
```

```
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][0]), point1=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][1],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][1],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][1]), point2=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][0]+1,
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+i][0]))


Crossbeam_C=[0]*((Number_of_Layer+1)*(Number_of_Column_in_C_plane+1))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_C_plane+1):

Crossbeam_C[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysType
=
    CARTESIAN,
name='Coordsys-Crossbeam_'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of
_Column_in_A_plane+Number_of_Column_in_B_plane+2+i+1), origin=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+Number_of_Column_in_B_plane+2+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0]), point1=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+Number_of_Column_in_B_plane+2+i][1],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][1],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][1]), point2=(
    Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+Number_of_Column_in_B_plane+2+i][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
```

```
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0]+1,
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_B_plane+2+i][0]))

Crossbeam_D=[0]*((Number_of_Layer+1)*(Number_of_Column_in_D_plane+1))
for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_D_plane+1):

Crossbeam_D[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysType
=
      CARTESIAN,
name='Coordsys-Crossbeam_'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of
_Column_in_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane
+1)+i+1), origin=(
      Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i
][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][0],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][0]),
point1=(
      Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i
][1],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][1],
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][1]),
point2=(
      Truss_member_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colu
mn_in_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i
][0],
Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][0]
+100,
Truss_member_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+1+Number_of_Column_in_B_plane+1+(Number_of_Column_in_C_plane+1)+i][0])
)


Verticalbeam_choord_T=[0]*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plan
e+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)
for i in
range(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_i
```

```
n_B_plane+Number_of_Column_in_D_plane+4):

Verticalbeam_choord_T[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coor
dSysType=
    CARTESIAN,
name='Coordsys-Crossbeam_'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Nu
mber_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_pla
ne+4)+i), origin=(
    Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+i][0],
Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0],
Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0]), point1=(
    Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+i][1],
Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[1],
Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[1]), point2=(
    Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+i][0]-1,
Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0],
Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i]
[0]))

Crossbeam_Dia_A_1=[0]*(Number_of_Dia_A_L)
for i in range(Number_of_Dia_A_L):

Crossbeam_Dia_A_1[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_A_1_'+str(i+1), origin=(
    Dia_A_L_x[i][0], Dia_A_L_y[i][0], Dia_A_L_z[i][0]), point1=(
    Dia_A_L_x[i][1], Dia_A_L_y[i][1], Dia_A_L_z[i][1]), point2=(
    Dia_A_L_x[i][0], Dia_A_L_y[i][0]+1, Dia_A_L_z[i][0]))

Crossbeam_Dia_A_2=[0]*(Number_of_Dia_A_R)
for i in range(Number_of_Dia_A_R):

Crossbeam_Dia_A_2[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_A_2_'+str(i+1), origin=(
    Dia_A_R_x[i][0], Dia_A_R_y[i][0], Dia_A_R_z[i][0]), point1=(
    Dia_A_R_x[i][1], Dia_A_R_y[i][1], Dia_A_R_z[i][1]), point2=(
    Dia_A_R_x[i][0], Dia_A_R_y[i][0]+1, Dia_A_R_z[i][0]))

Crossbeam_Dia_B_1=[0]*(Number_of_Dia_B_L)
```

```
for i in range(Number_of_Dia_B_L):

Crossbeam_Dia_B_1[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_B_1_'+str(i+1), origin=(
    Dia_B_L_x[i][0], Dia_B_L_y[i][0], Dia_B_L_z[i][0]), point1=(
    Dia_B_L_x[i][1], Dia_B_L_y[i][1], Dia_B_L_z[i][1]), point2=(
    Dia_B_L_x[i][0], Dia_B_L_y[i][0]+1, Dia_B_L_z[i][0]))

Crossbeam_Dia_B_2=[0]*(Number_of_Dia_B_R)
for i in range(Number_of_Dia_B_R):

Crossbeam_Dia_B_2[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_B_2_'+str(i+1), origin=(
    Dia_B_R_x[i][0], Dia_B_R_y[i][0], Dia_B_R_z[i][0]), point1=(
    Dia_B_R_x[i][1], Dia_B_R_y[i][1], Dia_B_R_z[i][1]), point2=(
    Dia_B_R_x[i][0], Dia_B_R_y[i][0]+1, Dia_B_R_z[i][0]))

Crossbeam_Dia_C_1=[0]*(Number_of_Dia_C_L)
for i in range(Number_of_Dia_C_L):

Crossbeam_Dia_C_1[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_C_1_'+str(i+1), origin=(
    Dia_C_L_x[i][0], Dia_C_L_y[i][0], Dia_C_L_z[i][0]), point1=(
    Dia_C_L_x[i][1], Dia_C_L_y[i][1], Dia_C_L_z[i][1]), point2=(
    Dia_C_L_x[i][0], Dia_C_L_y[i][0]+1, Dia_C_L_z[i][0]))

Crossbeam_Dia_C_2=[0]*(Number_of_Dia_C_R)
for i in range(Number_of_Dia_C_R):

Crossbeam_Dia_C_2[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_C_2_'+str(i+1), origin=(
    Dia_C_R_x[i][0], Dia_C_R_y[i][0], Dia_C_R_z[i][0]), point1=(
    Dia_C_R_x[i][1], Dia_C_R_y[i][1], Dia_C_R_z[i][1]), point2=(
    Dia_C_R_x[i][0], Dia_C_R_y[i][0]+1, Dia_C_R_z[i][0]))

Crossbeam_Dia_D_1=[0]*(Number_of_Dia_D_L)
for i in range(Number_of_Dia_D_L):

Crossbeam_Dia_D_1[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_D_1_'+str(i+1), origin=(
    Dia_D_L_x[i][0], Dia_D_L_y[i][0], Dia_D_L_z[i][0]), point1=(
    Dia_D_L_x[i][1], Dia_D_L_y[i][1], Dia_D_L_z[i][1]), point2=(
    Dia_D_L_x[i][0], Dia_D_L_y[i][0]+1, Dia_D_L_z[i][0]))

Crossbeam_Dia_D_2=[0]*(Number_of_Dia_D_R)
for i in range(Number_of_Dia_D_R):

Crossbeam_Dia_D_2[i]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordS
ysType=
    CARTESIAN, name='Coordsys-Crossbeam_Dia_D_2_'+str(i+1), origin=(
    Dia_D_R_x[i][0], Dia_D_R_y[i][0], Dia_D_R_z[i][0]), point1=(
    Dia_D_R_x[i][1], Dia_D_R_y[i][1], Dia_D_R_z[i][1]), point2=(
```

```
    Dia_D_R_x[i][0], Dia_D_R_y[i][0]+1, Dia_D_R_z[i][0]))


#
#Create local coordinate systems for CLT plate
#

Plate_CLT_A_=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    Plate_CLT_A_[i]=[0]*(100)

for m in range(Number_of_Layer+1):
  for i in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[i]):

Plate_CLT_A_[i][j]=mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(coordSysTy
pe=
    CARTESIAN, name='Coordsys-Plate_CLT_A_'+str(j+1)+'-'+str(i+1)+'-'+str(m+1),
origin=(
    Plate_member_coord_x_1[i][j][0],Plate_member_coord_y[m][1],0), point1=(
    Plate_member_coord_x_1[i][j][1],Plate_member_coord_y[m][1],0), point2=(
    Plate_member_coord_x_1[i][j][0],Plate_member_coord_y[m][1],-Wid_Pla))

Plate_CLT_A_1=[0]*(Number_of_Column_in_A_plane+1)
for i in range(Number_of_Column_in_A_plane+1):
    Plate_CLT_A_1[i]=filter(lambda x: x != 0, Plate_CLT_A_[i])


#
# Create spring coupling between the parts
#

# Spring_stiff_truss_ToBoTo_3tr3mo = [4000000000.0, 4000000000.0, 4000000000.0,
80000.0,80000.0,80000]
Spring_stiff_truss_ToDiBo_3tr3mo = [4000000000.0, 4000000000.0, 4000000000.0,
65000,65000,65000]



#A_
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-A-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n+1)+'_'+'4-A
-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Col
umn_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+1)+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_A[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-A-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
```

```
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_Ref_coord_
y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column
_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_Ref_coord_z[(m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
er_of_Column_in_D_plane+4)+n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_Ref_coord_
y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column
_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_Ref_coord_z[(m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
er_of_Column_in_D_plane+4)+n][0]),))))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
+1)+'_'+'4-A-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+1)+'_dof'+str(i+1),
orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_A[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_mem
ber_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0]),))),Regi
on(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_mem
ber_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0]),))))),  ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
```

```
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-A-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n+1)+'_'+'4-A
-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Col
umn_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+2)+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_A[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-A-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][1],Truss_Ref_coord_
y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column
_in_B_plane+Number_of_Column_in_D_plane+4)+n][1],Truss_Ref_coord_z[(m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
er_of_Column_in_D_plane+4)+n][1]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][1],Truss_Ref_coord_
y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column
_in_B_plane+Number_of_Column_in_D_plane+4)+n][1],Truss_Ref_coord_z[(m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
er_of_Column_in_D_plane+4)+n][1]),))))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
+2)+'_'+'4-A-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Nu
mber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+2)+'_dof'+str(i+1),
orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_A[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+n+2)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
+1][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_m
ember_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n+1][0]),)))
,Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-A-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n
+1][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n][0],Truss_m
ember_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_i
n_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n+1][0]),)))
), ), springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_A_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-4-Dia-A-L'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-A-L-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-A-L-ASS-'+str(n+1)].vertices.
findAt(

((Dia_A_conect_L_x[n][m],Dia_A_conect_L_y[n][m],Dia_A_conect_L_z[n][m]),)),),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-L-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
        ((Dia_A_conect_L_x[n][m],Dia_A_conect_L_y[n][m],Dia_A_conect_L_z[n][m]),)),),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for n in range(Number_of_Dia_A_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_A_L[n][0]+1)+'-'+'Part-4-Dia-A-L-'+str(n+1)+'-
1'+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_A_L[
n][0]+1)].vertices.findAt(
        ((Dia_A_L_x[n][0],Dia_A_L_y[n][0],Dia_A_L_z[n][0]),)),),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-L-'+str(n+1)+'-1'].vertice
s.findAt(
        ((Dia_A_L_x[n][0],Dia_A_L_y[n][0],Dia_A_L_z[n][0]),)),),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_A_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
```

```
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_A_L[n][1]+1)+'-'+'Part-4-Dia-A-L-'+str(n+1)+'-
2'+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_A_L[
n][1]+1)].vertices.findAt(
        ((Dia_A_L_x[n][1],Dia_A_L_y[n][1],Dia_A_L_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-L-'+str(n+1)+'-2'].vertice
s.findAt(
        ((Dia_A_L_x[n][1],Dia_A_L_y[n][1],Dia_A_L_z[n][1]),)))),        ),        springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])




for m in range(2):
  for n in range(Number_of_Dia_A_R):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-4-Dia-A-R'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-A-R-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-A-R-ASS-'+str(n+1)].vertices.
findAt(

((Dia_A_conect_R_x[n][m],Dia_A_conect_R_y[n][m],Dia_A_conect_R_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-R-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
        ((Dia_A_conect_R_x[n][m],Dia_A_conect_R_y[n][m],Dia_A_conect_R_z[n][m]),)))),        ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])



for n in range(Number_of_Dia_A_R):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_A_R[n][0]+1)+'-'+'Part-4-Dia-A-R-'+str(n+1)+'-
1'+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_A_R[
n][0]+1)].vertices.findAt(
        ((Dia_A_R_x[n][0],Dia_A_R_y[n][0],Dia_A_R_z[n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-R-'+str(n+1)+'-1'].vertic
```

```
es.findAt(
    ((Dia_A_R_x[n][0],Dia_A_R_y[n][0],Dia_A_R_z[n][0]),))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_A_R):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-3'+str(Dig_connect_colum_A_R[n][1]+1)+'-'+'Part-4-Dia-A-R-'+str(n+1)+'-
2'+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_A_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_A_R[
n][1]+1)].vertices.findAt(
    ((Dia_A_R_x[n][1],Dia_A_R_y[n][1],Dia_A_R_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-A-R-'+str(n+1)+'-2'].vertic
es.findAt(
    ((Dia_A_R_x[n][1],Dia_A_R_y[n][1],Dia_A_R_z[n][1]),))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

# B_
for m in range(Number_of_Layer+1):
 for n in range(Number_of_Column_in_B_plane+1):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-B-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+n+1)+'_'+'4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*
n+1)+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_B[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-B-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+n][0],Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+1+n][0],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+N
umber_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_pl
ane+4)+Number_of_Column_in_A_plane+1+n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-B-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+n][0],Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
```

```
_of_Column_in_A_plane+1+n][0],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+N
umber_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_pl
ane+4)+Number_of_Column_in_A_plane+1+n][0]),))))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_B_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n+1)+'_'+'4-B-'+str((m)*(Number_of_Column_in_A_plane
+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D
_plane+4)+2*n+1)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_B[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1)].vertices.fi
ndAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n][0],Truss_member_coord_y[(m)*(Number_of_Column_in
_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Colu
mn_in_D_plane+4)+Number_of_Column_in_A_plane+1+n][0],Truss_member_coord_z[(Numbe
r_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of
_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-B-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n][0],Truss_member_coord_y[(m)*(Number_of_Column_in
_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Colu
mn_in_D_plane+4)+Number_of_Column_in_A_plane+1+n][0],Truss_member_coord_z[(Numbe
r_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of
_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+n][0]),)))), ), springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_B_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-B-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+n+1)+'_'+'4-B-'+str((m)*(Number_of_Column_in_A_plane+Number_of_
Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*
n+2)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_B[n].id], regionPairs=((Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-B-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+n][1],Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+1+n][1],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+N
umber_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_pl
ane+4)+Number_of_Column_in_A_plane+1+n][1]),)),),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-B-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+n][1],Truss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+1+n][1],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+N
umber_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_pl
ane+4)+Number_of_Column_in_A_plane+1+n][1]),)))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_B_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n+2)+'_'+'4-B-'+str((m)*(Number_of_Column_in_A_plane
+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D
_plane+4)+2*n+2)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_B[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+2)].vertices.fi
ndAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n+1][0],Truss_member_coord_y[(m)*(Number_of_Column
_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_C
olumn_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1][0],Truss_member_coord_z[(
Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Num
ber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pl
ane+1+n+1][0]),)),),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-B-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(
```

```
((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n+1][0],Truss_member_coord_y[(m)*(Number_of_Column
_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_C
olumn_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1][0],Truss_member_coord_z[(
Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Num
ber_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pl
ane+1+n+1][0]),))))                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_B_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-4-Dia-B-L'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-B-L-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-B-L-ASS-'+str(n+1)].vertices.
findAt(

((Dia_B_conect_L_x[n][m],Dia_B_conect_L_y[n][m],Dia_B_conect_L_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-L-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
        ((Dia_B_conect_L_x[n][m],Dia_B_conect_L_y[n][m],Dia_B_conect_L_z[n][m]),))),          ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for n in range(Number_of_Dia_B_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_B_L[n][0]+1)+'-'+'Part-4-Dia-B-L-'+str(n+1)+'-
1'+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_B_L[
n][0]+1)].vertices.findAt(
        ((Dia_B_L_x[n][0],Dia_B_L_y[n][0],Dia_B_L_z[n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-L-'+str(n+1)+'-1'].vertice
s.findAt(
        ((Dia_B_L_x[n][0],Dia_B_L_y[n][0],Dia_B_L_z[n][0]),))),          ),          springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_B_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_B_L[n][1]+1)+'-'+'Part-4-Dia-B-L-'+str(n+1)+'-
```

```
2'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_B_L[
n][1]+1)].vertices.findAt(
      ((Dia_B_L_x[n][1],Dia_B_L_y[n][1],Dia_B_L_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-L-'+str(n+1)+'-2'].vertice
s.findAt(
      ((Dia_B_L_x[n][1],Dia_B_L_y[n][1],Dia_B_L_z[n][1]),)),)),      ),      springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_B_R):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-4-Dia-B-R'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-B-R-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-B-R-ASS-'+str(n+1)].vertices.
findAt(

((Dia_B_conect_R_x[n][m],Dia_B_conect_R_y[n][m],Dia_B_conect_R_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-R-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
      ((Dia_B_conect_R_x[n][m],Dia_B_conect_R_y[n][m],Dia_B_conect_R_z[n][m]),)),)),      ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_B_R):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_B_R[n][0]+1)+'-'+'Part-4-Dia-B-R-'+str(n+1)+'-
1'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_B_R[
n][0]+1)].vertices.findAt(
      ((Dia_B_R_x[n][0],Dia_B_R_y[n][0],Dia_B_R_z[n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-R-'+str(n+1)+'-1'].vertic
es.findAt(
      ((Dia_B_R_x[n][0],Dia_B_R_y[n][0],Dia_B_R_z[n][0]),)),)),      ),      springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_B_R):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
```

```
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_B_R[n][1]+1)+'-'+'Part-4-Dia-B-R-'+str(n+1)+'-
2'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_B_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_B_R[
n][1]+1)].vertices.findAt(
      ((Dia_B_R_x[n][1],Dia_B_R_y[n][1],Dia_B_R_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-B-R-'+str(n+1)+'-2'].vertic
es.findAt(
      ((Dia_B_R_x[n][1],Dia_B_R_y[n][1],Dia_B_R_z[n][1]),)))),      ),      springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


# C_
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_C_plane+1):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-C-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+Number_of_Column_in_B_plane+2+n+1)+'_'+'4-C-'+str((m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
er_of_Column_in_D_plane+4)+2*n+1)+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_C[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-C-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane
+2+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+Number_of_Column_in_B_plane+2+n][0],Truss_Ref_coord_y[(m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2
+n][0],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colum
n_in_A_plane+Number_of_Column_in_B_plane+2+n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+Number_of_Column_in_B_plane+2+n][0],Truss_Ref_coord_y[(m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2
+n][0],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colum
n_in_A_plane+Number_of_Column_in_B_plane+2+n][0]),))),      ),      springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
```

```
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_C_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+3-n+1)+'_'+'4-C-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+1)+'_dof'+s
tr(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_C[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Colum
n_in_B_plane+Number_of_Column_in_C_plane+3-n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_plane+3-
n][0],Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C
_plane+3-n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_plane+3-
n][0],Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C
_plane+3-n][0]),))))),                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_C_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-C-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+Number_of_Column_in_B_plane+2+n+1)+'_'+'4-C-'+str((m)*(Number_of_
Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Numb
```

```
er_of_Column_in_D_plane+4)+2*n+2)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_C[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-C-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane
+2+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+Number_of_Column_in_B_plane+2+n][1],Truss_Ref_coord_y[(m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2
+n][1],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colum
n_in_A_plane+Number_of_Column_in_B_plane+2+n][1]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+Number_of_Column_in_B_plane+2+n][1],Truss_Ref_coord_y[(m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+2
+n][1],Truss_Ref_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Colum
n_in_A_plane+Number_of_Column_in_B_plane+2+n][1]),))))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_C_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+2-n+1)+'_'+'4-C-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_
plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+2)+'_dof'+s
tr(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_C[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Colum
n_in_B_plane+Number_of_Column_in_C_plane+2-n+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+2-n][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_plane+2-
```

```
n][0],Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C
_plane+2-n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-C-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_p
lane+2-n][0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number
_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C_plane+2-
n][0],Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number
_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4
)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane+Number_of_Column_in_C
_plane+2-n][0]),))))),                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_C_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-4-Dia-C-L'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-C-L-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-C-L-ASS-'+str(n+1)].vertices.
findAt(

((Dia_C_conect_L_x[n][m],Dia_C_conect_L_y[n][m],Dia_C_conect_L_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-L-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
        ((Dia_C_conect_L_x[n][m],Dia_C_conect_L_y[n][m],Dia_C_conect_L_z[n][m]),)))),        ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for n in range(Number_of_Dia_C_L):
  for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_C_L[n][0]+1)+'-'+'Part-4-Dia-C-L-'+str(n+1)+'-
1'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_C_L[
n][0]+1)].vertices.findAt(
      ((Dia_C_L_x[n][0],Dia_C_L_y[n][0],Dia_C_L_z[n][0]),)),Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-L-'+str(n+1)+'-1'].vertic
es.findAt(
    ((Dia_C_L_x[n][0],Dia_C_L_y[n][0],Dia_C_L_z[n][0]),))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_C_L):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-3'+str(Dig_connect_colum_C_L[n][1]+1)+'-'+'Part-4-Dia-C-L-'+str(n+1)+'-
2'+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_C_L[
n][1]+1)].vertices.findAt(
    ((Dia_C_L_x[n][1],Dia_C_L_y[n][1],Dia_C_L_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-L-'+str(n+1)+'-2'].vertic
es.findAt(
    ((Dia_C_L_x[n][1],Dia_C_L_y[n][1],Dia_C_L_z[n][1]),))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(2):
 for n in range(Number_of_Dia_C_R):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-4-Dia-C-R'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-C-R-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-C-R-ASS-'+str(n+1)].vertices.
findAt(

((Dia_C_conect_R_x[n][m],Dia_C_conect_R_y[n][m],Dia_C_conect_R_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-R-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
    ((Dia_C_conect_R_x[n][m],Dia_C_conect_R_y[n][m],Dia_C_conect_R_z[n][m]),))),    ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_C_R):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-3'+str(Dig_connect_colum_C_R[n][0]+1)+'-'+'Part-4-Dia-C-R-'+str(n+1)+'
-1'+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_C_R[
```

```
n][0]+1)].vertices.findAt(
      ((Dia_C_R_x[n][0],Dia_C_R_y[n][0],Dia_C_R_z[n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-R-'+str(n+1)+'-1'].vertic
es.findAt(
      ((Dia_C_R_x[n][0],Dia_C_R_y[n][0],Dia_C_R_z[n][0]),))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_C_R):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_C_R[n][1]+1)+'-'+'Part-4-Dia-C-R-'+str(n+1)+'
-2'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_C_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_C_R[
n][1]+1)].vertices.findAt(
      ((Dia_C_R_x[n][1],Dia_C_R_y[n][1],Dia_C_R_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-C-R-'+str(n+1)+'-2'].vertic
es.findAt(
      ((Dia_C_R_x[n][1],Dia_C_R_y[n][1],Dia_C_R_z[n][1]),))),  ),  springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

# D_
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane+1):
    for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-D-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1
+n+1)+'_'+'4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+1)+'_dof'+str(i+1),
orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_D[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-D-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_C_pla
ne+1+Number_of_Column_in_B_plane+1+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0],Tru
ss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Numb
er_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pla
ne+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0],Truss_Ref
_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_
Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0]),)),Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0],Tru
ss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Numb
er_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pla
ne+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0],Truss_Ref
_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_
Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][0]),)))),  ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+4-n)+'_'+'4-D-'+str((m)*(Number_of_Column_in_A_pla
ne+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_
D_plane+4)+2*n+3)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_D[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4-n)].vertice
s.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Colu
mn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of
_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coor
d_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Col
umn_in_D_plane+3-n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+3)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Colu
mn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of
_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
```

```
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coor
d_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Col
umn_in_D_plane+3-n][0]),)))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
for m in range(Number_of_Layer+1):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+1
)+'_'+'4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Numb
er_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+1)+'_dof'+str(i+1),
orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_D[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][
0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][0],Truss_member_
coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][0]),)),),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+1)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][
0],Truss_member_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][0],Truss_member_
coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_pl
ane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)][0]),)))), ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane+1):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-D-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1
+n+1)+'_'+'4-D-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+2*n+2)+'_dof'+str(i+1),
orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_D[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-D-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
```

```
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Column_in_C_pla
ne+1+Number_of_Column_in_B_plane+1+n+1)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1],Tru
ss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Numb
er_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pla
ne+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1],Truss_Ref
_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_
Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_Ref_coord_x[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_
A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1],Tru
ss_Ref_coord_y[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Numb
er_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_pla
ne+1+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1],Truss_Ref
_coord_z[(m)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_
Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+Number_of_Column_in_C_plane+1+Number_of_Column_in_B_plane+1+n][1]),))),  ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+4-n)+'_'+'4-D-'+str((m)*(Number_of_Column_in_A_pla
ne+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_
D_plane+4)+2*n+2)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_D[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Colum
n_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4-n)].vertice
s.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Colu
mn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of
_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coor
d_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
```

```
n_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Col
umn_in_D_plane+3-n][0]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-D-'+str((m)*(Number_of_Colum
n_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_
Column_in_D_plane+4)+2*n+2)].vertices.findAt(

((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coord_y[(m)*(Number_of_Colu
mn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of
_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+
Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-n][0],Truss_member_coor
d_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane
+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_i
n_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Col
umn_in_D_plane+3-n][0]),)))),                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_D_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-4-Dia-D-L'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-D-L-ASS-'+str(n+1)+'_dof'
+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-D-L-ASS-'+str(n+1)].vertices.
findAt(

((Dia_D_conect_L_x[n][m],Dia_D_conect_L_y[n][m],Dia_D_conect_L_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-L-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
        ((Dia_D_conect_L_x[n][m],Dia_D_conect_L_y[n][m],Dia_D_conect_L_z[n][m]),))),        ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
for n in range(Number_of_Dia_D_L):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3'+str(Dig_connect_colum_D_L[n][0]+1)+'-'+'Part-4-Dia-D-L-'+str(n+1)+'
-1'+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_D_L[
n][0]+1)].vertices.findAt(
        ((Dia_D_L_x[n][0],Dia_D_L_y[n][0],Dia_D_L_z[n][0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-L-'+str(n+1)+'-1'].vertic
es.findAt(
        ((Dia_D_L_x[n][0],Dia_D_L_y[n][0],Dia_D_L_z[n][0]),))),        ),        springBehavior=ON,
```

```
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_D_L):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_D_L[n][1]+1)+'-'+'Part-4-Dia-D-L-'+str(n+1)+'
-2'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_1[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_D_L[
n][1]+1)].vertices.findAt(
      ((Dia_D_L_x[n][1],Dia_D_L_y[n][1],Dia_D_L_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-L-'+str(n+1)+'-2'].vertic
es.findAt(
      ((Dia_D_L_x[n][1],Dia_D_L_y[n][1],Dia_D_L_z[n][1]),)))),      ),      springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(2):
  for n in range(Number_of_Dia_D_R):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-4-Dia-D-R'+str(n+1)+'-'+str(m+1)+'-'+'Part-Dia-D-R-ASS-'+str(n+1)+'_dof
'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-Dia-D-R-ASS-'+str(n+1)].vertices.
findAt(

((Dia_D_conect_R_x[n][m],Dia_D_conect_R_y[n][m],Dia_D_conect_R_z[n][m]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-R-'+str(n+1)+'-'+str(m+
1)].vertices.findAt(
      ((Dia_D_conect_R_x[n][m],Dia_D_conect_R_y[n][m],Dia_D_conect_R_z[n][m]),))),      ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for n in range(Number_of_Dia_D_R):
   for i in range(6):
      mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
      axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
      ,
name='Springs-Pa-3'+str(Dig_connect_colum_D_R[n][0]+1)+'-'+'Part-4-Dia-D-R-'+str(n+1)+'
-1'+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_D_R[
n][0]+1)].vertices.findAt(
      ((Dia_D_R_x[n][0],Dia_D_R_y[n][0],Dia_D_R_z[n][0]),)),Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-R-'+str(n+1)+'-1'].vertic
es.findAt(
    ((Dia_D_R_x[n][0],Dia_D_R_y[n][0],Dia_D_R_z[n][0]),)))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for n in range(Number_of_Dia_D_R):
  for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-3'+str(Dig_connect_colum_D_R[n][1]+1)+'-'+'Part-4-Dia-D-R-'+str(n+1)+'
-2'+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_Dia_D_2[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str(Dig_connect_colum_D_R[
n][1]+1)].vertices.findAt(
    ((Dia_D_R_x[n][1],Dia_D_R_y[n][1],Dia_D_R_z[n][1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-4-Dia-D-R-'+str(n+1)+'-2'].vertic
es.findAt(
    ((Dia_D_R_x[n][1],Dia_D_R_y[n][1],Dia_D_R_z[n][1]),)))),    ),    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])




for m in range(Number_of_Layer+1):
 for n in range(Number_of_Column_in_A_plane+1):
  for j in range(Number_of_Timber[n]*(Number_of_partition_in_CLT_V+1)-1):
   for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
    ,
name='Springs-Pa-A-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+n+1)+'_-'+'Pa
rt-Con-CLT-'+str(j+1)+'-'+str(n+1)+'-'+str(m+1)+'-'+str(0+1)+'_dof'+str(i+1), orientation=
    mdb.models['Model-1'].rootAssembly.datums[Crossbeam_A[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-A-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+n+1)].vertices.findAt(
    ((CLT_Pin_Coord_x_1[n][j],CLT_Pin_Coord_y[m],CLT_Pin_Coord_z[0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(j+1)+'-'+str(n
+1)+'-'+str(m+1)+'-'+str(0+1)].vertices.findAt(
    ((CLT_Pin_Coord_x_1[n][j],CLT_Pin_Coord_y[m],CLT_Pin_Coord_z[0]),)))),    ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(Number_of_Layer+1):
 for n in range(Number_of_Column_in_C_plane+1):
  for j in range(Number_of_Timber[n]*(Number_of_partition_in_CLT_V+1)-1):
   for i in range(6):
    mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
    axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
```

```
      ,
name='Springs-Pa-C-ASS-'+str((m)*(Number_of_Column_in_A_plane+Number_of_Column_in_
C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_C
olumn_in_A_plane+Number_of_Column_in_B_plane+2+n+1)+'_'+'Part-Con-CLT-'+str(j+1)+'-'
+str(n+1)+'-'+str(m+1)+'-'+str(1+1)+'_dof'+str(i+1), orientation=
      mdb.models['Model-1'].rootAssembly.datums[Crossbeam_C[n].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-C-ASS-'+str((m)*(Number_of_Col
umn_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_
of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column_in_B_plane
+2+n+1)].vertices.findAt(
      ((CLT_Pin_Coord_x_1[n][j],CLT_Pin_Coord_y[m],CLT_Pin_Coord_z[1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(j+1)+'-'+str(n
+1)+'-'+str(m+1)+'-'+str(1+1)].vertices.findAt(
      ((CLT_Pin_Coord_x_1[n][j],CLT_Pin_Coord_y[m],CLT_Pin_Coord_z[1]),))),          ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[n]*(Number_of_partition_in_CLT_V+1)-1):
      for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Plat-CLT-A-'+str(int(j/(Number_of_partition_in_CLT_V+1))+1)+'-'+str(n+1)+'-'
+str(m+1)+'_'+'Part-Con-CLT-'+str(j+1)+'-'+str(n+1)+'-'+str(m+1)+'-'+str(0+1)+'_dof'+str(i
+1), orientation=

mdb.models['Model-1'].rootAssembly.datums[Plate_CLT_A_1[n][int(j/(Number_of_partition_in_
CLT_V+1))].id], regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(int(j/(Number_of_
partition_in_CLT_V+1))+1)+'-'+str(n+1)+'-'+str(m+1)].vertices.findAt(

((CLT_Pin_Coord_x_1[n][j],Plate_member_coord_y[m][1],CLT_Pin_Coord_z[0]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(j+1)+'-'+str(n
+1)+'-'+str(m+1)+'-'+str(0+1)].vertices.findAt(
      ((CLT_Pin_Coord_x_1[n][j],Plate_member_coord_y[m][1],CLT_Pin_Coord_z[0]),))),          ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
#
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[n]*(Number_of_partition_in_CLT_V+1)-1):
      for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Plat-CLT-A-'+str(int(j/(Number_of_partition_in_CLT_V+1))+1)+'-'+str(n+1)+'-'
+str(m+1)+'_'+'Part-Con-CLT-'+str(j+1)+'-'+str(n+1)+'-'+str(m+1)+'-'+str(1+1)+'_dof'+str(i
+1), orientation=

mdb.models['Model-1'].rootAssembly.datums[Plate_CLT_A_1[n][int(j/(Number_of_partition_in_
CLT_V+1))].id], regionPairs=((Region(
```

```
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(int(j/(Number_of_
partition_in_CLT_V+1))+1)+'-'+str(n+1)+'-'+str(m+1)].vertices.findAt(

((CLT_Pin_Coord_x_1[n][j],Plate_member_coord_y[m][1],CLT_Pin_Coord_z[1]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(j+1)+'-'+str(n
+1)+'-'+str(m+1)+'-'+str(1+1)].vertices.findAt(
      ((CLT_Pin_Coord_x_1[n][j],Plate_member_coord_y[m][1],CLT_Pin_Coord_z[1]),)))),      ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])




#
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[n]-1):
      for p in range(int(Wid_Pla)+1):
        for i in range(6):
          mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
          axis=FIXED_DOF,      dashpotBehavior=OFF,      dashpotCoefficient=0.0,      dof1=(i+1),
dof2=(i+1)
          ,
name='Springs-Part-ASS-Plate-'+str(p+1)+'-'+str(j+1)+'-'+str(n+1)+'-'+str(m+1)+'_'+'Part-Pl
ate-'+str(p+1)+'-'+str(j+1)+'-'+str(n+1)+'-'+str(m+1)+'_dof'+str(i+1), orientation=
          mdb.models['Model-1'].rootAssembly.datums[Plate_CLT_A_1[n][j].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(j+1)+'-'+str(n+1)
+'-'+str(m+1)].vertices.findAt(

((Plate_member_coord_x_1[n][j][1],Plate_member_coord_y[m][1],Plate_connection_coord_z[p
]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(j+1+1)+'-'+str(n
+1)+'-'+str(m+1)].vertices.findAt(

((Plate_member_coord_x_1[n][j][1],Plate_member_coord_y[m][1],Plate_connection_coord_z[p
]),))))), ), springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
#
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane):
      for p in range(int(Wid_Pla)+1):
        for i in range(6):
          mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
          axis=FIXED_DOF,      dashpotBehavior=OFF,      dashpotCoefficient=0.0,      dof1=(i+1),
dof2=(i+1)
          ,
name='Springs-Part-ASS-Plate-'+str(p+1)+'-'+str(Total_Number_of_Timber[n+1]+1)+'-'+str(n
+1)+'-'+str(m+1)+'_'+'Part-Plate-'+str(p+1)+'-'+str(Total_Number_of_Timber[n+1]+1+1)+'-'
+str(n+1)+'-'+str(m+1)+'_dof'+str(i+1), orientation=

mdb.models['Model-1'].rootAssembly.datums[Plate_CLT_A_1[n][Number_of_Timber[n]-1].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(Number_of_Timbe
```

```
r[n])+'-'+str(n+1)+'-'+str(m+1)].vertices.findAt(

((Plate_member_coord_x_1[n][Number_of_Timber[n]-1][1],Plate_member_coord_y[m][1],Plate
_connection_coord_z[p]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(0+1)+'-'+str(n+1
+1)+'-'+str(m+1)].vertices.findAt(

((Plate_member_coord_x_1[n][Number_of_Timber[n]-1][1],Plate_member_coord_y[m][1],Plate
_connection_coord_z[p]),))))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
#




for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_B_plane+2):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+n+1)+'_'+'Part-Con-CLT-'+str(n+1)+'-'+str(m+1)+'-'+str(
0+1)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[n].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+n+1)].vertices.fi
ndAt(
        ((Length_of_Truss,CLT_Pin_Coord_y[m],-Lz_co_B[n]),))),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(n+1)+'-'+str(
m+1)+'-'+str(0+1)].vertices.findAt(
        ((Length_of_Truss,CLT_Pin_Coord_y[m],-Lz_co_B[n]),))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_B_plane+2):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Part-ASS-Plate-'+str(Number_of_Timber[Number_of_Column_in_A_plane])+'-'+
str(Number_of_Column_in_A_plane+1)+'-'+str(m+1)+'_'+'Part-Con-CLT-'+str(n+1)+'-'+str(m
+1)+'-'+str(0+1)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[m].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(Number_of_Timbe
r[Number_of_Column_in_A_plane])+'-'+str(Number_of_Column_in_A_plane+1)+'-'+str(m+1)].
vertices.findAt(
```

```
        ((Length_of_Truss,Plate_member_coord_y[m][1],-Lz_co_B[n]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(n+1)+'-'+str(
m+1)+'-'+str(0+1)].vertices.findAt(
        ((Length_of_Truss,Plate_member_coord_y[m][1],-Lz_co_B[n]),)))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

#
for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+
Number_of_Column_in_A_plane+1+Number_of_Column_in_C_plane+1+Number_of_Column_i
n_B_plane+1+n+1)+'_'+'Part-Con-CLT-'+str(n+1)+'-'+str(m+1)+'-'+str(1+1)+'_dof'+str(i+1),
orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[m].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+Number_of_Col
umn_in_C_plane+1+Number_of_Column_in_B_plane+1+n+1)].vertices.findAt(
        ((0,CLT_Pin_Coord_y[m],-Width_of_Truss+Lz_co_D[n]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(n+1)+'-'+str(
m+1)+'-'+str(1+1)].vertices.findAt(
        ((0,CLT_Pin_Coord_y[m],-Width_of_Truss+Lz_co_D[n]),)))), ), springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])

for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_D_plane+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Part-ASS-Plate-'+str(0+1)+'-'+str(0+1)+'-'+str(m+1)+'_'+'Part-Con-CLT-'+str(
n+1)+'-'+str(m+1)+'-'+str(1+1)+'_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[m].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(0+1)+'-'+str(0+1
)+'-'+str(m+1)].vertices.findAt(
        ((0,Plate_member_coord_y[m][1],-Width_of_Truss+Lz_co_D[n]),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(n+1)+'-'+str(
m+1)+'-'+str(1+1)].vertices.findAt(
        ((0,Plate_member_coord_y[m][1],-Width_of_Truss+Lz_co_D[n]),)))),                    ),
springBehavior=ON, springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
#
for m in range(Number_of_Layer+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
```

```
        ,
name='Springs-Pa-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of
_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+1
)+'_'+'Part-Con-CLT-'+str(Number_of_Column_in_D_plane+1+1)+'-'+str(m+1)+'-'+str(1+1)+'
_dof'+str(i+1), orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[m].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(N
umber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pl
ane+Number_of_Column_in_D_plane+4)+1)].vertices.findAt(
        ((0,CLT_Pin_Coord_y[m],0),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(Number_of_C
olumn_in_D_plane+1+1)+'-'+str(m+1)+'-'+str(1+1)].vertices.findAt(
        ((0,CLT_Pin_Coord_y[m],0),)))),                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])


for m in range(Number_of_Layer+1):
    for i in range(6):
        mdb.models['Model-1'].rootAssembly.engineeringFeatures.TwoPointSpringDashpot(
        axis=FIXED_DOF, dashpotBehavior=OFF, dashpotCoefficient=0.0, dof1=(i+1), dof2=(i+1)
        ,
name='Springs-Part-ASS-Plate-'+str(0+1)+'-'+str(0+1)+'-'+str(m+1)+'_'+'Part-Con-CLT-'+str(
Number_of_Column_in_D_plane+1+1)+'-'+str(m+1)+'-'+str(1+1)+'_dof'+str(i+1),
orientation=
        mdb.models['Model-1'].rootAssembly.datums[Verticalbeam_choord_T[m].id],
regionPairs=((Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(0+1)+'-'+str(0+1
)+'-'+str(m+1)].vertices.findAt(
        ((0,Plate_member_coord_y[m][1],0),)),Region(

vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-'+str(Number_of_C
olumn_in_D_plane+1+1)+'-'+str(m+1)+'-'+str(1+1)].vertices.findAt(
        ((0,Plate_member_coord_y[m][1],0),)))),                    ),                    springBehavior=ON,
springStiffness=Spring_stiff_truss_ToDiBo_3tr3mo[i])
#
```

```
# # 'Springs-Part-ASS-Plate-'+str(0+1)+'-'+str(0+1)+'-'+str(n+1)
```

```
#
# Create the mesh
#
for i in range(Number_of_Column_in_A_plane+1):
    mdb.models['Model-1'].parts['Part-A-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-A-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-A-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-A-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-A-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_A_L):
    mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-A1-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_A_R):
    mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Truss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), ))
    mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Truss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-A2-'+str(i+1)].generateMesh()
```

```
for i in range(Number_of_Column_in_B_plane+1):
    mdb.models['Model-1'].parts['Part-B-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-B-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-B-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-B-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-B-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_B_L):
    mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-B1-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_B_R):
    mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), ))
    mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-B2-'+str(i+1)].generateMesh()

for i in range(Number_of_Column_in_C_plane+1):
    mdb.models['Model-1'].parts['Part-C-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-C-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-C-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-C-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-C-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_C_L):
    mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(
```

```
mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-C1-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_C_R):
    mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), ))
    mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-C2-'+str(i+1)].generateMesh()

for i in range(Number_of_Column_in_D_plane+1):
    mdb.models['Model-1'].parts['Part-D-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-D-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-D-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-D-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of_Trus
s+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-D-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_D_L):
    mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
    mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].edges.getByBoundingBox(0,0,0,Length_of
_Truss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.1)
    mdb.models['Model-1'].parts['Part-Dia-D1-'+str(i+1)].generateMesh()

for i in range(Number_of_Dia_D_R):
    mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].setElementType(elemTypes=(ElemType(
        elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), ))
    mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].seedEdgeBySize(constraint=FINER,
        deviationFactor=0.1, edges=
```

```
mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].edges.getByBoundingBox(-Length_of_Tru
ss,-Height_of_Truss,0,Length_of_Truss+(Width_of_section/2.0)+0.1,0,0), size=0.1)
   mdb.models['Model-1'].parts['Part-Dia-D2-'+str(i+1)].generateMesh()

mdb.models['Model-1'].parts['Part-3'].setElementType(elemTypes=(ElemType(
      elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-3'].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Heig
ht_of_Truss+(Width_of_section/2.0)+0.1,0), ))
mdb.models['Model-1'].parts['Part-3'].seedEdgeBySize(constraint=FINER,
      deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-3'].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Heig
ht_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.5)
mdb.models['Model-1'].parts['Part-3'].generateMesh()


mdb.models['Model-1'].parts['Part-4'].setElementType(elemTypes=(ElemType(
      elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-4'].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Heig
ht_of_Truss+(Width_of_section/2.0)+0.1,0), ))
mdb.models['Model-1'].parts['Part-4'].seedEdgeBySize(constraint=FINER,
      deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-4'].edges.getByBoundingBox(0,0,0,Length_of_Truss+0.1,Heig
ht_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.05)
mdb.models['Model-1'].parts['Part-4'].generateMesh()


mdb.models['Model-1'].parts['Part-Connec-CLT'].setElementType(elemTypes=(ElemType(
      elemCode=B32, elemLibrary=STANDARD), ), regions=(

mdb.models['Model-1'].parts['Part-Connec-CLT'].edges.getByBoundingBox(0,0,0,Length_of_Tru
ss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
mdb.models['Model-1'].parts['Part-Connec-CLT'].seedEdgeBySize(constraint=FINER,
      deviationFactor=0.1, edges=

mdb.models['Model-1'].parts['Part-Connec-CLT'].edges.getByBoundingBox(0,0,0,Length_of_Tru
ss+0.1,Height_of_Truss+(Width_of_section/2.0)+0.1,0), size=0.02)
mdb.models['Model-1'].parts['Part-Connec-CLT'].generateMesh()


mdb.models['Model-1'].parts['Part-Plate-1'].setElementType(elemTypes=(ElemType(
      elemCode=S4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
      hourglassControl=DEFAULT), ElemType(elemCode=S3, elemLibrary=STANDARD)),
      regions=(

mdb.models['Model-1'].parts['Part-Plate-1'].faces.getByBoundingBox(0,0,0,Length_of_Truss+0.1
,Height_of_Truss+(Width_of_section/2.0)+0.1,0), ))
mdb.models['Model-1'].parts['Part-Plate-1'].seedPart(deviationFactor=0.1,
      minSizeFactor=0.1, size=0.5)
mdb.models['Model-1'].parts['Part-Plate-1'].generateMesh()
```

Assign Boundary conditions and Loads

```
# for i in range(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4):
    # mdb.models['Model-1'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
      # distributionType=UNIFORM, fieldName='', localCsys=None, name='BC-'+str(i+1),
      # region=Region(
      # vertices=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i+1)].vertices.findAt(
      # ((Truss_member_coord_x[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
      # Truss_member_coord_y[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0],
      # Truss_member_coord_z[(Number_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4)+i][0]),))), u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET, ur3=SET)



Load distribute
# L_Lz_D=[0]*(Number_of_Column_in_D_plane+3)
# for i in range(Number_of_Column_in_D_plane+1):
   # L_Lz_D[i+1]=Lz_D[i]
   # L_Lz_D[0]=0
   # L_Lz_D[Number_of_Column_in_D_plane+2]=0

# L_Lz_B=[0]*(Number_of_Column_in_B_plane+3)
# for i in range(Number_of_Column_in_B_plane+1):
   # L_Lz_B[i+1]=Lz_B[i]
   # L_Lz_B[0]=0
   # L_Lz_B[Number_of_Column_in_B_plane+2]=0

# Load_D_=[0]*(Number_of_Column_in_D_plane+2)
# for i in range(Number_of_Column_in_D_plane+2):
   # Load_D_[i]=[0]*(Number_of_Layer+1)

# Load_B_=[0]*(Number_of_Column_in_B_plane+2)
# for i in range(Number_of_Column_in_B_plane+2):
   # Load_B_[i]=[0]*(Number_of_Layer+1)

# for i in range(Number_of_Column_in_D_plane+2):
  # for n in range(Number_of_Layer+1):
   # Load_D_[i][n]=((L_Lz_D[i]+L_Lz_D[i+1])/2)*(Height_of_Truss/(Number_of_Layer+1))*Wind_pressure_Windward[int(((Height_of_Truss/(Number_of_Layer+1))/2+1)+n*(Height_of_Truss/(Number_of_Layer+1)))]

# for i in range(Number_of_Column_in_B_plane+2):
  # for n in range(Number_of_Layer+1):
   #
```

```
Load_B_[i][n]=((L_Lz_B[i]+L_Lz_B[i+1])/2)*(Height_of_Truss/(Number_of_Layer+1))*Wind_
pressure_Leeward[int(((Height_of_Truss/(Number_of_Layer+1))/2+1)+n*(Height_of_Truss/(Nu
mber_of_Layer+1)))]



Step

# mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
# mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
      # 'S','PE','PEEQ','PEMAG','LE','U','RF','CF','SF','CSTRESS','CDISP'))



#
#Assign Load
#
#Windward Load
for n in range(Number_of_Layer+1):
   mdb.models['Model-1'].LineLoad(comp1=Load_D_[0][n], createStepName='Step-1', name=
   'Load_Windward'+'1'+'-'+str(n+1), region=Region(

edges=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Nu
mber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pla
ne+Number_of_Column_in_D_plane+4)+1)].edges.getByBoundingBox(0,(Height_of_Truss/(Num
ber_of_Layer+1))*(n),0,0,(Height_of_Truss/(Number_of_Layer+1))*(n+1),0)))

for i in range(Number_of_Column_in_D_plane+1):
  for n in range(Number_of_Layer+1):
     mdb.models['Model-1'].LineLoad(comp1=Load_D_[i+1][n],          createStepName='Step-1',
name=
     'Load_Windward'+str(i+2)+'-'+str(n+1), region=Region(

edges=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Nu
mber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pla
ne+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Column
_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+4-i)].edges.ge
tByBoundingBox(

0,(Height_of_Truss/(Number_of_Layer+1))*(n),Truss_member_coord_z[(Number_of_Layer+1)*(
Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_p
lane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_Colu
mn_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-i][0],

0,(Height_of_Truss/(Number_of_Layer+1))*(n+1),Truss_member_coord_z[(Number_of_Layer+1
)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_
B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+Number_of_C
olumn_in_C_plane+Number_of_Column_in_B_plane+Number_of_Column_in_D_plane+3-i][0])
))
#Leeward Load

for n in range(Number_of_Layer+1):
   mdb.models['Model-1'].LineLoad(comp1=Load_B_[0][n], createStepName='Step-1', name=
   'Load_Leeward'+'1'+'-'+str(n+1), region=Region(

edges=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Nu
```

```
mber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pla
ne+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1+1)].edges.getByB
oundingBox(Length_of_Truss,(Height_of_Truss/(Number_of_Layer+1))*(n),0,Length_of_Truss,(
Height_of_Truss/(Number_of_Layer+1))*(n+1),0)))

for i in range(Number_of_Column_in_B_plane+1):
  for n in range(Number_of_Layer+1):
     mdb.models['Model-1'].LineLoad(comp1=Load_B_[i+1][n],          createStepName='Step-1',
name=
     'Load_Leeward'+str(i+2)+'-'+str(n+1), region=Region(

edges=mdb.models['Model-1'].rootAssembly.instances['Part-3-'+str((Number_of_Layer+1)*(Nu
mber_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_Column_in_B_pla
ne+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+2+i+1)].edges.getBy
BoundingBox(

Length_of_Truss,(Height_of_Truss/(Number_of_Layer+1))*(n),Truss_member_coord_z[(Number
_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_of_
Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+1
+i+1][0],

Length_of_Truss,(Height_of_Truss/(Number_of_Layer+1))*(n+1),Truss_member_coord_z[(Numb
er_of_Layer+1)*(Number_of_Column_in_A_plane+Number_of_Column_in_C_plane+Number_o
f_Column_in_B_plane+Number_of_Column_in_D_plane+4)+Number_of_Column_in_A_plane+
1+i+1][0])))


  #Self Load

Self_pressure=S_Weight/(Total_Number_of_Timber[Number_of_Column_in_A_plane+1]*(int(Wi
d_Pla)*Len_Pla))


for m in range(Number_of_Layer+1):
  for n in range(Number_of_Column_in_A_plane+1):
    for j in range(Number_of_Timber[n]):
       mdb.models['Model-1'].Pressure(amplitude=UNSET, createStepName='Step-1',
       distributionType=UNIFORM,          field='',          magnitude=Self_pressure,
name='Self-Weight'+str(j+1)+'-'+str(n+1)+'-'+str(m+1),
       region=Region(

side1Faces=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Plate-'+str(j+1)+'-'+str(n
+1)+'-'+str(m+1)].faces.getByBoundingBox(
       Plate_member_coord_x_1[n][j][0],Plate_member_coord_y[m][1],-int(Wid_Pla),
       Plate_member_coord_x_1[n][j][1],Plate_member_coord_y[m][1],0)))


# mdb.models['Model-1'].DisplacementBC(amplitude=UNSET, createStepName='Step-1',
       # distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
       # 'BC-19', region=Region(
       #
vertices=mdb.models['Model-1'].rootAssembly.instances['Part-ASS-Con-CLT-5-1-2'].vertices.getS
equenceFromMask(
       # mask=('[#1 ]', ), )+\
       #
mdb.models['Model-1'].rootAssembly.instances['Part-4-D-8'].vertices.getSequenceFromMask(
       # mask=('[#2 ]', ), )), u1=0.5, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET,
```

 # ur3=UNSET)