NUMERICAL STRUCTURAL ANALYSIS OF STEEL FRAMES USED IN BELT CONVEYORS

Jurković, Damir Tin

UNIVERSITY OF RIJEKA

FACULTY OF ENGINEERING

Graduate University Study of Mechanical Engineering

Master thesis

NUMERICAL STRUCTURAL ANALYSIS OF STEEL FRAMES USED IN BELT CONVEYORS

Rijeka, September 2024 Damir Tin Jurković

0069087709

UNIVERSITY OF RIJEKA

FACULTY OF ENGINEERING

Graduate University Study of Mechanical Engineering

Master thesis

NUMERICAL STRUCTURAL ANALYSIS OF STEEL FRAMES USED IN BELT CONVEYORS

Thesis supervisor: prof.dr.sc. Marino Brčić

Thesis co-supervisor: prof.dr.sc. Sanjin Braut

Rijeka, September 2024

Damir Tin Jurković

0069087709

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij strojarstva

Diplomski rad

NUMERIČKA STRUKTURALNA ANALIZA ČELIČNIH OKVIRA KORIŠTENIH U TRAKASTIM TRANSPORTERIMA

Mentor: prof.dr.sc. Marino Brčić

Komentor: prof.dr.sc. Sanjin Braut

Rijeka, rujan 2024

Damir Tin Jurković

0069087709

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 12.03.2024.

Zavod: Zavod za tehničku mehaniku

Predmet: Metoda konačnih elemenata čvrstih tijela

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Damir Tin Jurković (0069087709)**

Studij: Sveučilišni diplomski studij strojarstva (1100)

Modul: Računarska mehanika i inženjerstvo (1120)

Zadatak: **Numerička strukturna analiza čeličnih okvira korištenih u trakastim transporterima / Numerical structural analysis of steel frames used in belt conveyors**

Opis zadatka:

Koristeći dostupne podatke, izvršiti numeričku analizu naprezanja i deformacija čeličnih konstrukcija trakastih transportera, prema traženim standardima. Dati parametarsku definiciju karakteristika tipičnih čeličnih struktura u odnosu na primijenjeno opterećenje. Pomoću dobivenih rješenja razviti alat za povezivanje dobivenih parametara sa komercijalnim MKE softverskim rješenjima.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Predsjednik povjerenstva za diplomski ispit:
izv. prof. dr. sc. Igor Bonefačić

Mentor:
prof. dr. sc. Marino Brčić

Komentor:
prof. dr. sc. Sanjin Braut

STATEMENT

I hereby declare that I have independently completed this thesis.

_____

Rijeka, September 2024 Damir Tin Jurković

ACKNOWLEDGMENTS

Hvala prof.dr.sc. Marinu Brčiću na ukazanom povjerenju i pomoći tijekom izrade rada.

I would like to express my sincere gratitude to everyone at the Danieli Group for their support and contributions.

Hvala obitelji na svemu.

Contents

1. Introduction

This thesis addresses the numerical modeling and optimization of self-supporting steel structures used for belt conveyors in ore transportation. The central aim is to develop a *Python*-based tool that seamlessly integrates with *SAP2000* for finite element analysis. This tool automates the process of model definition, encompassing constraints, loads, and load combinations, through *Python*-driven commands.

The tool's user interface, designed to meet company requirements, operates via *Excel*. Users input all necessary model parameters and initialize the tool, which can be configured to either perform a structural analysis or proceed with further optimization.

The automation of model definition and analysis substantially reduces the time required by engineers, thereby enhancing efficiency, while the optimization process is geared towards identifying cost-effective solutions that adhere to all specified constraints and industry standards.

The integration of advanced software tools with engineering methodologies addresses complex challenges in mining operations, leading to more reliable and cost-effective outcomes. The standardization capabilities of this tool also contribute to improved safety, reduced material waste, and enhanced sustainability, marking a significant advancement toward the industry's economic and environmental objectives.

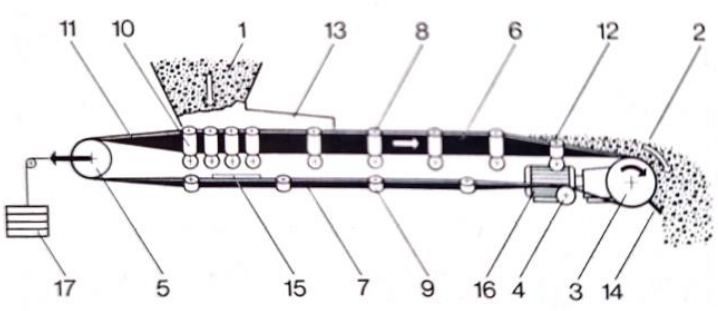
2. Overview of components and methodologies

This chapter provides a comprehensive overview of the essential components and methodologies involved in the numerical modeling and optimization of self-supporting steel structures. It begins with a detailed explanation of belt conveyors, followed by an introduction to the Finite Element Method (*FEM*) as a crucial tool for analyzing the structural integrity of the system. The chapter then explores the use of the genetic algorithm (*GA*) for optimizing the design. Finally, the chapter explores how parametrization is used to integrate all the previously discussed segments of the project.

2.1. Belt conveyors

Belt conveyors are continuous material handling systems characterized by belts reinforced with synthetic fabrics or steel cables, covered by rubber or synthetic materials.

These belts are supported by either straight or trough-shaped idlers [1]. Primarily used for onsite material transportation, belt conveyors ensure a continuous flow of material between operations without additional delays caused by loading or unloading. As the annual volume of material increases, the cost per ton of conveyor usage decreases, making them a cost-effective solution for material handling [2].



1. Feed
2. Discharge
3. Head pulley
4. Snub or deflecting pulley
5. Tail pulley
6. Top run
7. Return run
8. Top run idlers
9. Return run idlers
10. Feed rollers
11. Flat-to-trough transition
12. Trough-to-flat transition
13. Feed chute
14. Belt cleaner
15. Belt cleaner
16. Drive unit
17. Counterweight

Figure 2.1. Components of a belt conveyor

In a standard belt conveyor system, material is loaded onto the belt at the feeding point using a feed chute, ensuring even distribution and minimal spillage. The material is then conveyed along the top run of the belt, which is supported by corresponding idlers. As the belt progresses,

it transitions from a flat to a trough configuration to accommodate bulk material, before returning to a flat profile at the discharge point, where the material exits the system. The head pulley drives the belt and maintains the required tension, while the tail pulley assists in redirecting the belt and applying additional tension.

There are countless types of belt conveyors based on various factors. The design process begins with an analysis of the service requirements and the identification of the principal data characterizing the specific application. Workflow is illustrated in the figure below:

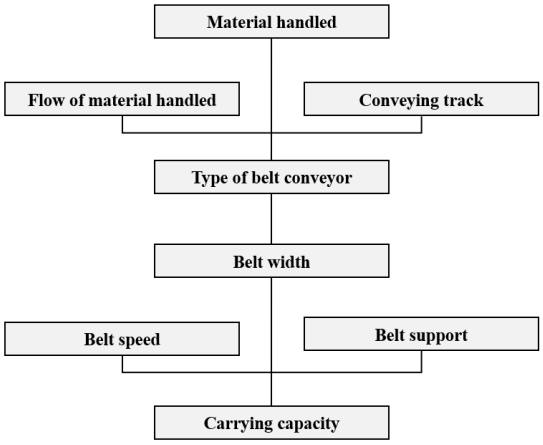


Figure 2.2. Typical design workflow for a belt conveyor system [1]

Belt conveyors are capable of transporting a wide range of materials, each with distinct properties that influence conveyor design [2]. One important factor is the angle of repose, β_R (Figure 2.3a), which is the acute angle that the surface of a normal, freely formed pile of material makes with the horizontal. Another critical factor is the angle of surcharge, β_S (Figure 2.3b), which refers to the angle that the surface of the material makes relative to the horizontal while the material is resting on a moving conveyor belt.

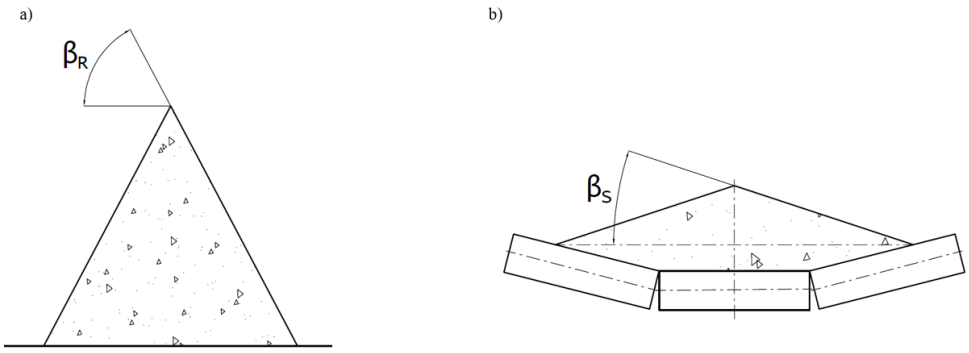


Figure 2.3. (a) Angle of repose in belt conveyor system and (b) Angle of surcharge in belt conveyor system

The flowability is determined by various material characteristics, including the size and shape of fine particles and lumps, material density, roughness or smoothness of the surface of the material particles, the proportion of fines and lumps, moisture content, and potential mechanical, chemical, or temperature effects of the material [1, 2].

Once the material to be conveyed is identified, along with its properties, the flow rate is calculated. Each conveyor is designed to handle a specific quantity of material within a given time frame, expressed as mass flow in tons per hour:

$$Q_{\mathrm{m}}\ [\mathrm{t/h}] \tag{2.1}$$

Using the bulk density of the material ρ in t/m^3, the mass flow can be converted in volume flow Q_{v} as follows:

$$Q_{\mathrm{v}} = \frac{Q_{\mathrm{m}}}{\rho}\ [\mathrm{m^3/h}] \tag{2.2}$$

Belt conveyors generally have a relatively low flight load and can be adapted to various routing configurations. This flexibility greatly influences the modeling of the steel structure supporting the conveyor. The travel path (Figure 2.4) can be adjusted with extended route lengths, allowing for the use of convex and concave vertical curves to avoid straight line constraint. This enables the conveyor to be aligned with the most economical profile alignment.

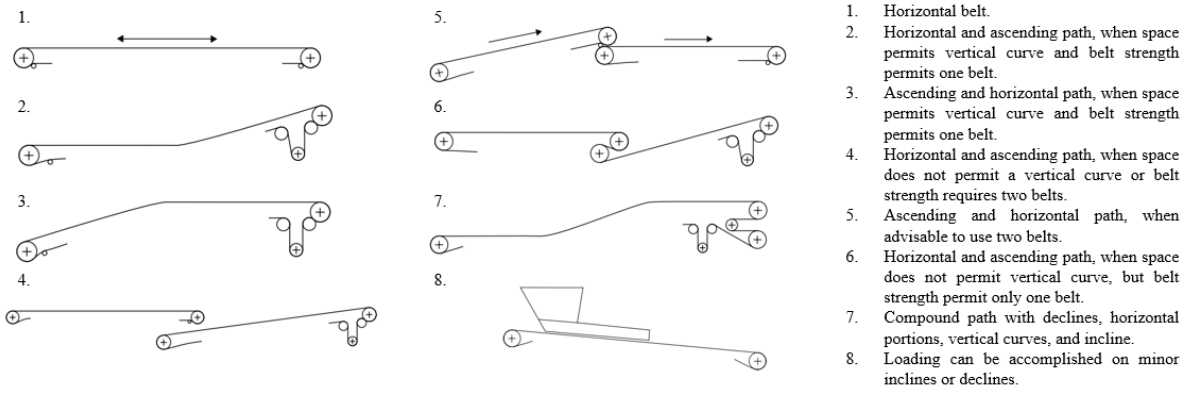


Figure 2.4. Belt conveyor routings [2]

The primary goal in selecting the type of belt conveyor is to ensure smooth and faultless transport of material without any roll-off or spillage. This guarantees a reliable flow of material. The critical conveying gradient angle, δ_{CR}, typically falls between 15° and 20°. When exceeding these critical values, specialized conveyors are required instead of standard belt conveyors without surface partitioning.

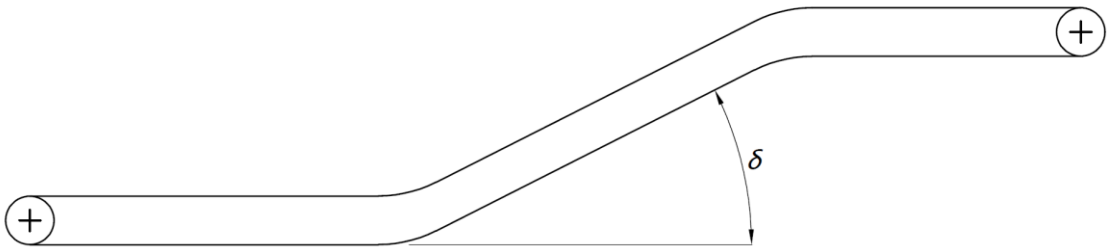


Figure 2.5. Standard belt conveyor

Belt widths, denoted as *B* in Figure 2.6, are selected based on applicable standards, such as those set by the Storage Equipment Manufacturers Association (*SEMA*), the International Organization for Standardization (*ISO*), the Deutsches Institut für Normung (*DIN*), or the Conveyor Equipment Manufacturers Association (*CEMA*). These standards provide guidance for the design process. The appropriate width is determined by factors like the size of lumps, the ratio of lumps to fines, and the material's angle of repose [1, 2].

Belt speed depends largely on the characteristics of the material being conveyed, the desired capacity, and the belt tension. For powdery materials, lower speeds are necessary to minimize dusting, particularly at loading and discharge points. Fragile materials also require reduced speeds to prevent degradation as the material moves over the idlers. Heavy, sharp-edged materials should be transported at moderate speeds.

Any difference in the forward velocity between the material being loaded and the receiving conveyor belt must be minimized to avoid turbulence. Additionally, vertical velocity during loading must be absorbed to prevent further disturbances in the material flow.

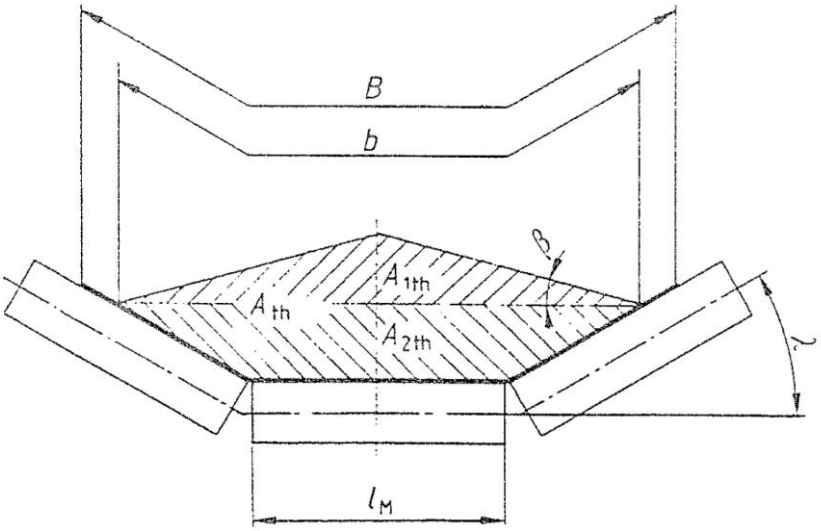


Figure 2.6. Section view of a belt conveyor [1]

The conveyor's belt is subjected to significant strain due to the impact of the material handled at the feeding point of the belt. This impact is quantified by the drop energy E_{f}, calculated as:

$$E_{\mathrm{f}} = m_{\mathrm{k}} \cdot K_{\mathrm{f}} \cdot g \cdot h_{\mathrm{f}} \,[\mathrm{Nm}] \tag{2.3}$$

where, m_{k} is the mass of a cubic lump with edge length k, K_{f} is the shape factor for lumps of different shapes, h_{f} is the height of free fall and g is the gravitational constant. All these values are selected from defined tables.

The belt conveyor support system is made of idlers designed to match the working conditions and required conveying capacity, with standardized lengths and diameters. Depending on the type and bulkiness of the material being conveyed, either a flat belt (Figure 2.7a) or a troughed belt is selected. Troughing, depending on the material demands, speed, and capacity, can vary between V-trough (Figure 2.7b), 3-part (Figure 2.7c), or 5-part configurations (Figure 2.7d).

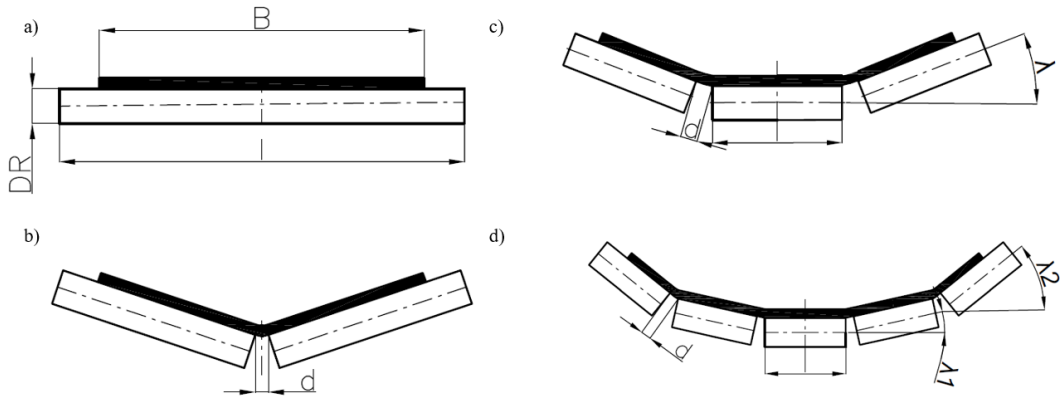


Figure 2.7. Flat and troughed belt configurations for idler systems: (a) Flat belt, (b) V-trough, (c) 3-part trough, (d) 5-part trough

Based on the selected troughing design and the previously selected belt width B, the standard idler tube length l is determined from the corresponding tables. Next, the idler diameter D_{R} is selected, ensuring that D_{R} is chosen to avoid an excessively high number of revolutions resulting from the belt speed. The idler speed is then calculated as:

$$n_{\mathrm{R}} = \frac{v}{\pi} \cdot \frac{60}{D_{\mathrm{R}}} \,[\mathrm{min}^{-1}] \tag{2.4}$$

Finally, the conveying capacity can be determined, which depends on the filling cross-section area A (as shown in Figure 2.6) and the conveying speed v. The theoretical volume capacity is calculated as:

$$Q_{\mathrm{v,th}} = A \cdot v \cdot 3600 \left[\frac{m^3}{s}\right] \tag{2.5}$$

Based on the previously selected configuration of the support system, the calculation of the filling cross-section area A can vary according to the rule shown in Eq.2.6. The corresponding types of cross-sections are illustrated in Figure 2.9:

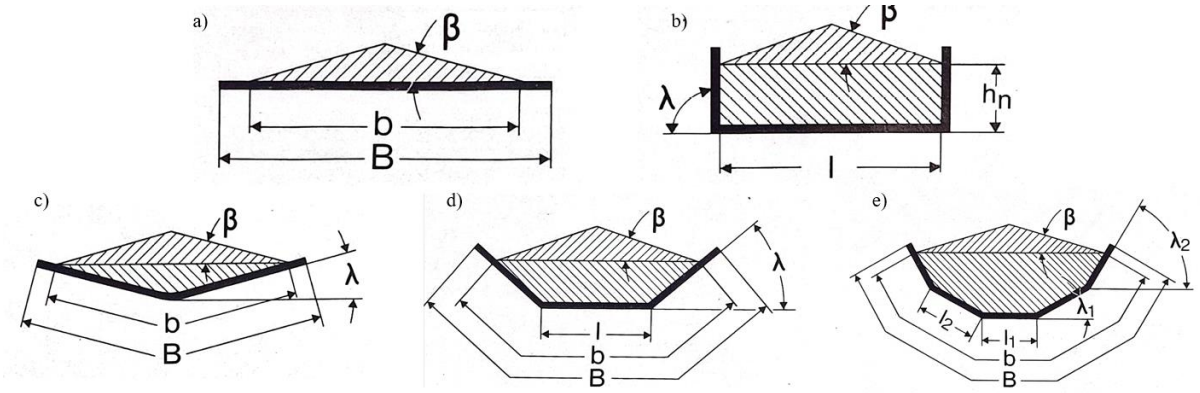


Figure 2.8. Types of cross-sections for different belt conveyor configurations: (a) Flat configuration, (b) Square trough, (c) V-trough, (d) 3-part trough, (e) 5-part trough

$$\begin{cases} A = \dfrac{b^2}{4} \cdot \tan\beta, & flat \\ A = \left(l + \dfrac{h_n}{\tan\lambda}\right) \cdot h_n + \left(\dfrac{l}{2} + \dfrac{h_n}{\tan\lambda}\right)^2 \cdot \tan\beta, & square\ trough \\ A = \dfrac{b^2}{4} \cdot \cos\lambda \cdot \sin\lambda + \left(\dfrac{b}{2} \cdot \cos\lambda\right) \cdot \tan\beta, & V-trough \\ A = \left(l + \dfrac{b-l}{2} \cdot \cos\lambda\right) \cdot \dfrac{b-l}{2} \cdot \sin\lambda + \left(\dfrac{l + (b-l) \cdot \cos\lambda}{2}\right)^2 \cdot \tan\beta, & 3-part \\ A = (l_1 + l_2 \cdot \cos\lambda_1) \cdot l_2 \cdot \sin\lambda_1 \\ + \left[l_1 + 2l_2 \cdot \cos\lambda_1 + \left(\dfrac{b - l_1 - 2l_2}{2}\right) \cdot \cos\lambda_2\right] \cdot \dfrac{(b - l_1 - 2l_2) \cdot \sin\lambda_2}{2} \\ + \left[\dfrac{l_1}{2} + l_2 \cdot \cos\lambda_1 + \left(\dfrac{b - l_1 - 2l_2}{2}\right) \cdot \cos\lambda_2\right] \cdot \tan\beta, & 5-part \end{cases} \tag{2.6}$$

Where b is the effective belt width, calculated using the rule demonstrated in Eq.2.7.

$$b = \begin{cases} 0.9 \cdot B - 50, & B \le 2000 \\ B - 250, & B > 2000 \end{cases} \tag{2.7}$$

All dimensions in the Eq.2.7 are in millimeters.

The theoretical volume capacity $Q_{\mathrm{v,th}}$ is multiplied by the bulk density ρ to obtain the theoretical capacity $Q_{\mathrm{m,th}}$:

$$Q_{\mathrm{m,th}} = Q_{\mathrm{v,th}} \cdot \rho \left[\frac{\mathrm{t}}{\mathrm{h}}\right] \tag{2.8}$$

Given a defined feeding rate ρ_1, the effective conveying capacity is:

$$Q_{\mathrm{v,eff}} = A \cdot v \cdot 3600 \left[\frac{\mathrm{m}^3}{\mathrm{s}}\right] \tag{2.9}$$

or

$$Q_{\mathrm{m,eff}} = Q_{\mathrm{v,eff}} \cdot \rho_1 \left[\frac{\mathrm{t}}{\mathrm{h}}\right] \tag{2.10}$$

2.2. Finite element method

The finite element method (FEM) is a numerical method used for solving engineering and mathematical physics problems by providing approximate solutions through the discretization of a continuum into a finite number of smaller, interconnected units known as finite elements [3].

Each finite element in the discretized continuum is interconnected with others either directly or indirectly through shared interfaces such as nodes, boundary lines, or surfaces. By applying known stress-strain relationships for the material, the behavior at each node is calculated based on the properties of the surrounding elements. The overall system is solved by combining the equations derived from each node.

There are various types of elements, including one-, two-, and three-dimensional elements. This thesis specifically addresses one-dimensional elements, commonly referred to as line elements. Line elements are represented by line segments connecting two nodes with a cross-sectional area A_{CS}. Figure 2.9 illustrates typical line element with cross-sectional area: Figure 2.9a shows a line element E_1 created by connecting two nodes, ID_1 and ID_2, while Figure 2.9b displays the cross-sectional area of corresponding line element.

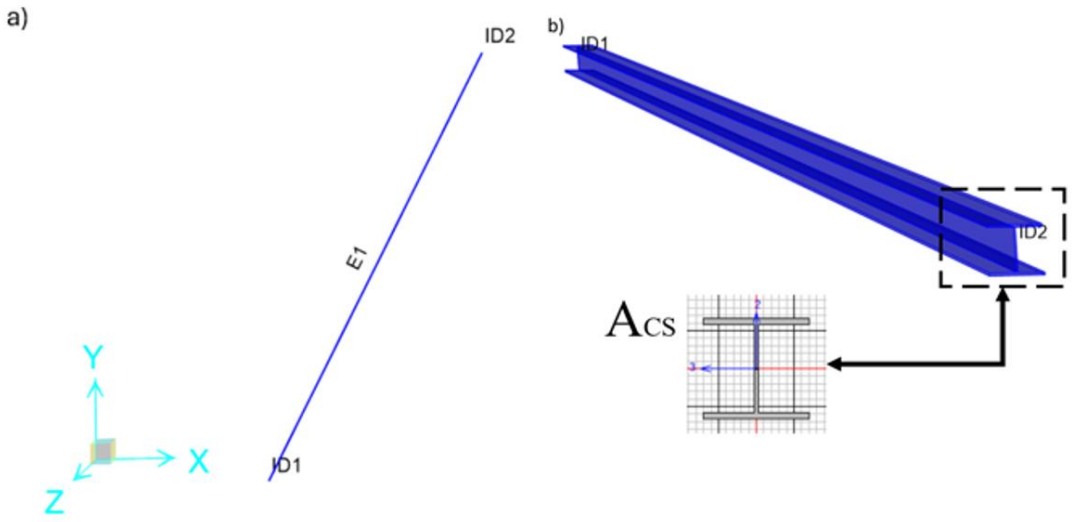


Figure 2.9. Line element with cross-sectional area in finite element analysis: (a) Line element E_1 with nodes ID1 and ID2, (b) Cross-sectional area of the line element

The finite element method involves two primary approaches: the force method and the displacement (or stiffness) method. The displacement method focuses on determining the displacements of the nodes as the unknowns. This problem is formulated through a matrix equation:

$$\begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & \dots & k_{1n} \\ k_{21} & k_{22} & k_{23} & \dots & k_{2n} \\ k_{31} & k_{32} & k_{33} & \dots & k_{3n} \\ \vdots & \boxdot & \boxdot & \boxdot & \vdots \\ k_{n1} & \boxdot & \boxdot & \dots & k_{nn} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d \\ \vdots \\ d_n \end{Bmatrix} \tag{2.11}$$

This can be compactly written as:

$$\{f\} = [k]\{d\} \tag{2.12}$$

where $\{f\}$ is the vector of element nodal forces, $[k]$ is the element stiffness matrix and $\{d\}$ is the vector of unknown generalized displacements. The individual nodal equilibrium equations for each element are then assembled into global nodal equilibrium equations:

$$\{F\} = [K]\{d\} \tag{2.13}$$

Here, $\{F\}$ represents the vector of global nodal forces, $[K]$ is the global stiffness matrix for the structure, and $\{d\}$ is the vector of known and unknown structure nodal degrees of freedom or generalized displacements. The determinant of the global stiffness matrix $[K]$ is initially singular, with a determinant of zero. To resolve this, appropriate boundary conditions must be applied, ensuring that $[K]$ becomes non-singular and thus solvable matrix.

This thesis focuses on a structure composed of *n* different beams arranged in three-dimensional space, where each beam can undergo displacements and rotations along all three axes. Therefore, the behavior of each beam must be described both in its local coordinate system and in the global coordinate system representing the entire structure.

A transformation matrix is used to obtain the general stiffness matrix for a beam element positioned in three-dimensional space. For the *n*th beam, as shown in Figure 2.10., the coordinates of ID_1 are x_1, y_1 and z_1, while the coordinates of ID_2 are x_2, y_2 and z_2. The angles θ_x, θ_y and θ_z are measured from the global *x, y* and *z* axis, respectively, to the local $\hat{x}$ axis, which is directed along the beam from node ID_1 to node ID_2.

The main target of the transformation is to solve the following equation:

$$\widehat{\boldsymbol{d}} = \boldsymbol{T}^* \boldsymbol{d} \tag{2.14}$$

where $\boldsymbol{T}^*$ represents the transformation matrix, which has to be defined. The transformation matrix enables the local displacement matrix $\widehat{\boldsymbol{d}}$ to be expressed in terms of the displacement components in the global coordinate system.

To solve this equation and determine the required transformation, the derivation of $\boldsymbol{T}^*$ begins by expressing $\widehat{\boldsymbol{d}} = \boldsymbol{d}$ in three dimensions as:

$$\hat{d}_x \hat{\boldsymbol{\imath}} + \hat{d}_x \hat{\boldsymbol{\jmath}} + \hat{d}_x \widehat{\boldsymbol{k}} = d_x \boldsymbol{i} + d_x \boldsymbol{j} + d_x \boldsymbol{k} \tag{2.15}$$

where $\hat{\boldsymbol{\imath}}, \hat{\boldsymbol{\jmath}}, \widehat{\boldsymbol{k}}$ are unit vectors associated with the local $\hat{x}, \hat{y}, \hat{z}$ axes, respectively, and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are the unit vectors associated with the global *x, y, z* axes. Taking the dot product of Eq.2.15 with $\hat{\boldsymbol{\imath}}$:

$$\hat{d}_x + 0 + 0 = d_x(\hat{\boldsymbol{\imath}} \cdot \boldsymbol{i}) + d_x(\hat{\boldsymbol{\imath}} \cdot \boldsymbol{j}) + d_x(\hat{\boldsymbol{\imath}} \cdot \boldsymbol{k}) \tag{2.16}$$

and using the definition of the dot product:

$$\hat{\boldsymbol{\imath}} \cdot \boldsymbol{i} = \frac{x_2 - x_1}{L_{E1}} = C_x \tag{2.17}$$

$$\hat{\boldsymbol{\imath}} \cdot \boldsymbol{j} = \frac{y_2 - y_1}{L_{E1}} = C_y \tag{2.18}$$

$$\hat{\boldsymbol{\imath}} \cdot \boldsymbol{k} = \frac{z_2 - z_1}{L_{E1}} = C_z \tag{2.19}$$

where L_{E1} represents the total length of element E_1, calculated as $L_{E1} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$ and C_x, C_y, and C_z represent the projections of $\hat{\boldsymbol{\imath}}$ on $\boldsymbol{i}$,

j, and ***k***, respectively, calculated as $C_{\mathrm{x}} = \cos(\theta_{\mathrm{x}})$, $C_{\mathrm{y}} = \cos(\theta_{\mathrm{y}})$ and $C_{\mathrm{z}} = \cos(\theta_{\mathrm{z}})$. Now, the $\hat{d}_{\mathrm{x}}$ can be expressed as:

$$\hat{d}_{\mathrm{x}} = C_{\mathrm{x}} d_{\mathrm{x}} + C_{\mathrm{y}} d_{\mathrm{y}} + C_{\mathrm{z}} d_{\mathrm{z}} \tag{2.20}$$

For a vector in space directed along the $\hat{x}$ axis, Eq.2.20 gives the components of that vector in the global *x, y,* and *z* directions. Using such approach, the transformation equation $\widehat{\boldsymbol{d}} = \boldsymbol{T}^{*}\boldsymbol{d}$ can be written explicitly as:

$$\begin{Bmatrix} \hat{d}_{1\mathrm{x}} \\ \hat{d}_{2\mathrm{x}} \end{Bmatrix} = \begin{bmatrix} C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} \end{bmatrix} \begin{Bmatrix} d_{1\mathrm{x}} \\ d_{1\mathrm{y}} \\ d_{1\mathrm{z}} \\ d_{2\mathrm{x}} \\ d_{2\mathrm{y}} \\ d_{2\mathrm{z}} \end{Bmatrix} \tag{2.21}$$

Thus, the transformation matrix $\boldsymbol{T}^{*}$ is:

$$\boldsymbol{T}^{*} = \begin{bmatrix} C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} \end{bmatrix} \tag{2.22}$$

The global stiffness matrix for an element, referred to the global axes, is given by:

$$\boldsymbol{k} = \boldsymbol{T}^{\mathrm{T}} \widehat{\boldsymbol{k}}\, \boldsymbol{T} \tag{2.23}$$

For this case, $\boldsymbol{T}$ is replaced by $\boldsymbol{T}^{*}$, and the global stiffness matrix $\boldsymbol{k}$ is obtained using the equation $\boldsymbol{k} = (\boldsymbol{T}^{*})^{\mathrm{T}} \widehat{\boldsymbol{k}}\, \boldsymbol{T}^{*}$ as follows:

$$\boldsymbol{k} = \begin{bmatrix} C_{\mathrm{x}} & 0 \\ C_{\mathrm{y}} & 0 \\ C_{\mathrm{z}} & 0 \\ 0 & C_{\mathrm{x}} \\ 0 & C_{\mathrm{y}} \\ 0 & C_{\mathrm{z}} \end{bmatrix} \frac{A_{\mathrm{CS}} E}{L_{\mathrm{E1}}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{\mathrm{x}} & C_{\mathrm{y}} & C_{\mathrm{z}} \end{bmatrix} \tag{2.24}$$

When calculated, the global coordinate element stiffness matrix $\boldsymbol{k}$ is equal to:

$$\boldsymbol{k} = \frac{A_{\mathrm{CS}} E}{L_{\mathrm{E1}}} \begin{bmatrix} C_{\mathrm{x}}^2 & C_{\mathrm{x}}C_{\mathrm{y}} & C_{\mathrm{x}}C_{\mathrm{z}} & -C_{\mathrm{x}}^2 & -C_{\mathrm{x}}C_{\mathrm{y}} & -C_{\mathrm{x}}C_{\mathrm{z}} \\ \square & C_{\mathrm{y}}^2 & C_{\mathrm{y}}C_{\mathrm{z}} & -C_{\mathrm{x}}C_{\mathrm{y}} & -C_{\mathrm{y}}^2 & -C_{\mathrm{y}}C_{\mathrm{z}} \\ \square & \square & C_{\mathrm{z}}^2 & -C_{\mathrm{x}}C_{\mathrm{z}} & -C_{\mathrm{y}}C_{\mathrm{z}} & 0 \\ \square & \square & \square & C_{\mathrm{x}}^2 & C_{\mathrm{x}}C_{\mathrm{y}} & C_{\mathrm{x}} \\ \square & \square & \square & \square & C_{\mathrm{y}}^2 & C_{\mathrm{y}} \\ Symm. & \square & \square & \square & \square & C_{\mathrm{z}}^2 \end{bmatrix} \tag{2.25}$$

The final structure is composed of N different beams (n = 1, 2, …, N), where nth beam is defined by two adjacent points i and j. For the nth beam, the stiffness matrix $\boldsymbol{k}^{\mathrm{n}}$ is calculated using the beam's length L^{n}, modulus of elasticity E^{n}, and cross-sectional area $A_{\mathrm{CS}}^{\mathrm{n}}$, as well as the positions of the ith and jth points. Depending on the boundary conidiations, certain elements of the displacement vector $\boldsymbol{d}^{\mathrm{n}}$ may be set to zero.

Once the stiffness matrix $\boldsymbol{k}^{\mathrm{n}}$ for the last element ($n = N$) is calculated, the total stiffness matrix $[K]$ for the structure is assembled, along with the nodal force matrix $\{F\}$, as follows:

$$[K] = \sum_{\mathrm{n}=1}^{\mathrm{N}} \boldsymbol{k}^{\mathrm{n}} \tag{2.26}$$

$$\{F\} = \sum_{\mathrm{n}=1}^{\mathrm{N}} \boldsymbol{f}^{\mathrm{n}} \tag{2.27}$$

By solving the matrix equation $\{F\} = [K]\{d\}$, the displacements for the structure are determined. Once the displacements are known, the stress for each element n, described by the ith and jth *nodes,* is calculated using:

$$\boldsymbol{\sigma}^{\mathrm{n}} = \frac{E^{\mathrm{n}}}{L^{\mathrm{n}}}\begin{bmatrix} -C_{\mathrm{x}}^{\mathrm{n}} & -C_{\mathrm{y}}^{\mathrm{n}} & -C_{\mathrm{z}}^{\mathrm{n}} & C_{\mathrm{x}}^{\mathrm{n}} & C_{\mathrm{y}}^{\mathrm{n}} & C_{\mathrm{z}}^{\mathrm{n}} \end{bmatrix} \begin{Bmatrix} d_{\mathrm{ix}} \\ d_{\mathrm{iy}} \\ d_{\mathrm{iz}} \\ d_{\mathrm{jx}} \\ d_{\mathrm{jy}} \\ d_{\mathrm{jz}} \end{Bmatrix} \tag{2.28}$$

Once Eq.2.28 is solved, the stress σ^{n} for the nth element is calculated.

2.3. Optimization algorithm

An optimization algorithm is used to incrementally improve the design until further improvement is not possible, or until the budgeted time or cost is reached. A generalized optimization problem contains multiple components, starting with the definition of the objective function:

$$minimize \quad f_{\mathrm{m}}(\boldsymbol{x}) \tag{2.29}$$

Here, f_m represents the mth objective function that needs to be minimized. Any optimization problem can be either a single-objective (where m = 1) or a multi-objective problem with M numbers of objectives (m = 1, ..., M). The vector $\boldsymbol{x}$ from Eq.2.29 represents a design vector containing values corresponding to different design variables:

$$\boldsymbol{x} = \left[x_{\mathrm{D},1}, x_{\mathrm{D},2}, \ldots, x_{\mathrm{D,n}}\right] \tag{2.30}$$

where $x_{D,i}$ denotes the *i*th design variable. Defined design vector consists of *n* different design variables. Each variable in the design vector is adjusted by the algorithm to minimize the objective function f_m within the given search space Ω:

$$\boldsymbol{x} \in \Omega \tag{2.31}$$

The search space Ω can consist of real numbers ($\Omega \subset \mathbb{R}$), natural numbers ($\Omega \subset N$), or categories. These search spaces are unlimited. However, due to limitations imposed by time or cost, the search space can also be constrained, where the *i*th design variable falls within the corresponding bounded space:

$$x_{\mathrm{D,i}} \in \langle x_{\mathrm{D,i}}^{\mathrm{L}}, x_{\mathrm{D,i}}^{\mathrm{U}} \rangle \tag{2.32}$$

where $x_{\mathrm{D,i}}^{\mathrm{L}}$ and $x_{\mathrm{D,i}}^{\mathrm{U}}$ represent the lower and upper bounds, respectively, for the *i*th design variable. Therefore, the total search space for *n* design variables is the product of all individual variable ranges:

$$\Omega = \prod_{\mathrm{i}=1}^{\mathrm{n}} \langle x_{\mathrm{D,i}}^{\mathrm{L}}, x_{\mathrm{D,i}}^{\mathrm{U}} \rangle \tag{2.33}$$

An optimization problem may also have constraints. A constraint is a condition represented as:

$$g_{\mathrm{j}}(\boldsymbol{x}) \tag{2.34}$$

where $g_j$represents the *j*th constraint. An optimization problem can have *J* constraints (*j* = 1, ..., *J*), which significantly increase the complexity of the problem. No matter how optimal the solution is, it is rejected if even a single constraint is violated. Constraints are typically expressed as ≤, ≥, or =. In cases involving strict inequalities (< or >), the feasible set does not include the constraint boundary.

The goal of optimization is to identify the best system design while adhering to a set of constraints.

2.3.1. Genetic algorithm

The genetic algorithm (GA) mimics the process of natural evolution, embodied by random selection and the survival of the fittest. The concept of genetics is based on the idea that genes within an individual's DNA define their traits and attributes [4].

Genes are functions within DNA and chromosomes are sequences of genes that collectively define a particular trait. A chromosome is represented as:

$$C_{\mathrm{i,j}} = \left(G_{1,\mathrm{j}}, G_{2,\mathrm{j}}, \ldots, G_{\mathrm{n,j}}\right) \tag{2.35}$$

where $C_{\mathrm{i,j}}$ represents the *i*th chromosome of the *j*th type, made of *n* different *j*th type genes $G_{\mathrm{n,j}}$. For instance, focusing only on an individual's height, Table 2.1 illustrates how the *i*th height chromosome can be formulated [4].

Table 2.1. Representation of height chromosome with gene attributes

Gene Attributes	Type	*j*th Gene	Chromosome
Tall	*Height*	$G_{1,\mathrm{height}} = Tall$	$C_{\mathrm{i,height}}$
Tall	*Height*	$G_{2,\mathrm{height}} = Tall$	
Short	*Height*	$G_{3,\mathrm{height}} = Short$	
Short	*Height*	$G_{4,\mathrm{height}} = Short$	
…	…	…	
Short	*Height*	$G_{\mathrm{n,height}} = Short$	

After applying Eq.2.35, the *i*th height chromosome $C_{\mathrm{i,height}}$ is represented as $C_{\mathrm{i,height}} = (Tall, Tall, Short, Short, \ldots, Short)$. This chromosome can also be encoded as a binary string, similar to how DNA is represented. A binary string of length *n*, depicted in Figure 2.10., corresponds to [5].

Figure 2.10. Binary string representation of a chromosome

The sampling method defines a random initial population by creating *m* different individual samples $C_{\mathrm{i,j}}$, which collectively form a defined population *P*:

$$P = \left(C_{1,\mathrm{j}}, C_{2,\mathrm{j}}, \ldots, C_{1,\mathrm{m}}\right) \tag{2.36}$$

A visualization of this rule is provided in Figure 2.11.

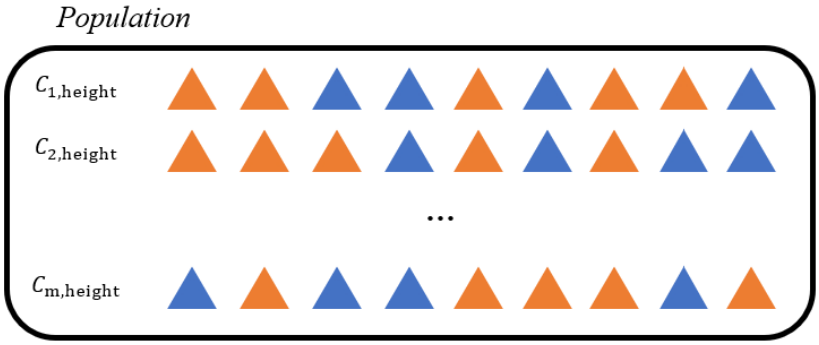


Figure 2.11. Visualization of the initial population formation

After the population has been formed, a selection process begins in which the algorithm selects chromosomes to serve as parents for the next generation. Given a population *P* with *m* chromosomes, the selection method produces a list of *m* parental pairs for the *m* children of the next generation. The selected pair may contain duplicates which can be eliminated by the algorithm.

For this thesis, tournament selection (Figure 2.12) is employed. The tournament selection process starts with a population *P* consisting of *m* chromosomes. Out of these *m* chromosomes, only *k* are randomly chosen to participate in the tournament. The fittest chromosome among these *k* participants is selected to proceed to the next stage of the algorithm, which is the crossover stage.

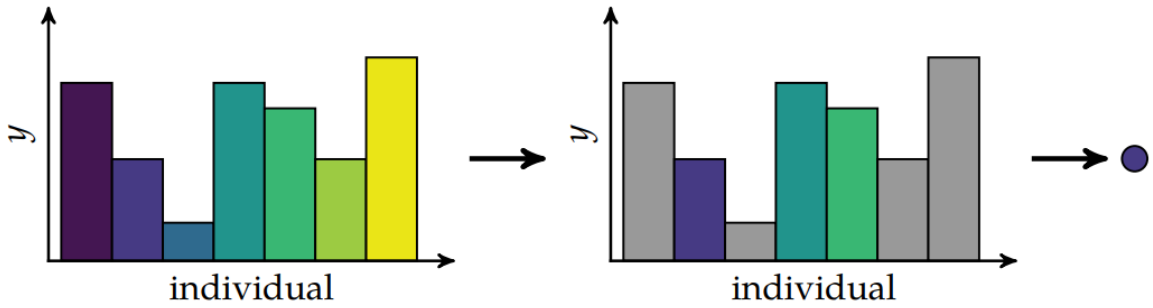


Figure 2.12. Tournament selection process for genetic algorithms

During this stage, the chromosomes of the selected parents are combined to produce children. In a single-point crossover, the first portion of parent A's chromosome forms the first portion of the child's chromosome, while the remaining portion of parent B's chromosome completes the child's chromosome. The crossover point, where the transition from one parent's chromosome to the other's occurs, is determined uniformly at random.

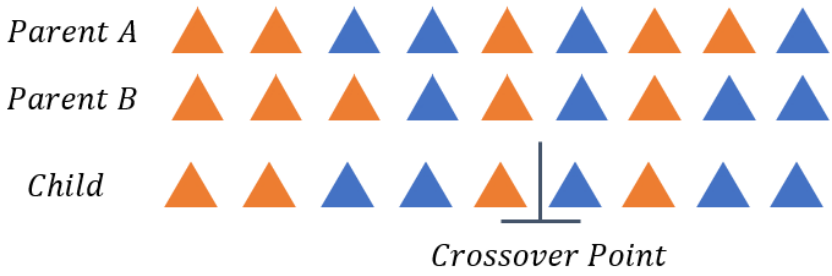


Figure 2.13. Single-point crossover in genetic algorithms

After the child's chromosomes are created from the crossover of parents, they enter the mutation stage (Figure 2.14). During mutation, each bit in a binary-valued chromosome has a small probability of being flipped, as determined by a defined mutation rate. The primary purpose of mutation is to introduce new traits spontaneously, allowing the genetic algorithm to explore a broader search space and discover potentially better solutions.

Figure 2.14. Mutation process in genetic algorithms

Genetic algorithms utilize evolutionary principles to explore complex solution spaces, including selection, crossover, and mutation. This method is particularly effective in scenarios where traditional optimization techniques struggle with non-linear, high-dimensional, or poorly understood problem landscapes. By encouraging diversity through mutation and crossover, genetic algorithms can escape local optima and offer solutions across a wide range of applications. The flexibility of this approach makes it suitable for solving problems that require balancing exploration with exploitation of the search space.

2.4. Parametrization

To enable changes to any segment of the analysis, various parameterizations must be performed. These parameterizations can involve aspects such as geometry, material properties, loads, and other factors. Parameterization is achieved by creating a series of functions that take specific parameters, enabling them to easily apply the necessary changes.

The typical application of parametrization is illustrated in Figure 2.15, with Figure 2.15a showing the processes used for modifying and creating the model. Alternatively, Figure 2.15b demonstrates how optimization stage can be implemented. After the analysis is complete, the

extracted values enter the optimization process as design variables, which are then optimized. These design variables, along with non-optimized inputs, are then fed back into the parameterization functions, where the model is redefined with new inputs.

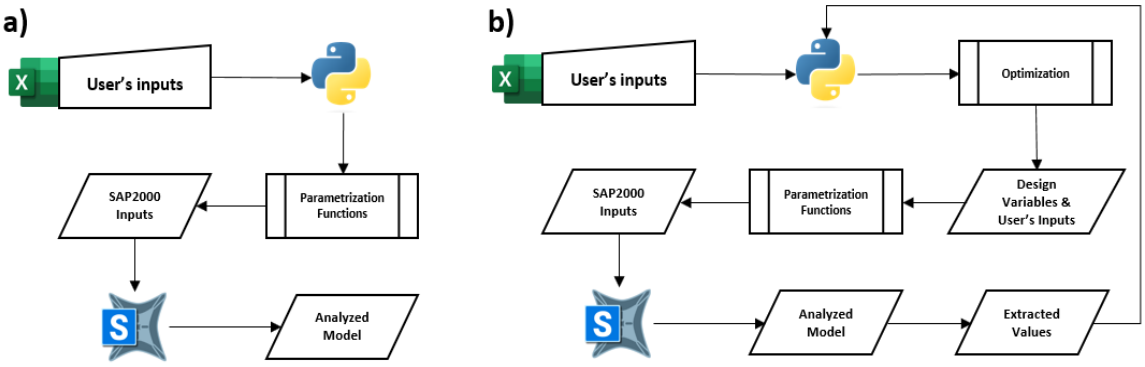


Figure 2.15. Application of parametrization in model modification and optimization: (a) Processes for modifying and creating the model, (b) Optimization using extracted values

3. Model definition

In the following section, a detailed description is provided of all the necessary steps for parameterizing the model. This includes defining all the nodes and calculating their coordinates, initializing nodes within *SAP2000*, and connecting the nodes to create frames. Furthermore, boundary conditions and connections will be described, along with the methods used for modeling them. Finally, actions and action combinations according to the Eurocode will be defined, which will be automatically generated for the model.

3.1. Geometry

Within the *Excel* file, the user defines a series of points by entering measured coordinates. Before inputting the data, the user has measured the point's coordinates in an *AutoCAD* drawing. The defined inputs consist of *n* points with their corresponding coordinates:

$$P_{\mathrm{t,i}} = \left(y_{\mathrm{t,i}}, z_{\mathrm{t,i}}\right) \tag{3.1}$$

where $P_{\mathrm{t,i}}$ represents the *i*th point of type *t*, and $\left(y_{\mathrm{t,i}}, z_{\mathrm{t,i}}\right)$ denote the point's *y* and *z* coordinates within the global coordinate system. Table 3.1 shows an example of the defined input made up from *n* points. As illustrated in Table 3.1, there are three types of points created by the user. $P_{\mathrm{S,i}}$ denotes the global *start* point of the structure, while $P_{\mathrm{E,i}}$ represent the global *end* point. Both are unique (i =1), meaning there can only be one start point and one end point. Conversely, $P_{\mathrm{C,i}}$ in Table 3.1 represents the *i*th *connection* point. Unlike start and end points, there can be either a single connection point (i = 1) or multiple *m* connection points (i = 1, …, m).

Table 3.1. Example of defined points and their types

Points	Type	Quantity
$P_{\mathrm{S,i}}$	*Start Point*	i = 1
$P_{\mathrm{C,i}}$	*Connection Point*	i = (1, ..., m)
...	...	
$P_{\mathrm{C,i}}$	*Connection Point*	
$P_{\mathrm{E,i}}$	*End Point*	i = 1

The points are then used within the tool to form the required vectors:

$$\boldsymbol{V}_{\mathrm{IP}} = \begin{bmatrix} P_{\mathrm{S},1} \\ P_{\mathrm{C},1} \\ \cdots \\ P_{\mathrm{C,m}} \\ P_{\mathrm{E},1} \end{bmatrix}; \boldsymbol{V}_{\mathrm{CP}} = \begin{bmatrix} P_{\mathrm{C},1} \\ P_{\mathrm{C},2} \\ \cdots \\ P_{\mathrm{C,m-1}} \\ P_{\mathrm{C,m}} \end{bmatrix}; \boldsymbol{V}_{\mathrm{SE}} = \begin{bmatrix} P_{\mathrm{S},1} \\ P_{\mathrm{E},1} \end{bmatrix} \tag{3.2}$$

Here, $\boldsymbol{V}_{\mathrm{IP}}$ represents a vector containing all *initial* points (*IP*), while $\boldsymbol{V}_{\mathrm{CP}}$ includes only the *connection* points (*CP*). Lastly, $\boldsymbol{V}_{\mathrm{SE}}$ is composed solely of the *start-end* points (*SE*). These vectors serve various functions within the tool.

Connection points indicate where two galleries converge, as shown in Figure 3.1a. Therefore, $P_{\mathrm{C,i}}$ cannot be part of vectors or groups representing isolated galleries, because the properties, releases, and other parameters differ. Additionally, $P_{\mathrm{C,i}}$ also marks a point where angle of a single gallery changes.

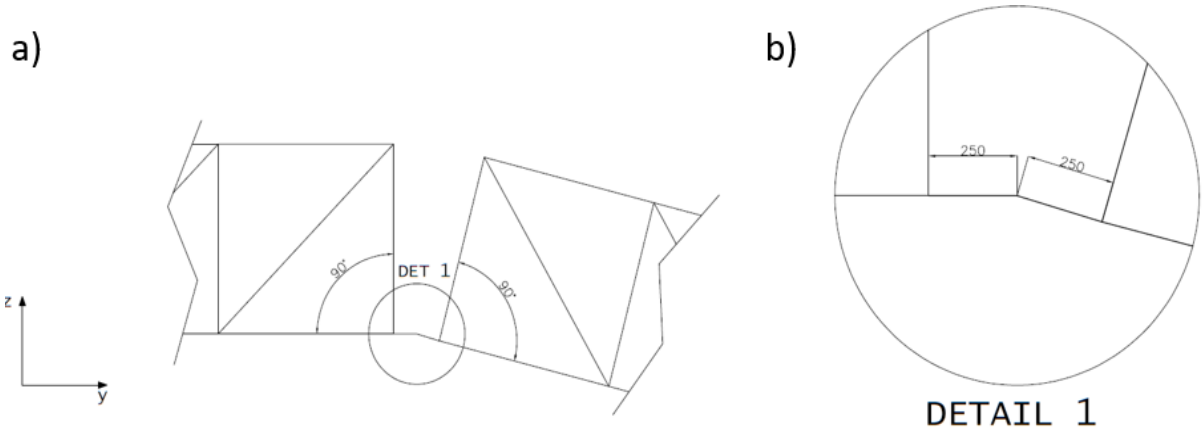


Figure 3.1. a) Connection between adjacent galleries, b) Detail view of connections between adjacent galleries with required dimensions

According to the detail shown in Figure 3.1b, it is necessary to model offset points that are 0.25 *m* away from the $P_{\mathrm{C,i}}$ in both directions along the *y*-axis and are located on the line connecting two adjacent points from $\boldsymbol{V}_{\mathrm{IP}}$. These offset points are defined as:

$$P_{\mathrm{NO,i}} = \left(y_{\mathrm{NO,i}}, z_{\mathrm{NO,i}}\right) \tag{3.3}$$

$$P_{\mathrm{PO,i}} = \left(y_{\mathrm{PO,i}}, z_{\mathrm{PO,i}}\right) \tag{3.4}$$

where $P_{\mathrm{NO,i}}$ represents the offset point positioned from the $P_{\mathrm{C,i}}$ in the negative *y* direction of the global coordinate system (*Negative Offset*), while $P_{\mathrm{PO,i}}$ represents the offset point positioned from the $P_{\mathrm{C,i}}$ in the positive *y* direction of the global coordinate system (*Positive Offset*). In Eq.3.3, $\left(y_{\mathrm{NO,i}}, z_{\mathrm{NO,i}}\right)$ represent the coordinates of $P_{\mathrm{NO,i}}$, while $\left(y_{\mathrm{PO,i}}, z_{\mathrm{PO,i}}\right)$ in Eq.3.4

represent the coordinates of $P_{\mathrm{PO,i}}$. The calculations for $P_{\mathrm{NO,i}}$ and $P_{\mathrm{PO,i}}$ are provided in Eq.3.5 and Eq.3.6, respectively. The coordinates are given by:

$$\left(y_{\mathrm{NO,i}}, z_{\mathrm{NO,i}}\right) = \left(y_{\mathrm{i+1}}^{\mathrm{NO}} - y_{\mathrm{SV,i}}^{\mathrm{NO}}, z_{\mathrm{i+1}}^{\mathrm{NO}} - z_{\mathrm{SV,i}}^{\mathrm{NO}}\right) \tag{3.5}$$

$$\left(y_{\mathrm{PO,i}}, z_{\mathrm{PO,i}}\right) = \left(y_{\mathrm{i}}^{\mathrm{PO}} + y_{\mathrm{SV,i}}^{\mathrm{PO}}, y_{\mathrm{i}}^{\mathrm{PO}} + y_{\mathrm{SV,i}}^{\mathrm{PO}}\right) \tag{3.6}$$

Here, starting from $P_{\mathrm{NO,i}}$, $\left(y_{\mathrm{i+1}}^{\mathrm{NO}}, z_{i+1}^{\mathrm{NO}}\right)$ represent the *y* and *z* coordinates of the *i*th point in the newly created vector $\boldsymbol{V}_{\mathrm{NO}} = \begin{bmatrix} P_{\mathrm{S,1}} \\ \cdots \\ P_{\mathrm{C,m}} \end{bmatrix}$, which excludes the last value from $\boldsymbol{V}_{\mathrm{IP}}$ because a *Negative Offset* point is not required between $P_{\mathrm{C,m}}$ and $P_{\mathrm{E,1}}$. Conversely, $\left(y_{\mathrm{i}}^{\mathrm{PO}}, z_{\mathrm{i}}^{\mathrm{PO}}\right)$ represent *y* and *z* coordinate for the $P_{\mathrm{PO,i}}$, selected from the new vector $\boldsymbol{V}_{\mathrm{PO}} = \begin{bmatrix} P_{\mathrm{C,1}} \\ \cdots \\ P_{\mathrm{E,1}} \end{bmatrix}$, which excludes the first value from $\boldsymbol{V}_{\mathrm{IP}}$ since *Positive Offset* point is not needed between $P_{\mathrm{S,1}}$ and $P_{\mathrm{C,1}}$. Furthermore, the coordinates $\left(y_{\mathrm{SV,i}}^{\mathrm{NO}}, z_{\mathrm{SV,i}}^{\mathrm{NO}}\right)$ are obtained from the *Negative Offset* scaled vector (*SV),* calculated in Eq.3.7, while $\left(y_{\mathrm{SV,i}}^{\mathrm{PO}}, z_{\mathrm{SV,i}}^{\mathrm{PO}}\right)$ are derived from *Positive Offset* scaled vector, as calculated in Eq.3.8. The vectors are defined as follows:

$$\overrightarrow{P_{\iota+1}^{\mathrm{NO}} P_{\iota}^{\mathrm{NO}}}' = \overrightarrow{u_{\mathrm{NO}}} \cdot k \tag{3.7}$$

$$\overrightarrow{P_{\iota}^{\mathrm{PO}} P_{\iota+1}^{\mathrm{PO}}}' = \overrightarrow{u_{\mathrm{PO}}} \cdot k \tag{3.8}$$

Here, $\overrightarrow{P_{\iota+1}^{\mathrm{NO}} P_{\iota}^{\mathrm{NO}}}'$ represents the *i*th *Negative Offset* scaled vector between two adjacent points selected from $\boldsymbol{V}_{NO}$. The scale factor *k* multiples the *i*th *Negative Offset* unit vector $\overrightarrow{u_{\mathrm{NO}}}$, which is calculated as $\overrightarrow{u_{\mathrm{NO}}} = \frac{\overrightarrow{P_{\mathrm{i+1}}^{\mathrm{NO}} P_{\mathrm{1}}^{\mathrm{NO}}}}{\|\overrightarrow{P_{\mathrm{i+1}}^{\mathrm{NO}} P_{\mathrm{1}}^{\mathrm{NO}}}\|}$, where $\overrightarrow{P_{\mathrm{i+1}}^{\mathrm{NO}} P_{\mathrm{1}}^{\mathrm{NO}}}$ is the *Negative Offset* direction vector between two adjacent points, calculated in Eq.3.9, and $\|\overrightarrow{P_{\mathrm{i+1}}^{\mathrm{NO}} P_{\mathrm{1}}^{\mathrm{NO}}}\|$ is the *Negative Offset* normalized vector between corresponding points, as calculated in Eq.3.10. Similarly, $\overrightarrow{u_{\mathrm{PO}}}$ represents the *Positive Offset* unit vector, calculated as $\overrightarrow{u_{\mathrm{PO}}} = \frac{\overrightarrow{P_{\mathrm{1}}^{\mathrm{PO}} P_{\mathrm{i+1}}^{\mathrm{PO}}}}{\|\overrightarrow{P_{\mathrm{1}}^{\mathrm{PO}} P_{\mathrm{i+1}}^{\mathrm{PO}}}\|}$, where $\overrightarrow{\mathrm{P_1^{PO}P_{i+1}^{PO}}}$ is the *Positive Offset* direction vector between two adjacent points, calculated in Eq.3.11, and $\|\overrightarrow{P_{\mathrm{1}}^{\mathrm{PO}} P_{\mathrm{i+1}}^{\mathrm{PO}}}\|$ is the *Positive Offset* normalized vector between corresponding points, as calculated from Eq.3.12. The scale factor *k*, which is 0.25 *m,* is equal to required distance from the offset point and is same for both calculations.

$$\overrightarrow{P_{\mathrm{i+1}}^{\mathrm{NO}} P_{\mathrm{1}}^{\mathrm{NO}}} = \left(y_{\mathrm{i+1}}^{\mathrm{NO}} - y_{\mathrm{i}}^{\mathrm{NO}}, z_{\mathrm{i+1}}^{\mathrm{NO}} - z_{\mathrm{i}}^{\mathrm{NO}}\right) \tag{3.9}$$

$$\|\overrightarrow{P_{i+1}^{NO}P_{i}^{NO}}\| = \sqrt{\left(y_{i+1}^{NO} - y_{i}^{NO}\right)^2 + \left(z_{i+1}^{NO} - z_{i}^{NO}\right)^2} \tag{3.10}$$

$$\overrightarrow{P_{i}^{PO}P_{i+1}^{PO}} = \left(y_{i}^{PO} - y_{i+1}^{PO}, z_{i}^{PO} - z_{i+1}^{PO}\right) \tag{3.11}$$

$$\|\overrightarrow{P_{i}^{PO}P_{i+1}^{PO}}\| = \sqrt{\left(y_{i}^{PO} - y_{i+1}^{PO}\right)^2 + \left(z_{i}^{PO} - z_{i+1}^{PO}\right)^2} \tag{3.12}$$

The calculated points are used to form a newly required vector:

$$\boldsymbol{V}_{SEO} = \begin{bmatrix} P_{S,1} \\ P_{NO,i} \\ P_{PO,i} \\ \cdots \\ P_{NO,m} \\ P_{PO,m} \\ P_{E,1} \end{bmatrix} \tag{3.13}$$

where $\boldsymbol{V}_{SEO}$ is a vector composed of *start – end* points ($P_{S,1}$, $P_{E,1}$) and *m* different *Positive* and *Negative Offset* points ($P_{NO,m}$, $P_{PO,m}$). Table 3.2 illustrates how pairs of points are initialized for further calculations within the defined tool.

Table 3.2. Initialization of point pairs for gallery segments

***j*th Pair**	P_{GS}	P_{ES}	**Quantity**
$(P_{GS}, P_{ES})_j$	$P_{S,1}$	$P_{NO,i}$	
$(P_{GS}, P_{ES})_j$	$P_{PO,i}$	$P_{NO,i+1}$	
...	...	...	$j = 1, ..., J$
$(P_{GS}, P_{ES})_j$	$P_{PO,m-1}$	$P_{NO,m}$	
$(P_{GS}, P_{ES})_j$	$P_{PO,m}$	$P_{E,1}$	

Here, P_{GS} represents an individual *gallery start* (*GS*) point with coordinates $(y_{GS}, z_{GS})_j$, and P_{ES} represents an individual *gallery end* (*GE*) point with coordinates $(y_{ES}, z_{ES})_j$. The *j*th gallery is described by the corresponding *j*th pair $(P_{GS}, P_{ES})_j$, as depicted in Figure 3.5.

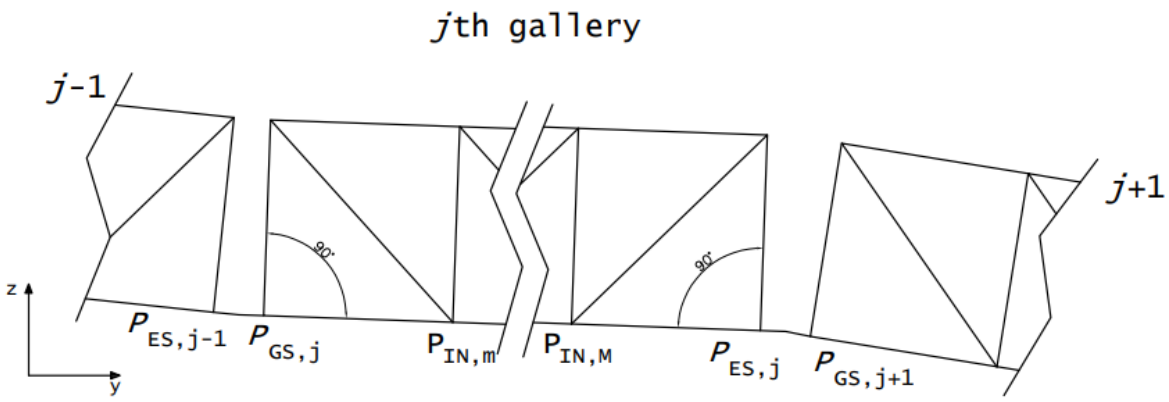


Figure 3.2. Points of the jth gallery

The *j*th gallery consists of *M* different *interior* points (*IN*):

$$\left(P_{\mathrm{IN,m}}\right)_{\mathrm{j}} = \left(y_{\mathrm{IN,m}}, z_{\mathrm{IN,m}}\right)_{\mathrm{j}} \tag{3.14}$$

where $\left(P_{\mathrm{IN,m}}\right)_{\mathrm{j}}$ represents the *m*th *interior* point of the corresponding *j*th gallery, as illustrated by Figure 3.2. The *j*th gallery has *M* individual *interior* points (m = 1, …, M). In Eq.3.14, $\left(y_{\mathrm{IN,m}}\right)_{\mathrm{j}}$ denotes the *y*-coordinate of the *m*th *interior* point, while $\left(z_{\mathrm{IN,m}}\right)_{\mathrm{j}}$ represent the *z*-coordinate of the *m*th *interior* point from the common *j*th gallery. The coordinates are determined by:

$$\left(y_{,\mathrm{i}}\right)_{\mathrm{j}} = \left(y_{\mathrm{i-1}} + \Delta y_{\mathrm{m}}\right)_{\mathrm{j}} \tag{3.15}$$

$$\left(z_{,\mathrm{i}}\right)_{\mathrm{j}} = \left(z_{\mathrm{i-1}} + \Delta z_{\mathrm{m}}\right)_{\mathrm{j}} \tag{3.16}$$

where $\left(y_{\mathrm{i-1}}\right)_{\mathrm{j}}$ and $\left(z_{\mathrm{i-1}}\right)_{\mathrm{j}}$ represent the *y* and *z* coordinates of the previous point in the *j*th gallery, respectively. The displacements $\left(\Delta y_{\mathrm{m}}\right)_{\mathrm{j}}$ and $\left(\Delta z_{\mathrm{m}}\right)_{\mathrm{j}}$, calculated from Eq.3.17 and Eq.3.18, describe how much the current point (*i*) has moved from the previous point (*i-1*) within the *j*th gallery.

$$\left(\Delta y_{\mathrm{m}}\right)_{\mathrm{j}} = \left(cos\theta \cdot L_{\mathrm{rel.}}\right)_{\mathrm{j}} \tag{3.17}$$

$$\left(\Delta z_{\mathrm{m}}\right)_{\mathrm{j}} = \left(sin\theta \cdot L_{\mathrm{rel.}}\right)_{\mathrm{j}} \tag{3.18}$$

where θ is the angle of the *j*th gallery, calculated using the coordinates of the *start* and *end* points of the *j*th gallery, $\left(P_{\mathrm{GS}}, P_{\mathrm{ES}}\right)_{\mathrm{j}}$, as follows: $\left(\theta\right)_{\mathrm{j}} = tan\left(\frac{z_{\mathrm{ES}} - z_{\mathrm{GS}}}{y_{\mathrm{ES}} - y_{\mathrm{GS}}}\right)_{\mathrm{j}}$. $L_{\mathrm{rel.}}$ from Eq.3.19 is determined based on the following rule:

$$(L_{\text{rel.}})_{\text{j}} = \begin{cases} (L_{\text{red.}})_{\text{j}}, & i \leq \frac{n_{\text{red.}}}{2} \\ \text{L}_{\text{FS}}, & \frac{n_{\text{red.}}}{2} \leq i \leq \frac{n_{\text{red.}}}{2} + (n_{\text{FS}})_{\text{j}} \\ (L_{\text{red.}})_{\text{j}}, & \frac{n_{\text{red.}}}{2} + (\text{n}_{\text{FS}})_{\text{j}} \leq i \leq n \end{cases} \tag{3.19}$$

Here, $(L_{\text{rel.}})_{\text{j}}$ represents the relative (*rel.*) length of the *j*th gallery, which can be either L_{FS} or $(L_{\text{red.}})_{\text{j}}$ depending on the condition. L_{FS} is the full-sized (*FS*) allowed distance between two adjacent interior points of the *j*th gallery, specified by the user in *Excel* and is same for all galleries, hence no *j* index is needed. In contrast, $(L_{\text{red.}})_{\text{j}}$ represents reduced (*red.*) length calculated within the tool, given by $(L_{\text{red.}})_{\text{j}} = \frac{1}{2}\left(L_2 - \frac{(n_{\text{FS}})_{\text{j}}}{2}\right)$ for the *j*th gallery, to respect the design rules specified by the structure designer. Here, $(n_{\text{FS}})_{\text{j}}$ represents the number of possible L_{FS} segments that can be used within the *j*th gallery, calculated as $L_{\text{FS}} = \left(\lfloor\frac{L_2}{L_{\text{FS}}}\rfloor - 1\right) \cdot 2$ where L_2, in both equations, represent half the length of the *j*th gallery, calculated as $L_2 = \frac{1}{2}\left(\sqrt{(y_{\text{ES}} - y_{\text{GS}})^2 + (z_{\text{ES}} - z_{\text{GS}})^2}\right)$. On the other hand, $n_{\text{red.}}$ represents the number of possible $(L_{\text{red.}})_{\text{j}}$ values for a single gallery. The $n_{\text{red.}}$ has a fixed value of $n_{\text{red.}} = 4$, which is why, like L_{FS}, it does not have a *j* index. The sum of two numbers gives the value of *n* which is calculated as $M = \sum_{\text{t=red.}}^{\text{FS}} n_{\text{t}}$.

The calculated points are used to form a new matrix:

$$\boldsymbol{M}_{\text{BP}} = \begin{bmatrix} (P_{\text{GS}})_{\text{j}} & (P_{\text{GS}})_{\text{j+1}} & \dots & (P_{\text{GS}})_{\text{J}} \\ (P_{\text{IN,m}})_{\text{j}} & (P_{\text{IN,m}})_{\text{j+1}} & \dots & (P_{\text{IN,m}})_{\text{J}} \\ \vdots & \vdots & \dots & \vdots \\ (P_{\text{IN,M}})_{\text{j}} & (P_{\text{IN,M}})_{\text{j+1}} & \dots & (P_{\text{IN,M}})_{\text{J}} \\ (P_{\text{ES}})_{\text{j}} & (P_{\text{ES}})_{\text{j+1}} & \dots & (P_{\text{ES}})_{\text{J}} \end{bmatrix} \tag{3.20}$$

where $\boldsymbol{M}_{\text{BP}}$ represents the matrix containing the *j*th gallery *bottom* points (*BP*), starting from *j*th gallery *start* point $(P_{\text{GS}})_{\text{j}}$, progressing through *M* total *interior* points $(P_{\text{IN,M}})_{\text{j}}$ and ending with *j*th gallery *end* point $(P_{\text{ES}})_{\text{j}}$. The matrix $\boldsymbol{M}_{\text{BP}}$ has a total of *J* columns (*j* = 1, …, *J*), where each column represents the points of a specific *j*th gallery, containing a total of *I* points.

The current calculation was performed in two-dimensional *y*-*z* space. To fully define each point in three-dimensional space, it is necessary to add an *x* coordinate to each point from $\boldsymbol{M}_{\text{BP}}$ and create *n* new matrices, each representing a corresponding part of the gallery.

Dimension *A* represents the width of the belt conveyor, including all necessary supporting elements, while dimension *B* refers to the width of the walkway between the belt conveyor and

the outer edge of the gallery. Half the width of the belt $A_2 = \frac{A}{2}$ is added to each point from the $\boldsymbol{M}_{\mathrm{BP}}$, resulting in a new matrix $\boldsymbol{M}_{\mathrm{PBW}}$, which represents the *positive belt width* points (*PBW)*. Conversely, adding negative A_2 creates the matrix $\boldsymbol{M}_{\mathrm{NBW}}$, which represents the *negative belt width* points (*NBW)*. To define the outer part of the galleries, $(A_2 + B)$ is added to $\boldsymbol{M}_{\mathrm{BP}}$, creating an *outer positive* points (*OP*) matrix $\boldsymbol{M}_{\mathrm{OP}}$, while subtracting $(A_2 + B)$ forms an *outer negative* points (*ON*) matrix $\boldsymbol{M}_{\mathrm{ON}}$. Additionally, $\boldsymbol{M}_0$ matrix is created by adding an *x*-coordinate of 0 to the corresponding points.

Matrices are transmitted from *Python* to *SAP2000* to generate points within the software, where each point is uniquely identified by an associated *ID*. These *ID*s represent the points with their respective coordinates and are returned to *Python*, where they are organized into matrices, as shown in Eq.3.21:

$$\boldsymbol{M}_{\mathrm{PBW}}^{\mathrm{ID}} = \begin{bmatrix} (ID_{\mathrm{i}})_{\mathrm{j}} & \dots & (ID_{\mathrm{i}})_{\mathrm{J}} \\ \vdots & \ddots & \vdots \\ (ID_{\mathrm{I}})_{\mathrm{j}} & \dots & (ID_{\mathrm{I}})_{\mathrm{J}} \end{bmatrix}; \boldsymbol{M}_{\mathrm{NBW}}^{\mathrm{ID}} = \begin{bmatrix} (ID_{\mathrm{i}})_{\mathrm{j}} & \dots & (ID_{\mathrm{i}})_{\mathrm{J}} \\ \vdots & \ddots & \vdots \\ (ID_{\mathrm{I}})_{\mathrm{j}} & \dots & (ID_{\mathrm{I}})_{\mathrm{J}} \end{bmatrix};$$

$$\boldsymbol{M}_{\mathrm{OP}}^{\mathrm{ID}} = \begin{bmatrix} (ID_{\mathrm{i}})_{\mathrm{j}} & \dots & (ID_{\mathrm{i}})_{\mathrm{J}} \\ \vdots & \ddots & \vdots \\ (ID_{\mathrm{I}})_{\mathrm{j}} & \dots & (ID_{\mathrm{I}})_{\mathrm{J}} \end{bmatrix}; \boldsymbol{M}_{\mathrm{ON}}^{\mathrm{ID}} = \begin{bmatrix} (ID_{\mathrm{i}})_{\mathrm{j}} & \dots & (ID_{\mathrm{i}})_{\mathrm{J}} \\ \vdots & \ddots & \vdots \\ (ID_{\mathrm{I}})_{\mathrm{j}} & \dots & (ID_{\mathrm{I}})_{\mathrm{J}} \end{bmatrix} \tag{3.21}$$

These matrices correspond to $\boldsymbol{M}_{\mathrm{PBW}}$, $\boldsymbol{M}_{\mathrm{NBW}}$, $\boldsymbol{M}_{\mathrm{OP}}$, $\boldsymbol{M}_{\mathrm{ON}}$, but are now described solely by their unique point *ID*s, while maintaining the same structure as before. Each matrix contains a total of *J* columns (*j* = 1, …, *J*), representing a total number of galleries, where each *j*th column contains a unique number of rows (*i = 1, …, I*), with *I* representing the total number of *ID*s for the corresponding *j*th gallery.

For future reference, this procedure will be described by the function in Eq.3.22:

$$\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}} = f_{\mathrm{ID}}(\boldsymbol{M}_{\mathrm{n}}) \tag{3.22}$$

where f_{ID} is the function required for generating points with corresponding *ID*s in *SAP2000*. It uses the initially established coordinates stored in $\boldsymbol{M}_{\mathrm{n}}$ and retrieves the corresponding coordinates in $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$.

In *SAP2000*, each element is defined by two unique *ID*s. Figure 3.3 illustrates this concept: Figure 3.3a shows two initial *ID*s, while Figure 3.3b demonstrates how these *ID*s are connected to form a frame.

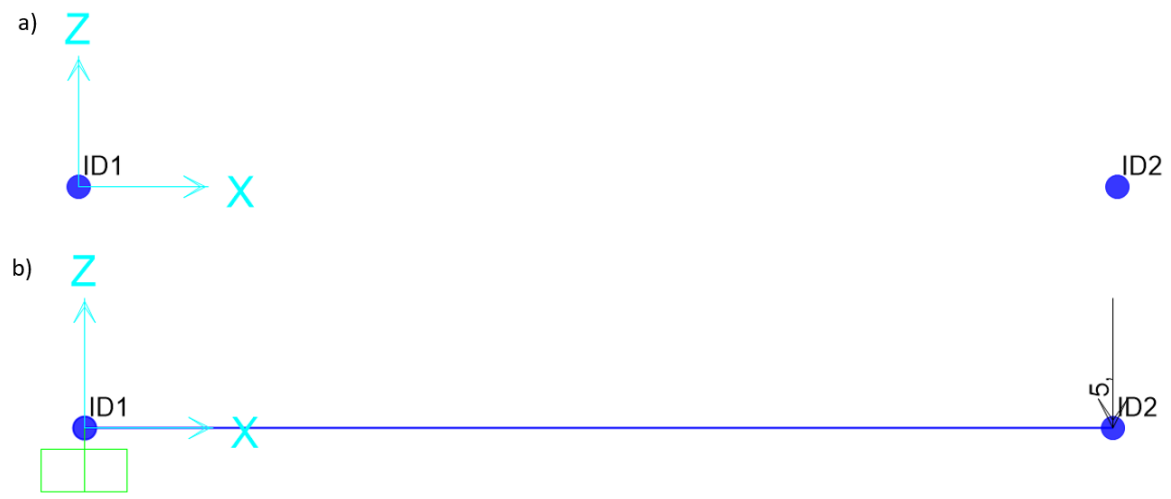


Figure 3.3. Mapping of unique IDs to form a frame in SAP2000: (a) Initial IDs and (b) ID connections

If this approach of connecting two points is automated within a loop, multiple frames can be generated by providing an appropriate *ID* matrix as input. As the loop iterates through the matrix or matrices, it selects a pair of *ID*s and sends them to *SAP2000*, where the corresponding frames are created. This automated process enables the efficient generation of numerous frames without any manual intervention.

The first function required for the automated generation of frame by linking two selected *ID*s is defined in Eq.3.23:

$$f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}) \tag{3.23}$$

where f_{L} establishes connections between points in the longitudinal (L) direction. It does this by using the current (i) and next ($i+1$) elements from the jth row of the nth input matrix $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$. This approach ensures that features are not created between galleries where none should exist. As previously discussed, connections are only formed within regions between galleries.

The next function necessary for automated generation of frames in either transversal or vertical direction is given in Eq.3.24:

$$f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}, \boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}) \tag{3.24}$$

Here, f_{VT} creates connections between *ID*s in the vertical (V) or transversal (T) directions, depending on the input matrices. It does this by utilizing the current (i) *ID* from the jth row of the nth input matrix $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$ and the current (i) *ID* from the jth row of the mth input matrix $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$.

The third function, defined in Eq.3.25, is essential for creating *X-bracing* within the structure:

$$f_{\mathrm{XB}}(\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}, \boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}) \tag{3.25}$$

The function f_{XB} constructs *X-bracing* (*XB*) by connecting the current (*i*) *ID* from the *j*th row of the *n*th $M_{\mathrm{n}}^{\mathrm{ID}}$ to the next (i+1) *ID* from the *j*th row of the *m*th matrix $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$ within one loop. It then crates the other diagonal of *X-bracing* by connecting the current (*i*) *ID* from the *j*th row of the *m*th matrix $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$ to the next (i+1) *ID* from the *j*th row of the *n*th matrix $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$.

The fourth function, required for forming *vertical angle bracing* across the structure, is presented in Eq.3.26:

$$f_{\mathrm{VAB}}(\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}, \boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}) \tag{3.26}$$

The function f_{VAB} creates vertical angle bracing between the lower and upper flanges of the gallery. The final required design is illustrated in b) Figure 3.4. To adjust the angle at the middle of the *j*th gallery, a dimension *d* is calculated using the formula $d = \lfloor \frac{I+1}{2} \rfloor$, where *I* denotes the number of *ID*s in the *j*th gallery.

In the *k*th step of the iteration for the *j*th gallery, the function f_{VAB} creates the initial angle bracing by connectiong the (*k+1*) *ID* from $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$ with the (*k*) *ID* from $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$. For the opposite angle bracing, it connects the (*k+d*) *ID* from $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$ with the (*k+1+d*) *ID* from $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$. This process is repeated *K* times (*k = 0, ..., K*) for the *j*th gallery. Figure 3.4 illustrates the procedure: Figure 3.4a shows the first step of the iteration, while Figure 3.4b displays the final design achieved after *K* iterations for the *j*th gallery. This process is repeated for all *J* galleries stored in the columns of $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$ and $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$.

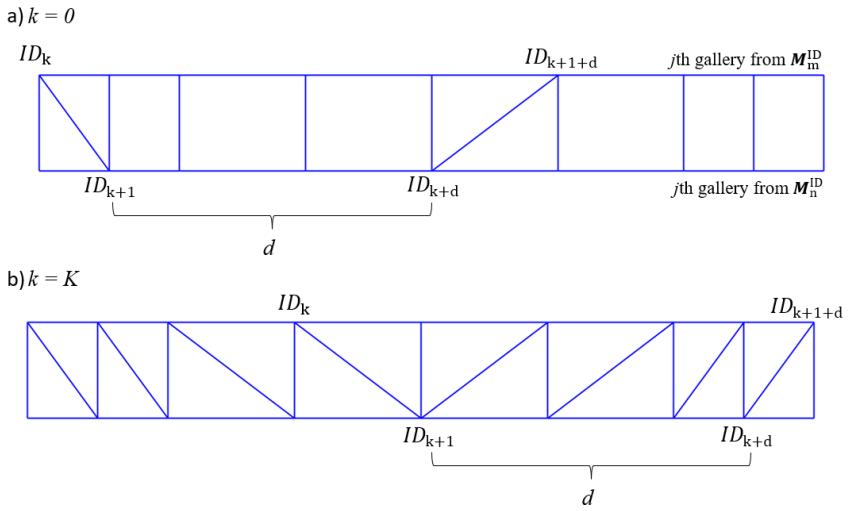


Figure 3.4. Vertical angle bracing process for galleries: (a) Initial iteration step and (b) Final design after K iterations

The next function, demonstrated in Eq.3.27, is required for creating *inverted V bracing* (*IVB*), which is primarily used for trestles:

$$f_{\mathrm{IVB}}\left(\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}, \boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}, \boldsymbol{M}_{\mathrm{p}}^{\mathrm{ID}}\right) \tag{3.27}$$

Here, the function f_{IVB} constructs required bracing by connecting the current (*i*) *ID* from the *j*th row of $\boldsymbol{M}_{\mathrm{p}}^{\mathrm{ID}}$ with the corresponding (*i+1*) *ID*s from the *j*th rows of both $\boldsymbol{M}_{\mathrm{n}}^{\mathrm{ID}}$ and $\boldsymbol{M}_{\mathrm{m}}^{\mathrm{ID}}$.

With all the necessary functions now defined, the next step is to detail how these functions are implemented to achieve the desired structural configurations.

Primary beams are constructed for the section of the gallery supporting the belt conveyors using $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{PBW}})$ and $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{NBW}})$. Additionally, $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{OP}})$ and $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{ON}})$ are employed to create the primary beams that define the edges of the gallery. To ensure structural stability, secondary beams are added to connect the outer parts of the gallery using $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{OP}}, \boldsymbol{M}_{\mathrm{ON}})$. *X-bracing* is created between the belt conveyor and edge of the gallery where walkway arrives, using $f_{\mathrm{XB}}(\boldsymbol{M}_{\mathrm{PBW}}, \boldsymbol{M}_{\mathrm{OP}})$ for the positive *x*-axis and $f_{\mathrm{XB}}(\boldsymbol{M}_{\mathrm{NBW}}, \boldsymbol{M}_{\mathrm{ON}})$ for the negative *x*-axis.

This concludes the design of the bottom part of the gallery. The next section focuses on the upper part of the gallery and the roof. Figure 3.5 illustrates the target design and the required perpendicular alignment of the upper points to the bottom points. It shows an A-A section of a single gallery, where $H_{\mathrm{g,int}}$ denotes the initial height of the gallery, E represents the height of the *raised heel,* and β is the roof angle. These values must be provided by the user via *Excel*.

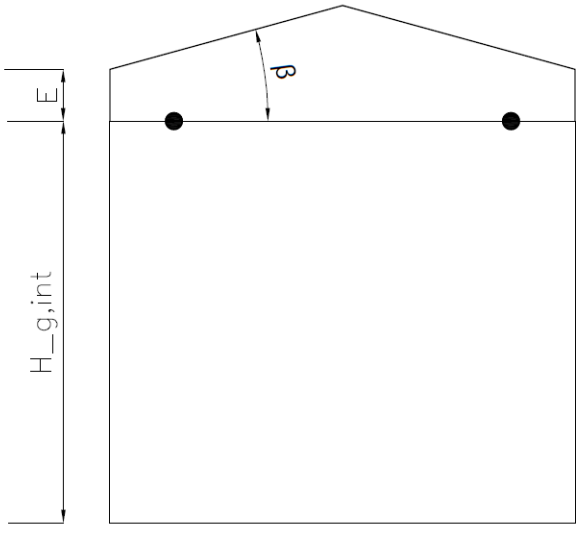


Figure 3.5. Representation of various height inputs

In the tool, inputs shown in Figure 3.5. are used to determine the distances of the required points from the reference *bottom* point:

$$H_{\mathrm{B}} = H_{\mathrm{g,int}} \tag{3.28}$$

$$H_{\mathrm{RHE}} = H_{\mathrm{g,int}} + E \tag{3.29}$$

$$H_{\mathrm{P}} = H_{\mathrm{g,int}} + E + \tan\beta \cdot (A_2 + B) \tag{3.30}$$

where, H_{B} represents the *base* gallery height, H_{RHE} represents the *raised heel end* point (*RHE*) and H_{P} denotes the roof's *peak* point (*P*).

Given that the calculation process is identical for all upper points, it can be streamlined using a function, as defined in Eq.3.31:

$$f_{\mathrm{UP}}(\boldsymbol{M}_{\mathrm{n}}, H_{\mathrm{n}}) \tag{3.31}$$

In this equation, f_{UP} represents the function that generates the *upper points* (*UP*). Here, $\boldsymbol{M}_{\mathrm{n}}$ denotes the *n*th input matrix, while H_{n} specifies the perpendicular distance from the upper points to the corresponding bottom points. The function operates on the previously established *n*th matrix, which consists of *J* rows, with each *j*th row representing a single gallery described by a total of *I* points. The output of f_{UP} is a new matrix, identical in shape to the original $\boldsymbol{M}_{\mathrm{n}}$, but containing only the calculated upper points, as shown in Eq.3.32.

$$\boldsymbol{M}_{\mathrm{n,up}} = \begin{bmatrix} \left(P_{\mathrm{i,up}}\right)_{\mathrm{j}} & \left(P_{\mathrm{i,up}}\right)_{\mathrm{j+1}} & \cdots & \left(IP_{\mathrm{i,up_i}}\right)_{\mathrm{J}} \\ \left(P_{\mathrm{i+1,up}}\right)_{\mathrm{j}} & \left(P_{\mathrm{i+1,up}}\right)_{\mathrm{j+1}} & \cdots & \left(P_{\mathrm{i+1,up}}\right) \\ \left(P_{\mathrm{i+2,up}}\right) & \left(P_{\mathrm{i+2,up}}\right)_{\mathrm{j+1}} & \cdots & \left(P_{\mathrm{i+2,up}}\right) \\ \vdots & \vdots & \vdots & \vdots \\ \left(P_{\mathrm{I,up}}\right)_{\mathrm{j}} & \left(P_{\mathrm{I,up}}\right)_{\mathrm{j+1}} & \cdots & \left(P_{\mathrm{I,up}}\right)_{\mathrm{J}} \end{bmatrix} \tag{3.32}$$

Where $\left(P_{\mathrm{i,up}}\right)_{\mathrm{j}}$ represents the *i*th upper point in the *j*th gallery, its coordinates are calculated using Eq.3.33:

$$\left(P_{\mathrm{i,up}}\right)_{\mathrm{j}} = (P_{\mathrm{i}})_{\mathrm{j}} + H_{\mathrm{n}} \cdot \overrightarrow{u_{\perp,\mathrm{i}}} \tag{3.33}$$

Here, $(P_{\mathrm{i}})_{\mathrm{j}}$ is the corresponding point in the matrix $\boldsymbol{M}_{\mathrm{n}}$, while $\overrightarrow{u_{\perp,\mathrm{i}}}$ represent the perpendicular unit vector required to describe the direction between $\left(P_{\mathrm{i,up}}\right)_{\mathrm{j}}$ and $(P_{\mathrm{i}})_{\mathrm{j}}$. The coordinates of the perpendicular unit vector $\overrightarrow{u_{\perp,\mathrm{i}}}$ are given in Eq.3.34-36:

$$x_{\perp,\mathrm{i}} = x_{\mathrm{U,i}} \tag{3.34}$$

$$y_{\perp,\mathrm{i}} = -z_{\mathrm{U,i}} \tag{3.35}$$

$$z_{\perp,\mathrm{i}} = y_{\mathrm{U,i}} \tag{3.36}$$

where $(x, y, z)_{\mathrm{U,I}}$ are the coordinates of the unit vector $\overrightarrow{u_{\mathrm{i}}}$ calculated from $(P_{\mathrm{i+1}})_{\mathrm{j}}$ to $(P_{\mathrm{i}})_{\mathrm{j}}$ using $\overrightarrow{u_{\mathrm{i}}} = \frac{\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}}}{\|\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}}\|}$, where $\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}}$ represent the direction vector calculated as $\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}} = (x_{\mathrm{i+1}} - x_{\mathrm{i}}, y_{\mathrm{i+1}} - y_{\mathrm{i}}, z_{\mathrm{i+1}} - z_{\mathrm{i}})_{\mathrm{J}}$, while $\|\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}}\|$ represent normalized vector calculated as $\left\|\overrightarrow{(P_{\mathrm{i+1}}P_{\mathrm{i}})_{\mathrm{J}}}\right\| = \sqrt{(x_{\mathrm{i+1}} - x_{\mathrm{i}})_{\mathrm{J}}^2 + (y_{\mathrm{i+1}} - y_{\mathrm{i}})_{\mathrm{J}}^2 + (z_{\mathrm{i+1}} - z_{\mathrm{i}})_{\mathrm{J}}^2}$.

By using $f_{\mathrm{UP}}(\boldsymbol{M}_{\mathrm{n}}, H_{\mathrm{n}})$, multiple points are created. For instance, the upper part of the gallery is defined with $\boldsymbol{M}_{\mathrm{OUP}} = f_{\mathrm{UP}}(\boldsymbol{M}_{\mathrm{OP}}, H_{\mathrm{g,int}})$ for the *outer upper positive* (*OUP*) points and $\boldsymbol{M}_{\mathrm{OUN}} = f_{\mathrm{UP}}(\boldsymbol{M}_{\mathrm{ON}}, H_{\mathrm{g,int}})$ for the *outer upper negative* (*OUN*) points. In this process, $\boldsymbol{M}_{\mathrm{OUP}}$ is derived by applying f_{UP} to the initial lower set of points stored in $\boldsymbol{M}_{\mathrm{OP}}$, using $H_{\mathrm{g,int}}$ to scale the perpendicular unit vectors $\overrightarrow{u_{\perp,\mathrm{i}}}$. The *raised heel* points are characterized by the dimensions *E*, corresponding to the total height H_{RHE}. Using $f_{\mathrm{UP}}(\boldsymbol{M}_{\mathrm{OP}}, H_{\mathrm{RHE}})$, matrices $\boldsymbol{M}_{\mathrm{PRHE}}$ and $\boldsymbol{M}_{\mathrm{NRHE}}$ are formed, representing the *positive* and *negative raised heel end* points, respectively. The final upper points are required to define the roof's peak, where $\boldsymbol{M}_{\mathrm{P}}$ represents the roof's peak points and is formed by using the $f_{\mathrm{UP}}(\boldsymbol{M}_{0}, H_{\mathrm{P}})$. Coordinates for all points are then created in *SAP2000* using the function f_{ID}, and retrieved back to *Python* as $\boldsymbol{M}_{\mathrm{OUP}}^{\mathrm{ID}}$, $\boldsymbol{M}_{\mathrm{OUP}}^{\mathrm{ID}}$, $\boldsymbol{M}_{\mathrm{PRHE}}^{\mathrm{ID}}$, $\boldsymbol{M}_{\mathrm{NRHE}}^{\mathrm{ID}}$, $\boldsymbol{M}_{\mathrm{P}}^{\mathrm{ID}}$.

Once the points are defined, they are used to create the necessary gallery features. Columns are defined by $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{OP}}, \boldsymbol{M}_{\mathrm{OUP}})$ and $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{ON}}, \boldsymbol{M}_{\mathrm{OUN}})$. *Vertical angle bracings* are defined by $f_{\mathrm{VAB}}(\boldsymbol{M}_{\mathrm{OP}}, \boldsymbol{M}_{\mathrm{OUP}})$ and $f_{\mathrm{VAB}}(\boldsymbol{M}_{\mathrm{ON}}, \boldsymbol{M}_{OUN})$. Longitudinal beams at the height H_B are defined by $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{OUP}})$ and $f_{\mathrm{L}}(\boldsymbol{M}_{\mathrm{OUN}})$, with additional *X-bracing* at the same height created using $f_{\mathrm{XB}}(\boldsymbol{M}_{\mathrm{OUP}}, \boldsymbol{M}_{\mathrm{OUN}})$. The *energy heel* is created using $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{OUP}}, \boldsymbol{M}_{\mathrm{PRHE}})$ and $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{OUN}}, \boldsymbol{M}_{\mathrm{NRHE}})$. Lastly, the roof rafters are defined by $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{PRHE}}, \boldsymbol{M}_{\mathrm{P}})$ and $f_{\mathrm{VT}}(\boldsymbol{M}_{\mathrm{NRHE}}, \boldsymbol{M}_{\mathrm{P}})$.

To fully define the roof structure, *purlins* need to be created by calculating the points necessary for their definition. Each *i*th roof section consists of three distinct *purlin* points on each side of the x-axis, making a total of six *purlin* points per section, as shown in Figure 3.6.

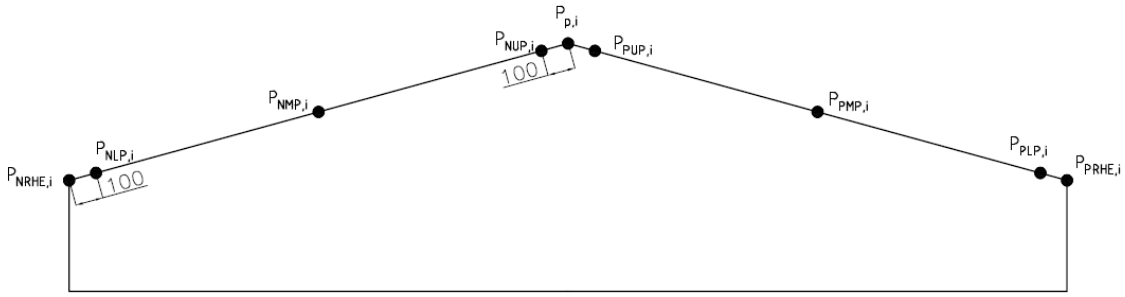


Figure 3.6. Peak and purlin points

The points $P_{\mathrm{NRHE,i}}$, $P_{\mathrm{P,i}}$ and $P_{\mathrm{PRHE,i}}$ are the *i*th points from the *j*th gallery, stored in the previously defined matrices $\boldsymbol{M}_{\mathrm{NRHE,i}}$, $\boldsymbol{M}_{\mathrm{P,i}}$ and $\boldsymbol{M}_{\mathrm{PRHE,i}}$. The remaining points shown in the figure need to be calculated. First, the *i*th negative *lower purlin* point $P_{\mathrm{NLP,i}}$ is located 0.1 *m* from $P_{\mathrm{NRHE,i}}$ along the line connecting $P_{\mathrm{NRHE,i}}$ and $P_{\mathrm{P,i}}$, with its coordinates determined by $P_{\mathrm{NLP,i}} = P_{\mathrm{NRHE,i}} + 0.1 \cdot \frac{\overrightarrow{P_{\mathrm{NRHE,1}}P_{\mathrm{P,1}}}}{\|\overrightarrow{P_{\mathrm{NRHE,1}}P_{\mathrm{P,1}}}\|}$. Next, *ith negative middle purlin* point $P_{\mathrm{NMP,i}}$ is found by averaging the coordinates of $P_{\mathrm{NRHE,i}}$ and $P_{\mathrm{P,i}}$ using $P_{\mathrm{NMP,i}} = \frac{1}{2}\left(x_{\mathrm{NRHE,i}} + x_{\mathrm{P,i}}, y_{\mathrm{NRHE,i}} + y_{\mathrm{P,i}}, z_{\mathrm{NRHE,i}} + z_{\mathrm{P,i}}\right)$. The final point, $P_{\mathrm{NUP,i}}$, is located 0.1 *m* from $P_{\mathrm{P,i}}$ along the line connecting $P_{\mathrm{P,i}}$ and $P_{\mathrm{NRHE,i}}$. Its coordinates are calculated similarly to $P_{\mathrm{NLP,i}}$, where $P_{\mathrm{NUP,i}}$ is now equal to $P_{\mathrm{P,i}} + 0.1 \cdot \frac{\overrightarrow{P_{\mathrm{p,1}}P_{\mathrm{NRHE,1}}}}{\|\overrightarrow{P_{\mathrm{p,1}}P_{\mathrm{NRHE,1}}}\|}$. The same calculations are repeated for the positive side, yielding the coordinates for the *positive lower purlin* point $P_{\mathrm{PLP,i}}$, *positive middle purlin* point $P_{\mathrm{PMP,i}}$ and *positive upper purlin* point $P_{\mathrm{PUP,i}}$. The points are stored in the corresponding matrices $\boldsymbol{M}_{\mathrm{NLP,i}}$, $\boldsymbol{M}_{\mathrm{NMP,i}}$, $\boldsymbol{M}_{\mathrm{NUP,i}}$, $\boldsymbol{M}_{\mathrm{PLP,i}}$, $\boldsymbol{M}_{\mathrm{PMP,i}}$, $\boldsymbol{M}_{\mathrm{PUP,i}}$ which are then sent to *SAP2000* using f_{ID} and retrieved back into *Python* as $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NLP,i}}$, $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NMP,i}}$, $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NUP,i}}$, $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PLP,i}}$, $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PMP,i}}$, $\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PUP,i}}$. The *purlins* are formed using $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NLP,i}}\right)$, $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NMP,i}}\right)$, $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NUP,i}}\right)$, $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PLP,i}}\right)$, $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PMP,i}}\right)$ and $f_{\mathrm{L}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PUP,i}}\right)$. The *X*-bracing for the roof is created using $f_{\mathrm{XB}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NLP,i}}, \boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NMP,i}}\right)$, $f_{\mathrm{XB}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NMP,i}}, \boldsymbol{M}^{\mathrm{ID}}_{\mathrm{NUP,i}}\right)$, $f_{\mathrm{XB}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PLP,i}}, \boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PMP,i}}\right)$ and $f_{\mathrm{XB}}\left(\boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PMP,i}}, \boldsymbol{M}^{\mathrm{ID}}_{\mathrm{PUP,i}}\right)$.

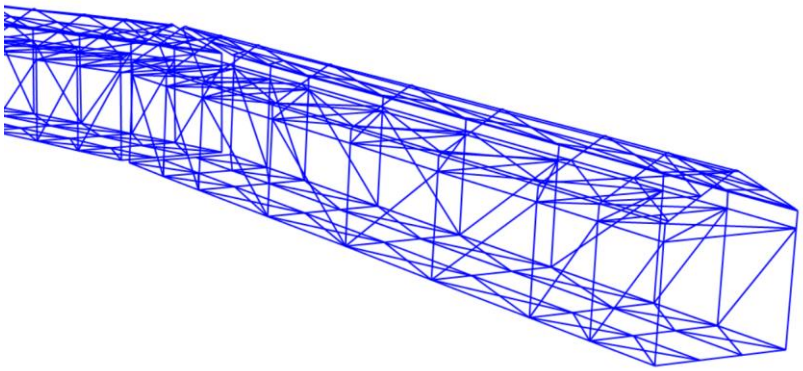


Figure 3.7. Defined structure of single gallery

This concludes the creation of the gallery section of the structure (Figure 3.7). The next step involves modeling the trestles that support the galleries at defined heights. Trestles are essential structural elements that provide vertical and lateral support to the galleries, ensuring stability and effective load distribution across the entire structure.

The structure is built from a total of *J* galleries, with trestles positioned between each pair of adjacent galleries. There are two main types of trestles: the *pinned support trestle* and the *fixed point trestle.*

The points for the *pinned support trestle* are symmetrical, enabling the calculations to be carried out only for the *positive* *x*-axis. Once calculated, the *x-coordinate* can be adjusted—either replaced by zero for *middle* points or multiplied by -1 for points on the *negative* x-axis.

The *start-end* points for the *j*th *pinned support trestle* are initialized from the corresponding first point stored in the *j*th column of $\boldsymbol{M}_{\mathrm{OP,TR}}$, excluding the first and last column of $\boldsymbol{M}_{\mathrm{OP}}$. These points are calculated as follows:

$$(P_{\mathrm{PSTS}})_{\mathrm{j}} = (x_{\mathrm{i}}, y_{\mathrm{i}}, z_{\mathrm{i}} - 0.5)_{\mathrm{j}} \tag{3.37}$$

$$(P_{\mathrm{PSTE}})_{\mathrm{j}} = (x_{\mathrm{i}}, y_{\mathrm{i}}, 0.7)_{\mathrm{j}} \tag{3.38}$$

Here, $(P_{\mathrm{PSTS}})_{\mathrm{j}}$ represent the *pinned support trestle start* point, located 0.5 *m* below the *i*th point of the *j*th column of $\boldsymbol{M}_{\mathrm{OP,TR}}$. The point $(P_{\mathrm{PSTE}})_{\mathrm{j}}$ represents the *pinned support trestle end* point, positioned 0.7 *m* away from the ground. These distances are chosen due to the connection details that will be modeled later.

The calculated points are used to determine the available height of trestle, once the required connection is excluded from the available space:

$$H_{\mathrm{A}} = z_{\mathrm{PSTS,j}} - z_{\mathrm{PSTS,j}} \tag{3.39}$$

where H_{A} represents the available height, calculated as a difference between the *z*-coordinates of the *start-end* points. On the other hand, H_{MA} represents the *maximum allowed* height between two secondary beams within a single trestle. The ratio between two values (Eq.3.40) determines the number of height segments H_{MA} that can be used within a single gallery trestle:

$$n_{\mathrm{MA}} = \left\lfloor \frac{H_{\mathrm{A}}}{H_{\mathrm{MA}}} \right\rfloor - 1 \tag{3.40}$$

where n_{MA} represents the number of H_{MA} height segments between adjecant secondary beams within a single trestle. The value of n_{MA} also determines the subtype of the *pinned support trestle,* represented by the *n*th type *T*, as shown below:

$$\begin{cases} T_1,\ n_{\mathrm{MA}} > 0\ and\ H_{\mathrm{A}} < 15\ m \\ T_2,\ n_{\mathrm{MA}} > 0\ and\ H_{\mathrm{A}} > 15\ m \\ T_3,\ n_{\mathrm{MA}} = 0 \\ T_4,\ n_{\mathrm{MA}} < 0 \end{cases} \tag{3.41}$$

For a T_1 *pinned support trestle*, the points for the *trestle secondary beams* are placed offset from the previous point, as presented in Eq.3.42:

$$\left(P_{\mathrm{TSB,i}}\right)_{\mathrm{j}} = (x_{\mathrm{i-1}}, y_{\mathrm{i-1}}, z_{\mathrm{i-1}} - H_{\mathrm{MA}} \cdot n_{\mathrm{MA}})_{\mathrm{j}} \tag{3.42}$$

Once $\left(P_{\mathrm{TSB,I}}\right)_{\mathrm{j}}$ is calculated, the remaining distance H_{R} between the *trestle secondary beams* $\left(P_{\mathrm{TSB,I}}\right)_{\mathrm{j}}$ and trestle *end* point $\left(P_{\mathrm{TSB,I}}\right)_{\mathrm{j}}$ is measured. Half of this distance is used to calculate the last *trestle secondary beams* point $\left(P_{\mathrm{TSB,I+1}}\right)_{\mathrm{j}}$ coordinates.

The calculated points are stored in the *corresponding* matrix representing representing the *j*th T_1 *pinned support trestle*.

$$\left(\boldsymbol{M}_{\mathrm{T1,P}}\right)_{\mathrm{j}} = \begin{bmatrix} (P_{\mathrm{PSTS}})_{\mathrm{j}} \\ \left(P_{\mathrm{TSB,i}}\right)_{\mathrm{j}} \\ \cdots \\ \left(P_{\mathrm{TSB,I}}\right)_{\mathrm{j}} \\ \left(P_{\mathrm{TSB,I+1}}\right)_{\mathrm{j}} \\ (P_{\mathrm{PSTE}})_{\mathrm{j}} \end{bmatrix} \tag{3.43}$$

As previously mentioned, the calculations were performed just for the positive side of the *x-axis*. Once completed, the *x-coordinate* of each point are modified. For the *middle* points, the *x-coordinate* is set to zero, forming the matrix $\left(\boldsymbol{M}_{\mathrm{T1,M}}\right)_{\mathrm{j}}$. For the *negative* *x*-axis points, the *x-*

coordinate is multiplied by -1, creating the matrix $\left(\boldsymbol{M}_{\mathrm{T1,N}}\right)_{\mathrm{j}}$. Using the function f_{ID}, the points from these matrices are sent to *SAP2000* and retrieved back into *Python* as corresponding $\left(\boldsymbol{M}_{\mathrm{T1,P}}^{\mathrm{ID}}\right)_{\mathrm{j}}$, $\left(\boldsymbol{M}_{\mathrm{T1,m}}^{\mathrm{ID}}\right)_{\mathrm{j}}$, $\left(\boldsymbol{M}_{\mathrm{T1,N}}^{\mathrm{ID}}\right)_{\mathrm{j}}$. The trestle's columns are then formed using $f_{\mathrm{L}}\left(\left(\boldsymbol{M}_{\mathrm{T1,P}}^{\mathrm{ID}}\right)_{\mathrm{j}}\right)$, utilizing a point selection pattern from f_{L}, even tough the columns may not necessarily be formed in *longitudinal directions*. Secondary beams are formed using $f_{\mathrm{VT}}\left(\left(\boldsymbol{M}_{\mathrm{T1,P}}^{\mathrm{ID}}\right)_{\mathrm{j}}, \left(\boldsymbol{M}_{\mathrm{T1,N}}^{\mathrm{ID}}\right)_{\mathrm{j}}\right)$, and the *inverted vertical bracing* is constructed with $f_{\mathrm{IVB}}\left(\left(\boldsymbol{M}_{\mathrm{T1,P}}^{\mathrm{ID}}\right)_{\mathrm{j}}, \left(\boldsymbol{M}_{\mathrm{T1,M}}^{\mathrm{ID}}\right)_{\mathrm{j}}, \left(\boldsymbol{M}_{\mathrm{T1,N}}^{\mathrm{ID}}\right)_{\mathrm{j}}\right)$.

For a T_2 type trestle, the columns are designed at an angle α_{TR}. In order to achive the required design, for every *positive* and *negative* point, except the *starting* point, the value $x_{\alpha} = \tan(\alpha_{\mathrm{TR}}) \cdot H_{\mathrm{i}}$ is added to the existing *x*-coordinate, where H_{i} represents the height from the current point (*i*) to the previous point (*i*-1). In the case of T_3 type trestles, a single secondary beam point $\left(P_{\mathrm{TSB,i}}\right)_{\mathrm{j}}$ is located halfway between $(P_{\mathrm{PSTS}})_{\mathrm{j}}$ and $(P_{\mathrm{PSTE}})_{\mathrm{j}}$. For T_4 type trestles, there are no secondary beam points $\left(P_{\mathrm{TSB,i}}\right)_{\mathrm{j}}$, and the features are created using only $(P_{\mathrm{PSTS}})_{\mathrm{j}}$ and $(P_{\mathrm{PSTE}})_{\mathrm{j}}$. For each trestle type, the process of creating matrices, initializing points in *SAP2000*, retrieving *ID*s, and connecting them to form structural features follows the same procedure as for T1.

The *start-end* points of the *fixed point trestle* are calculated similarly, using the previously demonstrated approach for calculating point coordinates when an angle is defined. In this case, the point P_{LRTS} is located 0.5 *m* away from the first point of the *J*th column of $\boldsymbol{M}_{\mathrm{OP}}$, while the *end* points P_{LRTE1} and P_{LRTE2} are positioned 0.7 *m* above the ground at a closing angle of θ_{TR}. These points are connected to form columns, and the *end* points are connected to form the secondary beams. The same initialization process is repeated for the opposite side, with an X-bracing placed between the two outer frames.

3.2. Connections and restraints

Connections and restraints are crucial to ensure the stability, proper load distribution, and overall structural integrity of the belt conveyor system.

The first connection that needs to be modeled creates a link between galleries and technological towers, using *sliding* and *shear key* connections. The *sliding* connection is initialized by defining six key points: two starting points from previously created connections, and four offset

points derived by modifying the *z*-coordinates. The starting points were previously created in *SAP2000* and are located in the first rows of the first columns of $\boldsymbol{M}_{\mathrm{OP}}^{\mathrm{ID}}$ for the *positive sliding connection point* (P_{PSCP}) and $\boldsymbol{M}_{\mathrm{ON}}^{\mathrm{ID}}$ for *negative sliding connection points* (P_{NSCP}). The coordinates for these points are saved correspondingly in $\boldsymbol{M}_{\mathrm{OP}}$ as ID_{PSCP} and in $\boldsymbol{M}_{\mathrm{ON}}$ as ID_{NSCP}. Next, four offset points are generated by reducing the *z*-coordinate of P_{PSCP} and P_{NSCP} by 0.25 m, resulting in $P_{\mathrm{PSCP,Z1}}$ and $P_{\mathrm{NSCP,Z1}}$, and then by 0.5 m, creating $P_{\mathrm{PSCP,Z2}}$ and $P_{\mathrm{NSCP,Z2}}$. These points are then created in *SAP2000*, with their *IDs* retrieved as $ID_{\mathrm{PSCP,Z1}}$, $ID_{\mathrm{NSCP,Z1}}$, $ID_{\mathrm{PSCP,Z2}}$ and $ID_{\mathrm{NSCP,Z2}}$. Frames are then connected, forming the required connection with the modified *rigid* frames, which have zero mass and weight, but increased cross-section and shear area compared to standard frames. Finally, the restraint blocking translation in *z* direction is applied at $ID_{\mathrm{PSCP,Z2}}$ and $ID_{\mathrm{NSCP,Z2}}$ to control the movement the sliding connection.

Next, the *shear key* connection between the galleries and technological towers needs to be modeled. First, the midpoint between P_{PSCP} and P_{NSCP} is determined and calculated as $P_{\mathrm{MSKCP}} = \frac{P_{\mathrm{PSCP}}+P_{\mathrm{NSCP}}}{2}$. This midpoint is then used as a reference for creating the *shear key* offset point $P_{\mathrm{MSKCP,Z2}}$, which is obtained by subtracting 0.5 m from the *z*-coordinate of P_{MSKCP}. Both points, P_{MSKCP} and $P_{\mathrm{MSKCP,Z2}}$, are created in *SAP2000,* with their *IDs* retrieved as ID_{MSKCP} and $ID_{\mathrm{MSKCP,Z2}}$. These points are then connected, and finally, a restrain is applied to $ID_{\mathrm{MSKCP,Z2}}$, disabling translation in the *x* and *z* directions, to fully model the required *shear key* connection.

Instanced process is then repeated for the end of structure using the last points stored in the last columns of $\boldsymbol{M}_{\mathrm{OP}}^{\mathrm{ID}}$ and $\boldsymbol{M}_{\mathrm{ON}}^{\mathrm{ID}}$, with coordinates stored in $\boldsymbol{M}_{\mathrm{OP}}$ and $\boldsymbol{M}_{\mathrm{ON}}$.

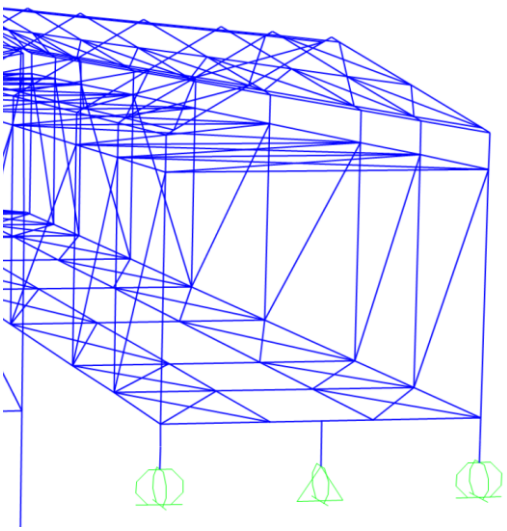


Figure 3.8. Defined constraints at the start of the structure

The second required connection is established between two adjacent galleries and consists of both *pinned base plate* and *shear key* connections. First, the *pinned base plate* connection is modeled by initializing points originally stored in $\boldsymbol{V}_{\mathrm{CP}}$ with added x-coordinates: $(A_2 + B)$ for $\boldsymbol{V}_{\mathrm{CPP}}$ and $-(A_2 + B)$ for $\boldsymbol{V}_{\mathrm{CPN}}$. These points are then created in *SAP2000* and retrieved with their *IDs* as $\boldsymbol{V}_{\mathrm{CPP}}^{\mathrm{ID}}$ and $\boldsymbol{V}_{\mathrm{CPN}}^{\mathrm{ID}}$. To establish the *pinned base plate* connection for the nth connection between the current (j) and the next ($j+1$) gallery, the last *ID* from the current gallery is connected to the ith *ID* from $\boldsymbol{V}_{\mathrm{CPP}}^{\mathrm{ID}}$, and the ith *ID* from $\boldsymbol{V}_{\mathrm{CPP}}^{\mathrm{ID}}$ is then connected to the first *ID* of the next gallery. For connections involving positive x-coordinates the instanced gallery is $\boldsymbol{M}_{\mathrm{OP}}^{\mathrm{ID}}$, and for negative x-coordinates, $\boldsymbol{M}_{\mathrm{ON}}^{\mathrm{ID}}$ and $\boldsymbol{V}_{\mathrm{CPN}}^{\mathrm{ID}}$ are used. All frames are connected using *rigid* elements. To create the pinned connection, an end release is assigned to the first frame in the direction of the material flow.

The *shear key* connection is established by first initializing the required points using the previously defined *unit vector* approach. These points are used to create frames that represent the *shear key*. To accurately model the *shear key's* properties, a transversal link is added between the frames, which restricts movement in specific directions (e.g., translation in certain axes). This setup effectively simulates the behavior of the *shear key* within the structure, ensuring it can transfer shear forces while limiting unwanted translations.

To create connections between trestles and galleries, the same approach and connection method used for adjacent galleries is applied here.

The final connection between adjacent galleries and between galleries and trestles is illustrated in Figure 3.9.

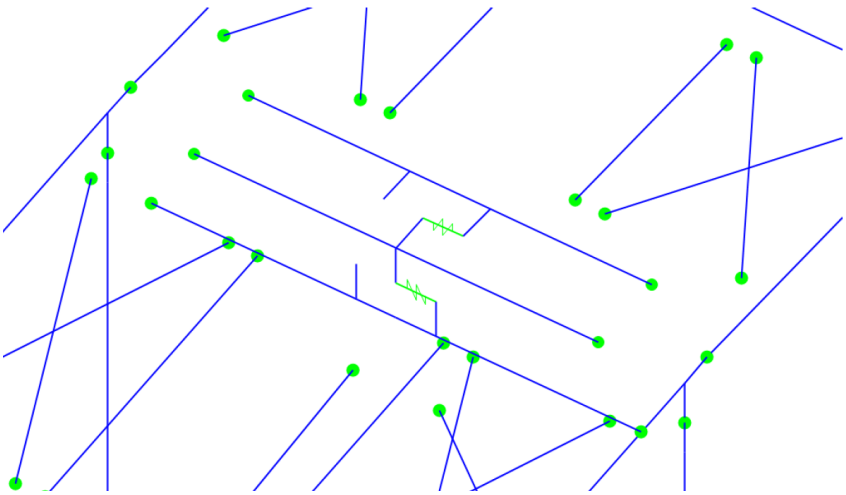


Figure 3.9. Defined connection in-between adjacent galleries

The final required connections for the structure are those between the ends of the trestles and the ground. According to the rules specified in Eq.3.41, the last point of each trestle is used as the initial point. A corresponding ground point is then determined by keeping the same coordinates but setting the *z*-coordinate to zero, representing the ground level. These points are connected with *rigid* frames, and a *pinned* support is applied at the end of the frames to anchor the structure to the ground.

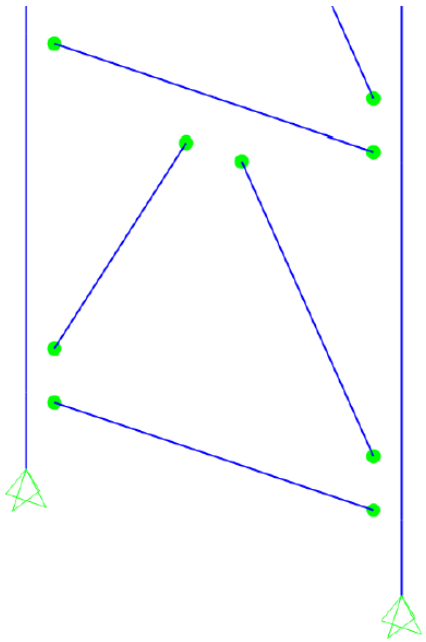


Figure 3.10. Trestle to ground connection with restraints

3.3. Actions

Structures are subjected to several types of actions, which vary in magnitude and can occur in different combinations. EN 1991 defines different actions that must be considered when designing a structure. These defined actions are then combined with corresponding partial factors, depending on whether the checks are done due to safety or serviceability of the structure.

An action refers to anything that could induce stress or deformation in a structure. This includes forces, moments, temperature variations, and other influences that impact the integrity of the structure. [6]

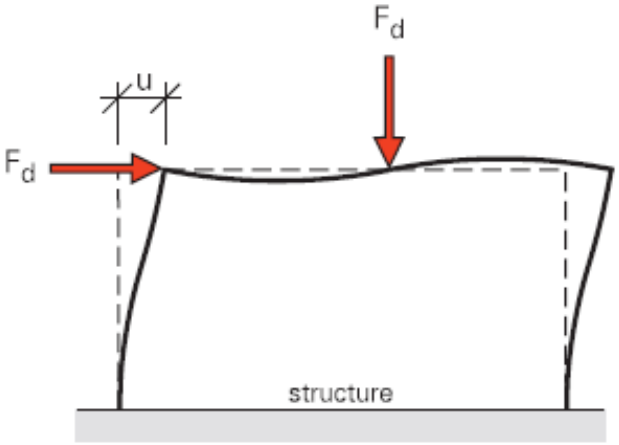


Figure 3.11. Definition of action [6]

A single action within a combination of actions is defined as:

$$F_{\mathrm{d}} = \gamma_{\mathrm{f}} \cdot F_{\mathrm{rep}} \tag{3.44}$$

where the γ_{f} represents the partial factor for actions, accounting for the possibility of unfavourable deviations from the representive values, and F_{rep} represents the value of the action calculated as:

$$F_{\mathrm{rep}} = \psi \cdot F_{\mathrm{k}} \tag{3.45}$$

Here, F_{k} is the characteristic value of the action, and ψ is the combination factor selected from tables based on the action type, the definition of combination and other relevant factors. Substituting F_{rep} back into Eq.3.44 above, the design value F_{d} becomes:

$$F_{\mathrm{d}} = \gamma_{\mathrm{f}} \cdot \psi \cdot F_{\mathrm{k}} \tag{3.46}$$

The use of partial factors for actions γ_{f} encposes the possibility of unfavourable devations in the action values and potential uncertentines in modeling the effect of these actions.

EN 1991 specifies the characteristic values of actions across different parts of the code, classifying them into different types: permanent load (G), variable action (Q), accidental actions (A).

3.3.1. Permanent actions

Permanent actions are those with limited variation in magnitude over time. These actions mainly involve the self-weight of structural and non-structural elements within the structure. In this case, the self-weight of the structure is automatically calculated by the *SAP2000*. To

account for additional segments of the structure like plates and bolts, a contingency of 20% has been considered.

Additionally, the self-weight of non-structural elements is taken into account. In this context, the non-structural elements include the cladding on lateral walls, which refers to the corrugated metal sheets used as external wall coverings; louvers and girts; roofing, where the self-weight of roof panels forms a significant part of the load; and the grating on truss beams, which consists of metal grids used as flooring on the bottom of the truss beams.

In addition to these structural and non-structural elements, permanent imposed loads also arise from the equipment housed within the structure. For this specific structure, these loads are due to the belt conveyors and associated components, including the belt itself, the belt frame, pulleys, rollers, motors, gearboxes, and piping. The combined weight of these components is applied as a line load distributed along the galleries supporting the belt conveyor system. This line load represents the continuous action exerted by the equipment on the structure.

3.3.2. Variable actions

Unlike permanent actions, variable actions change over time. These actions are defined in various parts of EN 1991.

Variable imposed loads generally arise from the usage of the building and can be classified as uniformly distributed loads q_{k} [kN/m^2], line loads q_{k} [kN/m] or concentrated loads Q_{k} [kN].

For instance, personnel standing on the walkways between belt conveyors and the edge of the gallery impose a variable uniformly distributed load on the walkways. Additionally, the material being conveyed on the belt conveyors exerts a variable imposed line load on the stringers of the gallery.

Moreover, the belt conveyor system within the gallery generates belt forces during operation and start-up conditions. These forces, calculated for both operational and start-up scenarios, are applied to the gallery as distributed forces.

Other variable actions include dust and snow loads. Dust load D_{s} represents the amount of dust that settles on surfaces as a consequence of material being conveyed. It is calculated as:

$$D_{\mathrm{s}} = \rho \cdot d_{\mathrm{s}} \tag{3.47}$$

where ρ is the bulk density, and d_{s} is the thickness of the dust layer.

Snow loads, on the other hand, represent the weight of snow on flat or inclined surfaces and are calculated according to EN 1991-1-3 for persistent and transient design situations. The snow load s is determined using the formula:

$$s = \mu_{\mathrm{i}} \cdot C_{\mathrm{e}} \cdot C_{\mathrm{t}} \cdot s_{\mathrm{k}} \tag{3.48}$$

where μ_i is the snow load shape coefficient based on the roof's geometry (mono-pitch, duo-pitch, or multi-span), C_{e} is the exposure coefficient (typically taken as 1 unless otherwise specified for different topographies), C_{t} is the thermal coefficient accounting for the snow load reduction on roofs with high thermal transmittance, and s_{k} is the characteristic value for snow load on the ground as specified in the *National Annexes*.

According to EN 1991-1-5, thermal loads caused by climatic or operational temperature changes are calculated by considering uniform temperature components ΔT_{u}, which represent the difference between the average temperature of an element T and its initial temperature T_0 for different season. For the summer conditions, the uniform temperature differencial is calculated as:

$$\Delta T_{\mathrm{u,summer}} = T_{\mathrm{summer}} - T_0 \tag{3.49}$$

where T_{summer} is calculated as $T_{\mathrm{summer}} = \frac{1}{2} \cdot \left(T_1 + T_{\mathrm{out,summer}}\right)$. Here, T_1 is the average internal temperature within the structure, and $T_{\mathrm{out,summer}}$ is the outside temperature during summer, calculates as $T_{\mathrm{out,summer}} = T_{\mathrm{max}} + T_4$ where T_{max} is the maximum air temperature in the shade, and T_4 is the temperature defined in *National Annexes* based on the surface color.

For winter conditions, the uniform temperature differential is calculated as:

$$\Delta T_{\mathrm{u,winter}} = T_{\mathrm{winter}} - T_0 \tag{3.50}$$

where T_{winter} is calculated as $T_{\mathrm{winter}} = \frac{1}{2} \cdot \left(T_2 + T_{\mathrm{out,winter}}\right)$. In this case, T_2 represents the average internal temperature within the structure during winter, and $T_{\mathrm{out,winter}}$ corresponds to the minimum shade air temperature T_{min}.

In addition to thermal loads, wind loads are defined according to EN 1991-1-4. Wind is characterized as moving air with a density of 1.25 kg/m^3 and variable velocity and direction. To model wind loads, EN 1991-1-4 specifies that the analysis should be based on the basic wind speed $v_{\mathrm{b},0}$, which is the average wind velocity that is statistically exceeded once every fifty years. This value represents the characteristic 10-minute mean wind velocity measured at

a height of 10 meters above ground level in an open area, as specified in the *National Annexes* of different countries.

Wind is influenced by the ground, with its effects varying based on the roughness of the terrain and the height above ground. To account for this, the mean wind velocity $v_{\mathrm{m}}(z)$ is defined as a function of height z with influence of terrain:

$$v_{\mathrm{m}}(z) = c_{\mathrm{r}}(z) \cdot c_0(z) \cdot v_{\mathrm{b}} \tag{3.51}$$

Here, v_{b} represents the basic wind velocity at a height of 10 m, calculated using Eq.3.52:

$$v_{\mathrm{b}} = v_{\mathrm{b},0} \cdot c_{\mathrm{dir}} \cdot c_{\mathrm{season}} \tag{3.52}$$

In this equation, $v_{\mathrm{b},0}$ is the basic wind speed, c_{dir} is the directional factor, and c_{season} is the seasonal factor. The roughness factor $c_{\mathrm{r}}(z)$ from Eq.3.53, which accounts for terrain, is determined by the following rule:

$$c_{\mathrm{r}}(z) = \begin{cases} k_{\mathrm{r}} \cdot ln\left(\frac{z}{z_0}\right), & z_{\min} < z < z_{\max} \\ c_{\mathrm{r}}(z_{\min}), & z < z_{\min} \end{cases} \tag{3.53}$$

In this formula, z_0 is the roughness length, and $z_{\min}$ is the minimum height, both of which are selected as tabular data. $z_{\max} = 200\ m$ is the maximum height, which is also the calibrated height according to EN 1991-1-4. The terrain factor k_{r} from Eq.3.54 is calculated as:

$$k_{\mathrm{r}} = 0.19 \left(\frac{z_0}{0.05}\right)^{0.07} \tag{3.54}$$

The orography factor $c_0(z)$ from Eq.3.51 accounts for the influence of hills and cliffs. If the average slope is less then 3°, the effects of orography are typically negligible and can be neglected. For slopes above 3°, the determination of the orography factor requires a specific procedure, which may be specified in the *National Annexes*.

With the basic wind velocity fully defined the basic wind pressure q_{b} is calculated using the air density ρ:

$$q_{\mathrm{b}} = \frac{1}{2} \rho \cdot v_{\mathrm{b}}^2 \tag{3.55}$$

The peak velocity pressure $q_{\mathrm{p}}(z)$ at height z is then calculated as:

$$q_{\mathrm{p}}(z) = \left(1 + 7 \cdot l_{\mathrm{v}}(z)\right) \cdot \frac{1}{2} \cdot \rho \cdot v_{\mathrm{m}}^2(z) = c_{\mathrm{e}}(z) \cdot q_{\mathrm{b}} \tag{3.56}$$

Here, $l_{\mathrm{v}}(z)$ represents the turbulence intensity at height z, determined by the rule:

$$l_{\mathrm{v}}(z) = \begin{cases} \dfrac{\sigma_{\mathrm{v}}}{v_{\mathrm{m}}(z)} = \dfrac{k_{\mathrm{l}}}{c_{\mathrm{o}}(z)\cdot \ln\left(\frac{z}{z_0}\right)}, & z_{\min} \le z \le z_{\max} \\ l_{\mathrm{v}}(z_{\min}), & z \le z_{\min} \end{cases} \tag{3.57}$$

In the context of Eq.3.58, σ_{v} is the standard deviation of the turbulent component of wind velocity, calculated as:

$$\sigma_{\mathrm{v}} = k_{\mathrm{r}} \cdot v_{\mathrm{b}} \cdot k_{\mathrm{l}} \tag{3.58}$$

where the k_{l} represents the turbulence factor used in both the turbulence intensity (Eq.3.57) and standard deviation calculations (Eq.3.58).

With both peak and basic velocity pressure calculated, their ratio can be used to express the exposure factor $c_{\mathrm{e}}(z)$ as:

$$c_{\mathrm{e}}(z) = \frac{q_{\mathrm{p}}(z)}{q_b} \tag{3.59}$$

This factor helps in determining the effect of wind exposure on the structure. The final wind pressure applied to the structure must account for both external and internal pressures, using corresponding pressure coefficients.

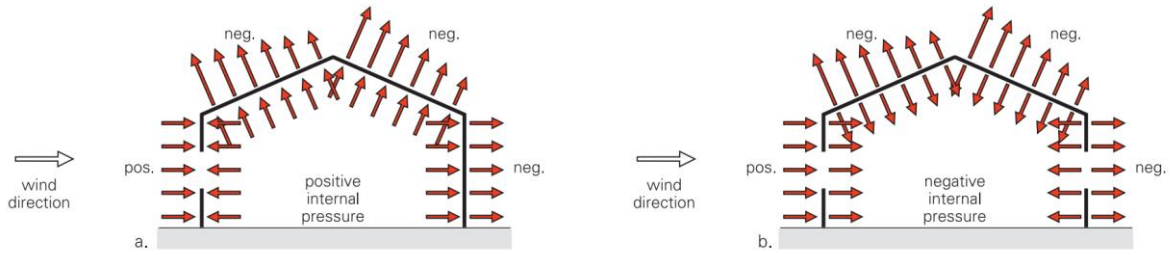


Figure 3.12. External and internal wind pressure calculations for structural design [6]

The external wind pressure w_{e} is calculated as:

$$w_{\mathrm{e}} = c_{\mathrm{pe}} \cdot q_{\mathrm{p}}(z_{\mathrm{e}}) \tag{3.60}$$

Here, $q_{\mathrm{p}}(z_{\mathrm{e}})$ represents the peak velocity pressure at the reference height z_{e}, which is the height used for calculating the external wind pressure.

The external pressure coefficient c_{pe} from Eq.3.60 is selected from tables and corresponds to different zones illustrated in Figure 3.13.

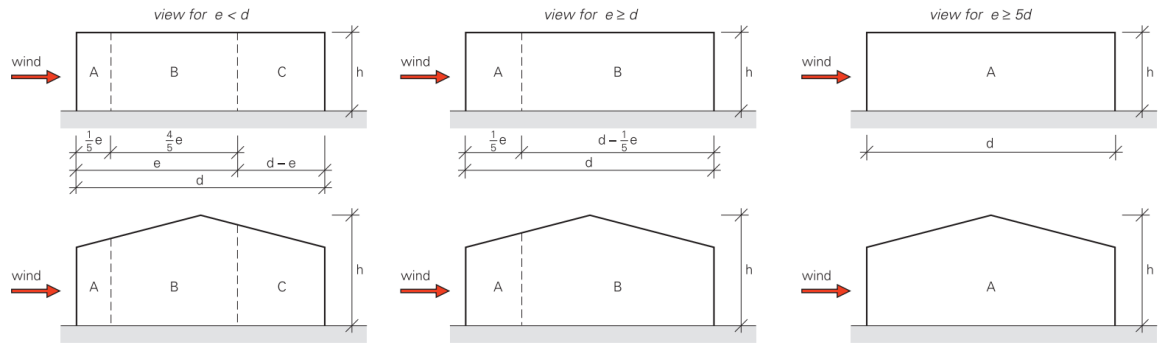


Figure 3.13. Wind pressure coefficient zones for external pressures [6]

Similarly, the internal wind pressure w_i is calculated as:

$$w_i = c_{pi} \cdot q_p(z_i) \tag{3.61}$$

In this equation, $q_p(z_i)$ represents the peak velocity pressure at the reference height z_i for internal pressure calculations. The internal pressure coefficient c_{pi} is determined based on the area of openings in the structure. It is calculated as either $c_{pi} = 0.75 \cdot c_{pe}$ or $c_{pi} = 0.9 \cdot c_{pe}$, depending on the ratio of the area of the opening in the dominant face to the area of the openings in the remaining faces.

3.3.3. Accidental action

In the context of this case, accidental actions are considered solely due to earthquakes. According to EN-1998-1, seismic action is denoted as A_{ed} and refer to the forces and effects induced by earthquakes that must be considered in the structural design of the building. The seismic action A_{ed} is calculated using the formula:

$$A_{ed} = A_{ek} \cdot \gamma_1 \tag{3.62}$$

Where A_{ek} represents the characteristic value of the seismic action, and γ_1 is the importance factor selected from tables based on the consequances of structural failure.

In seismic design, buildings are classified into different importance classes based on their function and occupancy, which determines the seismic actions considered. Importance Class II is the reference case, applying to ordinary buildings where the importance factor is $\gamma_1 = 1.0$, representing standard seismic considerations. Importance Class III includes buildings with large human occupancy or those housing unique and valuable contents, such as museums or archives, necessitating a higher importance factor due to their critical role and the significance

of their contents. Importance Class IV encompasses buildings crucial for civil protection after an earthquake, such as those essential for rescue operations and medical treatment of the injured, which are assigned an elevated importance factor to ensure they remain operational during and after a seismic event. Conversely, Importance Class I is designated for buildings of low economic importance with minimal and infrequent human occupancy, resulting in a lower importance factor that reflects their reduced criticality in the event of an earthquake.

To accurately model seismic actions, it is essential to conduct appropriate investigations to identify the ground conditions, as specified by tabular data. In brief, these conditions are categorized as follows: Type A ground is rock-like geological formations; Type B includes very dense sand or gravel; Type C refers to deep deposits of dense or medium-dense sand or gravel; and Type D represents deposits of loose to medium cohesionless soil.

The horizontal seismic action is characterized by two orthogonal components, which are assumed to be independent and represented by the same response spectrum. For these horizontal components, the elastic response spectrum $S_e(T)$ is defined by following rule:

$$S_e(T) = \begin{cases} a_g \cdot S \cdot \left[1 + \frac{T}{T_B} \cdot (\eta \cdot 2.5 - 1)\right], & 0 \le T \le T_B \\ a_g \cdot S \cdot \eta \cdot 2.5, & T_B \le T \le T_C \\ a_g \cdot S \cdot \eta \cdot 2.5 \left[\frac{T_C}{T}\right], & T_C \le T \le T_D \\ a_g \cdot S \cdot \eta \cdot 2.5 \left[\frac{T_C T_D}{T^2}\right], & T_D \le T \le 4s \end{cases} \tag{3.63}$$

In this equation, $S_e(T)$ represents the elastic response spectrum, which describes the maximum expected response of a structure with a single degree of freedom subjected to seismic action. The variable T is the vibration period of the structure. The design ground acceleration on type A ground is denoted by a_g, calculated as $1.0 \times AgR$, where AgR is the reference peak ground acceleration. The soil factor S adjusts the spectrum based on the type of ground the structure is built on, while η is the damping correction factor, typically taken as 1 for 5% viscous damping. The constants T_B, T_C, and T_D define specific periods within the response spectrum that correspond to different ranges of structural response: T_B marks the lower limit of the constant spectral acceleration branch, T_C the upper limit, and T_D the beginning of the constant displacement response range. These parameters are typically obtained from tabular data based on the spectrum type, ground conditions, and soil factor S. After accounting for all relevant parameters, the resulting response spectrum is illustrated in the figure below:

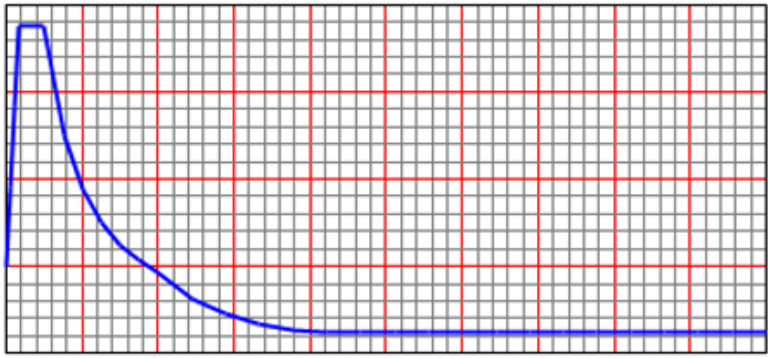


Figure 3.14. Elastic response spectrum for seismic action based on ground conditions

According to Eurocode 8, the internal effects of seismic action must account for the masses associated with all gravity loads, which are defined as:

$$\sum G_{\mathrm{k,j}} + \sum \psi_{\mathrm{E,i}} \cdot Q_{\mathrm{k,i}} \tag{3.64}$$

These terms collectively represent the seismic mass of the structure, a crucial factor for accurately determining the dynamic response under seismic conditions. Here, $\psi_{\mathrm{E,i}}$ denotes the combination coefficient for the *i*th variable action $Q_{\mathrm{k,i}}$. These coefficients are used to account for the likelihood that the variable loads $Q_{\mathrm{k,i}}$ may not be fully present throughout the structure during an earthquake.

Once the seismic mass and response spectrum are defined, the horizontal components of the seismic action must be defined as:

$$E_{\mathrm{Edx}} = S_{\mathrm{x}}(T) \cdot m_{\mathrm{x}} \cdot \mathrm{g} \tag{3.65}$$

$$E_{\mathrm{Edy}} = S_{\mathrm{y}}(T) \cdot m_{\mathrm{y}} \cdot \mathrm{g} \tag{3.66}$$

In these equations, E_{Edx} and E_{Edy} represent the action effects due to seismic action applied along the chosen horizontal *x* and orthogonal *y* axes of the structure, respectively.

The terms $S_{\mathrm{x}}(T)$ and $S_{\mathrm{y}}(T)$ represent the spectral accelerations in units of gravitational acceleration (g), indicating the peak acceleration expected from the seismic event. The parameters m_{x} and m_{y} are the effective seismic masses in the *x* and *y* directions. Both the spectral accelerations and effective masses are assigned by *SAP2000* based on the structural model and input data for the response spectrum. Since spectral accelerations are given in units of g, they must be converted to m/s^2 by multiplying defined values by the gravitational constant.

In accordance with EN 1998-1, the horizontal components of seismic action should be considered simultaneously. For this analysis, eight different combinations are defined to represent various seismic actions:

$$E_1 = E_x + 0.3E_y \tag{3.67}$$

$$E_2 = E_x - 0.3E_y \tag{3.68}$$

$$E_3 = -E_x + 0.3E_y \tag{3.69}$$

$$E_4 = -E_x - 0.3E_y \tag{3.70}$$

$$E_5 = E_y + 0.3E_x \tag{3.71}$$

$$E_6 = E_y - 0.3E_x \tag{3.72}$$

$$E_7 = -E_y + 0.3E_x \tag{3.73}$$

$$E_8 = -E_y - 0.3E_x \tag{3.74}$$

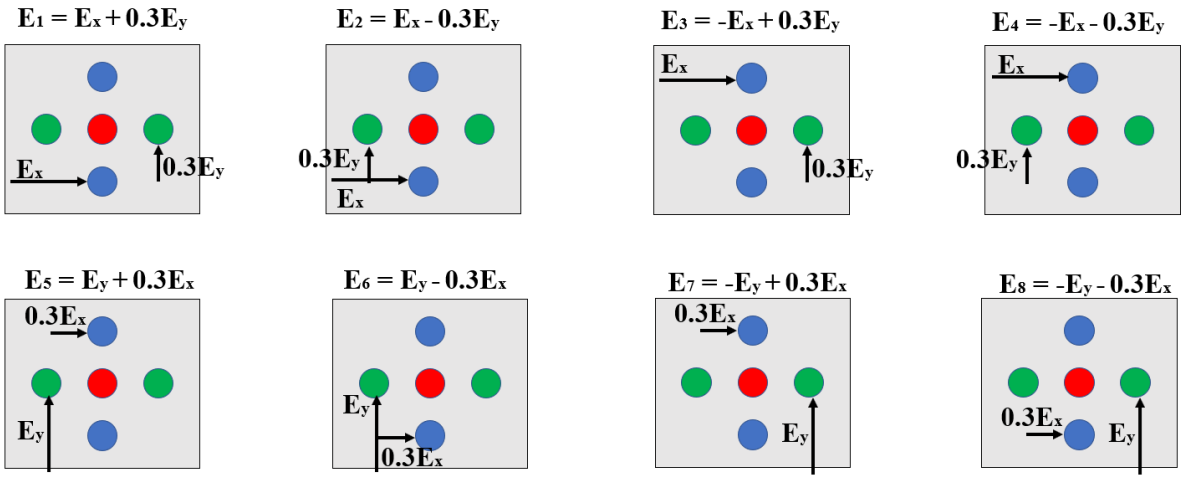


Figure 3.15. Seismic action combinations for horizontal components of seismic effects

3.4. Combinations of actions

Combinations of actions are defined for both the ultimate limit state and the serviceability limit state. These combinations are expressed as the sum of different design values of actions F_d for different scenarios:

$$\sum F_d \tag{3.75}$$

The ultimate limit states address potential structural failures or collapses that could compromise the safety of people and the integrity of the structure. These states consider scenarios such as the loss of equilibrium, failure due to excessive deformation or instability, and failure caused by fatigue or other time-dependent effects.

In contrast, serviceability limit states focus on structure's performance under normal conditions, including user comfort and the building's appearance. These states can be classified as either reversible or irreversible, depending on whether the effects of the actions persist after they are removed. Key concerns in serviceability limit states include deformations that affect the building's appearance or function, vibrations that cause discomfort or reduce structural effectiveness, and damage that could impact the building's appearance, durability, or functionality.

Combinations of actions can include permanent (G), variable (Q), and accidental (A) actions. These actions are associated with corresponding partial factors γ or combination factors Ψ. The factors for different actions are selected from defined tables. When combination factors appear with different variable actions, they are used as follows: $\Psi_{0,\mathrm{j}}Q_{\mathrm{k,j}}$ is used for ultimate limit states and irreversible serviceability limit states, $\Psi_{1,\mathrm{j}}Q_{\mathrm{k,j}}$ is used for ultimate states involving accidental actions and reversible serviceability limit states, $\Psi_{2,\mathrm{j}}Q_{\mathrm{k,j}}$ is used for ultimate limit states involving accidental actions and for long-term effects calculations.

For the purposes of the thesis, the first defined combination of actions is for the ultimate limit state in persistent and transient design situations. This combination is expressed as:

$$\sum F_{\mathrm{d}} = \sum_{\mathrm{i}} \gamma_{\mathrm{G,i}} G_{\mathrm{k,i}} + \gamma_{\mathrm{Q,1}} Q_{\mathrm{k,1}} + \sum_{\mathrm{j>1}} \gamma_{\mathrm{Q,j}} \Psi_{0,\mathrm{j}} Q_{\mathrm{k,j}} + (\gamma_{\mathrm{P}} P_{\mathrm{k}}) \tag{3.76}$$

Here, the combination of actions is determined by summing the permanent actions $\left(\sum_{\mathrm{i}} \gamma_{\mathrm{G,i}} G_{\mathrm{k,i}}\right)$, where $G_{\mathrm{k,i}}$ represents the *i*th permanent action, multiplied by the corresponding partial factor $\gamma_{\mathrm{G,i}}$ selected based on how the permanent action affects the combination of actions. A single governing variable action $\left(\gamma_{\mathrm{Q,1}} Q_{\mathrm{k,1}}\right)$ is added to the permanent actions, where $\gamma_{\mathrm{Q,1}}$ represents the partial factor for the variable action. The remaining variable actions $\left(\sum_{\mathrm{j>1}} \gamma_{\mathrm{Q,j}} \Psi_{0,\mathrm{j}} Q_{\mathrm{k,j}}\right)$ are included in the combination as the sum of variable actions multiplied by the partial factor $\gamma_{\mathrm{Q,j}}$ and the combination factor $\Psi_{0,\mathrm{j}}$. To fully define the combination of actions, other variable actions are used as the governing variable, while the initial governing variable becomes part of the sum. The term $(\gamma_{\mathrm{P}} P_{\mathrm{k}})$ is included to account for peak actions, which are not defined for the current case but could be included if necessary.

Next, the combination of actions for the ultimate limit state with the purpose of *accidental design situations* is considered. This combination is important for ensuring the resilience of structures under unexpected or extreme events. It is expressed as:

$$\sum F_{\mathrm{d}} = \sum_i G_{\mathrm{k,i}} + \gamma_{\mathrm{Q,1}} Q_{\mathrm{k,1}} + \sum_{\mathrm{j>1}} \Psi_{2,\mathrm{j}} Q_{\mathrm{k,j}} + T \tag{3.77}$$

In this equation, *T* represents the variable thermal actions, which are not included in other variable actions. As before, to fully define the combination, the governing variable action is rotated, now excluding *T* from the list of variable actions.

For the serviceability limit state, according to EN 1990, the characteristic combination of actions is defined as:

$$\sum F_{\mathrm{d}} = \sum_{\mathrm{i}} G_{\mathrm{k,i}} + Q_{\mathrm{k,1}} + \sum_{\mathrm{j>1}} \Psi_{0,\mathrm{j}} Q_{\mathrm{k,j}} + (P_{\mathrm{k}}) \tag{3.78}$$

Here, the part representing the permanent actions includes only the permanent actions $G_{\mathrm{k,i}}$, without any partial factors. This is because, in serviceability limit states, the focus is on the structure's performance under normal conditions, where the full characteristic value of the actions is considered without amplification by partial factors. The variable actions are again represented by the governing action and the remaining actions. The combination is fully defined when each possible governing action is used from the set of variable actions.

Finally, the combination of actions required for seismic design situations is defined as:

$$\sum F_{\mathrm{d}} = \sum_{\mathrm{i}} G_{\mathrm{k,i}} + A_{\mathrm{Ed}} + \sum_{\mathrm{j}} \Psi_{2,\mathrm{j}} Q_{\mathrm{k,j}} + (P_{\mathrm{k}}) \tag{3.79}$$

In this expression, A_{Ed} represents the seismic action in an ultimate limit state. This term is crucial for accounting for the effects of seismic events on the structure, ensuring that it can withstand such forces in accordance with the design criteria for safety and stability.

4. Analysis and optimization

The final section of the thesis demonstrates the application of the developed *Python*-based tool, exploring its capabilities and various approaches for finding optimal solutions. It highlights how the tool can be used to effectively address complex problems and optimize processes, showcasing its versatility and practical utility.

4.1. Standardized steel structures supporting belt conveyors

Using the previously defined model and parametric definitions, it is now possible to efficiently standardize and test various models. Figure 4.1. illustrates the progression of standardized belt width: BW650 (*belt width of 650 mm*), BW800, BW1000, BW1200, and BW1400. Overall geometric data is generated by the user in *Excel* and is then used in *Python* as an input to the parametric functions. Such an approach eliminates the need for manual modeling and reduces the time spent on repetitive tasks and eliminates the possibility of human error.

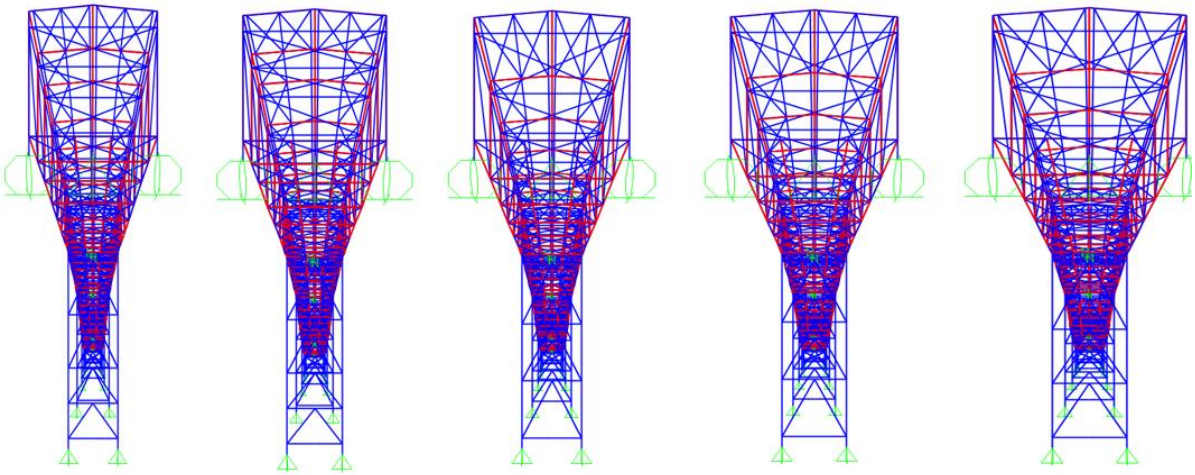


Figure 4.1. Standardized steel structures supporting belt conveyors

In addition to standardization for different belt widths, the parametric definition enables the creation of an unlimited variety of conveyors by specifying different lengths and heights through coordinate data, as shown in Figure 4.2. This approach allows for the design of models that meet various requirements for conveyor length and elevation.

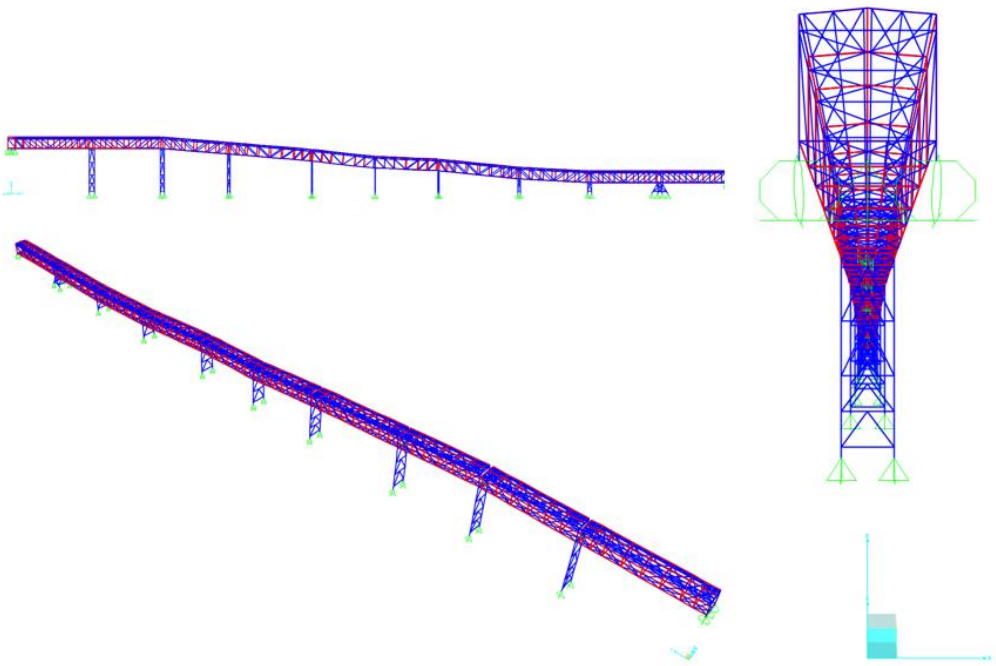


Figure 4.2. Demonstration of steel structures supporting belt conveyors with expanded length and height

4.2. Steel check according to Eurocode

Inside *SAP2000* it is possible to check assigned sections according to Eurocode which involves calculating various values to verify sections under defined actions. The most relevant metric used for upcoming optimization is the demand/capacity ratio (D/C), which compares the applied load (demand) to a section's load-carrying capacity.

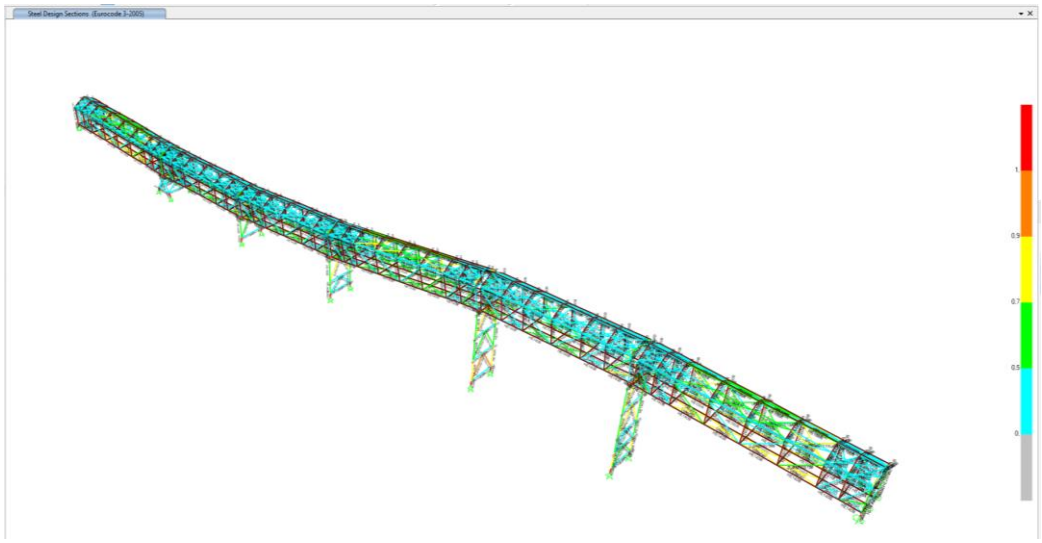


Figure 4.3. D/C ratio calculated for every frame of the steel structure

4.2.1. Trial and error approach for finding sections to pass steel check

The trial-and-error approach for identifying frame sections that meet the steel check criteria, based on the D/C ratio calculated by *SAP2000*, is illustrated in the flow diagram shown in Figure 4.4. Each frame in the model is assigned to a corresponding group of elements within *SAP2000*. For instance, frames representing the columns of trestles are categorized into their specific group. This grouping method is applied to all other frames as well. Each group is then assigned a specific type of section, which could be a wide section, a channel, or double channel sections.

Within the *Python* script, a database of all possible sections for each type is defined. During the first iteration of the section-finding function, the first section from the list is assigned to the corresponding groups based on the type of section that can be applied to that group. The model, now defined with these sections, is then sent to *SAP2000* along with the defined actions and combinations of actions. *SAP2000* automatically calculates the D/C ratio for the sections. If the D/C ratio exceeds 1, it indicates that the frame has failed the check. The function is set up so that if a single frame within a group fails the check, the entire group is considered to have failed. If every member of a group passes the check, the group and its respective frames are stored in a verified group list. If not, the process enters the next iteration of the section-finding function, where the next possible section from the list is applied to the group that failed the check in the previous run. This process is repeated until all groups successfully pass the check.

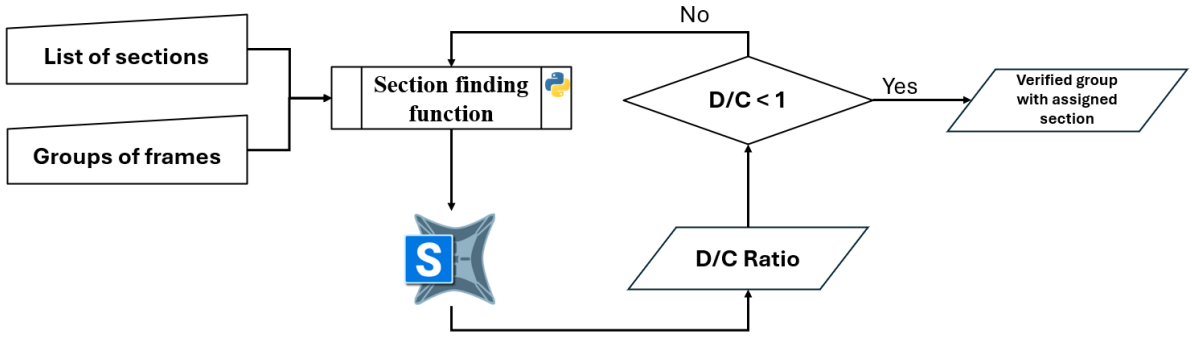


Figure 4.4. Flow diagram of the trial-and-error function created for finding sections of frames that pass the steel check

4.3. Analysis of the reference case

The reference case is fully defined with all geometric data, groups, sections, materials, restraints, actions, and combinations of actions. All these elements are assigned using *Excel* input, which is processed by the corresponding functions in a *Python* script. Geometric data is assigned as previously explained. Load action values are automatically applied to specific frames or areas, requiring the user only to define the value, thereby minimizing time and effort. The combinations of actions are automatically assigned based on predefined equations, with the *Python* tool generating all possible scenarios and rotations of governing actions for each case.

Given the infinite number of possibilities and configurations, the following analysis focuses on the 153rd ultimate limit state (ULS) combination of actions. This combination encompasses all permanent actions, while using the variable belt force actions as the single governing variable action and considering all other variable actions as the sum.

This specific combination of actions serves as a benchmark for the upcoming optimization. The ultimate limit state represents the maximum load-bearing capacity or failure condition, and it will be used as the critical state for further analysis and optimization.

Furthermore, the instance analysis is conducted for the BW650 model. The defined model first enters the section-finding function, where, after the required number of iterations, every member and group successfully pass the steel check based on the demand-to-capacity (D/C) ratio, as illustrated in Figure 4.5.

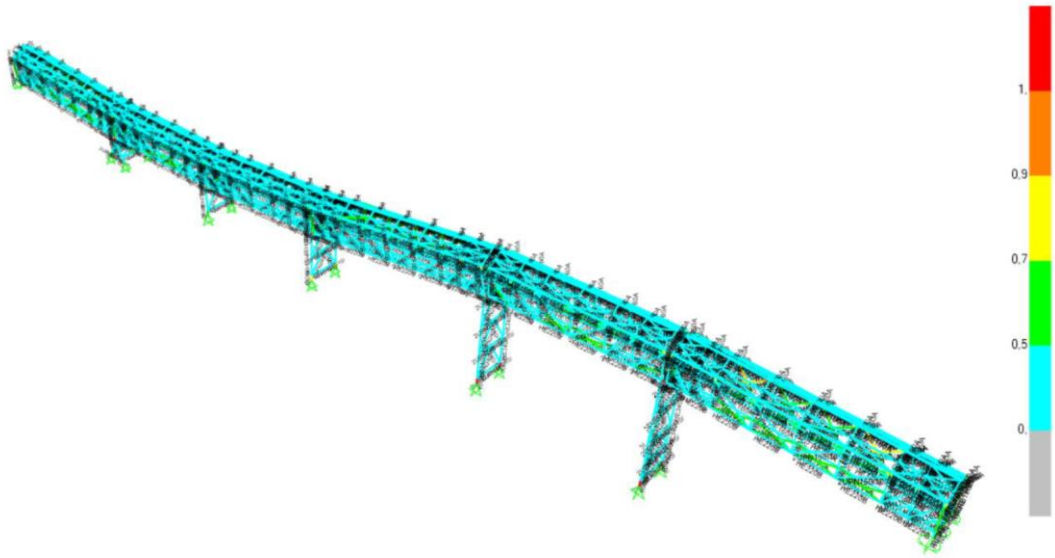


Figure 4.5. D/C ratio for BW650 model before optimization

Based on the selected sections and applied actions, the total displacement of the structure is illustrated in Figure 4.6. As observed, the critical areas of displacement are approximately in the middle between the start and the end of a single gallery, which is expected due to the bending moments being most pronounced in these regions. This occurs because the structure is subjected to maximum flexural stresses in these spans, where the load distribution and the absence of intermediate supports contribute to higher deflections.

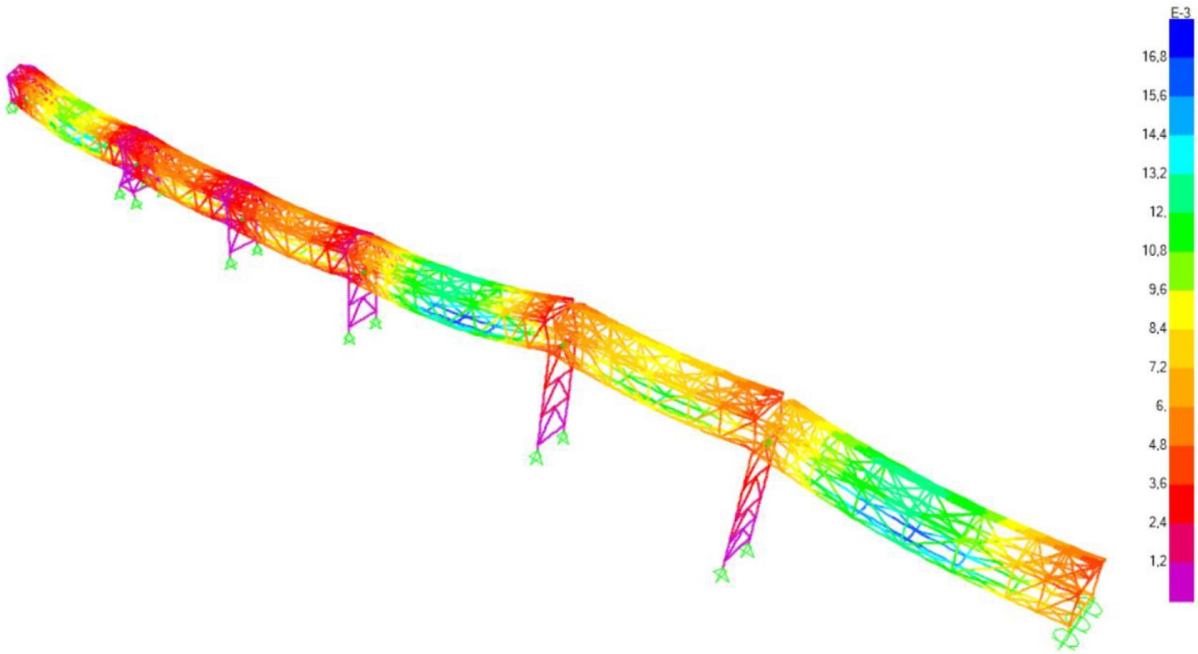


Figure 4.6. Total displacement of the BW650 model before optimization

4.4. Optimization of the reference model

The reference case is established. The main target of optimization is to minimize the volume while decreasing total displacement of a single node to under 15 *mm* and keeping the D/C ratio at a similar value, as the structure has already passed the steel check. The volume of the structure is calculated using the following equation:

$$V_{\mathrm{TOT}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} A_{\mathrm{cs,i}} \cdot L_{\mathrm{i}} \tag{4.1}$$

where V_{TOT} represents the total volume of the structure, calculated as the sum of the volumes of individual frame elements. Each frame's volume is determined by multiplying its cross-sectional area $A_{\mathrm{cs,I}}$ of the *i*th frame by its length L_{i}.

Based on the described problem, the optimization function is defined as follows:

$$\text{minimize} \qquad \frac{V_\text{i}(\mathbf{x})}{\text{V}} \tag{4.2}$$

where the V represents the expected value of target volume, while the V_i represents the volume of the *i*th structure, retrieved from *SAP2000*, after generating it with design values stored within the design vector **x** which is defined as:

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5] \tag{4.3}$$

The vector **x** consists of individual design variables which are shown in Table 4.1, together with lower and upper bounds.

Table 4.1. Design variables

x_i	x_i^L	x_i^U	Unit	Representative Symbol
x_1	2	6	m	L_FS
x_2	2.5	3	m	H_B
x_3	0	2	°	α_TR
x_4	25	30	°	θ_TR
x_5	2	4	m	H_MA

The first constrain is oriented towards limiting maximum deformation, defined as:

$$\frac{U_\text{i} - U_\text{allowed}}{U_\text{allowed}}$$

Here, the U_allowed is equal to 15 *mm*, as explained at the beginning, while the U_i represents the *i*th total displacement of a single node retrieved from analysis initialized in *SAP2000*.

The second constrain ensures that the *D/C* ratio remains in approximately the same range as in the reference case:

$$\frac{(D/C)_\text{i} - (D/C)_\text{ref.}}{(D/C)_\text{ref.}}$$

Out of every generated solution, only the best one is saved at the end. The design variables for this optimal solution are listed in the following table:

Table 4.2. Best solution design variables

x_i	$Value$	Unit
x_1	5.67	m
x_2	2.52	m
x_3	0.34	°
x_4	25.24	°
x_5	3.87	m

The design variables produce the following design, with the total displacement below 15 *mm*, as illustrated in the accompanying figure:

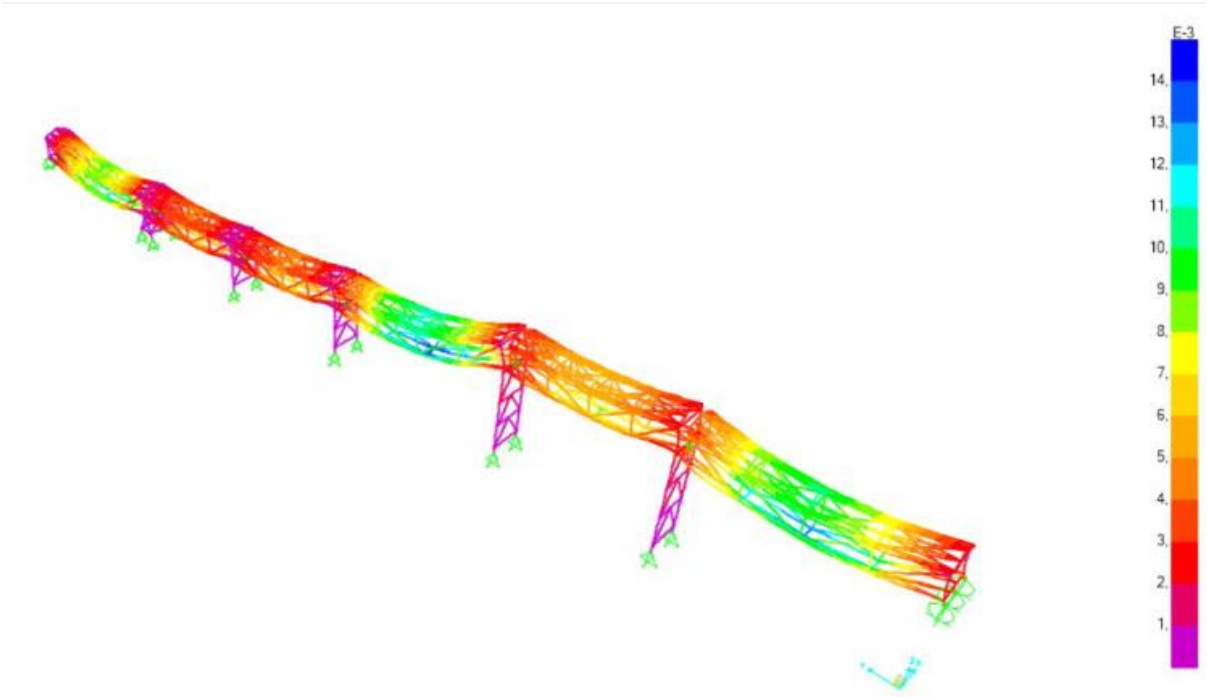


Figure 4.7. Total displacement of the BW650 model after optimization

The final steel check, shown in Figure 4.9, confirms that the steel check is satisfied once again. As observed, the structure passes the check with values comparable to those from the reference run.

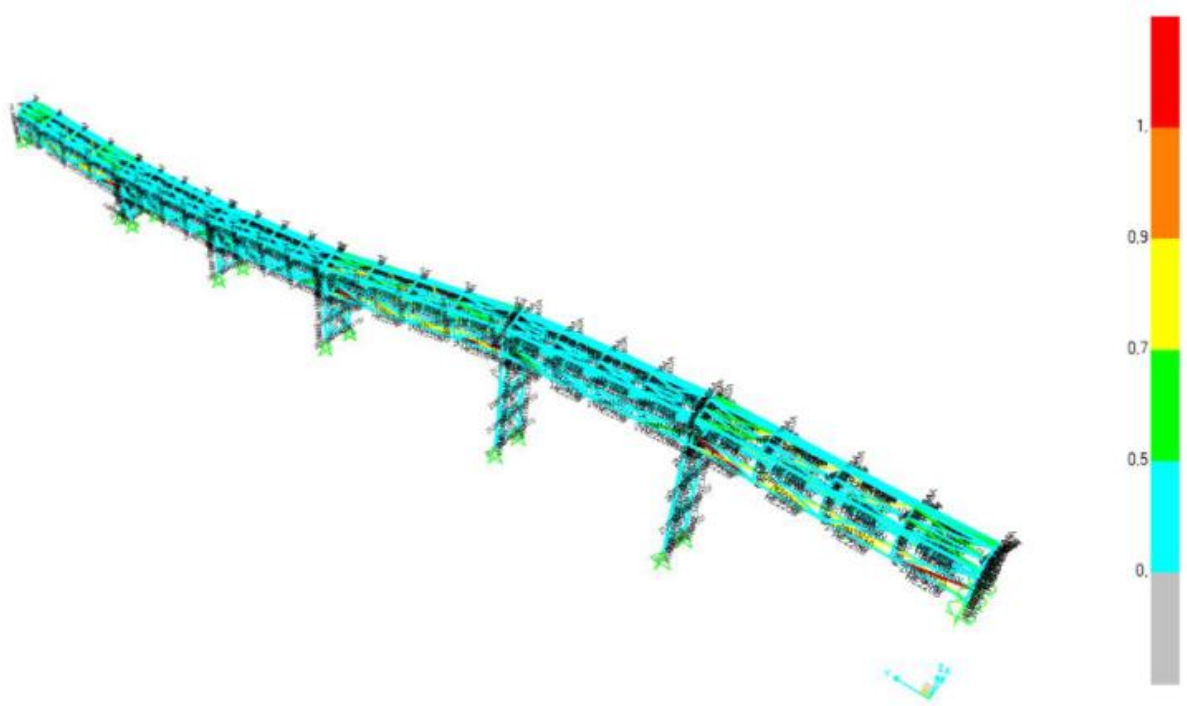


Figure 4.8. D/C ratio for the BW650 model after optimization

The initial measured volume of the structure was 21.89 m³. After optimizing the design variables, the volume of the revised structure is reduced to 19.77 m³, while successfully achieving a total displacement of less than 15 mm. If the total volume is multiplied by the density of used S355 steel, the weight of the structure was reduced from 171,836.5 *kg* to 155,194.5 *kg*, saving a total of 16,642 kg of raw material.

5. Conclusion

This thesis successfully developed a *Python*-based tool for the numerical modeling and optimization of self-supporting steel structures used in ore transportation via belt conveyors. The integration with *SAP2000* for finite element analysis enabled a streamlined process, where complex model definitions, including constraints, loads, and load combinations, were automated. The *Excel*-based user interface allowed for easy input of model parameters, facilitating a smooth transition between structural analysis and optimization phases.

The tool significantly improved engineering efficiency by speeding up model definition and reducing the possibility of human error. The optimization component, using genetic algorithms, identified cost-effective designs that met industry standards and constraints, ensuring the final designs were both structurally and economically efficient.

The tool's standardization capabilities contributed to improved safety, reduced material waste, and enhanced sustainability, while also making it easier to model similar designs repeatedly. By enabling the consistent application of Eurocode standards, the tool ensured compliance with relevant safety and performance regulations, thereby enhancing the overall reliability of the belt conveyor structures.

In practical application, the tool demonstrated its versatility by standardizing and optimizing various belt conveyor models with different inputs. The case study presented in the thesis confirmed the tool's ability to minimize material usage while maintaining structural integrity, as evidenced by the satisfactory demand/capacity ratios and controlled displacements.

In conclusion, this research meets a crucial need in the mining industry by combining a software approach with traditional engineering methods. Future work could explore further enhancements, such as incorporating additional optimization algorithms or expanding the tool's applicability to other types of structures.

Bibliography

[1] Alles, R.: “Conveyor Belt System Design”, ContiTech Transportbandsysteme, Hannover, 1994.

[2] Conveyor Equipment Manufacturers Association (CEMA): “Belt Conveyors for Bulk Materials”, Conveyor Equipment Manufacturers Association, United States of America, 2002.

[3] Logan, D.: “A First Course in the Finite Element Method 4th Edition”, Thomson, Wisconsin, 2007.

[4] Russel Rhinehart, R.: “Engineering Optimization: Applications, Methods and Analysis”, John Wiley & Sons Inc, Hoboken,2018.

[5] Kochenderfer, M.; Wheeler, T.: “Algorithms for Optimization”, The MIT Press, Massachusetts, 2019.

[6] Snijder, H.H.; Steenbergen, M.; van Eldik, C.H: “Structural basics: Analysis and design of steel structures for buildings according to Eurocode 0, 1 and 3”, Bouwen met Staal, 2019.

List of Symbols

β_{R}	angle of repose
β_{S}	angle of surcharge
Q_{m}	mass flow of conveyed material
ρ	bulk density
Q_{v}	volume flow of conveyed material
δ_{CR}	critical conveying gradient angle
B	belt width
E_{f}	drop energy
m_{k}	mass of a cubic lump
K_{f}	shape factor
g	gravitational acceleration
h_{f}	height of free fall
L	standard idler tube length
D_{R}	idler diameter
n_{R}	idler speed
v	conveying speed
A	filling cross-section area
$Q_{\mathrm{v,th}}$	theoretical volume capacity
b	effective belt width
ρ_1	defined feeding rate
$Q_{\mathrm{v,eff}}$	effective conveying capacity
$\{f\}$	vector of element nodal forces
$[k]$	element stiffness matrix
$\{d\}$	vector of unknown generalized displacements
$\{F\}$	vector of global nodal forces
$[K]$	global stiffness matrix
T^*	transformation matrix

$\hat{d}$	local displacement
k	global stiffness matrix
L_{E1}	total length of element
L^{n}	beam's length
E^{n}	modulus of elasticity
σ^{n}	stress
f_m	the *m*th objective function
Ω	search space
$x_{\mathrm{D,i}}^{\mathrm{L}}$	lower bound
$x_{\mathrm{D,i}}^{\mathrm{U}}$	upper bound
g_j	the *j*th constraint
$C_{\mathrm{i,j}}$	chromosome
$G_{\mathrm{n,j}}$	gene
P	population
$P_{\mathrm{t,i}}$	the *i*th point of type *t*
$ID_{\mathrm{t,i}}$	the *i*th ID of type *t*
$\boldsymbol{V}_{\mathrm{t,i}}$	the *i*th vector made from points of type *t*
$\boldsymbol{M}_{\mathrm{t,i}}$	the *i*th matrix made from points of type *t*
$\overrightarrow{P_{\mathrm{A}}P_{\mathrm{B}}}'$	the offset scaled vector
k	scale factor
$\overrightarrow{u_{\mathrm{AB}}}$	unit vector
$\overrightarrow{P_{\mathrm{A}}P_{\mathrm{B}}}$	direction vector
$\|\overrightarrow{P_{\mathrm{A}}P_{\mathrm{B}}}\|$	normalized vector
Δy	displacement
θ	angle of the gallery
$(L_{\mathrm{rel.}})_{\mathrm{j}}$	relative length of the *j*th gallery
L_{FS}	full-sized span
$(L_{\mathrm{red.}})_{\mathrm{j}}$	reduced span

$(n_{\mathrm{FS}})_{\mathrm{j}}$	number of possible L_{FS} segments
L_2	half the length of the *j*th gallery
$n_{\mathrm{red.}}$	number of possible $(L_{\mathrm{red.}})_{\mathrm{j}}$ values for a single gallery
f_{ID}	function for generating points with corresponding *ID*s in *SAP2000*
f_{L}	function for creating connections between points in the longitudinal direction
f_{VT}	function for creating connections between points in vertical or transversal direction
f_{VAB}	function for creating vertical angle bracing
f_{IVB}	function for creating inverted V bracing
$H_{\mathrm{g,int}}$	initial height of gallery
E	height of raised heel
β	roof angle
H_{B}	base gallery height
H_{RHE}	raised heel end point
H_{P}	roof's peak point
f_{UP}	function for generating upper points
H_{n}	perpendicular distance from upper to bottom points
H_{A}	available height
H_{MA}	maximum allowed height
n_{MA}	number of H_{MA} segments
α_{TR}	angle of trestle
θ_{TR}	angle of fixed-point trestle
γ_{f}	partial factor for actions
F_{rep}	value of action
F_{k}	characteristic value of action
Ψ	combination factor
F_{d}	design value
q_{k}	uniformly distributed load
q_{k}	line load

Q_{k}	concentrated loads
D_{s}	dust load
d_{s}	thickness of the dust layer
s	snow load
μ_i	snow load shape coefficient
C_{e}	exposure coefficient
C_{t}	thermal coefficient
ΔT_{u}	uniform temperature components
T	average temperature
T_0	initial temperature
T_1	average internal temperature during summer
$T_{\mathrm{out,summer}}$	outside temperature during summer
T_{max}	maximum air temperature in shade
T_4	temperature defined in *National Annexes* based on surface color
T_2	average internal temperature during winter
$T_{\mathrm{out,winter}}$	minimum shade air temperature T_{min}
$v_{\mathrm{b,0}}$	basic wind speed
$v_{\mathrm{m}}(z)$	mean wind velocity
v_{b}	basic wind velocity at 10 m
c_{dir}	directional factor
c_{season}	seasonal factor
$c_{\mathrm{r}}(z)$	roughness factor
z_0	roughness length
z_{min}	minimum height
z_{max}	maximum height
k_{r}	terrain factor
$c_0(z)$	orography factor
q_{b}	basic wind pressure
$q_{\mathrm{p}}(z)$	peak velocity pressure

$l_{\mathrm{v}}(z)$	turbulence intensity
σ_{v}	standard deviation of turbulent component of wind velocity
k_{l}	turbulence factor
$c_{\mathrm{e}}(z)$	exposure factor
w_{e}	external wind pressure
$q_{\mathrm{p}}(z)$	peak velocity pressure
c_{pe}	external pressure coefficient
w_{i}	internal wind pressure
c_{pi}	internal pressure coefficient
γ_1	importance factor
$S_{\mathrm{e}}(T)$	elastic response spectrum
T	vibration period
a_{g}	design ground acceleration on type A ground
AgR	reference peak ground acceleration
S	soil factor
η	damping correction factor
$Q_{\mathrm{k,i}}$	variable load
E_{Edx}	action effect due to seismic action applied along *x* axes
E_{Edy}	action effect due to seismic action applied along orthogonal *y* axes
$S_{\mathrm{x}}(T)$	spectral acceleration
m_{x}	effective seismic masses in *x* direction
m_{y}	effective seismic masses in *y* direction
F_{d}	design values of actions
$G_{\mathrm{k,i}}$	*i*th permanent action
A_{Ed}	seismic action in ultimate limit state

List of figures

List of tables

Abstract

This thesis presents the development of a *Python*-based tool designed to improve the numerical modeling and optimization of self-supporting steel structures used in ore transportation via belt conveyors. The tool integrates seamlessly with *SAP2000*, a commercial finite element analysis software, to automate complex model definition processes including the application of constraints, loads, and load combinations. By automating these tasks, the tool significantly reduces the time and effort required for engineering analysis, streamlining project execution. The optimization component focuses on identifying cost-effective designs that satisfy all relevant constraints and standards. Additionally, the tool's standardization features enhance safety, reduce material waste, and ensure better compliance with industry norms.

Keywords: numerical modeling, optimization, steel structures, belt conveyors, Python, SAP2000, finite element method, genetic algorithm, standardization

Sažetak

Ovaj rad predstavlja razvoj alata unutar *Python* okruženja dizajniranog za poboljšanje numeričkog modeliranja i optimizacije samonosivih čeličnih konstrukcija korištenih za prijenos rude pomoću trakastih transportera. Alat se integrira sa *SAP2000*, komercijalnim softverom za analizu metodom konačnih elemenata, kako bi automatizirao složene procese definiranja modela, uključujući primjenu rubnih uvjeta, opterećenja i kombinacija opterećenja. Automatizacijom ovih zadataka, alat značajno smanjuje vrijeme i napor potreban za inženjersku analizu, čime se pojednostavljuje izvođenje projekata. Komponenta optimizacije usmjerena je na pronalaženje isplativih rješenja koja zadovoljavaju sva relevantna ograničenja, uvjete i standarde. Dodatno, standardizacijske značajke doprinose poboljšanoj sigurnosti, smanjenju otpada materijala i boljoj usklađenosti s industrijskim normama.

Ključne riječi: numeričko modeliranje, optimizacija, čelične konstrukcije, trakasti transporteri, Python, SAP2000, metoda konačnih elemenata, genetski algoritmi, standardizacija