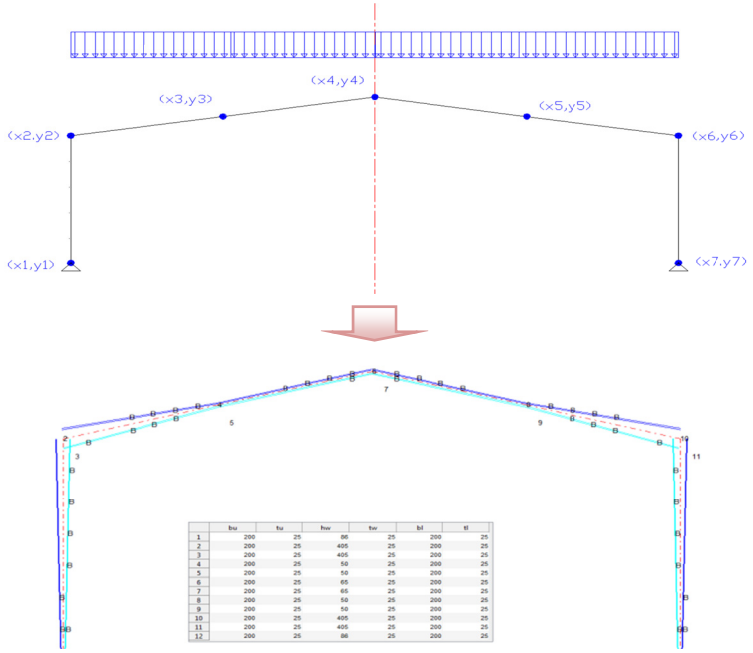
Design and optimization of steel portal frames according to Eurocode using genetic algorithm

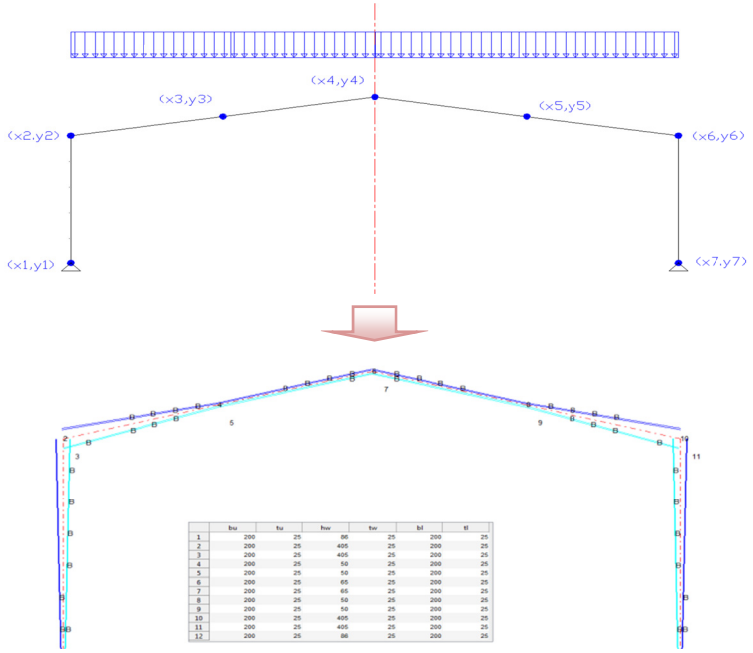


Ghassan Numan

Avdelningen för Konstruktionsteknik
Lunds Tekniska Högskola
Lunds Universitet, 2012

Rapport TVBK - 5215

Design and optimization of steel portal frames according to Eurocode using genetic algorithm



Ghassan Numan

Avdelningen för Konstruktionsteknik
Lunds Tekniska Högskola
Lunds Universitet, 2012

Rapport TVBK - 5215

Department of Structural Engineering
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

Avdelning för Konstruktionsteknik
Lund Tekniska Högskola
Box 118
221 00 LUND

Design and optimization of steel portal frames according to Eurocode using genetic algorithm

Design och optimering av stål portalramer enligt Eurokod

Ghassan Numan

2012

Rapport TVBK-5215
ISSN 0349-4969
ISRN: LUTVDG/TVBK-12/5215(146)

Master’s thesis
Supervisor: Daniel Honfi, Ph.D. student at Department of Structural Engineering
July 2012

Foreword

This thesis is done during Autumn and Summer 2012 at the Division of Structural Engineering, with support of Division of Structural Mechanics, Lund Institute of Technology and the Department of the Civil and Environmental Engineering, Chalmers University of Technology.

First I would like to thank my supervisor Ph.D. student Daniel Honfi at the Division of Structural Engineer, LTH for his unceased and helpful support during the thesis period. I want also to thank Prof. Mohammad Al-Emrani at the Department of the Civil and Enviromental Engineeing, Chalmers, Prof. Per Johan Gustafsson and Prof. Ola Dahlblom at the Division of Structural Mechanics, LTH for their help and support.

Abstract

One of the essential engineer's jobs is to achieve the most economical technical solutions. Weight optimization is important since it provides a structure that can carry the applied loads in addition to fulfilling the structural requirements. In this project, a Matlab-algorithm has been developed to find the optimum design of steel portal frames according to "Eurocode 3: Design of steel structures" with regard to the weight. To get the final design of the frame, some inputs have to be specified by the user such as the material, the coordinates, the loads and the distribution pattern of the bracings.

The algorithm goes through three essential steps before getting the optimal frame:

- ✓ **Analyzing the frame with help of CALFEM-toolbox:** In this step, the frame is geometric nonlinearly analyzed due to certain load combination according to the Ultimate Limit State (ULS) and Serviceability Limit State (SLS) criterion. The internal forces and the displacement established and the axial force diagram, shear force diagram, bending moment diagram and the deformed shape of the frame to be calculated and plotted.
- ✓ **Checking the capacity of the frame according to Eurocode:** Some constraints with regard to EC should not be violated. These checks may refer to the frame capacity, checking the capacity against the risk of buckling and the deformations which should not be exceed the Serviceability Limit State (SLS) limitations.
- ✓ **Finding the optimal design of the frame using Genetic Algorithm optimization method:** Genetic Algorithm (GA) is an iterative searching method based on the evolution of species' principle. The algorithm repeat the two steps above every iteration cycle trying to find the best design that has the minimum weight without violating the limitations. Since it is an iterative process, the time required to find the optimum design depends on some factors such as the speed of the computer, the number of variables the size of frame mesh etc.

The project ends with some testing examples in which a frame with width span of 20 m, and a height of 6.5 m and uniformly distributed loads as snow loads are acting on the roof of the frame. In these examples, the algorithm has been tested to compare the design that have been got in case of fully braced frame, unbraced and by letting the algorithm to find the optimal number and position of the bracings along the frame. Another comparison has been done to see the difference of the design in case of absence /existing of the deformation limitations.

Table of contents

1 Introduction

1.1 Background

In steel structures, the designer aims to achieve a frame solution that fulfills the requirements of Ultimate Limit State (ULS) and Serviceability Limit State (SLS) with minimum weight and price.
Nowadays, most of the commercial design softwares are used to analyze and design the steel structures depending on the inputs of the designer. This is not the final solution, because it needs to refine in order to reach the optimal solution that can be tough and time consuming depending on the experience of the designer.
Genetic Algorithm (GA), – based optimization methods might be helpful to overcome this problem and lead to more economical structural designs. The reason behind choosing GA optimization method among the others is because of its efficiency, rapidity and the ability of using it in almost any kinds of problems. More about GA in the next chapters. Few attempts have been carried out in the past years to use GA at the optimization of steel portal frames (Chen & Hu, 2008) (Hradil, et al., 2010).

1.2 Purpose and goal

The aim of this project is to develop an algorithm using Matlab (MathWorks, 2011) in order to find the optimal steel portal frame design for certain load configurations. The optimization is with respect to the weight and refers to some design variables such as the dimensions of the cross section, the geometry of the frame or location and the number of the external and the internal bracings. The design is carried out according to Eurocode 3: Design of steel structures (CEN, 2005).

1.3 Limitations

The limitations of the current study are as follows:

- All the sections of the frame are treated like I-cross sections, see section 3.2.4.
- Elastic global analysis with in-plane imperfections and geometric nonlinearity.
- Cross section class 4 is not included, see section 2.1.1.3.

2 Theory

2.1 Portal Steel frames

Figure 1: Steel portal frames for industrial buildings (http://www.solid-structures.com).

The early types of steel portal frames founded during World War II were developed due to the need of cheap and quick solution for single-storey industrial structures. The word “portal” in Latin means “gate”, hence the shape of the portal frames. It consists of the vertical members, called *stanchions* or columns and roofing members called *rafters*. The connection points between the stanchions and the rafters are called *eaves* and the connection point of two rafters which are located at the symmetric line of the frame called the *ridges*, see Figure 2.

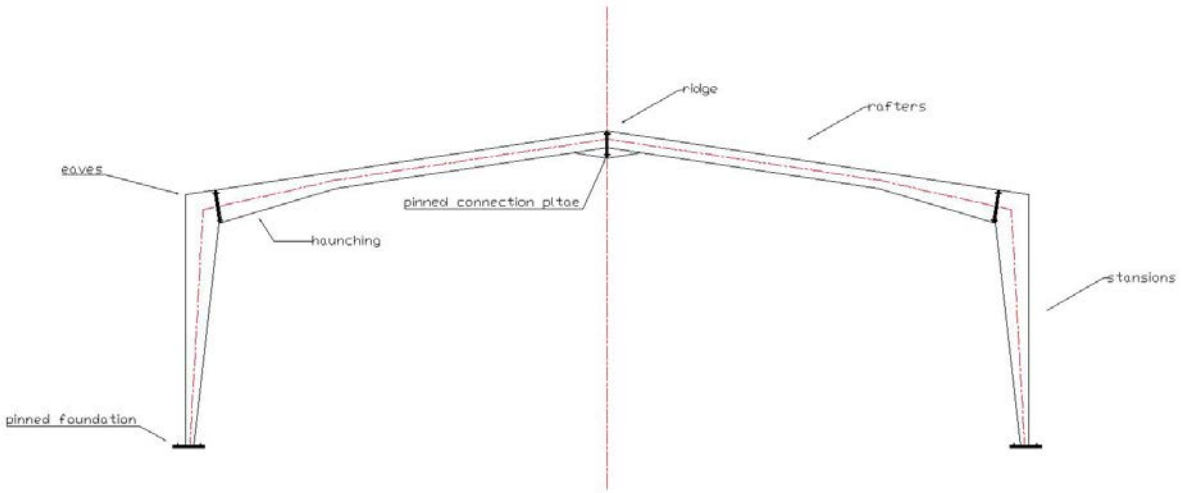


Figure 2: Steel portal frame.

Both the rafters and the stanchions are often made of welded slender plate elements and usually have tapered form and variable cross section in order to sustain the axial load, the shear forces, the bending moment and the deformations and to provide an economic solution. Generally the spacing of center-to-center frames (out of plane distance) is from 6 – 7.5 m and the average eaves height is between 6 – 15 m. Moment-resistance connections (haunches) are required at the eaves and the ridges in form of higher cross sections. The connection between the columns and the foundation is designed to be either pinned or fixed depending on the type of the foundation and other factors. Frames with pinned connection are heavier since less bending moment has to be taken by the frame at the foundations; instead high moment concentrated at the eaves. While at the fixed foundation, the bending moment is higher at the foundations and it is better distributed along the whole structure, see Figure 3 below. However maintaining of fixed foundations is more expensive than the pinned alternatives.

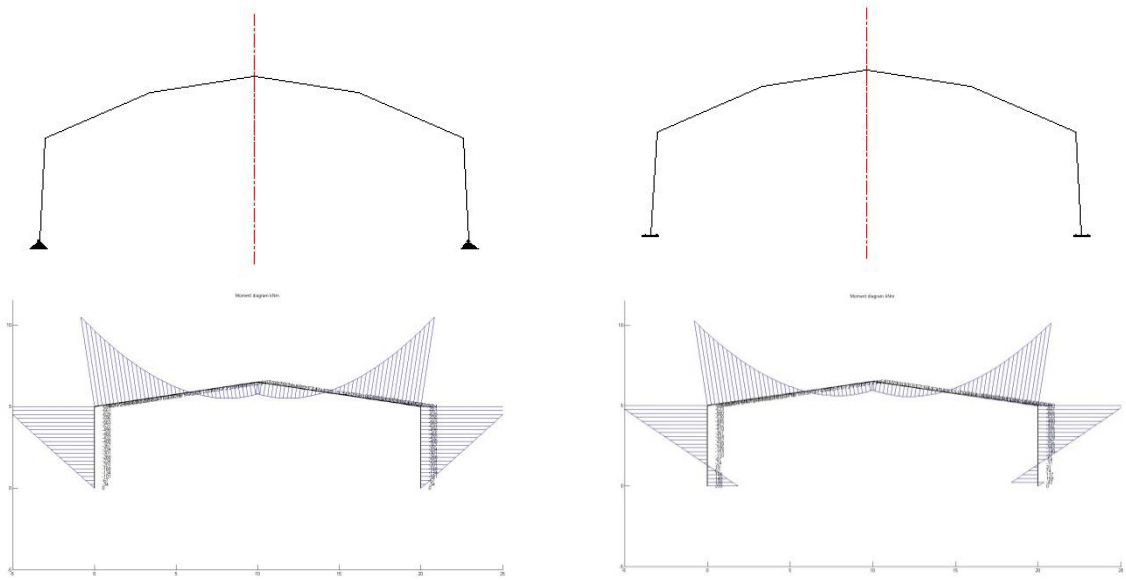


Figure 3: Moment diagram of pinned respective fixed foundation.

For easy transportation from the factory to the site, it is required to divide the frame into smaller elements usually at the eaves; and the ridges and reconnect them in form of pinned/screwed or welded connections. However, these connections are weak due to high moment at these points since the difficulty of providing high tension resistance bolts; therefore the lever arm of the bolts group in form of steel plate must be increased in order to take the stresses in a proper way (INSDAG, 2006).

There are three main types of steel portal frames, see Figure 4 below:

1. **Fixed or Rigid portal frame:** where all the connections between the frame elements are rigid and therefore the bending moment is more likely to be distributed along the frame.

2. **Two pinned portal frame:** where the supports of the frame is made to be pinned, this leads to the frame takes a bigger bending moment than the fixed one and therefore the frame has bigger sections and heavier.
3. **Three pin portal frame:** where the supports and the connection at the center line of the frame are hinged. It helps during the transportation of the frame and reduces the moment at the hinges, but at same way, the deflection increases.

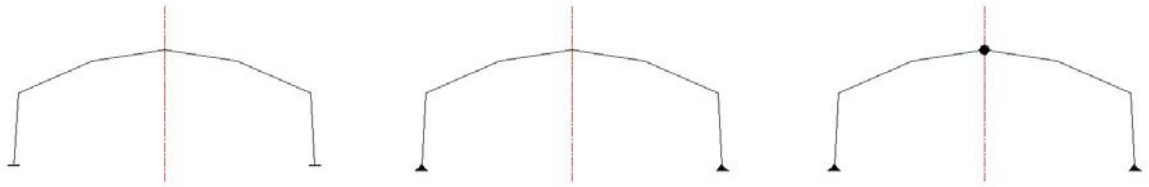


Figure 4: The three main types of steel portal frames: fixed, two pinned and three pinned portal frames.

2.1.1 Design according to Eurocode 3

2.1.1.1 General

In this project, the design is according to Eurocode 3 (EC3): Design of steel structures (CEN, 2005).The material of the frame is decided to be of steel type S235 with ultimate strength, *fy* of 235 MPa and yield strength, *fu* of 360 MPa. The elasticity modulus, *E*=210 GPa and shear modulus, *G* =81 GPa. The design in this project is done according to the Ultimate Limit State (ULS) and the Serviceability Limit State (SLS) criterion:

$$Q_{ULS}=1{,}35g+1{,}5q \quad (1)$$

$$Q_{SLS}=1{,}0g+1{,}0q \quad (2)$$

where:

Q_{ULS} is the design load according to ULS load combination.

Q_{SLS} is the design load according to SLS load combination.

g is permanent load (self-weight).

q is the variable load (snow and wind loads)

2.1.1.2 Definition of the global and the local coordinate system

In the following sections, the global and the local coordinate system is defined as shown in Figure 5 below:

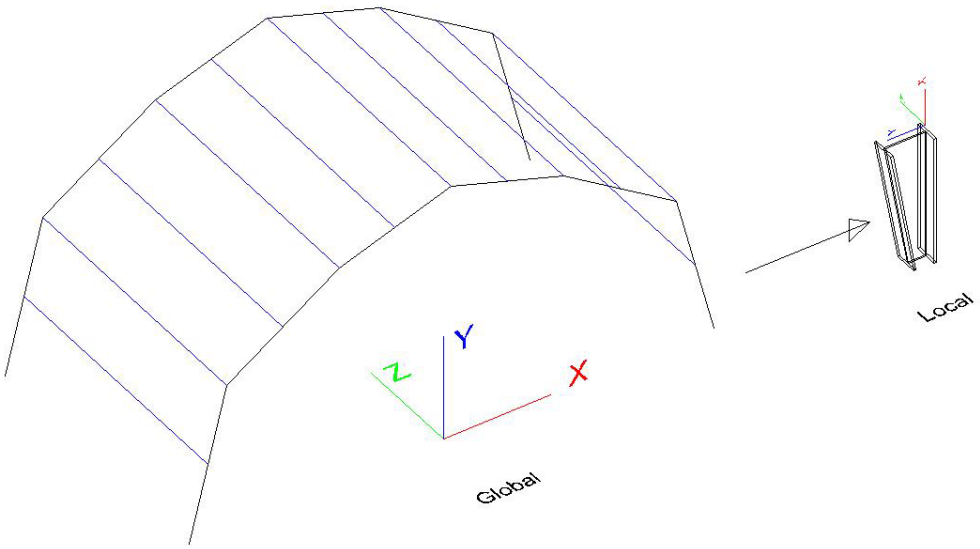


Figure 5: The definition of the global (the whole frame) and the local coordinate system (a cross section in an individual element)

2.1.1.3 Classification of cross section

According to EC3, section 5.5, the cross section is divided into four classes with plastic, elastic behavior or buckling depending on the slenderness of plates of the web and flanges (Greiner, et al., 2011).

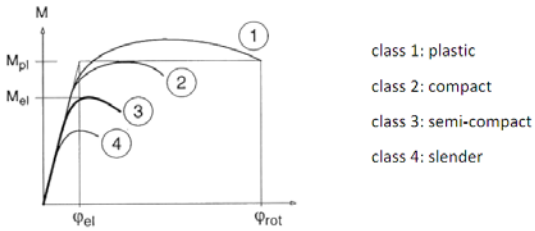


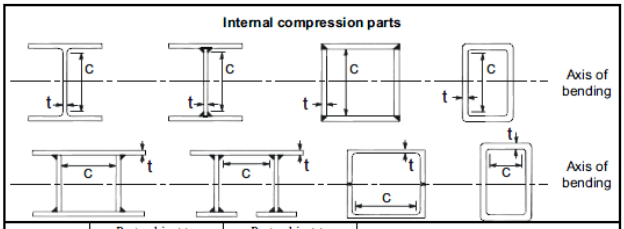
Figure 6: Classes 1 and 2 have plastic behavior while classes 3 and 4 have elastic behavior.

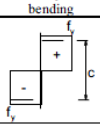
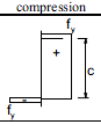
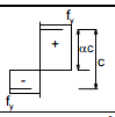
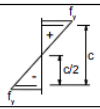
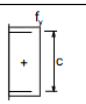
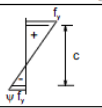
The cross section in Class 4 is slender and buckling occurs before reaching the yield stress, therefore the cross section should be reduced and only effective area should be used. In this project the sections of class 4 are not taken into account and only sections with class 1, 2 and 3 will be used.

The procedure of the classification for the web and the flange is according Figure 7 below (table 5.2, EC3). Classification procedure for class 1, 2 and 3 depending of the slenderness of the plates, c/t ratio (the plate length/ the thickness of the plate) compared with $\boldsymbol{\varepsilon = \sqrt{(235/fy)}}$.

Table 5.2 (sheet 1 of 3): Maximum width-to-thickness ratios for compression parts

Internal compression parts



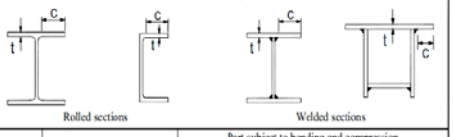
Class	Part subject to bending	Part subject to compression	Part subject to bending and compression
Stress distribution in parts (compression positive)			
1	$c/t \le 72\varepsilon$	$c/t \le 33\varepsilon$	when $\alpha > 0{,}5$: $c/t \le \frac{396\varepsilon}{13\alpha - 1}$ when $\alpha \le 0{,}5$: $c/t \le \frac{36\varepsilon}{\alpha}$
2	$c/t \le 83\varepsilon$	$c/t \le 38\varepsilon$	when $\alpha > 0{,}5$: $c/t \le \frac{456\varepsilon}{13\alpha - 1}$ when $\alpha \le 0{,}5$: $c/t \le \frac{41{,}5\varepsilon}{\alpha}$
Stress distribution in parts (compression positive)			
3	$c/t \le 124\varepsilon$	$c/t \le 42\varepsilon$	when $\psi > -1$: $c/t \le \frac{42\varepsilon}{0{,}67 + 0{,}33\psi}$ when $\psi \le -1$*) : $c/t \le 62\varepsilon(1-\psi)\sqrt{(-\psi)}$

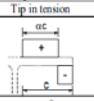
$\varepsilon = \sqrt{235/f_y}$	f_y	235	275	355	420	460
	ε	1,00	0,92	0,81	0,75	0,71

*) $\psi \le -1$ applies where either the compression stress $\sigma \le f_y$ or the tensile strain $\varepsilon_y > f_y/E$

Table 5.2 (sheet 2 of 3): Maximum width-to-thickness ratios for compression parts

Outstand flanges



Class	Part subject to compression	Part subject to bending and compression	
		Tip in compression	Tip in tension
Stress distribution in parts (compression positive)			
1	$c/t \le 9\varepsilon$	$c/t \le \frac{9\varepsilon}{\alpha}$	$c/t \le \frac{9\varepsilon}{\alpha\sqrt{\alpha}}$
2	$c/t \le 10\varepsilon$	$c/t \le \frac{10\varepsilon}{\alpha}$	$c/t \le \frac{10\varepsilon}{\alpha\sqrt{\alpha}}$
Stress distribution in parts (compression positive)			
3	$c/t \le 14\varepsilon$	$c/t \le 21\varepsilon\sqrt{k_\sigma}$ For k_σ see EN 1993-1-5	

$\varepsilon = \sqrt{235/f_y}$	f_y	235	275	355	420	460
	ε	1,00	0,92	0,81	0,75	0,71

Figure 7: Classification of the web and the flange according to table 5.2 (sheet 1 and 2 of 3, the left and the right figure above respectively).

2.1.1.4 Cross section capacity

2.1.1.4.1 Axial capacity

According to EC3 section 6.2.4:

$$\frac{N_{Ed}}{N_{c,Rd}} \le 1{,}0 \qquad (3)$$

Where:

N_{Ed} is the design value of the compression force.

$N_{c,Rd} = \frac{Af_y}{\gamma_{M0}}$, the design resistance of the cross-section for class section 1, 2 and 3.

γ_{M0} is partial safety factor defined in the National Annex. The recommended value is 1,0.

2.1.1.4.2 Bending moment capacity

According to EC3 section 6.2.5:

$$\frac{M_{Ed}}{M_{c,Rd}} \leq 1{,}0 \tag{4}$$

Where:

$M_{c,Rd}$ is the bending moment design resistance of the cross-section.

$M_{c,Rd} = \frac{W_{pl} f_y}{\gamma_{M0}}$ is the design resistance of the cross-section for class 1 and 2.

$M_{c,Rd} = \frac{W_{el} f_y}{\gamma_{M0}}$ is the design resistance of the cross-section for class 3.

$M_{c,Rd} = \frac{W_{eff} f_y}{\gamma_{M0}}$ is the design resistance of the cross-section for class 4.

M_{Ed} is the design value of the bending moment.

W_{pl}, W_{el} and W_{eff} are the plastic-, the elastic- and the effective sectional modulus respectively.

2.1.1.4.3 Shear capacity

According to EC3, section 6.2.6:

$$\frac{V_{Ed}}{V_{c,Rd}} \leq 0{,}5 \tag{5}$$

Where:

$V_{c,Rd}$ is the shear design resistance of the cross-section.

$V_{pl,Rd} = \frac{A_v (f_y/\sqrt{3})}{\gamma_{M0}}$ is the design plastic shear resistance.

$A_{v=\eta \sum(h_w t_w)}$, for welded I sections when the load is parallel to the flanges.

h_w is the hight of web.

t_w is the thickness of the web.

η ,see EN 1993-1-5 but can also be taken as 1,0 as conservative value.

It is possible that the designed shear force, V_{Ed} is larger than the half of the shear design resistance $V_{c,Rd}$ but for the simplification of the problem and to avoid the eventual reduction of the moment resistance when V_{Ed} is larger than 0,5 $V_{c,Rd}$, the limitation of the shear force is assumed as above.

2.1.1.5 Combination of axial force and bending

The combination of axial force and bending is introduced by the following formula (Trahair, et al., 2007):

$$\left(\frac{N_{Ed}}{N_{c,Rd}}\right)+\left(\frac{M_{Ed}}{M_{c,Rd}}\right)\leq 1{,}0 \qquad (6)$$

In this study, the shear force is always limited to $V_{c,Ed}\leq 0{,}5\,V_{c,Rd}$, therefore the interaction of the shear with the moment and the axial force is not considered.

2.1.1.6 Imperfection for global analysis of frames

According to EC3, section 5.3.2, for frames which are sensitive to buckling in a sway mode, an equivalent imperfection as initial imperfection has to be taken in account in form of horizontal point forces acting on the upper edges of the both columns of the frame, see Figure 8. The direction of the point loads depend on the sign of the normal forces at the columns, but it doesn't have any effect on the results since the frame is symmetric and both loads are acting at the same direction. Using of imperfection method is like substitute the check of buckling in major axis (y-axis) because the imperfection method already includes buckling about the major axis according to the second order theory (CEN, 2005).

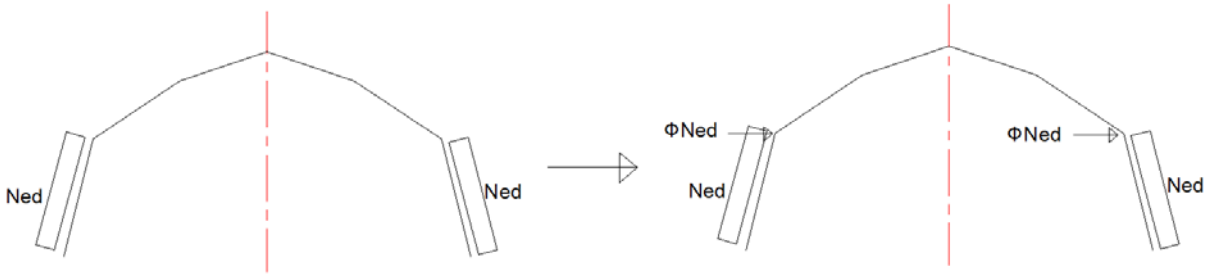


Figure 8: The equivalent forces of the initial imperfection.

The global initial imperfections of sway, Φ:

$$\Phi=\varphi_0\alpha_h\alpha_m \qquad (7)$$

Where:

φ_0 is basic value = 1/200.

α_h is the reduction factor for height of the frame, h:

$$\alpha_h=\frac{2}{\sqrt{h}}\ \text{but}\ \frac{2}{3}\leq\alpha_h\leq 1{,}0 \qquad (8)$$

h is the height of the frame.

α_m is the reduction factor for the number of columns in a row:

$$\alpha_m = \sqrt{0{,}5\left(1 + \frac{1}{m}\right)} \tag{9}$$

m is the number of columns in a row.

The bow imperfections are not considered in this study, since their effect is expected to be less significant.

2.1.1.7 Buckling resistance of members

2.1.1.7.1 Buckling resistance in z-direction (out of plane)

According to EC3 section 6.3.1, for compressed members the following requirements should be fulfilled:

$$\frac{N_{Ed}}{N_{b,Rd}} \leq 1{,}0 \tag{10}$$

Where:

N_{Ed} is the design value of the compression force.

$N_{b,Rd}$ is the buckling resistance of the compression force.

$N_{b,Rd} = \frac{\chi_z A f_y}{\gamma_{M1}}$ for bending class 1, 2 and 3.

χ_z is reduction factor for the relevant buckling mode.

$$\chi_z = \frac{1}{\phi + \sqrt{\phi^2 - \overline{\lambda^2}}} \tag{11}$$

$$\phi = 0.5\left[1 + \alpha\left(\overline{\lambda} - 0{,}2\right) + \overline{\lambda}^2\right] \tag{12}$$

α is an imperfection factor according to EC3, Table 6.1 which depends on the buckling curves, EC3 figure 6.4.

$\overline{\lambda}$ is the non-dimensional slenderness.

$$\overline{\lambda} = \sqrt{\frac{Af_y}{N_{cr}}} \tag{13}$$

$$N_{cr} = \frac{\pi^2 E I_z}{(\beta l)^2} \tag{14}$$

I_z is the moment of inertia about z-axis.

β is the Euler length factor, assumed to be =1,0 in case of buckling about z-axis (the minor axis).

l is length between two lateral bracings.

If $\bar{\lambda} \leq 0{,}2$ or $\frac{N_{Ed}}{N_{cr}} \leq 0{,}04$ then the buckling effects may be ignored.

2.1.1.7.2 Resistance against lateral torsional buckling

According to EC3 section 6.3.2, the following requirements should be fulfilled:

$$\frac{M_{Ed}}{M_{b,Rd}} \leq 1{,}0 \tag{15}$$

Where:

M_{Ed} is the design value of the moment.

$M_{b,Rd}$ is the design buckling resistance moment.

$M_{b,Rd} = \frac{\chi_{Lt} W_y f_y}{\gamma_{M1}}$

Where

W_y is the appropriate section modulus as follows:

$W_y = W_{pl,y} = A_c y_{c +} A_t y_t$ for bending class 1 or 2 cross-sections

$W_y = W_{el,y} = \frac{I_y}{y}$ for bending class 3 cross-sections

$W_y = W_{eff,y}$ for bending class 4 cross-sections

χ_{LT} is the reduction factor for the relevant buckling mode.

$$\chi_{LT} = \frac{1}{\phi_{LT} + \sqrt{{\phi_{LT}}^2 - \overline{{\lambda_{LT}}^2}}} \leq 1{,}0 \tag{16}$$

$$\phi_{LT} = 0.5\left[1 + \alpha_{LT}\left(\overline{\lambda}_{LT} - 0{,}2\right) + \overline{\lambda_{LT}}^2\right] \tag{17}$$

$$\overline{\lambda_{LT}} = \sqrt{\frac{W f_y}{M_{cr}}} \tag{18}$$

α_{LT} is an imperfection factor according to EC3, Table 6.3 which depends on the buckling curves.

M_{cr} is the elastic critical moment for lateral-torsional buckling for mono-symmetric prismatic beams (Salvadori, 1955). It should be mentioned that the critical moment for a tapered beam is much more complicated.

$$M_{cr} = \frac{\pi^2}{(kl)^2}\sqrt{\frac{I_w}{I_z} + \frac{GI_t(kl)^2}{\pi^2 EI_z}} \tag{19}$$

Where:

I_z is moment of inertia about the minor axis.

I_t is the warping torsional constant

$$I_t = \sum \frac{b_i t_i^3}{3} \tag{20}$$

I_w is St. Venants torsion constant of the cross section

$$I_w = (1 - \beta_f)\beta_f I_z h_s^2 \tag{21}$$

Where:

$$\beta f = \frac{I_{fc}}{(I_{fc} + I_{ft})} \tag{22}$$

I_{fc}, I_{ft} is the moment of inertia of the compression and tension flanges, respectively, about the minor axis of the entire section.

G is shear modulus = 81 GPa

kl is effective length between points of restrains against buckling. k is assumed to be 1,0 in the current study.

The compressed flanges are the sensitive parts of the element that should be checked due to the lateral torsional buckling. The effective length (kl) is the length between two lateral bracings. For the external flanges, the purlins and the wall rails provide a “natural” lateral bracing against buckling. Figure 9 below is showing that in case of negative moment, the internal bracings are important against the risk of lateral torsional buckling for the lower flanges. While on the contrary, the external bracings are important in case of positive moment to prevent the lateral torsional buckling for the upper flanges in addition to their job of setting the roof and the walls on the outer side of the frame.

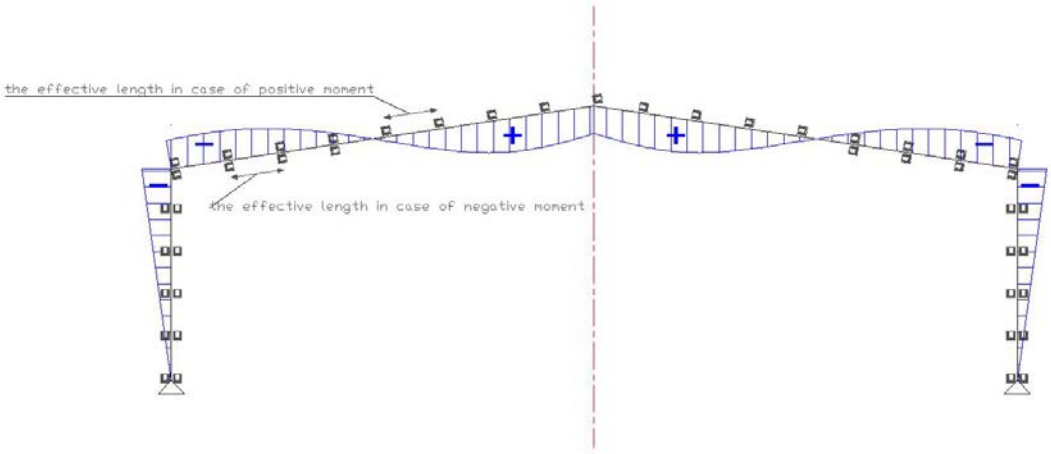


Figure 9: The distribution of the bracings according to sign of the moment diagram. Theoretically, the internal bracings are useless in case of positive moment since the compressed flanges which is susceptible for the lateral torsional buckling is at the external side and vice versa.

To prevent the risk of the lateral torsional buckling for the non-braced internal flanges in case of negative moment, a special kind of bracing is used, see Figure 10. In case of positive moment, the external flanges between two purlins are susceptible for the lateral torsional buckling and therefore the cross section should be checked; while in case of negative moment, the internal flanges (the compressed flange) between two torsional restraints are susceptible.

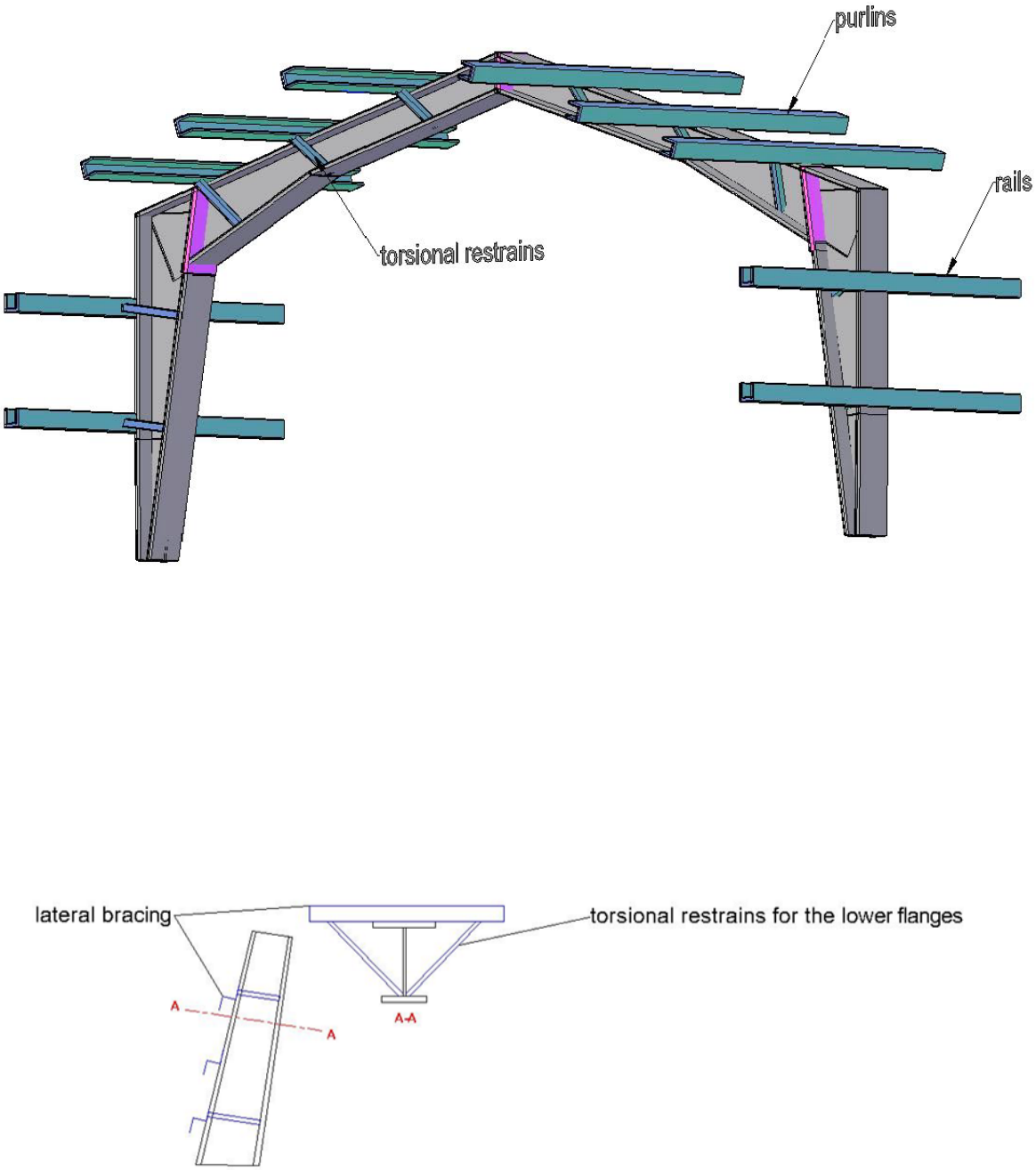


Figure 10: Lateral bracings for the upper flanges and torsional restrains for the lower flanges.

2.1.1.8 Interaction flexural and lateral torsional buckling

According to EC3, section 6.3.3, members which are loaded with a combination of compression and bending should be checked by interaction formulas for both the major and the minor axis:

- **Interaction formula 1:**

$$\frac{N_{Ed}}{\frac{\chi_y N_{Rk}}{\gamma_{M1}}} + k_{yy}\frac{M_{Ed}}{\frac{\chi_{LT} M_{y,Rk}}{\gamma_{M1}}} \leq 1 \qquad (23)$$

- **Interaction formula 2:**

$$\frac{N_{Ed}}{\frac{\chi_z N_{Rk}}{\gamma_{M1}}} + k_{zy}\frac{M_{Ed}}{\frac{\chi_{LT} M_{y,Rk}}{\gamma_{M1}}} \leq 1 \qquad (24)$$

Where:

χ_y and are the reduction factors due to the flexural buckling, see section 3.1.1.7.1 for χ_z.
χ_y is assumed to be equal to 1,0 as it is already included in the imperfection calculations. It means that there is no reduction due to the flexural buckling in the major direction.

χ_{LT} is the reduction factor due to lateral torsional buckling, see section 3.1.1.7.2.

k_{yy} k_{zy} are interaction factors according to method 2, Annex B, EC3.

From now on, the main interaction formulas are going to be abbreviated as follows since they are going to be used often in the next chapters:

The description of the interaction formula (IF)	**The number of the equation in this project**	**The abbreviation**
IF of the share force	5	Shear check
IF of the combination of the axial and the bending	6	IF0
Interaction formula 1	23	IF1
Interaction formula 2	24	IF2

Table 1: The abbreviation of the interaction formulas.

2.2 Structural optimization

2.2.1 General

Structural optimization means finding the best structure that can transfer a certain load in the space to a fix support. The best structure means use of material in the most economical way,

i.e. finding as less costs and as light structure as possible that can sustain the load, see Figure 11.

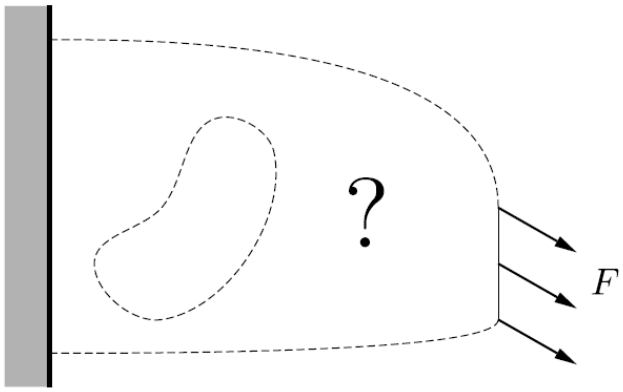


Figure 11: finding the best structure that can transfer load F from a point in the space to a fix support (Christensen, et al., 2008).

However, the optimal structure doesn't necessary refer to the lightest weight; it can also refer to other parameters, e.g. being less sensitive to buckling or being as stiff as possible. Since the structure should fulfill some structural, manufacturing or other type of requirements, it is necessary to introduce different constraints in the optimization problems. The constrains can be displacements, stresses or geometry. The evaluation of a certain objective like the cost as an example is done by the *objective function* and a set of other measures as constrains. The structural optimization formula often consists of the following functions and variables (Christensen, et al., 2008):

- ***Objective function (f):*** refers to the goodness of the preferred design, which is wanted to be as minimal/maximal as possible. If it is maximum, then the variables may be the stiffness or the resistance against the buckling, while in the other hand if it is minimum, then the variables can be the costs, the displacement or the weight of the designed structure.
- ***Design variable (x):*** describes the variables of the design, which are changed during the optimization process in an iterative way. These variables can be for example the cross section of the elements or the coordinates of the nodes etc.
- ***State variable (y):*** describes the behavioral response of the structure. For the mechanical aspect, the response of the structure under a certain load condition can be the displacement, the internal forces or the strain.

The demonstration below explains the idea:

$$(Structural\ Optimization)\begin{cases} minimize\ f(x,y) with\ response\ to\ x\ and\ y \\ subject\ to \begin{cases} behavioral\ constrains\ on\ y \\ design\ constrains\ on\ x \\ stability\ constrains \end{cases} \end{cases}$$

2.2.2 Genetic algorithm (GA)

In this project an optimization Matlab-function is going to be used. This function called "Genetic Algorithm" (MathWorks, 2011).

Genetic algorithm (GA) is an optimization method based on the evolution of species. GA became popular today due to the necessity of finding a fast and an effective method to solve wide types of searching and optimization problems. The classical optimization methods have some weakness that they are often stuck and never escape from the local optima. That is because most of these methods based on hill climbing technique which depends on the position of the starting point and direction of searching, see Figure 12 below (Etzkorn, 2011). How fast and how accurate the processing of finding the optimum depends on the choice of the initial solution. It is common that traditional methods are used to solve problems with continuous rather than discrete variables, because it is often based on function-derivation techniques that required continuous functions (Deb, 1997).

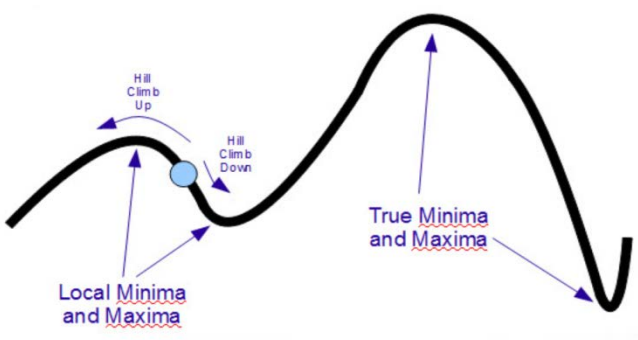


Figure 12: Hill climbing technique of searching. Finding of the true optima is depending on the position of the start point and the direction of searching (Etzkorn, 2011).

One of the most important inspirations for GA is Darwin's "survival-of-the-fittest" principle for the natural evolution. According to this principle an offspring with better traits than its siblings has more chance to survive at the same environment, while a weaker one has less chance and it may be eliminated from the population. A simple example by the well-known biologist Richard Dawkins (Dawkins, 1976), that the tall trees in the mountains were shorter at the beginning of the evolutionary process. The taller offspring had more opportunity to survive since they could reach the sun light and rain better than the short ones. Therefore these taller trees could generate more offspring that held the same genes and this led to that the shorter trees gradually eliminated, see Figure 13 below.

Figure 13: Tall trees survive better than short in the mountains' environment.

GA is an iteration procedure; it handles a group of solutions in every iteration cycle instead of a single one and tries to find the fit among them. Every group of solutions is called population and consists of random solutions. Every individual in the population is in form of a coding string of constant length. At every iteration, the population is updated through three important operators: *selection*, *crossover* and *mutation* and a new and fitter generation created, see Figure 14 below. The GA procedure ends when it reaches a certain number of generations or when no more improvement of populations can be obtained (Deb, 1997).

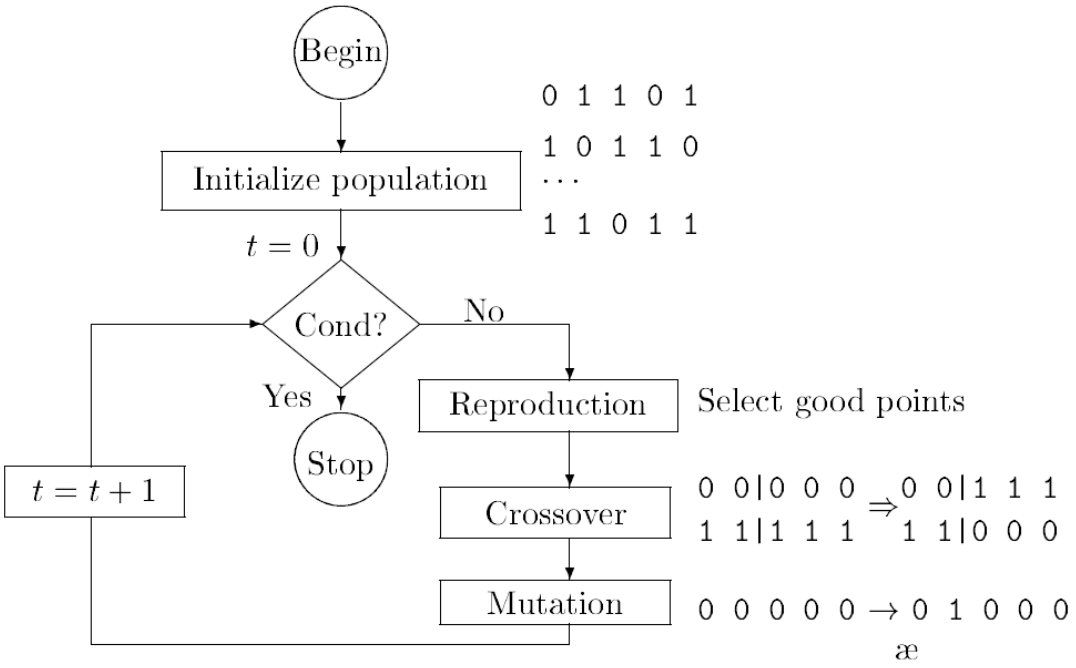


Figure 14: flow chart that explains GA process (Deb, 1997).

2.2.2.1 The representation of GA-chromosomes

Since the concept of GA is similar to natural evolution, GA consists of some terminologies that are derived from natural selection. The genetic information are stored in the *chromosomes* and every chromosome is divided into several portions called *genes* (Sivanandam & Deepa, 2008), while GA-chromosomes are in form of binary-strings which consist of *N* number of substrings. Every substring x_i has a certain length l_i and refers to a certain variable of the problem:

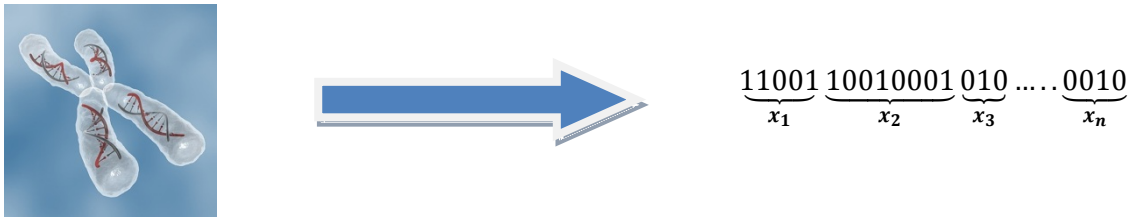


Figure 15: An illustration showing the concept of GA from the chromosomes.

This means that every string has 2^{l_i} alternatives, for example a string of 5 digits has 2^5=3125 alternatives. The upper boundary of an alternative x_i^{max} is (11..111) and the lower boundary x_i^{min} is (00..000). The length of a substring depends also on its precision, for example if the required precision is 3 digits, then the length of the substring is $(x_i^{max} - x_i^{min})/0.001$ or according to:

$$l_i = log_2\left(\frac{(x_i^{max} - x_i^{min})}{\epsilon_i}\right) \quad (25)$$

Where:

ϵ_i is the desired precision.

The length of the string is simply the summation of the substrings (genes) (Deb, 1997).

The total genetic information stored in a string called *genotype*, while *phenotype* is the final appearance of the individual. The procedure of decoding/translating the properties saved in genotype to phenotype called *genotype-phenotype mapping* (Sivanandam & Deepa, 2008).

2.2.2.2 Selection

Selection is the first operation to be applied to the population. It is simply based on the idea of picking the good individuals and put them in a *mating-pool*. The essential idea of the selection is picking an *above-average* string, duplicate it and insert it again in the mating-pool. One of the reproduction operations is called *ranking* selection scheme. In this method, the whole population is ranked in an ascending order, see Figure 16 below. The worst individual gets the lowest rank, which is 1, while in the other hand the best individual takes the highest rank which is *N*. The best string will be reproduced into two copies, while the worse will not be copied. Another simple type of reproduction operator is called *tournament* selection scheme. In this method two strings are randomly chosen to tournament and the best of these two is selected. Thereafter this best gets two copies which are inserted in the mating-pool (Deb, 1997).

Figure 16: A demonstration of ranking selection scheme.

2.2.2.3 Crossover

The essential idea of crossover method is based on randomly – but based on selection methods – taking two strings (parents) from the mating-pool, cutting them in a certain places of the strings and finally switching the cut-portions between the two strings. There are several methods of crossover operators:

- **Single-point crossover:** two string-parents have to be taken, cutting them in an arbitrary place and the right cut portion of both string to be switched to create new offspring:

$$\underset{Parents}{\left\{\begin{matrix}1111111 \mid 111\\ 0000000 \mid 000\end{matrix}\right\}} \rightarrow \underset{Offsprings}{\left\{\begin{matrix}1111111000\\ 0000000111\end{matrix}\right\}}$$

- **Two-point crossover:** almost the same idea as in the single-point crossover, but now it is two cuts instead to ensure more exchange between the parents:

$$\underset{Parents}{\begin{Bmatrix}0000000 \mid 111 \mid 00\\ 1111111 \mid 000 \mid 11\end{Bmatrix}} \rightarrow \underset{Offsprings}{\begin{Bmatrix}1111111000011\\ 0000000111100\end{Bmatrix}}$$

- **Multi-point crossover:** the parent-strings have to be cut in several places and the sliced portions to be exchanged between them:

$$\underset{Parents}{\begin{Bmatrix}1111111111\\ 0000000000\end{Bmatrix}} \rightarrow \underset{Offsprings}{\begin{Bmatrix}1001100011\\ 0110011100\end{Bmatrix}}$$

We can see from the procedure of the crossover operator that the chance of producing good children depends on the location of the cut and it is therefore a random process. However, the good parents will not be lost since many copies of them will be created by the next reproduction (Deb, 1997).

2.2.2.4 Mutation

Mutation is one of the breeding cycles that ensure more variety of strings and prevent GA from trapping in a local optimum. It is also useful way to recover the irreversible loss of the good genes. There are some mutation techniques:

- **Interchanging:** a bit of a string is selected randomly to be interchanged:

$$\{0000000000\} \rightarrow \{0001000000\}$$

- **Reversing:** a bit of a string is selected randomly and the next bit to the right has to be reversed with the selected bit:

$$\{0010000000\} \rightarrow \{0001000000\}$$

Mutation probability (P_m) decided how often a string will be mutated. If P_m=0 means no mutation occurs, elsewise if P_m=100 %, the whole chromosome will be changed. It is important to keep the mutation probability low as there is risk that GA may change to random search operator instead (Sivanandam & Deepa, 2008).

2.2.2.5 The penalty function

When the constraints are violated, a numerical penalty is assigned to the offspring to reduce its fitness as a punishment which makes it less likely to be selected. It may even lead to the elimination from the mating-pool. Imagine a frame that has been created during the processes above (selection, crossover and mutation). The maximum vertical deformation of this frame is limited to L/250 which should not be exceeded. In this case, if the limits violated, a numerical penalty should be added – as a punishment – to the weight of the frame to make it heavier and definitely unacceptable. The numerical penalty is set to be linearly proportional with respect to the magnitude of the violation.

3 Computer program / proposed algorithm

3.1 General

The purpose of the current algorithm is to find the optimum design of a symmetric frame consisting of 6 elements (2 tapered columns and 4 tapered beams, see Figure 17 below).

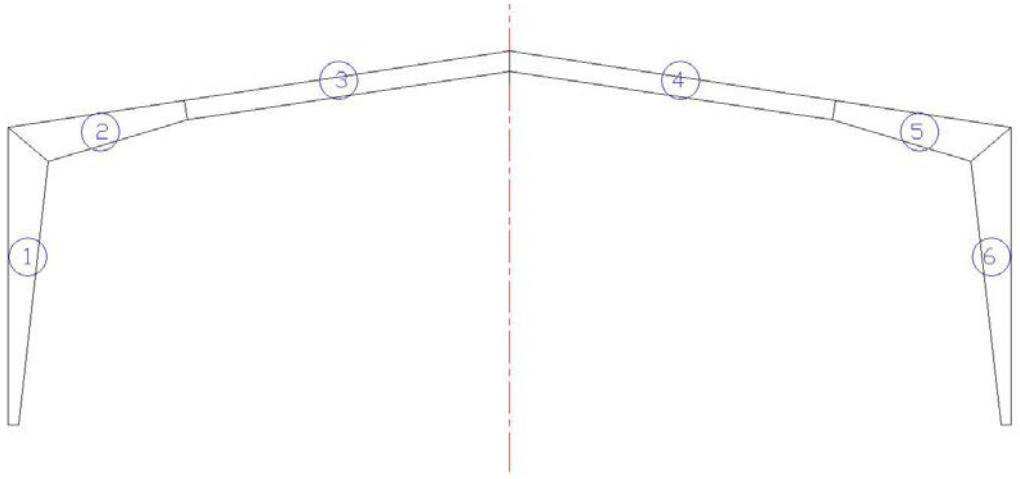


Figure 17: The studied frame in this project consist of 6 tapered elements.

3.2 Input values

3.2.1 The material properties

The material properties of the frame are set as default as steel type S235. The elasticity modulus, E=210 GPa and shear modulus, G= 81 GPa. The properties can be changed by the user.

3.2.2 The loads

The loads are the essential inputs of the algorithm as in principle the function is going to find the optimum shape of the frame for a given loading. The user is free to input vertical loads (snow loads) and horizontal loads (wind loads) over the 6 elements as shown in Figure 18. The self-weight of the frame is calculated automatically and be added to the vertical loads. The loads are uniformly distributed.

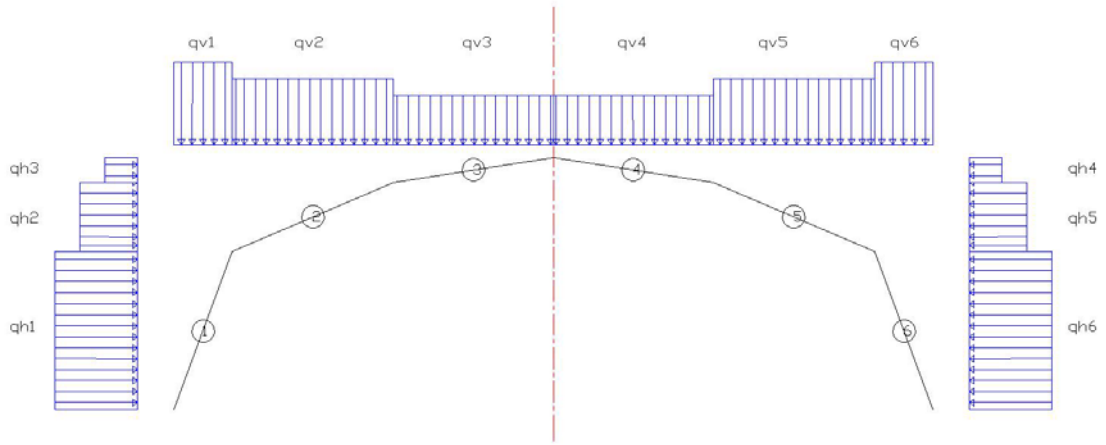


Figure 18: Vertical and horizontal loads acting on the elements.

3.2.3 The main coordinates

As the input number of the main elements of the frames is 6, it means that we have 6+1=7 coordinates to be inputted, see section 4.2.1 and see Figure 19 below.

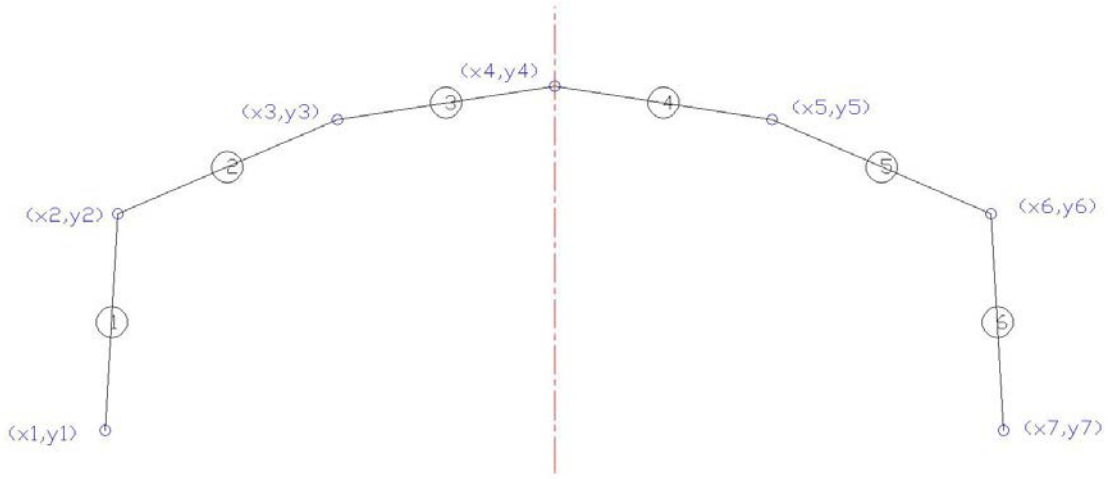


Figure 19: The main elements and the main coordinates.

3.2.4 The limits of the dimensions of the cross section

As the frame is designed to be consisting of 6 main elements, the user has to input the dimensions limitations of both ends of each of these elements. It means that the user has to specify 6x2 cross sections for the whole structure. It is optional to input the dimensions as constants or as variables within certain limitation and letting the algorithm to find the best choice see Figure 20 below.

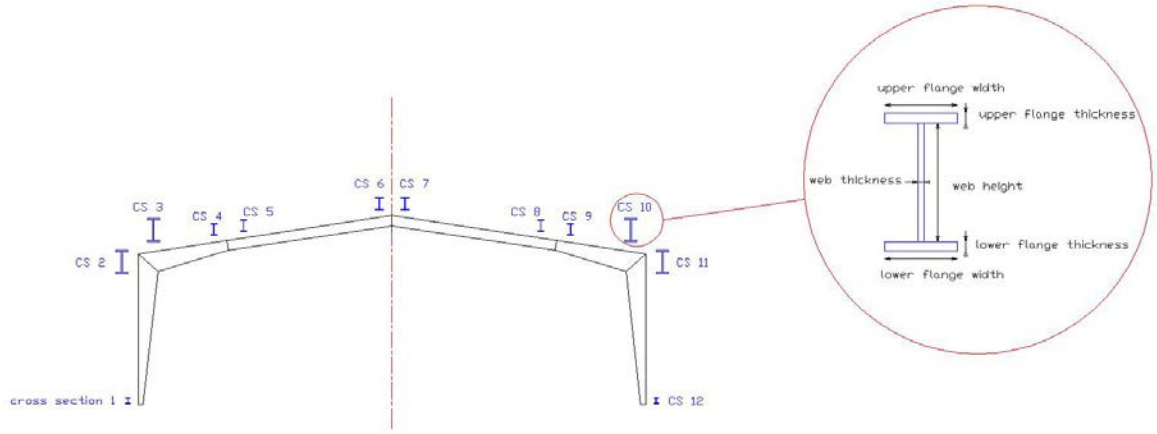


Figure 20: The cross sections of the main elements.

3.2.5 The bracings

In order to ensure a stable frame with respect to buckling, two kinds of bracings have to be identified; External bracings against lateral torsional buckling for the compressed external flanges and internal bracings against lateral torsional buckling for the compressed internal flanges, see section 2.1.1.7. Existing of any of these kinds of bracings is enough to prevent the flexural buckling as well. The number and the position of the bracings to be defined by the user as inputs. The optimum number and the position of the bracings can be found by the algorithm as well. In the frame analyses step with respect to the finite element method, the main elements are divided into subelements. The number of the subelements depends on the number of the bracings in the main elements. For example, if we have 6 bracings in a certain element, the algorithm generates 5 subelements between the bracings, see Figure 21 below. The precision of the analysis depends on the number of the subelements. With more subelements, the analysis is more precise.

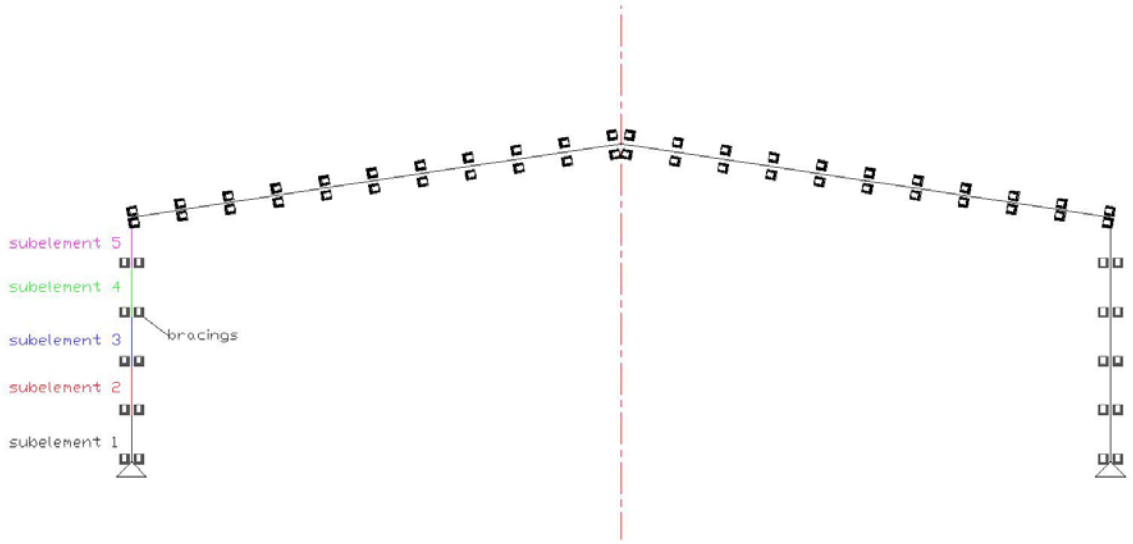


Figure 21: The main elements are divided into subelements depending on the number of the bracings.

It is decided to consider the main nodes of the frame as braced at both flanges even if it is not defined as input. The costs of the material, the welding costs and the weight of the bracings are not considered in this project.

3.2.6 The type of the supports

Usually there are three types of supports (depending on the number of degree of freedoms) that can be used in such frames:

- **Fixed**: This is restricted against the horizontal, the vertical and the rotational movement.
- **Pinned**: This is restricted against the horizontal and the vertical movement.
- **Roller**: This is restricted only against the vertical movement.

In this project it is free for the user to use one of these three types, however it is common to use either the fixed or the pinned, but it is also recommended to use pinned foundation in both sides of the frame as it is more easier to maintain in comparison to the fixed one.

3.3 Constraints

Some constraints have to be taken into account regarding the final design of the frame with regard to the outputs of the algorithm:

3.3.1 Constraint 1: Initial coordinates (geometry)

The main coordinates of the frame have to be specified as an input. These coordinates refer to main nodes of the frame and they lie at the center line of the cross sections. The final shape of the frame will be created within this geometry.

3.3.2 Constraint 2: Cross section dimensions

The user should set the limitation of the cross section dimensions. These dimensions have maximum and minimum limitation and the algorithm will find the optimum between these

limitations. For example the height of the web is set as a variable and let the algorithm to "play" with it; however it may set as a constant and set other dimensions as variables instead, see section 3.2.4.

3.3.3 Constraint 3: Frame displacement

According to the Serviceability Limit State (SLS), the maximum allowable horizontal deformation is limited to $L/250$, while the maximum allowable horizontal deformation is limited to $L/150$, where L is the span width of the frame.

3.3.4 Constraint 4: The stresses

The stresses are limited to the EC-criterions mentioned in section 2.1.1.

3.4 Matlab functions

3.4.1 General

This project consist in principle of three main processes, see Figure 22:

1. Analyzing the frame due to a certain set of loads with help of CALFEM-toolbox (Austrell, et al., 2004) which means finding deformation, axial force, shear force and bending moment diagrams.
2. Checking the capacity of the frame based on the analyses above according to the EC3 constrains and according to ULS and SLS criterion.
3. Finding the optimal design of the frame using the GA-function.

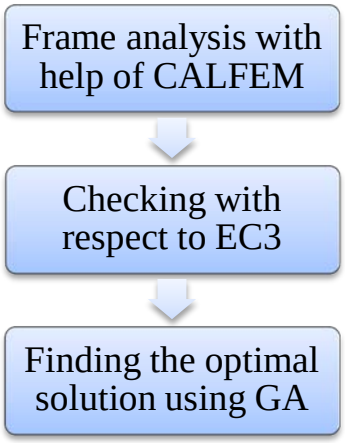


Figure 22: The three main processes during the project.

3.4.2 Calculation of the internal forces and the displacements

A function is developed with help of CALFEM-toolbox (Austrell, et al., 2004) using second order theory to plot the internal forces diagrams (the axial forces-, the shear forces- and the bending moment diagram) and the deformed shape of the frame exposed to a certain load and of the whole frame (see Figure 23).

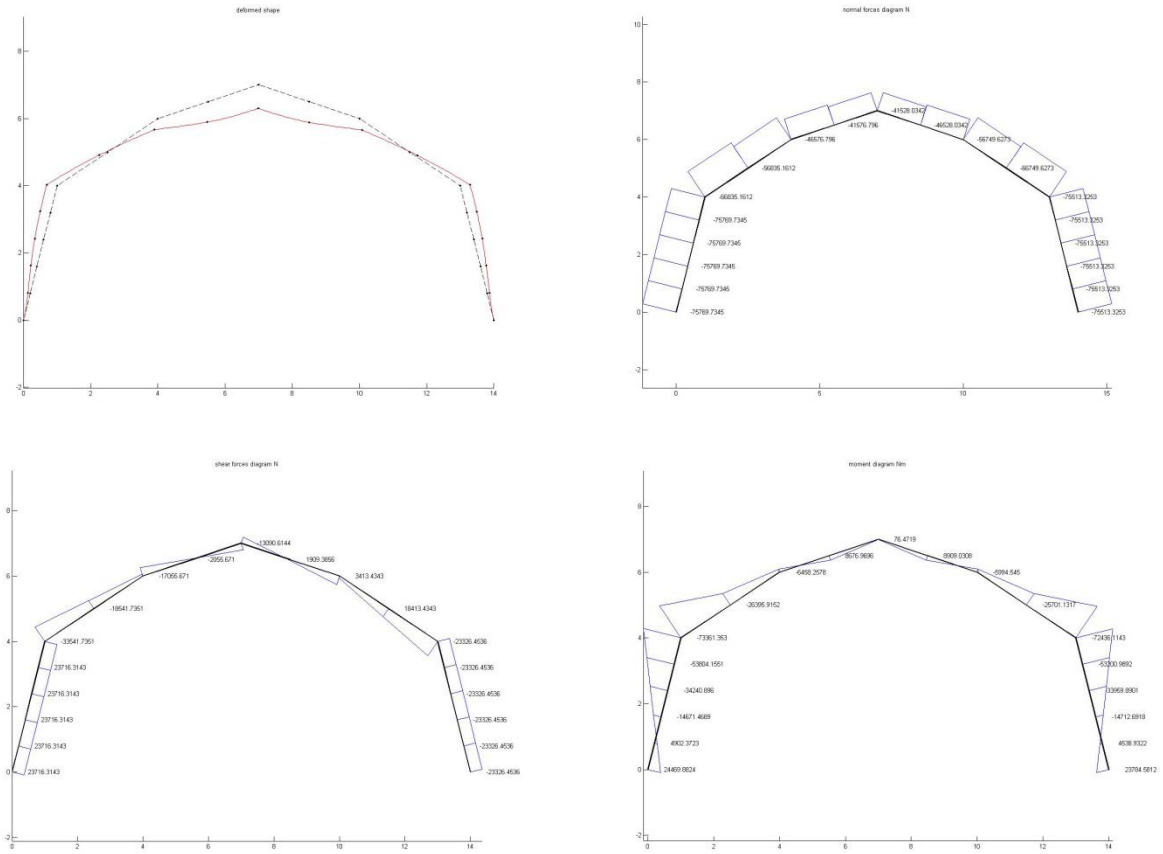


Figure 23: the deformed shape, the axial force, the shear force and the bending moment diagram.

3.4.3 Frame checking according to Eurocode 3

Checking the frame according to EC3, see section 3.1.1 with respect to some important aspects:

- Finding the classification of the cross section.
- Checking the capacity of the cross section.
- Checking against flexural buckling.
- Checking against lateral torsional buckling.
- Calculating the utilization of the frame elements using the interaction formulas given in equation 4, 5, 22 and 23 (shear check, IF0, IF1 and IF2), see Figure 24 below.
- Checking the horizontal and the vertical deformation of the frame with respect to SLS.

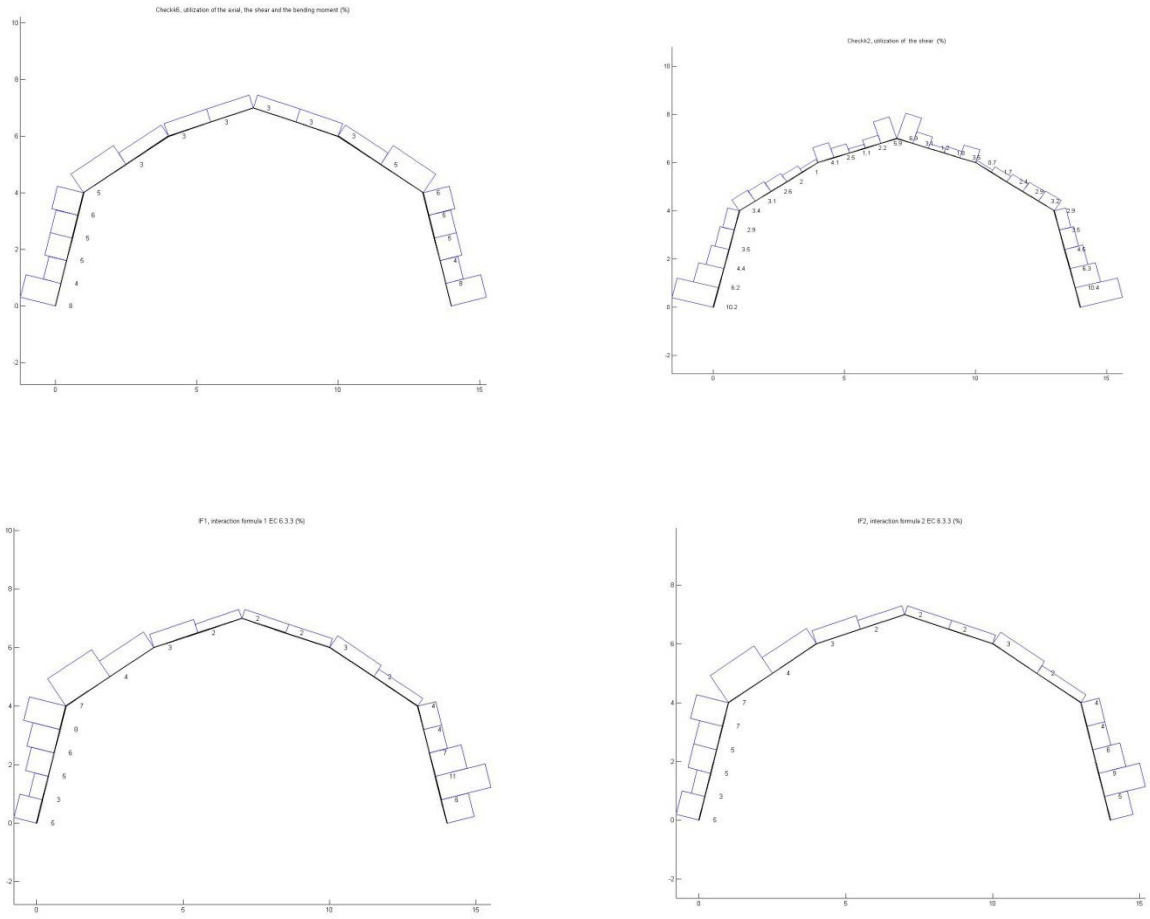


Figure 24: The utilization of the frame elements according to the interaction formulas (Shear check, IF0, IF1, IF2).

3.4.4 GA-Module

After analyzing the frame using CALFEM and checking it with respect to EC3, the last step is finding the optimal solution, i.e. finding a feasible frame with minimum weight with help of the "Global Optimization Toolbox" (MathWorks, 2011). The GA provides a wide range of options to control the selection, the crossover and the mutation operators to set that help finding the shortest way to the optimal solution.

4 Case study

The following flow chart explains the whole process of the function from the inputs to the final design:

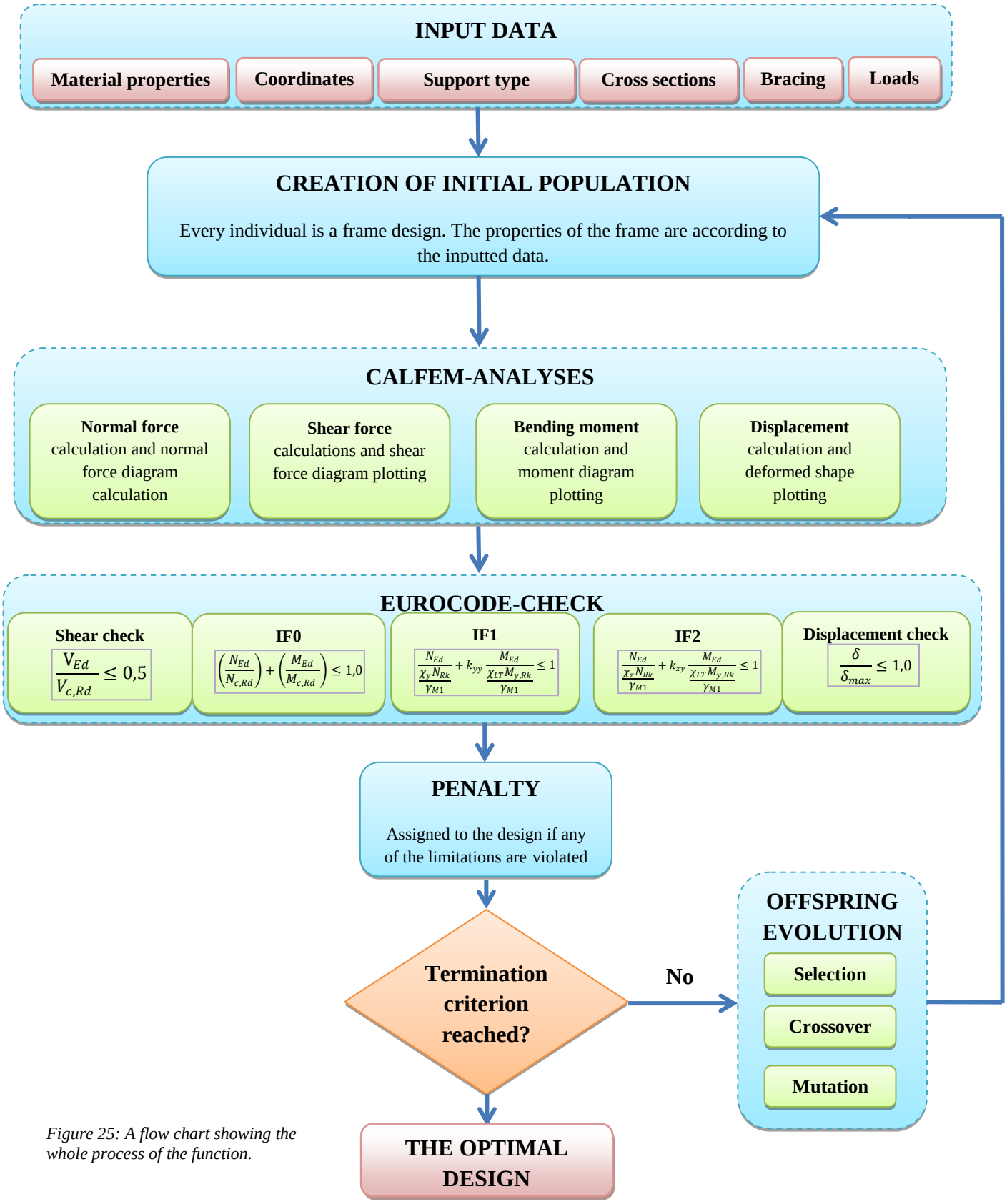


Figure 25: A flow chart showing the whole process of the function.

For further explanation of the function, a frame of 20 m span width and 6.5 m height is studied, see Figure 26 below. The slop of the roofs is 15 %.

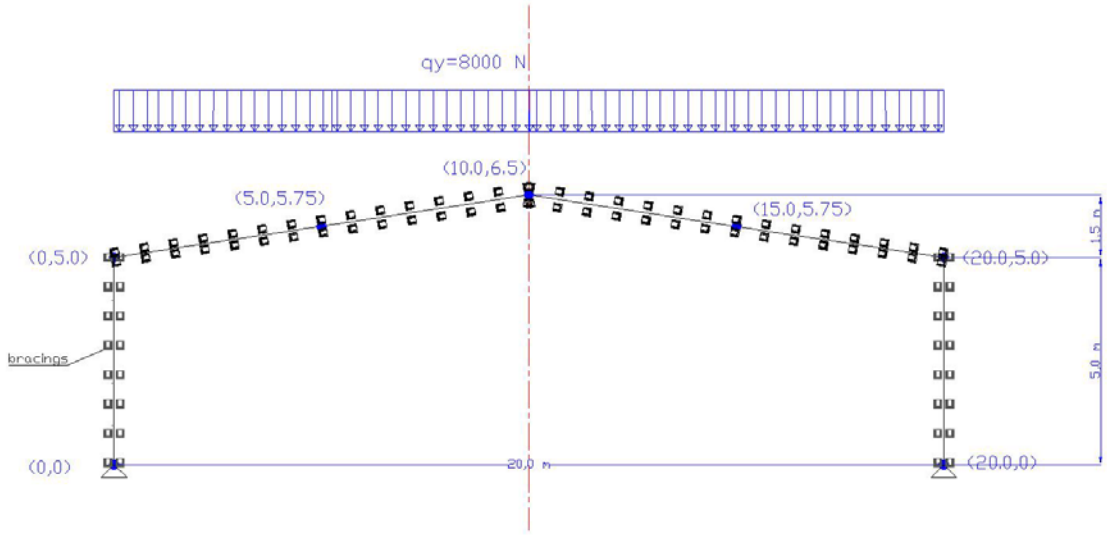


Figure 26: A frame of 20m span width and height of 6.5m is studied.

The inputs have to be defined by the user are:

- **The material of the frame:** the material is decided to be of steel type S235 with elasticity modulus *E* of 210 GPa and shear modulus *G* of 81 GPa.
- **The coordinates (the geometry) of the frame:** since the frame is decided to be consist of 6 main elements (see Figure 24), 7 two dimensional coordinates to be defined by the user (see table 2 below). In this case study, the frame is decided to be symmetric, but it is also possible to input unsymmetrical frame.

X (m)	Y (m)
0	0
0.5	0
5.0	5.75
10.0	6.5
15.0	5.75
20.0	5.0
20.0	0

Table 2: The coordinates of the frame.

- **The boundary conditions:** the supports are decided to be pinned which means free rotation and fixed against the movement at x- and y-axels. Pinned foundation means zero-moment at the supports.
- **The cross sections:** the frame consists of 6 main elements and therefore there are 12 cross sections to be specified. Every cross section has I-shape which consists of the upper flange, the web and the lower flange. The user has to input the width and the thickness of the upper and the lower flange and the height and the thickness of the web in mm-units.

In this example, the height of the webs at the both haunches are decided to be variables between 200-800 mm (one variable of one side is enough as the frame is symmetric). This is to test the algorithm to find the optimal design within these limitations. The other dimensions are decided to be constants according to the table below:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	80	25	200	25
2	200	25	(200-800)	25	200	25
3	200	25	(200-800)	25	200	25
4	200	25	60	25	200	25
5	200	25	60	25	200	25
6	200	25	60	25	200	25
7	200	25	60	25	200	25
8	200	25	60	25	200	25
9	200	25	60	25	200	25
10	200	25	(200-800)	25	200	25
11	200	25	(200-800)	25	200	25
12	200	25	80	25	200	25

Table 3: The dimensions of the cross sections at the main nodes.

- **The bracings:** the frame in this example is decided to be fully braced (both the external and the internal bracings) to reduce the buckling risk. The distance between two bracings is about 0.2 m.
- **The loads:** uniformly distributed loads of 8 kN/m applied vertically (as snow loads) on the rafters (roofs) of the frame.

After inputting the required data, the GA-function will generate the initial population. The generated population is in form of binary strings to facilitate the searching process done by GA. The binary strings will be translated by GA as ordinary variables to be handled in the next processes (CALFEM-analyses and EC-check module). The initial population is generated randomly at the first generation, but they will be developed using the GA-functions: selection, crossover and mutation at the next generations. Every individual of this population is a frame design with properties according to inputted data. If some of the inputs are decided to be variables which can be the dimensions of the cross section or the bracings distribution pattern, each individual is going to have dimensions/pattern within the inputted maximum and the minimum limitations. The number of the population and generations should be specified by the user. The bigger population is, the easier for GA to find the optimal design among them, but longer searching time. The number of population in this example is set to be 500.

The frame designs that have been created by GA (the individuals) are analyzed using CALFEM-toolbox with respect to the inputted loads. The analyses are geometrically non-linear and include calculation and plotting the diagrams of the internal forces such as the normal force, the shear force and the bending moment. The CALFEM-functions are used called `beam2g` for the calculation of stiffness matrix (K-matrix) and `beam2gs` for the non-linear calculations of the internal forces. It also includes the calculation of the displacement and plotting the deformed shape of the frame.
Due to the risk of buckling in a sway mode, an equivalent imperfection as initial imperfection has to be taken into account in form of horizontal point loads acting on the upper edges of the both columns of the frame (see section 2.1.1.6). A finite element mesh is created for every main element. Every main element is divided into a number of subelements (mesh elements). The finer the mesh is, the more accurate the analyses are, but longer computing time. In this example, the length of subelements is the distance between two sequent bracings (external or internal bracings or both). The distance of the subelement in the same main element is equal i.e. constant mesh per main element. In this example, the distance of each subelement is about 0.2 m which is also the distance between two adjacent bracings. In case of absence of bracings as we are going to see in example 2 (see section 5.2), the user have to define the size of the mesh i.e. the number of subelements.

The results of the CALFEM-analyses go throw a "filter" called Eurocode-check. This includes checking the capacity of the frame against normal force, shear force and bending moment. These checks are done by using Shear-check and IF0 (see section 2.1.1). It also includes checks against buckling risk which can refer to flexural buckling and lateral torsional buckling, IF1 includes checks against in-plan flexural buckling (buckling against y-axis) and lateral torsional buckling. IF2 includes out-of-plane flexural buckling (buckling against z-axis) and lateral torsional buckling. Finally the displacements check is done according the Serviceability Limit State, SLS-criterion. The utilization of these checks should not exceed the limitations which are 50% for the shear check and 100% for IF0, IF1, IF2 and displacement check.

If these limitations are violated, a numerical penalty is assigned to the weight of the frame to make it heavier and less likely to be selected in the next generation and gradually eliminated from the population. The amount of the penalty depends on how big the violations are. The bigger violations are, the higher the penalty is.

The population is developed/evoluted generation after generation by the three essential GA-functions: selection, crossover and mutation. In this process, the fit design is selected; copies of it are made and returned it back to the mating-pool, while heavy designs will gradually be eliminated. Hence the attitude of GA from Darwin's principle of evolution "survival of fittest". Heavier designs might be created at the initial population, by crossover and mutation process. Heavy designs can also be created in case they have been punished by the penalty function due to the violation of EC-limitations. It is an iterative process trying to develop the population every cycle/generation. In every iteration cycle, each individual of the population which is a design frame is analyzed using CALFEM-toolbox, checked by EC-check module and finally selected/eliminated by the GA-functions. The process is terminated when the terminations criterion are reached. These criterions can be the maximum number of

generation or when there are no developments in the population within a certain marginal. In this example the termination criterions are unlimited and the process has been stopped manually after about 4 minutes (about 207 generations). The function have found the solution at a relatively short time since the number of population is set to be 10 only and since it is only one variable to be found by the algorithm (only the height of the webs at the haunches), see Figure 27 below:

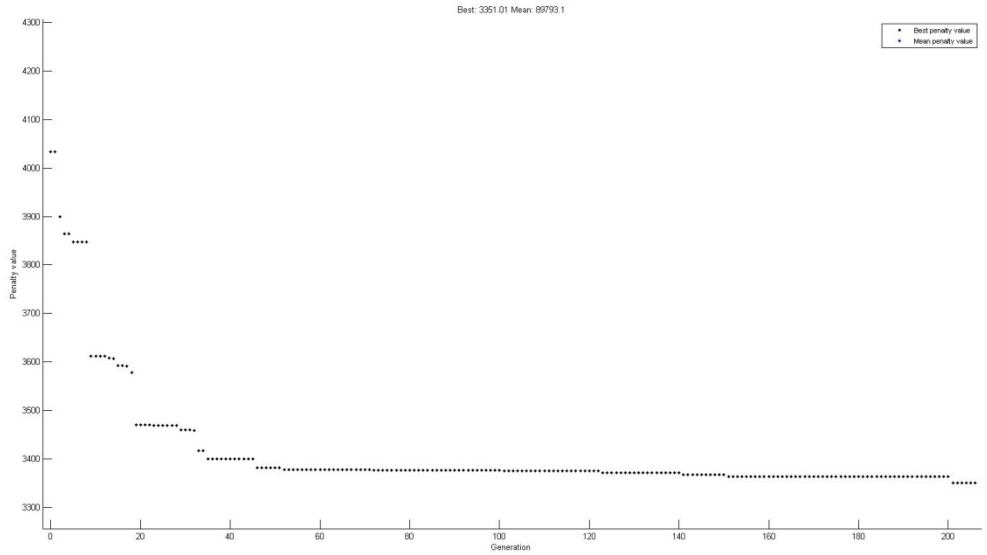


Figure 27: The fitness figure showing the development of the generations.

The process ends up by finding the optimal design which has the minimum weight and within the EC-limitations, see Figure 28 below:

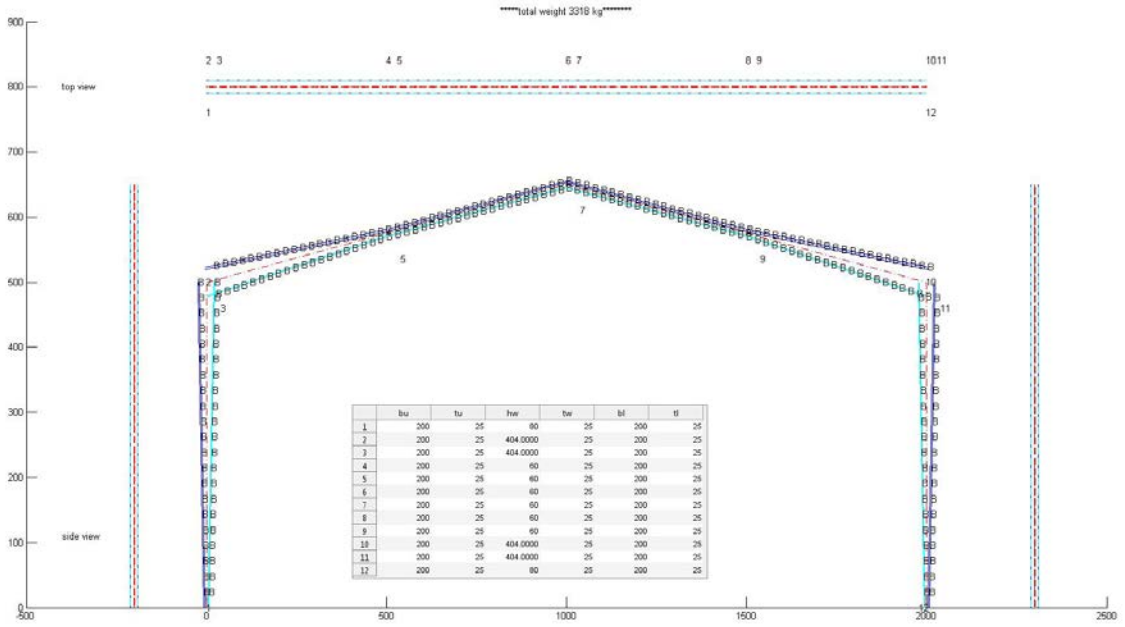


Figure 28: The optimal design the GA found.

The optimal design that GA found has total weight of 3318 kg and the optimal height of web at the haunches is 408 mm (set to be variable input between 200-800 mm at the beginning). The following table shows the results of the cross sections (the same table can be found together plotted with the optimal design above generated automatically by the function):

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	80	25	200	25
2	200	25	404 (200-800)	25	200	25
3	200	25	404 (200-800)	25	200	25
4	200	25	60	25	200	25
5	200	25	60	25	200	25
6	200	25	60	25	200	25
7	200	25	60	25	200	25
8	200	25	60	25	200	25
9	200	25	60	25	200	25
10	200	25	404 (200-800)	25	200	25
11	200	25	404 (200-800)	25	200	25
12	200	25	80	25	200	25

Table 4: The results of the dimensions of the cross sections at the haunches of the optimal design.

The following figures showing the analysis results of the optimal design that the GA has found. It doesn't mean that these analyses are done once at the end of the searching process for the optimal design only, but it is done for every individual of the population at each iteration cycle. This can explain why the searching process can take relatively long time.

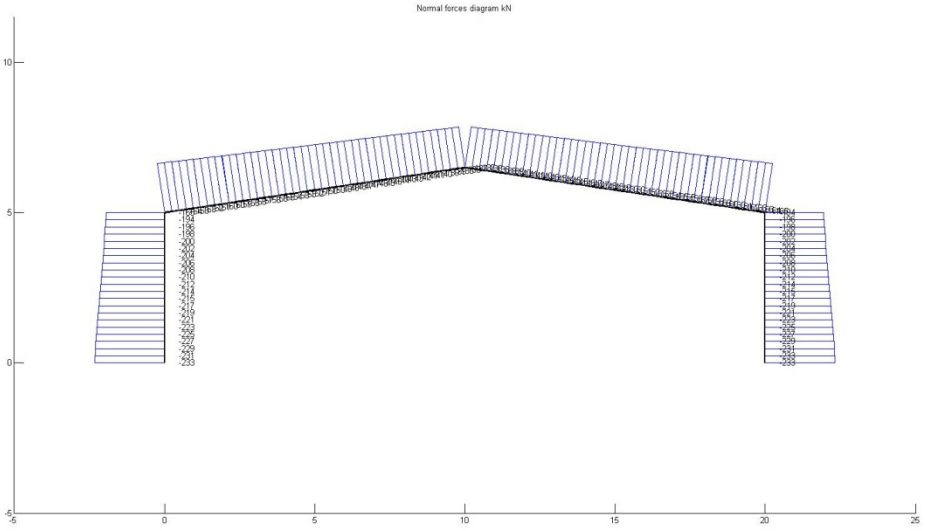


Figure 29: Normal force diagram. Max value at the support is -233 kN.

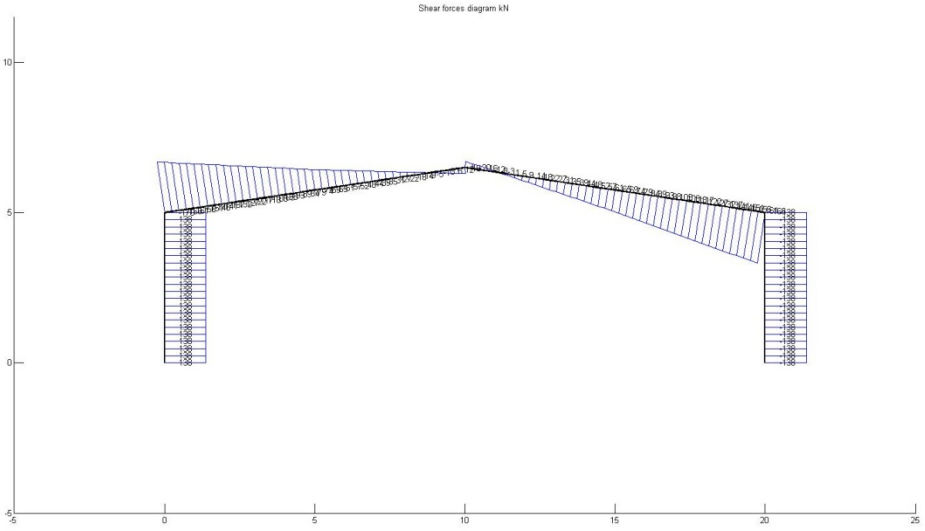


Figure 30: Shear force diagram. Max value at the haunches is -170 kN.

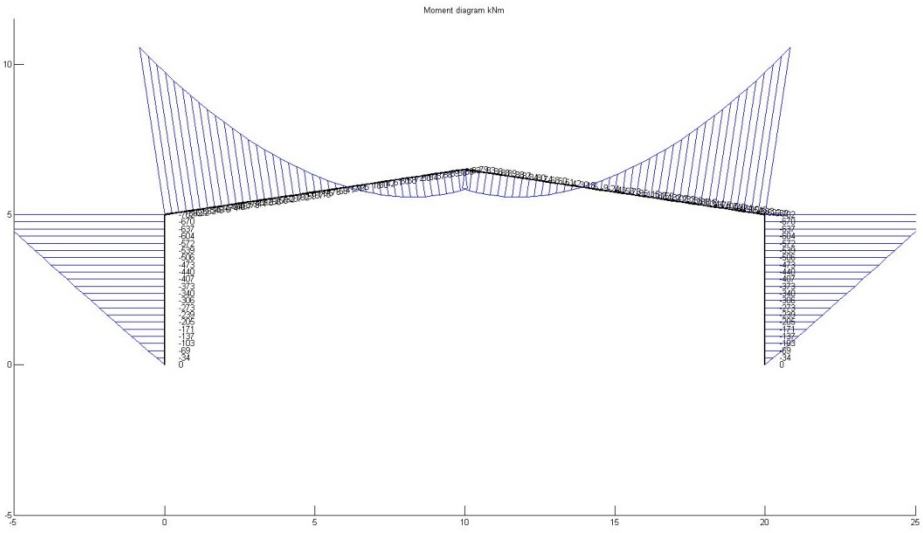


Figure 31: Moment diagram. Max value at the haunches is -702 kN.m.

Since the Serviceability Limit State, SLS-criterion is not taken into account in this example i.e. no limitation for the displacements, we get high vertical deformation of about 235 mm which is 294% of the SLS-limitation (max vertical displacement is 80 mm), see Figure 32 below:

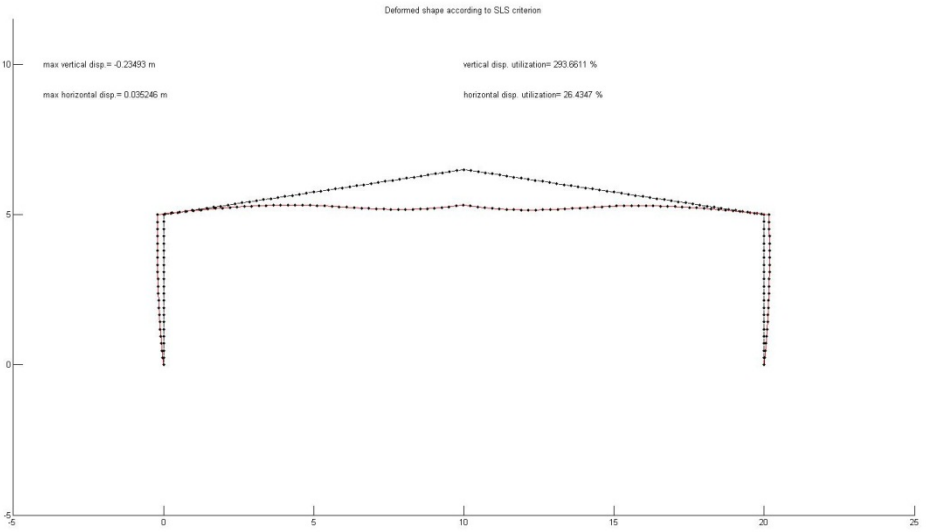


Figure 32: The deformed shape. Max. vertical displacement is -0.235 m at the middle. Max. horizontal displacement is 0.0367m at the haunches.

The following Figures are showing the results of the Eurocode-checks and utilizations for the optimal design:

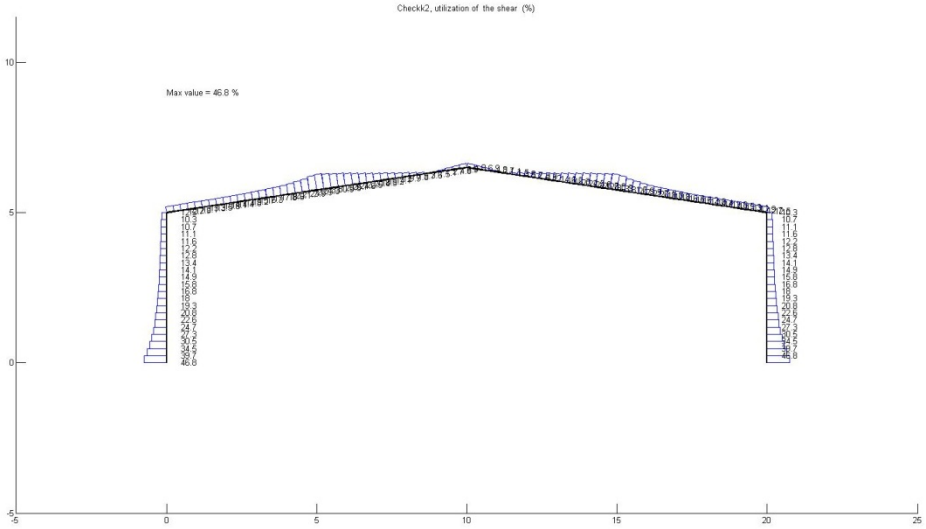


Figure 33: Shear check. Max utilization value is 46.8% at the supports.

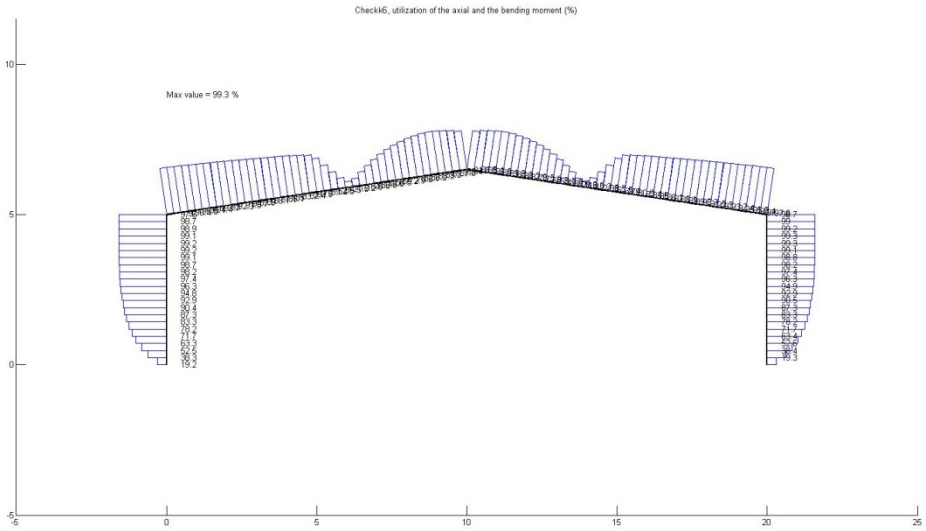


Figure 34: IF0, interaction formula of the normal force and bending moment. Max value is 99.3% at the haunches due to the high moment.

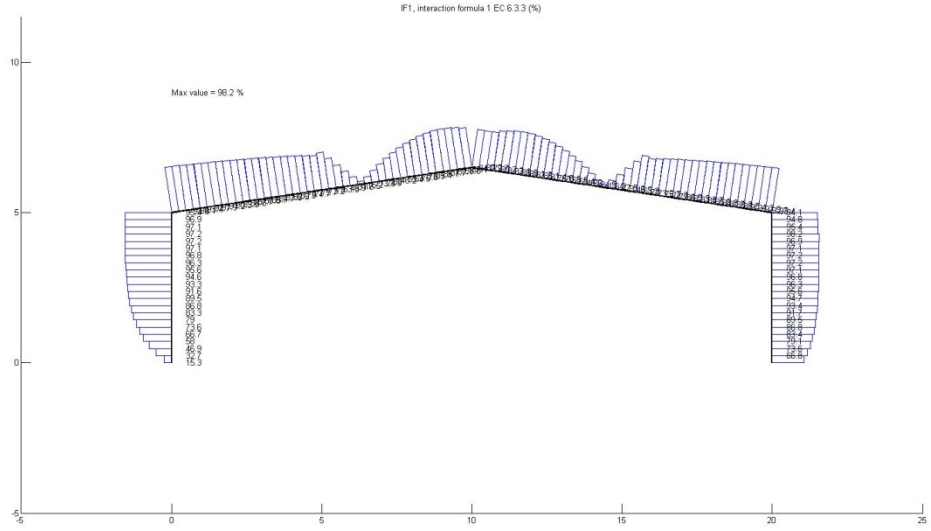


Figure 35:IF1, interaction formula that includes in-plan flexural buckling and lateral torsional buckling. Max value is 98.2 %.

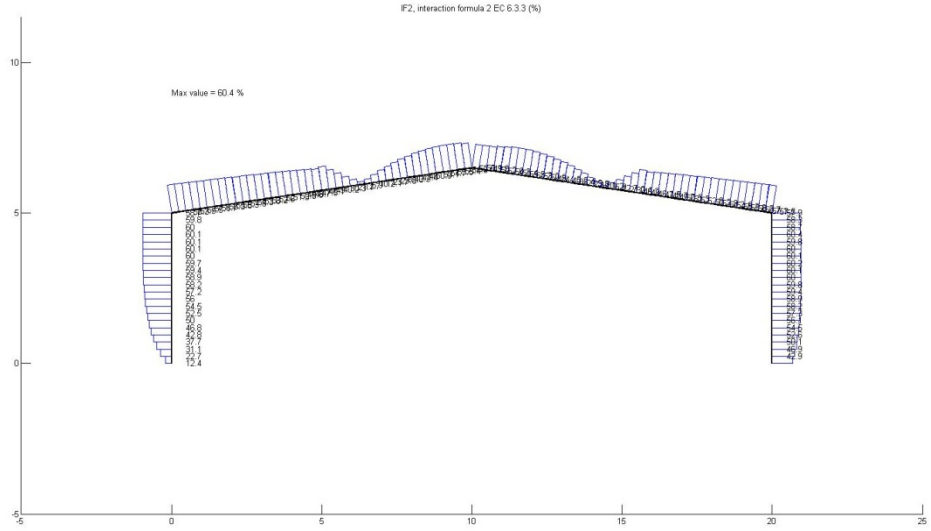


Figure 36: IF2, interaction formula that includes out-of-plan flexural buckling and lateral torsional buckling. Max value is 60.4 %.

From the utilization values, we can tell that the design we get is the optimal since the values are closed to 100% for IF0 and IF1 (max limitation is 100%) and 46.8% for the shear check (max limitation is 50%).

5 Testing examples

The following testing examples show some design results that the algorithm has found. Example 1 and 2 are designed as braced and unbraced frames respectively to see the difference in the design in case of exist/absence of the bracings. Example 3 is similar to 1 and 2, but the function here was tested to find the optimal bracings distribution pattern in addition of finding the optimal design. At the first three designs, the SLS criterion is not considered i.e. the displacement of the frame is not limited. Example 4 and 4 is similar to 1 and 2 respectively, but the SLS criterion is taken into the account.

5.1 Example 1

In this example, the optimal design of a steel portal frame is determined with respect to the weight. The supports are set to be pinned and the loads are uniformly distributed over the rafters and equal to 8000 N. The loads are according to ULS criterion. The span width of the frame is set to be 20.0 m and maximum height of 6.5 m. The distributing of the bracing pattern is as it showed in Figure 37 below, which is in about 0.22 m distance between each other, which in turn is meant to be less buckling risk in this example. The deformation of the frame is not considered in this example.

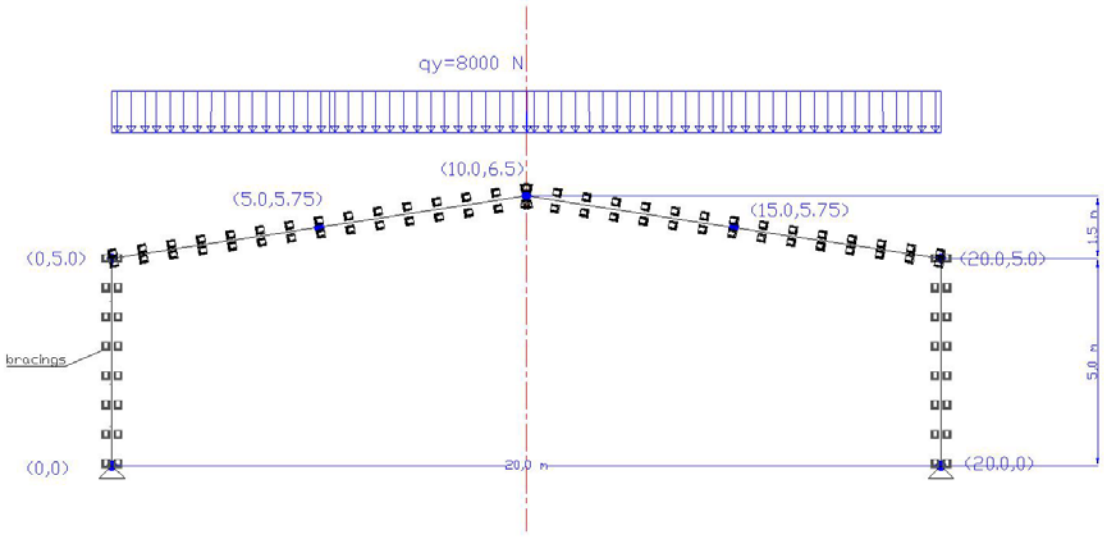


Figure 37: The initial inputs of the frame.

The heights of the webs at the main nodes of the structure are set to be variables between 50-1500 mm (see figure 38). Other dimensions such as the thickness of the web, the width and the thickness of the flanges are set to be constants. Since the structure is symmetric, only 3 parameters have to be designed during the optimization. The horizontal- and the vertical deformation are unlimited in this example, i.e. the SLS criterion is not followed.

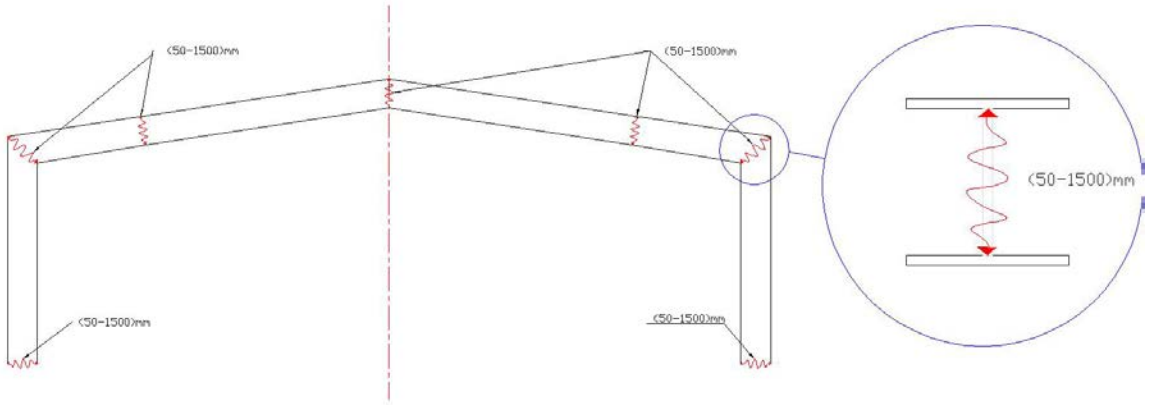


Figure 38: As input, the heights of the webs at the main nodes of the structure are decided to vary between 50-1500 mm.

The population size is set to be 500. The algorithm has run for about 42 hours. The figure below shows the development of the fitness value during the generations. It shows that the algorithm has found the solution earlier. It is noticed that there is no development/change in the generations after about 150 generation which means that the elapsed time for finding the solution is about 9 hours only.

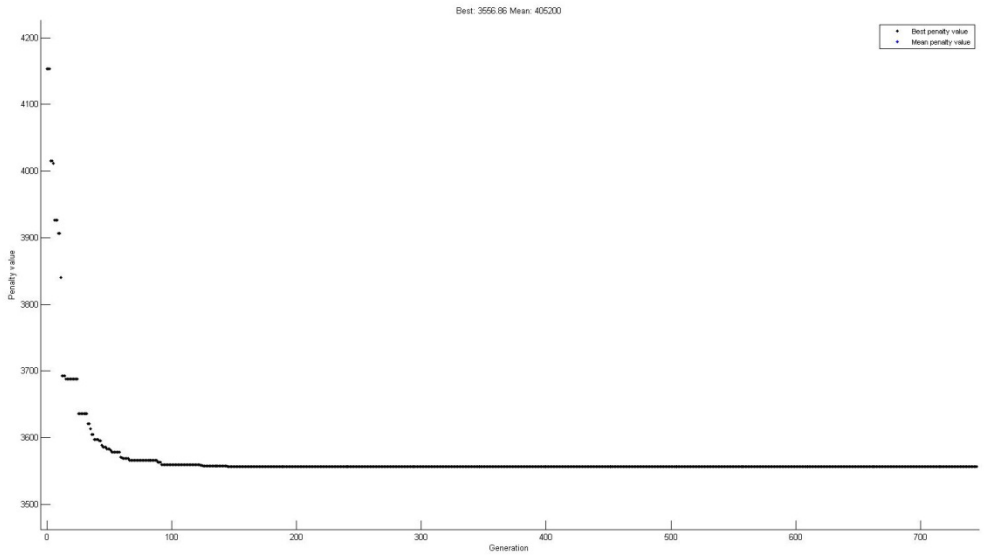


Figure 39: The fitness figure which is showing the development of the generation.

The algorithm found a frame with total weight of 3294 kg as shown in Figure 40 below.

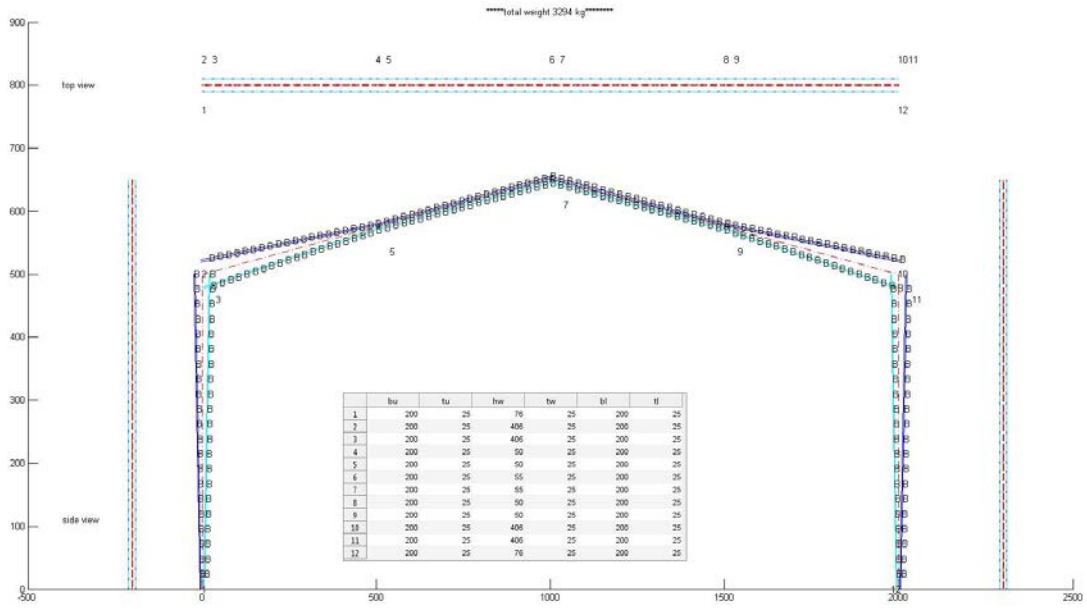


Figure 40: The optimum shape of the frame that the GA has found. The letter B in the frame refers to the internal and the external bracings. The table shows the dimensions of each cross section of the frames' main elements.

The following table shows the dimensions of the cross sections at the edges of frame's main elements, see section 3.2.4. The shape of the frame seems feasible since the web height at the supports are small where the moment is equal to zero since it is pinned. It is also noticeable that the frame has high webs at the launches to bear the high moment. The height of the web at the centerline is also small since the moment value is also small, see Figure 41:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	76 (50-1500)	25	200	25
2	200	25	409 (50-1500)	25	200	25
3	200	25	409 (50-1500)	25	200	25
4	200	25	50 (50-1500)	25	200	25
5	200	25	50 (50-1500)	25	200	25
6	200	25	55 (50-1500)	25	200	25
7	200	25	55 (50-1500)	25	200	25
8	200	25	50 (50-1500)	25	200	25
9	200	25	50 (50-1500)	25	200	25
10	200	25	409 (50-1500)	25	200	25
11	200	25	409 (50-1500)	25	200	25
12	200	25	76 (50-1500)	25	200	25

Table 5: The dimensions of the cross section of the frame's main elements. It is obvious to mention that all dimensions are constants, except the height of the webs (which are marked in different color in the table) which are set to be as variables between 50-1500 mm and letting the algorithm to find the optimal solution within this interval.

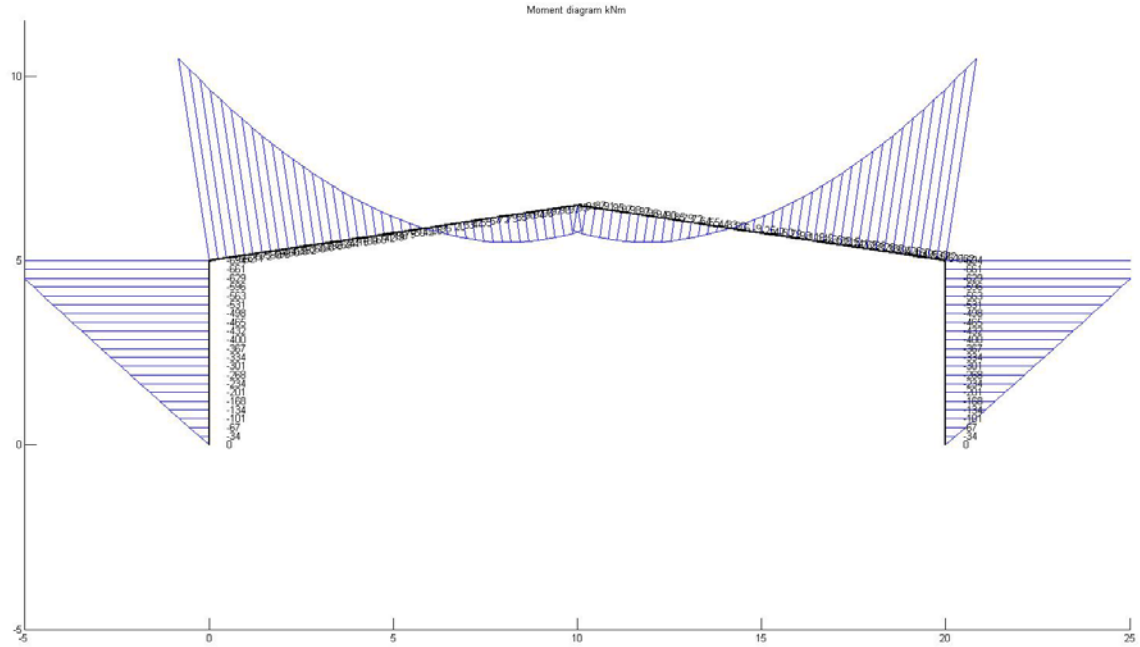


Figure 41: The moment diagram.

It seems that it is the optimal shape since we get maximum utilization of 99.2% of interaction formula of the axial and the bending (IF0) (see Figure 43) and 97.8% at the interaction formula 1 (IF1), see Figure 44. It is obvious that the maximum utilization that can be reached

is 100%, otherwise in case of being exceeded, it is going to be punished by the penalty function and going gradually to eliminated.

The maximum utilization of the Shear check, IF0, IF1 and IF2 are 49 %, 99.2%, 97.8%, and 60.2% respectively. The following figures showing the results of the utilizations:

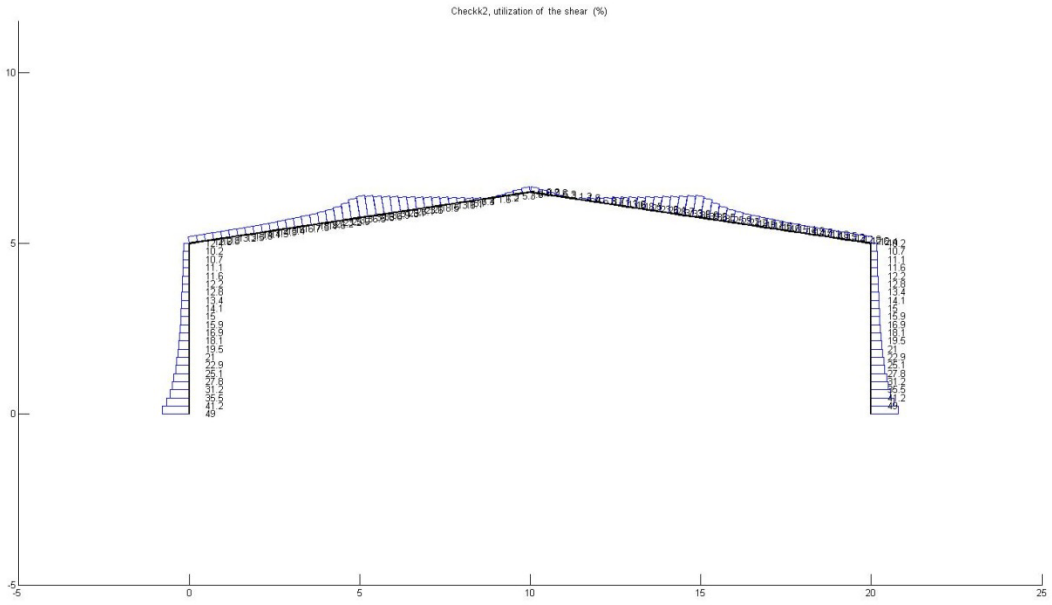


Figure 42: The utilization of shear (Shear check)(eq. 4, section 2.1.1.4.3).

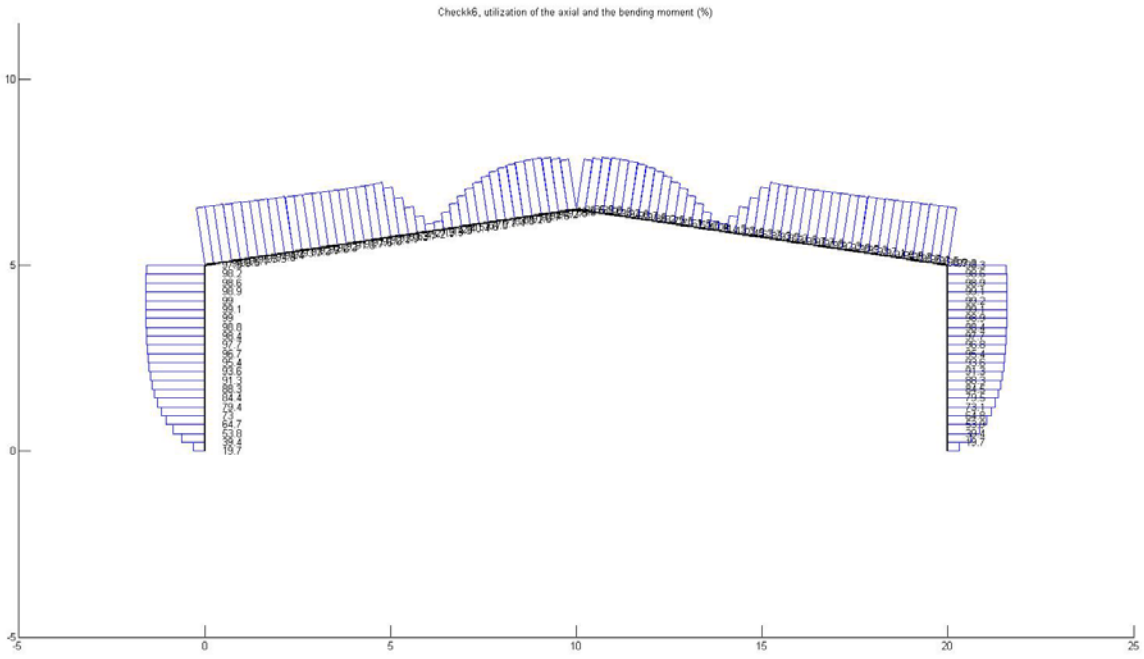


Figure 43: The utilization of combination of axial and moment (IF0) (eq. 5, section 2.1.1.5).

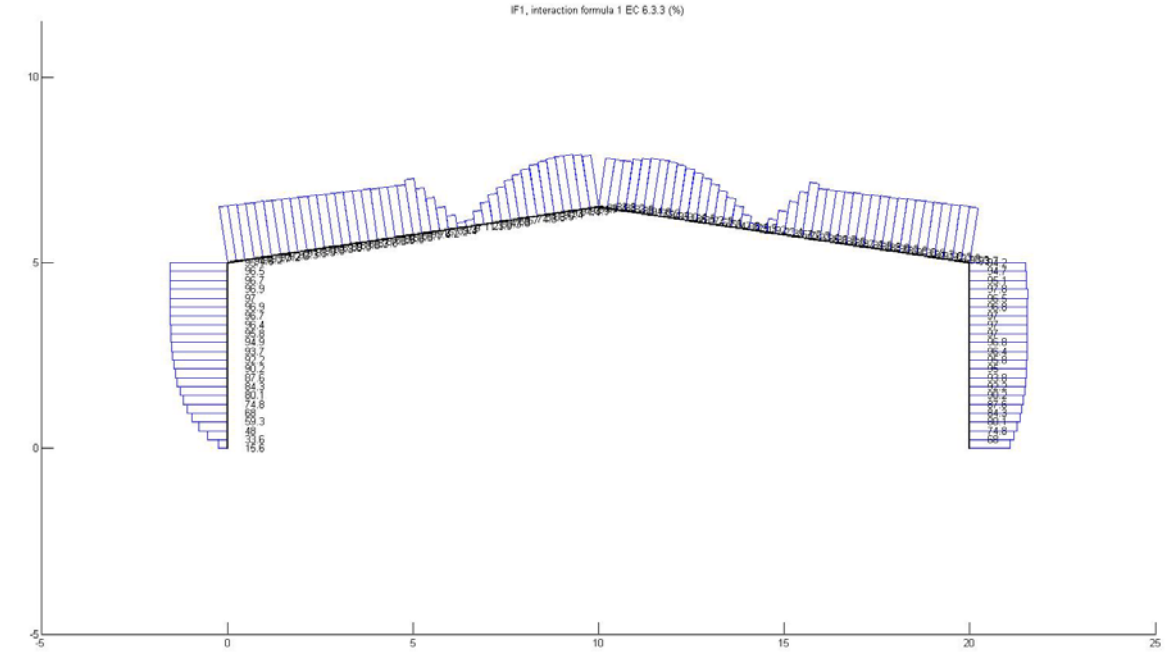


Figure 44: The utilization according to interaction formula 1(IF1) (eq. 23, section 2.1.1.8), EC3 6.3.3.

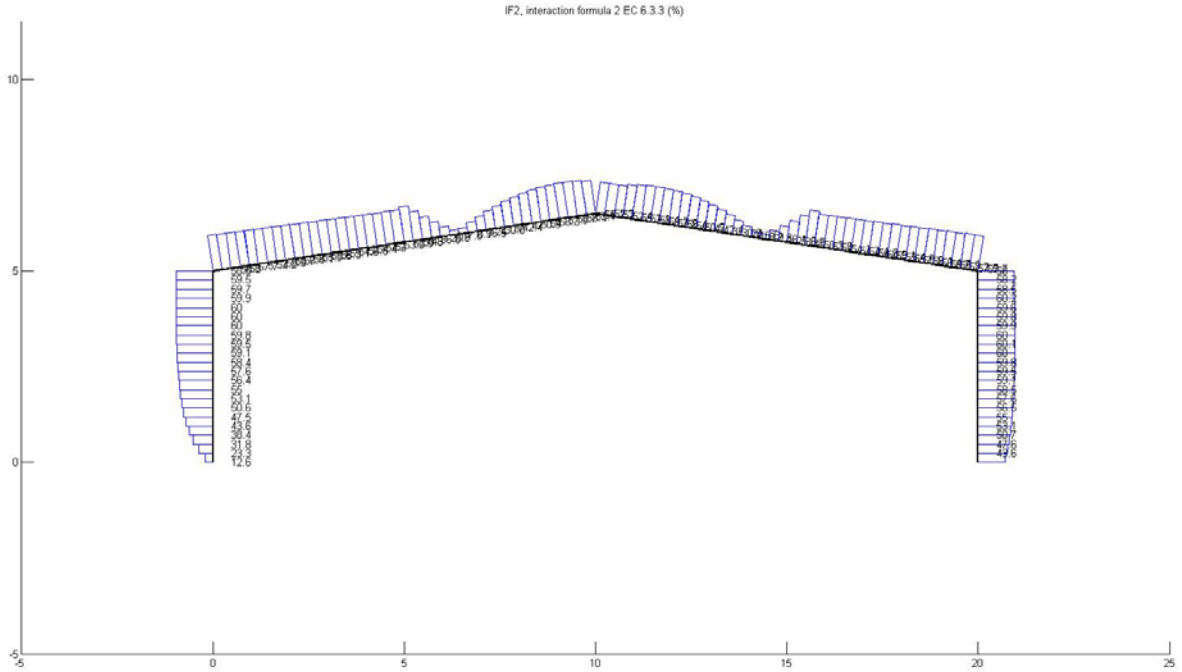


Figure 45: The utilization according to interaction formula 2 (IF2) (eq. 23, section 2.1.1.8), EC3 6.3.3.

As the deformation is not considered in this example, we get very high vertical deformation which is 315.4 % of the SLS limits (max L/250), see Figure 46 below.

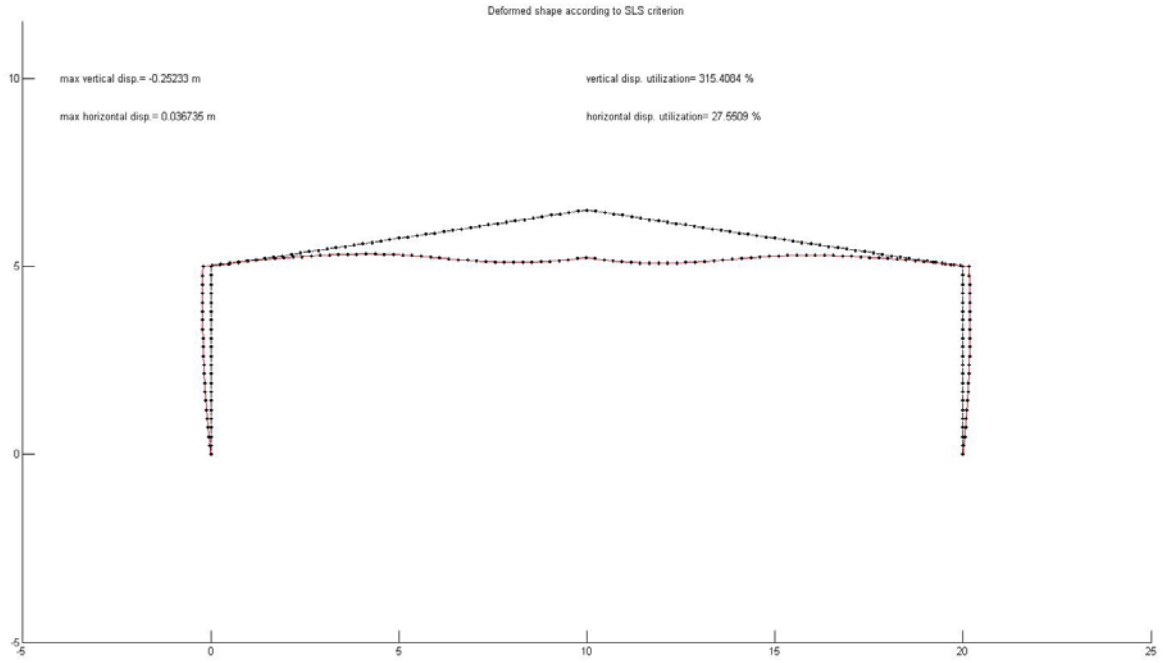


Figure 46: The deformation shape of the frame.

5.2 Example 2

Example 2 is similar to Example 1 above, the only difference is the absence of the bracings along the frame, unless the natural bracings the main nodes of the frame which are automatically generated, see section 3.2.5, see Figure 47. It is obvious that the main idea in this example is to compare shape of the frame founded by the algorithm with and without the bracings. The height of the web in this example is set be to as variables between (50-1500) mm to ensure wider space for the algorithm to find among, see Figure 38.

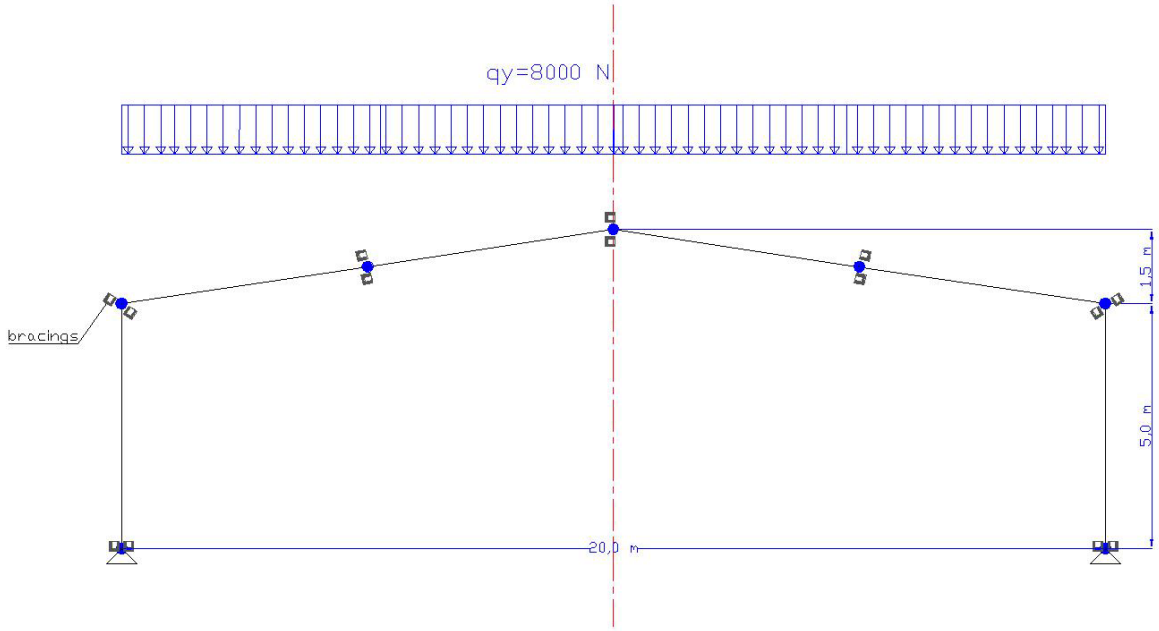


Figure 47: The initial inputs of the frame. In this example, the idea is to test the same frame as the previous example, but without the bracings, except the natural/false bracings at the main nodes of the frame.

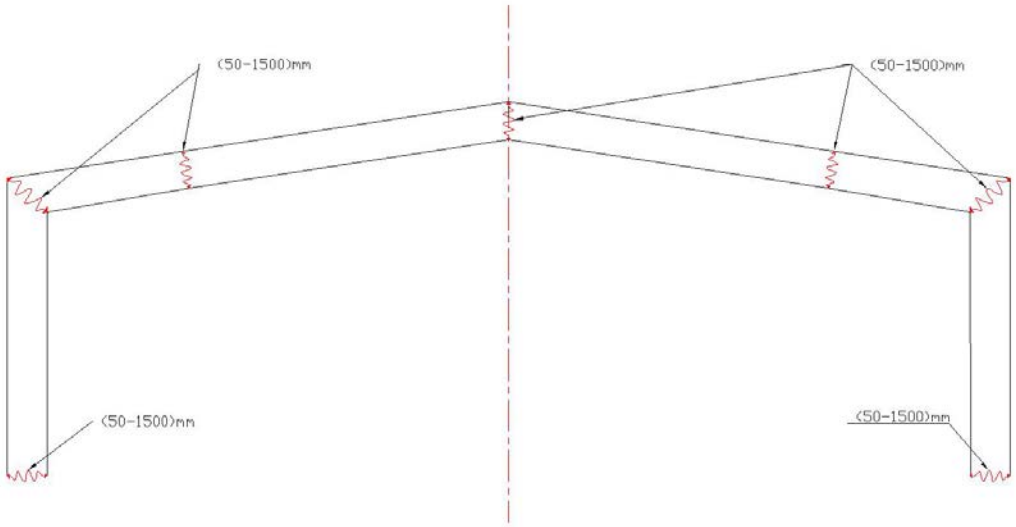


Figure 48: As input, the heights of the webs at the main nodes of the structure are decided to vary between 50-1500 mm.

Again the loads are according to ULS and there are no limitations for the deformation i.e. the SLS criterion is ignored in this example as well. The following frame was founded after 8 hours running with maximum generations of 140 generations. The algorithm has found the solution after about 90 generations, see figure 49 below:

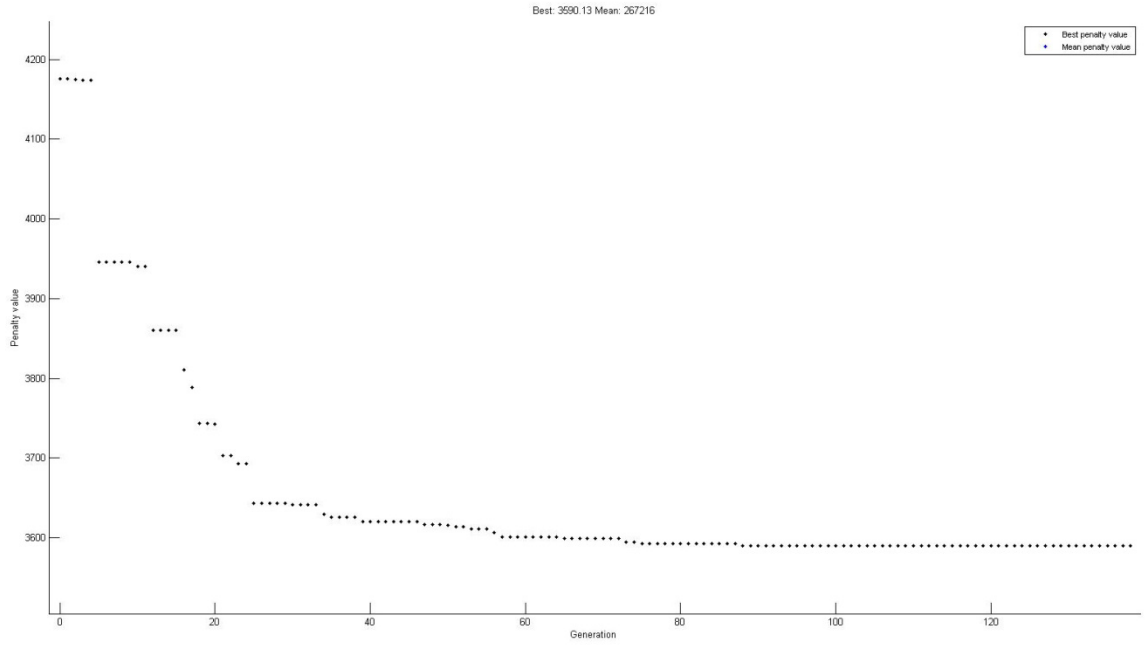


Figure 49: The development of the generations.

The algorithm found a frame with total weight of 3590 kg as shown in Figure 50 below.

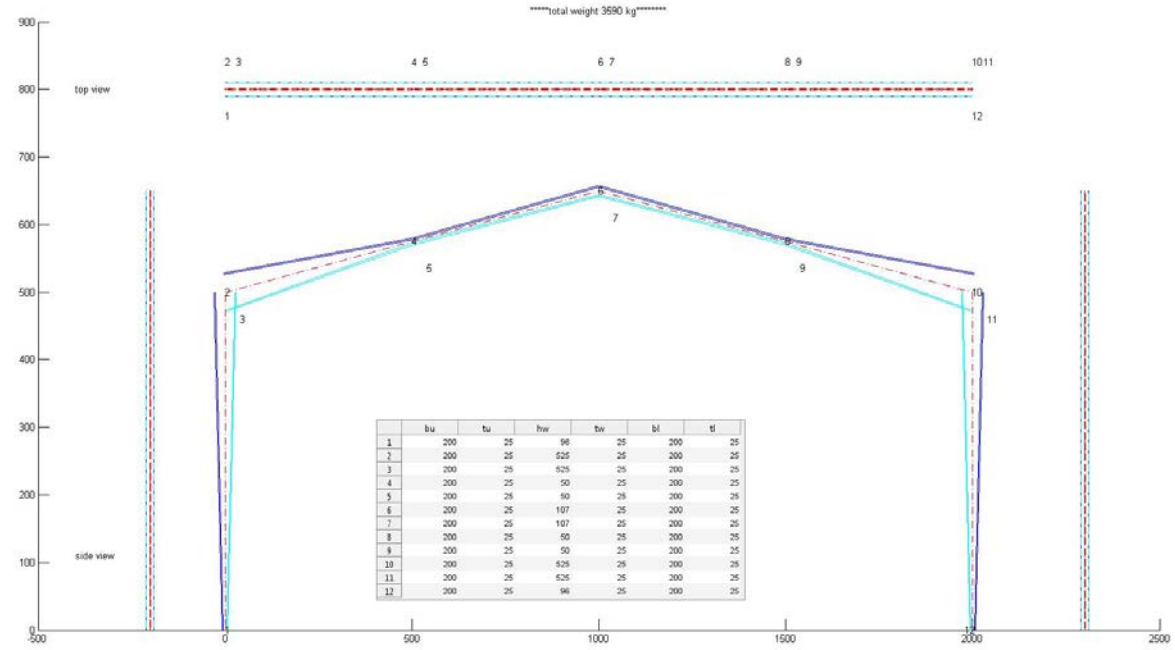


Figure 50: The shape of the whole frame.

The following table showing the result of the dimensions of the frame's cross sections:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	96 (50-1500)	25	200	25
2	200	25	525 (50-1500)	25	200	25
3	200	25	525 (50-1500)	25	200	25
4	200	25	50 (50-1500)	25	200	25
5	200	25	50 (50-1500)	25	200	25
6	200	25	107 (50-1500)	25	200	25
7	200	25	107 (50-1500)	25	200	25
8	200	25	50 (50-1500)	25	200	25
9	200	25	50 (50-1500)	25	200	25
10	200	25	525 (50-1500)	25	200	25
11	200	25	525 (50-1500)	25	200	25
12	200	25	96 (50-1500)	25	200	25

Table 6: The dimensions of the cross section of the frame's main elements.

It seems that it is the optimal shape since we get maximum utilization of 100.4 % at the interaction formula 1, see Figure 53. It is obvious that the maximum utilization that can be reached is 100%, otherwise in case of being exceeded, it is going to be punished by the penalty function and going gradually to eliminated.

The maximum utilization of the shear check, IF0, IF1 and IF2 are 39.4 %, 71.8%, 100.4%, and 68.3% respectively. The following figures showing the results of the utilizations:

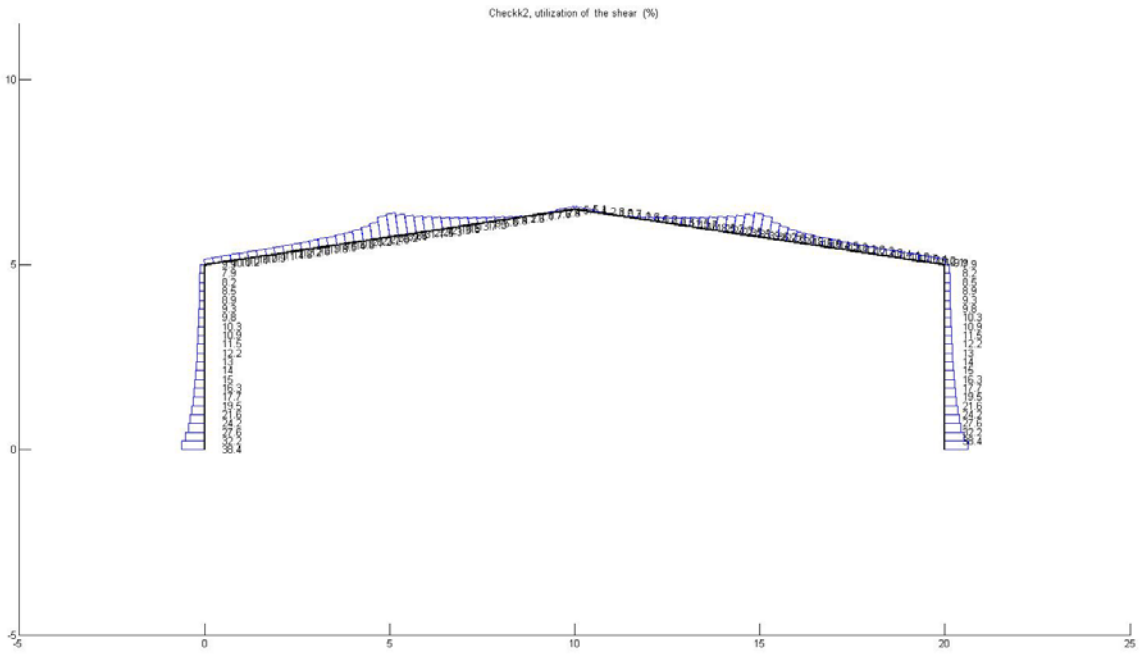


Figure 51: The utilization of shear.

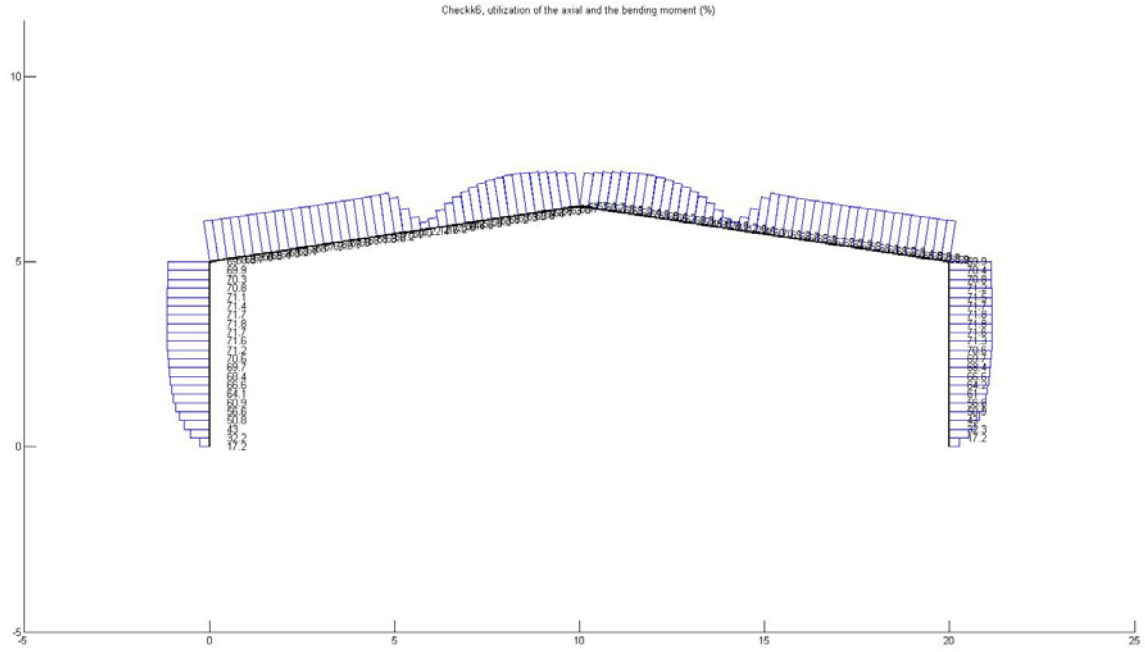


Figure 52: The utilization of combination of axial and moment (IF0).

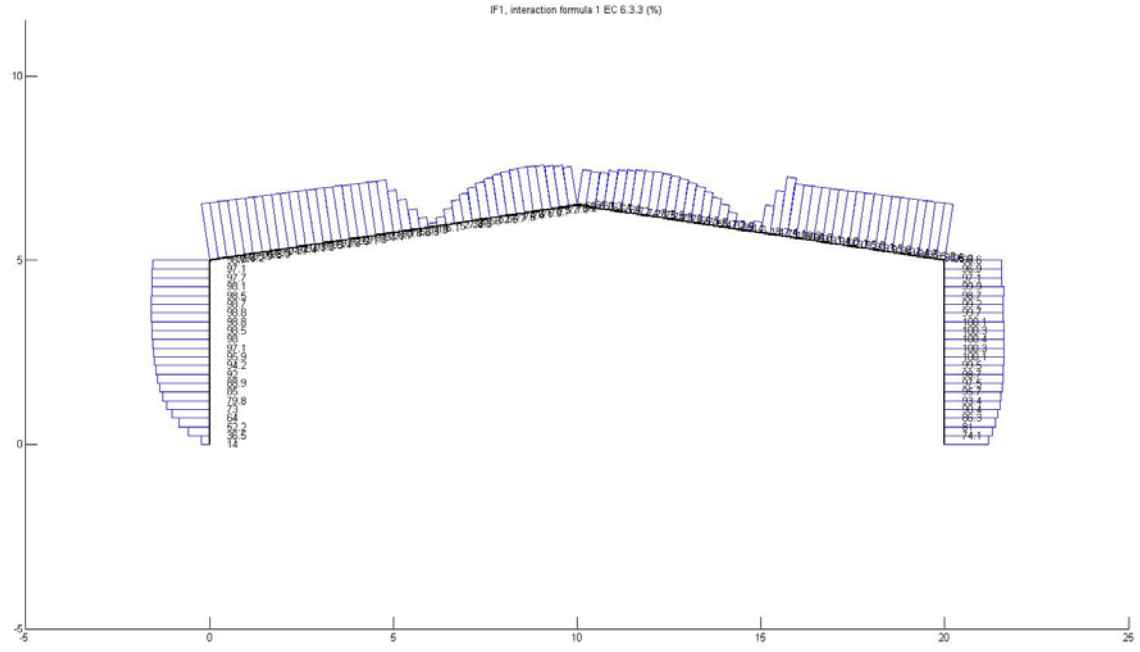


Figure 53: The utilization according to interaction formula 1(IF1).

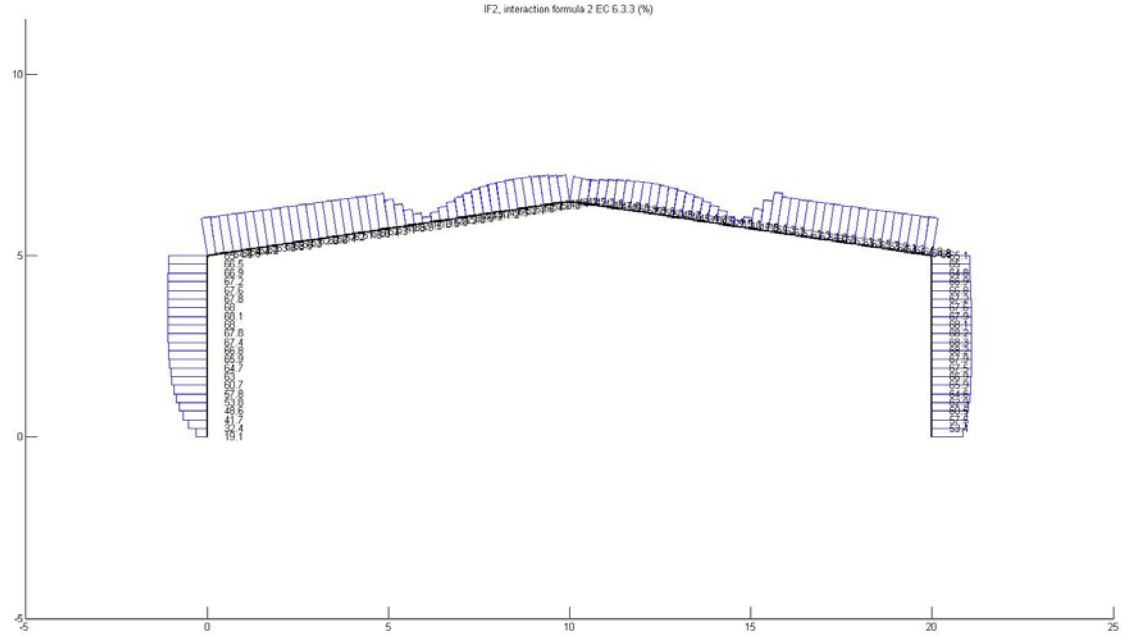


Figure 54: The utilization according to interaction formula 2(IF2).

As the deformation is not considered in this example, we get very high vertical deformation which is 182 % of the SLS limits (max L/250), see Figure 55 below.

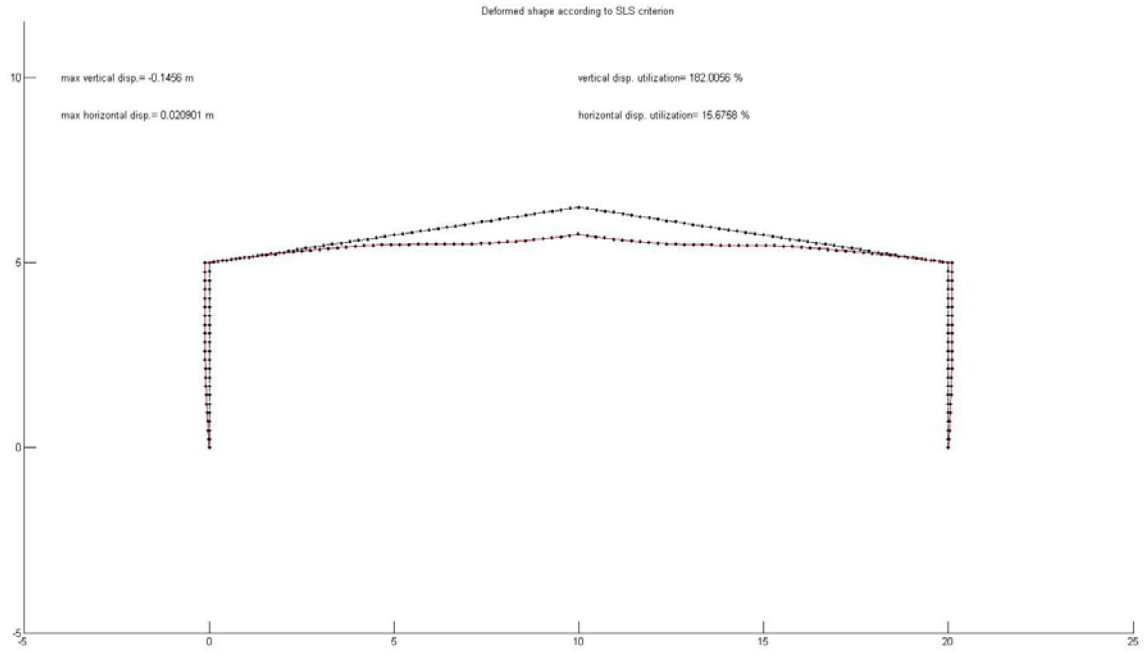


Figure 55: The deformation shape of the frame.

We can notice that the heights of the webs at the haunches are high in comparison to the columns to resist the high moment and they are higher than in example 1 which is fully braced, see Figure 56. This is firstly to resist the local buckling due to the absence of the bracings along the frame and to resist the high moment.

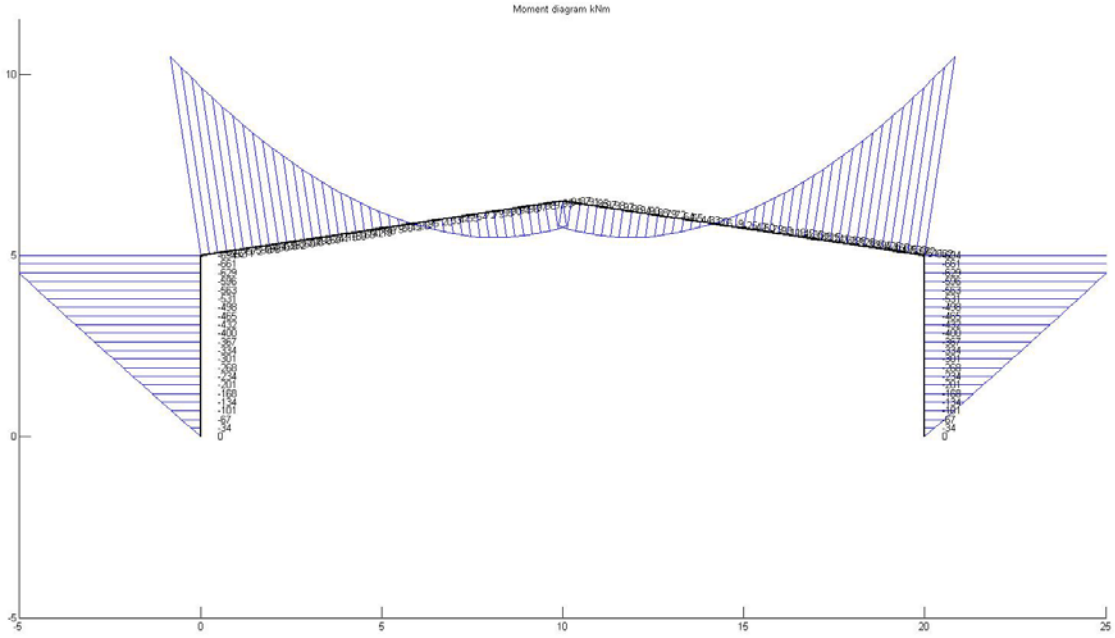


Figure 56: The moment diagram.

5.3 Example 3

Example 3 is similar to Example 1 and 2 above, the only difference is letting the algorithm to find the optimum pattern of the bracings distribution along the whole frame. The idea of this example is to compare how the design is going to be in case of full-braced, unbraced and optimum bracings pattern distribution. The height of the web in this example is set be to as variables between (50-1500) mm to ensure wider space for the algorithm to find among, see Figure 57.

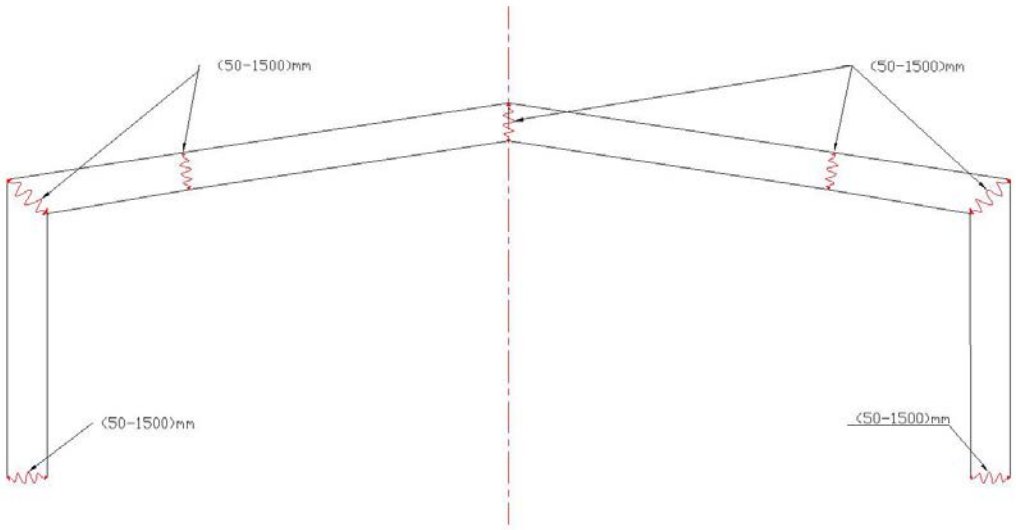


Figure 57: As input, the heights of the webs at the main nodes of the structure are decided to vary between 50-1500 mm.

Again the loads are according to ULS criterion and there are no limitations for the deformation i.e. the SLS criterion is ignored in this example as well. The algorithm was run

for about 12 hours and the maximum generation that has been reached is 206, see Figure 58 below:

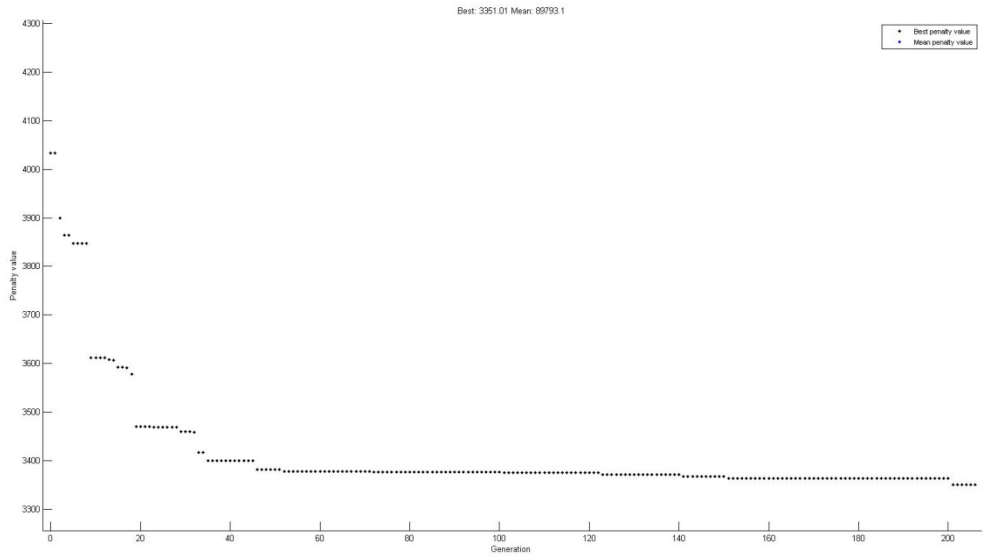


Figure 58: The fitness figure.

The following frame with total weight of 3311 kg was founded with optimum bracings distribution pattern, see Figure 59 below:

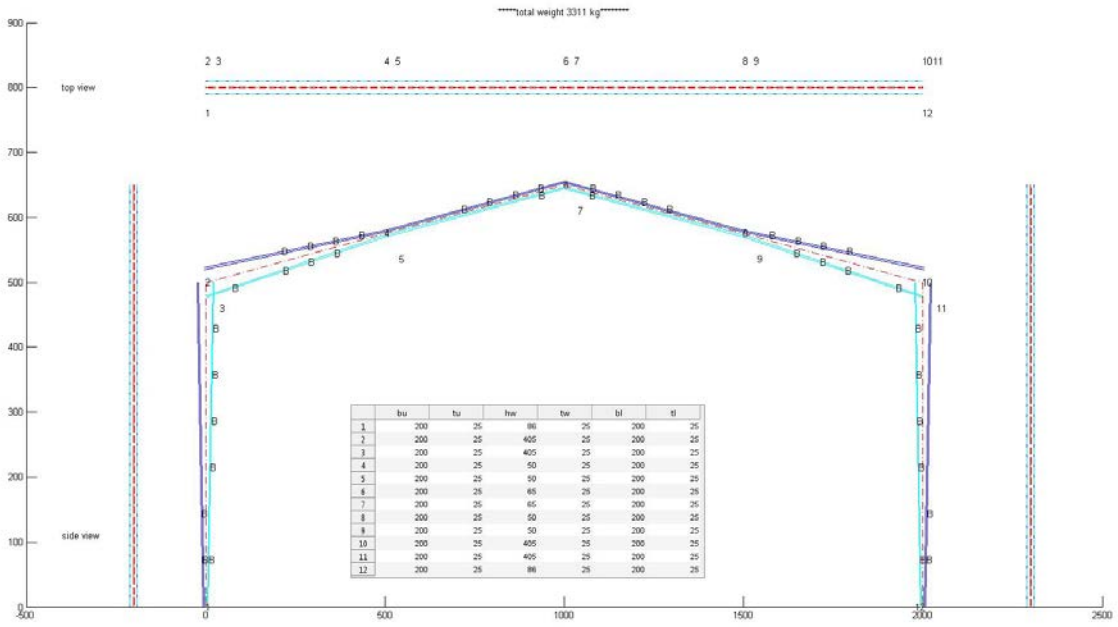


Figure 59: The shape of the whole frame with optimum bracings distribution pattern.

The following table showing the result of the dimensions of the frame's cross sections:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	86 (50-1500)	25	200	25
2	200	25	405 (50-1500)	25	200	25
3	200	25	405 (50-1500)	25	200	25
4	200	25	50 (50-1500)	25	200	25
5	200	25	50 (50-1500)	25	200	25
6	200	25	65 (50-1500)	25	200	25
7	200	25	65 (50-1500)	25	200	25
8	200	25	50 (50-1500)	25	200	25
9	200	25	50 (50-1500)	25	200	25
10	200	25	405 (50-1500)	25	200	25
11	200	25	405 (50-1500)	25	200	25
12	200	25	86 (50-1500)	25	200	25

Table 7: The dimensions of the cross section of the frame's main elements.

We can notice that the distribution pattern of the bracings is following the moment diagram of the frame since the bracings positions are in places where there are buckling risks. The bracings are concentrated at the upper compressed flanges at the middle of the frame since the moment is positive which means bigger risk for lateral torsional buckling for the upper flanges; While in the other hand we can see that the bracings are concentrated at the compressed lower flanges at the columns and the haunches due to big negative moment to reduce the risk of the lateral torsional buckling at the internal flanges. Figure 60 shows the moment diagram.

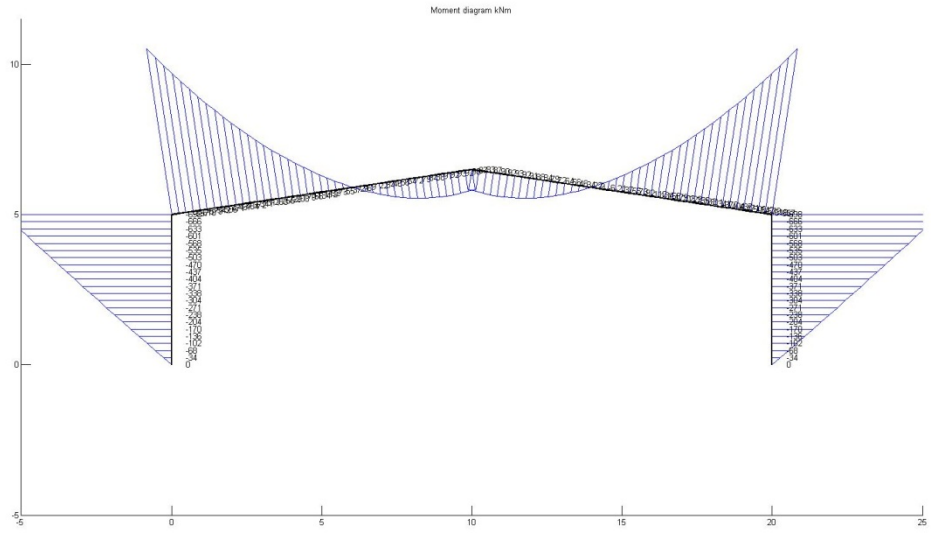


Figure 60: The moment diagram.

It seems that it is the optimal shape since we get maximum utilization of 98.1 % at IF0, the interaction of the axial and the bending moment (see Figure 62) and also maximum utilization of 97.3 % at interaction formula 1 (IF1), see Figure 63.

The maximum utilization of the shear check, IF0, IF1 and IF2 are 43.5 %, 98.1%, 97.3%, and 60.1% respectively. The following figures showing the results of the utilizations:

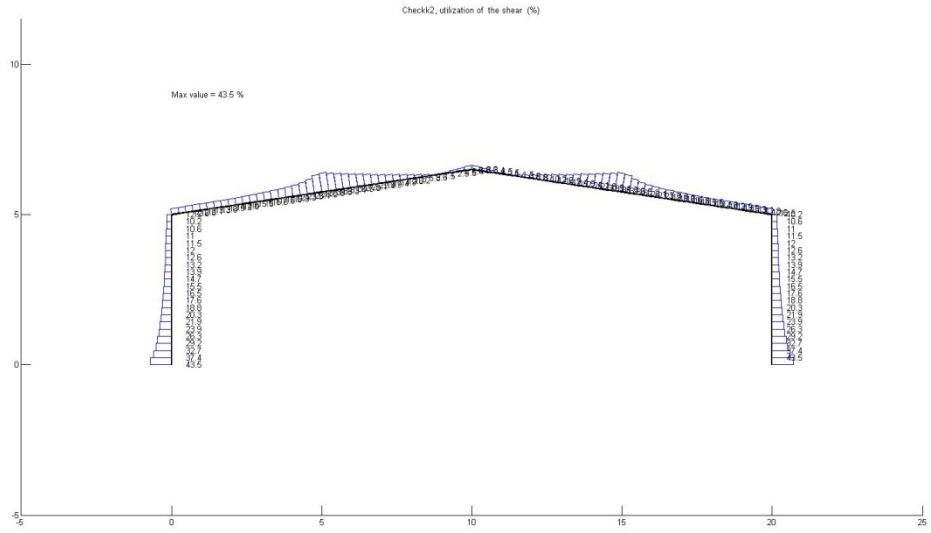


Figure 61: The utilization of shear.

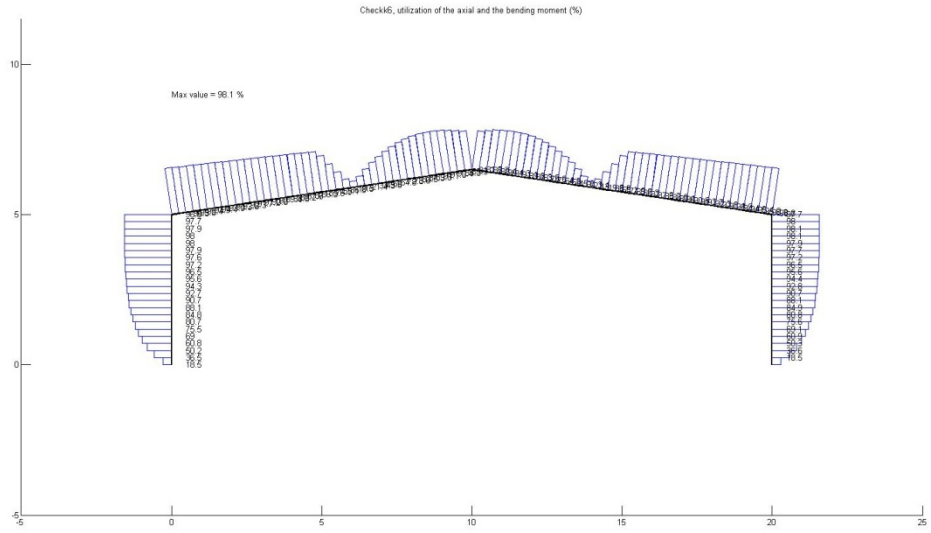


Figure 62: The utilization of combination of axial and moment (IF0).

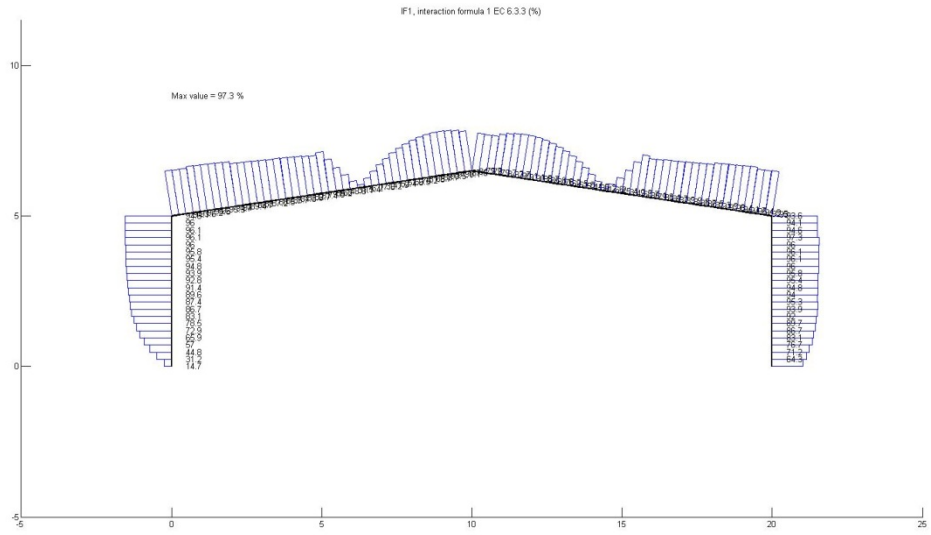


Figure 63: The utilization according to interaction formula 1(IF1).

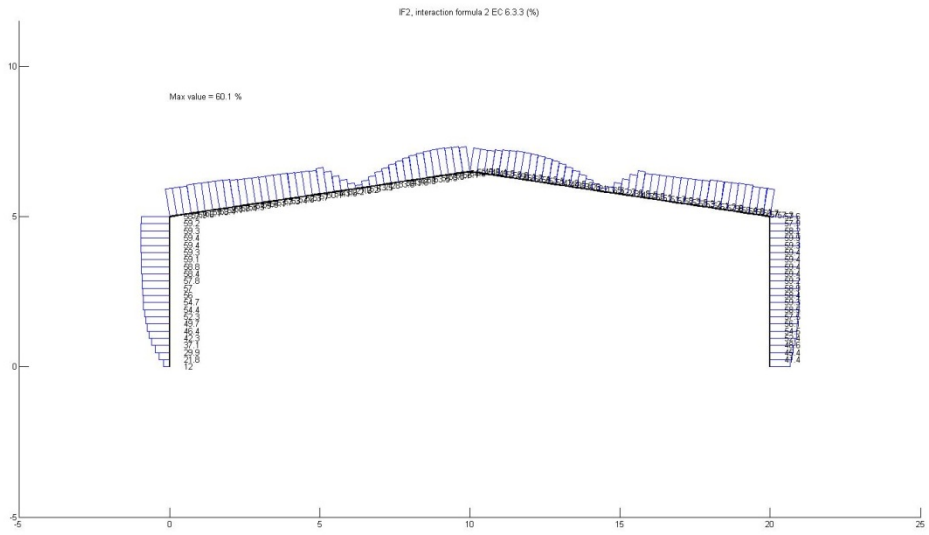


Figure 64: The utilization according to interaction formula 2 (IF2).

Since the deformation is not considered in this example, we get very high vertical deformation which is 298.3 % of the SLS limits (max L/250), see Figure 65 below.

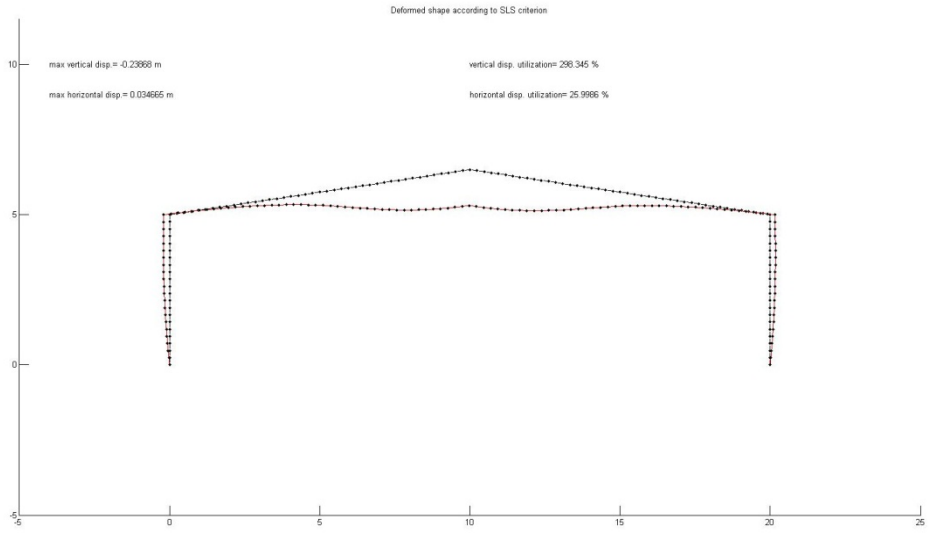


Figure 65: The deformation shape of the frame.

5.4 Example 4

In this example, the optimal design of a steel portal frame is determined with respect to the weight. The supports are set to be pinned and the loads are uniformly distributed over the rafters and equal to 8000 N. The design is according to both the ULS and SLS criterion, i.e. the horizontal and the vertical deformations are limited to L/150 and L/250 respectively. The span width of the frame is set to be 20.0 m and maximum height of 6.5 m. The distributing of

the bracing pattern is as it showed in Figure 66 below, which is in about 0.22 m distance between each other, which in turn is meant to be less buckling risk in this example.

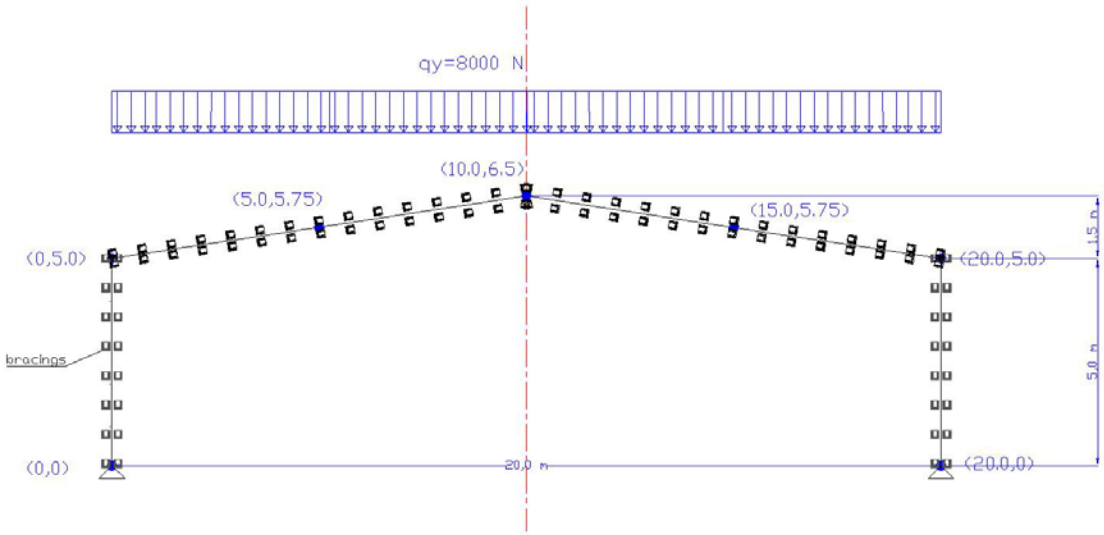


Figure 66: The initial inputs of the frame.

The heights of the webs at the main nodes of the structure are set to be variables between 50-1500 mm (see figure 67). Other dimensions such as the thickness of the web, the width and the thickness of the flanges are set to be constants. Since the structure is symmetric, only 3 parameters have to be designed during the optimization. The horizontal- and the vertical deformation are unlimited in this example.

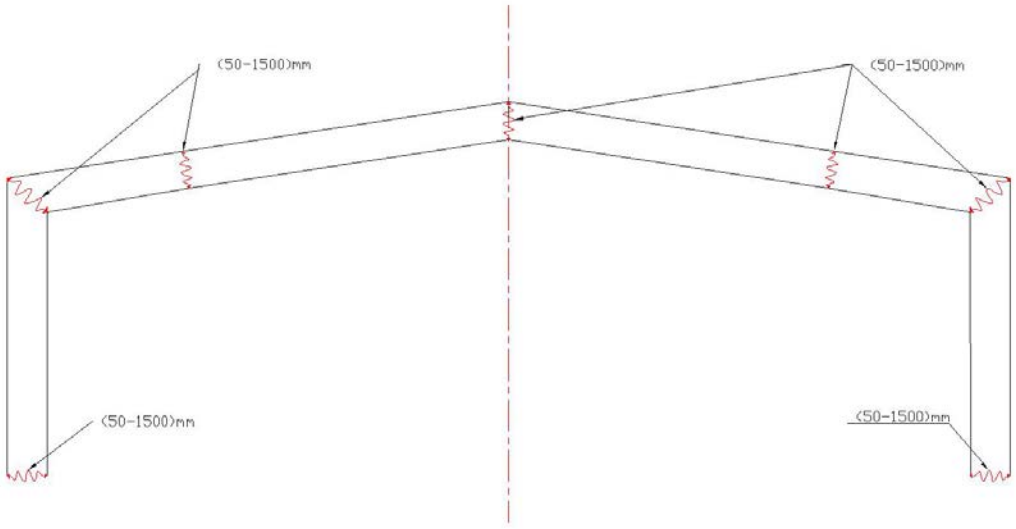


Figure 67: As input, the heights of the webs at the main nodes of the structure are decided to vary between 50-1500 mm.

The population size is set to be 500. The algorithm found a frame with total weight of 3915 kg as shown in Figure 68 below.

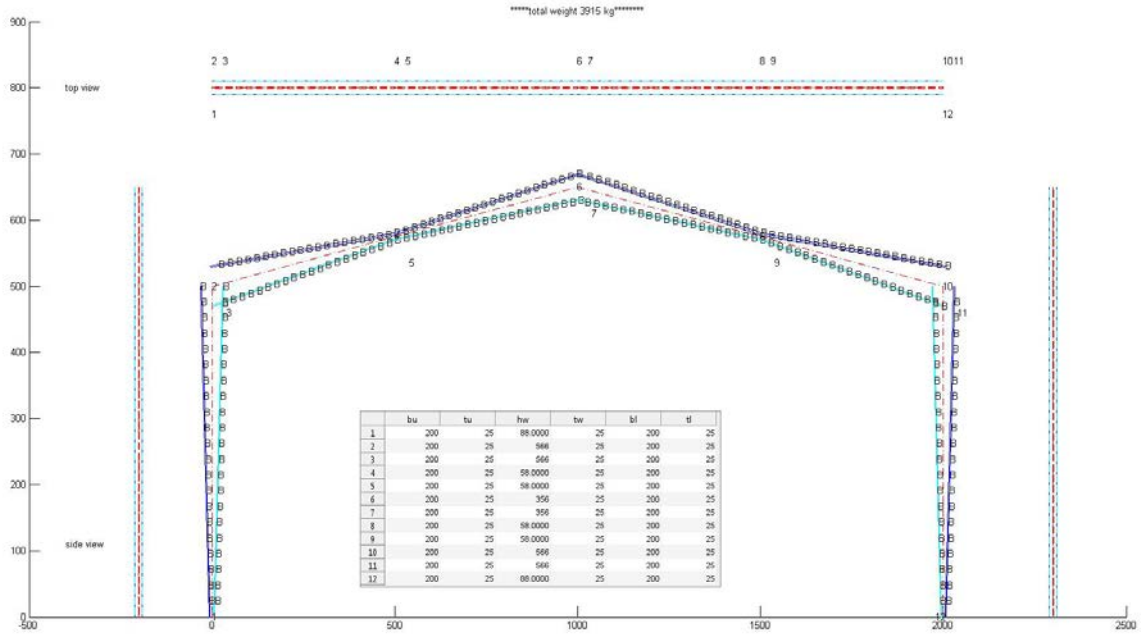


Figure 68: The optimum shape of the frame that the algorithm has found. The letter B in the frame refers to the internal and the external bracings. The table shows the dimensions of each cross section of the frames' main elements.

The following table shows the dimensions of the cross sections at the edges of frame's main elements, see section 3.2.4:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	88 (50-1500)	25	200	25
2	200	25	566 (50-1500)	25	200	25
3	200	25	566 (50-1500)	25	200	25
4	200	25	58 (50-1500)	25	200	25
5	200	25	58 (50-1500)	25	200	25
6	200	25	356 (50-1500)	25	200	25
7	200	25	356 (50-1500)	25	200	25
8	200	25	58 (50-1500)	25	200	25
9	200	25	58 (50-1500)	25	200	25
10	200	25	566 (50-1500)	25	200	25
11	200	25	566 (50-1500)	25	200	25
12	200	25	88 (50-1500)	25	200	25

Table 8: The dimensions of the cross section of the frame's main elements. It is obvious to mention that all dimensions are constants, except the height of the webs (which are marked in different color in the table) which are set to be as variables between 50-1500 mm and letting the algorithm to find the optimal solution within this interval.

It seems that it is the optimal shape since we get maximum vertical deformation of 99.97%, see Figure 69. It is obvious that the maximum utilization that can be reached is 100%, otherwise in case of being exceeded; it is going to be punished by the penalty function and going gradually to be eliminated.

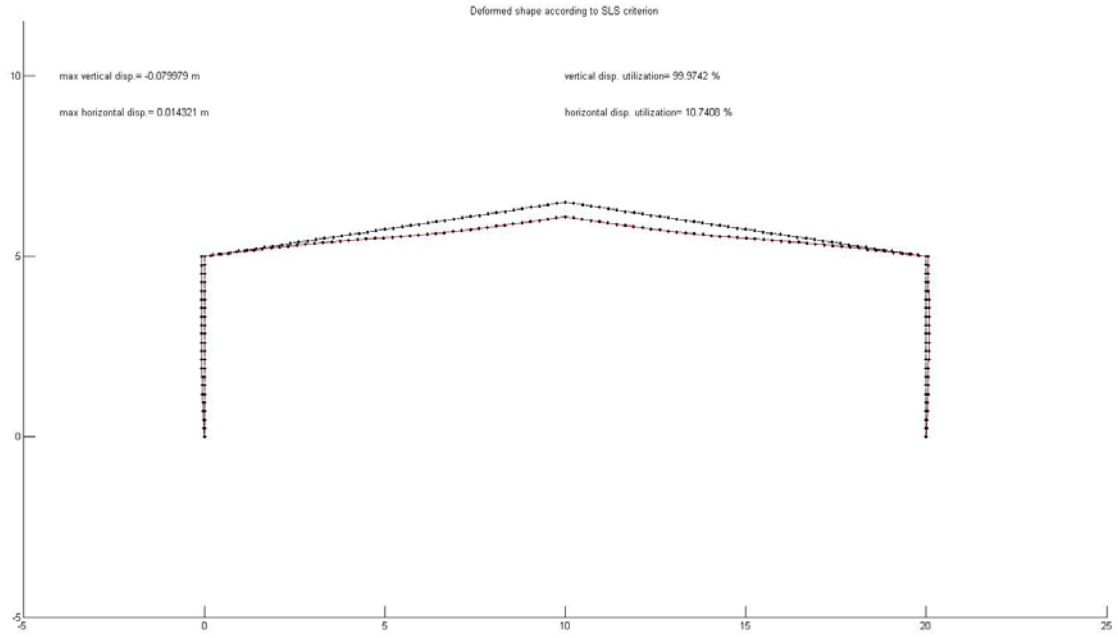


Figure 69: The deformation shape of the frame.

The maximum utilization of the shear check, IF0, IF1 and IF2 are 40.2 %, 65.2%, 63.4%, and 40% respectively. The following figures showing the results of the utilizations:

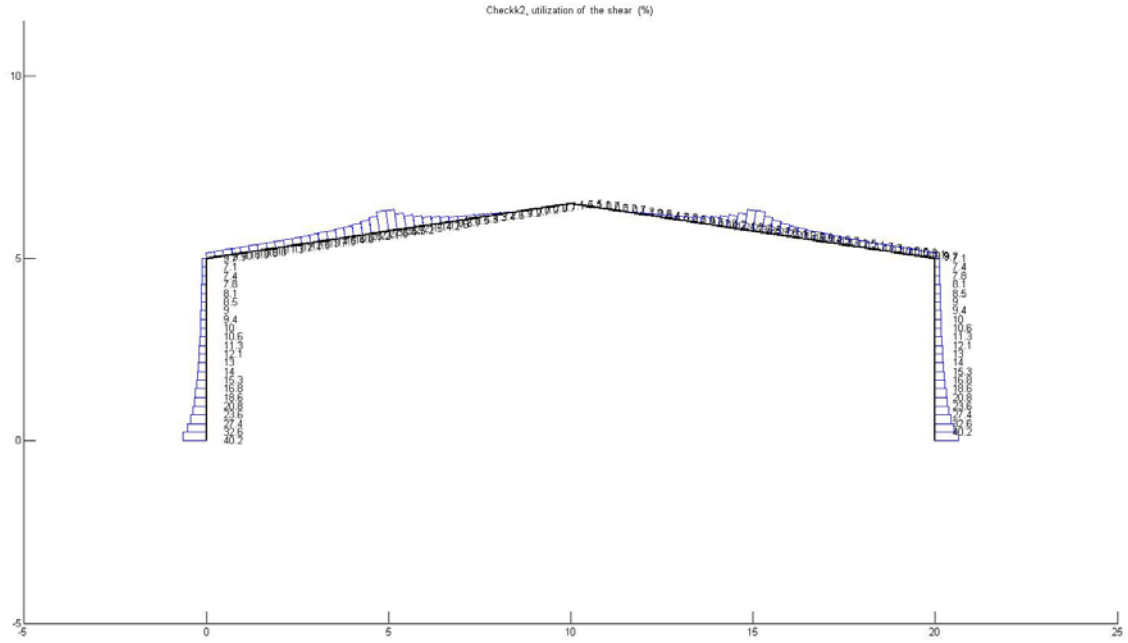


Figure 70: The utilization of shear (shear check).

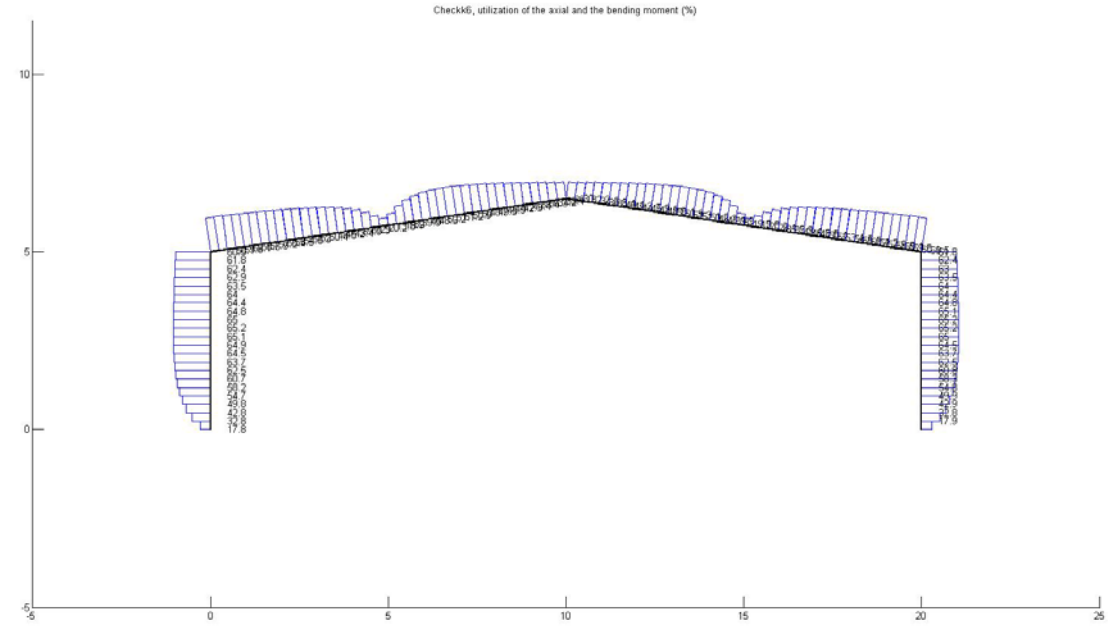


Figure 71: The utilization of combination of axial and moment (IF0).

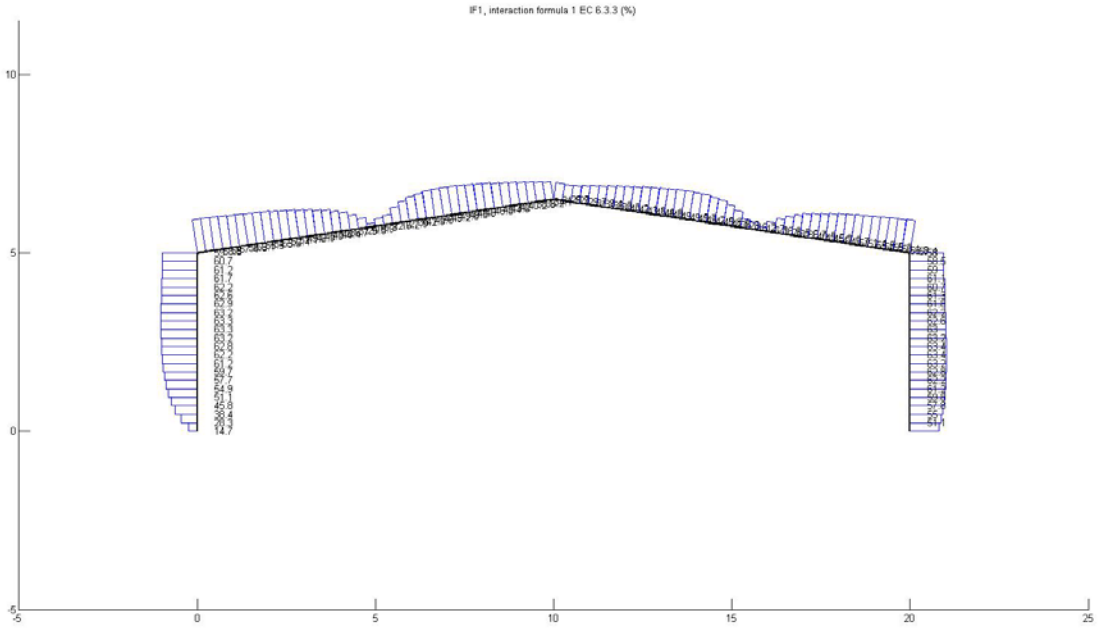


Figure 72: The utilization according to interaction formula 1(IF1).

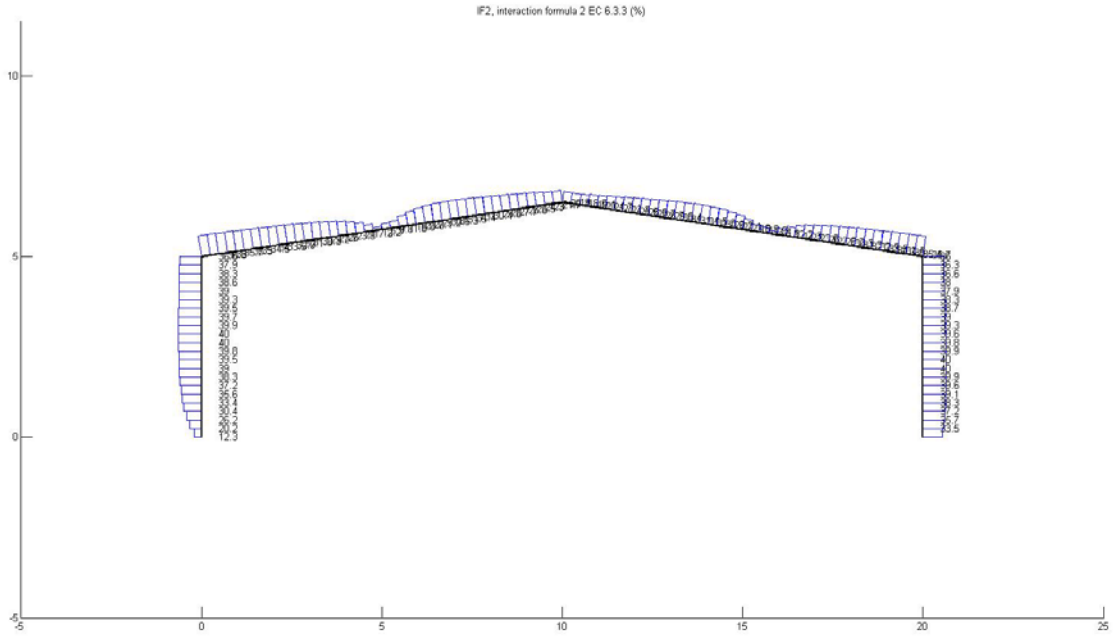


Figure 73: The utilization according to interaction formula 2 (IF2).

5.5 Example 5

Example 4 is similar to Example 3 above, the only difference is the absence of the bracings along the frame, unless the natural bracings at the main nodes of the frame which are automatically generated, see section 3.2.5, see Figure 74. It is obvious that the main idea in this example is to compare the shape of the frame founded by the algorithm with and without the bracings.

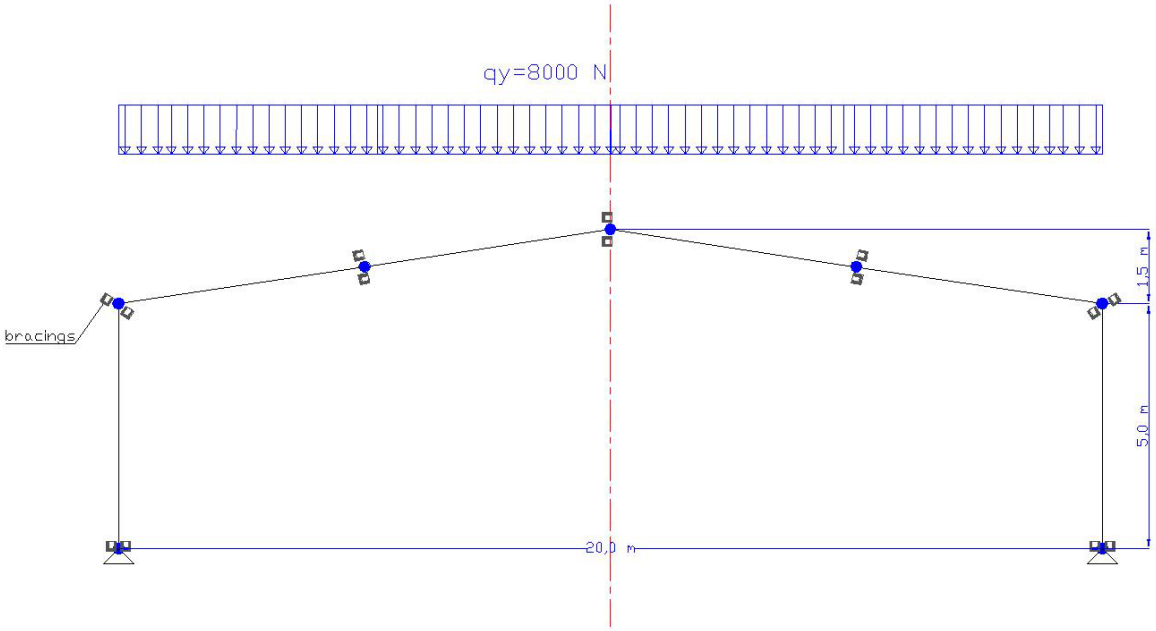


Figure 74: The initial inputs of the frame. In this example, the idea is to test the same frame as the previous example, but without the bracings, except the natural/false bracings at the main nodes of the frame.

The following frame was found by the algorithm:

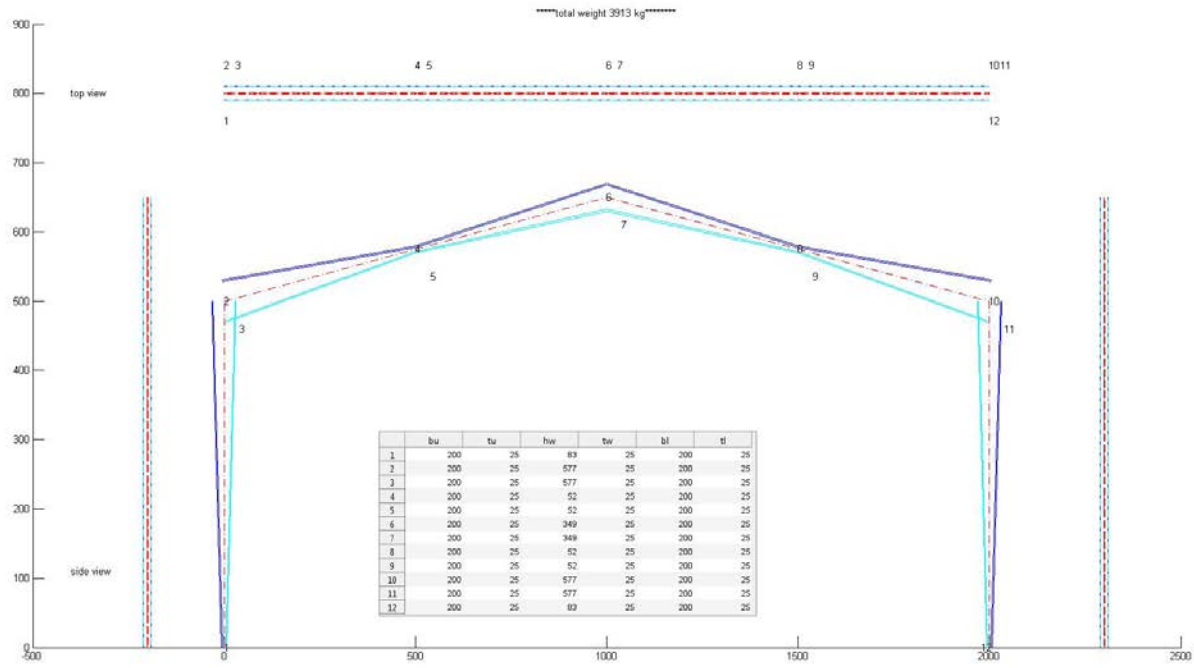


Figure 75: The shape of the whole frame.

The following table showing the result of the dimensions of the frame's cross sections:

Number of cross sections at the edges of the main numbers of the frame	bu, the width of the upper flange (mm)	tu, the thickness of the upper flange (mm)	hw, the height of the web (mm)	tw, the thickness of the web (mm)	bl, the width of the lower flange (mm)	tl, the thickness of lower flange (mm)
1	200	25	83 (50-1500)	25	200	25
2	200	25	577 (50-1500)	25	200	25
3	200	25	577 (50-1500)	25	200	25
4	200	25	52 (50-1500)	25	200	25
5	200	25	52 (50-1500)	25	200	25
6	200	25	349 (50-1500)	25	200	25
7	200	25	349 (50-1500)	25	200	25
8	200	25	52 (50-1500)	25	200	25
9	200	25	52 (50-1500)	25	200	25
10	200	25	577 (50-1500)	25	200	25
11	200	25	577 (50-1500)	25	200	25
12	200	25	83 (50-1500)	25	200	25

Table 9: The dimensions of the cross section of the frame's main elements.

The maximum utilization of the shear check, IF0, IF1 and IF2 are 42.4 %, 64.5%, 91.7%, and 63.5% respectively. The following figures showing the results of the utilizations:

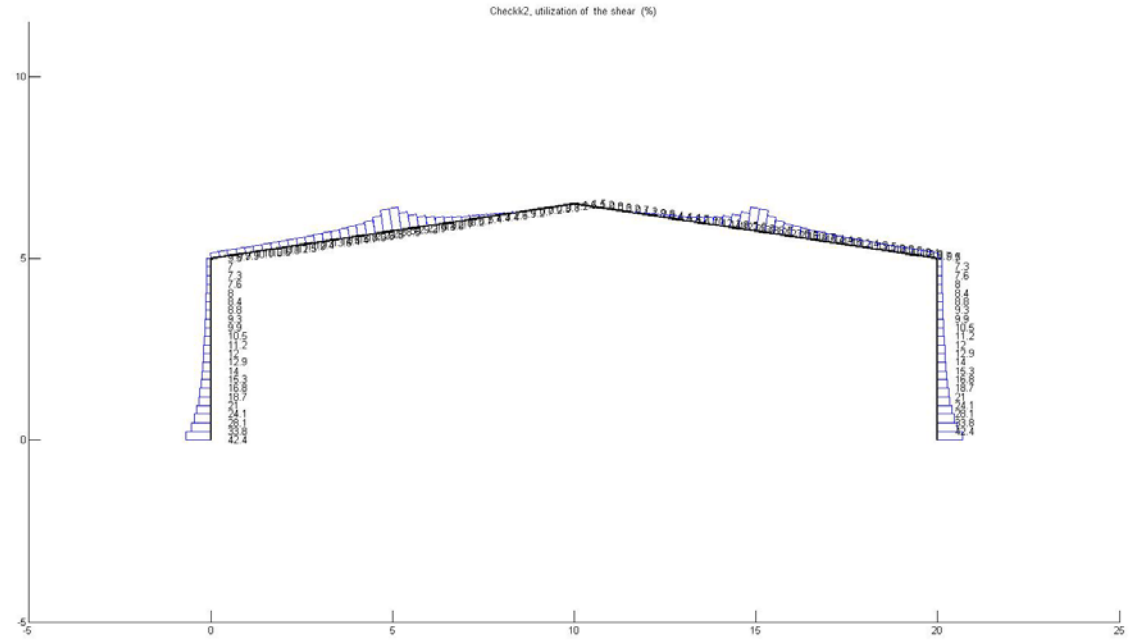


Figure 76: The utilization of shear.

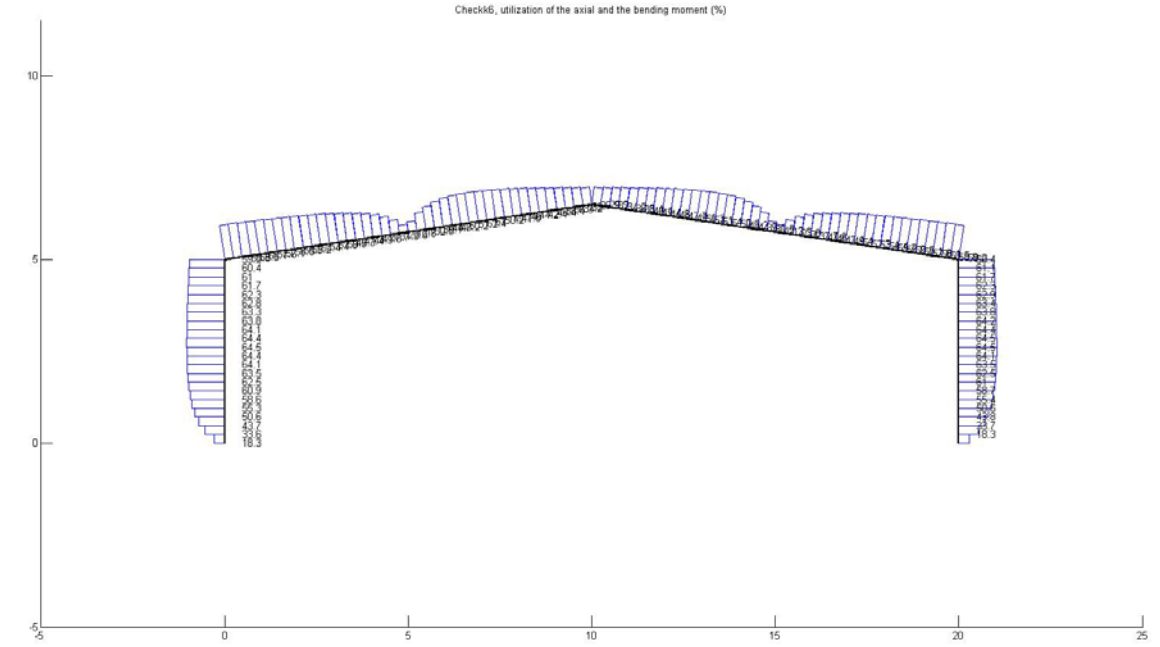


Figure 77: The utilization of combination of axial and moment.

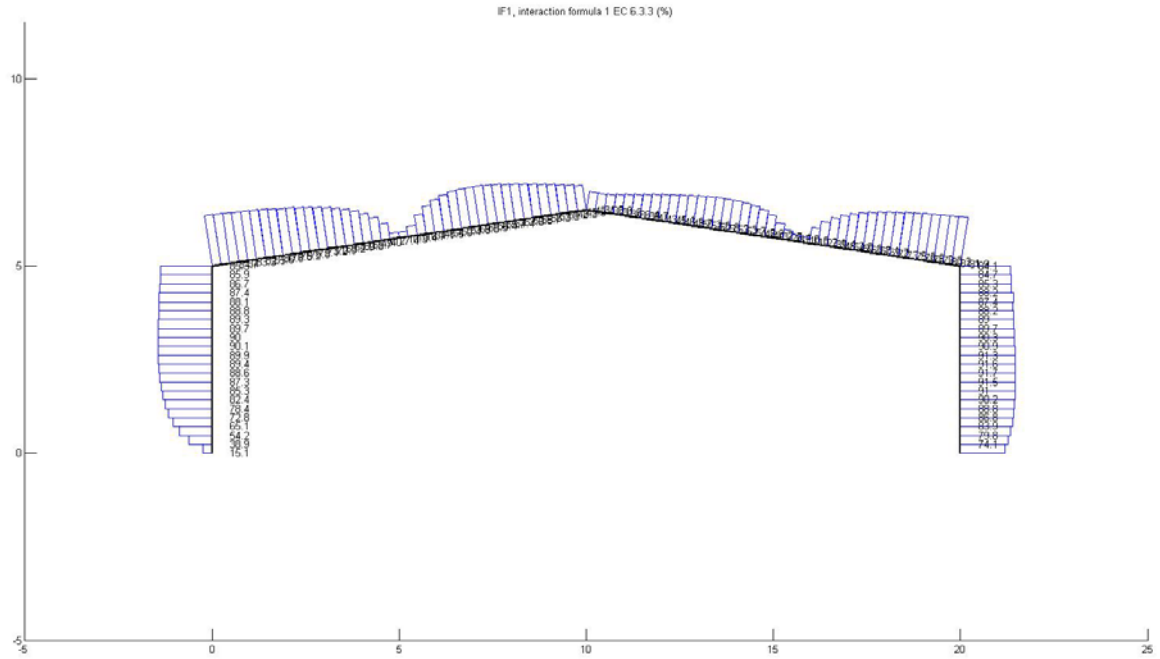


Figure 78: The utilization according to interaction formula 1.

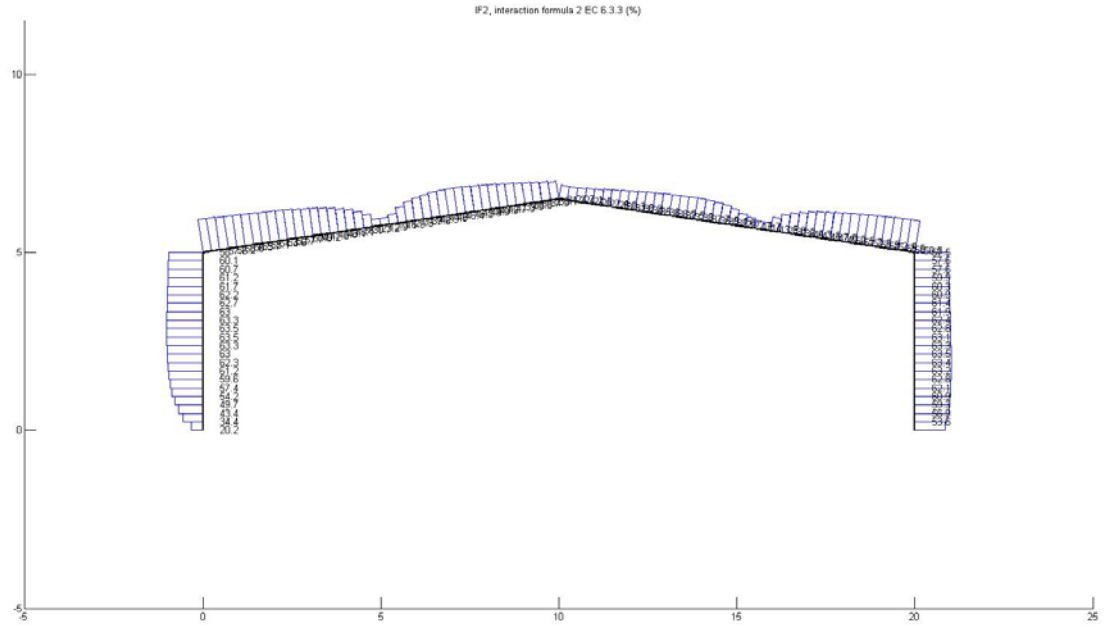


Figure 79: The utilization according to interaction formula 2.

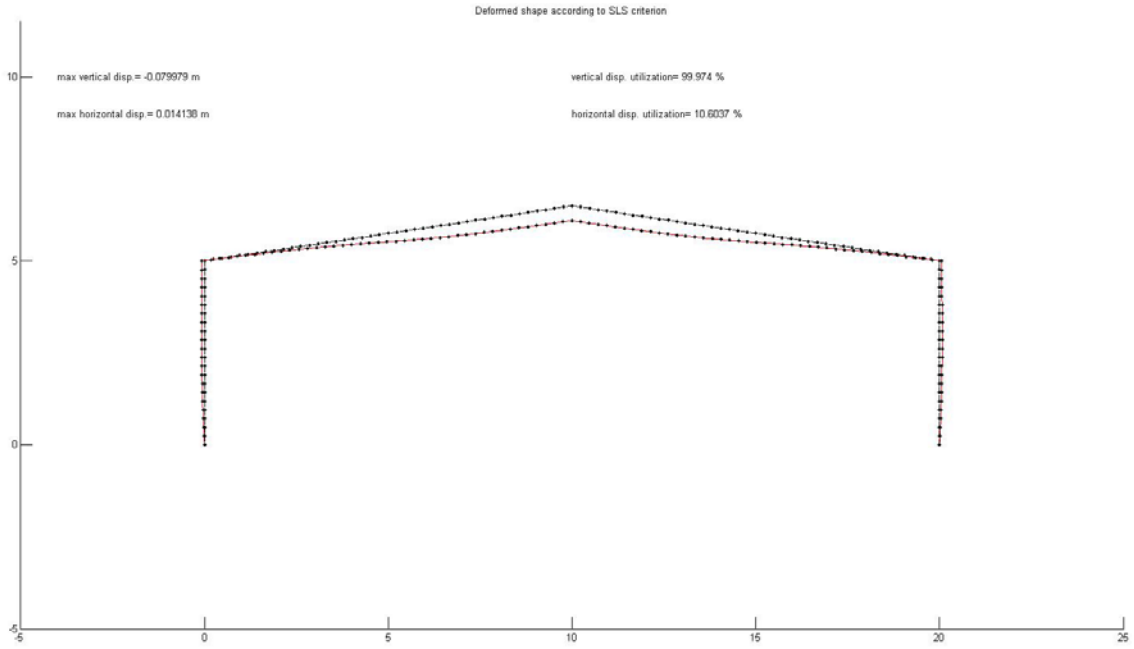


Figure 80: The deformation shape of the frame.

The total weight of this frame is 3913 kg. We can notice that the heights of the webs at the sides are high in comparison to the columns to resist the high moment, see Figure 81. This is firstly to resist the local buckling due to the absence of the bracings along the frame and to keep the vertical deformation within the SLS criterion.

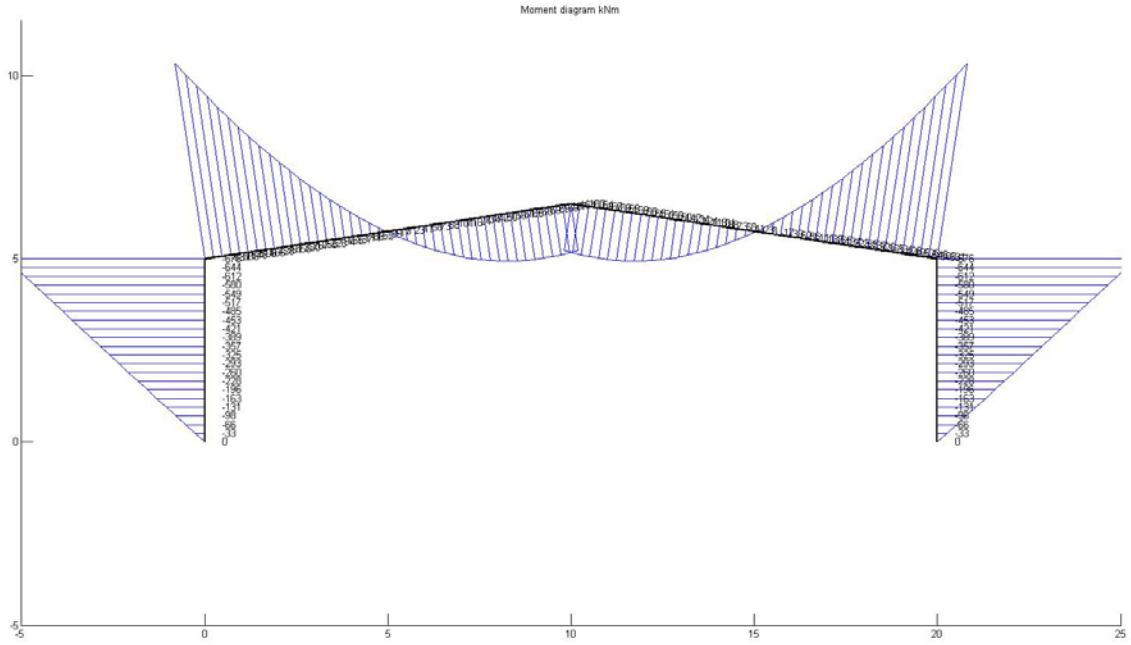


Figure 81: The moment diagram.

It seems that there is no a big difference between the two examples since SLS criterion is the critical and dominating. It can also happen that two designs with small difference in the dimensions have the same weight (one with higher webs at the sides and less high at the center and vice versa), and keeping acceptable SLS criterion.

6 Conclusions

The computing time is essential since the algorithm is in principle based on searching manner to find the optimum design among millions of possibilities. Many factors can affect the speed of the algorithm. One of the most essential factors is the speed of the computer. Using of powerful computer with multi processors can reduce the searching time and make the algorithm more effective and practically valuable. Another factor that can reduce the calculation time is by minimizing the range of limitation of the variables. An initial test to be done in advance is recommended since it is not easy to guess in which range the variables are.

According to the finite element principle, the finer element mesh is the more precise the results are. But finer mesh/more subelements means taking of longer computing time. The tests that have been done during this project showed that a number of divisions/subelements of 15 portions are enough to get a feasible result.

In this project, cross section class 4 is not taken into account with an aim to simplify the calculations as the project period is limited and to focus on other essential issues instead. This prevention leads to that the designed structures are heavier since less slender plates are used in class 1, 2 and 3 than in class 4.

The value of the numeric penalty should be chosen feasibly and proportional with amount of the violation. Choosing of non-proportional penalty leads to that GA cannot differ between offsprings that violate the limits with low or high grade. High penalty value causes to that GA cannot choose low violated parents which can be close to the optimum.

The choice of the number of variables and their intervals affects the time required to find the optimum solution since it affects the number of the possibilities and the time required for GA to search among them. It is a good idea to choose only the required variables such as the heights of the webs and to limit the intervals of them within expected and feasible limits.

Since the units of the variables are in millimeters and since they are integers (whole numbers) because there is no need to use precision less than 1 mm practically, the vector of the variables the GA generate, x is likely to be integer. This can reduce the elapsed time for the algorithm to find the optimum design since using of non-integer variables with many decimals means longer strings which are generated by GA and more unnecessary possibilities to search among.

The external and the internal bracings have essential role for the stability of the structure against buckling issues. The testing design results show that the existing of the bracings along the whole structure produces lighter designs than the unbraced designs. The reason is that it is natural to prevent the buckling risk by using heavy design with thick plates and wide/high cross section dimensions. Finding the optimum numbers and position of the bracings pattern by the algorithm is efficient since it is an optimum design with less number of bracings as possible i.e. by putting the bracings at the necessary places only. This can reduce the costs of the welding, the material etc. However, the way the algorithm has found of distributing the bracings is one way or another not clear. It seems that the algorithm sometimes doesn’t follow a certain pattern and it doesn’t even follow the moment diagram as it is expected either,

especially when the number of the bracings is large and it is hard for the algorithm to search among a huge number of probabilities.

Trying to get as light frame as possible by choosing thin plates at the initial design (as not variable inputs) is not always a good idea as it ends up with some violations in the final design. The reason is that it is impossible to find a design that can fulfill the criterions with dimensions less than the required.

Finding a design that can fulfill the SLS criterion means heavy structure with high webs at the launchings. This design is compulsory to stand the deformation (especially vertical deformation) caused by SLS load combination. However, the utilization of SLS limitations will dominate over the other utilizations like the capacity or the buckling. The domination of SLS utilization refers to that the design have reached the optimum (around 100% at the SLS utilization), while the other utilization are still at lower levels. This means that the algorithm will find a number of designs that can fulfill the SLS criterion (and of course the other utilizations since they are lower) with almost the same weight. This is true by having high web at the launches and less web height at the center line or vice versa and keeping the same weight.

7 Suggestions for further works

This project is a start point for a long – but interesting – way to develop a complete and practical algorithm for designing optimal steel portal frames.

The first thing that can come to the mind of developing the algorithm is including of cross section class 4. Since cross section class 4 is not included, the design is still conservative and relatively heavier as the reduction of the cross section due to the buckling is prevented.

It could be a good idea to calculate the costs of the frame instead of the weight to be optimized. The costs may include the material, the manufacturing, the labor, the welding costs etc. since installing the structure including the welding costs can be high and unneglectable. For instance, the welding of the bracings costs can be higher than the steel material itself, i.e. thicker frame's plates (thicker flanges) can be cheaper than installing bracings to reduce the risk of buckling. Although the labor costs depend on which country we are i.e. the material can be more expensive than the labor costs and vice versa. It is recommended in the further works at the algorithm to find a balance between material costs and labor costs for the optimization process.

To easify the problem in this project, the frame considered to be as one unit to prevent going into detailed calculation of the capacity and stiffness of the connections. The capacity of the connection depends on its type such as welded or screwed connection etc. It is recommended for further work to provide an algorithm with practical and wider range of problems covering to include the calculation of the connections. The algorithm should at least cover the designing of three pinned frame type which is a popular design for industrial buildings and storage locals.

8 Bibliography

Austrell, P.-E.et al., 2004. *CALFEM: A Finite Element Toolbox.* Lund: KFS i Lund AB.

CEN, 2005. *European Committee For Standardisation. Eurocode 3: Design of steel structures – Part 1-1: General rules and rules for buildings.*. Brussels: s.n.

Chen, Y. & Hu, K., 2008. Optimal design of steel portal frames based on genetic algorithms. *Higher Education Press and Springer-Verlag.*

Christensen, Klarbring, P. W. & Anders, 2008. *An Introduction to Structural Optimization.* Netherlands: Springer.

Dawkins, R., 1976. *The Selfish Gene.* New York: Oxford University Press.

Deb, K., 1997. Genetic Algorithm in Search and Optimization: The Technique and Applications.

Etzkorn, B., 2011. *Non-Linear Optimization Using Palisade's Evolver.* [Online]
Available at: http://www.benetzkorn.com/2011/09/non-linear-model-optimization-using-palisades-evolver/
[Accessed 16 05 2012].

Greiner, R. et al., 2011. *Design Guidelines For Cross-Section And Member Design According To Eurocode 3 With Particular Focus On Semi-Compact Sections.* Graz: Institute for Steel Structures and Shell Structures.

Hradil, P., Mielonen, M. & Fülöp, L., 2010. Advanced design and optimization of steel portal frames. *Rakenteiden Mekaniikka (Journal of Structural Mechanics),* Volume 43, pp. 44-60.

INSDAG, 2006. *Teaching Material On Structural Steel Design For Civil / Structural Engineering Students & Faculties.* [Online]
Available at: http://www.steel-insdag.org/NewWebsite/TeachingMaterial/chapter36.pdf
[Accessed 28 04 2012].

MathWorks, 2011. *Global Optimization Toolbox Software Version 3.2 (R2011b).* s.l.:s.n.

MathWorks, 2011. *MATLAB Software Version 7.13 (R2011b),* s.l.: s.n.

Salvadori, M. G., 1955. Lateral buckling of i-beams. *Journal of Structural Division,* pp. 1156-1177.

Sivanandam, S. N. & Deepa, S. N., 2008. *Introduction to Genetic Algorithms.* Berlin: Springer.

Structures, I. f. S.-. a. S., 2011. *DESIGN GUIDELINES FOR CROSS-SECTION AND MEMBER DESIGN ACCORDING TO EUROCODE 3 WITH PARTICULAR FOCUS ON SEMI-COMPACT SECTIONS.* Graz: s.n.

Trahair, N., Bradford, M., Nethercot, D. & Gardner, L., 2007. *The Behaviour and Design of Steel Structures to EC3.* London & New York: Taylor & Francis e-Library.

9 Appendix

9.1 The Matlab – code

```
clc
clear all
close all


% ------------GA parameters-------------
PopSize = 2;            % population size
NumOfGens =2;           % number of generations
MutRate   = 0.05;         % mutation rate

options = gaoptimset('PopulationType', 'doubleVector',...
                     'PlotFcns', {@gaplotbestf},...
                     'Generations', NumOfGens,...
                     'PopulationSize', PopSize,...
                     'TolCon',1,...
                     'StallGenLimit',Inf,...
                     'StallTimeLimit', Inf,...
                     'SelectionFcn', @selectionstochunif,...
                     'FitnessScalingFcn', @fitscalingrank,...
                     'EliteCount', 1,...
                     'CrossoverFraction', 0.8,...
                     'CrossoverFcn', {@crossoverscattered, 1},...
                     'MutationFcn', {@mutationadaptfeasible, MutRate}...
                      );

tic
%--------------------- The max and min marginals of the variables ---------
%-------------- The cross sections
hwL=[50,50,50,50];hwU=[1500,1500,1500,1500];% the height of the web
buL=[50,50,50];buU=[1500,1500,1500];% the width of the upper flange
tuL=[1,1,1];tuU=[100,100,100];% the thickness of the upper flange
twL=[1,1,1];twU=[100,100,100];% the thickness of the web
blL=[50,50,50];blU=[1500,1500,1500];% the width of the lower flange
tlL=[1,1,1];tlU=[100,100,100];% the  thickness of the lower flange

%-------------- The bracings
% Every bracing vector consist of a binary string. For example a string of
8 bits means 8 propabilities of bracings disribution.
BL=[0,0,0,0,0,0];BU=[63,63,63,63,63,63];%the bracings. 6 bits string
bracings. 22 bits string
%
BL=[0,0,0,0,0,0];BU=[4194303,4194303,4194303,4194303,4194303,4194303];%the



% [x,fval,exitflag,output,population] = ga(@(x) test5(x),
19,[],[],[],[],[hwL,buL,tuL,twL,blL,tlL],[hwU,buU,tuU,twU,blU,tlU],[],[1:19
], options);
% [x,fval,exitflag,output,population] = ga(@(x) test5(x),
7,[],[],[],[],[hwL,twL],[hwU,twU],[],[1:7], options);
% [x,fval,exitflag,output,population] = ga(@(x) test5(x),
25,[],[],[],[],[hwL,buL,tuL,twL,blL,tlL,BL],[hwU,buU,tuU,twU,blU,tlU,BU],[]
,[1:25], options);
```

```
% [x,fval,exitflag,output,population] = ga(@(x) test5(x), 9,[],[],[],[],[50
50 50 50 50 1 1 50 1],[1500 1500 1500 1500 1500 100 100 1500
100],[],[1,2,3,4,5,6,7,8,9], options);
% [x,fval,exitflag,output,population] = ga(@(x) test5(x),
10,[],[],[],[],[hwL,BL ],[hwU,BU ],[],[1:10], options);
% [x,fval,exitflag,output,population] = ga(@(x) test5(x), 4,[],[],[],[],[50
50 50 50 ],[1500 1500 1500 1500 ],[],[1,2,3,4], options);
[x,fval,exitflag,output,population] = ga(@(x) penalty(x),
1,[],[],[],[],[200],[800],[],[1], options);
% [x,fval] = ga(@(x) test3(x), 3,[],[],[],[],[],[],[], options);
ET=toc/3600;%elapsed time in minuts

%----------------- Saving the figures ------------------------------
saveas(gcf,['Fittness'])
save('x','x')
save('ET','ET')
run crossssecnonleniar % is running to recalculate the final design and
plot the figures.
saveas(Defshape,['Defshape'])
saveas(Normalforce,['Normalforce'])
saveas(Shear,['Shear'])
saveas(Moment,['Moment'])
saveas(If02,['ShearCheck'])
saveas(If0,['If0'])
saveas(If1,['If1'])
saveas(If2,['If2'])
saveas(CompCls,['CompCls'])
saveas(BenCls,['BenCls'])
saveas(Frame,['Frame'])

%--------------------------- End --------------------------------
```

```
function [z]=penalty(x)
%********************************** penalty
****************************************
%-----------------------------------Purpose-------------------------------
-------------
% Assigning penalty as punishment on the weight of the frame in case any of
% the limitations is violated.
%-----------------------------------Inputs--------------------------------
------------

%-----------------------------------Outputs-------------------------------
-------------
%Fittness (the weight).
%*************************************************************************
*************

E=210e9;fy=235;G=81e9;gammam0=1.0;

% Since many tests have been done, many rows are unactive (commented). Only
certain rows are active depending on the studied case.

%------------------------ The bracings in case of fixed number and
location-------------------
% Long vector of bracing means many subelements and fine mesh.

% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the first element.
% Bracing2=[1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the second element.
% Bracing3=[1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the third element.
% Bracing1=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the first
element.
% Bracing2=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the second
element.
% Bracing3=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the third
element.
% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the first element.
% Bracing2=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the second element.
% Bracing3=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the third element.
Bracing1=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the first element.
Bracing2=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the second element.
Bracing3=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the third element.



%------------------------ The cross sections -----------------------
%----------cross section dimensions----------------
%cross section dimensions at the first and the last edge of each member in
%mm
%[the width of the upper flange, the thickness of the upper flange,
%the hight of the web, the thickness of the web
%the width of the lower flange, the thickness of the lower flange]

%------------------------ 19 variable cross sections ---------------
```

```
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% bu1=x(5);bu2=x(6);bu3=x(7);
% tu1=x(8);tu2=x(9);tu3=x(10);
% tw1=x(11);tw2=x(12);tw3=x(13);
% bl1=x(14);bl2=x(15);bl3=x(16);
% tl1=x(17);tl2=x(18);tl3=x(19);
% Crosssec=[bu1 tu1 hw1 tw1 bl1 tl1
%           bu1 tu1 hw2 tw1 bl1 tl1
%
%           bu2 tu2 hw2 tw2 bl2 tl2
%           bu2 tu2 hw3 tw2 bl2 tl2
%
%           bu3 tu3 hw3 tw3 bl3 tl3
%           bu3 tu3 hw4 tw3 bl3 tl3
%
%
%
%           bu3 tu3 hw4 tw3 bl3 tl3
%           bu3 tu3 hw3 tw3 bl3 tl3
%
%           bu2 tu2 hw3 tw2 bl2 tl2
%           bu2 tu2 hw2 tw2 bl2 tl2
%
%           bu1 tu1 hw2 tw1 bl1 tl1
%           bu1 tu1 hw1 tw1 bl1 tl1]/1000;

%------------------------- Varialbe loacation of the bracings along the
whole frame. --------------
% Every bracing vector consist of a binary string. For example a string of
8 bits means 8 propabilities of bracings disribution.
%-------------------- 6 vectors of bracings 8 binary string each--------
% EB11=de2bi(x(20),8)';EB12=de2bi(x(21),8)';
% EB21=de2bi(x(22),8)';EB22=de2bi(x(23),8)';
% EB31=de2bi(x(24),8)';EB32=de2bi(x(25),8)';
% Bracing1=[EB11,EB12];Bracing2=[EB21,EB22];Bracing3=[EB31,EB32];
%
%------------------------ 7 variable cross sections ---------------
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% bu1=200;bu2=200;bu3=200;
% tu1=25;tu2=25;tu3=25;
% tw1=x(5);tw2=x(6);tw3=x(7);
% bl1=200;bl2=200;bl3=200;
% tl1=25;tl2=25;tl3=25;
% Crosssec=[bu1 tu1 hw1 tw1 bl1 tl1
%           bu1 tu1 hw2 tw1 bl1 tl1
%
%           bu2 tu2 hw2 tw2 bl2 tl2
%           bu2 tu2 hw3 tw2 bl2 tl2
%
%           bu3 tu3 hw3 tw3 bl3 tl3
%           bu3 tu3 hw4 tw3 bl3 tl3
%
%
%
%           bu3 tu3 hw4 tw3 bl3 tl3
%           bu3 tu3 hw3 tw3 bl3 tl3
%
%           bu2 tu2 hw3 tw2 bl2 tl2
%           bu2 tu2 hw2 tw2 bl2 tl2
%
%           bu1 tu1 hw2 tw1 bl1 tl1
```

```
%           bu1 tu1 hw1 tw1 bl1 tl1]/1000;

%-------------------- 6 vectors of bracings 22 binary string each--------
% EB11=de2bi(x(5),22)';EB12=de2bi(x(6),22)';
% EB21=de2bi(x(7),22)';EB22=de2bi(x(8),22)';
% EB31=de2bi(x(9),22)';EB32=de2bi(x(10),22)';
% Bracing1=[EB11,EB12];Bracing2=[EB21,EB22];Bracing3=[EB31,EB32];

%------------------------ 1 variable cross sections ---------------
Crosssec=[200 25 80 25 200 25
          200 25 x 25 200 25

          200 25 x 25 200 25
          200 25 60 25 200 25

          200 25 60 25 200 25
          200 25 60 25 200 25



          200 25 60 25 200 25
          200 25 60 25 200 25

          200 25 60 25 200 25
          200 25 x 25 200 25

          200 25 x 25 200 25
          200 25 80 25 200 25]/1000;

%------------------------ 4 variable cross sections ---------------
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% Crosssec=[200 25 hw1 25 200 25
%           200 25 hw2 25 200 25
%
%           200 25 hw2 25 200 25
%           200 25 hw3 25 200 25
%
%           200 25 hw3 25 200 25
%           200 25 hw4 25 200 25
%
%
%
%           200 25 hw4 25 200 25
%           200 25 hw3 25 200 25
%
%           200 25 hw3 25 200 25
%           200 25 hw2 25 200 25
%
%           200 25 hw2 25 200 25
%           200 25 hw1 25 200 25]/1000;
%
%-------------------- 6 vectors of bracings 6 binary string each--------
% EB11=de2bi(x(5),6)';EB12=de2bi(x(6),6)';
% EB21=de2bi(x(7),6)';EB22=de2bi(x(8),6)';
% EB31=de2bi(x(9),6)';EB32=de2bi(x(10),6)';
% BRACING1=[EB11,EB12];BRACING2=[EB21,EB22];BRACING3=[EB31,EB32];

%the mesh (subelement) number for each main element is decided to be 21.
```

```
% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 1 is equal to the length of the vector
% Bracing2=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 2 is equal to the length of the vector
% Bracing3=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 3 is equal to the length of the vector

%-------------------- 6 vectors of bracings 6 binary string each--------
%putting the bracings on certain positions of the main element.
% for i=1:6
% Bracing1(3*i+1,1)=BRACING1(i,1);Bracing1(3*i+1,2)=BRACING1(i,2);
% Bracing2(3*i+1,1)=BRACING2(i,1);Bracing2(3*i+1,2)=BRACING2(i,2);
% Bracing3(3*i+1,1)=BRACING3(i,1);Bracing3(3*i+1,2)=BRACING3(i,2);
% end

%---------------------- The main coordinates ----------------------------
maincoord=[0 0;0 5;5 5.75;10 6.5;15 5.75;20 5;20 0];

%---------------------- The loads ----------------------------
Eq=[1 0 0
    2 0 -.8e4
    3 0 -.8e4
    4 0 -.8e4
    5 0 -.8e4
    6 0 0];

%----------weight of the frame--------------
weight=0;w=zeros(length(Eq),1);
density=7600;%the density of the steel=7600kg/m^3
for i=1:max(size(maincoord))-1
    x1=maincoord(i,1);y1=maincoord(i,2);
    x2=maincoord(i+1,1);y2=maincoord(i+1,2);
    l=sqrt((x2-x1)^2+(y2-y1)^2);
    A1=Crosssec(2*i-1,1)*Crosssec(2*i-1,2)+Crosssec(2*i-1,3)*Crosssec(2*i-
1,4)+Crosssec(2*i-1,5)*Crosssec(2*i-1,6);

A2=Crosssec(2*i,1)*Crosssec(2*i,2)+Crosssec(2*i,3)*Crosssec(2*i,4)+Crosssec
(2*i,5)*Crosssec(2*i,6);
    weight=weight+l*(A1+A2)/2*density;weight1=l*(A1+A2)/2*density;
    w(i)=weight1;%weight of every main element
end
clear x1;clear x2;clear y1;clear y2;clear A1,clear A2,clear length;

%--------------------ULS---------------------
%in ULS, the ULS load compinations are used and the utilities have to be
%checked
EqULS=Eq;
EqULS(:,3)=w(:,1)*-10*1.35+Eq(:,3)*1.5;%loads compination according to ULS
%the weight multiplied by -10 in order to transfere it from kg to N and (-)
coz it is in downward direction
[Checkk6,IF1,IF2,Compressionclass,Bendingclass,Checkk2]=Nonlinear1(E,fy,G,g
ammam0,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,EqULS);
%--------------------SLS---------------------
%in SLS, the SLS load compinations are used and the max deformation have to
be
%checked
EqSLS=Eq;
```

```
EqSLS(:,3)=w(:,1)*-10*1.0+Eq(:,3)*1.0;%loads compination according to SLS
%the weight multiplied by -10 in order to transfere it from kg to N and (-)
coz it is in downward direction
[ahmax,avmax]=Esnon(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,EqSLS);
%max horizontal and vertical deformation of the frame
SigmaVmax=(maincoord(end,1)-maincoord(1,1))/250;%max allowable vertical
deformation of the frame
SigmaHmax=(maincoord(end,1)-maincoord(1,1))/150;%max allowable horizontal
deformation of the frame

%---------------panelty functions------------
%---------------Checkk2
%adding 100 kg for every 1% violation
G0=0;
for i=1:length(Checkk2)
    if Checkk2(i)>.5
        g=max((Checkk2(i)-.5)*10000,0);

    else
        g=0;
    end
    G0=G0+g;
end

%---------------Checkk6
%adding 100 kg for every 1% violation
G1=0;
for i=1:length(Checkk6)
    if Checkk6(i)>1
        g=max((Checkk6(i)-1)*10000,0);

    else
        g=0;
    end
    G1=G1+g;
end

%---------------IF1
%adding 500 kg for every 1% violation
G2=0;
for i=1:length(IF1)
    if IF1(i)>1
        g=max(   ((IF1(i)-1)*100) *500   ,0);
    else
        g=0;
    end
    G2=G2+g;
end

%--------------IF2
%adding 500 kg for every 1% violation
G3=0;
for i=1:length(IF2)
    if IF2(i)>1
           g=max(   ((IF2(i)-1)*100) *500   ,0);
    else
        g=0;
    end
    G3=G3+g;
end
```

```
%-------------vertical deformation
% Here you can either take the SLS criterion into account or not by
% activate/unactivate G4.
u4=(abs(avmax)/SigmaVmax)*100;
 G4=max((u4-100)*100,0);

 %-------------horizontal deformation
u5=(abs(ahmax)/SigmaHmax)*100;
 G5=max((u5-100)*100,0);

 %-------------compression class
 G6=sum(Compressionclass==4)*1000;

 %-------------bending class
 G7=sum(Bendingclass==4)*1000;

%-------------the bracings
G8=sum(sum([Bracing1,Bracing2,Bracing3]))*2;

 %-----------the penalty
 z=weight+G0+G1+G2+G3+G5+G6+G7+G8;

%----------------------------- End ----------------------------------
```

```
%********************************** crossssecnonleniar
****************************************
%-----------------------------------Purpose----------------------------------
-------------
% Used to run the final design (the optimal) the GA have found, plot
% and save the figures.
%-----------------------------------Inputs-----------------------------------
------------
% x: a vector includes the optimal dimensions/bracings pattern of the
% optimal design. x-vector is loaded automatically form GA by this function
at the end of GA process.
%-----------------------------------Outputs----------------------------------
-------------
% The resultant plots of the optimal frame.
%************************************************************************
*************
close all
clear all
clc
E=210e9;fy=235;G=81e9;gammam0=1.0;

% Since many tests have been done, many rows are unactive (commented). Only
certain rows are active depending on the studied case.

%------------------------ The bracings in case of fixed number and
location-------------------
% Long vector of bracing means many subelements and fine mesh.

% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the first element.
% Bracing2=[1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the second element.
% Bracing3=[1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1;0 0;0 0;0
0;0 0;0 0;1 1;0 0;0 0;0 0;0 0;0 0;1 1];%bracings of the third element.
% Bracing1=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the first
element.
% Bracing2=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the second
element.
% Bracing3=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the third
element.
% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the first element.
% Bracing2=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the second element.
% Bracing3=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0];%bracings of the third element.
Bracing1=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the first element.
Bracing2=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the second element.
Bracing3=[1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1
1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1;1 1];%bracings of the third element.


%------------------------ The cross sections -----------------------
%----------cross section dimensions----------------
%cross section dimensions at the first and the last edge of each member in
%mm
%[the width of the upper flange, the thickness of the upper flange,
%the hight of the web, the thickness of the web
```

```
%the width of the lower flange, the thickness of the lower flange]

load('x')% loading the vector x of the final design.
%------------------------ 19 variable cross sections ---------------
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% bu1=x(5);bu2=x(6);bu3=x(7);
% tu1=x(8);tu2=x(9);tu3=x(10);
% tw1=x(11);tw2=x(12);tw3=x(13);
% bl1=x(14);bl2=x(15);bl3=x(16);
% tl1=x(17);tl2=x(18);tl3=x(19);
% Crosssec=[bu1 tu1 hw1 tw1 bl1 tl1
%           bu1 tu1 hw2 tw1 bl1 tl1
%
%           bu2 tu2 hw2 tw2 bl2 tl2
%           bu2 tu2 hw3 tw2 bl2 tl2
%
%           bu3 tu3 hw3 tw3 bl3 tl3
%           bu3 tu3 hw4 tw3 bl3 tl3
%
%
%
%           bu3 tu3 hw4 tw3 bl3 tl3
%           bu3 tu3 hw3 tw3 bl3 tl3
%
%           bu2 tu2 hw3 tw2 bl2 tl2
%           bu2 tu2 hw2 tw2 bl2 tl2
%
%           bu1 tu1 hw2 tw1 bl1 tl1
%           bu1 tu1 hw1 tw1 bl1 tl1]/1000;

%------------------------- Varialbe loacation of the bracings along the
whole frame. --------------
% Every bracing vector consist of a binary string. For example a string of
8 bits means 8 propabilities of bracings disribution.
%--------------------- 6 vectors of bracings 8 binary string each--------
% EB11=de2bi(x(20),8)';EB12=de2bi(x(21),8)';
% EB21=de2bi(x(22),8)';EB22=de2bi(x(23),8)';
% EB31=de2bi(x(24),8)';EB32=de2bi(x(25),8)';
% Bracing1=[EB11,EB12];Bracing2=[EB21,EB22];Bracing3=[EB31,EB32];
%
%------------------------ 7 variable cross sections ---------------
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% bu1=200;bu2=200;bu3=200;
% tu1=25;tu2=25;tu3=25;
% tw1=x(5);tw2=x(6);tw3=x(7);
% bl1=200;bl2=200;bl3=200;
% tl1=25;tl2=25;tl3=25;
% Crosssec=[bu1 tu1 hw1 tw1 bl1 tl1
%           bu1 tu1 hw2 tw1 bl1 tl1
%
%           bu2 tu2 hw2 tw2 bl2 tl2
%           bu2 tu2 hw3 tw2 bl2 tl2
%
%           bu3 tu3 hw3 tw3 bl3 tl3
%           bu3 tu3 hw4 tw3 bl3 tl3
%
%
%
%           bu3 tu3 hw4 tw3 bl3 tl3
%           bu3 tu3 hw3 tw3 bl3 tl3
%
```

```
%           bu2 tu2 hw3 tw2 bl2 tl2
%           bu2 tu2 hw2 tw2 bl2 tl2
%
%           bu1 tu1 hw2 tw1 bl1 tl1
%           bu1 tu1 hw1 tw1 bl1 tl1]/1000;

%-------------------- 6 vectors of bracings 22 binary string each--------
% EB11=de2bi(x(5),22)';EB12=de2bi(x(6),22)';
% EB21=de2bi(x(7),22)';EB22=de2bi(x(8),22)';
% EB31=de2bi(x(9),22)';EB32=de2bi(x(10),22)';
% Bracing1=[EB11,EB12];Bracing2=[EB21,EB22];Bracing3=[EB31,EB32];

%------------------------ 1 variable cross sections ---------------
Crosssec=[200 25 80 25 200 25
          200 25 x 25 200 25

          200 25 x 25 200 25
          200 25 60 25 200 25

          200 25 60 25 200 25
          200 25 60 25 200 25



          200 25 60 25 200 25
          200 25 60 25 200 25

          200 25 60 25 200 25
          200 25 x 25 200 25

          200 25 x 25 200 25
          200 25 80 25 200 25]/1000;

%------------------------ 4 variable cross sections ---------------
% hw1=x(1);hw2=x(2);hw3=x(3);hw4=x(4);
% Crosssec=[200 25 hw1 25 200 25
%           200 25 hw2 25 200 25
%
%           200 25 hw2 25 200 25
%           200 25 hw3 25 200 25
%
%           200 25 hw3 25 200 25
%           200 25 hw4 25 200 25
%
%
%
%           200 25 hw4 25 200 25
%           200 25 hw3 25 200 25
%
%           200 25 hw3 25 200 25
%           200 25 hw2 25 200 25
%
%           200 25 hw2 25 200 25
%           200 25 hw1 25 200 25]/1000;
%
%-------------------- 6 vectors of bracings 6 binary string each--------
% EB11=de2bi(x(5),6)';EB12=de2bi(x(6),6)';
% EB21=de2bi(x(7),6)';EB22=de2bi(x(8),6)';
% EB31=de2bi(x(9),6)';EB32=de2bi(x(10),6)';
```

```
% BRACING1=[EB11,EB12];BRACING2=[EB21,EB22];BRACING3=[EB31,EB32];

%the mesh (subelement) number for each main element is decided to be 21.
% Bracing1=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 1 is equal to the length of the vector
% Bracing2=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 2 is equal to the length of the vector
% Bracing3=[0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0;0 0;0 0];%the number of the subelements at b the main
element 3 is equal to the length of the vector

%-------------------- 6 vectors of bracings 6 binary string each--------
%putting the bracings on certain positions of the main element.
% for i=1:6
% Bracing1(3*i+1,1)=BRACING1(i,1);Bracing1(3*i+1,2)=BRACING1(i,2);
% Bracing2(3*i+1,1)=BRACING2(i,1);Bracing2(3*i+1,2)=BRACING2(i,2);
% Bracing3(3*i+1,1)=BRACING3(i,1);Bracing3(3*i+1,2)=BRACING3(i,2);
% end



%----------main coordinate of the whole frame----------
% maincoord=[0 0;1 4;4 6;7 7;10 6;13 4;14 0]; %coordinates for the
centerline of the
% elements of the main frame
maincoord=[0 0;0 5;5 5.75;10 6.5;15 5.75;20 5;20 0];

%--------------interpolation of A,I and Z-------------------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
s=sum(devidingsnum);%the total number of the subelements

DevC=zeros(s,6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);
for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=x/2;
    for j=1:devidingsnum(.5*i+.5)



        DevC(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j+x/2;
        n=n+1;
    end
         m=n;
end
clear C1;clear C2;clear m; clear n;clear x;clear x1;clear x2; clear x3;
DevA=zeros(s,1);
DevI=zeros(s,1);DevIz=zeros(s,1);
DevZ=zeros(s,1);
for i=1:s
```

```
bu=DevC(i,1);tu=DevC(i,2);hw=DevC(i,3);tw=DevC(i,4);
bl=DevC(i,5);tl=DevC(i,6);
DevA(i)=bu*tu+hw*tw+bl*tl;
DevZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevA(i);
DevI(i)=bl*tl^3/12+bl*tl*(DevZ(i)-tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevZ(i))^2;
DevIz(i)=bu^3*tu/12+tw^3*hw/12+bl^3*tl/12;
end
clear bu;clear tu;clear hw; clear tw;clear bl; clear tl;
%
DevCSap=[(DevC(:,3)+DevC(:,2)+DevC(:,6)),DevC(:,1),DevC(:,2),DevC(:,4:6)];

%----------EP generation-----------
%section properties E,A and I.
Ep=zeros(s,3);
for i=1:s
    Ep(i,:)=[E DevA(i) DevI(i)];
end

%------------Coord generation-------------
Coord=zeros((s+1),2);
n=1;m=1;
for i =1:elementnum
    x=(maincoord((i+1),1)-maincoord(i,1))/devidingsnum(i);
     y=(maincoord((i+1),2)-maincoord(i,2))/devidingsnum(i);
     for j=1:devidingsnum(i)
         Coord((j+m),1)=maincoord(i,1)+j*x;
         Coord((j+m),2)=maincoord(i,2)+j*y;
         n=n+1;
     end
     m=n;
end
clear m; clear n;

%----------Ex & Ey generation------------
Ex=zeros(s,2);Ey=zeros(s,2);
for i =1:s
    Ex(i,1)=Coord(i,1);Ex(i,2)=Coord(i+1,1);
    Ey(i,1)=Coord(i,2);Ey(i,2)=Coord(i+1,2);
end


%-------------Edof generation----------------
Edof=zeros(s,7);
for i =1:s
    x=3*i-2;
    Edof(i,:)=[i x x+1 x+2 x+3 x+4 x+5];
end
clear x;

%--------------EQ generation--------------
%the loads input for each member.
%  first columb of the matrix Eq refers to the number of the element,
%  second columb of the matrix Eq refers to the loads in the x-direction,
%  first columb of the matrix Eq refers to the loads in the y-direction,

Eq=[1 0 0
    2 0 -8000
    3 0 -8000
    4 0 -8000
```

```
    5 0 -8000
    6 0 0];
Eq1=Eq;%copy the loads in onrder to use them later for calculatino
according to SLS

%----------weight of the frame--------------
weight=0;w=zeros(length(Eq),1);
density=7600;%the density of the steel=7600kg/m^3
for i=1:max(size(maincoord))-1
    x1=maincoord(i,1);y1=maincoord(i,2);
    x2=maincoord(i+1,1);y2=maincoord(i+1,2);
    l=sqrt((x2-x1)^2+(y2-y1)^2);
    A1=Crosssec(2*i-1,1)*Crosssec(2*i-1,2)+Crosssec(2*i-1,3)*Crosssec(2*i-
1,4)+Crosssec(2*i-1,5)*Crosssec(2*i-1,6);

A2=Crosssec(2*i,1)*Crosssec(2*i,2)+Crosssec(2*i,3)*Crosssec(2*i,4)+Crosssec
(2*i,5)*Crosssec(2*i,6);
    weight=weight+l*(A1+A2)/2*density;weight1=l*(A1+A2)/2*density;
    w(i)=weight1;%weight of every main element
end
clear x1;clear x2;clear y1;clear y2;clear A1,clear A2,clear length;

%--------------------ULS---------------------
%in ULS, the ULS load compinations are used and the utilities have to be
%checked
EqULS=Eq;
EqULS(:,3)=w(:,1)*-10*1.35+Eq(:,3)*1.5;%loads compination according to ULS
%the weight multiplied by -10 in order to transfere it from kg to N and (-)
coz it is in downward direction
Eq=EqULS;

%------------Eqlokal, deviding the global distributed vertical load (like
snow load) into two component,
%vertical and horizontal depending on the angel of the element.
Eqlokal=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);%horizontal distance
y1=maincoord(i,2);y2=maincoord(i+1,2);%vertical distance
Eqlokal(i,:)=[Eq(i,1),Eq(i,3)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,3)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the global load times cos(angel), and the
%horizontal component is the global load times sin(angel).
end
clear x1; clear x2; clear y1; clear y2;

%--------------Eqlokalh, deviding the horizontal component of the load that
have been got
%from Eqlokal into to two componenets, vertical and horizontal in case of
inclined
%element
%------------EqWind, deviding the global distributed horizontal load (like
wind load) into two component,
%vertical and horizontal depending on the angel of every main element.
Eqlokalh=zeros(elementnum,3);EqWind=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);
y1=maincoord(i,2);y2=maincoord(i+1,2);
Eqlokalh(i,:)=[Eqlokal(i,1),Eqlokal(i,2)*(y2-y1)/((x2-x1)^2+(y2-
y1)^2)^.5,Eqlokal(i,2)*(x2-x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the horzontal coponent times cos(angel), and the
%horizontal component is the horizontal component times sin(angel).
```

```
%--------------the wind loads
EqWind(i,:)=[Eq(i,1),Eq(i,2)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,2)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
end
clear x1; clear x2; clear y1; clear y2;

%----------EQh, redistributing av the loads that have been got from
Eqlokalh over the sublements
%----------EQWind, redistributing av the loads that have been got from
EqHoriz over the sublements
n=0;m=0;
EQh=zeros(s,2);EQWind=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQh(j+m,1:2)=Eqlokalh(i,2:3);
        EQWind(j+m,1:2)=EqWind(i,2:3);%the wind loads
         n=n+1;
    end
    m=n;
end
clear n;clear m;
Eq=Eqlokal;

%----------EQ, redistributing av the loads that have been got from Eqlokal
over the sublements
n=0;m=0;
EQ=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQ(j+m,1:2)=Eq(i,2:3);
         n=n+1;
    end
    m=n;
end
clear n;clear m;

%------------L generation, the length of the subelements--------
L=zeros(s,1);
for i =1:s
x1=Coord(i,1);y1=Coord(i,2);x2=Coord(i+1,1);y2=Coord(i+1,2);
L(i)=((x2-x1)^2+(y2-y1)^2)^.5;
end
clear x1; clear x2; clear y1; clear y2;

%---------P generation, transfere the distributed loads from EQh to
%vertical and horzontal point loads acting on the nodes of the subelements
pph=0;Phh=zeros(s+1,1);%the horizontal point loads
ppv=0;Pvv=zeros(s+1,1);%the vertical point loads
for i=1:s
    Ph=EQh(i,2)*L(i)/2;
    Phh(i)=Ph+pph;Phh(i+1)=Ph;pph=Ph;

    Pv=EQh(i,1)*L(i)/2;
    Pvv(i)=Pv+ppv;Pvv(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%---------PWind generation, transfere the distributed loads from EQWind to
%vertical and horzontal point loads acting on the nodes of the subelements
```

```
pph=0;PhhWind=zeros(s+1,1);%the horizontal point loads
ppv=0;PvvWind=zeros(s+1,1);%the vertical point loads
for i=1:s
    Ph=EQWind(i,2)*L(i)/2;
    PhhWind(i)=Ph+pph;PhhWind(i+1)=Ph;pph=Ph;

    Pv=EQWind(i,1)*L(i)/2;
    PvvWind(i)=Pv+ppv;PvvWind(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%----------Inetial values for the iteration-----
eps=0.0001;
N=zeros(s,1);N(1)=.01;
N0=ones(s,1);
n=0;

%--------------bc-----------------------
bc=[1 0;2 0;Edof(end,5) 0;Edof(end,6) 0];% here it is assumed that the both
supports of
%the frame is fixed.

%----------Es for first order theory, calclation of the inner forces from
the
%first order theory that can be used then to calculate the forces due to
%imperfection effect.
[Es]=Es(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,Eq);
[Esedge]=Esedge(Bracing1,Bracing2,Bracing3,Crosssec,Es);

% %---------------Imperfection for global analysis of frames EC3 5.3.2-----
----
[NN1,NN2]=imperfection(maincoord,Bracing1,Bracing2,Bracing3,Esedge);

%**************************************************************************
****************
%------------------------------Iteration procedure-------------------------
------------------
while (abs((N(1)-N0(1))/N0(1))>eps)
    n=n+1;


%----------puting the nodal forces on the accual nodes------
f=zeros(Edof(end),1);
for i=1:s+1

f(3*i-2)=Phh(i)+PvvWind(i);%puting the vertincal point loads on the nodes
with vertical degree of freedom
f(3*i-1)=Pvv(i)+PhhWind(i);%puting the horzontal point loads on the nodes
with horizontal degree of freedom

end

%extra horizontal force added on the upper edge of the both columns caused
%by the effect of the imperfection
i1=devidingsnum(1)*3+1;
i2=(sum(devidingsnum)-devidingsnum(length(devidingsnum)))*3+1;%the location
of the horizontal DOF of the upper edge of the both columns
f(i1)=f(i1)+NN1*0;f(i2)=f(i2)+NN2*0;%puting the extra forces of the
imperfection at the actual nodes.
```

```
%------------K generation-----------------
K=zeros(Edof(end));
for i=1:s
    [Ke,fe]=beam2g(Ex(i,:),Ey(i,:),Ep(i,:),N(i),EQ(i,2));
    [K,f]=assem(Edof(i,:),K,Ke,f,fe);
end

%--------------bc, Ed and a----------------------
% bc=[1 0;2 0;3 0;Edof(end,5) 0;Edof(end,6) 0;Edof(end,7) 0];% here it is
assumed that the both supports of
%the frame is fixed.
a=solveq(K,f,bc);
Ed=extract(Edof,a); %the displacements ux,uy and r locally for each member.



%--------------stresses calculation--------------------
Es=zeros(s*2,3);
for i=1:s
  Es((2*i-
1):(2*i),:)=beam2gs(Ex(i,:),Ey(i,:),Ep(i,:),Ed(i,:),N(i),EQ(i,2));
end
N0=N;
for i=1:2:s*2;
    N((.5*i+.5),1)=Es(i,1);% calculating the normal forces for every first
    % beam edge and save them in N-matrix to reuse them in the next
iteration.
end

if (n>20)
    disp('The solution doesn''t converge')
    break
end
end
%*********************************END************************************

%------------finding the max horizontal and vertical displacements of the
frame
for i=1:3:length(a)-4
    if abs(a(i+3))>abs(a(i))
        ahmax1=a(i+3);% max horzontal displacement.
    end
    if abs(a(i+4))>abs(a(i+1))
        avmax1=a(i+4);%max vertical displacement
    end
end
ahmax=ahmax1;avmax=avmax1;
%%
%-----------------Eurocode checks------------------
%ECcheck is a function to check the cross sections accodring to EC3 for
%steel. Check1 is a check for axial force capacity. Check2 is a check for
%shear capacity. Check3 is a check for bending moment capacity. Check6 is a
%check for all the three capacities above the axial, the shear and the
%bending togather. Every check is saved in a matrix for every submember,
%for exampel Check1 is a matrix (or a vector) with dimension of [the number
of subelements *1]
%each row of this matrix refers to a submemeber with indication of either
%one or zero.
```

```
%One means thet the submember has passed the check or
%zero which means that the submemeber has not pass the check.
%
[Esedge,D,DevedgesA,DevedgesZ,DevedgesI]=Esedge(devidingsnum,Crosssec,Es);

%---------------D, devided cross section for each subelement---------
D=zeros(s+length(devidingsnum),6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);

for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=0;
    for j=1:devidingsnum(.5*i+.5)+1


        D(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j;
        n=n+1;
    end
         m=n;
end
clear C1;clear C2;clear m; clear n;clear x;clear x1; clear x2; clear x3;
mm=size(D);nn=mm(1);
DevedgesA=zeros(nn,1);
DevedgesI=zeros(nn,1);
DevedgesZ=zeros(nn,1);
for i=1:nn
bu=D(i,1);tu=D(i,2);hw=D(i,3);tw=D(i,4);
bl=D(i,5);tl=D(i,6);
DevedgesA(i)=bu*tu+hw*tw+bl*tl;
DevedgesZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevedgesA(i)
;
DevedgesI(i)=bl*tl^3/12+bl*tl*(DevedgesZ(i)-
tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevedgesZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevedgesZ(i))^2;
end
%---------------Esedge generation,Es at the edges-----------------
mm=size(Crosssec);nn=mm(1);
Esedge=zeros(nn*2,3);
n=0;m=0;n1=0;m1=0;
for i=1:elementnum
    for j=1:devidingsnum(i)

        Esedge(j+m1,:)=Es(2*j+m-1,:);
        n=2*j+m-1;n1=n1+1;
    end

    Esedge(j+m1+1,:)=Es(2*j+m,:);n=n+1;n1=n1+1;
    m=n;m1=n1;
end
clear n;clear m; clear n1;clear m1;

[Compressionclass,Bendingclass]=Class(D,fy,Esedge);
[NRd,VRd,MRd,WW]=resistance(D,DevedgesA,DevedgesZ,DevedgesI,Bendingclass,ga
mmam0,fy);
```

```
[Checkk6,Checkk2]=ECcheck(NRd,VRd,MRd,Crosssec,Bracing1,Bracing2,Bracing3,E
sedge);
% %----------claculation of the mean value of utilization that calculated
in Check 6 at the both edges of the subelement.
% Checkk6=zeros(s,1);m=0;m1=0;n=0;m2=0;
% for i=1:length(devidingsnum)
%     for j=1:devidingsnum(i)
%         Checkk6(j+m1)=(Check6(j+m+m2)+Check6(j+m+m2+1))/2;
%         n=n+1;
%     end
%     m=n;m1=n;m2=m2+1;
% end


%%
%---------------check buckling against compression in z-direction----------
----------------
%---------Ncr in the z-direction----------
[Xiz,Lambdaz]=bucklingz(Bracing1,Bracing2,Bracing3,DevA,DevIz,E,L,Crosssec,
fy);


%%
%------------------checking the lateral torsional buckling around the major
axis--------------------------
[XILT1,LLvector,McrVector]=LTbuckling1(Bracing1,Bracing2,Bracing3,DevedgesI
,Esedge,L,D,WW,E,G,fy);


%%
%----------------interaction formela-----------------------
Xiy=1.0;% Xi,y the reduction factor duo to bending in y-direction assumed
to be =1.0 because it is already included in the imperfection calculations.
[IF1,IF2,kyy,kzy]=Interaction(Bracing1,Bracing2,Bracing3,Esedge,NRd,MRd,Ben
dingclass,Lambdaz,Xiz,Xiy,XILT1);
%

%--------------plot the deformed and undeformed shape according to SLS-----
---
%---------------------SLS----------------------
%in SLS, the SLS load compinations are used and the max deformation have to
be
%checked
EqSLS=Eq1;
EqSLS(:,3)=w(:,1)*-10*1.0+Eq1(:,3)*1.0;%loads compination according to SLS
%the weight multiplied by -10 in order to transfere it from kg to N and (-)
coz it is in downward direction
SLS(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,EqSLS);
Defshape=gcf;

%--------------plot the forces------------------
%--------------plot the normal forces---------------
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,1);Es(1,1)]);
for i=1:s
    eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,1);Es(2*i,1)],plotpar,10e-6);
    text(Coord(i,1)+.5,Coord(i,2),num2str(round(Es(2*i-1,1)/1000)));
end
text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(round(Es(2*i,1)/1000)));
```

```
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
title('Normal forces diagram kN')
Normalforce=gcf;

%--------------plot the shear forces-----------------
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,2);Es(1,2)]);
for i=1:s
    eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,2);Es(2*i,2)],plotpar,10e-6);
        text(Coord(i,1)+.5,Coord(i,2),num2str(round(Es(2*i-1,2)/1000)));
end
text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(round(Es(2*i,2)/1000)));
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
title('Shear forces diagram kN')
Shear=gcf;

%---------------plot the moment--------------------
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
for i=1:s
%     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
    eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)],plotpar,8e-6);
        text(Coord(i,1)+.5,Coord(i,2),num2str(round(Es(2*i-1,3)/1000)));
end
text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(round(Es(2*i,3)/1000)));
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
title('Moment diagram kNm')
Moment=gcf;

%----------------------ploting the utilities----------------------
%---------------plot Checkk6, utilization of the axial and the bending
moment (%)------
Checkk6=round(Checkk6*1000)/10;
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
for i=1:s
%     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
    eldia2(Ex(i,:),Ey(i,:),[Checkk6(i)*-1;Checkk6(i)*-1],plotpar,16e-3);
        text(Coord(i,1)+.5,Coord(i,2),num2str(Checkk6(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Checkk6(i)));
text(0,9,['Max value = ',num2str(max(Checkk6)),' %'])
title('Checkk6, utilization of the axial and the bending moment (%)')
If0=gcf;

%---------------plot Checkk2, utilization of the shear  (%)------
Checkk2=round(Checkk2*1000)/10;
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
for i=1:s
%     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
    eldia2(Ex(i,:),Ey(i,:),[Checkk2(i)*-1;Checkk2(i)*-1],plotpar,16e-3);
```

```
        text(Coord(i,1)+.5,Coord(i,2),num2str(Checkk2(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Checkk6(i)));
text(0,9,['Max value = ',num2str(max(Checkk2)),' %'])
title('Checkk2, utilization of  the shear  (%)')
If02=gcf;

%-------------Plot IF1, interaction formula 1 EC 6.3.3 (%)-------------
IF1=round(IF1*1000)/10;
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
for i=1:s
%     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
    eldia2(Ex(i,:),Ey(i,:),[IF1(i)*-1;IF1(i)*-1],plotpar,16e-3);
        text(Coord(i,1)+.5,Coord(i,2),num2str(IF1(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Checkk6(i)));
text(0,9,['Max value = ',num2str(max(IF1)),' %'])
title('IF1, interaction formula 1 EC 6.3.3 (%)')
If1=gcf;

%--------------plot IF2, interaction formula 2 EC 6.3.3 (%)--------------
IF2=round(IF2*1000)/10;
figure
  plotpar=[2 1];
%   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
for i=1:s
%     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
    eldia2(Ex(i,:),Ey(i,:),[IF2(i)*-1;IF2(i)*-1],plotpar,16e-3);
        text(Coord(i,1)+.5,Coord(i,2),num2str(IF2(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Checkk6(i)));
text(0,9,['Max value = ',num2str(max(IF2)),' %'])
title('IF2, interaction formula 2 EC 6.3.3 (%)')
If2=gcf;

%-------------------section classes plot--------------------------------
%
Classplot(Compressionclass,Bendingclass,devidingsnum,Ex,Ey,Coord,maincoord)
n=0;n1=0;
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)+1
        if j<devidingsnum(i)+1
            n=n+1;
            CompClassplot(n)=Compressionclass(j+n1);
            BendClassplot(n)=Bendingclass(j+n1);
        end

    end
    n1=j+n1;
end
clear n; clear n1;
CompClassplot=CompClassplot';
%--------------plot Compression Class--------------
```

```
figure
plotpar=[2 1];
for i=1:s
    eldia2(Ex(i,:),Ey(i,:),[CompClassplot(i)*-1;CompClassplot(i)*-
1],plotpar,16e-2);
        text(Coord(i,1)+.5,Coord(i,2),num2str(CompClassplot(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
title('Compression class')
CompCls=gcf;

BendClassplot=BendClassplot';
%--------------plot Bending Class--------------
figure
plotpar=[2 1];
for i=1:s
    eldia2(Ex(i,:),Ey(i,:),[BendClassplot(i)*-1;BendClassplot(i)*-
1],plotpar,16e-2);
        text(Coord(i,1)+.5,Coord(i,2),num2str(BendClassplot(i)));
end
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
title('Bending class')
BenCls=gcf;

%--------------------frame plot-----------------------

frameplot(Bracing1,Bracing2,Bracing3,weight,Crosssec,maincoord)
Frame=gcf;

% %**********************Check plot*************************************
% %----------------------NRd,VRd & MRd vectors adjusting------------------
----------
% n=0;n1=0;
% for i=1:length(devidingsnum)
%     for j=1:devidingsnum(i)+1
%         if j<devidingsnum(i)+1
%             n=n+1;
%             NRD(n)=NRd(j+n1);
%             VRD(n)=VRd(j+n1);
%             MRD(n)=MRd(j+n1);
%         end
%
%     end
%     n1=j+n1;
% end
% clear n; clear n1;
% NRD=NRD';VRD=VRD';MRD=MRD';
%
% %-------------------NRD plot-------------
% figure
% NRD=round(NRD/1000);
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[NRD(i)*-1;NRD(i)*-1],plotpar,16e-4);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(NRD(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('NRD kN')
```

```
%
% %-------------------VRD plot-------------
% figure
% VRD=round(VRD/1000);
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[VRD(i)*-1;VRD(i)*-1],plotpar,16e-4);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(VRD(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('VRD kN')
%
% %-------------------MRD plot-------------
% figure
% MRD=round(MRD/1000);
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[MRD(i)*-1;MRD(i)*-1],plotpar,16e-4);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(MRD(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('MRD kNm')
%
% %-------------------Xiz plot-------------
% figure
% % Xiz=Xiz;
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[Xiz(i)*-1;Xiz(i)*-1],plotpar,16e-1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(Xiz(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('Xiz ')
%
% %-------------------Lambdaz plot-------------
% figure
% % Xiz=Xiz;
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[Lambdaz(i)*-1;Lambdaz(i)*-1],plotpar,16e-1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(Lambdaz(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('Lambdaz ')
%
% %-------------------LLvector plot-------------
% figure
% % Xiz=Xiz;
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[LLvector(i)*-1;LLvector(i)*-1],plotpar,16e-
1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(LLvector(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('LLvector ')
```

```
%
% %-------------------McrVector plot-------------
% figure
% McrVector=round(McrVector/1000);
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[McrVector(i)*-1;McrVector(i)*-1],plotpar,16e-
4);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(McrVector(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('McrVector kNm')
%
% %-------------------kyy plot-------------
% figure
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[kyy(i)*-1;kyy(i)*-1],plotpar,16e-1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(kyy(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('kyy ')
%
% %-------------------kzy plot-------------
% figure
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[kzy(i)*-1;kzy(i)*-1],plotpar,16e-1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(kzy(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('kzy ')
%
% %-------------------kzy plot-------------
% figure
% plotpar=[2 1];
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[XILT1(i)*-1;XILT1(i)*-1],plotpar,16e-1);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(XILT1(i)));
% end
% axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
% title('XILT1 ')

%------------------------- End -------------------------------
```

```
function
[Checkk6,IF1,IF2,Compressionclass,Bendingclass,Checkk2]=Nonlinear1(E,fy,G,g
ammam0,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,Eq)

%********************************** Nonlinear1
**************************************
%-----------------------------------Purpose-------------------------------
-------------
% Collecting the most important functions in one funtion to be called by
% the objective/fitness fucntion.
%-----------------------------------Inputs--------------------------------
------------
%E: the elasticity of steel.
%fy: yield strength of steel.
%G: the shear modulus of steel.
%gammam0: a safety factor.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%maincoord: the coordinates at the main nodes of the frame.
%Eq: the horizontal and vertical loads. Uniformly distributed loads applied
over the 6 main elements.
%-----------------------------------Outputs-------------------------------
-------------
%Checkk2: a vector includes the shear check for each subelement.
%IF1: a vector includes the utilization of the interaction formula 1 for
each subelement.
%IF2: a vector includes the utilization of the interaction formula 1 for
each subelement.
%Compressionclass: a vector of the compression class of each subelement.
%Bendingclass: a vector of the bending class of each subelement.
%Checkk6: a vactor includes the interaction formula of axial and moment for
each subelement.
%**************************************************************************
*************

%----------------- Nonlinearly calculating the internal forces ------------
---------------------------------
[ahmax,avmax,Esnon1,DevA,DevI,L]=Esnon(E,Bracing1,Bracing2,Bracing3,Crossse
c,maincoord,Eq);
[Esedge1,D,DevedgesA,DevedgesZ,DevedgesI]=Esedge(Bracing1,Bracing2,Bracing3
,Crosssec,Esnon1);

%----------------- Eurocode check-----------------------------------------
---------------------------------
[Compressionclass,Bendingclass]=Class(D,fy,Esedge1);
[NRd,VRd,MRd,WW]=resistance(D,DevedgesA,DevedgesZ,DevedgesI,Bendingclass,ga
mmam0,fy);
[Checkk6,Checkk2]=ECcheck(NRd,VRd,MRd,Crosssec,Bracing1,Bracing2,Bracing3,E
sedge1);

% %--------------- Check buckling against compression in z-direction ------
---------------------------------
[Xiz,Lambdaz]=bucklingz(Bracing1,Bracing2,Bracing3,DevA,DevI,E,L,Crosssec,f
y);

% %--------------- Checking the lateral torsional buckling around the major
axis -----------------------------
```

```
XILT1=LTbuckling1(Bracing1,Bracing2,Bracing3,DevedgesI,Esedge1,L,D,WW,E,G,f
y);%

% %--------------- Interaction formela ------------------------
Xiy=1.0;% Xi,y the reduction factor duo to bending in y-direction assumed
to be =1.0 because it is already included in the imperfection calculations.
[IF1,IF2]=Interaction(Bracing1,Bracing2,Bracing3,Esedge1,NRd,MRd,Bendingcla
ss,Lambdaz,Xiz,Xiy,XILT1);
```

```
function
[Xiz,Lambdaz]=bucklingz(Bracing1,Bracing2,Bracing3,DevA,DevIz,E,L,Crosssec,
fy)

%*********************************
bucklingz*************************************
%-----------------------------------Purpose-------------------------------
-------------
%Check buckling against compression in z-direction
%-----------------------------------Inputs--------------------------------
------------
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%DevA: a vector of the average sub-area of each subelement.
%DevIz: a vector of the average sub-moment-of-intertia-in z-direction of
each subelement.
%E: the elasticity of steel.
%L: a vector of lengths of the 6 main elements.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%fy: yield strength of steel.
%-----------------------------------Outputs-------------------------------
-------------
%Xiz: reduction factor for buckling in z-direction.
%Lambdaz: the non-dimensional slenderness, lambda according to EC3 section
6.3.1.3
%**********************************************************************
*************


%---------Ncr in the z-direction----------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
s=sum(devidingsnum);%the total number of the subelements

Bracing=[Bracing1;Bracing2;Bracing3;flipud(Bracing3);flipud(Bracing2);flipu
d(Bracing1)];%the distribution of the bracing is symetric, it means that
frist half of the frame is similar to the second.

m=0;n=0;m1=0;n1=0;l=0;o1=0;
Ncr=zeros(s,1);betaz=1.0;

for i=1:length(devidingsnum)

    for j=1:devidingsnum(i)+1

       if j~=devidingsnum(i)+1
            l=l+L(j+m1);  %the length of main element to be checked.
            if o1==0;I1=DevIz(j+m1);o1=1;end
          if Bracing(j+m+1,1)==1 || Bracing(j+m+1,2)==1 ||
j==devidingsnum(i)   %the edge of the element ends when the sign of the
moment changes or by the existing of an external bracing or by the end fo
the main element cos it is assumed as a bracing also.
              I2=DevIz(j+m1);I=(I1+I2)/2;o1=0;
             Ncr(j+m1)=pi^2*E*I/(betaz*l)^2;
             l=0;

          end
```

```
       end

      n=n+1;
      if j~=devidingsnum(i)+1;n1=n1+1;end%some indications belongs to the
length of the element
    end

    m=n;m1=n1;l=0;
end
clear m;clear n;clear m1;clear n1;clear o2;clear l;clear I1;clear I2;clear
I;

%---------storing Ncr in a matrix of the number of the subelements. every
%element has the same Ncr value along the checked length.
Ncr1=zeros(s,1);
o=1;
for i=1:s

        if Ncr(i)~=0
            for v=o:i
                Ncr1(v)=Ncr(i);
            end
        o=i+1;
        end

end
clear o;

%-------------find the buckling curve and inperfection factors in z-
direction for each of the subelements-----
%finding which of the five buckling curves according to EC3 Table 6.2
%for welded I-section in z-direction if the thickness of the flange tf
%<=40mm then it is bukling curve c.
%if tf > 40mm, then it is curve d.
%to find the imperfection factors according to EC3 Table 6.1 which depend
%on which buckling curve.
Buckilngcurvez=zeros(length(devidingsnum),1);Imperfectionfactor=zeros(lengt
h(devidingsnum),1);
for i=1:2:length(devidingsnum)*2
    if Crosssec(i,2)<=.04
        Buckilngcurvez(.5*i+.5)=4;%because we have five buckling curves,
then every curve will get a number from 1-5.
        Imperfectionfactor(.5*i+.5)=.49;%the imperfectionfactor depends on
which buckling cruve according to EC3 Table 6.1.
    else
        Buckilngcurvez(.5*i+.5)=5;
        Imperfectionfactor(.5*i+.5)=.76;
    end
end

n=0;m=0;
DevBuckilngcurvez=zeros(sum(devidingsnum),1);DevImperfectionfactor=zeros(su
m(devidingsnum),1);
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)
    DevBuckilngcurvez(j+m)=Buckilngcurvez(i);
    DevImperfectionfactor(j+m)=Imperfectionfactor(i);
    n=n+1;
    end
```

```
    m=n;
end

%the non-dimensional slenderness, lambda according to EC3 section 6.3.1.3
%lambda=sqrt(A*fy/Ncr)
Lambdaz=zeros(s,1);
for i=1:s
    Lambdaz(i)=sqrt(DevA(i)*fy*1e6/Ncr1(i));
end

%--------------------
%the reduction factor for axial force capacity, Chi of the main memebers in
%z-direction according to EC3 section 6.3.1.2
Xiz=zeros(s,1);
for i=1:s
    phi=.5*(1+DevImperfectionfactor(i)*(Lambdaz(i)-.2)+Lambdaz(i)^2);
    Xiz(i)=1/(phi+sqrt(phi^2-Lambdaz(i)^2));
    if  Xiz(i)>=1
         Xiz(i)=1;
    end
end

%----------------------------------End-----------------------------------
```

```
function [Compressionclass,Bendingclass]=Class(D,fy,Esedge)

%********************************** Class
****************************************
%-----------------------------------Purpose-------------------------------
-------------
% Finding the classification of the cross sections
%-----------------------------------Inputs--------------------------------
------------
%D:a matrix includes the sub cross sections for each subelement.
%fy: yield strength of steel.
%Esedge: the internal forces at the both edges of the subelements.
%-----------------------------------Outputs-------------------------------
-------------
%Compressionclass: a vector of the compression class of each subelement.
%Bendingclass: a vector of the bending class of each subelement.
%*************************************************************************
*************

mm=size(D);nn=mm(1);
epsilon=sqrt(235/fy);
Bendingclass=zeros(nn,1);%a matrix to save the classes of cross sections in
bending.
Compressionclass=zeros(nn,1);%a matrix to save the classes of cross
sections in compression.
for i=1:nn
    bu=D(i,1);tu=D(i,2);hw=D(i,3);tw=D(i,4);bl=D(i,5);tl=D(i,6);
    if Esedge(i,3)>0 %only the compressed flange must be checked.
        %the upper flange is compressed if the momemt is positive and vice
versa.

        b=bu;t=tu;
    else b=bl;t=tl;
    end
    %the cross section in bending
    %web in bending
    if hw/tw>124*epsilon
        cbw=4; %the class of the web in bending.
    elseif hw/tw<=124*epsilon && hw/tw>83*epsilon
        cbw=3;
    elseif hw/tw<=83*epsilon && hw/tw>72*epsilon
        cbw=2;
    else cbw=1;
    end
    %flange in compression

    if ((b/2-tw/2)/2)/t>14*epsilon
        cbf=4; %the class of the web in compression.
    elseif ((b/2-tw/2)/2)/t<=14*epsilon && ((b/2-tw/2)/2)/t>10*epsilon
        cbf=3;
    elseif ((b/2-tw/2)/2)/t<=10*epsilon && ((b/2-tw/2)/2)/t>9*epsilon
        cbf=2;
    else cbf=1;
    end
    Bendingclass(i)=max(cbw,cbf);
    %the cross section in compression
    %web in compression
    if hw/tw>42*epsilon
        ccw=4; %the class of the web in bending.
    elseif hw/tw<=42*epsilon && hw/tw>38*epsilon
        ccw=3;
```

```
    elseif hw/tw<=38*epsilon && hw/tw>33*epsilon
        ccw=2;
    else ccw=1;
    end
    %flange in compression
    if ((b/2-tw/2)/2)/t>14*epsilon
        ccf=4; %the class of the web in compression.
    elseif ((b/2-tw/2)/2)/t<=14*epsilon && ((b/2-tw/2)/2)/t>10*epsilon
        ccf=3;
    elseif ((b/2-tw/2)/2)/t<=10*epsilon && ((b/2-tw/2)/2)/t>9*epsilon
        ccf=2;
    else ccf=1;
    end
    Compressionclass(i)=max(ccw,ccf);

end

%------------------------------End---------------------------------------
```

```
function
[Checkk6,Checkk2]=ECcheck(NRd,VRd,MRd,Crosssec,Bracing1,Bracing2,Bracing3,E
sedge)

%********************************** ECcheck
**************************************
%-----------------------------------Purpose----------------------------------
-------------
%Checking the capacity of the frame with respect to the normal force, shear
and bending
%-----------------------------------Inputs-----------------------------------
------------
%NRd: a vector of the capacity of each subelements against normal force.
%VRd: a vector of the capacity of each subelements against shear force.
%MRd: a vector of the capacity of each subelements against moment force.
%Crossseca: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Esedge: the internal forces at the both edges of the subelements.
%-----------------------------------Outputs----------------------------------
-------------
%Checkk6: a vactor includes the interaction formula of axial and moment for
each subelement.
%Checkk2: a vector includes the shear check for each subelement.
%**************************************************************************
*************

devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
s=sum(devidingsnum);%the total number of the subelements
% elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
D=zeros(s+length(devidingsnum),6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);

for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=0;
    for j=1:devidingsnum(.5*i+.5)+1


        D(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j;
        n=n+1;
    end
         m=n;
end
mm=size(D);nn=mm(1);
DevedgesA=zeros(nn,1);
DevedgesI=zeros(nn,1);
DevedgesZ=zeros(nn,1);
```

```
for i=1:nn
bu=D(i,1);tu=D(i,2);hw=D(i,3);tw=D(i,4);
bl=D(i,5);tl=D(i,6);
DevedgesA(i)=bu*tu+hw*tw+bl*tl;
DevedgesZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevedgesA(i)
;
DevedgesI(i)=bl*tl^3/12+bl*tl*(DevedgesZ(i)-
tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevedgesZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevedgesZ(i))^2;
end

%-----------Check 1, check the normal capacity--------
mm=size(D);nn=mm(1);
Check1=zeros(nn,1);%check 1 for the axial capacity
for i=1:nn
   if abs(NRd(i))>abs(Esedge(i,1)) %Check 1 passes if NRd>NEd
       Check1(i)=1;
   else
       Check1(i)=0;%Check 1 doesnt pass if NRd<NEd
   end
end

%-----------Check 2, check the shear capacity--------
mm=size(D);nn=mm(1);
Check2=zeros(nn,1);%check 2 for the shear capacity
for i=1:nn
    Check2(i)=abs(Esedge(i,2))/VRd(i);
%    if abs(VRd(i))>abs(Esedge(i,2)) %Check 2 pass if VRdVEd
%        Check2(i)=1;
%    else
%        Check2(i)=0;%Check 2 doesnt pass if VRd<VEd
%    end
end

Checkk2=zeros(s,1);m=0;m1=0;n=0;m2=0;
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)
        Checkk2(j+m1)=(Check2(j+m+m2)+Check2(j+m+m2+1))/2;
        n=n+1;
    end
    m=n;m1=n;m2=m2+1;
end

%-----------Check 3, check the moment capacity--------
mm=size(D);nn=mm(1);
Check3=zeros(nn,1);%check 3 for the moment capacity
for i=1:nn
   if abs(MRd(i))>abs(Esedge(i,3)) %Check 3 pass if MRd>MEd
       Check3(i)=1;
   else
       Check3(i)=0;%Check 2 doesnt pass if MRd<MEd
   end
end

%-----------Check 4 combination between bending and shear------
% mm=size(D);nn=mm(1);
% Check4=zeros(nn,1);%Check shear and bending
% for i=1:nn
%     if abs(Esedge(i,2))<abs(VRd(i))/2% The effect on the moment
resistance caused by the
```

```
%         %shear my be neglected if VEd<VRd/2
%         Check4(i)=1;% check4 is passed
%     else
%         Check4(i)=0;%check 4 is not passed
%         rho=(2*Esedge(i,2)/VRd(i)-1)^2;% reduction for the moment
capacity, rho must be done
%
% %         MRd(i)=(Wpl(i)-rho*(D(i,3)*D(i,4))^2/4/D(i,4))*fy/gammam0;%
reduction for the moment capacity
%     end
% end

%-----------Check 6 combination between bending, shear and axial------
mm=size(D);nn=mm(1);
Check6=zeros(nn,1);
for i=1:nn
%     if abs(Esedge(i,2))>abs(VRd(i))/2 && abs(VRd(i))>abs(Esedge(i,2))%
The effect on the moment resistance caused by the
%         %shear my be neglected if VEd<VRd/2
%         rho=(2*abs(Esedge(i,2))/VRd(i)-1)^2;% reduction for the moment
capacity, rho must be done
%         MRd(i)=(1-rho)*MRd(i);%the moment capacity reduced by (1-rho) if
VEd>VRd/2
%     end
%         if (abs(Esedge(i,1)/NRd(i))+abs(Esedge(i,3)/MRd(i)))<=1% this
check include axial, shear and the bending capacity
%             %Check 6 passes if (NEd/NRd)+(MEd/MRd)<=1
%             Check6(i)=1;%Check 6 passes if (NEd/NRd)+(MEd/MRd)<=1
%
%         else
%         Check6(i)=0;%Check 6 not passes if (NEd/NRd)+(MEd/MRd)>=1
%         end
        Check6(i)=(abs(Esedge(i,1)/NRd(i))+abs(Esedge(i,3)/MRd(i)));
end

Checkk6=zeros(s,1);m=0;m1=0;n=0;m2=0;
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)
        Checkk6(j+m1)=(Check6(j+m+m2)+Check6(j+m+m2+1))/2;
        n=n+1;
    end
    m=n;m1=n;m2=m2+1;
end

%-----------------------------------End-----------------------------------
```

```
function
[Es,DevA,DevZ,DevI,L,weight]=Es(E,Bracing1,Bracing2,Bracing3,Crosssec,mainc
oord,Eq)

%********************************** Es
****************************************
%-----------------------------------Purpose-------------------------------
-------------
% Linearly calculation of the internal forces of each subelements.
%-----------------------------------Inputs--------------------------------
------------
%E: the elasticity of steel.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%maincoord: the coordinates at the main nodes of the frame.
%Eq: the horizontal and vertical loads. Uniformly distributed loads applied
over the 6 main elements.
%-----------------------------------Outputs-------------------------------
-------------
%Es: the internal forces of each subelement generated by the CALFEM-fuction
beam2s.
%DevA: a vector of the average area of each subelement.
%DevZ: a vector of the average ceter of gravity of each subelement.
%DevI: a vector of the average moment of intertia about y-direction of each
subelement.
%L: a vector of lengths of the 6 main elements.
%weight: a vector contains the weight of the main 6 elements.
%**************************************************************************
*************

%----------weight of the frame--------------
weight=0;
density=7600;%the density of the steel=7600kg/m^3
for i=1:max(size(maincoord))-1
    x1=maincoord(i,1);y1=maincoord(i,2);
    x2=maincoord(i+1,1);y2=maincoord(i+1,2);
    l=sqrt((x2-x1)^2+(y2-y1)^2);
    A1=Crosssec(2*i-1,1)*Crosssec(2*i-1,2)+Crosssec(2*i-1,3)*Crosssec(2*i-
1,4)+Crosssec(2*i-1,5)*Crosssec(2*i-1,6);

A2=Crosssec(2*i,1)*Crosssec(2*i,2)+Crosssec(2*i,3)*Crosssec(2*i,4)+Crosssec
(2*i,5)*Crosssec(2*i,6);
    weight=weight+l*(A1+A2)/2*density;
end
clear x1;clear x2;clear y1;clear y2;clear A1;clear A2;clear l;

%--------------interpolation of A,I and Z-------------------
% devidingsnum=[5,2,2,2,2,5];
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.

elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
s=sum(devidingsnum);%the total number of the subelements
```

```
DevC=zeros(s,6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);
for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=x/2;
    for j=1:devidingsnum(.5*i+.5)


        DevC(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j+x/2;
        n=n+1;
    end
         m=n;
end
clear C1;clear C2;clear m; clear n;
DevA=zeros(s,1);
DevI=zeros(s,1);DevIz=zeros(s,1);
DevZ=zeros(s,1);
for i=1:s
bu=DevC(i,1);tu=DevC(i,2);hw=DevC(i,3);tw=DevC(i,4);
bl=DevC(i,5);tl=DevC(i,6);
DevA(i)=bu*tu+hw*tw+bl*tl;
DevZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevA(i);
DevI(i)=bu*tl^3/12+bl*tl*(DevZ(i)-tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevZ(i))^2;
DevIz(i)=bu^3*tu/12+tw^3*hw/12+bl^3*tl/12;
end


%----------EP generation-----------
%section properties E,A and I.
Ep=zeros(s,3);
for i=1:s
    Ep(i,:)=[E DevA(i) DevI(i)];
end

%------------Coord generation-------------
Coord=zeros((s+1),2);
n=1;m=1;
for i =1:elementnum
    x=(maincoord((i+1),1)-maincoord(i,1))/devidingsnum(i);
     y=(maincoord((i+1),2)-maincoord(i,2))/devidingsnum(i);
     for j=1:devidingsnum(i)
         Coord((j+m),1)=maincoord(i,1)+j*x;
         Coord((j+m),2)=maincoord(i,2)+j*y;
         n=n+1;
     end
     m=n;
end
clear m; clear n;

%------------L generation, the length of the subelements--------
L=zeros(s,1);
for i =1:s
x1=Coord(i,1);y1=Coord(i,2);x2=Coord(i+1,1);y2=Coord(i+1,2);
L(i)=((x2-x1)^2+(y2-y1)^2)^.5;
```

```
end

%----------Ex & Ey generation------------
Ex=zeros(s,2);Ey=zeros(s,2);
for i =1:s
    Ex(i,1)=Coord(i,1);Ex(i,2)=Coord(i+1,1);
    Ey(i,1)=Coord(i,2);Ey(i,2)=Coord(i+1,2);
end


%-------------Edof generation----------------
Edof=zeros(s,7);
for i =1:s
    x=3*i-2;
    Edof(i,:)=[i x x+1 x+2 x+3 x+4 x+5];
end

%----------------------------------------
Eqlokal=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);
y1=maincoord(i,2);y2=maincoord(i+1,2);
Eqlokal(i,:)=[Eq(i,1),Eq(i,3)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,3)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
end
Eq=Eqlokal;
n=0;m=0;
EQ=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQ(j+m,1:2)=Eq(i,2:3);
         n=n+1;
    end
    m=n;
end
clear m; clear n;
% EQ(:,1)=0;
%------------K generation-----------------
K=zeros(Edof(end));f=zeros(Edof(end),1);
for i=1:s
    [Ke,fe]=beam2e(Ex(i,:),Ey(i,:),Ep(i,:),EQ(i,:));
    [K,f]=assem(Edof(i,:),K,Ke,f,fe);
end

%--------------bc, Ed and a-----------------------
bc=[1 0;2 0;Edof(end,5) 0;Edof(end,6) 0];% here it is assumed that the both
supports of
% the frame is fixed.
a=solveq(K,f,bc);
Ed=extract(Edof,a);

%--------------stresses calculation---------------------
Es=zeros(s*2,3);
for i=1:s
  Es((2*i-1):(2*i),:)=beam2s(Ex(i,:),Ey(i,:),Ep(i,:),Ed(i,:),EQ(i,:),2);
end

%--------------------------- End ----------------------------------------
```

```
function
[Esedge,D,DevedgesA,DevedgesZ,DevedgesI]=Esedge(Bracing1,Bracing2,Bracing3,
Crosssec,Es)
%********************************** Esedge
****************************************
%-----------------------------------Purpose----------------------------------
-------------
% Calculation of the internal forces at the edge of each subelements.
%-----------------------------------Inputs-----------------------------------
------------
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%Es: the internal forces of each subelement generated by the CALFEM-fuction
beam2s.
%-----------------------------------Outputs----------------------------------
-------------
%Esedge: calculate the internal forces at the edge of each subelements.
%D: a matrix includes the sub cross sections for each subelement.
%DevedgesA: a vector of the average area at the edge of each subelement.
%DevedgesZ: a vector of the average ceter of gravity at the edge of each
subelement.
%DevedgesI: a vector of the average moment of intertia about y-direction at
the edge of each subelement.
%*************************************************************************
*************

devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
s=sum(devidingsnum);%the total number of the subelements
elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
D=zeros(s+length(devidingsnum),6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);

for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=0;
    for j=1:devidingsnum(.5*i+.5)+1


        D(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j;
        n=n+1;
    end
         m=n;
end
clear C1;clear C2;clear m; clear n;
mm=size(D);nn=mm(1);
DevedgesA=zeros(nn,1);
DevedgesI=zeros(nn,1);
DevedgesZ=zeros(nn,1);
```

```
for i=1:nn
bu=D(i,1);tu=D(i,2);hw=D(i,3);tw=D(i,4);
bl=D(i,5);tl=D(i,6);
DevedgesA(i)=bu*tu+hw*tw+bl*tl;
DevedgesZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevedgesA(i)
;
DevedgesI(i)=bu*tl^3/12+bl*tl*(DevedgesZ(i)-
tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevedgesZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevedgesZ(i))^2;
end

%----------------Esedge generation,Es at the edges-----------------
mm=size(Crosssec);nn=mm(1);
Esedge=zeros(nn*2,3);
n=0;m=0;n1=0;m1=0;
for i=1:elementnum
    for j=1:devidingsnum(i)

        Esedge(j+m1,:)=Es(2*j+m-1,:);
        n=2*j+m-1;n1=n1+1;
    end

    Esedge(j+m1+1,:)=Es(2*j+m,:);n=n+1;n1=n1+1;
    m=n;m1=n1;
end
clear n;clear m; clear n1;clear m1;

%---------------------------------- End ----------------------------------
```

```
function frameplot(Bracing1,Bracing2,Bracing3,weight,Crosssec,maincoord)
%********************************** resistance
****************************************

%-----------------------------------SLS---------------------------------
---------
% Plotting the whole frame. The plot includes a front ,
% side and top view of the frame. It shows also the bracings distribution
% pattern in form of B-letters. A table of the dimensions of the cross
% sections are also included.
%-----------------------------------Inputs-------------------------------
------------
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%weight: a vector contains the weight of the main 6 elements.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%maincoord: the coordinates at the main nodes of the frame.
%-----------------------------------Outputs-----------------------------
-------------
%Frame plot.
%************************************************************************
***********

devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
B1=flipud(Bracing1);B2=flipud(Bracing2);B3=flipud(Bracing3);
Bracing=[Bracing1(1:max(size(Bracing1))-
1,:);Bracing2(1:max(size(Bracing2))-1,:);Bracing3(1:max(size(Bracing3))-
1,:); B3(1:length(B3)-1,:);B2(1:length(B2)-1,:);B1(1:length(B1)-1,:)];%the
distribution of the bracing is symetric, it means that frist half of the
frame is similar to the second.
elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
s=sum(devidingsnum);%the total number of the subelements

maincoord=maincoord*100;
Crosssec=Crosssec*100;
n=0;m=0;n1=1;n2=0;
mm=size(Crosssec);nn=mm(1);
for i=1:2:nn

bu1=Crosssec(i,1);tu1=Crosssec(i,2); hw1=Crosssec(i,3); tw1=Crosssec(i,4);
bl1=Crosssec(i,5); tl1=Crosssec(i,6);
bu2=Crosssec(i+1,1);tu2=Crosssec(i+1,2); hw2=Crosssec(i+1,3);
tw2=Crosssec(i+1,4); bl2=Crosssec(i+1,5); tl2=Crosssec(i+1,6);

z1=(bu1*tu1*(tl1+hw1+tu1/2)+hw1*tw1*(hw1/2+tl1)+bl1*tl1*(tl1/2))/(bu1*tu1+h
w1*tw1+bl1*tl1);
z2=(bu2*tu2*(tl2+hw2+tu2/2)+hw2*tw2*(hw2/2+tl2)+bl2*tl2*(tl2/2))/(bu2*tu2+h
w2*tw2+bl2*tl2);
h1=(tu1+hw1+tl1);h2=(tu2+hw2+tl2);

x1=maincoord(.5*i+.5,1);y1=maincoord(.5*i+.5,2);
x2=maincoord(.5*i+1.5,1);y2=maincoord(.5*i+1.5,2);
```

```
y11=y1-z1*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);          x11=x1+z1*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y12=y1-(z1-tl1)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);    x12=x1+(z1-tl1)*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y13=y1+(h1-z1-tu1)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2); x13=x1-(h1-z1-
tu1)*(y2-y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y14=y1+(h1-z1)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);     x14=x1-(h1-z1)*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);

y21=y2-z2*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);          x21=x2+z2*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y22=y2-(z2-tl2)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);    x22=x2+(z2-tl2)*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y23=y2+(h2-z2-tu2)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2); x23=x2-(h2-z2-
tu2)*(y2-y1)/sqrt((x2-x1)^2+(y2-y1)^2);
y24=y2+(h2-z2)*(x2-x1)/sqrt((x2-x1)^2+(y2-y1)^2);     x24=x2-(h2-z2)*(y2-
y1)/sqrt((x2-x1)^2+(y2-y1)^2);

figure(30)
%------------ploting the side view of the frame
l1=line([x11,x21],[y11,y21]);set(l1,'LineStyle','-','Color','c');
l2=line([x12,x22],[y12,y22]);set(l2,'LineStyle','-','Color','c');%ploting
the lower flange
l3=line([x13,x23],[y13,y23]);l4=line([x14,x24],[y14,y24]);%ploting the
upper flange
l5=line([x1,x2],[y1,y2]);set(l5,'LineStyle','-.','Color','r')%ploting the
centre line

S=(maincoord(end,1)/100-maincoord(1,1)/100);
%------------ploting the top view of the frame
Vlevel=800;%vertical level
l6=line([x1,x2],[Vlevel,Vlevel]);set(l6,'LineStyle','-
.','Color','r')%ploting the centre line
l7=line([x1,x2],[Vlevel-bu1/2,Vlevel-
bu2/2]);l8=line([x1,x2],[Vlevel+bu1/2,Vlevel+bu2/2]);%ploting the upper
flange
l9=line([x1,x2],[Vlevel-bl1/2,Vlevel-bl2/2]);set(l9,'LineStyle','--
','Color','c');l10=line([x1,x2],[Vlevel+bl1/2,Vlevel+bl2/2]);set(l10,'LineS
tyle','--','Color','c');%ploting the lower flange
l11=line([x1,x2],[Vlevel-tw1/2,Vlevel-tw2/2]);set(l11,'LineStyle','--
','Color','r');l12=line([x1,x2],[Vlevel+tw1/2,Vlevel+tw2/2]);set(l12,'LineS
tyle','--','Color','r');%ploting the web

%------------ploting the right side view of the frame
n2=n2+1;
RHlevel=-S*10;%right horizontal level
if n2<=elementnum/2
l13=line([RHlevel,RHlevel],[y1,y2]);set(l13,'LineStyle','-
.','Color','r')%ploting the centre line
l14=line([RHlevel-bu1/2,RHlevel-
bu2/2],[y1,y2]);l15=line([RHlevel+bu1/2,RHlevel+bu2/2],[y1,y2]);%ploting
the upper flange
l16=line([RHlevel-bl1/2,RHlevel-bl2/2],[y1,y2]);set(l16,'LineStyle','--
','Color','c');l17=line([RHlevel+bl1/2,RHlevel+bl2/2],[y1,y2]);set(l17,'Lin
eStyle','--','Color','c');%ploting the lower flange
l18=line([RHlevel-tw1/2,RHlevel-tw2/2],[y1,y2]);set(l18,'LineStyle','--
','Color','r');l19=line([RHlevel+tw1/2,RHlevel+tw2/2],[y1,y2]);set(l19,'Lin
eStyle','--','Color','r');%ploting the web
end
```

```
%------------ploting the left side view of the frame
RHlevel=S*115;%left horizontal level
if n2>elementnum/2
l13=line([RHlevel,RHlevel],[y1,y2]);set(l13,'LineStyle','-
.','Color','r')%ploting the centre line
l14=line([RHlevel-bu1/2,RHlevel-
bu2/2],[y1,y2]);l15=line([RHlevel+bu1/2,RHlevel+bu2/2],[y1,y2]);%ploting
the upper flange
l16=line([RHlevel-bl1/2,RHlevel-bl2/2],[y1,y2]);set(l16,'LineStyle','--
','Color','c');l17=line([RHlevel+bl1/2,RHlevel+bl2/2],[y1,y2]);set(l17,'Lin
eStyle','--','Color','c');%ploting the lower flange
l18=line([RHlevel-tw1/2,RHlevel-tw2/2],[y1,y2]);set(l18,'LineStyle','--
','Color','r');l19=line([RHlevel+tw1/2,RHlevel+tw2/2],[y1,y2]);set(l19,'Lin
eStyle','--','Color','r');%ploting the web
end

%------------ploting the bracings
X=(x21-x11)/devidingsnum(.5*i+.5);Y=(y21-y11)/devidingsnum(.5*i+.5);
X1=(x24-x14)/devidingsnum(.5*i+.5);Y1=(y24-y14)/devidingsnum(.5*i+.5);
for j=1:devidingsnum(.5*i+.5)
    n1=n1+1;
    if n1~=length(Bracing)+1
    if Bracing(n1,1)==1;text((x14+j*X1),(y14+j*Y1),'B');end%ploting the
extrnal bracings
    if Bracing(n1,2)==1; text((x11+j*X),(y11+j*Y),'B');end%ploting the
internal bracings
    end
end

end

%------------table plot-------------
H = gcf;
dat = Crosssec*10;
cnames = {'bu','tu','hw','tw','bl','tl'};
rnames = {'1','2','3','4','5','6','7','8','9','10','11','12'};
t =
uitable('Parent',H,'Data',dat,'ColumnName',cnames,'RowName',rnames,'Positio
n',[700 150 490 240]);

%--------------text plot------------
%-----------side view text
mm=size(maincoord);nn=mm(1);
mm1=size(Crosssec);nn1=mm1(1);
maincoord1=zeros(nn1,2);

maincoord1(1,:)=maincoord(1,:);maincoord1(end,:)=maincoord(end,:);

text(maincoord1(1,1),maincoord1(1,2),num2str(1))%side view text for the
first cross section
text(maincoord1(end,1)-S,maincoord1(end,2),num2str(nn1))%side view text for
the last cross section

text(maincoord1(1,1),Vlevel-S*2,num2str(1))%top view text for the first
cross section
text(maincoord1(end,1),Vlevel-S*2,num2str(nn1))%top view text for the last
cross section
n=0;
for i=2:(nn-1)
n=i-1;
```

```
maincoord1(n*2,:)=maincoord(n+1,:);
maincoord1(n*2+1,:)=maincoord(n+1,:);

text(maincoord1(n*2,1),maincoord1(n*2+1,2),num2str(n*2))%side view text
text(maincoord1(n*2,1)+S*2,maincoord1(n*2+1,2)-S*2,num2str(n*2+1))%side
view text

text(maincoord1(n*2,1),Vlevel+S*2,num2str(n*2))%top view text
text(maincoord1(n*2,1)+30,Vlevel+S*2,num2str(n*2+1)) %top view text
end



text(-400,Vlevel,'top view')
text(-400,110,'side view')

title(['*****total weight ',num2str(round(weight)),' kg********'])

%---------------------------- End --------------------------------
```

```
function
[NN1,NN2]=imperfection(maincoord,Bracing1,Bracing2,Bracing3,Esedge)

%********************************** imperfection
**************************************
%-----------------------------------Purpose-------------------------------
-------------
% Calculation of the Imperfection for global analysis of frame EC3 5.3.2
%-----------------------------------Inputs--------------------------------
------------
%maincoord: the coordinates at the main nodes of the frame.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Esedge: calculate the internal forces at the edge of each subelements.
%-----------------------------------Outputs-------------------------------
-------------
%NN1: the imperfection in form of point load acting on the upper edge of
the first column of the frame.
%NN2: the imperfection in form of point load acting on the upper edge of
the second column of the frame.
%*************************************************************************
*************

%---------------Imperfection for global analysis of frames EC3 5.3.2-------
--
%phi=phi0*alphah*alpham (formula 5.5 EC3)
phi0=1/200;%the base value
hh=max(maincoord);h=hh(1,2);%h=the hight of the frame
alphah=2/sqrt(h);%alphah=2/sqrt(h), but 2/3<=alphah<=1,0
if alphah <=2/3%2/3<=alphah<=1,0
    alphah=2/3;
elseif alphah >=1.0
    alphah=1.0;
end
m=2;%the number of the colums (2 in our structure)
alpham=sqrt(.5*(1+1/m));
phi=phi0*alphah*alpham;%imperfection factor

%--------------finding the max axial force in columns
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
m=0;
for i=1:length(devidingsnum)
    if i==1  || i==length(devidingsnum) %the first and the last element of
the structure are columns
        j2=devidingsnum(i)+m+1;%the axial force at the last edge of the
column
        j1=j2-devidingsnum(i);%the axial force at the first edge of the
column
        maxN=max(Esedge(j1,1),Esedge(j2,1));%the max axial force at the
edges of the column
        if i==1
            maxN1=maxN;%max axial force for column 1
        else maxN2=maxN;%max axial force for column 2
        end
    end
    m=devidingsnum(i)+m+1;
```

```
end
clear m;
%extra horizontal force added on the upper edge of the both columns caused
%by the effect of the imperfection
NN1=phi*maxN1;NN2=phi*maxN2;

%---------------------------------End-----------------------------------
```

```
function
[IF1,IF2,kyy,kzy]=Interaction(Bracing1,Bracing2,Bracing3,Esedge,NRd,MRd,Ben
dingclass,Lambdaz,Xiz,Xiy,XILT1)

%********************************** Interaction
**************************************

%-----------------------------------Purpose----------------------------------
-------------
% Calculation of the interaction formula 1 and 2, EC3  (6.61).
%-----------------------------------Inputs-----------------------------------
------------
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Esedge: calculate the internal forces at the edge of each subelements.
%NRd: a vector of the capacity of each subelements against normal force.
%MRd: a vector of the capacity of each subelements against moment force.
%Bendingclass: a vector of the bending class of each subelement.
%Lambdaz: the non-dimensional slenderness, lambda according to EC3 section
6.3.1.3
%Xiz: reduction factor for buckling in z-direction.
%Xiy: reduction factor for buckling in y-direction.
%XILT1: reduction factor for the lateral torsional buckling.
%-----------------------------------Outputs----------------------------------
-------------
%IF1: a vector includes the utilization of the interaction formula 1 for
each subelement.
%IF2: a vector includes the utilization of the interaction formula 1 for
each subelement.
%kyy: a vector includes interaction factor of each subelement.
%kzy: a vector includes interaction factor of each subelement.
%**************************************************************************
***********

%----------------interaction formela-----------------------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
s=sum(devidingsnum);%the total number of the subelements
IF1=zeros(s,1);IF2=zeros(s,1);
m=0;n=0;m1=0;
for i=1:length(devidingsnum)

    for j=1:devidingsnum(i)

 psi=Esedge(j+1+m,3)/Esedge(j+m,3);
      if abs(psi)>1
          psi=1/psi;
      end
%---------linear moment diagram
      Cm=.6+.4*psi;
      if Cm <=.4% Cm =0,6+0,4psi >=0,4
      Cm=.4;
      end

%------------NED, NRD, MED and MRD
NED=(Esedge(j+m+1,1)+Esedge(j+m,1))/2;%mean value of Ned at the both edges
of the elemnt.
```

```
NED=abs(NED);
NRD=(NRd(j+m+1)+NRd(j+m))/2;%mean value of Nrd at the both edges of the
elemnt.
MED=(Esedge(j+m+1,3)+Esedge(j+m,3))/2;%mean value of Med at the both edges
of the elemnt.
MRD=(MRd(j+m+1)+MRd(j+m))/2;%mean value of Mrd at the both edges of the
elemnt.

%-------------Bending cross section class
if Bendingclass(j+m)>Bendingclass(j+m+1)%bending cross section class taken
as the largest of the cross section classes at the both edges of the
element to be in the safe side.
    Bclass=Bendingclass(j+m);
else Bclass=Bendingclass(j+m+1);
end

kyy(j+m1)=Cm;
%---------kyy & kzz Table B.1 EC3: interaction factors kij.
if Bclass == 1|| Bclass == 2%for bending cross section class 1 and 2.
%     kyy(j+m1)=min( Cm*(1+(Lambdaz(j+m1)-.2)*NED/(Xiz(j+m1)*NRD)) ,
Cm*(1+.8*NED/(Xiz(j+m1)*NRD))     );
%     kyy(j+m1)=Cm*(1+(Lambdaz(j+m1)-.2)*NED/(Xiz(j+m1)*NRD));
%     if kyy(j+m1)> Cm*(1+.8*NED/(Xiz(j+m1)*NRD));
kyy(j+m1)=Cm*(1+.8*NED/(Xiz(j+m1)*NRD));end
     kzy(j+m1)=.6*kyy(j+m1);
else%for bending cross section class 3 and 4.
%     kyy(j+m1)=min( Cm*(1+.6*Lambdaz(j+m1)*NED/(Xiz(j+m1)*NRD)) ,
Cm*(1+.6*NED/(Xiz(j+m1)*NRD))       );
%     kyy(j+m1)=Cm*(1+.6*Lambdaz(j+m1)*NED/(Xiz(j+m1)*NRD));
%     if kyy(j+m1)> Cm*(1+.8*NED/(Xiz(j+m1)*NRD));
kyy(j+m1)=Cm*(1+.6*NED/(Xiz(j+m1)*NRD));end
     kzy(j+m1)=.8*kyy(j+m1);
end

%----------------interaction formela
IF1(j+m1)=abs(NED)/(Xiy*NRD)+kyy(j+m1)*abs(MED)/(XILT1(j+m1)*MRD);%interact
ion formula 1 EC3  (6.61)
IF2(j+m1)=abs(NED)/(Xiz(j+m1)*NRD)+kzy(j+m1)*abs(MED)/(XILT1(j+m1)*MRD);%in
teraction formula 1 EC3  (6.61)

      n=n+1;
    end
    m=n+1;m1=n;
end
kyy=kyy';
kzy=kzy';

%------------------------------ End -----------------------------------
```

```
function
[XILT1,LLvector,McrVector]=LTbuckling1(Bracing1,Bracing2,Bracing3,DevedgesI
,Esedge,L,D,WW,E,G,fy)

%********************************** LTbuckling1
****************************************

%-----------------------------------Purpose----------------------------------
-------------
% Check against lateral torsional buckling.
%-----------------------------------Inputs-----------------------------------
------------
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%DevedgesI: a vector of the average moment of intertia about y-direction at
the edge of each subelement.
%Esedge: calculate the internal forces at the edge of each subelements.
%L: a vector of lengths of the 6 main elements.
%D: a matrix includes the sub cross sections for each subelement.
%WW: a vector includes the avarage values of the sectional modulus of the
first and the second edge of the subelement.
%E: the elasticity of steel.
%G: the shear modulus of steel.
%fy: yield strength of steel.
%-----------------------------------Outputs----------------------------------
-------------
%XILT1: reduction factor for the lateral torsional buckling.
%LLvector: an optional vector used to check the function.
%McrVector: an optional vector used to check the function.
%*************************************************************************
***********

%-----------------checking the lateral torsional buckling around the major
axis--------------------------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
% PsiC1=[1 1;.75 1.141;.5 1.323;.25 1.563;0 1.879;-.25 2.281;-.5 2.281;-.75
2.927;-1 2.752];%Table 1 C constants for members not loaded directly.
s=sum(devidingsnum);%the total number of the subelements
mm=size(D);nn=mm(1);
XILT=zeros(nn,1);
Bracing=[Bracing1;Bracing2;Bracing3;flipud(Bracing3);flipud(Bracing2);flipu
d(Bracing1)];%the distribution of the bracing is symetric, it means that
frist half of the frame is similar to the second.
m=0;n=0;m1=0;n1=0;o=0;l=0;o2=0;
for i=1:length(devidingsnum)

    for j=1:devidingsnum(i)+1
        %--------in case of positive moment or braced compressed flange----
       if Esedge(j+m,3)>=0&&j~=devidingsnum(i)+1
           if o2==1; l=l+L(j+m1-1)+L(j+m1); else l=l+L(j+m1);end %the
length of element to be checked.
            if o2==1;  bu1=D(j+m-1,1);tu1=D(j+m-1,2);hw1=D(j+m-
1,3);tw1=D(j+m-1,4);bl1=D(j+m-1,5);tl1=D(j+m-
1,6);I1=(tu1*bu1^3)/12+(hw1*tw1^3)/12+(tl1*bl1^3)/12;W1=WW(j+m-
1);o2=0;o=1;%storing the properties of the first edge of the elemnt that is
going to be checkeed.
```

```
            elseif
o~=1;bu1=D(j+m,1);tu1=D(j+m,2);hw1=D(j+m,3);tw1=D(j+m,4);bl1=D(j+m,5);tl1=D
(j+m,6);I1=DevedgesI(j+m);W1=WW(j+m);o=1;%when the sign of the moment
changes in the middle of two nodes, it is better to start the next elemnt
form the previous node to be in the safe side.
            end
          if (Esedge(j+m+1,3)<=0||Bracing(j+m+1,1)==1||j==devidingsnum(i))
&& j~=devidingsnum(i)+1%the edge of the element ends when the sign of the
moment changes or by the existing of an external bracing or by the end fo
the main element cos it is assumed as a bracing also.
              if Esedge(j+m+1,3)<=0&&Bracing(j+m+1,2)==0;o2=0;end%an
indication that the sign of the moment is changing. there is no need to
take in the account the previous node if the next node is braced becose the
previous element has already checked before the changing of the moment.


bu2=D(j+m+1,1);tu2=D(j+m+1,2);hw2=D(j+m+1,3);tw2=D(j+m+1,4);bl2=D(j+m+1,5);
tl2=D(j+m+1,6);%storing the properties of the second edge of the elemnt
that is going to be checkeed.

bu=(bu1+bu2)/2;tu=(tu1+tu2)/2;hw=(hw1+hw2)/2;tw=(tw1+tw2)/2;bl=(bl1+bl2)/2;
tl=(tl1+tl2)/2;%calculating the avarage values of the dimensions of the
first and the second edge of the element.

I2=(tu2*bu2^3)/12+(hw2*tw2^3)/12+(tl2*bl2^3)/12;I=(I1+I2)/2;%calculating
the avarage values of the moment of inertia about the minor axis of the
first and the second edge of the element.
        W2=WW(j+m+1);W=(W1+W2)/2;%calculating the avarage values of the
sectional modulus of the first and the second edge of the element.

        %-------------It, St. Venants torsion constant of the cross section
for themain elements
        It=(1/3)*(bu*tu^3+hw*tw^3+bl*tl^3);
        %-------------Bf=(Ifc)/(Ifc+Ift)
        Ifc=bu^3*tu/12;Ift=bl^3*tl/12;Bf=(Ifc)/(Ifc+Ift);
%         %-------------Shear Center, e
%         e=tu*bu^3*(tu/2+hw+tl/2)/(tu*bu^3+tl*bl^3);
        %-------------Iw, warping constant
        hs=tu/2+hw+tl/2;%distance between shear centre of the two flanges
of the cross section
        Iw=(1-Bf)*Bf*I*hs^2;
%         %-----------finding the values of C1 depending on the shape of
the moment diagram.
%         psi=Esedge(j+m,3)/Esdege(j+m+1,3);if
psi>1;psi=Esedge(j+m+1,3)/Esdege(j+m,3);end
%          for k=1:length(PsiC1)-1
%           if psi>=PsiC1(k)&&psi<PsiC1(k+1)
%
x1=PsiC1(k,1);x2=psi;x3=PsiC1(k+1,1);y1=PsiC1(k,2);y3=PsiC1(k+1,2);
%             C1=y1+(y3-y1)*(x2-x1)/(x3-x1);
%           end
%          end


Mcr=pi^2*E*I/l^2*sqrt(Iw/I+G*It*l^2/pi^2/E/I);Lvector(j+m)=(l);Mcrvector(j+
m)=(Mcr);

        %-------------find the buckling curve and inperfection factors for
lateral torsional buckling for each of the subelements-----
        %according to Table 6.4 Ec3 for welded I-sections if h/b<=2 then
        %the buckling curve is c otherwise if h/b>2, then the buckling
```

```
        %curve is d.
    if bu/(tu+hw+tl)<=2
%         Buckilngcurve=3;%because we have five buckling curves, then every
curve will get a number from 1-5.
        Imperfectionfactor=.49;%the imperfectionfactor depends on which
buckling cruve according to EC3 Table 6.1.
    else
%         Buckilngcurvez=4;
        Imperfectionfactor=.76;
    end

         %---------the reduction factor according to EC3 6.3.2.2
         lambdaLt=sqrt(W*fy*10^6/Mcr);%lambda for lateral torsional
buckling.
         phiLt=.5*(1+Imperfectionfactor*(lambdaLt-.2)+lambdaLt^2);% Phi for
lateral torsional buckling.
         XiLt=1/(phiLt+sqrt(phiLt^2-lambdaLt^2));%Xi LT, reduction factor
for lateral torsional buckling.
         if XiLt>1;XiLt=1; end% Xi LT <= 1,0
         XILT(j+m)=(XiLt);
         o=0; l=0;

          end

        %--------in case of negative moment and not braced compressed
flange
       elseif j~=devidingsnum(i)+1
           if o2==1; l=l+L(j+m1-1)+L(j+m1); else l=l+L(j+m1);end %the
length of element to be checked.
            if o2==1;  bu1=D(j+m-1,1);tu1=D(j+m-1,2);hw1=D(j+m-
1,3);tw1=D(j+m-1,4);bl1=D(j+m-1,5);tl1=D(j+m-1,6);I1=DevedgesI(j+m-
1);W1=WW(j+m-1);o2=0; o=1;%storing the properties of the first edge of the
elemnt that is going to be checkeed.
            elseif
o~=1;bu1=D(j+m,1);tu1=D(j+m,2);hw1=D(j+m,3);tw1=D(j+m,4);bl1=D(j+m,5);tl1=D
(j+m,6);I1=(tu1*bu1^3)/12+(hw1*tw1^3)/12+(tl1*bl1^3)/12;W1=WW(j+m);o=1;%whe
n the sign of the moment changes in the middle of two nodes, it is better
to start the next elemnt form the previous node to be in the safe side.
            end
          if (Esedge(j+m+1,3)>=0||Bracing(j+m+1,2)==1||j==devidingsnum(i))
&& j~=devidingsnum(i)+1%the edge of the element ends when the sign of the
moment changes or by the existing of an external bracing or by the end fo
the main element cos it is assumed as a bracing also.
              if Esedge(j+m+1,3)>=0&&Bracing(j+m+1,1)==0; o2=0;end%an
indication that the sign of the moment is changing. there is no need to
take in the account the previous node if the next node is braced becose the
previous element has already checked before the changing of the moment.


bu2=D(j+m+1,1);tu2=D(j+m+1,2);hw2=D(j+m+1,3);tw2=D(j+m+1,4);bl2=D(j+m+1,5);
tl2=D(j+m+1,6);%storing the properties of the second edge of the elemnt
that is going to be checkeed.

bu=(bu1+bu2)/2;tu=(tu1+tu2)/2;hw=(hw1+hw2)/2;tw=(tw1+tw2)/2;bl=(bl1+bl2)/2;
tl=(tl1+tl2)/2;%calculating the avarage values of the dimensions of the
first and the second edge of the element.

I2=(tu2*bu2^3)/12+(hw2*tw2^3)/12+(tl2*bl2^3)/12;I=(I1+I2)/2;%calculating
the avarage values of the moment of inertia about the minor axis of the
first and the second edge of the element.
```

```
          W2=WW(j+m+1);W=(W1+W2)/2;%calculating the avarage values of the
sectional modulus of the first and the second edge of the element.
          %-------------It, St. Venants torsion constant of the cross
section for themain elements
          It=(1/3)*(bu*tu^3+hw*tw^3+bl*tl^3);
          %-------------Bf=(Ifc)/(Ifc+Ift)
          Ift=bu^3*tu/12;Ifc=bl^3*tl/12;Bf=(Ifc)/(Ifc+Ift);
          %-------------Iw, warping constant
          hs=tu/2+hw+tl/2;
          Iw=(1-Bf)*Bf*I*hs^2;


Mcr=pi^2*E*I/l^2*sqrt(Iw/I+G*It*l^2/pi^2/E/I);Lvector(j+m)=(l);Mcrvector(j+
m)=(Mcr);

          %-------------find the buckling curve and inperfection factors
for lateral torsional buckling for each of the subelements-----
        %according to Table 6.4 Ec3 for welded I-sections if h/b<=2 then
        %the buckling curve is c otherwise if h/b>2, then the buckling
        %curve is d.
    if bu/(tu+hw+tl)<=2
%         Buckilngcurve=3;%because we have five buckling curves, then every
curve will get a number from 1-5.
        Imperfectionfactor=.49;%the imperfectionfactor depends on which
buckling cruve according to EC3 Table 6.1.
    else
%         Buckilngcurvez=4;
        Imperfectionfactor=.76;
    end


         %---------the reduction factor according to EC3 6.3.2.2
         lambdaLt=sqrt(W*fy*10^6/Mcr);%lambda for lateral torsional
buckling.
         phiLt=.5*(1+Imperfectionfactor*(lambdaLt-.2)+lambdaLt^2);% Phi for
lateral torsional buckling.
         XiLt=1/(phiLt+sqrt(phiLt^2-lambdaLt^2));%Xi LT, reduction factor
for lateral torsional buckling.
         if XiLt>1;XiLt=1; end% Xi LT <= 1,0
          XILT(j+m)=(XiLt);
        o=0; l=0;

          end

       end

      n=n+1;
      if j~=devidingsnum(i)+1;n1=n1+1;end%some indications belongs to the
length of the element
    end

    m=n;m1=n1;o=0;l=0;
end
clear m;clear n;clear m1;clear n1;clear o;clear o2;clear l;
% XILT=[0;.9;0;0;.8;0;0;.7;0;.6;.5;0  ;0;.5;0;0;.6;0;0;0;0;.7;.8;0];
%---------storing XiLt in a matrix of the number of the subelements. every
%element has the same XiLt value along the checked length.
XILT1=zeros(s,1);
m=0;n=0;o=1;m1=0;n1=0;
for i=1:length(devidingsnum)
```

```
    for j=1:devidingsnum(i)
        if XILT(j+m)~=0
            for v=o:j
                XILT1(v+m1)=XILT(j+m);
            end
        o=j+1;
        end
        n=n+1;n1=n1+1;
    end
    m=n+1;o=1;m1=n1; n=n+1;
end
clear m;clear n;clear m1;clear n1;clear o;

%---------storing Lvector in a matrix of the number of the subelements.
every
%element has the same Lvector value along the checked length.
LLvector=zeros(s,1);
m=0;n=0;o=1;m1=0;n1=0;
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)
        if Lvector(j+m)~=0
            for v=o:j
                LLvector(v+m1)=Lvector(j+m);
            end
        o=j+1;
        end
        n=n+1;n1=n1+1;
    end
    m=n+1;o=1;m1=n1; n=n+1;
end
clear m;clear n;clear m1;clear n1;clear o;

%---------storing Mcrvector in a matrix of the number of the subelements.
every
%element has the same Mcrvector value along the checked length.
McrVector=zeros(s,1);
m=0;n=0;o=1;m1=0;n1=0;
for i=1:length(devidingsnum)
    for j=1:devidingsnum(i)
        if Mcrvector(j+m)~=0
            for v=o:j
                McrVector(v+m1)=Mcrvector(j+m);
            end
        o=j+1;
        end
        n=n+1;n1=n1+1;
    end
    m=n+1;o=1;m1=n1; n=n+1;
end
clear m;clear n;clear m1;clear n1;clear o;

%-------------------------------- End ----------------------------------
```

```
function
[NRd,VRd,MRd,WW]=resistance(D,DevedgesA,DevedgesZ,DevedgesI,Bendingclass,ga
mmam0,fy)

%********************************** resistance
****************************************

%-----------------------------------Purpose----------------------------------
-------------
% Calculating the resistance against the axial force, shear force and
bending moment.
%-----------------------------------Inputs-----------------------------------
------------
%D: a matrix includes the sub cross sections for each subelement.
%DevedgesA: a vector of the average area at the edge of each subelement.
%DevedgesZ: a vector of the average ceter of gravity at the edge of each
subelement.
%DevedgesI: a vector of the average moment of intertia about y-direction at
the edge of each subelement.
%Esedge: calculate the internal forces at the edge of each subelements.
%Bendingclass: a vector of the bending class of each subelement.
%gammam0: a safety factor.
%fy: yield strength of steel.
%-----------------------------------Outputs----------------------------------
-------------
%NRd: a vector of the capacity of each subelements against normal force.
%VRd: a vector of the capacity of each subelements against shear force.
%MRd: a vector of the capacity of each subelements against moment force.
%WW: a vector includes the avarage values of the sectional modulus of the
first and the second edge of the subelement.
%**************************************************************************
***********

%---------------cross section resistatnce-----------------
mm=size(D);nn=mm(1);
NRd=zeros(nn,1);VRd=zeros(nn,1);MRd=zeros(nn,1);
WW=zeros(nn,1);
% gammam0=1;
for i =1:nn
  bu=D(i,1);tu=D(i,2);hw=D(i,3);tw=D(i,4);bl=D(i,5);tl=D(i,6);
  %----------axial resistance
  NRd(i)=  DevedgesA(i)*fy*1e6/gammam0;%axial resistance
  %----------shear resistance
  Av=(hw*tw);%shear area
  VRd(i)=Av*fy*1e6/sqrt(3)/gammam0;%shear resistance
  %----------moment resistance
  a1=bl*tl;a2=(DevedgesZ(i)-tl)*tw;a3=(hw-DevedgesZ(i)+tl)*tw;a4=bu*tu;
  z1=DevedgesZ(i)-tl/2;z2=(DevedgesZ(i)-tl)/2;
  z3=(hw+tl-DevedgesZ(i))/2;z4=tl+hw+tu/2-DevedgesZ(i);
  if Bendingclass(i)==1||Bendingclass(i)==2%if the cross section in class 1
or 2
      %then the plastic sectional modulus, Wplastic should be used,
otherwise Welastic used for
      %section class 3 and 4.
      WW(i)=a1*z1+a2*z2+a3*z3+a4*z4;%Wplastic
  else WW(i)=DevedgesI(i)/DevedgesZ(i);% Welastic=I/y
  end
   MRd(i)=WW(i)*fy*1e6/gammam0;%moment resistance
end
clear a1;clear a2;clear a3;clear a4;clear z1;clear z2;clear Z3;clear z4;
```

```
%------------------------------ End ------------------------------------
```

```
function SLS(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,Eq)
%********************************** resistance
****************************************

%-----------------------------------SLS---------------------------------
---------
% Calculating the displacement and plotted the deformed shape for the frame
according to the
% Serviceability Limit State (SLS).
%-----------------------------------Inputs-------------------------------
------------
%E: the elasticity of steel.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%maincoord: the coordinates at the main nodes of the frame.
%Eq: the horizontal and vertical loads. Uniformly distributed loads applied
over the 6 main elements.
%-----------------------------------Outputs-----------------------------
-------------

%************************************************************************
***********

%--------------interpolation of A,I and Z------------------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
s=sum(devidingsnum);%the total number of the subelements

DevC=zeros(s,6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);
for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=x/2;
    for j=1:devidingsnum(.5*i+.5)


        DevC(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j+x/2;
        n=n+1;
    end
         m=n;
end
clear C1;clear C2;clear m; clear n;
DevA=zeros(s,1);
DevI=zeros(s,1);DevIz=zeros(s,1);
DevZ=zeros(s,1);
for i=1:s
bu=DevC(i,1);tu=DevC(i,2);hw=DevC(i,3);tw=DevC(i,4);
bl=DevC(i,5);tl=DevC(i,6);
```

```
DevA(i)=bu*tu+hw*tw+bl*tl;
DevZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevA(i);
DevI(i)=bu*tl^3/12+bl*tl*(DevZ(i)-tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevZ(i))^2;
DevIz(i)=bu^3*tu/12+tw^3*hw/12+bl^3*tl/12;
end


%----------EP generation-----------
%section properties E,A and I.
Ep=zeros(s,3);
for i=1:s
    Ep(i,:)=[E DevA(i) DevI(i)];
end

%------------Coord generation-------------
Coord=zeros((s+1),2);
n=1;m=1;
for i =1:elementnum
    x=(maincoord((i+1),1)-maincoord(i,1))/devidingsnum(i);
     y=(maincoord((i+1),2)-maincoord(i,2))/devidingsnum(i);
     for j=1:devidingsnum(i)
         Coord((j+m),1)=maincoord(i,1)+j*x;
         Coord((j+m),2)=maincoord(i,2)+j*y;
         n=n+1;
     end
     m=n;
end

%----------Ex & Ey generation------------
Ex=zeros(s,2);Ey=zeros(s,2);
for i =1:s
    Ex(i,1)=Coord(i,1);Ex(i,2)=Coord(i+1,1);
    Ey(i,1)=Coord(i,2);Ey(i,2)=Coord(i+1,2);
end


%-------------Edof generation---------------
Edof=zeros(s,7);
for i =1:s
    x=3*i-2;
    Edof(i,:)=[i x x+1 x+2 x+3 x+4 x+5];
end

%------------Eqlokal, deviding the global distributed vertical load (like
snow load) into two component,
%vertical and horizontal depending on the angel of the element.
Eqlokal=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);%horizontal distance
y1=maincoord(i,2);y2=maincoord(i+1,2);%vertical distance
Eqlokal(i,:)=[Eq(i,1),Eq(i,3)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,3)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the global load times cos(angel), and the
%horizontal component is the global load times sin(angel).
end

%-------------Eqlokalh, deviding the horizontal component of the load that
have been got
```

```
%from Eqlokal into to two componenets, vertical and horizontal in case of
inclined
%element
%------------EqWind, deviding the global distributed horizontal load (like
wind load) into two component,
%vertical and horizontal depending on the angel of every main element.
Eqlokalh=zeros(elementnum,3);EqWind=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);
y1=maincoord(i,2);y2=maincoord(i+1,2);
Eqlokalh(i,:)=[Eqlokal(i,1),Eqlokal(i,2)*(y2-y1)/((x2-x1)^2+(y2-
y1)^2)^.5,Eqlokal(i,2)*(x2-x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the horzontal coponent times cos(angel), and the
%horizontal component is the horizontal component times sin(angel).

%--------------the wind loads
EqWind(i,:)=[Eq(i,1),Eq(i,2)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,2)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
end

%----------EQh, transfere the distributed loads tha have been got from
Eqlokalh into point loads
%----------EQWind, redistributing av the loads that have been got from
EqHoriz over the sublements
n=0;m=0;
EQh=zeros(s,2);EQWind=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQh(j+m,1:2)=Eqlokalh(i,2:3);
        EQWind(j+m,1:2)=EqWind(i,2:3);%the wind loads
         n=n+1;
    end
    m=n;
end

Eq=Eqlokal;

%----------EQ, redistributing av the loads that have been got from Eqlokal
over the sublements
n=0;m=0;
EQ=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQ(j+m,1:2)=Eq(i,2:3);
         n=n+1;
    end
    m=n;
end

%------------L generation, the length of the subelements--------
L=zeros(s,1);
for i =1:s
x1=Coord(i,1);y1=Coord(i,2);x2=Coord(i+1,1);y2=Coord(i+1,2);
L(i)=((x2-x1)^2+(y2-y1)^2)^.5;
end

%---------P generation, transfere the distributed loads from EQh to
%vertical and horzontal point loads acting on the nodes of the subelements
pph=0;Phh=zeros(s+1,1);%the horizontal point loads
ppv=0;Pvv=zeros(s+1,1);%the vertical point loads
```

```
for i=1:s
    Ph=EQh(i,2)*L(i)/2;
    Phh(i)=Ph+pph;Phh(i+1)=Ph;pph=Ph;

    Pv=EQh(i,1)*L(i)/2;
    Pvv(i)=Pv+ppv;Pvv(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%---------PWind generation, transfere the distributed loads from EQWind to
%vertical and horzontal point loads acting on the nodes of the subelements
pph=0;PhhWind=zeros(s+1,1);%the horizontal point loads
ppv=0;PvvWind=zeros(s+1,1);%the vertical point loads
for i=1:s
    Ph=EQWind(i,2)*L(i)/2;
    PhhWind(i)=Ph+pph;PhhWind(i+1)=Ph;pph=Ph;

    Pv=EQWind(i,1)*L(i)/2;
    PvvWind(i)=Pv+ppv;PvvWind(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%----------Inetial values for the iteration-----
eps=0.0001;
N=zeros(s,1);N(1)=.01;
N0=ones(s,1);
n=0;

%--------------bc-----------------------
bc=[1 0;2 0;Edof(end,5) 0;Edof(end,6) 0];% here it is assumed that the both
supports of
%the frame is fixed.

%----------Es for first order theory, calclation of the inner forces from
the
%first order theory that can be used then to calculate the forces due to
%imperfection effect.
[Ess]=Es(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,Eq);
[Esedgee]=Esedge(Bracing1,Bracing2,Bracing3,Crosssec,Ess);

% %---------------Imperfection for global analysis of frames EC3 5.3.2-----
----
[NN1,NN2]=imperfection(maincoord,Bracing1,Bracing2,Bracing3,Esedgee);

%**************************************************************************
*****************
%------------------------------Iteration procedure-------------------------
------------------
while (abs((N(1)-N0(1))/N0(1))>eps)
    n=n+1;


%----------puting the nodal forces on the accual nodes------
f=zeros(Edof(end),1);
for i=1:s+1

f(3*i-2)=Phh(i)+PvvWind(i);%puting the vertincal point loads on the nodes
with vertical degree of freedom
```

```
f(3*i-1)=Pvv(i)+PhhWind(i);%puting the horzontal point loads on the nodes
with horizontal degree of freedom

end

%extra horizontal force added on the upper edge of the both columns caused
%by the effect of the imperfection
i1=devidingsnum(1)*3+1;
i2=(sum(devidingsnum)-devidingsnum(length(devidingsnum)))*3+1;%the location
of the horizontal DOF of the upper edge of the both columns.
f(i1)=f(i1)+NN1;f(i2)=f(i2)+NN2;%puting the extra forces of the
imperfection at the actual nodes.


%------------K generation-----------------
K=zeros(Edof(end));
for i=1:s
    [Ke,fe]=beam2g(Ex(i,:),Ey(i,:),Ep(i,:),N(i),EQ(i,2));
    [K,f]=assem(Edof(i,:),K,Ke,f,fe);
end

%--------------bc, Ed and a----------------------
% bc=[1 0;2 0;3 0;Edof(end,5) 0;Edof(end,6) 0;Edof(end,7) 0];% here it is
assumed that the both supports of
%the frame is fixed.
a=solveq(K,f,bc);
Ed=extract(Edof,a); %the displacements ux,uy and r locally for each member.

%--------------stresses calculation---------------------
Esnon=zeros(s*2,3);
for i=1:s
  Esnon((2*i-
1):(2*i),:)=beam2gs(Ex(i,:),Ey(i,:),Ep(i,:),Ed(i,:),N(i),EQ(i,2));
end
N0=N;
for i=1:2:s*2;
    N((.5*i+.5),1)=Esnon(i,1);% calculating the normal forces for every
first
    % beam edge and save them in N-matrix to reuse them in the next
iteration.
end

if (n>20)
    disp('The solution doesn''t converge')
    break
end
end
%**********************************END***********************************
aSLS=a;
%-------------finding the max horizontal and vertical displacements of the
frame
for i=1:3:length(a)-4
    if abs(a(i+3))>abs(a(i))
        ahmax1=a(i+3);% max horzontal displacement.
    end
    if abs(a(i+4))>abs(a(i+1))
        avmax1=a(i+4);%max vertical displacement
    end
end
ahmax=ahmax1;avmax=avmax1;
```

```
%--------------plot the deformed and undeformed shape according to SLS-----
---
figure(1)
plotpar=[2 1 0];
eldraw2(Ex,Ey,plotpar);
plotpar=[1 4 0];
eldisp2(Ex,Ey,Ed,plotpar,5);
axis([maincoord(1,1)-5, maincoord(end,1)+5,min(maincoord(:,2))-
5,max(maincoord(:,2))+5])
SigmaVmax=(maincoord(end,1)-maincoord(1,1))/250;%max allowable vertical
deformation of the frame
u4=(abs(avmax)/SigmaVmax)*100;
text(-4,10,['max vertical disp.= ',num2str(avmax),'
m']);text(10,10,['vertical disp. utilization= ',num2str(u4),' %'])
SigmaHmax=(maincoord(end,1)-maincoord(1,1))/150;%max allowable horizontal
deformation of the frame
u5=(abs(ahmax)/SigmaHmax)*100;
text(-4,9,['max horizontal disp.= ',num2str(ahmax),'
m']);text(10,9,['horizontal disp. utilization= ',num2str(u5),' %'])
title('Deformed shape according to SLS criterion')

%--------------plot the forces------------------
% %--------------plot the normal forces---------------
% figure(2)
%   plotpar=[2 1];
% %   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Esnon(1,1);Esnon(1,1)]);
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[Esnon(2*i-1,1);Esnon(2*i,1)],plotpar,16e-6);
%     text(Coord(i,1)+.5,Coord(i,2),num2str(Esnon(2*i-1,1)));
% end
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Esnon(2*i,1)));
% title('normal forces diagram N')
%
% %--------------plot the shear forces------------------
% figure(3)
%   plotpar=[2 1];
% %   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,2);Es(1,2)]);
% for i=1:s
%     eldia2(Ex(i,:),Ey(i,:),[Esnon(2*i-1,2);Esnon(2*i,2)],plotpar,16e-6);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(Esnon(2*i-1,2)));
% end
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Esnon(2*i,2)));
% title('shear forces diagram N')
%
% %---------------plot the moment---------------------
% figure(4)
%   plotpar=[2 1];
% %   [sfac]=eldia2(Ex(1,:),Ey(1,:),[Es(1,3);Es(1,3)]);
% for i=1:s
% %     [sfac]=eldia2(Ex(i,:),Ey(i,:),[Es(2*i-1,3);Es(2*i,3)]);
%     eldia2(Ex(i,:),Ey(i,:),[Esnon(2*i-1,3);Esnon(2*i,3)],plotpar,16e-6);
%         text(Coord(i,1)+.5,Coord(i,2),num2str(Esnon(2*i-1,3)));
% end
% text(Coord(i+1,1)+.5,Coord(i+1,2),num2str(Esnon(2*i,3)));
% title('moment diagram Nm')

%------------------------- End ---------------------------------------
```

```
function
[ahmax,avmax,Esnon,DevA,DevI,L]=Esnon(E,Bracing1,Bracing2,Bracing3,Crosssec
,maincoord,Eq)

%********************************** Esnon
**************************************
%-----------------------------------Purpose-------------------------------
-------------
% Nonlinearly calculation of the internal forces of each subelements.
%-----------------------------------Inputs--------------------------------
------------
%E: the elasticity of steel.
%Bracing1: bracings distribution vector of the first element.
%Bracing2: bracings distribution vector of the second element.
%Bracing3: bracings distribution vector of the third element.
%Crosssec: a 12*6 matrix includes the cross sections of the main elements
at the edegs.
%maincoord: the coordinates at the main nodes of the frame.
%Eq: the horizontal and vertical loads. Uniformly distributed loads applied
over the 6 main elements.
%-----------------------------------Outputs-------------------------------
-------------
%ahmax: the max horizontal displacements of the frame.
%avmax: the max vertical displacements of the frame.
%Esnon: the internal forces of each subelement generated by the CALFEM-
fuction beam2s.
%DevA: a vector of the average area of each subelement.
%DevZ: a vector of the average ceter of gravity of each subelement.
%DevI: a vector of the average moment of intertia about y-direction of each
subelement.
%L: a vector of lengths of the 6 main elements.
%**************************************************************************
*************

%--------------interpolation of A,I and Z-------------------
devidingsnum=[length(Bracing1)-1;length(Bracing2)-1;length(Bracing3)-
1;length(Bracing3)-1;length(Bracing2)-1;length(Bracing1)-1];%number of
deviding of every main element. each main elemet
%will devided into number of sub elements.
elementnum=(max(size(devidingsnum)));%number of main elements before
deviding
s=sum(devidingsnum);%the total number of the subelements

DevC=zeros(s,6);
n=0;m=0;
mm=size(Crosssec);nn=mm(1);
for i=1:2:nn
    x1=0;x3=1;

    C1=Crosssec(i,:);C2=Crosssec(i+1,:);
    x=1/devidingsnum(.5*i+.5);
    x2=x/2;
    for j=1:devidingsnum(.5*i+.5)


        DevC(j+m,:)=C1+(x2-x1)/(x3-x1)*(C2-C1);
        x2=x*j+x/2;
        n=n+1;
    end
         m=n;
```

```
end
clear C1;clear C2;clear m; clear n;
DevA=zeros(s,1);
DevI=zeros(s,1);DevIz=zeros(s,1);
DevZ=zeros(s,1);
for i=1:s
bu=DevC(i,1);tu=DevC(i,2);hw=DevC(i,3);tw=DevC(i,4);
bl=DevC(i,5);tl=DevC(i,6);
DevA(i)=bu*tu+hw*tw+bl*tl;
DevZ(i)=(bu*tu*(tl+hw+tu/2)+hw*tw*(hw/2+tl)+bl*tl*(tl/2))/DevA(i);
DevI(i)=bu*tl^3/12+bl*tl*(DevZ(i)-tl/2)^2+hw^3*tw/12+hw*tw*(hw/2+tl-
DevZ(i))^2+bu*tu^3/12+bu*tu*(tu/2+hw+tl-DevZ(i))^2;
DevIz(i)=bu^3*tu/12+tw^3*hw/12+bl^3*tl/12;
end


%----------EP generation-----------
%section properties E,A and I.
Ep=zeros(s,3);
for i=1:s
    Ep(i,:)=[E DevA(i) DevI(i)];
end

%------------Coord generation-------------
Coord=zeros((s+1),2);
n=1;m=1;
for i =1:elementnum
    x=(maincoord((i+1),1)-maincoord(i,1))/devidingsnum(i);
     y=(maincoord((i+1),2)-maincoord(i,2))/devidingsnum(i);
     for j=1:devidingsnum(i)
         Coord((j+m),1)=maincoord(i,1)+j*x;
         Coord((j+m),2)=maincoord(i,2)+j*y;
         n=n+1;
     end
     m=n;
end

%----------Ex & Ey generation------------
Ex=zeros(s,2);Ey=zeros(s,2);
for i =1:s
    Ex(i,1)=Coord(i,1);Ex(i,2)=Coord(i+1,1);
    Ey(i,1)=Coord(i,2);Ey(i,2)=Coord(i+1,2);
end


%-------------Edof generation---------------
Edof=zeros(s,7);
for i =1:s
    x=3*i-2;
    Edof(i,:)=[i x x+1 x+2 x+3 x+4 x+5];
end


%------------Eqlokal, deviding the global distributed vertical load (like
snow load) into two component,
%vertical and horizontal depending on the angel of the element.
Eqlokal=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);%horizontal distance
y1=maincoord(i,2);y2=maincoord(i+1,2);%vertical distance
```

```
Eqlokal(i,:)=[Eq(i,1),Eq(i,3)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,3)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the global load times cos(angel), and the
%horizontal component is the global load times sin(angel).
end

%--------------Eqlokalh, deviding the horizontal component of the load that
have been got
%from Eqlokal into to two componenets, vertical and horizontal in case of
inclined
%element
%------------EqWind, deviding the global distributed horizontal load (like
wind load) into two component,
%vertical and horizontal depending on the angel of every main element.
Eqlokalh=zeros(elementnum,3);EqWind=zeros(elementnum,3);
for i=1:elementnum
    x1=maincoord(i,1);x2=maincoord(i+1,1);
y1=maincoord(i,2);y2=maincoord(i+1,2);
Eqlokalh(i,:)=[Eqlokal(i,1),Eqlokal(i,2)*(y2-y1)/((x2-x1)^2+(y2-
y1)^2)^.5,Eqlokal(i,2)*(x2-x1)/((x2-x1)^2+(y2-y1)^2)^.5];
%the vertical component is the horzontal coponent times cos(angel), and the
%horizontal component is the horizontal component times sin(angel).

%--------------the wind loads
EqWind(i,:)=[Eq(i,1),Eq(i,2)*(y2-y1)/((x2-x1)^2+(y2-y1)^2)^.5,Eq(i,2)*(x2-
x1)/((x2-x1)^2+(y2-y1)^2)^.5];
end

%----------EQh, transfere the distributed loads tha have been got from
Eqlokalh into point loads
%----------EQWind, redistributing av the loads that have been got from
EqHoriz over the sublements
n=0;m=0;
EQh=zeros(s,2);EQWind=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQh(j+m,1:2)=Eqlokalh(i,2:3);
        EQWind(j+m,1:2)=EqWind(i,2:3);%the wind loads
         n=n+1;
    end
    m=n;
end

Eq=Eqlokal;

%----------EQ, redistributing av the loads that have been got from Eqlokal
over the sublements
n=0;m=0;
EQ=zeros(s,2);
for i=1:elementnum
    for j=1:devidingsnum(i)
        EQ(j+m,1:2)=Eq(i,2:3);
         n=n+1;
    end
    m=n;
end

%------------L generation, the length of the subelements--------
L=zeros(s,1);
for i =1:s
```

```
x1=Coord(i,1);y1=Coord(i,2);x2=Coord(i+1,1);y2=Coord(i+1,2);
L(i)=((x2-x1)^2+(y2-y1)^2)^.5;
end

%---------P generation, transfere the distributed loads from EQh to
%vertical and horzontal point loads acting on the nodes of the subelements
pph=0;Phh=zeros(s+1,1);%the horizontal point loads
ppv=0;Pvv=zeros(s+1,1);%the vertical point loads
for i=1:s
    Ph=EQh(i,2)*L(i)/2;
    Phh(i)=Ph+pph;Phh(i+1)=Ph;pph=Ph;

    Pv=EQh(i,1)*L(i)/2;
    Pvv(i)=Pv+ppv;Pvv(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%---------PWind generation, transfere the distributed loads from EQWind to
%vertical and horzontal point loads acting on the nodes of the subelements
pph=0;PhhWind=zeros(s+1,1);%the horizontal point loads
ppv=0;PvvWind=zeros(s+1,1);%the vertical point loads
for i=1:s
    Ph=EQWind(i,2)*L(i)/2;
    PhhWind(i)=Ph+pph;PhhWind(i+1)=Ph;pph=Ph;

    Pv=EQWind(i,1)*L(i)/2;
    PvvWind(i)=Pv+ppv;PvvWind(i+1)=Pv;ppv=Pv;
end
clear pph;clear ppv;clear Ph;clear Pv;

%----------Inetial values for the iteration-----
eps=0.0001;
N=zeros(s,1);N(1)=.01;
N0=ones(s,1);
n=0;

%--------------bc-----------------------
bc=[1 0;2 0;Edof(end,5) 0;Edof(end,6) 0];% here it is assumed that the both
supports of
%the frame is fixed.

%----------Es for first order theory, calclation of the inner forces from
the
%first order theory that can be used then to calculate the forces due to
%imperfection effect.
[Ess]=Es(E,Bracing1,Bracing2,Bracing3,Crosssec,maincoord,Eq);
[Esedgee]=Esedge(Bracing1,Bracing2,Bracing3,Crosssec,Ess);

% %---------------Imperfection for global analysis of frames EC3 5.3.2-----
----
[NN1,NN2]=imperfection(maincoord,Bracing1,Bracing2,Bracing3,Esedgee);

%**************************************************************************
****************
%-----------------------------Iteration procedure-------------------------
------------------
while (abs((N(1)-N0(1))/N0(1))>eps)
    n=n+1;
```

```
%----------puting the nodal forces on the accual nodes------
f=zeros(Edof(end),1);
for i=1:s+1

f(3*i-2)=Phh(i)+PvvWind(i);%puting the vertincal point loads on the nodes
with vertical degree of freedom
f(3*i-1)=Pvv(i)+PhhWind(i);%puting the horzontal point loads on the nodes
with horizontal degree of freedom

end

%extra horizontal force added on the upper edge of the both columns caused
%by the effect of the imperfection
i1=devidingsnum(1)*3+1;
i2=(sum(devidingsnum)-devidingsnum(length(devidingsnum)))*3+1;%the location
of the horizontal DOF of the upper edge of the both columns.
f(i1)=f(i1)+NN1;f(i2)=f(i2)+NN2;%puting the extra forces of the
imperfection at the actual nodes.


%------------K generation-----------------
K=zeros(Edof(end));
for i=1:s
    [Ke,fe]=beam2g(Ex(i,:),Ey(i,:),Ep(i,:),N(i),EQ(i,2));
    [K,f]=assem(Edof(i,:),K,Ke,f,fe);
end

%--------------bc, Ed and a----------------------
% bc=[1 0;2 0;3 0;Edof(end,5) 0;Edof(end,6) 0;Edof(end,7) 0];% here it is
assumed that the both supports of
%the frame is fixed.
a=solveq(K,f,bc);
Ed=extract(Edof,a); %the displacements ux,uy and r locally for each member.

%--------------stresses calculation---------------------
Esnon=zeros(s*2,3);
for i=1:s
  Esnon((2*i-
1):(2*i),:)=beam2gs(Ex(i,:),Ey(i,:),Ep(i,:),Ed(i,:),N(i),EQ(i,2));
end
N0=N;
for i=1:2:s*2;
    N((.5*i+.5),1)=Esnon(i,1);% calculating the normal forces for every
first
    % beam edge and save them in N-matrix to reuse them in the next
iteration.
end

if (n>20)
    disp('The solution doesn''t converge')
    break
end
end
%********************************* End of itiration
**********************************

%-------------finding the max horizontal and vertical displacements of the
frame
```

```
for i=1:3:length(a)-4
    if abs(a(i+3))>abs(a(i))
        ahmax1=a(i+3);% max horzontal displacement.
    end
    if abs(a(i+4))>abs(a(i+1))
        avmax1=a(i+4);%max vertical displacement
    end
end
ahmax=ahmax1;avmax=avmax1;

%**********************************End**************************************
*****************
```