Chapter 8

Topological Optimization and Reliability

8.1 Topological Optimization

Optimization can be defined as a procedure by which it is possible to find a solution or a set of optimal solutions for a given function or set of functions, which govern a specific problem, subject to restrictions. So, in topological optimization, there is the intention of providing the best distribution of material from a fixed design space. Hence, this innovation favors industries in different sectors, considering that designing mechanical parts and components with large stiffness and small weight has become a common necessity. It has been applied, for example, in aircraft wing spars webs, as represented in Fig. 8.1.

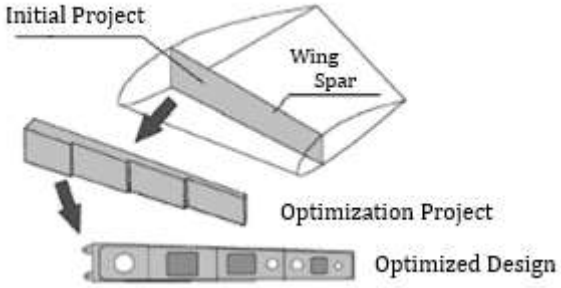

Figure 8.1 – Process of topological optimization of an aircraft wing spar

The topological optimization procedure can be outlined as shown in Fig. 8.2, which displays the use of the Finite Element Method (FEM) together with an optimization algorithm, which introduces numerical strategies in the search for optimal engineering solutions, to obtain an optimized domain.

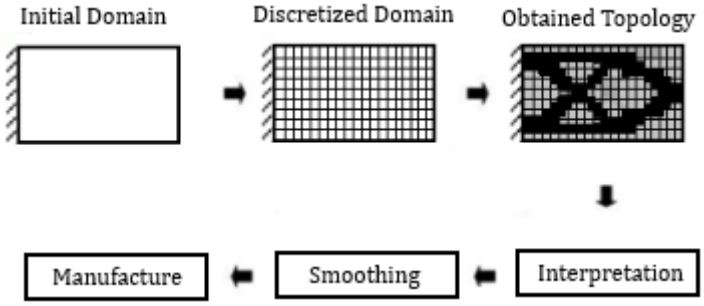

Figure 8.2 – Topological optimization design procedure

It is worth mentioning that in the practical application of topological optimization, some aspects are fundamental. For example, in Fig. 8.2 (obtained topology), points with intermediate shades between black and white, called gray scales, can appear. These points indicate the presence of elements with an intermediate thickness between the maximum and the null. These thicknesses may not be feasible to

be implemented in practice, but they usually occur, that is, the presence of the gray scale is inherent in obtaining the optimal solution, Brasil (2017). The image of the obtained structure by topological optimization (TO) represents an excellent starting point that needs to be interpreted, in order to obtain the final design of the structure to be adopted in practice in the industry. This interpretation process is called refinement or smoothing, and can be done using image processing methods, or simply by designing a structure based on the image obtained by TO, often adhering to CAD/CAE software. In some cases, the results generated by TO are not intuitive and it is necessary to check the final structure using the Finite Element Method. The last step is the manufacture of the structure.

8.2 The Finite Element Method in Topological Optimization

To better exemplify this whole process, we will deal with some examples such as rectangular plates and beams with concentrated loads and distributed loads with various boundary conditions, using MATLAB, a high-level programming language that allows the solution of countless scientific problems with its accessible syntax, excellent debugging tools and extensive graph manipulation tools. Therefore, it allows the user to focus on the physical and mathematical context of the optimization problem without being distracted by technical implementation problems. The optimization algorithm will consist of determining the thickness of each element, in order to minimize the total mass function of the structure, respecting the limits of allowable stresses imposed.

The Finite Element Method (FEM) consists of discretizing the domain of the structure in several subdomains (bars, triangles, quadrilaterals, tetrahedrons etc.), called elements, of small but finite dimensions, united in points called nodes. An equilibrium equation is assembled for each element and then these equations are combined to determine an expression that represents the structure as a whole. Next, the displacements, strains and stresses in the domain of the structure are determined.

The displacements of a point in a solid continuum are modeled by a vector **u**. In the case of a plate, in a two-dimensional domain of *x* and *y* axes, we have a 2 x 1 vector.

$$\mathbf{u} = \begin{Bmatrix} u(x,y,t) \\ v(x,y,t) \end{Bmatrix} \tag{8.1}$$

From the displacements, the strains are obtained by the application of a differential operator **D**:

$$\boldsymbol{\varepsilon} = \mathbf{D}\mathbf{u} \tag{8.2}$$

so, in the case of a plate, we have the following 3 x 2 operator:

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \mathbf{Du} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{8.3}$$

The next step is to obtain the stress vector, from the strains, using, for simplicity, Hooke's law, in matrix form:

$$\boldsymbol{\sigma} = \mathbf{E}\boldsymbol{\varepsilon} = \mathbf{EDu} \tag{8.4}$$

In the case of a plate, the stress vector is 3 x 1:

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{8.5}$$

in the Plane Stress hypothesis, Hooke's Law is expressed by the following 3 x 3 matrix:

$$\mathbf{E} = \frac{E_m}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1 - \nu}{2} \end{bmatrix} \tag{8.6}$$

where ν is the Poisson's ratio and E_m is Young's modulus.

Our sample structure will be a rectangular plate. In order to implement the FEM in MATLAB, it is necessary to input the dimensions of the structure to be analyzed and how it will be discretized. In this specific case, Argyris' 4 nodes square elements will be used, and one must define the number of elements, nodes and size for each element, in order to generate the coordinates of the nodes and the connectivity between them. In addition to inserting all the physical specificities of the material, such as Poisson's ratio and the Young's modulus.

```
%Dimensions
compr=8;
alt=2;
%divisions
```

```
ndx=24; ndy=6; %number of divisions on x and y
nel=ndx*ndy; %number of elements
nno=(ndx+1)*(ndy+1); %number of nodes
dx=compr/ndx; dy=alt/ndy; %size of each element
%admissible stress of steel
csadm = 225e6;
%generating the nodes coordinates
gcoord=zeros(nno,2);
x=0;y=0;k=0;
for i=1:ndx+1
    for j=1:ndy+1
        k=k+1;
        gcoord(k,1)=x;
        gcoord(k,2)=y;
        y=y+dy;
    end
    y=0;
    x=x+dx;
end
%Elements connectivity
nodel=zeros(nel,4); %matrix finite element
kel=0;kaux=0;
for i=1:ndx
    for j=1:ndy
        kaux=kaux+1;
        kel=kel+1;
        nodel(kel,1)=kaux+ndy+2;
        nodel(kel,2)=kaux+1;
        nodel(kel,3)=kaux;
        nodel(kel,4)=kaux+ndy+1;
    end
```

```
    kaux=kaux+1;
end
%problem dimensions
nglpn=2;%number of degrees of freedom per node
nds=nno*nglpn; %number of system displacements
nnel=4; %number of nodes per element
ndpel=nnel*nglpn;%number of displacements per element
%physical data of the elements
t0=0.1; %inital thickness
EM=200e9; %steel elasticity module
nu=0.3; %Poisson's ratio of steel
EL=EM/(1-nu*nu);
G=EM/2/(1+nu); %shear module
E=EL*[1 nu 0;nu 1 0;0 0 (1-nu)/2]; %Elasticity matrix
```

Then it will be sought to obtain the stresses present in each element. For that end, it is necessary to determine certain matrices and vectors. In this way, it is defined the loads, constraints, system stiffness matrix and the displacements.

```
%matrix: number of degrees of freedom per node
LN=zeros(nno,nglpn);
%boundary conditions
%cantilever beam
 for i= k-ndy:k
     LN(i,:)=[-1 -1];
 end
%matrix LN
ngl=0;
for i=1:nno
    for j=1:nglpn
        if  LN(i,j)== 0
```

```
            ngl=ngl+1;
            LN(i,j)=ngl;
        end
    end
end
ngr=ngl;
for i=1:nno
    for j=1:nglpn
        if LN(i,j)<0
            ngr=ngr+1;
            LN(i,j)=ngr;
        end
    end
end
%matrix and vector initialization
K=zeros(nds,nds);
p=zeros(nds,1);
P=zeros(nds,1);
Tens=zeros(nel,3);
q=zeros(8,1);
% loading vector P
%vertical load
%V=-100e4;
%  for i=1:ndy+1
%     P(LN(i,2))=V/(ndy+1);
%  end
%stiffness matrices of the elements
nd=ones(1,4);
for iel=1:nel
    for j=1:nnel
        nd(j)=nodel(iel,j);
```

```
        end
        xa=gcoord(nd(1),1);xb=gcoord(nd(2),1);
        yb=gcoord(nd(2),2);yc=gcoord(nd(3),2);
        t = vt(iel,1);
    %rectangle dimensions
        a=(xa-xb)/2;
        b=(yb-yc)/2;
    %constants
        c1=EL*t*b/3/a;
        c2=c1/2;
        c3=EL*t*nu/4;
        c4=G*t*a/3/b;
        c5=c4/2;
        c6=G*t/4;
    %
        kd(1,1)=c1;kd(1,2)=c3;kd(1,3)=-
c1;kd(1,4)=c3;kd(1,5)=-c2;kd(1,6)=-
c3;kd(1,7)=c2;kd(1,8)=-c3;
        kd(2,2)=c1;kd(2,3)=-c3;kd(2,4)=c2;kd(2,5)=-
c3;kd(2,6)=-c2;kd(2,7)=c3;kd(2,8)=-c1;
        kd(3,3)=c1;kd(3,4)=-
c3;kd(3,5)=c2;kd(3,6)=c3;kd(3,7)=-c2;kd(3,8)=c3;
        kd(4,4)=c1;kd(4,5)=-c3;kd(4,6)=-
c1;kd(4,7)=c3;kd(4,8)=-c2;
        kd(5,5)=c1;kd(5,6)=c3;kd(5,7)=-c1;kd(5,8)=c3;
        kd(6,6)=c1;kd(6,7)=-c3;kd(6,8)=c2;
        kd(7,7)=c1;kd(7,8)=-c3;
        kd(8,8)=c1;
    %
        ks(1,1)=c4;ks(1,2)=c6;ks(1,3)=c5;ks(1,4)=-
c6;ks(1,5)=-c5;ks(1,6)=-c6;ks(1,7)=-c4;ks(1,8)=c6;
```

```
        ks(2,2)=c4;ks(2,3)=c6;ks(2,4)=-c4;ks(2,5)=-
c6;ks(2,6)=-c5;ks(2,7)=-c6;ks(2,8)=c5;
        ks(3,3)=c4;ks(3,4)=-c6;ks(3,5)=-c4;ks(3,6)=-
c6;ks(3,7)=-c5;ks(3,8)=c6;

ks(4,4)=c4;ks(4,5)=c6;ks(4,6)=c5;ks(4,7)=c6;ks(4,8)=-c5;
        ks(5,5)=c4;ks(5,6)=c6;ks(5,7)=c5;ks(5,8)=-c6;
        ks(6,6)=c4;ks(6,7)=c6;ks(6,8)=-c4;
        ks(7,7)=c4;ks(7,8)=-c6;
        ks(8,8)=c4;
    %
        k=kd+ks;
    %symmetry
    for i=2:8
            for j=1:i-1
                k(i,j)=k(j,i);
            end
        end
    %sum in the system stiffness matrix
        kl=0;
        d = ones(1,8);
        for n=1:nnel
            kl=kl+1;
            d(kl)=LN(nd(n),1);
            kl=kl+1;
            d(kl)=LN(nd(n),2);
        end
        for i=1:ndpel
            for j=1:ndpel
                K(d(i),d(j))=K(d(i),d(j))+k(i,j);
            end
```

```
        end
    end
    %System solution
    %calculation of displacements
    disp('displacements')
    p(1:ngl)=K(1:ngl,1:ngl)\(P(1:ngl)-
K(1:ngl,ngl+1:nds)*p(ngl+1:nds));
    disp(p)
    %calculation of support reactions
    disp('Esforços Nodais inclusive reacoes de apoio')
    P(ngl+1:nds)=K(ngl+1:nds,1:ngl)*p(1:ngl)+K(ngl+1:ngl
+1,ngl+1:ngl+1)..
        *p(ngl+1:nds);
    disp(P)
    %Stress
    disp('Stresses at the central point of the elements')
    disp('sigma_x,    sigma_y,    tau_xy')
    for iel=1:nel
        for j=1:nnel
            nd(j)=nodel(iel,j);
        end
        xa=gcoord(nd(1),1);xb=gcoord(nd(2),1);
        yb=gcoord(nd(2),2);yc=gcoord(nd(3),2);
    %  rectangle dimensions
        a=(xa-xb)/2;b=(yb-yc)/2;
    % constants
        ca=1/4/a;cb=1/4/b;
    %  matrix B=L*N calculated in the center of the element
x=y=0
        B=[ca 0 -ca 0 -ca 0 ca 0;0 cb 0 cb 0 -cb 0 -cb;cb
ca cb -ca -cb -ca -cb ca];
```

```
%
    kl=0;
    for n=1:nnel
        kl=kl+1;
        d(kl)=LN(nd(n),1);
        kl=kl+1;
        d(kl)=LN(nd(n),2);
    end
    for i=1:ndpel
        q(i)=p(d(i));
    end
    tau=E*B*q;
    Tens(iel,:)=tau';
end
disp(Tens)
```

Subsequent to obtaining the stresses, in order to have an optimization analysis based on the maximum allowable stresses, the maximum and minimum principal stresses and the maximum shear stresses are obtained, based on the stresses present in each element with components in x and y, given by Eqs. (8.7) and (8.8).

$$\sigma_{1,2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x + \sigma_y}{2}\right)^2 + {\tau_{yx}}^2} \tag{8.7}$$

$$\tau_{max} = \sqrt{\left(\frac{\sigma_x + \sigma_y}{2}\right)^2 + {\tau_{yx}}^2} \tag{8.8}$$

```
%Normal Stress at each element [MPa]
%sigma_1,    sigma_2,    sigma_max
smax = zeros(3,3);
for i=1:nel
    smax(i,1)=(Tens(i,1)+Tens(i,2))/2+(((Tens(i,1)-
Tens(i,2))/2)^2+Tens(i,3)^2)^0.5;
```

```
        smax(i,2)=(Tens(i,1)+Tens(i,2))/2-(((Tens(i,1)-
Tens(i,2))/2)^2+Tens(i,3)^2)^0.5;
        if abs(smax(i,1))>abs(smax(i,2))
           smax(i,3)=smax(i,1);
        else
           smax(i,3)=smax(i,2);
        end
    end
    for i=1:nel
        smax(i,3)=abs(smax(i,3));
    end
    %disp(smax)
    %vector maximum stress [MPa]
    vmax = zeros(nel,1);
    for i=1:nel
        vmax(i)= smax(i,3);
    end
```

8.3 The KINITRO Algorithm

Now there is an interesting basis that will allow us to program the optimization process of the structure and for that we will turn to a theoretical scheme of the whole process.

Initially a classic optimization problem can be defined as: Determine $\mathbf{x} \in \Re^n$ that minimizes the objective function $f(\mathbf{x})$ subject to:

Equality constraints: $$g_j(\mathbf{x}) = 0;\ j = 1, l \tag{8.9}$$

Inequality constraints: $$g_j(\mathbf{x}) \leq 0; j = l + 1, m \tag{8.10}$$

The Lagrangian function of the problem is defined as

$$\Lambda(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \sum_{i=1}^{m} u_i g_i(\mathbf{x}) \tag{8.11}$$

where $\mathbf{u} \in \Re^m$ is the vector of the Lagrange multipliers.

The variables, individually, represent certain factors of a certain project, Brasil and Silva (2018). These values are independent of each other and constantly changed during the optimization process as requested by the problem-solving tool. In this case, the design variables are the thickness vector (v_t), represented previously by the vector **x**, and the number of divisions in x and y directions.

$$v_{t\ (k,1)} = [vt_1\ vt_2\ \dots], k = 1, \dots, \text{nel} \tag{8.12}$$

Constraints are a set of limitations imposed on the system. They can be of inequality and equality, indicating maximum or minimum values that should not be exceeded. Here, the restrictions are non-linear inequalities and refer to the allowable stress, $\bar{\sigma} = 225 * 10^6\ N$, of the material and minimum thickness, $v_{tmin} = 0.0001\ m$.

$$g_1 = \sigma_{max} \leq \sigma_{adm} = \sigma_{max} - \bar{\sigma} \tag{8.13}$$

$$g_2 = v_t \geq v_{tmin} = v_{tmin} - v_t \tag{8.14}$$

The objective function *f*(**x**) corresponds to a single value, connected with the whole project. It must optimize the project in order to maximize it, minimize it or reach a desired value. The volume is the most important data of the project, since from it, the quantity of material to be used is known. It is understood that, by optimizing the volume, that is, minimizing it, the entire project is optimized. Therefore, the volume (*Vol*) will be the objective function:

$$Area_{(1,k)} = \begin{bmatrix} dx_1 * dy_1 \\ \dots \end{bmatrix}, k = 1, \dots, \text{nel} \tag{8.15}$$

$$Vol = Area * vt \tag{8.16}$$

To solve the problem, the KNITRO, Artelys (2021) algorithm will be used, which has an easy coding interface for MATLAB. The function returns, among other parameters, the optimal value of the design variables, the value of the function, Lagrange multipliers etc. The illustration of the arguments is shown in Fig. 8.3. It is worth mentioning the importance of choosing a good algorithm, because without one "the chessboard irregularity" can occur, Bensoe and Sigmund (2003), due to the high dependence of the algorithm on all the parameters of the project. Therefore, a good choice allows us to be able to maintain good results, regardless of initial values, or any other parameter, thus providing solid and consistent results.

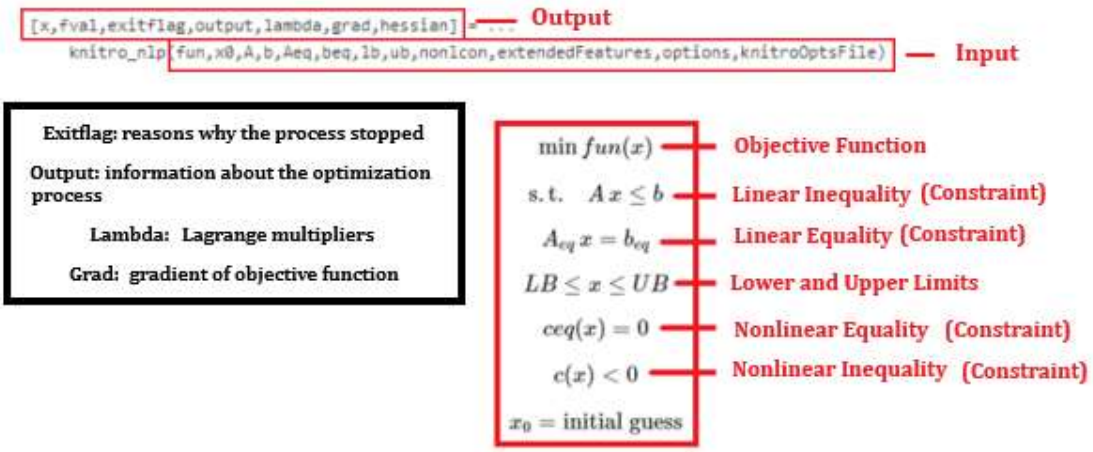

Figure 8.3 – Arguments of the KNITRO function in MATLAB, Artelys (2021)

Based on what was described, the initial formulation of the problem for optimization in MATLAB follows the given format:

```
vol = @(vt) Area*vt;
options = knitro_options('outlev',3);
[vt,fval,exitflag,output]                                    =
knitro_nlp(vol,x0,[],[],[],[],[],[],@nonlcon,[],options,[
]);
```

It is important to note that "*@nonlcon*" is performed in a function script that admits a set of initial values and returns two vectors $\mathbf{c}(\mathbf{x})$ and $\mathbf{c}_{eq}(\mathbf{x})$. In addition, "options", Artelys (2021), has a series of options capable of not only modifying how the optimization module operates, but also how the information is made available. In this case we have an "outlev" capable of informing the amount of process information that will be available to the user, such as providing a quick summary of the process to all information, including all iterations.

Note that the knitro function has several input and output parameters. The optimization problem shown in Fig. 8.3 and in the initial formulation is the same governed by Eqs. 8.9, 8.10 and 8.11, written in a more detailed way, separating linear from non-linear restrictions, as well as the lower and upper limits of the function.

After the optimization process, as a way of presenting the results, a "GRID", mesh or chessboard will be made, using the "gray scale", expressing in mathematical and visual terms both the maximum stress and the thickness, in a similar way as shown in Fig. 8.1, and programmed as follows.

```
    %Generate the X and Y grid arrays using the MESHGRID
function.
    clf;
    %Maximum Stress
    x = (1:ndx+1);
```

```
y = (1:ndy+1);
[X,Y] = meshgrid(x,y);
%Thickness
x1 = (1:ndx+1);
y1 = (1:ndy+1);
[X1,Y1] = meshgrid(x1,y1);
%Generator of Matrix Z
Z = ones(ndy+1,ndx+1);
Z1 = ones(ndy+1,ndx+1);
kx=0;
for j=1:ndx
    for i=1:ndy
        kx=kx+1;
        Z(i,j)= vmax(kx);
        Z(ndy+1,j)= 0;
        Z(i,ndx+1)= 0;
        Z(ndy+1,ndx+1)=0;
        Z1(i,j)= vt(kx);
        Z1(ndy+1,j)= 0;
        Z1(i,ndx+1)= 0;
        Z1(ndy+1,ndx+1)=0;
    end
end
%Plots
%Maximum Stress
subplot(2,1,1)
s = pcolor(X,Y,Z);
title('Maximum Stress')
colormap(flipud(gray));
colorbar;
axis image;
```

```
%Thickness
subplot(2,1,2)
s1 = pcolor(X1,Y1,Z1);
title('Thickness')
colormap(flipud(gray));
colorbar;
axis image;
```

There are some relevant points to note before presenting the results obtained in the optimization. It is known that the finer the discretization of the mesh, the more accurate the results will be. However, the constant increase results in an almost exponential computational cost. For this reason, discretization was established in two and three times of the initial dimensions of the structure, which have an initial thickness of 0.1 m.

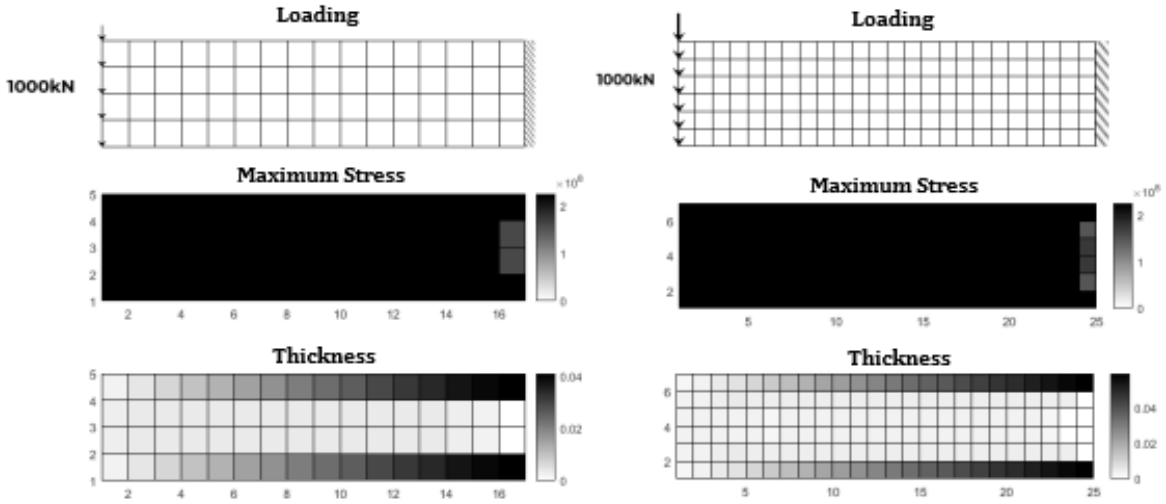

Figure 8.4 – Cantilever plate with concentrated loading on the edge (x2 and x3)

Table 8.1 - Cantilever plate with concentrated loading on the edge

Dimensions:	8x2	
Initial Volume:	1.6 m^3	
	x2	x3
Final Volume:	0.201368 m^3	0.194917 m^3
Reduction:	87.41%	87.82%

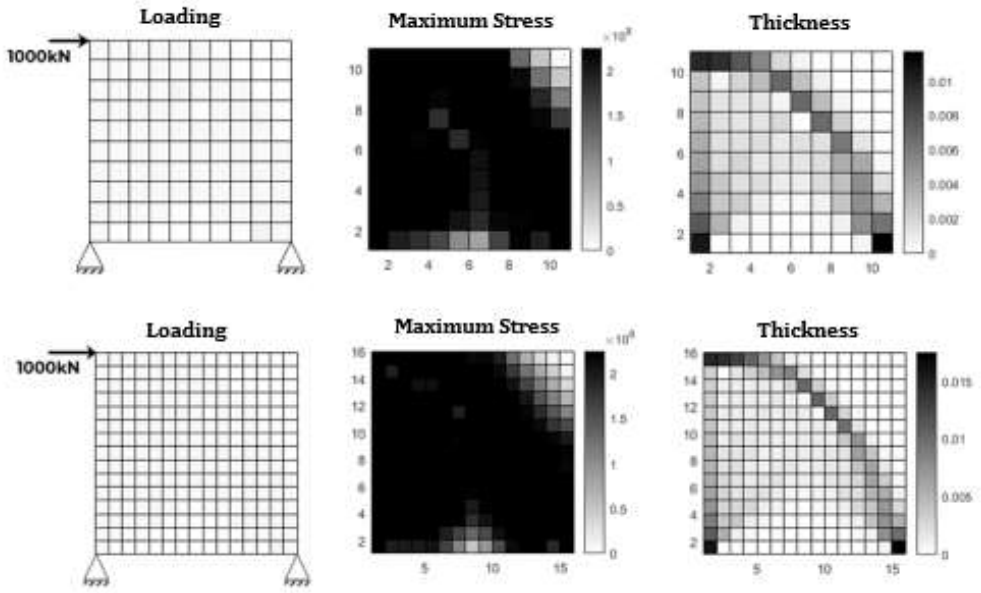

Figure 8.5 – Simply supported plate with concentrated side-loading (x2 e x3)

Table 8.2 - Cantilever plate with concentrated loading on the edge

Dimensions:	5x5	
Initial Volume:	2.5 m^3	
	x2	x3
Final Volume:	0.055845 m^3	0.056045 m^3
Reduction:	97.77%	97.76%

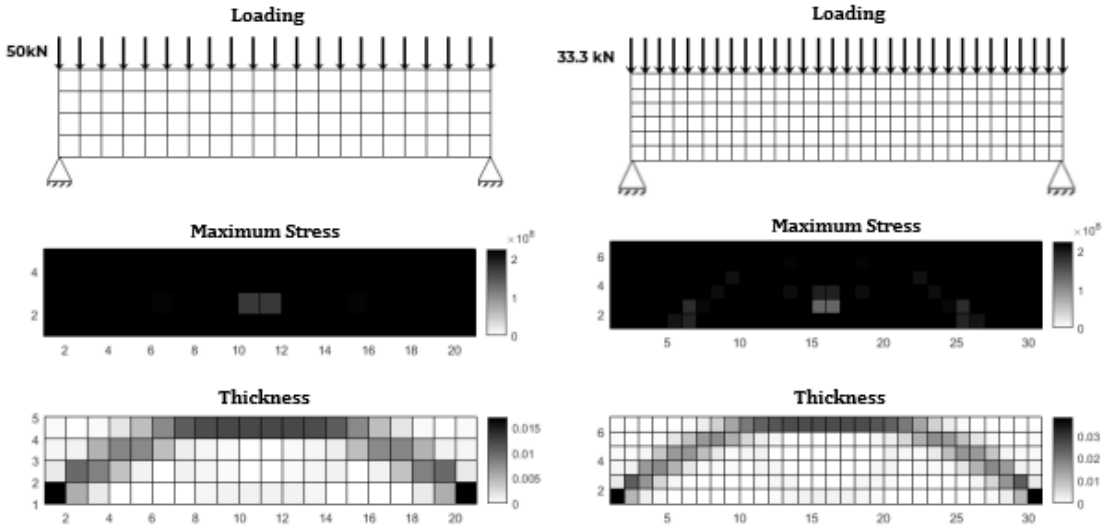

Figure 8.6 – Simply supported beam with uniformly distributed loading (x2 e x3)

Table 8.3 - Double-based beam with uniformly distributed loading

Dimensions:	10x2	
Initial Volume:	2 m^3	
	x2	x3
Final Volume:	0.072808 m^3	0.108060 m^3
Reduction:	96.36%	94.60%

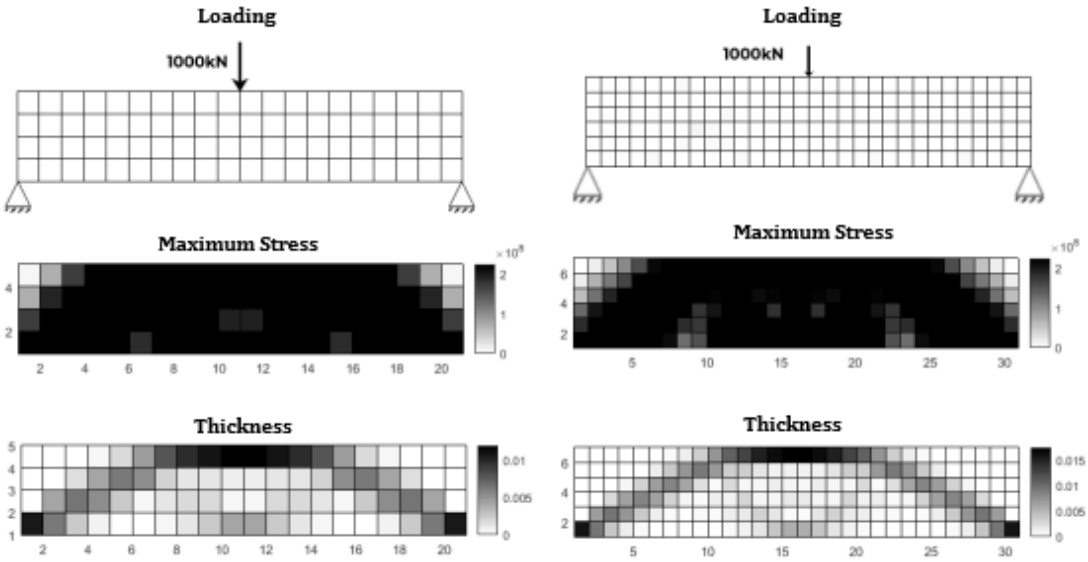

Figure 8.7 – Simply supported beam with concentrated centered loading (x2 e x3)

Table 8.4 – Simply supported beam with concentrated centered loading

Dimensions:	10x2	
Initial Volume:	2 m^3	
	x2	x3
Final Volume:	0.061673 m^3	0.061700 m^3
Reduction:	96.92%	96.91%

8.4 Reliability

After programming the entire optimization process and obtaining the results, it is possible to add a highly relevant aspect, which is the Structural Reliability.

Reliability is understood as the ability of an equipment or human being to perform their expected functions properly under specific conditions during a given

period of time, in the absence of breaks or failures. It is an area of study that aims to evaluate and optimize the reliability of systems through techniques derived from probability and statistics theories. In history, the concept of reliability acquired technological significance after the end of the First World War, when it was used for comparative studies carried out on airplanes with one, two or four engines, in order to measure the number of accidents per flight hour. It was during World War II that as a result of a failure of German V-1 missiles, mathematician Robert Lusser proposed a probability law for a serial component product. And only in 1963, a first association brought together engineers from the reliability sector and the first periodical for the dissemination of works in the area appeared in the United States.

A general problem is built around the idea of "discrete events", being developed to help follow a model over time, since it often involves a complex logical structure of its elements, Jin (1993), necessary to determine the relevant amounts that are of interest. The simulation based on this structure is often called simulation of discrete events. Therefore, the study of the Monte Carlo method is a great alternative for such simulations and requires the understanding of different areas of knowledge: Probability, to describe processes and random experiments; Statistics to analyze the data; Computer Science for efficient implementation of algorithms and Mathematical Programming to formulate and solve optimization problems.

The computational method uses random numbers and statistics to solve problems, since currently several numerical problems in Finance, Engineering and Statistics are solved with the Monte Carlo method. The interest in this study is to apply the technique in order to make the budget of a structural project feasible in terms of mass.

A limit mass,m_{el}, will be obtained based on the available budget. In the case of structural reliability analysis, Nowak and Collins (2021), this means, in the simplest approach, sampling each random variable to provide a sample value. With changes in the allowable stress and using the optimization process described in this Chapter, it is possible to obtain the mass of the structure, m_e. Therefore, using the Eq. (8.17), we can interpret the situation.

$$M = m_{el} - m_e \tag{8.17}$$

Equation (8.17) is then verified using the sample value set. If the function is violated (i.e. M <0), the structure or structural element has 'failed', Melchers and Beck (2018). This procedure will be run *n* times in order to obtain a considerable set of values referring to *M*, so being possible to obtain its mean ($\bar{M}$) and standard deviation ($\widehat{M}$). With this whole set of data, it is possible to calculate the reliability index (β), a factor that measures the distance between the origin and the average value present in a normal distribution, following Eq. (8.18):

$$\beta = {\widehat{M}}/{\bar{M}} \tag{8.18}$$

Based on the reliability index, the probability of failure can be calculated using Eq. (8.19):

$$P = \Phi(-\beta) \tag{8.19}$$

where Φ is the standard normal cumulative function.

$$\Phi(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{\frac{-z^2}{2}} dz \tag{8.20}$$

However, there are often hardware limitations, so it is necessary to use other methods besides Monte Carlo, due to the fact that it has a high computational cost. Consequently, the classic method for calculating the probability of failure is also used, given Eq. (8.21).

$$P_f = \frac{\text{Non favorable cases}}{\text{Total cases}} \tag{8.21}$$

A hundred iterations were performed for the processes and the limit mass was based on the final mass obtained in the optimization process. Using the final value of the volume and the density of the material, it's possible to find the limit mass, being initially increased by 10% of the base value, as shown in Eq. (8.22).

$$m_{el} = (Vol * 7800) * (1 + 10\%) \tag{8.22}$$

```
%limit mass
porcent = input('Percentage for the limit mass ? ');
mel = (fval*7800)*(1+(porcent/100));
Minter = input('Amount of iterations ?  ');
me = ones(Minter,1);
tic;
for h=1:Minter
    csadm = norminv(rand(),260e6,26e6);
    reliability; %script to run the optimization
process
    me(h,1) = (fval*7800); %[kg]
end
time2=toc;
```

```
      %M
      M = mel - me;
      %average
      M_mean = mean(M);
      %standard deviation
      M_std = std(M);
      %beta
      beta = (M_mean)/(M_std);
      %standard normal cumulative function
      fun2 = @(z) exp(-(((z).^2)/2));
      int_fun2 = integral(fun2,-Inf,-beta);
      P_phi = (1/(sqrt(2*pi)))*int_fun2;
      T4 = table(beta,P_phi,time2,...

'VariableNames',{'Beta','FailureProbability','Runtime',},
...
          'RowNames',{'Results'});
      disp(T4)
      %Classic Method - Reliability
      cont = 0;
      for i = 1:Minter
          if(M(i)<0)
                cont = cont +1;
          end
      end
      falha2 = cont/Minter;
      probabilidadefalha2 = falha2*100;
      T5 = table(probabilidadefalha2,...
          'VariableNames',{'ProbabilidadeFalha2'},...
          'RowNames',{'Resultado (Confiabilidade2)'});
      disp (T5)
```

Table 8.5 - Cantilever plate with concentrated loading on the edge

Monte Carlo Method		
	x2	x3
Beta	2.6553	2.72879
Failure Probability	0.39614%	0.31871%
Classic Method		
	x2	x3
Failure Probability	1%	2%
Runtime	0.26546 hours	2.40272 hours

Table 8.66 – Simply supported plate with concentrated side-loading

Monte Carlo Method		
	x2	x3
Beta	2,5092	2,9796
Failure Probability	0,60507%	0,14430%
Classic Method		
	x2	x3
Failure Probability	2%	0%
Runtime	2,0745 hours	27,9 hours

Table 8.7 – Simply supported beam with uniformly distributed loading

Monte Carlo Method		
	x2	x3
Beta	2.844	2.674
Failure Probability	0.22273%	0.37477%
Classic Method		
	x2	x3
Failure Probability	0%	1%
Runtime	0.57631 hours	12.235 hours

Table 8.8 – Simply supported beam with concentrated centered loading

Monte Carlo Method		
	x2	x3
Beta	2.5157	2.6779
Failure Probability	0,59400%	0,37040%
Classic Method		
	x2	x3
Failure Probability	1%	1%
Runtime	0.48483 hours	13.885 hours